Simulating EXPSPACE Turing machines using P systems with active membranes

Artiom Alhazov[1,2], Alberto Leporati[1], Giancarlo Mauri[1], Antonio E. Porreca[1], and Claudio Zandron[1]

[1] Dipartimento di Informatica, Sistemistica e Comunicazione
Università degli Studi di Milano-Bicocca
Viale Sarca 336/14, 20126 Milano, Italy
artiom.alhazov@unimib.it
{leporati,mauri,porreca,zandron}@disco.unimib.it
[2] Institute of Mathematics and Computer Science
Academy of Sciences of Moldova
Academiei 5, Chişinău MD-2028 Moldova
artiom@math.md

1 Introduction

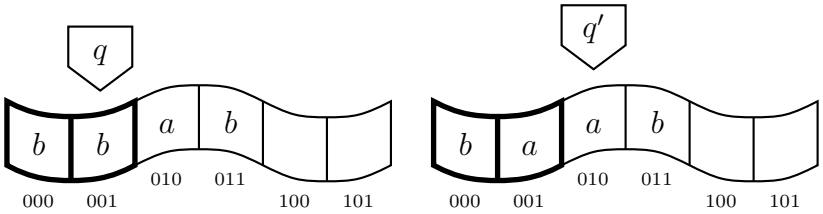
P systems with active membranes [2] are parallel computation devices inspired by the internal working of biological cells. Their main features are a hierarchy of nested membranes, partitioning the cell into regions, and *multisets* of symbol-objects describing the chemical environment. The system evolves by applying rules such as non-cooperative multiset rewriting (i.e., objects are individually rewritten), communication rules that move the objects between adjacent regions, and membrane division rules that increase the number of membranes in the system. The membranes also possess an *electrical charge* that works as a local state, regulating the set of rules applicable during each computation step. The rules, in turn, may change the charge of the membrane where they take place.

In order to solve computational problems one usually employs *polynomial-time uniform families* of P systems with active membranes, consisting of a P system Π_n for each input length n (as for Boolean circuits) and a single Turing machine constructing Π_n from n in polynomial time. The actual input is then encoded as a multiset of objects, and placed inside an input membrane of Π_n. The space required by a family of P systems (in terms of number of membranes and objects) for solving a decision problem can then be analysed as a function of n. It is already known that polynomial-space P systems and polynomial-space Turing machines are equal in computing power [3], but the proof of this result does not generalise to larger space bounds. In this paper we show the key ideas needed in order to prove the exponential-space analogue of that result by directly simulating deterministic exponential-space Turing machines using P systems.

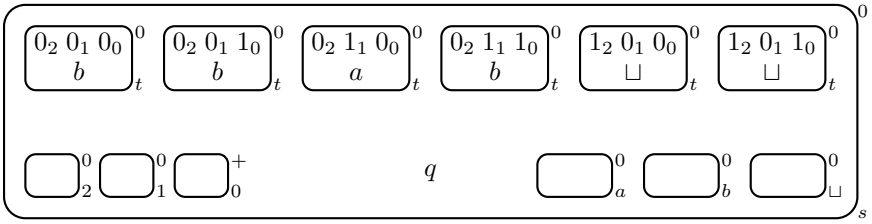
For the full technical details of the results presented here we refer the reader to the paper "The computational power of exponential-space P systems with active membranes" [1] by the same authors.

2 Simulating Turing machines

We describe how deterministic Turing machines working in exponential space can be simulated by P systems by means of an example. Let M be a Turing machine processing an input x of length $n = 2$ and requiring $2^n = 4$ auxiliary tape cells (the total length of the tape is then 6); assume that the alphabet of M consists of the symbols a and b. Suppose that the current configuration $\mathcal{C}$ of M is the one depicted on the left of the following picture, and that the transition it performs leads it to the configuration $\mathcal{C}'$ on the right. In the picture, the tape cells of M are identified by a binary index.



We encode the configuration $\mathcal{C}$ of M as the following configuration of the P system Π simulating it:



In this picture, the label of a membrane is indicated at its lower-right corner, while its electrical charge (+, 0, or −) is at its upper-right corner. The symbols located inside the membranes represent the objects in the configuration of Π.

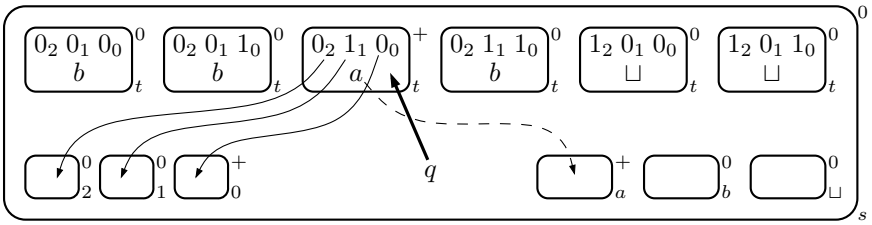
The P system, beside its external membrane s, possesses 6 membranes labelled by t (called the *tape-membranes*) corresponding to the tape cells of M; each tape-membrane contains 3 subscripted bit-objects encoding the index of the corresponding tape cell (the subscript represents the position of the bit in the index; for instance, the presence of bit-object 1_0 indicates that 1 is the least significant bit). Furthermore, each tape-membrane contains an object representing the symbol written in the corresponding tape cell of M, where $\sqcup$ represents a blank cell. Only one tape membrane is part of the initial configuration of Π, as it can be at most polynomial in size; the other ones are created by membrane division during an initialisation phase of Π, before simulating the first step of M.

A *state-object* q represents the current state of M; this object will also regulate the simulation of the next step of M. The position of the tape head is encoded in binary as the electrical charges of the membranes $0, 1, 2$ (the *position-membranes*); the label of each membrane represents the position of the corresponding bit, while its charge the value of the bit: a neutral charge represents a 0,

and a positive charge a 1. In the example above, the charges of membranes $2, 1, 0$ are $0, 0, +$, encoding the binary number 001 (decimal 1).

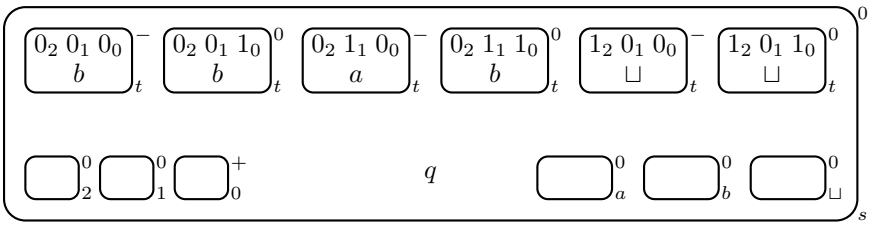
Finally, the auxiliary membranes labelled by $a, b, \sqcup$ (the *symbol-membranes*) in the lower-right corner correspond to the tape symbols of M, and are used in order to read the symbol on the current tape cell.

The following picture shows how the next step of M is simulated.



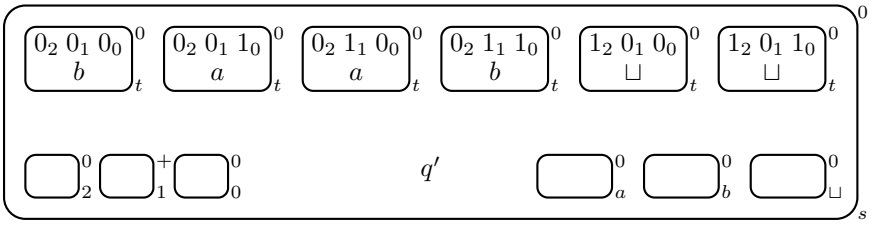
First, the object q nondeterministically guesses a tape-membrane t (all such membranes are indistinguishable from the outside) and enters it (thick arrow in the picture) while changing the charge to positive. The change of charge enables the bit-objects inside it to move to the corresponding position-membranes (along the thin arrows), where their values are compared to the charges of the membranes; this allows us to check whether the tape-membrane we guessed is indeed the one under the tape head of M. In the mean time, the object a is sent to the corresponding symbol-membrane (dashed arrow) in order to change the charge to positive.

Since in the example the tape-membrane that was chosen is not the correct one, an error-object is produced by one of the mismatched position-bits, and the configuration of Π is restored to the initial one, with the following exception: the charge of the tape-membrane is set to *negative*, so it will not be chosen again. The P system then proceeds by guessing another tape-membrane among the remaining (neutrally charged) ones. After a number of wrong guesses, the configuration of Π will be similar to the following one.



When the tape-membrane corresponding to the current cell of M is finally guessed, we can perform the actual simulation of the computation step (updating the position of the head, the symbol on the tape, and the state of M). The state-object may first read the tape symbol by looking at the only positively charged symbol-membrane; it can then update the charges of the position-membranes (from the least to the most significant bit) in order to increment or decrement the binary number they encode, produce the new tape symbol (b in the example)

and finally rewrite itself as the new state of M (q' in the example). The charges of all tape- and symbol-membranes are also reset to neutral by using auxiliary objects. The configuration of Π corresponding to the new configuration of M thus becomes the following one.



Now the P system continues simulating the next steps of M, until an accepting (resp., rejecting) state is reached; when this happens, the P system produces a YES (resp., NO) object that is sent out from the outermost membrane as the result of the computation.

3 Conclusions and open problems

The simulation described in the previous section can be carried out by a polynomial time uniform family of P systems with active membranes operating in space $O\big(s(n) \log s(n)\big)$, where $s(n)$ is the space required by the simulated Turing machine on inputs of length n. Since an analogous result holds in the opposite direction [3, Theorem 5], the two classes of devices solve exactly the same decision problems when working withing an exponential space limit.

The techniques employed here do not carry over to the simulation of super-exponential space Turing machines, since they would require a super-polynomial number of subscripted objects in order to encode tape positions; this amount of objects (and their associated rules) cannot be constructed using a polynomial-time uniformity condition. Novel techniques will be probably needed in order to prove that the equivalence of Turing machines and P systems also holds for larger space bounds.

References

1. Alhazov, A., Leporati, A., Mauri, G., Porreca, A.E., Zandron, C.: The computational power of exponential-space P systems with active membranes. In: Martínez-del-Amor, M.A., Păun, Gh., Pérez-Hurtado, I., Romero-Campero, F.J., Valencia-Cabrera, L. (eds.) Tenth Brainstoming Week on Membrane Computing, vol. I, pp. 35–60. Fénix Editora (2012)
2. Păun, Gh.: P systems with active membranes: Attacking NP-complete problems. Journal of Automata, Languages and Combinatorics 6(1), 75–90 (2001)
3. Porreca, A.E., Leporati, A., Mauri, G., Zandron, C.: P systems with active membranes working in polynomial space. International Journal of Foundations of Computer Science 22(1), 65–73 (2011)