

# Uniformity conditions in natural computing

Niall Murphy<sup>1\*</sup> and Damien Woods<sup>2\*\*</sup>

<sup>1</sup> Department of Computer Science, National University of Ireland Maynooth, Ireland  
nmurphy@cs.nuim.ie

<sup>2</sup> California Institute of Technology, Pasadena, CA 91125, USA. woods@caltech.edu

**Abstract.** We investigate computing models that are presented as families of finite computing devices with a uniformity condition on the entire family. Examples include circuits, membrane systems, DNA computers, cellular automata, tile assembly systems, and so on. However, in this list there are actually two distinct kinds of uniformity condition. The first is the most common and well-understood, where each input length is mapped to a single computing device that computes on the finite set of inputs of that length. The second, called semi-uniformity, is where each input is mapped to a computing device for that input. The former notion is well-known and used in circuit complexity, while the latter notion is frequently found in literature on nature-inspired computation from the past 20 years or so. Are these two notions distinct or not? For many models it has been found that these notions are in fact the same, in the sense that the choice of uniformity or semi-uniformity leads to characterisations of the same complexity classes. Here, we buck this trend and show that these notions are actually distinct: we give classes of uniform membrane systems that are strictly weaker than their semi-uniform counterparts. This solves a known open problem in the theory of membrane systems.

## 1 Introduction

Many of the early DNA computing algorithms [13,20,21,24] involved mapping a specific instance of an **NP**-hard problem (such as Maximal Clique) to a set of DNA strands and lab protocols and then using well-known biomolecular techniques to solve the problem. To assert generality for such an algorithm one would define a mapping from arbitrary problem instances to sets of DNA polymers and experimental protocols. In order to claim that this mapping is not doing the essential computation, it would have to be easily computable (for example, logspace computable). Circuit uniformity (introduced by Borodin [8]) provides a well-established framework where we map each input length  $n \in \mathbb{N}$  to a circuit  $c_n \in C$ , with a suitably simple mapping. However, the DNA computing algorithms cited above do something different, they map a specific *instance* of the problem to a specific computing device. This latter notion is called *semi-uniformity* [29,26], and in fact quite a number of nature-inspired computational models use semi-uniformity. This raises the immediate question of whether the notions of uniformity and semi-uniformity are computationally equivalent.

It has been shown in a number of models that whether one chooses to use uniformity or semi-uniformity does not affect the power of the model. However, in this paper we show that these notions are not equivalent. We prove that choosing one notion over another gives characterisations of completely different complexity classes, including known distinct classes.

In Section 3 we prove this result for a computational model called membrane systems (also known as P-systems) [28]. Membrane computing is a branch of natural computing which defines computational models that are inspired by the structure and function of

---

\* Supported by the Irish Research Council for Science, Engineering and Technology.

\*\* Supported by Junta de Andalucía grant TIC-581 (Spain) and National Science Foundation Grant 0832824, the Molecular Programming Project (USA).

living cells. The membrane computing model is sufficiently formal that the uniform versus semi-uniform question has been clearly stated as Open Problem C in [30].

Why is this result surprising? We know that every class of problems solved by a uniform family of devices is contained in the analogous semi-uniform class, since the former is a restriction of the latter. However, in all membrane system models studied to date, the classes of problems solved by semi-uniform and uniform families turned out to be equal, see, e.g., [3,22,35]. Specifically, if we want to solve some problem, by specifying a family of membrane systems (or some other model), it is often much easier to first use the more general notion of semi-uniformity, and then subsequently try to find a uniform solution. In almost all cases where a semi-uniform family was given for some problem [2,23,26,35], at a later point a uniform version of the same result was published [1,3,26]. Here we prove that this improvement is not always possible.

Our result proves something general about families of finite devices that is independent of particular formalisms and can be applied to other computational models besides membrane systems. To demonstrate this principle, in Appendix A we define the notion of semi-uniformity for Boolean circuits and obtain an analogous result (although the proof is easier for the circuit model).

Besides membrane systems and circuits, some other models that use notions of uniformity and semi-uniformity include families of molecular and DNA computers, tile assembly systems, neural networks, branching programs and cellular automata [5,9,25,33,34]. In Section 4 we argue that our result is of importance to the natural computing community since it highlights that the (seemingly harmless) choice between uniformity and semi-uniformity in these models may lead to drastic changes in computational power. In this paper we see this when solving problems from the class **NL**, a class with many interesting, yet tractable, problems [15,17]. Furthermore, our work suggests that this question should be asked of other nature-inspired, and more standard, models.

## 1.1 Statement of main result

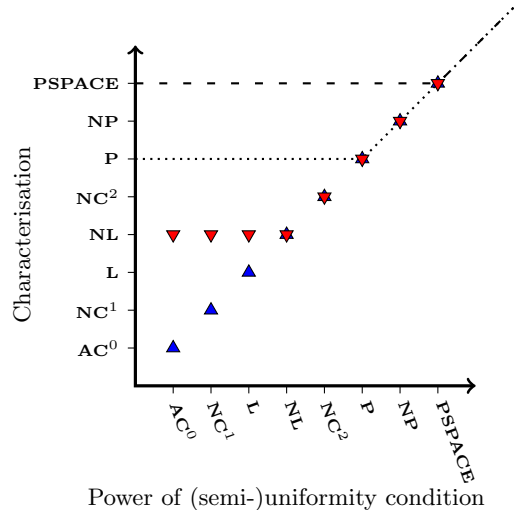
We show that a class of problems, that is characterised by  $\mathbf{AC}^0$ -uniform membrane systems of a certain type, is a strict subset of another class that is characterised by  $\mathbf{AC}^0$ -semi-uniformity systems of the same type. Besides their respective use of uniformity and semi-uniformity both models are identical, this shows that for the membrane systems we consider, semi-uniformity is a strictly stronger notion than uniformity. Specifically, we show that the uniform systems characterise  $\mathbf{AC}^0$  and the semi-uniform systems characterise **NL**, two classes known to be distinct. In the notation of membrane systems our main result is written as follows (explanations of notation are found in Section 2).

**Theorem 1.**  $\mathbf{AC}^0 = (\mathbf{AC}^0, \mathbf{AC}^0)\text{-PMC}_{\mathcal{AM}^0_d} \subsetneq (\mathbf{AC}^0)\text{-PMC}^*_{\mathcal{AM}^0_d} = \mathbf{NL}$

The left hand equality is proved in this paper, while the right hand equality was given in [23]. In Figure 1, Theorem 1 is illustrated by the leftmost pair of triangles. Essentially, the figure shows that if we use  $\mathbf{AC}^0$  uniformity, the systems characterise  $\mathbf{AC}^0$ , while with  $\mathbf{AC}^0$  semi-uniformity they characterise **NL**.

In fact we can also state a more general result for a number of complexity classes below **NL**, for brevity we keep the list short.

**Theorem 2.** *Let  $\mathbf{C} \in \{\mathbf{AC}^0, \mathbf{NC}^1, \mathbf{L}\}$  and assume that  $\mathbf{NC}^1 \subsetneq \mathbf{L} \subsetneq \mathbf{NL}$  then  $\mathbf{C} = (\mathbf{C}, \mathbf{C})\text{-PMC}_{\mathcal{AM}^0_d} \subsetneq (\mathbf{C})\text{-PMC}^*_{\mathcal{AM}^0_d} = \mathbf{NL}$*



**Fig. 1.** Complexity classes characterised by the membrane systems studied in this paper. Characterisations by uniform systems are denoted by  $\blacktriangle$ , and semi-uniform by  $\blacktriangledown$ . For example, Theorem 1 is illustrated by the fact that  $\mathbf{AC}^0$ -uniform systems characterise  $\mathbf{AC}^0$ , and that  $\mathbf{AC}^0$ -semi-uniform systems characterise  $\mathbf{NL}$ . Also shown are the previously known  $\mathbf{P}$  [22] (.....) and  $\mathbf{PSPACE}$  [3,36] (- -) results, where semi-uniform and uniform classes have the same power.

This shows that, roughly speaking, uniform  $\mathcal{AM}_{-d}^0$  membrane systems are essentially powerless, they are as weak and as strong as their uniformity condition. In Figure 1, Theorem 2 is illustrated by the six triangles to the left of (and including) the uniformity condition  $\mathbf{L}$ .

The essential ideas behind the proof of these theorems are as follows. In the semi-uniform case the membrane systems have  $\mathbf{NL}$  power. However we go on to prove that in the uniform case, the systems are severely crippled. We show that even though such membrane systems in a uniform family may have an  $\mathbf{NL}$ -complete prediction problem, there is another uniform family that solves the same problem where each system can be predicted in  $\mathbf{AC}^0$ . This, along with some other tools, is used to show that if the power of the uniformity notion is  $\mathbf{AC}^0$  or more, then the power of the entire family of systems is determined by the power of the uniformity condition.

## 2 Definitions for membrane systems

In this section we define membrane systems and some complexity classes, these definitions are based on those from [12,26,27,28,36]. The set of all multisets over a set  $A$  is denoted  $\mathbf{MS}(A)$ .

### 2.1 Active membrane systems

Active membrane systems are a class of membrane systems with membrane division rules. Here division rules act only on elementary membranes, which are membranes that do not contain other membranes (i.e. leaves in the membrane structure).<sup>3</sup>

<sup>3</sup> The more complicated non-elementary membrane division rule is also considered in the literature (where membranes containing other membranes can divide and replicate all of their substructure). All results

**Definition 3.** An active membrane system without charges is a 6-tuple  $\Pi = (O, \mu, M, H, L, R)$  where,

1.  $O$  is the alphabet of objects (the set of object types);
2.  $\mu = (V_\mu, E_\mu, r)$  is a tree, with root  $r$ , representing the membrane structure, where  $V_\mu$  is a finite subset of  $\mathbb{N}$  and  $E_\mu \subsetneq V_\mu \times V_\mu$ ,  $\nexists(v, r) \in E_\mu, r \in V_\mu$ ;
3.  $M : V_\mu \rightarrow \text{MS}(O)$  maps membranes to their multisets;
4.  $H$  is the finite set of membrane labels;
5.  $L : V_\mu \rightarrow H$  maps membranes to their labels;
6.  $R$  is a finite set of developmental rules of the following types (where  $a, b, c \in O$  and  $u \in \text{MS}(O)$ ,  $h \in H$ ):
  - (a)  $[a \rightarrow u]_h$  (object evolution),
  - (b)  $a [ ]_h \rightarrow [b]_h$  (communication in),
  - (c)  $[a]_h \rightarrow [ ]_h b$  (communication out),
  - (d)  $[a]_h \rightarrow b$  (membrane dissolution),
  - (e)  $[a]_h \rightarrow [b]_h [c]_h$ , (elementary membrane division).

The vertices  $V_\mu$  of the membrane structure tree  $\mu$  are the individual membranes of the system. The parent of all membranes in the system (the root vertex  $r$  in  $\mu$ ) is called the “skin” and has label  $0 \in H$ . A *configuration*  $\mathcal{C}$  of a membrane system is a tuple  $(\mu, M, L)$  whose elements are defined in Definition 3. A *permissible encoding* of a membrane system  $\langle \Pi \rangle$ , or a configuration  $\langle \mathcal{C} \rangle$ , encodes all multisets in a unary manner. For example, a multiset must be specified in the format  $[a, a, a, b, b]$ , rather than  $a^3b^2$ , in order to ensure that at most a polynomial number of objects are initially encoded in a system.

The rules in the set  $R$  are applied to a configuration according to the following principles:

- All the rules are applied in a *maximally parallel manner*. In each timestep, each object in a membrane can only be used for one rule (non-deterministically chosen when there are several possibilities), but any object which can evolve by a rule of any form must do so.
- If a membrane labelled  $h$  is divided by a rule of type (e) and there are objects in this membrane which evolve via rules of type (a), then we assume that first the evolution (a) rules are used, and then the division (e) rules. This process takes only one step.
- The rules associated with membranes labelled with  $h$  are used for membranes with that label. In each timestep, a membrane can be the subject of only one rule of types (b)–(e).

A *computation* of a membrane system is a sequence of configurations such that each configuration (except the initial one) is obtained from the previous one by a transition (one-step maximally parallel application of the rules). Membrane systems are non-deterministic, therefore on a given input there are multiple possible computations. A computation that reaches a configuration where no more rules are applicable is called a *halting computation*.

**Definition 4.** A recogniser membrane system is a membrane system  $\Pi$  such that:

1. all computations halt,
2. **yes**, **no**  $\in O$ ,
3. the object **yes** or object **no** (but not both) appear in the multiset of the membrane with label 0 (the skin),
4. and this happens only in the halting configuration.

---

in this paper hold when we permit non-elementary division, however we omit this detail as it adds unnecessary complications to our definitions and proofs.

## 2.2 Complexity classes

A problem is a set  $X = \{x_1, x_2, \dots\} \subseteq \Sigma^*$  and its complement is  $\bar{X} = \Sigma^* - X$  where  $\Sigma$  is some finite alphabet. We say that a *family*  $\Pi$  of membrane systems recognises a problem if for each  $x \in \Sigma^*$  there is some  $\Pi \in \Pi$  that decides if  $x \in X$ . We denote by  $|x| = n$  the length of any instance  $x \in \Sigma^*$ . Throughout this paper,  $\mathbf{AC}^0$  circuits are **DLOGTIME**-uniform, polynomial sized (in input length  $n$ ), constant depth, circuits with AND, OR and NOT gates, and unbounded fan-in [7].  $\mathbf{FP}$ ,  $\mathbf{FL}$ , and  $\mathbf{FAC}^0$  are the classes of functions that are respectively computable by deterministic Turing Machines in polynomial time, by deterministic Turing machines using logarithmic space, and by **DLOGTIME**-uniform polynomial-sized alternating circuits with unbounded fan-in and constant depth.

**Definition 5.** *Let  $\mathcal{R}$  be a class of recogniser membrane systems and let  $t : \mathbb{N} \rightarrow \mathbb{N}$  be a total function. Let  $\mathbf{E}$  and  $\mathbf{F}$  be classes of functions. The class of problems solved by a  $(\mathbf{E}, \mathbf{F})$ -uniform family of membrane systems of type  $\mathcal{R}$  in time  $t$ , denoted  $(\mathbf{E}, \mathbf{F})\text{-MC}_{\mathcal{R}}(t)$ , contains all problems  $X$  such that:*

- *There exists an  $\mathbf{F}$ -uniform family of membrane systems,  $\Pi = \{\Pi_1, \Pi_2, \dots\}$  of type  $\mathcal{R}$ : that is, there exists a function  $f \in \mathbf{F}$ ,  $f : \{1\}^* \rightarrow \Pi$  such that  $f(1^n) = \Pi_n$ , where  $|x| = n$ .*
- *There exists an input encoding function  $e \in \mathbf{E}$ ,  $e : X \cup \bar{X} \rightarrow \text{MS}(I)$  such that  $e(x)$  is the input multiset, which is placed in a specific input membrane of  $\Pi_n$ , where  $|x| = n$  and  $I \subsetneq O$  is the set of input objects.*
- *$\Pi$  is  $t$ -efficient:  $\Pi_n$  always halts in at most  $t(n)$  steps.*
- *The family  $\Pi$  is sound with respect to  $(X, e, f)$ ; that is if there is an accepting computation of the system  $\Pi_{|x|}$  on input multiset  $e(x)$  then  $x \in X$ .*
- *The family  $\Pi$  is complete with respect to  $(X, e, f)$ ; that is, for each input  $x \in X$ , every computation of the system  $\Pi_{|x|}$  on input multiset  $e(x)$  is accepting.*

We now define semi-uniform families of membrane systems where a single function (rather than two) is used to construct the family. For each instance  $x \in X \cup \bar{X}$  we have a (possibly unique) membrane system which does not need a separately constructed input, a clear departure from the spirit of circuit uniformity.

**Definition 6.** *Let  $\mathbf{H}$  be a class of functions. The class of problems solved by a  $(\mathbf{H})$ -semi-uniform family of membrane systems of type  $\mathcal{R}$  in time  $t$ , denoted  $(\mathbf{H})\text{-MC}_{\mathcal{R}}^*(t)$ , contains all problems  $X$  such that:*

- *There exists a  $\mathbf{H}$ -semi-uniform family  $\Pi = \{\Pi_{x_1}, \Pi_{x_2}, \dots\}$  of membrane systems of type  $\mathcal{R}$ : that is, there exists a function  $h \in \mathbf{H}$ ,  $h : X \cup \bar{X} \rightarrow \Pi$  such that  $h(x_i) = \Pi_{x_i}$ .*
- *$\Pi$  is  $t$ -efficient:  $\Pi_x$  always halts in at most  $t(|x|)$  steps.*
- *The family  $\Pi$  is sound with respect to  $(X, h)$ ; that is, for each  $x \in X \cup \bar{X}$  if there exists an accepting computation of the system  $\Pi_x$  then  $x \in X$ .*
- *The family  $\Pi$  is complete with respect to  $(X, h)$ ; that is, for each  $x \in X$  every computation of the system  $\Pi_x$  is accepting.*

We define the set of languages decided by a uniform family, and respectively semi-uniform family of membrane systems in polynomial time to be

$$(\mathbf{E}, \mathbf{F})\text{-PMC}_{\mathcal{R}} = \bigcup_{k \in \mathbb{N}} (\mathbf{E}, \mathbf{F})\text{-MC}_{\mathcal{R}}(n^k) \quad \text{and} \quad (\mathbf{H})\text{-PMC}_{\mathcal{R}}^* = \bigcup_{k \in \mathbb{N}} (\mathbf{H})\text{-MC}_{\mathcal{R}}^*(n^k).$$

When the symbols  $\mathbf{E}$ ,  $\mathbf{F}$ , and  $\mathbf{H}$  are replaced by complexity class names such as  $\mathbf{AC}^0$ ,  $\mathbf{L}$  or  $\mathbf{P}$  it means that the uniformity conditions under consideration are in the function

versions of these classes. For example, if we let  $\mathbf{E} = \mathbf{F} = \mathbf{AC}^0$  then we mean that the functions  $e \in \mathbf{E}$  and  $f \in \mathbf{F}$  are computable in uniform  $\mathbf{FAC}^0$  and we say we have an  $\mathbf{AC}^0$ -uniform family.

Let  $\mathcal{AM}_{-d}^0$  denote the class of membrane systems that obey Definition 4, and Definition 3 but without dissolution (rule  $(d)$ ). Thus  $(\mathbf{AC}^0, \mathbf{AC}^0)\text{-PMC}_{\mathcal{AM}_{-d}^0}$  (respectively,  $(\mathbf{AC}^0)\text{-PMC}_{\mathcal{AM}_{-d}^0}^*$ ) denotes the class of problems solvable by  $\mathbf{AC}^0$ -uniform (respectively,  $\mathbf{AC}^0$ -semi-uniform) families of active membrane systems without charges in polynomial time with no dissolution rules.

*Remark 7.* A membrane system is said to be *confluent* if it is both sound and complete. That is, a membrane system  $\Pi$  is *confluent* if all computations of  $\Pi$  with the same input  $x$  (properly encoded) give the same result; either always “accepts” or else always “rejects”.

In a confluent membrane system, given a fixed initial configuration, the system non-deterministically chooses one from a number of valid computations (configuration sequences), but all of these computations must lead to the same result, either all accepting or all rejecting.

**Definition 8.** *Given a configuration  $C_s$  of an  $\mathcal{AM}_{-d}^0$  system  $\Pi$ , we say that an object type  $o_s$  in a membrane labeled  $h_s$  in  $C_s$  eventually evolves object type  $o_t$  in membrane labeled  $h_t$  if there is a computation where there is an unbroken sequence of rules (an object type in the right hand side of rule  $i$  in the sequence is in the left hand side of rule  $i + 1$ , and correct membrane labels are used) that begin with an object of type  $o_s$  and the result of the sequence is an object of type  $o_t$  in a membrane labeled  $h_t$ .*

### 3 Proof of main result: uniform families without dissolution

The equality on the right hand side of Theorem 1 states that certain  $(\mathbf{AC}^0)$ -semi-uniform systems characterise  $\mathbf{NL}$ . This was shown in [23], we quote the result:

**Theorem 9.**  $(\mathbf{AC}^0)\text{-PMC}_{\mathcal{AM}_{-d}^0}^* = \mathbf{NL}$

In this section we show that the analogous  $(\mathbf{AC}^0, \mathbf{AC}^0)$ -uniform systems characterise  $\mathbf{AC}^0$ , which proves the left hand side equality of Theorem 1.

The following theorem is key to the proof of Theorems 1 and 2. Roughly speaking, Theorem 10 states that in uniform membrane systems of the type we consider, the uniformity condition dominates the computational power of the system. By letting  $\mathbf{E} = \mathbf{F} = \mathbf{AC}^0$ , the statement of Theorem 10 gives us the left hand side equality in Theorem 1. By letting  $\mathbf{E} = \mathbf{F} \in \{\mathbf{AC}^0, \mathbf{NC}^1, \mathbf{L}\}$  we get the left hand side of Theorem 2. The remaining classes quoted in the theorem serve to illustrate Figure 1.

**Theorem 10.** *Let  $\mathbf{E}, \mathbf{F} \in \{\mathbf{AC}^0, \mathbf{NC}^1, \mathbf{L}, \mathbf{NL}, \mathbf{NC}^2, \mathbf{P}, \mathbf{NP}, \mathbf{PSPACE}\}$  and let  $\mathbf{F} \subseteq \mathbf{E}$ . Then  $(\mathbf{E}, \mathbf{F})\text{-PMC}_{\mathcal{AM}_{-d}^0} = \mathbf{E}$ .*

The proof works by showing that for each uniform membrane family  $\Pi$  (of polynomial sized recogniser  $\mathcal{AM}_{-d}^0$  systems), that decides problem  $X$ , there exists another family  $\Pi'$  that decides  $X$  but where every system in  $\Pi'$  is so greatly simplified that  $\Pi'$  can be evaluated by a uniform family of  $\mathbf{AC}^0$  circuits.

First we observe that Points (3) and (4) from Definition 4 allow us to define the following subsets of the objects  $O$  in an  $\mathcal{AM}_{-d}^0$  system.  $O_{\text{yes}} = \{o \mid o \in O \text{ and } o \text{ eventually evolves yes}\}$ ,  $O_{\text{no}} = \{o \mid o \in O \text{ and } o \text{ eventually evolves no}\}$ , and  $O_{\text{other}} = O \setminus (O_{\text{yes}} \cup O_{\text{no}})$ .

**Lemma 11.**  $O_{\text{yes}} \cap O_{\text{no}} = \emptyset$ .

*Proof.* Assume that object  $o \in O_{\text{yes}} \cap O_{\text{no}}$ , this implies that both a **yes** and a **no** object are produced by the confluent system which contradicts point (3) of Definition 4.  $\square$

**Lemma 12.** For  $\Pi_n$  in an arbitrary uniform family of recogniser  $\mathcal{AM}_{-d}^0$  systems, a size-two input alphabet  $I = \{\mathbf{a}, \mathbf{b}\}$ , such that exactly one of  $\mathbf{a}, \mathbf{b}$  is from set  $O_{\text{yes}}$  and the other from set  $O_{\text{no}}$ , is both necessary and sufficient in the sense that this restriction does not alter the computing power of the system  $\Pi_n$ .

*Proof.* It can be seen, as follows, that it is *necessary* that the input alphabet  $I$  contains at least 1 object from  $O_{\text{yes}}$  and 1 from  $O_{\text{no}}$ . In a uniform family of recogniser membrane systems (see Definition 4) each membrane system  $f(1^n) = \Pi_n$  either accepts or rejects each input  $e(x)$  where  $x \in \Sigma^n$  (see Definition 5), and in particular there are families where for each  $n$  a nonzero number of inputs are accepted and a nonzero number of inputs rejected. Thus, in such general cases, each  $\Pi_n$  has the potential to evolve either of the **yes** or **no** objects. In Lemma 11 we saw that  $O_{\text{yes}} \cap O_{\text{no}} = \emptyset$ , and so the input set  $I$ , of each  $\Pi_n$ , has at least one object in  $O_{\text{yes}} \cap I$  and at least one object in  $O_{\text{no}} \cap I$ .

It can be seen that it is *sufficient* for set  $I$  to contain a single object from each set  $O_{\text{yes}}, O_{\text{no}}$  as follows. Given a membrane system  $\Pi_n$  which is a member of a family that recognises  $X$  and is uniform by functions  $(e, f)$  with  $|I| > 2$ , we describe another system  $\Pi'_n$  in a uniform family by functions  $(e', f')$  that also recognises  $X$ . The system  $\Pi'_n$  is identical to  $\Pi_n$  except that: the set of objects is  $O' = O \cup \{\mathbf{a}, \mathbf{b}\}$ , the set of input objects (and the range of  $e'$ ) is  $I' = \{\mathbf{a}, \mathbf{b}\}$ , and there are two extra type (a) evolution rules for the input membrane:  $[\mathbf{a} \rightarrow I \cap O_{\text{yes}}]_{\text{in}}, [\mathbf{b} \rightarrow I \cap O_{\text{no}}]_{\text{in}}$ . In the first step of its computation,  $\Pi'_n$  uses exactly one of the input objects ( $\mathbf{a}$  or  $\mathbf{b}$ , created by  $e'(x)$ ) to generate the set of objects produced by  $e(x)$  (as well as some “extra” objects,  $(I \cap O_{\text{yes}}) \setminus e(x)$  or  $(I \cap O_{\text{no}}) \setminus e(x)$ ). That is,  $\Pi'_n$  evolves those objects from  $I$  that  $e(x)$  produces for  $\Pi_n$  (and some extra objects which do not change the outcome of the computation because they are either all from  $O_{\text{yes}}$  or else all from  $O_{\text{no}}$ ). Thus uniform families of recogniser  $\mathcal{AM}_{-d}^0$  systems need at most an input alphabet  $I = \{\mathbf{a}, \mathbf{b}\}$  with exactly one of  $\mathbf{a}, \mathbf{b}$  from set  $O_{\text{yes}}$  and the other from set  $O_{\text{no}}$ .  $\square$

Lemma 12 permits us to consider only those systems that have two input objects  $I = \{\mathbf{a}, \mathbf{b}\}$ . Thus we restrict attention to the case where the input encoding function is of the form  $e' : X \rightarrow \{\mathbf{a}, \mathbf{b}\}$ . We say that  $e'$  is a characteristic function with range  $\{\mathbf{a}, \mathbf{b}\}$ .

**Lemma 13.** Let  $\Pi$  be a uniform family (by functions  $e$  and  $f$ ) of confluent recogniser  $\mathcal{AM}_{-d}^0$  systems (with  $I = \{\mathbf{a}, \mathbf{b}\}$  via Lemma 12) which recognises instances of  $X$ . There exists a family  $\Pi_{\text{min}}$  that also recognises instances of  $X$  but uses a uniformity function  $f_{\text{min}}$  whose range is a set with only two membrane systems, both of which can be evaluated in  $\mathbf{AC}^0$ .

*Proof.* Consider the membrane system  $f(n) = \Pi_n \in \Pi$ . The essential property of this system is that one object in its input set  $I = \{\mathbf{a}, \mathbf{b}\}$  eventually evolves to **yes** while the other eventually evolves to **no**. That is:  $\mathbf{a} \in O_{\text{yes}} \Rightarrow \mathbf{b} \in O_{\text{no}}$  and  $\mathbf{a} \in O_{\text{no}} \Rightarrow \mathbf{b} \in O_{\text{yes}}$ .

However, this essential property is captured by two extremely simple membrane systems, called  $\Pi_{\text{pos}}$  and  $\Pi_{\text{neg}}$ , each with only 4 objects and a single membrane labelled 0. Both systems have  $i = 0$  as the input membrane and are of the following form:

$$\Pi = (\{\mathbf{a}, \mathbf{b}, \text{yes}, \text{no}\}, (\{0\}, \emptyset, 0), \{(0, \emptyset)\}, \{0\}, \{(0, 0)\}, R)$$

where  $\Pi = \Pi_{\text{pos}}$  if the rules are  $R = \{[\mathbf{a} \rightarrow \text{yes}]_0, [\mathbf{b} \rightarrow \text{no}]_0\}$  and  $\Pi = \Pi_{\text{neg}}$  if the rules are  $R = \{[\mathbf{a} \rightarrow \text{no}]_0, [\mathbf{b} \rightarrow \text{yes}]_0\}$ . Therefore if there is a family  $\Pi$ , uniform by the pair  $(e, f)$ , that solves  $X$ , then there is another family  $\Pi_{\text{min}}$  that is uniform by  $(e, f_{\text{min}})$  and also solves  $X$  and the range of  $f_{\text{min}}$  is  $\Pi_{\text{min}} = \{\Pi_{\text{pos}}, \Pi_{\text{neg}}\}$ . Both members of  $\Pi_{\text{min}}$  can be easily evaluated in  $\mathbf{AC}^0$ .  $\square$

Next, we prove Theorem 10, which we restate:

**Theorem: (10).** *Let  $\mathbf{E}, \mathbf{F} \in \{\mathbf{AC}^0, \mathbf{NC}^1, \mathbf{L}, \mathbf{NL}, \mathbf{NC}^2, \mathbf{P}, \mathbf{NP}, \mathbf{PSPACE}\}$  and let  $\mathbf{F} \subseteq \mathbf{E}$ . Then  $(\mathbf{E}, \mathbf{F})\text{-PMC}_{\mathcal{AM}_d^0} = \mathbf{E}$ .*

*Proof.* First, we prove the upper-bound  $(\mathbf{E}, \mathbf{F})\text{-PMC}_{\mathcal{AM}_d^0} \subseteq \mathbf{E}$  for the classes  $\mathbf{E}$  and  $\mathbf{F}$  as given in the statement. As we have seen in Lemma 13, each problem in the class  $(\mathbf{E}, \mathbf{F})\text{-PMC}_{\mathcal{AM}_d^0}$  is solved by a family composed only of membrane systems  $\Pi_{\text{pos}}$  and  $\Pi_{\text{neg}}$ . To simulate this family on input  $x$  we first compute  $f_{\text{min}}(1^{|x|})$ , noting that  $f_{\text{min}} \in \mathbf{F} \subseteq \mathbf{E}$ . We then evaluate the resulting membrane system (either  $f_{\text{min}}(1^{|x|}) = \Pi_{\text{pos}}$  or  $f_{\text{min}}(1^{|x|}) = \Pi_{\text{neg}}$ ) on input  $e(x) \in \{a, b\}$ . As observed in Lemma 13, the simplified membrane systems  $\Pi_{\text{pos}}$  and  $\Pi_{\text{neg}}$ , on the simplified input  $e(x) \in \{a, b\}$ , can be simulated in  $\mathbf{AC}^0$ , which is the weakest  $\mathbf{E}$  that we consider.

The lower-bound  $\mathbf{E} \subseteq (\mathbf{E}, \mathbf{F})\text{-PMC}_{\mathcal{AM}_d^0}$  is easy to show. We use the fact, shown above, that  $e$  is a characteristic function with access to the input word. Thus the following simple family computes any problem from  $\mathbf{E}$ : function  $e(x) = \{\mathbf{a}\}$  if  $x \in X$  and  $e(x) = \{\mathbf{b}\}$  if  $x \notin X$ , and  $f_{\text{min}}$  is the constant function  $f_{\text{min}}(n) = \Pi_{\text{pos}}$ .  $\square$

## 4 Conclusion

Quite a number of nature-inspired computing models are specified as infinite families of finite devices. Since we usually want a computing device to be as general as possible, solving a problem with a uniform family is in a sense more satisfactory than with a semi-uniform family (for example see [19]). Nevertheless, for many models, both notions were shown to coincide in the sense that they characterise the same complexity classes. Our main result bucks that trend and shows that uniformity and semi-uniformity are provably distinct notions that can lead to large differences in computing power.

Although proven specifically for membrane systems, our main result should be applicable to families of devices in general. In Appendix A we illustrate this by applying our technique to Boolean OR circuits. The provable difference in the power of uniform and semi-uniform families of OR circuits in Theorem 19 echoes our main result for membrane systems in Theorem 1. However, we note that both uniform and semi-uniform families of active membranes with dissolution ( $\mathcal{AM}_{+d}^0$ ) and also that both uniform and semi-uniform families of standard Boolean circuits characterise the same class, namely  $\mathbf{P}$  [22]. These facts are summarised in Table 1.

### 4.1 Uniformity and semi-uniformity in natural computing

Many nature-inspired models of computation assume that the input is preprocessed into a form that is suitable for the model (e.g. multisets of objects for membrane systems, or DNA polymers for DNA computers). Below, we represent such preprocessing with an encoding function  $\mathcal{E}$  (for example  $e$  from Definition 5 acts as  $\mathcal{E}$  in membrane systems).

Furthermore, a molecular computing device is restricted to solving only problem instances of a certain size or type and this restriction comes from limiting properties

	$\mathbf{AC}^0$ -uniform	$\mathbf{AC}^0$ -semi-uniform
Circuits	$\mathbf{P}$	$\mathbf{P}$
$\mathcal{AM}_{+d}^0$	$\mathbf{P}$	$\mathbf{P}$
OR circuits	$\subseteq \mathbf{AC}^0$	$\mathbf{NL}$
$\mathcal{AM}_{-d}^0$	$\mathbf{AC}^0$	$\mathbf{NL}$

**Table 1.** Computational power of four models of  $\mathbf{AC}^0$ -uniform and  $\mathbf{AC}^0$ -semi-uniform families.  $\mathcal{AM}_{+d}^0$ : active membrane systems with dissolution.  $\mathcal{AM}_{-d}^0$ : without dissolution.

such as molecular species and their concentrations, experimental apparatus type and size, laboratory protocols, and so on. For this reason a molecular algorithm that works on a large, or possibly infinite, set of inputs is naturally specified as a meta-protocol (i.e. an algorithm) that scales concentrations, experimental apparatuses and protocols appropriately for each input instance. This meta-protocol defines a (potentially infinite) family of experimental setups and should be easily computable, in other words, uniform.

Using Definitions 5 and 6 as inspiration we can separate the uniformity meta-protocol into two functions  $\mathcal{E}$  and  $\mathcal{F}$  for arbitrary families of computing devices (biomolecular or otherwise) as follows:

- The function  $\mathcal{E}$  specifies all parts of the device that depend on the input itself.
- The function  $\mathcal{F}$  specifies all parts of the device that are independent of the input, or depend only on input size.

Roughly speaking, if any part of the device (program) is specified by the function  $\mathcal{E}$ , then we say the family is *semi-uniform*. If the device can be specified using only  $\mathcal{F}$ , then we say the family is *uniform*. (For example, the  $h$  functions in Definitions 6 and 15 are examples of  $\mathcal{E}$  functions since they make use of the input word to specify the device.)

With this in mind we can see that many designs for DNA computers are semi-uniform, for example Lipton [20], Ouyang et al. [24], Head et al. [13], and Liu et al. [21]. These systems encode the problem instance (using an  $\mathcal{E}$  function) as a series of steps in a protocol that filters out invalid candidate outputs and produces the output of the computation. Also, there are many examples of uniform families in the literature of natural computing, for example Lagoudakis and LaBean [19], Seelig and Soloveichik [32], Qian and Winfree [31], Doty et al. [10], and Barish et al. [6].

We believe that DNA computers are well-suited to, and will find most applications in, solving problems of low computational complexity (say within  $\mathbf{NL}$ ). In such scenarios it may well be possible to apply our technique and show that the seemingly benign choice between uniformity and semi-uniformity leads to large, and provable, differences in computational power.

## References

1. A. Alhazov, C. Martín-Vide, and L. Pan. Solving a PSPACE-complete problem by recognizing P Systems with restricted active membranes. *Fundamenta Informaticae*, 58(2):67–77, 2003.
2. A. Alhazov and L. Pan. Polarizationless P Systems with active membranes. *Grammars*, 7:141–159, 2004.
3. A. Alhazov and M. J. Pérez-Jiménez. Uniform solution to QSAT using polarizationless active membranes. In J. Durand-Lose and M. Margenstern, editors, *Machines, Computations and Universality (MCU)*, volume 4664 of *LNCS*, pages 122–133, Orléans, France, Sept. 2007. Springer.
4. E. Allender. Applications of time-bounded Kolmogorov complexity in complexity theory. In O. Watanabe, editor, *Kolmogorov Complexity and Computational Complexity*, chapter 1, pages 4–22. Springer, 1992.

5. M. Amos. *Theoretical & Experimental DNA Computation*. Natural Computing Series. Springer, 2005.
6. R. D. Barish, R. Schulman, P. W. K. Rothemund, and E. Winfree. An information-bearing seed for nucleating algorithmic self-assembly. *PNAS*, 106(15):6054–6059, 2009.
7. D. A. M. Barrington, N. Immerman, and H. Straubing. On uniformity within  $NC^1$ . *Journal of Computer and System Sciences*, 41(3):274–306, 1990.
8. A. Borodin. On relating time and space to size and depth. *SIAM Journal on Computing*, 6(4):733–744, 1977.
9. M. Cook, D. Soloveichik, E. Winfree, and J. Bruck. Programmability of chemical reaction networks. In *Algorithmic Bioprocesses*, pages 543–584. Springer, 2009.
10. D. Doty, J. H. Lutz, M. J. Patitz, S. M. Summers, and D. Woods. Intrinsic universality in self-assembly. In *Proceedings of the 27th International Symposium on Theoretical Aspects of Computer Science*, 2010.
11. R. Greenlaw, H. J. Hoover, and W. L. Ruzzo. *Limits to parallel computation: P-completeness Theory*. Oxford University Press, New York, Oxford, 1995.
12. M. A. Gutiérrez-Naranjo, M. J. Pérez-Jiménez, A. Riscos-Núñez, and F. J. Romero-Campero. Computational efficiency of dissolution rules in membrane systems. *International Journal of Computer Mathematics*, 83(7):593–611, 2006.
13. T. Head, G. Rozenberg, R. S. Bladergroen, C. K. D. Breek, P. H. M. Lommerse, and H. P. Spaink. Computing with DNA by operating on plasmids. *Biosystems*, 57(2):87 – 93, 2000.
14. N. Immerman. *Descriptive Complexity*. Springer, 1999.
15. B. Jenner. Knapsack problems for NL. *Information Processing Letters*, 54(3):169–174, 1995.
16. N. D. Jones. Space-bounded reducibility among combinatorial problems. *Journal of Computer and System Sciences*, 11(1):68–85, 1975.
17. N. D. Jones, Y. E. Lien, and W. T. Laaser. New problems complete for nondeterministic log space. *Theory of Computing Systems*, 10(1):1–17, Dec. 1976.
18. R. E. Ladner. The circuit value problem is log space complete for P. *SIGACT News*, 7(1):18–20, 1975.
19. M. Lagoudakis and T. LaBean. 2D DNA Self-Assembly for Satisfiability. In *Proceedings of the 5th DIMACS Workshop on DNA Based Computers*, volume 54, pages 141–154, 1999.
20. R. J. Lipton. DNA solution of hard computational problems. *Science*, 268(5210):542–545, 1995.
21. Q. Liu, L. Wang, A. G. Frutos, A. E. Condon, R. M. Corn, and L. M. Smith. DNA computing on surfaces. *Nature*, 403(6766):175–179, 2000.
22. N. Murphy and D. Woods. Active membrane systems without charges and using only symmetric elementary division characterise P. In *8th International Workshop on Membrane Computing*, volume 4860 of *LNCS*, pages 367–384. Springer, 2007.
23. N. Murphy and D. Woods. A characterisation of NL using membrane systems without charges and dissolution. *Unconventional Computing, 7th International Conference, LNCS*, 5204:164–176, 2008.
24. Q. Ouyang, P. D. Kaplan, S. Liu, and A. Libchaber. DNA solution of the maximal clique problem. *Science*, 278(5337):446–449, 1997.
25. I. Parberry. *Circuit complexity and neural networks*. MIT Press, 1994.
26. M. J. Pérez-Jiménez, A. Romero-Jiménez, and F. Sancho-Caparrini. Complexity classes in models of cellular computing with membranes. *Natural Computing*, 2(3):265–285, 2003.
27. G. Păun. P Systems with active membranes: Attacking NP-Complete problems. *Journal of Automata, Languages and Combinatorics*, 6(1):75–90, 2001.
28. G. Păun. *Membrane Computing*. Springer-Verlag, Berlin, 2002.
29. G. Păun. Membrane computing. In *Fundamentals of computation theory*, volume 2751 of *Lecture Notes in Comput. Sci.*, pages 284–295. Springer, Berlin, 2003.
30. G. Păun. Further twenty six open problems in membrane computing. In *Proceedings of the Third Brainstorming Week on Membrane Computing, Sevilla (Spain)*, pages 249–262. Félix Editoria, 2005.
31. L. Qian and E. Winfree. A simple DNA gate motif for synthesizing large-scale circuits. In *14th International Meeting on DNA Computing*, volume 5347 of *LNCS*, pages 70–89. Springer, 2009.
32. G. Seelig and D. Soloveichik. Time-complexity of multilayered DNA strand displacement circuits. In *15th International Meeting on DNA Computing*, volume 5877, pages 144–153. Springer, 2009.
33. D. Soloveichik and E. Winfree. The computational power of Benenson automata. *Theoretical Computer Science*, 344:279–297, 2005.
34. D. Soloveichik and E. Winfree. Complexity of self-assembled shapes. *SIAM Journal of Computing*, 36(6):1544–1569, 2007.
35. P. Sosík. The computational power of cell division in P systems: Beating down parallel computers? *Natural Computing*, 2(3):287–298, Aug. 2003.
36. P. Sosík and A. Rodríguez-Patón. Membrane computing and complexity theory: A characterization of PSPACE. *Journal of Computer and System Sciences*, 73(1):137–152, 2007.

## A Appendix: Uniform and semi-uniform circuit complexity

The main result of this paper shows a general principle that can be applied to other models. We demonstrate this by showing an analogous result for the complexity of uniform and semi-uniform families of Boolean circuits of OR gates. The proof for circuits is easier than for membrane systems, this is due to the fact that we purposely choose a simple circuit model that exhibits the uniform/semi-uniform gap, as well as the fact that circuits are a relatively well-understood model. It would be interesting to see if such a gap can be shown for more general circuit models.

**Definition 14 (Uniform circuit family [11]).** *A uniform circuit family  $\{\alpha_n\}$  is an infinite collection of circuits, where each circuit has  $n$  input gates and a single output gate, such that there is a function  $f$  (computable within some resource bound) such that  $f(1^n) = \alpha_n$ . We say a uniform family of circuits  $\{\alpha_n\}$  decides a language  $X$  if for each  $x \in \{0, 1\}^n$ , circuit  $\alpha_n$  evaluates to 1 if  $x \in X$  and 0 if  $x \notin X$ .*

The complexity of uniform circuits has been well explored, and it is well-known [11,14] that  $\mathbf{AC}^0$ -uniform circuits (with AND, OR and NOT gates) solve exactly the problems in  $\mathbf{P}$ . We introduce a definition of semi-uniform families of Boolean circuits inspired by Definition 6.

**Definition 15 (Semi-uniform circuit family).** *A semi-uniform circuit family  $\{\alpha_x\}$  is an infinite collection of Boolean circuits, each with constant gates (instead of input gates) and a single output gate, such that there is a function  $h$  (computable within some resource bound) such that  $h(x) = \alpha_x$  and  $x \in \{1, 0\}^*$ . We say a semi-uniform family of circuits  $\{\alpha_x\}$  decides a language  $X$  if for each  $x$ , circuit  $\alpha_x$  evaluates to 1 if  $x \in X$  and 0 if  $x \notin X$ .*

The problem of circuit prediction (known as CVP) captures the computational complexity of simulating and constructing semi-uniform circuits and is  $\mathbf{P}$ -complete [18]. Thus semi-uniform and uniform families of Boolean circuits characterise  $\mathbf{P}$ .

To show a difference in the computational power of uniform and semi-uniform family of circuits (analogous to what we have seen for membrane systems in Section 3) we restrict the Boolean circuits by prohibiting AND and NOT gates, we refer to these as “OR circuits”.

The problem of predicting an OR circuit is defined as follows:

*Problem 16 (OR Circuit Value Problem (ORCVP)).*

INSTANCE: A Boolean circuit  $\alpha$  (using only disjunctive logic, i.e. no AND nor NOT gates), inputs  $x_1, \dots, x_n$ , and a designated gate  $y$ .

QUESTION: Is the value of gate  $y$  in  $\alpha$  true on input  $x_1, \dots, x_n$ ?

The problem ORCVP is  $\mathbf{NL}$ -complete. We quickly observe that  $\text{ORCVP} \in \mathbf{NL}$ : a path from the output gate to an input gate that has been assigned 1 is non-deterministically chosen, if the path is valid then accept, else reject.

The canonical  $\mathbf{NL}$ -complete problem is the graph reachability problem “ $s$ - $t$ -connectivity”, or STCON [16]. An instance is a directed graph and two vertices  $s, t$  and we ask whether there is a directed path from  $s$  to  $t$ . We can reduce an STCON instance to an ORCVP instance in  $\mathbf{FAC}^0$  as follows. Each vertex in the graph becomes an OR gate in the circuit. Each directed edge in the graph becomes a wire in the circuit. Add a wire linking a constant gate with value 1 to the gate representing the special vertex  $s$ . Let the gate representing the special vertex  $t$  be the circuit’s output gate.

If there is a path from  $s$  to  $t$  in the graph then the constant gate with value 1 causes each successive OR gate to have value 1, including the output gate. If there is no path

from  $s$  to  $t$ , then the 1 input never propagates to the output gate. Thus the value of the gate  $y$  is 1 iff there is a directed path from  $s$  to  $t$  in the graph.

Now we are ready to compare the power of uniform and semi-uniform families of OR circuits.

**Lemma 17.**  $\mathbf{AC}^0$ -semi-uniform families of OR circuits characterise  $\mathbf{NL}$ .

*Proof.* We use the  $\mathbf{AC}^0$  reduction provided above for  $\mathbf{STCON} \leq \mathbf{ORCVP}$  to define a semi-uniform family of circuits which recognises instances of  $\mathbf{STCON}$  by simply using the reduction as the function  $h$  in Definition 15. Each circuit in the family outputs 1 if  $x \in \mathbf{STCON}$  and outputs 0 if  $x \notin \mathbf{STCON}$ . Thus semi-uniform families of OR circuits solve  $\mathbf{NL}$ -complete problems.

Each member of a semi-uniform OR family is an instance of  $\mathbf{ORCVP}$  and thus can be evaluated in non-deterministic logspace.  $\square$

Now we compare this with a characterisation for uniform families of OR circuits.

**Lemma 18.**  $\mathbf{AC}^0$ -uniform families of OR circuits are upper-bounded by  $\mathbf{AC}^0$ .

*Proof.* We show that each function that is computable by an  $\mathbf{AC}^0$ -uniform OR circuit family is also computable in  $\mathbf{AC}^0$ . Let circuit  $\alpha_{|x|} = f(1^{|x|})$  be the  $|x|^{th}$  member of a uniform (by function  $f$ ) family of OR circuits, with input gates  $\{x_1, \dots, x_{|x|}\}$  and a single output gate  $y$ . Note that each gate in circuit  $\alpha_{|x|}$  is in one of the two disjoint sets  $G_o$ , gates that are on a path to the output gate, or  $G_d$ , gates that are not. Now consider an  $\mathbf{AC}^0$ -uniform OR circuit family<sup>4</sup>In  $\alpha'_{|x|}$ , input gate  $x'_i$  is connected to gate  $g'_o$  iff  $x_i$  is in  $G_o$  of  $\alpha_{|x|}$ . Clearly,  $\alpha'_{|x|}$  accepts  $x$  iff  $\alpha_{|x|}$  accepts  $x$ .

A uniform circuit family of depth 1 is trivially computable in  $\mathbf{AC}^0$ , so if the uniformity function  $f$  (see Definition 14) is computable in uniform  $\mathbf{FAC}^0$  then the problem decided by the family is in uniform  $\mathbf{AC}^0$ .  $\square$

Combining Lemmas 17 and 18 we have:

**Theorem 19.** *The set of problems solved by  $\mathbf{AC}^0$ -uniform families of OR only circuits  $\subseteq \mathbf{AC}^0 \subsetneq \mathbf{NL} =$  the set of problems solved by  $\mathbf{AC}^0$ -semi-uniform families of OR only circuits.*

We remark that for semi-uniform circuits, If the function  $h$  is more difficult to compute than the problem of circuit prediction then the function  $h$  defines the complexity of the family. (For example, if  $h$  is  $\mathbf{PSPACE}$  computable then we can reduce a  $\mathbf{PSPACE}$ -complete problem to  $\mathbf{ORCVP}$  giving an “artificially” huge amount of power to the family.) On the other hand, if a uniform family of circuits requires more computing resources to construct a circuit than are needed to evaluate that circuit it is unknown how this affects the set of problems solved by the family. For example, we do not know if  $\mathbf{P}$ -uniform constant depth unbounded fan-in Boolean circuits can solve problems outside of  $\mathbf{DLOGTIME}$ -uniform  $\mathbf{AC}^0$  [4].

<sup>4</sup> To assert uniformity for this replacement of many OR gates by two we should give an  $\mathbf{FAC}^0$  function (algorithm) but we omit the details.