行政院國家科學委員會專題研究計畫　成果報告

無限狀態系統之自動驗證(3/3)

計畫類別：個別型計畫
計畫編號：NSC93-2213-E-002-003-
執行期間：93 年 08 月 01 日至 94 年 10 月 31 日
執行單位：國立臺灣大學電機工程學系暨研究所

計畫主持人：顏嗣鈞

計畫參與人員：林春成，李宜益，范家豪 ，張子璿 ，高浩仁

報告類型：完整報告
報告附件：出席國際會議研究心得報告及發表論文
處理方式：本計畫可公開查詢

中　華　民　國 95 年 1 月 30 日

無限狀態系統之自動驗證

2002.08.01 至 2005.10.31
主持人 顏嗣鈞 教授
國立臺灣大學電機工程學系

中文摘要:

在計算機科學的研究領域中,「計算理論」(Theory of Computation) 具有相當悠久的歷史。除了其本身深具困難度與挑戰性外，計算理論對於計算機科學的其他研究領域分支，亦深具影響力。嚴格說來，計算理論提供了這些其他相關研究(compiler、programming languages、pattern recognition、等等) 紮實的理論基礎以及分析工具。隨著時間的演進，計算理論的研究重點也有所不同，從七、八零年代較著重於純理論的研究(重點在automata theory、 formal languages、complexity theory、 等等)，演變到現今的潮流是將計算理論多年發展所累積的知識與技術，應用到real-world problems 的解決上。這包括利用「自動機理論」(automata theory)來協助電腦軟、硬體系統的驗證(verification)，利用「樹狀自動機」(tree automata)來提供XML 理論基礎與其相關問題的分析描述… 等等。

隨著電腦軟、硬體的複雜化，正規驗證與分析技術，近年來逐漸受到國際學術界與工業界的重視。未來的工業產品，若不能運用自動化的分析技術來協助提升品質，將很難保有競爭力。這個趨勢在工業界可以看到各種正規驗證技術的成功運用案例，以及各種結合傳統與正規技術的廣泛運用；而功能驗證的技術更是大型EDA公司（如Cadence與Synopsys）技術發展的重點。 傳統的驗證，侷限於「有限狀態系統」(finite-state systems)的驗證，但近幾年的研究重點，已漸漸導向各種「無限狀態系統」(infinite-state systems) 的驗證。

「薄膜計算」(membrane computing)為「分子計算」(molecular computing) 的一分支， 為1998 Gh. Paun 所提出的另一類「非傳統式計算模式」(nonstandard computational model)。 (其他非傳統式計算模式包括國科會目前所大力推動的「量子計算」(quantum computing))。 薄膜計算探討的計算模式，也是一類無限狀態系統。

在此三年期研究計畫中， 主持人以計算理論為基礎，探討多類無限狀態系統的驗證，獲得許多具體成果。在接下來的報告中，我們將對於研究成果逐一敘述。

1 Introduction

Automated verification has emerged as a fast growing research area from both theoretical and practical viewpoints in the computer science and engineering community. As computer software and hardware systems become more and more complicated, it is highly desirable that such systems be verified and analyzed automatically. Simply speaking, automated verification refers to the process of given a *system* S and a *property* P, verifying whether the behavior of system S satisfies property P algorithmically. Usually P and S are described by some *description languages*. Models known to be useful for defining S include *finite automata*, *timed automata*, *Petri nets*, among others, and *temporal logic* is a useful language to describe P. In contrast, the conventional method of *testing* (or *simulation*) analyzes a system's behaviors only with respect to a finite (and usually small) set of input samples. For complex hardware/software systems, however, testing is capable of only revealing a small portion of the possible behaviors of the system in a limited amount of time, which in turn, does not guarantee error-freedom even when the test reports no abnormalities. *Automated verification*, on the other hand, is to use rigorous mathematics and logical reasoning to verify whether a given system is completely free of errors under all circumstances. As one might expect, automated verification is again a computation-intensive process, which requires other than the so-called brute-force strategy in order for the method to be applicable to real-world applications. (The increasing complexity exponentially associated with the growth of a system size is known as the *state explosion problem*.) Over the past decade, a tremendous advance in the research of automated verification has resulted in a number of new and promising techniques capable of verifying systems up to 10^{20} states, and such techniques have been successfully applied to real hardware elements (such as CPU), and software protocols. It is expected that the trend (of applying automated verification to verifying real-world hardware/software systems) will continue, and more and more companies will adopt *automated verification* as a standard procedure in the development phase of their products.

Infinite-state systems are systems for which the number of system states is beyond any

fixed constant (i.e. infinite). The *infinity* could come from, for instance, *unbounded data structures* and/or *unbounded control structures* involved in the system. The former includes the models of *timed automata*, *push-down automata*, *communicating finite-state machines*, and *counter machines*, for instance, whereas *Petri nets* and *parameterized systems* are examples of the latter.

Built upon the knowledge regarding the verification of finite state systems accumulated in the past two decades, a great deal of research has been devoted to analyzing and verifying infinite-state systems (see, e.g., [11, 12]) in recent years. The shift from 'finite' to 'infinite-state' is also to reflect the nature of most systems in the real-world being 'infinite,' one way or another.

P systems in membrane computing are another model of infinite systems that have attracted much attention recently. There has been a flurry of research activities in the area of membrane computing (a branch of molecular computing) initiated five years ago by Gheorghe Paun [29]. Membrane computing identifies an unconventional computing model, namely a P system, from natural phenomena of cell evolutions and chemical reactions. Due to the built-in nature of maximal parallelism inherent in the model, P systems have a great potential for implementing massively concurrent systems in an efficient way that would allow us to solve currently intractable problems (in much the same way as the promise of quantum and DNA computing) once future bio-technology (or silicon-technology) gives way to a practical bio-realization (or chip-realization).

The Institute for Scientific Information (ISI) has recently selected membrane computing as a fast "Emerging Research Front" in Computer Science (*http://esi-topics.com/erf/october2003.html*). A P system is a computing model, which abstracts from the way the living cells process chemical compounds in their compartmental structure. Thus, regions defined by a membrane structure contain objects that evolve according to given rules. The objects can be described by symbols or by strings of symbols, in such a way that multisets of objects are placed in regions of the membrane structure. The membranes themselves are organized as a Venn diagram or a tree structure where one membrane

may contain other membranes. By using the rules in a nondeterministic, maximally parallel manner, transitions between the system configurations can be obtained. A sequence of transitions shows how the system is evolving. Various ways of controlling the transfer of objects from a region to another and applying the rules, as well as possibilities to dissolve, divide or create membranes have been studied. P systems were introduced with the goal to abstract a new computing model from the structure and the functioning of the living cell (as a branch of the general effort of Natural Computing – to explore new models, ideas, paradigms from the way nature computes). Membrane computing has been quite successful: many models have been introduced, most of them Turing complete and/or able to solve computationally intractable problems (NP-complete, PSPACE-complete) in a feasible time (polynomial), by trading space for time. (See the P system website at *http://psystems.disco.unimib.it* for a large collection of papers in the area, and in particular the monograph [30].)

Our investigation in the three years of this research project has led to the following publications:

1. H. Yen and L. Yu, Decidability Analysis of Self-Stabilization for Infinite State Systems, *Fundamenta Informaticae*, Vol. 70, No. 4, 387-402, 2006.

2. F. Wang and H. Yen, Reachability Solution Characterization of Parametric Real-time Systems, *Theoretical Computer Science*, Vol. 328 , pp. 187-201, 2004.

3. H. Yen, and L. Yu, Petri Nets with Simple Circuits, in the Proceedings of the *Ninth International Computing and Combinatorics Conference, (COCOON 2003)* (LNCS 2697), pp. 149-158, July 25-28, 2003, Big Sky, MT, USA.

4. H. Yen and L. Yu, Dependability Analysis of a Class of Probabilistic Petri Nets, in Proc. of *10th IEEE Pacific Rim Int'l Symp. on Dependable Computing, (PRDC 2004)* pp. 373-382, March 3-5, 2004, Tahiti, French Polynesia.

5. C. Chen, T. Lin, and H. Yen, Modelling and Analysis of Asynchronous Circuits and Timing Diagrams Using Parametric Timed Automata, in Proc. of the *23rd IASTED*

Int'l Conf. on Modelling, Identification and Control (MIC 2004), February 23-25, 2004 Grindelwald, Switzerland.

6. O. Ibarra, H. Yen, and Z. Dang, On Various Notions of Parallelism in P Systems, *International Journal of Foundations of Computer Science*, Vol. 16, No. 4, pp. 683-706, August 2005.

7. O. Ibarra, and H. Yen, On Deterministic Catalytic Systems, in Proc. of the *10th International Conference on Implementation and Application of Automata (CIAA 2005)*, (LNCS 3845), June 27V29, 2005, Sophia Antipolis, France.

8. C. Li, Z. Dang, O. Ibarra, and H. Yen, Signaling P Systems and Verification Problems, in *Proc. of the 32nd International Colloquium on Automata, Languages and Programming (ICALP'05)*, (LNCS 3580), pp. 1462-1473, July 11-15, 2005, Lisboa, Portugal.

9. O. Ibarra, S. Woodworth, H. Yen and Z. Dang, On Sequential and 1-Deterministic P Systems, in Proc. of the *11th International Computing and Combinatorics Conference (COCOON 2005)*, (LNCS 3595) , August 16 - 19, Kunming, Yunnan, China.

10. O. Ibarra, S. Woodworth H. Yen, and Z. Dang, On Symport/Antiport Systems and Semilinear Sets, in Proc. of the *6th International Workshop on Membrane Computing (WMC6) , (LNCS 3850)*, July 18 - 21, 2005, Vienna, Austria.

In what follows, we summarize what we have accomplished during the course of this project in greater detail.

2 Research results

• **Decidability Analysis of Self-Stabilization for Infinite State Systems** (by H. Yen and L. Yu) *Fundamenta Informaticae*, Vol. 70, No. 4, 387-402, 2006.

The notion of *self-stabilization* was introduced by Dijkstra [10] 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 [10], 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 [19] for a bibliography of self-stabilization). In a somewhat different direction of research, Gouda, Howell and Rosier ([17]) 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 [17]). To our knowledge, [17] also pioneers the introduction of the notion of self-stabilization to the model of Petri nets. In a subsequent paper [9], 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 [9], 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 complexity 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 [9]. 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 [9] by investigating, from the decidability viewpoint, the problem of deciding whether a given system is self-stabilizing for a wide range of infinite-state systems, including *lossy vector addition systems with states*, *one-counter machines*, *conflict-free Petri nets*, *lossy counter machines*, and *lossy channel systems*. As it turns out, the decidability of self-stabilization for *lossy vector addition systems with states*, *one-counter machines*, and *conflict-free Petri nets* can be established in a unified setting, taking advantage of the existence of a *periodic* witness for non-self-stabilizing infinite computations. For *lossy counter machines* and *lossy channel systems*, however, the self-stabilization problem turns out to be undecidable.

• **Reachability Solution Characterization of Parametric Real-time Systems** (by F. Wang and H. Yen) *Theoretical Computer Science*, Vol. 328 , pp. 187-201, 2004.

Timed automata have been a popular model in the research of formal description and verification of real-time systems [2]. In real-world applications, systems are usually described

with unknown parameters to be analyzed. Here we use the term *timing parameters* to refer to those parameters which are compared with clocks in either timed automata [3] or parametric TCTL formulae [36, 37, 38]. A timed automaton extended with unknown timing parameters is called a *timing parameter automaton (TPA)*. A *valuation* of unknown parameters making the goal state reachable in a TPA is called a *solution*. In this paper, we are mainly concerned with the following problem:

> The *reachability solution characterization (RSC)* problem: Given a TPA A and a goal predicate η, formulate a representation for the solution space of A with respect to η.

By 'formulating a representation' we mean finding a proper characterization for the solution space so as to allow queries arisen frequently in verification (such as emptiness, membership, etc) to be answered effectively.

In [3], it has been shown that the emptiness problem (i.e., the problem of deciding whether there exists a parameter valuation under which the associated timed language is nonempty) becomes undecidable when three or more clocks are compared with unknown parameters in TPAs. Knowing such a limitation, a line of subsequent research has been focused on the RSC problem for a number of restricted versions of TPAs. These positive results obtained in the last few years have all been focused on unknown timing parameters in the specification of logic formulae. But in practice, it is more likely that design engineers will use unknown parameters in the system behaviour descriptions. Moreover, design engineers will be more interested in knowing the condition for solution parameters valuations than in knowing whether there exists a solution parameter valuation. In this work, we identify three subclasses of TPAs and investigate the complexity issue of their RSC problems. The three subclasses are called *upper-bound TPAs*, *lower-bound TPAs*, and *bipartite TPAs*. Consider a TPA and w.l.o.g., we assume that only $\leq$ and $<$ are used in the predicates of the TPA. An *upper-bound parameter* θ is one that only appears to the right of an inequality operator (e.g., $x < \theta, x \leq \theta$), whereas a *lower-bound parameter* θ appears to the left of an inequality operator (e.g., $\theta < x, \theta \leq x$). *Upper-bound* (resp. *lower-bound*) TPAs are those whose unknown parameters are all *upper-bound* (resp. *lower-bound*) parameters. *Bipartite* TPAs

refer to those for which every unknown parameter is either a lower-bound parameter or an upper-bound parameter, but not both. Bipartite TPAs were considered in a recent article [20] in which the emptiness problem (undecidable for general TPA [3]) was shown to be decidable for such automata. In our setting, unknown parameters range over the set of natural numbers. As the work of [1] shows, unknown parameters of integer values can be used for modelling, for instance, the maximal number of retransmissions in the *Bounded Retransmission Protocol* (BRP), which is a data link protocol used by Philips.

Intuitively, what makes *upper-bound* (resp. *lower-bound*) TPAs easier to analyze, in comparison with their general counterparts, lies in the fact that for each of such TPAs, the solution space is *upward-closed* (resp. *downward-closed*). (A set S over k-dimensional vectors of natural numbers, for some k, is called *upward-closed* (resp., *downward-closed*) if $\forall u \in S, v \geq u \Longrightarrow v \in S$ (resp., $\forall u \in S, v \leq u \Longrightarrow v \in S$)). It is well known that an upward-closed set (resp., downward-closed set) is completely characterized by its *minimal* (resp., *maximal*) elements, which always form a finite set although the set might not be effectively computable in general. As we shall see later in this paper, we are able to give a complexity bound for the sizes of the minimal elements for a given upper-bound TPA. Our analysis is carried out in a way similar to a strategy proposed in [35] (by Valk and Jantzen), in which a sufficient and necessary condition was derived under which the set of minimal elements of an upward-closed set is guaranteed to be effectively computable. (Note, however, that [35] reveals no complexity bounds for the sizes of the minimal elements.) Taking advantage of certain properties offered by timed automata, we are able to refine Valk and Jantzen's approach to yield complexity bounds for the sizes of the minimal elements for the upward-closed sets associated with upper-bound TPAs, allowing us to characterize their solution spaces. This in turn answers the *RSC* problem for upper-bound TPAs. To a certain extent, our result supplements the work of [20] (in which the emptiness problem was shown to be decidable for bipartite TPAs) by tackling a more general problem. We are also able to extend our analysis to the model of upper-bound *timing parameter vector addition systems with states (TPVASSs)*, each of which can be viewed as a TPA equipped

with counters without zero-test capabilities. Once the sizes of minimal elements become available, finding all such elements can be done by exhaustive search using the region graph technique, although it would clearly be interesting to develop smarter (and more efficient) algorithms. Some complexity results are also derived for lower-bound TPAs. For bipartite TPAs, we are able to show that their solution spaces are not semilinear in general, in spite of the fact that the emptiness problem is decidable [20].

We feel that the method developed in this paper for analyzing upward-closed sets is interesting in its own right. Our strategy provides a refinement over the approach proposed in [35] in the sense that the sizes of the minimal elements can now be deduced, provided that certain conditions are met. It would be interesting to seek additional applications of our technique.

• **Petri Nets with Simple Circuits** (by H. Yen, and L. Yu) in the Proceedings of the *Ninth International Computing and Combinatorics Conference, (COCOON 2003)* (LNCS 2697), pp. 149-158, July 25-28, 2003, Big Sky, MT, USA.

Petri nets (*PNs*, for short) have been a popular model for reasoning about the behaviors of concurrent systems [32]. The *reachability* problem is among the most important problems in the study of PNs. Reachability analysis is key to the solutions of such PN problems as *liveness*, *fairness*, *controllability*, *model checking* and more. In addition, identifying a tight complexity bound for the reachability problem of PNs remains a great challenge in the community of theoretical computer science. Although known to be decidable, the existing algorithms for the problem is still not even primitive recursive [26] (see also [23]), while the problem is also known to be exponential space hard [24].

Integer linear programming (*ILP*, for short) has long been a useful tool for the reachability analysis of PNs. It is well known that in a PN $\mathcal{P}$ with initial marking μ_0, a marking μ is reachable from μ_0 *only if* there exists a column vector x such that the *state equation* $\mu_0 + A \cdot x = \mu$ holds, where A is the *addition matrix* of $\mathcal{P}$. Although the converse does not

necessarily hold, there are restricted classes of PNs for which the state equation is sufficient and necessary to capture reachability. Most notable is the class of *circuit-free* PNs (i.e., Petri nets without circuits) as well as the class of PNs without *token-free* circuits in every reachable marking [39]. (A *circuit* of a Petri net is simply a closed path (i.e., a cycle) in the Petri net graph.) Other subclasses for which reachability has been thoroughly studied and solved are *conflict-free*, *normal* [21, 39], *sinkless* [21, 39], *BPP-net* [42], *trap-circuit* [22, 41], and *extended trap-circuit* PN [41] to name a few. For each of them, deciding reachability can be equated with solving an ILP problem. A question arises: Can we enlarge the PN class while still retaining the desirable property of reachability being characterizable by ILP? Affirmative answer to this question is one of the contributions of this paper.

Circuits in BPP-nets are referred to as $\oplus$-circuits (a simple type of circuit where every transition having exactly one input place, and the firing of transition removing exactly one token from it's sole input place). In normal PNs, no transition is capable of decreasing the token count of a minimal circuit, and such circuits are called $\odot$-circuits. (A circuit is *minimal* iff the set of places in it does not properly include that in any other circuit.) Our new PN class, called *simple-circuit Petri nets* (*sc-PNs*, for short), consists of those in which each minimal circuit is either a $\odot$-circuit, or a $\oplus$-circuit which is not properly included in any non-$\oplus$-circuit. By relaxing the constraints on circuits, our sc-PNs properly contain that of conflict-free, normal, trap-circuit, extended trap-circuit, and BPP-nets.

To analyze sc-PNs, the technique of the so-called *decomposition approach* is used. Given a computation σ of a PN, the basic idea is to rearrange σ into some canonical form $\sigma_1\sigma_2\cdots\sigma_n$ with each of them being of some 'simpler form'. By a 'simpler form' we mean the sub-PN induced by each of the segments has its reachability set characterizable by certain well-understood and easily solvable formulations, such as ILP. For cases, we can also place a bound on the number of segments in the above canonical computation. Demonstrating the applicability of the decomposition approach to sc-PNs is another contribution of our work. It is worth noting that our analysis yields an ILP formulation for the reachability problem in which the initial and final markings are regarded as *parameters*, as opposed to being

constants as in many of the traditional reachability analysis of PNs. The complexity of model checking with respect to a number of temporal logics is also investigated.

It is known that the class of marked graphs, in spite of its simplicity, has found important applications in various areas, including specification and verification of asynchronous circuits, supervisory control, among others. BPP-nets are closely related to the model of Basic Parallel Processes, which is an important branch of Process Algebra. As for normal Petri nets, a recent article shows an interesting application in AI planning. Although our study of simple-circuit Petri nets in this paper is primarily from a theoretical viewpoint, properly containing several well-known classes (such as marked graphs, BPP-nets, and normal Petri nets) makes this new class of Petri nets appealing as far as enhancing the expressive power while keeping the analytical complexity manageable is concerned. Seeking additional applications of this new Petri net class is among one of our future research topics.

• **Dependability Analysis of a Class of Probabilistic Petri Nets** (by H. Yen and L. Yu) in Proc. of *10th IEEE Pacific Rim Int'l Symp. on Dependable Computing, (PRDC 2004)* pp. 373-382, March 3-5, 2004, Tahiti, French Polynesia.

As modern hardware and software systems are becoming more complex and at the same time required to be more dependable, there is an ever-increasing need for new evaluation techniques; the advantage of analytical evaluation over experimental one lies in its usefulness in abstracting the essentials of systems (so that various levels of system details can be abstracted out) and analyzing or predicting system behaviors (especially while a system is being designed or implemented), and its being generally far more cost effective than its experiment-based counterpart. With the increasing interest in developing dependable systems, the study of problems regarding *termination*, *controllability*, and *self-stabilization*, among others, has also been gaining increasing popularity in the computer science community due to the following reasons. Dependable systems are often associated with properties like *safety* ('something bad never happens'), *liveness* ('something good eventually happens'), *fault-tolerance* (error detection, recovery and masking), etc. The safety property requires that undesired or failure

states be avoided at all times during the course of a computation. The liveness property asserts that a certain desired condition be true eventually. The notion of self-stabilization was introduced by Dijkstra [10] to describe a system having the behavior that regardless of its starting configuration, the computation is guaranteed to return to a *legitimate configuration* eventually. By a legitimate configuration we mean a configuration reachable from its initial configuration. Since a self-stabilizing system has the ability to 'correct' itself even in the presence of certain unpredictable errors leading itself to an illegitimate configuration, one can assert that a self-stabilizing system is, in a sense, fault-tolerant.

In view of the above, it becomes apparent that automatic verification of various properties associated with concurrent/distributed systems is critical in the process of designing and analyzing dependable systems. While techniques for the automatic verification of *finite-state systems* are relatively well studied, one of the main challenges in the domain of verification is concerned with the development of new techniques capable of coping with problems beyond the finite state framework.

The aim of this paper focuses on investigating the following problem. Given an infinite-state system, determine whether the system meets certain criteria (termination, controllability, and self-stabilization) frequently required in dependable computing environments. Taking into consideration that many real-world systems are nondeterministic (or stochastic, to be more precise) in nature, the system model under our investigation is not only infinite-state but also *probabilistic*, allowing us to ask questions such as 'something happens with probability 1', for instance. The systems under investigation are modelled as *Petri nets*, which have been regarded as one of the most successful models for describing the behaviors of systems of concurrent nature. In spite of their popularity, the high expressive power of Petri nets renders most of the nontrivial problems for this model highly intractable or even unsolvable. As a result, it is of interest from the theoretical and practical viewpoints to investigate problems with respect to restricted (either structurally or behaviorally) versions of Petri nets, in hope of making simpler solutions feasible and well as gaining more insights into the factors that make general Petri nets difficult to analyze.

A Petri net is *conflict-free* if every place which is an input of more than one transition is on a self-loop with each such transition [25]; therefore, once a transition becomes enabled, the only way to disable it is to fire the transition itself.

Probabilistic techniques, capable of modelling unreliable or unpredictable behaviors of systems, are extensively used in the analysis of the performance and dependability of hardware and software systems. In this paper, we consider a *probabilistic* version of conflict-free Petri nets, in which each marking (i.e., configuration) is associated with a *transition probability function* characterizing the firing of each enabled transition. We investigate through a technique recently developed in [40] (called the *valuation method*) a number of important dependability-related problems, including *termination with probability 1*, *self-stabilization with probability 1*, and *controllability with probability 1*, etc. The idea of the valuation-based approach for Petri nets is to associate a valuation in $\{0, 1, 2, ...\infty\}$ with each marking, and if the set of markings of zero valuation is *forward-closed*, then the valuation along any computation is non-increasing, and in many cases, has the tendency to move towards the ground level (i.e., valuation zero) of which the marking sometimes constitute the set of states of interest, e.g., the termination set.

The main contribution of this paper lies in the development of a unified approach (extending of the work of [40]) for reasoning about various dependability-related problems for probabilistic conflict-free Petri nets. In addition to the results themselves, we feel that the valuation-based approach for the analysis of probabilistic conflict-free Petri net is also interesting in its own right, and may have other applications to the analysis of other probabilistic Petri net models.

• **Modelling and Analysis of Asynchronous Circuits and Timing Diagrams Using Parametric Timed Automata** (by C. Chen, T. Lin, and H. Yen) in Proc. of the *23rd IASTED Int'l Conf. on Modelling, Identification and Control (MIC 2004)*, February 23-25, 2004 Grindelwald, Switzerland.

When designers do circuit design, they usually use some informal notations or diagrams

such as circuit diagrams, ASM charts, and timing diagrams in practice. To take advantage of existing verification tools to verify the design, one has to transform these informal diagrams into some formal models, for which formal verification techniques (such as model checking) are available. In asynchronous circuit design, the intended behaviors of asynchronous circuits are commonly specified using *signal transition graphs* (STG), which are essentially Petri nets whose transitions are labelled by events corresponding to rising and falling of signals. A STG could be transformed into an equivalent automaton (state graph). Once a state graph is given, it is easy to draw the corresponding Karnaugh map and derive the next-state function. In the process of doing so, decomposition is involved which may lead to hazards; as a result, when deriving the decomposed functions, we must try to avoid the occurrence of hazards, which is a tedious procedure. For an asynchronous circuit in which the delays of all the constituent components are known, the behaviors of such a circuit can be modelled using a timed automaton, and consequently the correctness of the circuit (such as whether the design is hazard-free) can be deducted by, for instance, model-checking the timed automaton. In practice, however, one might be more interested in the following question: '*Under what delay constraints will the circuit still behave correctly?*' Clearly it is not feasible to try all the possibilities of delay constraints since the number of such constraints to be tested is in general infinite. To overcome this difficulty, we propose the use of *parametric timed automata* (PTA) which are basically timed automata augmented by unknown parameters. And the question boils down to finding some (or all) valuations of the unknown parameters that ensure correctness of the modelled circuit, if the solution space is not empty. In theory, such a problem for PTAs is undecidable in general; however, there are certain subclasses of PTAs for which their solution spaces are simpler, and many analytical problems are indeed decidable.

In our setting, the lower and upper bounds of delays in asynchronous circuits can be parameters. We model the delays of each gate using parametric timed automata. The input signals can also be modelled as timed automata to express the timing bounds on their frequency or some other protocols they assumed to follow. We can simulate all the possible

behaviors of the circuits including all choice of delays by combining these timed automata. An STG represents a protocol of interaction between a component and its environment. The produced circuits must follow this protocol. The STG's specification could be transformed into an automaton with error transitions. By adding error transitions for all output events, one can be ensured that if error transitions enable, the computation will not go to a legal state. So the verification problem is: *With what delay bounds the set of behaviors of the circuits will contain a behavior not included in the semantics of the STG?* In other words, the problem is to decide under what delay bounds is the error state reachable in the composition of all the above mentioned automata. To answer this question, we apply the forward reachability algorithm for parametric timed automata. The reachability problem of parametric timed automata is undecidable but the same problem is decidable in L/U automata, a subclass of parametric time automata. The bi-bounded delay model under our consideration is just an L/U automaton because the lower bounds of delays always occur negatively; on the contrary, the upper bounds of delays occur positively. There are no timing parameters other than the lower bounds and upper bounds and it makes our delay model surely a L/U automaton. As a result, the reachability problem of our delay model is decidable.

The second informal model considered in this work is the model of the so-called *timing diagrams*, which are convenient but informal diagrams for hardware and protocol design. K. Fisler [13] and N. Amla [4] both have introduced the approaches for model checking timing diagrams; Fisler [13] gave a formal definition of timing diagrams in her dissertation. Timing diagrams depict the temporal behaviors between signals including the patterns of value changes on signals, precedence relationships with timing constraints, and synchronization. Fisler also defined two kinds of semantics of timing diagrams for verification. Invariant semantics is suited to formal verification and the timing diagram patterns are required to hold in all times. Iterative semantics represents cycles of behaviors and the entire timing diagram is viewed as behaviors that must be guaranteed repeatedly. For efficient model checking, Amla et al. [4] introduced a class of timing diagram, regular timing diagram (RTD). Amla

et al. defined the semantics of regular timing diagram in a reformulation of the iterative semantics. They permit timing diagrams to be satisfied in an overlapping manner. For simplicity, we do not allow unspecified signal values.

Fisler [13] has proved that containment of regular languages in non-regular timing diagram languages is decidable. Although this result is helpful to check whether the regular language generated by a system is contained by the specification expressed in non-regular timing diagram; the containment testing lies in PSPACE. Amla et al. [4] defined a subclass of timing diagram, regular timing diagram, and introduced an efficient decompositional model checking method for it. They used $\forall$-automata ("dual-run" automata) to provide an efficient emptiness checking method which the time complexity is a small polynomial in the sizes of RTD and the space complexity is logarithmic in the sizes of both the system and RTD. But because of the non-decidability, they all did not consider the timing diagram with parameters in timing constraints which could be useful for designers in practice.

To extend the ability of timing diagrams, we allow parameters to occur in the timing bounds on timing diagrams. We transform RTD into PTA and perform the reachability algorithm; moreover, after the transformation of the specification represented by RTD, we perform model checking methods on systems represented by automata to check the reasonability of the specification. This gives us an approach to tune our specification and make it more reasonable.

• **On Various Notions of Parallelism in P Systems** (by O. Ibarra, H. Yen, and Z. Dang) *International Journal of Foundations of Computer Science*, Vol. 16, No. 4, pp. 683-706, August 2005.

In the standard semantics of P systems [29, 30], each evolution step of a system G is a result of applying all the rules in G in a maximally parallel manner. More precisely, starting from the initial configuration, w, the system goes through a sequence of configurations, where each configuration is derived from the directly preceding configuration in one step by the

application of a multi-set of rules, which are chosen nondeterministically. For example, a catalytic rule $Ca \rightarrow Cv$ in membrane q is applicable if there is a catalyst C and an object (symbol) a in the preceding configuration in membrane q. The result of applying this rule is the evolution of v from a. If there is another occurrence of C and another occurrence of a, then the same rule or another rule with Ca on the left hand side can be applied. Thus, in general, the number of times a particular rule is applied at anyone step can be unbounded. We require that the application of the rules is maximal: all objects, from all membranes, which *can be* the subject of local evolution rules *have to* evolve simultaneously. Configuration z is reachable (from the starting configuration) if it appears in some execution sequence; z is halting if no rule is applicable on z.

In this paper, we study a different definition of maximal parallelism. Let G be a P system and $R = \{r_1, ..., r_k\}$ be the set of (distinct) rules in all the membranes. (Note that r_i uniquely specifies the membrane the rule belongs to.) We say that G operates in maximal parallel mode if at each step of the computation, a maximal subset of R is applied, and at most one instance of any rule is used at every step (thus at most k rules are applicable at any step). For example, if r_i is a catalytic rule $Ca \rightarrow Cv$ in membrane q and the current configuration has two C's and three a's in membrane q, then only one a can evolve into v. Of course, if there is another rule r_j, $Ca \rightarrow Cv'$, in membrane q, then the other a also evolves into v'. Throughout the paper, we will use this definition of maximal parallelism. Here, we look at the computing power of P systems under three semantics of parallelism. For a positive integer $n \leq k$, define:

n-Max-Parallel: At each step, nondeterministically select a maximal subset of at most n rules in R to apply (this implies that no larger subset is applicable).

$\leq n$-Parallel: At each step, nondeterministically select any subset of at most n rules in R to apply.

n-Parallel: At each step, nondeterministically select any subset of exactly n rules in R to apply.

In all three cases, if any rule in the subset selected is not applicable, then the whole subset

is not applicable. When $n = 1$, the three semantics reduce to the **Sequential** mode.

• **On Deterministic Catalytic Systems** (by O. Ibarra, and H. Yen) in Proc. of the *10th International Conference on Implementation and Application of Automata (CIAA 2005)*, (LNCS 3845), June 27V29, 2005, Sophia Antipolis, France.

There has been a great deal of research activities in the area of membrane computing (a branch of natural computing) initiated by Gheorghe Păun in a seminal paper [28] over six years ago (see also [29]). Membrane computing identifies an unconventional computing model, namely a P system, from natural phenomena of cell evolutions and chemical reactions. Due to the built-in nature of maximal parallelism inherent in the model, P systems have a great potential for implementing massively parallel systems in an efficient way that would allow us to solve currently intractable problems once future bio-technology (or silicon-technology) gives way to a practical bio-realization (or chip-realization).

A P system is a computing model, which abstracts from the way the living cells process chemical compounds in their compartmental structure. Thus, regions defined by a membrane structure contain objects that evolve according to given rules. The objects can be described by symbols or by strings of symbols, in such a way that multisets of objects are placed in regions of the membrane structure. The membranes themselves are organized as a tree structure (this can be represented by a Venn diagram) where one membrane may contain other membranes. By using the rules in a nondeterministic, maximally parallel manner, transitions between the system configurations can be obtained. A sequence of transitions shows how the system is evolving. Various ways of controlling the transfer of objects from a region to another and applying the rules, as well as possibilities to dissolve, divide or create membranes have been studied. P systems were introduced with the goal to abstract a new computing model from the structure and the functioning of the living cell (as a branch of the general effort of Natural Computing – to explore new models, ideas, paradigms from the way nature computes). Membrane computing has been quite successful: many models have been introduced, most of them Turing complete and/or able to solve computationally

intractable problems (NP-complete, PSPACE-complete) in a feasible time (polynomial), by trading space for time. (See the P system website at http://psystems.disco.unimib/it for a large collection of papers in the area, and in particular the monograph [30].)

In the standard semantics of P systems [29, 30], each evolution step of a system P is a result of applying all the rules in P in a maximally parallel manner. More precisely, starting from the initial configuration, w, the system goes through a sequence of configurations, where each configuration is derived from the directly preceding configuration in one step by the application of a multiset of rules, which are chosen nondeterministically. For example, a catalytic rule $Ca \rightarrow Cv$ in membrane m is applicable if there is a catalyst C and an object (symbol) a in the preceding configuration in membrane m. The result of applying this rule is the evolution of v from a. If there is another occurrence of C and another occurrence of a, then the same rule or another rule with Ca on the left hand side can be applied. Thus, in general, the number of times a particular rule is applied at anyone step can be unbounded. We require that the application of the rules is maximal: all objects, from all membranes, which *can be* the subject of local evolution rules *have to* evolve simultaneously. Configuration z is reachable (from the starting configuration) if it appears in some execution sequence; z is halting if no rule is applicable on z.

Two popular models of P systems are the catalytic system [29] and the symport/antiport system [31]. An interesting subclass of the latter was studied in [14] – each system is *deterministic* in the sense that the computation path of the system is unique, i.e., at each step of the computation, the maximal multiset of rules that is applicable is unique. It was shown in [14] that any recursively enumerable unary language $L \subseteq o^*$ can be accepted by a deterministic 1-membrane symport/antiport system. Thus, for symport/antiport systems, the deterministic and nondeterministic versions are equivalent and they are universal. It also follows from the construction in [33] that for another model of P systems, called communicating P systems, the deterministic and nondeterministic versions are equivalent as both can accept any unary recursively enumerable language. However, the deterministic-versus-nondeterministic question was left open in [14] for the class of catalytic systems (these

systems have rules of the form $Ca \to Cv$ or $a \to v$), where the proofs of universality involve a high degree of parallelism [33, 15]. We answer this question negatively in this paper. Since nondeterministic catalytic systems are universal, our result also gives the first example of a variant of P systems for which the nondeterministic version is universal, but the deterministic version is not.

For a catalytic system serving as a *language acceptor*, the system starts with an initial configuration wz, where w is a fixed string of catalysts and noncatalysts not containing any symbol in z, and $z = a_1^{n_1}...a_k^{n_k}$ for some nonnegative integers $n_1, ..., n_k$, with $\{a_1, ..., a_k\}$ a distinguished subset of noncatalyst symbols (the input alphabet). At each step, a maximal multiset of rules are nondeterministically selected and applied in parallel to the current configuration to derive the next configuration (note that the next configuration is not unique, in general). The string z is accepted if the system eventually halts. Unlike nondeterministic 1-membrane catalytic system acceptors (with 2 catalysts) which are universal, we are able to show using a graph-theoretic approach that the Parikh map of the language ($\subseteq a_1^*...a_k^*$) accepted by any deterministic catalytic system is a simple semilinear set which can also be effectively constructed. For deterministic 1-membrane catalytic systems using only rules of type $Ca \to Cv$, we show the set of reachable configurations from a given initial configuration to be effective semilinear. In contrast, the reachability set is no longer semilinear in general if rules of type $a \to v$ are also used. Our result generalizes to multi-membrane catalytic systems.

We also consider deterministic catalytic systems which allow rules to be prioritized. Three such systems, namely, *statically prioritized*, *strongly prioritized* and *weakly prioritized* catalytic systems, are investigated. For statically prioritized systems, rules are divided into different priority groups, and if a rule in a higher priority group is applicable, then no rules from a lower priority group can be used. For both strongly prioritized and weakly prioritized systems, the underlying priority relation is a *strict partial order* (i.e., irreflexive, asymmetric, and transitive). Under the semantics of strong priority, if a rule with higher priority is used, then no rule of a lower priority can be used even if the two rules do not compete for objects.

This notion of strong priority coincides with the semantics of the priority relation used in [29]. For weakly prioritized systems, a rule is applicable if it cannot be replaced by a higher priority one. For these three prioritized systems, we obtain contrasting results by showing that deterministic strongly and weakly prioritized catalytic systems are universal, whereas statically prioritized systems only accept semilinear sets.

• **Signaling P Systems and Verification Problems** (by C. Li, Z. Dang, O. Ibarra, and H. Yen) in *Proc. of the 32nd International Colloquium on Automata, Languages and Programming (ICALP'05)*, (LNCS 3580), pp. 1462-1473, July 11-15, 2005, Lisboa, Portugal.

P systems [29, 30] are abstracted from the way the living cells process chemical compounds in their compartmental structure. A P system consists of a finite number of membranes, each of which contains a multiset of objects (symbols). The membranes are organized as a Venn diagram or a tree structure where a membrane may contain other membranes. The dynamics of the P system is governed by a set of rules associated with each membrane. Each rule specifies how objects evolve and move into neighboring membranes. In particular, a key feature of the model of P systems is that rules are applied in a nondeterministic and maximally parallel manner. Despite the short (only five years) history of membrane computing, there has already been a notably large collection of papers in the area (see the P systems website: psystems.disco.unimb/it) and membrane computing has been selected as a fast "Emerging Research Front" in Computer Science by the Institute for Scientific Information (ISI) (esi-topics.com/erf/october2003.html). Due to the key feature inherent in the model, P systems have a great potential for implementing massively concurrent systems in an efficient way that would allow us to solve currently intractable problems (in much the same way as the promise of quantum and DNA computing). It turns out that P systems are a powerful model: even with only one membrane (i.e., 1-region P systems) and without priority rules, P systems are already universal [29, 34]. In such a one-membrane P system, rules are in the form of $u \rightarrow v$, which, in a maximally parallel manner, replaces multiset u (in current configuration which is a multiset of symbol objects) with multiset v.

Signals are a key to initiate biochemical reactions between and inside living cells. Many examples can be found in a standard cell biology textbook [6]. For instance, in signal transduction, it is known that guanine-nucleotide binding proteins (G proteins) play a key role. A large heterotrimeric G protein, one of the two classes of G proteins, is a complex consisting of three subunits: G_α, G_β, and G_γ. When a ligand binds to a G protein-linked receptor, it serves as a signal to activate the G protein. More precisely, the GDP, a guanine nucleotide, bound to the G_α subunit in the unactivated G protein is now displaced with GTP. In particular, the G protein becomes activated by being dissociated into a G_β-G_γ complex and a G_α-GTP complex. Again, the latter complex also serves as a signal by binding itself to the enzyme adenylyl cyclase. With this signal, the enzyme becomes active and converts ATP to cyclic AMP. As another example, apoptosis (i.e., suicide committed by cells, which is different from necrosis, which is the result from injury) is also controlled by death signals such as a CD95/Fas ligand. The signal activates caspase-8 that initiates the apoptosis. Within the scope of Natural Computing (which explores new models, ideas, paradigms from the way nature computes), motivated by these biological facts, it is a natural idea to study P systems, a molecular computing model, augmented with a signaling mechanism.

In this paper, we investigate one-membrane signaling P systems (signaling systems in short) where the rules are further equipped with signals. More precisely, in a signaling system M, we have two types of symbols: object symbols and signals. Each configuration is a pair consisting of a set S of signals and a multiset α of objects. Each rule in M is in the form of $s, u \to v, s'$ or $s, u \to \Lambda$, where s, s' are signals and u, v are multisets of objects. The rule is enabled in the current configuration (S, α) if s is present in the signal set S and u is a sub-multiset of the multiset α. All the rules are fired in maximally parallel manner. In particular, in the configuration as a result of the maximally parallel move, the new signal set is formed by collecting the set of signals s' that are emitted from all the rules actually fired during the move (and every signal in the old signal set disappears). Hence, a signal may trigger an unbounded number of rule instances in a maximally parallel move.

We focus on verification problems of signaling systems; i.e., algorithmic solutions to a

verification query on whether a given signaling system does satisfy some desired behavioral property. Such solutions not only help us understand the power of the maximally parallelism that is pervasive in P systems but also would provide a way to validate a (signaling) P system in vitro through digital computers when the P system is intended to simulate living cells. However, since one-membrane P systems are Turing-complete, so are signaling systems. Therefore, to study the verification problems, we have to look at restricted signaling systems. A signaling system is non-cooperative if each rule is in the form of $s, a \to \Lambda$ or in the form of $s, a \to bc, s'$, where a, b, c are object symbols. All the results can be generalized to non-cooperative signaling systems augmented with rules $s, a \to v, s'$. We study various reachability queries for non-cooperative signaling systems M; i.e., given two formulas $Init$ and *Goal* that define two sets of configurations, are there configurations C_{init} in $Init$ and C_{goal} in *Goal* such that C_{init} can reach C_{goal} in zero or more maximally parallel moves in M? We show that, when $Init$ is a Presburger formula (roughly, in which one can compare integer linear constraints over multiplicities of symbols against constants) and *Goal* is a region formula (roughly, in which one can compare multiplicities of symbols against constants), the reachability query is decidable. Notice that, in this case, common reachability queries like halting and configuration reachability are expressible. We also show that introducing signals into P systems indeed increases its computing power; e.g., non-cooperative signaling systems are strictly stronger than non-cooperative P systems (without signals). On the other hand, when *Goal* is a Presburger formula, the query becomes undecidable. Our results generalize to queries expressible in a subclass of a CTL temporal logic and to non-cooperative signaling systems with rules $S, a \to v, S'$ (i.e., the rule is triggered with a set of signals in S). We also study the case when a signal has bounded strength and, in this case, non-cooperative signaling systems become universal.

Non-cooperative signaling systems are also interesting for theoretical investigation, since the signaling rules are context-sensitive and the systems are still nonuniversal as we show. In contrast to this, rules $a \to v$ in a non-cooperative P system are essentially context-free. It is difficult to identify a form of restricted context-sensitive rules that are still nonuniversal. For

instance, a communicating P system (CPS) with only one membrane [33] is already universal, where rules are in the form of $ab \rightarrow a_x b_y$ or $ab \rightarrow a_x b_y c_{come}$ in which a, b, c are objects, x, y (which indicate the directions of movements of a and b) can only be *here* or *out*. Also one membrane catalytic systems with rules like $Ca \rightarrow Cv$ (where C is a catalytic) are also universal. More examples including non-cooperative signaling systems with promoters, which will be discussed further in this section, are also universal. Our non-cooperative signaling systems use rules in the form of $s, a \rightarrow v, s'$, which are in a form of context-sensitive rules, since the signals constitute part of the triggering condition as well as the outcome of the rules.

At the heart of our decidability proof, we use a form of upper-closed sets to serve as a symbolic representation for configuration sets and prove that the symbolic representation is invariant under the backward reachability relation of a non-cooperative signaling system. From the studies in symbolic model-checking [8] for classic transition systems, our symbolic representation also demonstrates a symbolic model-checking procedure at least for reachability. In our undecidability proof, we use the well-known result on the Hilbert's Tenth Problem: any r.e. set (of integer tuples) is also Diophantine. We note that, for P systems that deal with symbol objects, proofs for universality almost always use the theoretical tool through matrix grammar with appearance checking [27] or through Minsky's two-counter machines. Here, we employ a new tool using Diophantine equations, which facilitates an elegant proof of the universality result. With multiplication being easily implemented under maximal parallelism, we feel that our new technique is of interest in its own right and might find additional applications in P systems.

Signaling mechanisms have also been noticed earlier in P system studies. For instance, in a one-membrane P system with promoters [7], a rule is in the form of $u \rightarrow v|p$ where p is a multiset called a promoter. The rule fires as usual in a maximally parallel manner but only when objects in the promoter all appear in the current configuration. Notice that, since p may not be even contained in u, a promoter, just as a signal, may trigger an unbounded number of rule instances. Indeed, one can show that a signaling system can be directly

simulated by a one-membrane P system with promoters. However, since one-membrane non-cooperative P systems with promoters are known to be universal [7], our decidability results on non-cooperative signaling systems have a nice implication: our signals are strictly weaker than promoters (and hence have more decidable properties). The decidability results also imply that, as shown in the paper, non-cooperative signaling systems and vector addition systems (i.e., Petri nets) have incomparable computing power, though both models have a decidable configuration-to-configuration reachability. This latter implication indicates that the maximal parallelism in P systems and the "true concurrency" in Petri nets are different parallel mechanisms. Other signaling mechanisms such as in [5] are also promoter-based.

• **On Sequential and 1-Deterministic P Systems** (by O. Ibarra, S. Woodworth, H. Yen and Z. Dang) in Proc. of the *11th International Computing and Combinatorics Conference (COCOON 2005)*, (LNCS 3595) , August 16 - 19, Kunming, Yunnan, China.

Initiated five years ago by Gheorghe Paun [29] as a branch of molecular computing, *membrane computing* identifies an unconventional computing model, namely a P system, from natural phenomena of cell evolutions and chemical reactions. A P system abstracts from the way the living cells process chemical compounds in their compartmental structure. Thus, regions defined by a membrane structure contain objects that evolve according to given rules. The objects can be described by symbols or by strings of symbols, in such a way that multisets of objects are placed in regions of the membrane structure. The membranes themselves are organized as a Venn diagram or a tree structure where one membrane may contain other membranes. By using the rules in a nondeterministic, maximally parallel manner, transitions between the system configurations can be obtained. A sequence of transitions shows how the system is evolving. Various ways of controlling the transfer of objects from a region to another and applying the rules, as well as possibilities to dissolve, divide or create membranes have been studied.

Membrane computing has been quite successful: many models have been introduced, most of them Turing complete and/or able to solve computationally intractable problems

(NP-complete, PSPACE-complete) in a feasible time (polynomial), by trading space for time. (See the P system website at *http://psystems.disco.unimib.it* for a large collection of papers in the area, and in particular the monograph [30].) Due to the built-in nature of maximal parallelism inherent in the model, P systems have a great potential for implementing massively concurrent systems in an efficient way that would allow us to solve currently intractable problems (in much the same way as the promise of quantum and DNA computing) once future bio-technology (or silicon-technology) gives way to a practical bio-realization (or chip-realization). In fact, the Institute for Scientific Information (ISI) has recently selected membrane computing as a fast "Emerging Research Front" in Computer Science (*http://esi-topics.com/ erf/october2003.html*).

In the standard definition of a P system, the computation is carried out in a maximally parallel and nondeterministic manner [29, 30]. However, an interesting class of P systems with symport/antiport rules was studied in [14] where each system is *deterministic* in the sense that the computation path of the system is unique; i.e., at each step of the computation, the maximal multiset of rules that is applicable is unique. It was shown in [14] that any recursively enumerable unary language $L \subseteq o^*$ can be accepted by a deterministic 1-membrane symport/antiport system. Thus, for symport/antiport systems, the deterministic and nondeterministic versions are equivalent.

The construction of the deterministic system in [14] is such that the size of the maximal multiset of rules that is applicable at every step of the computation is either 1 or 2. We refer to this system as 2-deterministic. In general, a k-deterministic system is one in which the maximal multiset of rules applicable at each step is at most k. An interesting case is when $k = 1$, i.e., the system is 1-deterministic.

A concept, which is more general than 1-deterministic, is that of sequential mode of computation in P systems; i.e., at every step, only one nondeterministically chosen rule instance is applied. Clearly, when a P system is 1-deterministic, then the system (which, by definition, is still maximally parallel) can be treated as a sequential system. So if a class of systems is nonuniversal under the sequential mode, then any 1-deterministic such

system in the class is also nonuniversal. Sequential P systems (also called asynchronous P systems) have been studied in various places in the literature. Here, we present results that complement these earlier results. In particular, we show the following:

1. Any sequential P system with cooperative rules (i.e., rules of the form $u \rightarrow v$, where u, v are strings of symbols) with rules for membrane creation and membrane dissolution can be simulated by a vector addition system (VAS), provided the rules are not prioritized and the number of membranes that can be created during the computation is bounded by some fixed positive integer. Hence the reachability problem (deciding if a configuration is reachable from the start configuration) is decidable. It follows that 1-deterministic such systems have a decidable reachability problem. Interestingly and somewhat surprisingly, if such cooperative systems are allowed to create an unbounded number of new membranes during the course of the computation, then they become universal.

2. A sequential communicating P system language acceptor (CPA) is equivalent to a partially blind multicounter machine (PBCM) [18]. Several interesting corollaries follow from this equivalence, for example:

   (a) The emptiness problem for CPAs is decidable.

   (b) The class of CPA languages is a proper subclass of the recursive languages.

   (c) The language $\{a^n b^n \mid n \geq 1\}^*$ cannot be accepted by a CPA.

   (d) For every r, there is an $s > r$ and a language that can be accepted by a quasirealtime CPA with s membranes that cannot be accepted by a quasirealtime CPA with r membranes. (In a CPA, we do not assume that the CPA imports an input symbol from the environment at every step. Quasirealtime means that the CPA has to import an input symbol from the environment with delay of no more than k time steps for some nonnegative integer k independent of the computation.)

   (e) A quasirealtime CPA is strictly weaker than a linear time CPA. (Here, linear

time means that the CPA accepts an input of length n within cn time for some constant c.)

(f) The class of quasirealtime CPA languages is not closed under Kleene + and complementation.

We note that the relationship between PBCMs and sequential symport/antiport P systems (similar to communication P systems) has been studied recently in [16], but only for systems with symbol objects and not as language acceptors. Thus, the results in [16] deal only with tuples of nonnegative integers defined by P systems and counter machines. For example, it was shown in [16] that a set of tuples of nonnegative integers that is definable by a partially blind counter machine can be defined by a sequential symport/antiport system with two membranes. Our new results above cannot be derived from the results in [16].

3. The results for CPA above generalize to cooperative system acceptors with membrane dissolution and bounded creation rules. Hence, the latter are also equivalent to PBCMs.

4. Any recursively enumerable unary language can be accepted by a 1-deterministic 1-membrane CPA with prioritized rules.

5. The reachability problem for sequential catalytic systems with prioritized rules (hence, for 1-deterministic such machines as well) is NP-complete. It follows from this result that a 1-deterministic catalytic system with prioritized rules can only accept recursive languages.

Note that from items 4 and 5 above, when the rules are prioritized, there are 1-deterministic systems that are universal and 1-deterministic systems that are not universal. In contrast, from item 1, without prioritized rules, 1-deterministic systems are not universal.

• **On Symport/Antiport Systems and Semilinear Sets** (by O. Ibarra, S. Woodworth H. Yen, and Z. Dang) in Proc. of the *6th International Workshop on Membrane Computing (WMC6) , (LNCS 3850)*, July 18 - 21, 2005, Vienna, Austria.

A general problem of clear interest in the area of membrane computing or P systems is to find classes of nonuniversal P systems that correspond to (i.e., characterize) known families of languages or subsets of N^k (where N is the set of nonnegative integers, and k is a positive integer), and to investigate their closure and decidability properties. For example, P system characterizations of ET0L, bounded languages accepted by multihead finite automata, and context-sensitive languages are known. Here, we give characterizations of semilinear sets in terms of restricted models of symport/antiport systems.

A popular model of a P system is the symport/antiport system first introduced in [31]. It is a system whose rules closely resemble the way membranes transport objects between themselves in a purely communicating manner. Symport/antiport systems (SA systems) have rules of the form (u, out), (v, in), and $(u, out; v, in)$ where u, v are multisets that are represented as strings (the order in which the symbols are written is not important, since we are only interested in the multiplicities of each symbol). A rule of the form (u, out) in membrane i sends the elements of u from membrane i out to the membrane (directly) containing i. A rule of the form (v, in) in membrane i transports the elements of v into membrane i from the membrane enclosing i. Hence this rule can only be used when the elements of v exist in the outer membrane. A rule of the form $(u, out; v, in)$ simultaneously sends u out of the membrane i while transporting v into membrane i. Hence this rule cannot be applied unless membrane i contains the elements in u and the membrane surrounding i contains the elements in v. The rules are applied in a nondeterministic maximally parallel manner. In general, the number of times a particular rule is applied at anyone step can be unbounded. We require that the application of the rules is maximal: all objects, from all membranes, which *can be* the subject of local evolution rules *have to* evolve simultaneously. Note that there may be several maximal multisets of rules applicable in a step, but we nondeterministically select only one such multiset to apply.

Formally an SA system is defined as

$$M = (V, H, \mu, w_1, \cdots, w_{|H|}, E, R_1, \cdots, R_{|H|}, i_o)$$

where V is the set of objects (symbols) the system uses. H is the set of membrane labels.

The membrane structure of the system is defined in μ. The initial multiset of objects within membrane i is represented by w_i, and the rules are given in the set R_i. E is the set of objects which can be found within the environment, and i_o is the designated output membrane. (When the system is used as a recognizer or acceptor, there is no need to specify i_o.) A large number of papers have been written concerning symport/antiport systems. For example, it has been shown that "minimal" such systems (with respect to the number of membranes, the number of objects, the maximum "size" of the rules) are universal in the sense that they can simulate the computation of Turing machines or, equivalently, counter machines. See the P system website at http://psystems.disco.unimib.it for papers in symport/antiport systems and in the general area of membrane computing, and in particular the monograph [30]. In this paper, we introduce restricted models of symport/antiport systems that are used as acceptors or generators of sets of tuples of nonnegative integers and show that they characterize exactly the semilinear sets.

First, we look at systems that are acceptors. One model is called *simple SA*. The system consists of $k+1$ membranes, arranged in a 2-level structure: membranes m_1, m_2, .., m_k (the *input membranes*) are at the same level and enclosed in membrane m_{k+1} (the *skin membrane*). The set of objects is $V = F \cup \{o\}$, where F is a *finite* set of objects not containing the distinguished symbol o. The restriction is that in the rules of the forms (v, in) and $(u, out; v, in)$, v does not contain o's. Thus, the number of o's in each membrane can only be decreased. The environment initially contains a fixed (finite) multiset over F. The system accepts a k-tuple $(n_1, ..., n_k)$ of nonnegative integers if, when the k input membranes are given $o^{n_1}, ..., o^{n_k}$ and no o's in membrane m_{k+1} (with some fixed strings $w_1, ..., w_{k+1} \in F^*$ in membranes $m_1, ..., m_{k+1}$, respectively), the system halts (i.e., no rule in any of the membranes is applicable). We show that a set $R \subseteq N^k$ is accepted by a simple SA if and only if it is a semilinear set. (This result generalizes to the case when there is an infinite supply of o's in the environment, and the v's can contain o's in the rules in the skin membrane m_{k+1}.) As a consequence, the class of sets of tuples accepted by these SAs are closed under union, intersection, and complementation. Moreover, the emptiness,

disjointness, containment, and equivalence problems for simple SAs are decidable. When the model is generalized to a multi-level structure, the set of tuples accepted need no longer be semilinear. In particular, suppose we have a k-membrane SA, where membrane m_i is enclosed in membrane m_{i+1} for $1 \leq i \leq k-1$. Membrane m_1 is the only input membrane and membrane m_k is the skin membrane. Again, in the rules (v, in) and $(u, out; v, in)$, v does not contain o's. We call this model a *k-membrane cascade SA*. Note that the system accepts a subset of N. We show that 3-membrane cascade SAs can accept nonsemilinear subsets of N. We also prove that their emptiness problem is undecidable by showing that they can simulate the computations of two-counter machines.

The k-membrane cascade SA can be generalized. A *k-membrane extended cascade SA* has a set of objects $V = F \cup \Sigma_r$, where now the input alphabet is $\Sigma_r = \{a_1, ..., a_r\}$ $(r \geq 1)$. Again the rules are restricted in that in the rules of the forms (v, in) and $(u, out; v, in)$, v does not contain any symbol in Σ_r. The environment initially contains only a fixed multiset over F. Also, there are fixed strings $w_1, ..., w_k \in F^*$ such that the system starts with $w_1 a_1^{n_1} ... a_r^{n_r}$ in membrane m_1 (the input membrane) and w_i in membrane m_i for $2 \leq i \leq k$. If the system halts, then we say that the r-tuple $(n_1, ..., n_r)$ is accepted. We show that a set $R \subseteq N^r$ is accepted by a 1-membrane extended cascade SA if and only if it is semilinear. However, 2-membrane extended cascade SAs can accept nonsemilinear sets, and their emptiness problem is undecidable, even for $r = 2$ (i.e., there are two symbols in the input alphabet). Note that for the case $r = 1$ (i.e., Σ contains only a single symbol), the set of unary numbers is semilinear (since this is a special case of the result above for 2-level simple SA).

We then consider symport/antiport models that are used as generators. One such model is a 2-level symport/antiport system with membranes $m_1, ..., m_k, m_{k+1}$, where membranes $m_1, ..., m_k$ are at the same level, and they are enclosed in the skin membrane m_{k+1}. There is an infinite supply of o's in the environment (but the initial multiplicities of symbols in F in the environment are fixed). We require that for membranes $m_1, ..., m_k$, in the rules of the forms (u, out) and (u, out, v, in), u does not contain o's. Note that there is no restriction on the rules in the skin membrane. We say that $(n_1, ..., n_k)$ is generated if, when started with no

o's in the system and fixed $w_i \in F^*$ in membrane m_i $(1 \leq i \leq k+1)$, the system halts with $o^{n_1}, ..., o^{n_k}$ in membranes $m_1, ..., m_k$. We call this system a *simple SA generator*. We show that a set $R \subseteq N^k$ is generated by a simple SA generator if and only if R is a semilinear set. Again, generalizing the model to have at least 3 levels would allow it to generate a nonsemilinear set. In fact, for any recursively enumerable (RE) set R, the set $\{2^n \mid n \in R\}$ can be accepted by a 3-level system, while R can be accepted by a 4-level system.

We also look at a 1-membrane symport/antiport system with a set of objects $V = F \cup \Sigma_r$, where $\Sigma_r = \{a_1, ..., a_r\}$, and whose rules are restricted so that in the rules of the forms (u, out) and $(u, out; v, in)$, u does not contain any symbol in Σ_r. Thus symbols in Σ_r can only be transported from the environment into the membrane (note that, by the restriction, once these symbols enter the membrane, they remain in the membrane). The system starts with a fixed string $w \in F^*$. The environment initially contains a fixed multiset over F and an infinite supply of each a_i $(1 \leq i \leq r)$. We show that the sets of r-tuples generated by these systems are exactly the semilinear sets over N^r. However, when there are 2 membranes, where again, the second (i.e., innermost) membrane cannot transport symbols in Σ_r into the first (skin) membrane, the set of tuples generated by such a system need not be semilinear. In fact, for any RE set R, the set $\{(2^n, 0) \mid n \in R\}$ can be generated by a 2-membrane system with input alphabet Σ_2, while the set $\{(n, 0, 0) \mid n \in R\}$ can be generated by a 2-membrane system with input alphabet Σ_3.

References

[1] P. Abdulla, A. Annichini, and A. Bouajjani, Symbolic Verification of Lossy Channel Systems: Application to the Bounded Retransmission Protocol, in *Proc. TACAS'99*, LNCS 1579, pp. 208–222, 1999.

[2] R. Alur, and D. Dill, Automata for Modeling Real-Time Systems, in *Proc. 17th ICALP*, LNCS 443, pp. 332–335, 1990.

[3] R. Alur, T. Henzinger, M. Vardi, Parametric Real-Time Reasoning, in *Proc. 25th ACM STOC*, pp. 592–601, 1993.

[4] Nina Amla, E. Allen Emerson, and Kedar S. Namjoshi. Efficient decompositional model checking for regular timing diagrams. In *Conference on Correct Hardware Design and Verification Methods*, pages 67–81, 1999.

[5] Ioan I. Ardelean, Matteo Cavaliere, and Dragos Sburlan. Computing using signals: From cells to P Systems. In *Second Brainstorming Week on Membrane Computing, Sevilla, Spain, February 2-7 2004*, pages 60–73, Sevilla, Spain, February 2-7 2004.

[6] W. M. Becker, L. J. Kleinsmith, and J. Hardin. *The World of the Cell (5th Edition)*. Benjamin Cummings (San Francisco), 2003.

[7] P. Bottoni, C. Martin-Vide, Gh. Paun, and G. Rozenberg. Membrane systems with promoters/inhibitors. *Acta Informatica*, 38(10):695–720, 2002.

[8] E. M. Clarke, O. Grumberg, and D. A. Peled. *Model Checking*. The MIT Press, 1999.

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

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

[11] J. Esparza. More infinite results. In Proc. of INFINITY'96, Research Report MIP-9614, University of Passau, 1996.

[12] J. Esparza. Decidability of model-checking for infinite-state concurrent systems. Acta Informatica, 34:85-107, 1997.

[13] K. Fisler. *A Unified Approach to Hardware Verification Through a Heterogeneous Logic of Design Diagrams*. PhD thesis, Computer Science Department, Indiana University, August 1996.

[14] R. Freund, G. Păun, On deterministic P systems, See P Systems Web Page at http://psystems.disco.unimib.it.

[15] R. Freund, L. Kari, M. Oswald, P. Sosik, Computationally universal P systems without priorities: two catalysts are sufficient, Theoretical Computer Science 330 (2) (2005) 251–266.

[16] Frisco, P.: About P systems with symport/antiport, *Second Brainstorming Week on Membrane Computing, Sevilla, Spain*, 2004, 224–236.

[17] Gouda, M., Howell, R., Rosier, L.: The instability of self-stabilization, *Acta Informat.*, **27**, 1990, 697–724.

[18] Greibach, S. A.: Remarks on blind and partially blind one-way multicounter machines, *Theoretical Computer Science*, **7**(3), 1978, 311–324.

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

[20] T. Hune, J. Romijn, M. Stoekinga, and F. Vaandrager, Linear Parametric Model Checking of Timed Automata, in *Proc. TACAS*, LNCS 2031, pp. 189-203, 2001.

[21] Howell, R., Rosier, L. and Yen, H. Normal and sinkless Petri nets, *J. of Computer and System Sciences* **46**, 1-26, 1993.

[22] Ichikawa, A. and Hiraishi, K. Analysis and control of discrete event systems represented by Petri nets, *LNCIS* **103** ,115-134, 1987.

[23] Kosaraju, R. Decidability of reachability in vector addition systems, *Proc. the 14th Annual ACM Symposium on Theory of Computing*, 267-280, 1982.

[24] Lipton, R. *The reachability problem requires exponential space*, Technical Report 62, Yale University, Dept. of CS., Jan. 1976.

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

[26] Mayr, E. An algorithm for the general Petri net reachability problem, *SIAM J. Comput.* **13**, 441-460, 1984.

[27] C. Martin-Vide and Gh. Paun. Computing with membranes (P systems): universality results. In *MCU*, volume 2055 of *Lecture Notes in Computer Science*, pages 82–101. Springer, 2001.

[28] G. Păun, Computing with membranes, Turku University Computer Science, Research Report No. 208, 1998.

[29] G. Păun, Computing with membranes, Journal of Computer and System Sciences 61 (1) (2000) 108–143.

[30] G. Păun, Membrane Computing: An Introduction, Springer-Verlag, 2002.

[31] A. Păun, G. Păun, The power of communication: P systems with symport/antiport, New Generation Computers 20 (3) (2002) 295–306.

[32] Reisig, W., Petri Nets: An Introduction, Springer-Verlag New York, Inc., New York, NY, 1985.

[33] P. Sosik, P systems versus register machines: two universality proofs, in: Pre-Proceedings of Workshop on Membrane Computing (WMC-CdeA2002), Curtea de Argeş, Romania, 2002, pp. 371–382.

[34] P. Sosik and R. Freund. P systems without priorities are computationally universal. In *WMC-CdeA2002*, volume 2597 of *Lecture Notes in Computer Science*, pages 400–409. Springer, 2003.

[35] R. Valk, and M. Jantzen, The Residue of Vector Sets with Applications to Decidability in Petri Nets, *Acta Informatica*, 21, 643-674, 1985.

[36] F. Wang, Parametric Timing Analysis for Real-Time Systems, *Information and Computation*, 130(2), 131-150, 1996. Also in *Proc. 10th IEEE LICS*, 1995.

[37] F. Wang, Parametric Analysis of Computer Systems, *Formal Methods in System Design*, 17, 39-60, 2000.

[38] F. Wang, and H. Yen, Parametric Optimization of Open Real-Time Systems, in *Proc. SAS 2001*, LNCS 2126, pp. 299-318, 2001.

[39] Yamasaki, H. Normal Petri nets, *Theoretical Comput. Science* **31**, 307-315, 1984.

[40] Yen, H. A valuation-based analysis of conflict-free Petri nets, *Systems and Control Letters* **45(5)**, 387-395, 2002.

[41] Yen, H. On the regularity of Petri net languages, *Inform. and Comput.,* **124(2)**, 168-181, 1996.

[42] Yen, H. On reachability equivalence for BPP-nets, *Theoretical Computer Science,* **179**, 301-317, 1997.