A characterisation of NL using membrane systems without charges and dissolution

Niall Murphy[1] and Damien Woods[2]

[1] Department of Computer Science, National University of Ireland, Maynooth, Ireland
nmurphy@cs.nuim.ie
[2] Department of Computer Science and Artificial Intelligence, University of Seville, Seville, Spain
d.woods@cs.ucc.ie

Abstract. We apply techniques from complexity theory to a model of biological cellular membranes known as membrane systems or P-systems. Like circuits, membrane systems are defined as uniform families. To date, polynomial time uniformity has been the accepted uniformity notion for membrane systems. Here, we introduce the idea of using $\mathbf{AC}^0$ and **L**-uniformities and investigate the computational power of membrane systems under these tighter conditions. It turns out that the computational power of some systems is lowered from **P** to **NL**, so it seems that our tighter uniformities are more reasonable for these systems. Interestingly, other systems that are known to be lower bounded by **P** are shown to retain their computational power under the new uniformity conditions. Similarly, a number of membrane systems that are lower bounded by **PSPACE** retain their power under the new uniformity conditions.

1 Introduction

Membrane systems [14] are a model of computation inspired by living cells. In this paper we explore the computational power of cell division (mitosis) and dissolution (apoptosis) by investigating a variant of the model called active membranes [13]. An instance of the model consists of a number of (possibly nested) membranes, or compartments, which themselves contain objects. During a computation, the objects, depending on the compartment they are in, become other objects or pass through membranes. In the active membrane model it is also possible for a membrane to completely dissolve, and for a membrane to divide into two child membranes.

This membrane model can be regarded as a model of parallel computation, however it has a number of features that make it somewhat unusual when compared to other parallel models. For example, object interactions are nondeterministic so confluence plays an important role, membranes contain multisets of objects, there are many parameters to the model, etc. In order to clearly see the power of the model we analyse it from the computational complexity point of view, the goal being to characterise the model in terms of the set of problems

that it can solve in reasonable time. One can also interpret our results as classifying the computational complexity of simulating biological phenomena that are modelled by the membrane systems under consideration.

Another, more specific, motivation is the so-called **P**-conjecture [15] which states that recogniser membranes systems with division rules (active membranes), but without charges, characterise **P**. On the one hand, it was shown that this conjecture does not hold for systems with non-elementary division as **PSPACE** upper [18] and lower [1] bounds were found for this variant (non-elementary division is where a membrane containing multiple membranes and objects may be copied in a single timestep). On the other hand, the **P**-conjecture was thought to hold for all active membrane systems without dissolution rules, when Gutiérrez-Naranjo et al. [7] gave a **P** upper bound. The corresponding **P** lower bound (trivially) came from the fact that the model is defined to be **P**-uniform.

However, here we argue that the aforementioned **P** lower bound highlights a problem with using **P**-uniformity, as it does not tell us whether this membrane model itself has (in some sense) the ability to solve all of **P** in polynomial time, or if the uniformity condition is providing the power. In this paper we show that when we use weaker, and more reasonable, uniformity conditions the model does not in fact have the ability to solve all problems in **P** (assuming $\mathbf{P} \neq \mathbf{NL}$). We find that with either $\mathbf{AC}^0$ or **L**-uniformity the model characterises **NL** in the semi-uniform case, and we give an **NL** upper bound for the uniform case. We also show that the **PSPACE** lower and upper bounds mentioned above still hold under these restricted uniformity conditions.

Using the notation of membrane systems (defined in Section 2) our upper bound on **L**-uniform and **L**-semi-uniform families of membrane systems can be stated as follows.

Theorem 1. $\mathbf{PMC}_{\mathcal{AM}^0_{-d}} \subseteq \mathbf{NL}$

Essentially this theorem states that polynomial time active membrane systems, without dissolution rules, solve no more than those problems in **NL**. Despite the fact that these systems run for polynomial time (and can even create exponentially many objects and membranes), they can not solve all of **P** (assuming $\mathbf{NL} \neq \mathbf{P}$). This result is illustrated by the bottom four nodes in Figure 1.

The upper bound in Theorem 1 is found by showing that the construction in [7] can be reduced to an instance of the **NL**-complete problem s-t-connectivity (STCON). The full proof appears in Section 3. Next we give a corresponding lower bound.

Theorem 2. $\mathbf{NL} \subseteq \mathbf{PMC}_{\mathcal{AM}^0_{-d,-u}}$

To show this lower bound we provide an $\mathbf{AC}^0$-semi-uniform membrane family that solves STCON. The full proof is in Section 4 and the result is illustrated by the bottom left two nodes in Figure 1. Therefore, in the semi-uniform case we have a characterisation of **NL**.

Corollary 1. $\mathbf{NL} = \mathbf{PMC}_{\mathcal{AM}^0_{-d,-u}}$

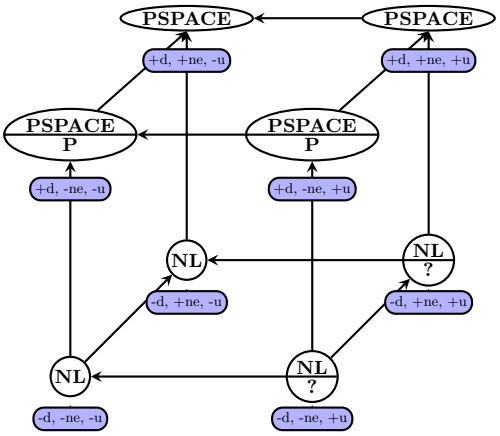


Fig. 1. A diagram showing the currently known upper and lower bounds on the variations of the model. The top part of a node represents the best known upper bounds, and the lower part the best known lower bounds. An undivided node represents a characterisation. Arrows represent inclusions.

We have not yet shown an analogous lower bound result for uniform families. In Section 4.1 we briefly explore some issues relating to this problem.

So far we have shown that four models, that characterise **P** when polynomial time uniformity is used are actually upper bounded by **NL** when restricted to be $\mathbf{AC}^0$-uniform (or **L**-uniform). Interestingly, we also show that two other polynomial time uniform membrane systems that are known [11] to characterise **P** actually retain this **P** characterisation when restricted to be $\mathbf{AC}^0$-uniform (or **L**-uniform). This result is stated as a **P** lower bound on membrane systems with dissolution:

Theorem 3. $\mathbf{P} \subseteq \mathbf{PMC}_{\mathcal{AM}^0_{+d,+u}}$

The proof appears in Section 5 and is illustrated by the top front two nodes in Figure 1.

In Section 2.4 we observe that the known **PSPACE** characterisations (top two nodes in Figure 1) remain unchanged under $\mathbf{AC}^0$-uniformity conditions.

2 Membrane Systems

In this section we define membrane systems and complexity classes. These definitions are from Păun [13, 14], and Sosík and Rodríguez-Patón [18]. We also introduce the notion of $\mathbf{AC}^0$-uniformity for membrane systems.

2.1 Active membrane systems

Active membranes systems are a class of membrane systems with membrane division rules. Division rules can either only act on elementary membranes, or else

on both elementary and non-elementary membranes. An elementary membrane is one which does not contain other membranes (a leaf node, in tree terminology).

Definition 1. *An active membrane system without charges is a tuple* $\Pi = (O, H, \mu, w_1, \ldots, w_m, R)$ *where,*

1. $m > 1$ *is the initial number of membranes;*
2. O *is the alphabet of objects;*
3. H *is the finite set of labels for the membranes;*
4. μ *is a membrane structure, consisting of* m *membranes, labelled with elements of* H*;*
5. $w_1, \ldots, w_m$ *are strings over* O*, describing the multisets of objects placed in the* m *regions of* μ*.*
6. R *is a finite set of developmental rules, of the following forms:*
   (a) $[\ a \ \rightarrow \ u\ ]_h$, *for* $h \in H,\ a \in O,\ u \in O^*$
   (b) $a[\ ]_h \rightarrow [\ b\ ]_h$, *for* $h \in H,\ a, b \in O$
   (c) $[\ a\ ]_h \rightarrow [\ ]_h\ b$, *for* $h \in H,\ a, b \in O$
   (d) $[\ a\ ]_h \rightarrow b$, *for* $h \in H,\ a, b \in O$
   (e) $[\ a\ ]_h \rightarrow [\ b\ ]_h\ [\ c\ ]_h$, *for* $h \in H,\ a, b, c \in O$.
   (f) $[\ a\ [\ ]_{h_1}\ [\ ]_{h_2}\ [\ ]_{h_3}\ ]_{h_0} \rightarrow [\ b\ [\ ]_{h_1}\ [\ ]_{h_3}]_{h_0}\ [\ c\ [\ ]_{h_2}\ [\ ]_{h_3}]_{h_0}$, *for* $h_0, h_1, h_2, h_3 \in H,\ a, b, c \in O$.

These rules are applied according to the following principles:

- All the rules are applied in maximally parallel manner. That is, in one step, one object of a membrane is used by at most one rule (chosen in a non-deterministic way), but any object which can evolve by one rule of any form, must evolve.
- If at the same time a membrane labelled with h is divided by a rule of type (e) or (f) and there are objects in this membrane which evolve by means of rules of type (a), then we suppose that first the evolution rules of type (a) are used, and then the division is produced. This process takes only one step.
- The rules associated with membranes labelled with h are used for membranes with that label. At one step, a membrane can be the subject of only one rule of types (b)-(f).

The environment is an indissoluble membrane that is the ultimate parent of all other membranes in the system.

2.2 Recogniser membrane systems

In this paper we study the language recognising variant of membrane systems that solves decision problems.

Definition 2. *A* recogniser membrane system *is a membrane system such that the result of the computation (a solution to the instance) is "yes" if a distinguished object* `yes` *appears in the environment or "no" if* `no` *appears.*

Such a membrane system is called *deterministic* if for each input a unique sequence of configurations exists. A membrane system is called *confluent* if it always halts and, starting from the same initial configuration, it always gives the same result, either always "yes" or always "no". Therefore, the following interpretation holds: given a fixed initial configuration, a confluent membrane system non-deterministically chooses one from a number of valid configuration sequences, but all of them must lead to the same result.

2.3 Complexity classes

Here we introduce the notion of $\mathbf{AC}^0$-uniformity to membrane systems.

Previous work on the computational complexity of membrane systems used (Turing machine) polynomial time uniformity [16]. Consider a decision problem X, i.e. a set of instances $X = \{x_1, x_2, \ldots\}$ over some finite alphabet such that to each x_i there is an unique answer "yes" or "no". We say that a *family* of membrane systems solves a decision problem if each instance of the problem is solved by some family member. We denote by $|x| = n$ the length of any instance $x \in X$. $\mathbf{AC}^0$ circuits are **DLOGTIME**-uniform, polynomial sized (in input length n), constant depth, circuits with AND, OR, and NOT gates, and unbounded fanin [4].

Definition 3 ($\mathbf{AC}^0$-uniform families of membrane systems). *Let $\mathcal{D}$ be a class of membrane systems and let $f : \mathbb{N} \to \mathbb{N}$ be a total function. The class of problems solved by uniform families of membrane systems of type $\mathcal{D}$ in time f, denoted by $\mathbf{MC}_{\mathcal{D}}(f)$, contains all problems X such that:*

- *There exists an* $\mathbf{AC}^0$-uniform *family of membrane systems, $\Pi_X = (\Pi_X(1), \Pi_X(2), \ldots)$ of type $\mathcal{D}$: that is, there exists an $\mathbf{AC}^0$ circuit family such that on unary input 1^n the n^{th} member of the circuit family constructs $\Pi_X(n)$. We refer to this circuit family as the* family machine.
- *There exists an $\mathbf{AC}^0$-uniform circuit family such that on input $x \in X$, of length $|x| = n$, the n^{th} member of the family encodes x as a multiset of input objects placed in the distinct input membrane h_{in}. We refer to this circuit family as the* input encoding machine.
- *Each $\Pi_X(n)$ is* sound*: $\Pi_X(n)$ starting with an encoded input x of length n expels out a distinguished object* `yes` *if and only if the answer to x is "yes".*
- *Each $\Pi_X(n)$ is* confluent*: all computations of $\Pi_X(n)$ with the same input x of size n give the same result; either always "yes" or else always "no".*

– *Π_X is f-efficient: $\Pi_X(n)$ always halts in at most $f(n)$ steps.*

Using this definition of $\mathbf{AC}^0$-uniform families, we define $\mathbf{AC}^0$-*semi-uniform families of membrane systems* $\Pi_X = (\Pi_X(x_1); \Pi_X(x_2); \ldots)$ such that there exists an $\mathbf{AC}^0$-uniform circuit family which, on an input $x \in X$ of length $|x| = n$, constructs membrane system $\Pi_X(x)$. Here a single circuit family (which we refer to as the *input encoding machine*) is used to construct the semi-uniform membrane family, and so the problem instance is encoded using objects, membranes, and rules. In this case, for each instance of X we have a special membrane system which therefore does not need a separately constructed input. The resulting class of problems is denoted by $\mathbf{MC}_{\mathcal{D},-u}(f)$. Obviously, $\mathbf{MC}_{\mathcal{D}}(f) \subseteq \mathbf{MC}_{\mathcal{D},-u}(f)$ for a given class $\mathcal{D}$ and a complexity [3] function f.

Logspace, or $\mathbf{L}$, uniform families of membrane systems are defined analogously, where we use two deterministic logspace Turing machines, instead of the two $\mathbf{AC}^0$ circuit families, for the uniformity conditions. Similarly we define $\mathbf{L}$-semi-uniformity using a logspace Turing machine instead of an $\mathbf{AC}^0$ circuit family.

We define $\mathbf{PMC}_{\mathcal{D}}$ and $\mathbf{PMC}_{\mathcal{D},-u}$ as

$$\mathbf{PMC}_{\mathcal{D}} = \bigcup_{k\in\mathbb{N}} \mathbf{MC}_{\mathcal{D}}(O(n^k)),\ \mathbf{PMC}_{\mathcal{D},-u} = \bigcup_{k\in\mathbb{N}} \mathbf{MC}_{\mathcal{D},-u}(O(n^k)).$$

In other words, $\mathbf{PMC}_{\mathcal{D}}$ (and $\mathbf{PMC}_{\mathcal{D},-u}$) is the class of problems solvable by uniform (respectively semi-uniform) families of membrane systems in polynomial time. We denote by $\mathcal{AM}^0$ the classes of membrane systems with active membranes and no charges. We denote by $\mathcal{AM}^0_{-ne}$ the classes of membrane systems with active membranes and only elementary membrane division and no charges. We denote by $\mathcal{AM}^0_{+ne}$ the classes of membrane systems with active membranes, and both non-elementary and elementary membrane division and no charges. We denote by $\mathbf{PMC}_{\mathcal{AM}^0_{-d}}$ the classes of problems solvable by uniform families of membrane systems in polynomial time with no charges and no dissolution rules.

In this paper we are using **DLOGTIME-$\mathbf{AC}^0$**-uniformity which can be somewhat cumbersome to analyse, therefore in our proofs we use an $\mathbf{AC}^0$ equivalent model called the constant time Concurrent Random Access Machine (constant time CRAM) [2, 8].

Definition 4 (CRAM [8]). *A CRAM is a concurrent-read concurrent write PRAM with a polynomial number of processors. Each processor is able to shift a word in memory by a polynomial number of bits.*

2.4 $\mathbf{AC}^0$-uniformity and PSPACE results

Membrane systems with active membranes, without charges, and using non-elementary division have been shown to characterise **PSPACE** [1, 18]. For the lower bound, a $\mathbf{P}$-uniform membrane system is given [1] that solves instances of QSAT in polynomial time. Clearly, stricter uniformity notions have no affect on

the **PSPACE** upper bound. We now show that the use of $\mathbf{AC}^0$-uniformity does not change this lower bound.

The family machine inputs the numbers n and m representing the number of variables and clauses of the QSAT instance, and uses them to construct a polynomial number of objects, rules and membranes. We observe that the construction in [1] is in $\mathbf{AC}^0$: the most complicated aspect involves multiplication by constants (essentially addition) which is known [9] to be in $\mathbf{AC}^0$. Although we omit the details, it is not difficult to see that a constant time CRAM constructs the membrane system in constant time from n and m. Similarly, the encoding of the instance as objects to be placed in the input membrane involves only addition.

3 NL upper bound on active membranes without dissolution rules

Previously the upper bound on all active membrane systems without dissolution was **P** [7]. As an aside, we remark that this is a very enlightening proof since it first highlighted the importance of dissolution. Without dissolution, membrane division, even non-elementary division, can be modelled as a special case of object evolution. It is also worth noting that these systems can create exponential numbers of objects and membranes, yet they can not compute anything outside **P**.

Since membrane systems are usually **P**-uniform, this **P** upper bound was considered a characterisation of **P**. However, having a lower bound of the same power as the uniformity condition is somewhat unsatisfactory, as it tells us little about the computing power of the actual membrane system itself. This is because the input encoding machine (in the uniform and semi-uniform case) takes an instance of the problem as input, thus if the problem is contained in the set of problems solvable by the encoder it simply outputs a *yes* or *no* object directly. In this section we show that if we tighten the uniformity condition to be $\mathbf{AC}^0$, or even **L**, it is possible to decide in **NL** whether or not the system accepts. We give an overview rather than the full details.

The proof of the **P** upper bound in [7] involves the construction of a *dependency graph* representing all possible computation paths of a membrane system on an input. The dependency graph for a membrane system Π is a directed graph $G_\Pi = (V_\Pi, E_\Pi)$. Each vertex a in the graph is a pair $a = (v, h) \in \Gamma \times H$, where Γ is the set of objects and H is the set of membrane labels. An edge connects vertex a to vertex b if there is an evolution rule such that the left hand side of the rule has the same object-membrane pair as a and the right has an object-membrane pair matching b.

If we can trace a path from the vertex $(\texttt{yes}, \text{env})$ (indicating an accepting computation) back to a node representing the input it is clear that this system must be an accepting one. It is worth noting that, unlike upper bound proofs for a number of other computational models, the dependency graph does not

model entire configuration sequences, but rather models only those membranes and objects that lead to a `yes` output.

The original statement of the proof constructed the graph in polynomial time and a path was found from the accepting node to the start node in polynomial time. We make the observation that the graph G_Π can be constructed in deterministic logspace. We omit the details, but our claim can be verified by checking that the construction in [7] can easily be computed using only a fixed number of binary counters. Also we note that the problem of finding a path from the accepting vertex to one of the input vertices is actually an instance of MSTCON, a variation of the **NL**-complete problem STCON. STCON is also known as PATH [17] and REACHABILITY [12].

Definition 5 (STCON). *Given a directed graph $G = (V, E)$ and vertices $s, t \in V$, is there a directed path in G from s to t?*

Definition 6 (MSTCON). *Given a directed graph $G = (V, E)$, vertex $t \in V$ and $S \subseteq V$, is there a directed path in G from any element of S to t?*

MSTCON is **NL**-complete as a logspace machine, or $\mathbf{AC}^0$ circuit can add a new start vertex s', with edges from s' to each vertex in S, to give an instance of STCON.

Since we have shown that the problem of simulating a membrane system without charges and without dissolution can be encoded as an **NL**-complete problem we have proved Theorem 1. The proof holds for both $\mathbf{AC}^0$ and **L**-uniformity, as well as for both uniform and semi-uniform families of membrane systems without dissolution.

4 NL lower bound for semi-uniform active membranes without dissolution

Here we provide a proof of Theorem 2 by giving a membrane system that solves STCON in a semi-uniform manner.

The algorithm works by representing edges in the problem instance graph as object evolution rules. There is only one membrane which serves as the input and output membrane. The system is initialised with an s object in this membrane. If there are edges from s to any other nodes in the graph then have evolution rules indicating this. For example edges $(s, b), (s, c), (s, d)$ are represented as the rule $[s \rightarrow bcd]$. In this manner the presence of an object in a configuration indicates that the system is currently at this node while following (or simulating) each different path through the graph in parallel. If the t object is ever evolved the system outputs a `yes` object and halts. Otherwise, a `no` object is output from the system.

We now give a proof of Theorem 2.

Proof. Each instance of the problem STCON is of the form $((V, E)\, s, t)$. We let n and m be the number of vertices and edges in the graph respectively. We

assume an ordering on instances (say by n and then lexicographically). We define a function $f(k)$, computable in $\mathbf{AC}^0$, that maps the k^{th} instance to the following membrane system Π_k.

- The set of labels is $\{h\}$,
- The initial membrane structure is $[\ ]_h$.
- The working objects $\{\ \texttt{yes},\ \texttt{no}\} \cup \{c_i \mid 0 \leq i \leq |V|+2\} \cup V$.
- The initial multiset is $\left\{c_{|V|+2}, s\right\}$.

In the input membrane we place the object node given by s.

The evolution rules are as follows. If vertex v_i has out degree $d \in \mathbb{N}$ and we have d edges $\{(v_i, v_{j1}), (v_i, v_{j2}), \ldots, (v_i, v_{jd})\}$ then we encode it as a type (a) rule

$$[\ v_i \rightarrow u_i\ ]_h \text{ where } u_i = v_{j1}, v_{j2}, \ldots, v_{jd}.$$

When the object t is evolved we want it to become a `yes` object and send it out to the environment.

$$[\ t\ ]_h \rightarrow [\ ]_h\, \texttt{yes}$$

We also have a counter that counts down in parallel with the above steps.

$$[\ c_i\ \rightarrow c_{i-1}\ ]_h \text{ where } i \in \{1, 2, \ldots, |V|+2\}$$

If we output a `yes`, this occurs on or before timestep $2n$. Therefore, when the counter reaches zero, there must not have been a `yes` object, so we output a `no` to the environment.

$$[\ c_0\ ]_h \rightarrow [\ ]_h\, \texttt{no}$$

This family of membrane systems is easily constructed by a logspace Turing machine. However, if we wish to use $\mathbf{AC}^0$-uniformity we need to insist on a limited out-degree d on all nodes. We can make this restriction without loss of generality. A CRAM to construct the above family for this restricted version of STCON will run in $d+1$ time steps. Each processor of the CRAM works with one edge of the graph. There is a register assigned for each node in the graph. Each processor writes the source node of its edge to the matching register, this will be the left hand side of the rule. The processor will continue to write to this same register in the following timesteps. In the next d time steps the processor tries to write its destination node to this register. If the register is being used by another processor, it waits and tries to write again the next time step. Once it writes its node successfully it stops. The CRAM then outputs the contents of the registers which are the membrane rules of the system.

Note that we encode the edges of the graph as rules, rather than objects. In the membrane computing framework, for uniform membrane systems, inputs must be specified (encoded) as objects. Therefore our algorithm is semi-uniform as we require a different membrane system for each unique problem instance. □

4.1 Differences between circuit and membrane uniformity

To date we have no lower bound for uniform families of active membrane systems without dissolution. Our search for such a lower bound has highlighted some interesting differences between circuit and membrane uniformity.

In circuit complexity we assume a reasonable binary encoding of the input to the circuit so we only need to consider bounding the complexity of the family machine which constructs the circuit family. However with uniform families of active membrane systems we construct our input multiset with an input encoding machine. The family machine that constructs the membrane system $\Pi(n)$ takes a unary number n as input, where n is input length, similar to circuit uniformity. However the input encoding machine takes the actual input instance, this potentially allows it to solve the problem.

For example, consider the following membrane system. Its family machine is **DLOGTIME-AC**^{0} but the input encoding machine is **NC**^{1}. The input encoding machine processes the input in such a way that it becomes trivial to solve the problem PARITY.

PARITY is the problem of telling whether the number of 1 symbols in the input word is odd. This problem is known [5] to be outside of **AC**^{0}, and so **AC**^{0} would be a reasonable uniformity condition in this case.

Our family machine takes as input $n \in \mathbb{N}$ and constructs a set of objects $\{\texttt{odd}_{1^i0^j}, \texttt{even}_{1^i0^j} \mid i, j \geq 0 \text{ such that } i + j = n\}$. Objects `yes` and `no` are also created. A type (a) rule is created mapping every `odd` object with i "1" symbols to the `even` object with $i-1$ "1" symbols in it. A type (a) rule is created mapping every `even` object with i "1" symbols to the `odd` object with $i-1$ "1" symbols in it. A rule is created from object $\texttt{odd}_{00\ldots0}$ to `yes` and from $\texttt{even}_{00\ldots0}$ to `no`.

The **NC**^{1}-input encoding machine rearranges the input word w by moving all 1 symbols to the left and all 0 symbols to the right, to give w'. Then the symbol $\texttt{even}_{w'}$ is placed in the input membrane. (Note, the complexity of this problem has been previously analysed [2]).

As the system runs, the initial object evolves alternately between `odd` and `even` until only 0 symbols are left in the subscript, then a `yes` (or `no`) is evolved indicating the input word contained an odd (or even) number of 1 symbols.

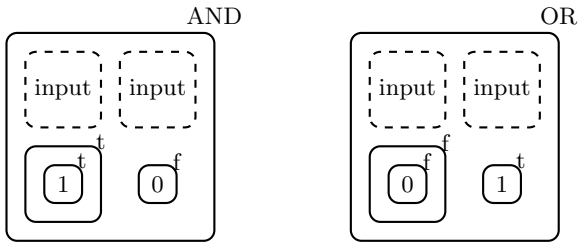
It is possible to decide the parity of such preprocessed binary strings with an **AC**^{0} circuit. This indicates that our preprocessing step (the input encoding machine) was too powerful. Also, it can be noted that for circuits it is open whether or not **P**-uniform **AC**^{0} = **DLOGTIME-AC**^{0}, an analogous statement does not hold for membrane systems. Essentially the use of a **P**-uniform input encoding machine allows the system to solve at least the problems in **P**.

5 P lower bound on uniform families of active membrane systems with dissolving rules

So far we have seen that by tightening the uniformity condition from **P** to **AC**^{0} we lower the power of some models from **P** down to **NL** (see Figure 1). In this

section we show that this does not happen for all models with at least **P** power. More precisely, we prove Theorem 3 by showing that $\mathbf{AC}^0$-uniform, polynomial time, membrane systems with dissolution are lower bounded by **P**. Naturally this result also holds for the semi-uniform case.

Proof. A constant time CRAM encodes an instance of the Circuit Value problem (CVP) [10] as a $\mathbf{PMC}_{\mathcal{AM}^0_{+d,+u}}$ membrane system using the gadget membranes and rules shown in Figure 2. The figure shows AND and OR gadgets: a NOT gadget can be made with the rules $[\ T\ ]_{\text{NOT}} \to [\ ]_{\text{NOT}}F$, $[\ F\ ]_{\text{NOT}} \to [\ ]_{\text{NOT}}T$. The resulting membrane system directly solves the instance of CVP in polynomial time.



$T[\ ]_t \to [\ T\ ]_t$
$[\ T\ ]_t \to \lambda$
$F[\ ]_f \to [F]_f$
$[\ F\ ]_f \to \lambda$
$[\ 1\ ]_{\text{AND}} \to [\ ]_{\text{AND}}T$
$[\ 0\ ]_{\text{AND}} \to [\ ]_{\text{AND}}F$

$F[\ ]_f \to [\ F\ ]_f$
$[\ F\ ]_f \to \lambda$
$T[\ ]_t \to [\ T\ ]_t$
$[\ T\ ]_t \to \lambda$
$[\ 0\ ]_{\text{OR}} \to [\ ]_{\text{OR}}F$
$[\ 1\ ]_{\text{OR}} \to [\ ]_{\text{OR}}T$

Fig. 2. AND and OR gadgets which can be nested together to simulate a circuit. Here "input" is either T, F, or a nested gadget membrane.

To ensure uniformity we have an input membrane (inside the skin membrane) where the initial input assignments for each variable are placed. For example if input gate i is true and input gate j is false we would have input objects T_i and F_j in the input membrane. When the computation starts the truth assignments descend into the encoded circuit until they reach their appropriate "input gate" gadget where they start the computation. We simulate multiple fanouts by outputting multiple copies of the resulting truth value of each gate. We also give each gadget a unique label and the output of each gate would be tagged. The output of a gate moves up through the layers of the membrane system until it reaches the correct gate according to its tag. □

6 Future directions

We have introduced $\mathbf{AC}^0$ uniform active membrane systems and shown an **NL** characterisation of semi-uniform systems without dissolution, this is an improve-

ment over the previous **P** upper bound. Interestingly some existing **P** [11] and **PSPACE** [1, 18] characterisations remain unchanged under the tighter uniformity conditions. This is the first characterisation of an active membrane system that is not either **P** or **PSPACE**. This raises the possibility that other variants may characterise other complexity classes such as **NP** or the arguably more realistic **NC** hierarchy [6].

We have yet to show a lower bound for uniform active membranes without dissolution. Perhaps there is a way to further tighten the upper bound, this would be the first gap between the computing power of the uniform and semi-uniform versions of an active membrane model.

In Section 4.1 we briefly explore the possibility of having different uniformity conditions and encoding conditions.

Acknowledgements

Niall Murphy is funded by the Irish Research Council for Science, Engineering and Technology. Damien Woods is supported by Science Foundation Ireland grant 04/IN3/1524 and Junta de Andalucía grant TIC-581. We would like to thank Mario J. Pérez-Jiménez and Agustín Riscos-Núñez and the other members of the Research Group on Natural Computing in Seville for interesting discussions and for spotting an ambiguity in an earlier version of our uniformity definition.

References

1. 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.
2. E. Allender and V. Gore. On strong separations from AC^0. *DIMACS Series in Discrete Mathematics and Theoretical Computer Science*, 13:21–37, 1993.
3. J. L. Balcázar, J. Diaz, and J. Gabarró. *Structural complexity I.* Springer-Verlag New York, Inc., New York, NY, USA, 2nd edition, 1988.
4. 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.
5. M. L. Furst, J. B. Saxe, and M. Sipser. Parity, circuits and the polynomial-time hierarchy. *Theory of Computing Systems (formerly Mathematical Systems Theory)*, 17(1):13–27, 1984.
6. R. Greenlaw, H. J. Hoover, and W. L. Ruzzo. *Limits to parallel computation:P-completeness Theory.* Oxford University Press, New York, Oxford, 1995.
7. 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.
8. N. Immerman. Expressibility and parallel complexity. *SIAM Journal on Computing*, 18(3):625–638, 1989.

9. R. M. Karp and V. Ramachandran. Parallel algorithms for shared memory machines. In J. van Leeuwen, editor, *Handbook of Theoretical Computer Science*, volume A, chapter 17, pages 869–941. Elsevier, Amsterdam, 1990.
10. R. E. Ladner. The circuit value problem is log space complete for P. *SIGACT News*, 7(1):18–20, 1975.
11. N. Murphy and D. Woods. Active membrane systems without charges and using only symmetric elementary division characterise P. In *Membrane Computing, 8th International Workshop, WMC 2007 Thessaloniki, Greece, Revised Papers*, volume 4860 of *LNCS*, pages 367–384. Springer-Verlag, 2007.
12. C. H. Papadimitriou. *Computational Complexity*. Addison Wesley, 1993.
13. G. Păun. P Systems with active membranes: Attacking NP-Complete problems. *Journal of Automata, Languages and Combinatorics*, 6(1):75–90, 2001. and CDMTCS TR 102, Univ. of Auckland, 1999 (www.cs. auckland.ac.nz/CDMTCS).
14. G. Păun. *Membrane Computing. An Introduction*. Springer-Verlag, Berlin, 2002.
15. G. Păun. Further twenty six open problems in membrane computing. In *Proceedings of the Third Brainstorming Week on Membrane Computing, Sevilla (Spain), January 31st - February 4th*, pages 249–262, 2005.
16. 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.
17. M. Sipser. *Introduction to the Theory of Computation*. PWS Publishing Company, 1996.
18. 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.