Eight More Research Directions

Artiom Alhazov

artiom@math.md

.

Direction list

- Counting
- Anti-membranes
- Channels
- Global
- Concistency
- cP
- PLingua
- RuleForms

Counting 1

- “**How much information is stored in a configuration of a membrane system with symbol objects?**“
- Modulo isomorphism: as "size" it is enough to specify the number m of membranes and t of objects.
- In total: infinite unless we bound the alphabets (membrane alphabet (labels) and object alphabet (symbols),
  One alternative: fix the alphabets, in particular |O|=n kinds of objects, the number m of membranes and maximal number k of objects of each kind in each region. Then the total number of objects is not fixed, but bounded by m*n*k.

Counting 2

- **One membrane, modulo isomorphism**. t=0-->1 (empty multiset), t=1-->1, t=2-->2 (aa,ab), t=3-->(aaa,aab,abc), t=4-->5, t=5-->7, t=6-->11. Can be specified by a recurrent two-argument function. Seems to correspond to number sequence https://oeis.org/A000041

Counting 3

- **Unlabeled membrane structures without objects, modulo isomorphism**.
- m=1-->1 (only skin),
- m=2-->1 (two nested membranes),
- m=3-->2 ([[][]] and [[[]]]),
- m=4-->4, m=5-->9.
  Seems to correspond to number sequence https://oeis.org/A000081
- Note: with labels, already m=2 gives 2 different configurations, [[]_2]_1 and [[]_1]_1.

Counting 4

- **One membrane, one label, one polarization, no isomorphism**.
- Fixing the alphabet size |O|=n, the number of different multisets of cardinality t is the number of t-combinations with repetitions of set O, equal to C(n+t-1,t), where C(n,k)=n!/(k!*(n-k)!). https://en.wikipedia.org/wiki/Combination#Number_of_combinations_with_repetition

Counting 5

- **m membranes, |O|=n, at most k objects of any kind in any membrane, no isomorphism**. Then the number of different configurations would seem to be equal to the number of different membrane structures with m membranes, multiplied by (mn)^(k+1). Correction: but even in this case it is not so simple, because [[a][]] and [[][a]] are the same configuration; the expression above only holds under the assumption that there are no indistinguishable membranes, e.g., all membranes have different labels.

Counting 6

- Goal: to have a general formula for each typical set of parameters specifying "size", of all configurations of this "size".
  If counting tree structures is difficult, start with tissue.
  Why? to understand how much information is indeed stored in a configuration of a P system, because the general impression that, with m membranes and t objects, there are approximately exponentially many different configurations, is too inaccurate and in some settings incorrect.

Counting 7

- This topic is easy to start thinking about, and hopefully it would motivate you to take a pen and a piece of paper and try to count something.
  When a problem is clarified, one can try to use favorite programming language to automatically enumerate and count configurations until some small bound, but still beyond the point where one would introduce too many mistakes by hand.

Anti-membranes 1

- Reminder: rules of types []h→[]j []k, []h[]h'→λ ; could also be with objects.
- A. ALHAZOV, R. FREUND, S. IVANOV: **(Tissue) P Systems with Anti-Membranes**. In Seventeenth Brainstorming Week on Membrane Computing (Orellana-Martín, D.; Păun, Gh.; Riscos-Núñez, A.; Andreu-Guzmán, J. A., Eds.), Sevilla. RGNC report 1/2019, University of Seville, Artes Gráficas Moreno, S.L., 2019, 29–30.
- http://www.gcn.us.es/files/17bwmc/029_AntiMembranes.pdf
- and
- A. Alhazov, R. Freund, S. Ivanov: **P Systems with Anti-Membranes**. In Proceedings of the 20th International Conference on Membrane Computing, CMC20, Curtea de Argeş (Păun, Gh., Ed.). Bibliostar, Râmnicu Vâlcea, 2019, 249–256. http://membranecomputing.net/cmc20/pdf/procCMC20.pdf#page=250

Anti-membranes 2

- 1) Can we still do anything non-trivial if changing membrane labels is forbidden? Is it possible, e.g., to simulate boolean circuits?
- 2) What if we forbid changing labels but allow a limited (3?) number of polarizations? let's say annihilation needs _some_ form of polarization agreement
- 3) Descriptional complexity of a small universal NFPAMS
- 4) Which ingredients are needed to solve SAT with anti-membranes?
- 5) How we can exploit deeper membrane structures? For instance, annihilation of nested membranes outside-in performs an ordered sequence of membrane dissolutions.
- 6) antiMembranes for efficiency? In any way that is not a trivial translation of the previous research from objects to membranes.

Channels

- For symport/antiport P systems, in tissue case, it is _usually_ assumed that channels do not admit any parallelism. There has been a few exceptions. 1) Some Rudi's talk with PPT slides many years ago, where cells were represented by huge colored circles, I do not remember the title. 2)
- A. Alhazov, R. Freund, M. Oswald: **Tissue P Systems with Antiport Rules and Small Numbers of Symbols and Cells**. In: De Felice C., Restivo A. (eds) Developments in Language Theory. DLT 2005. Lecture Notes in Computer Science 3572. Springer, Berlin, Heidelberg, 2005, 100-111.
- https://doi.org/10.1007/11505877_9
- , where in Ot'P, primed letter t indicated that it was allowed to have distinct channels (i,j) and (j,i). 3) A more recent paper
- H. Adorna, A. Alhazov, L. Pan, B. Song: **Simulating Evolutional Symport/Antiport by Evolution-Communication and vice versa in Tissue P Systems with Parallel Communication**. In: Gheorghe M., Rozenberg G., Salomaa A., Zandron C. (eds) Membrane Computing. CMC 2017. Lecture Notes in Computer Science 10725. Springer, Cham, 2018, 1-14.
- https://doi.org/10.1007/978-3-319-73359-3_1
- relating evolutional symport/antiport with evolution-communication -- in order to make it possible having direct simulation with a slowdown by a factor of a constant, communication needed to be massively parallel. 4) Older research on neural P systems, probably by [Krishna,Rama], long time before spiking… anyway, that last one was quite a different model.
- - Parallel VS sequential channels in tP systems.
- Improve results with mcre from NP∪co-NP to PSPACE.

Global

- considered by A.Păun and once briefly by myself. This relates to problem **(Q6)** in Gheorghe's open problem list http://www.gcn.us.es/?q=18bwmc_openproblems . IF membrane structure is static and we do not care about descriptional complexity, making all rules global does not seem to restrict us at all: objects can always be renamed when moved, so they know where they are ☺
- However, the total number of rules in this reduction may increase, and this technique becomes more complicated, or even impossible, with dissolution. BTW, this may open an interesting discussion at solving hard problems in polytime. Besides, not all membranes are created equal: by definition, elementary membrane division is not applicable to membranes that are (currently) non-elementary, and the skin cannot be dissolved or divided (and sometimes it is forbidden for any object to enter it) - this trick _might_ help distinguishing membranes when needed, however, requiring non-determinism or complicated simulation. On the other hand, with sufficient ingredients one working region is already enough, so we should stay in a restricted enough settings.
- A. Păun: **On P Systems with Global Rules**. In: Jonoska N., Seeman N.C. (eds) DNA Computing. DNA 2001. Lecture Notes in Computer Science, vol 2340. Springer, Berlin, Heidelberg, 2002, 329-339.
- https://doi.org/10.1007/3-540-48017-X_31
- A. Alhazov, R. Freund: **On the Efficiency of P Systems with Active Membranes and Two Polarizations**. In: Mauri G., Păun G., Pérez-Jiménez M.J., Rozenberg G., Salomaa A. (eds) Membrane Computing. WMC 2004. Lecture Notes in Computer Science, vol 3365. Springer, Berlin, Heidelberg, 2005,
- https://doi.org/10.1007/978-3-540-31837-8_8
- A. Alhazov, R. Freund, S. Ivanov: **Length P Systems**. Fundamenta Informaticae 134(1-2), 2014, 17-37.
- https://doi.org/10.3233/FI-2014-1088

Consistency

- Reminder: here applicability does not only depend on lhs. I have heard about a practical use of this mode in a BWMC2019 discussion from Agustin (though I forgot which application it was for, so I would not know what reference to cite).
- Usually in membrane computing rule applicability only depends on the left side of the rule (whether all reactants are present in the current configuration, and, possibly, whether some additional conditions are satisfied, e.g., promoters, inhibitors, etc.).
- Consider rules changing membrane polarization. Allow to apply multiple rules (maximal parallelism), as long as the polarization in their rhs is the same. Let me call it "polarization agreement". Need to be precise, probably need to choose the polarization corresponding to at least one applied rule, if possible.
- Other examples of maximal consistency:
- - Parallel string rewriting without conflicts [D. Besozzi], many years ago, reference needed.
- - Rudi's target agreement/label agreement, original reference needed.
- - Any other shared resource to agree upon?
- Overall, I believe this feature deserves more attention.

cP

- = P systems with **complex objects**, see [Nicolescu]. Reminder: prolog-like rules using power of term rewriting and unification. Very powerful model, e.g., a solution of the **Travelling Salesman Problem** has been reported with **five** rules only [CooperNicolescu_ACMC2017].
- Some longer time ago the colleagues in my institute wanted to attack with P systems the problem of finding Gröbner basis. Unfortunately, the data structures that can be represented and efficiently processed by usual P systems are limited, and hence they are not suited well to work, e.g., with **dynamic ordered lists of strings** (a solution via Turing machine is not elegant). It turns out that cP systems are much more flexible in representing and efficiently processing complicated data structures.
- Some problems that have been addressed besides universality/computational completeness and NP-hard problems, by usual P systems:
- - sorting https://doi.org/10.1007/3-540-29937-8_8 ,
- - dictionary search and update http://univagora.ro/jour/index.php/ijccc/issue/download/44/pdf_165 ,
- - inflections http://www.math.md/publications/csjm/issues/v17-n2/10082/ , annotating affixes https://doi.org/10.1007/978-3-642-54239-8_7 ,
- - firing squad synchronization problem
- https://doi.org/10.1007/978-3-540-95885-7_9
- (more problems and solutions can be found in
- Applications of Membrane Computing, 2005 and Membrane Computing Handbook).
- Need: more problems that are practical, well defined and sufficiently simple (simpler than Gröbner basis), to be attacked by cP systems, but not completely trivial (needing, say, more than two rules).

PLingua 1

- 1. Simulator. It would be very useful for theory to have a proper tool computing **the set of all possible transitions** from a given configuration.
- (Yes, I remember you are more focused on applications like zebra muscles, and you are quite concerned that it does not scale well. However, enough theory is anyway done, and there can be multiple simulators for PLingua. A few times I have been so upset that I thought about programming something like that myself, but what I do is normally not compatible, not user-friendly and definitely not in Java)
- Basically, having fixed the current configuration C, for each rule r it is easy to compute the maximal number max(r,C) of times it can be applied in parallel. By dividing, for each object a in lhs(r), |C|_a by |lhs(r)|_a rounding down, and taking the minimum. Same works in a distributed way, assuming proper flattening, possibly on the fly. In the _worst_ case, all possibilities are among the combinations, for each rule r, of applying it from 0 to max(r,C) times. It only takes to verify that the multiset union of lhs(r) times the number of applications of r, summed over all rules r, is contained in r. That would be asynchronous mode. For any other mode, compliance is also to be checked. Of course, for maxpar that would be non-applicability of ANY further rule to the idle objects. Of course, in particular cases the set of possible transitions could be computed more efficiently.

PLingua 2

- 2. Semantics and membranes. The recent advances in PLingua, like defining user rule types, seem to be quite useful. Yet, the main value I see is being able to specify rules other than the most usual ones in the model (and, in particular, being able to combine rules from different models), and a comfortable way to write them is secondary, though also nice.
- A thing which is often related to syntax is how to apply it, the most needed versions being "in max. parallelism" and "sequentially". In particular, rules (a) are normally treated as parallel, even though sequential version has been considered, while rules (b),(c),(d),(e),… are normally considered sequential, even though without polarizations alternative semantics has been studied.
- Imagining rules involving more than one membrane, we need to be more precise. I proposed to indicate for each user type of rules (how exactly is a secondary question) which membranes are resources and which membranes are context. Then, resources are lhs(r) and contexts are like promoters. Clearly, under usual definitions a rule would be sequential with respect to resources and parallel with respect to contexts.
- If that is too difficult, usually it is enough to have implicit semantics: any membrane written completely in lhs(r) is a reactant, a membrane written in rhs(r) is a product, and a membrane containing "→" is a context. Then, [a→u] is a parallel rule, but [a]→[u] would be a sequential rule. Similarly, it automatically follows in [[]→[][]] that the external membrane is a context, while internal membranes are resources, so we already know what is parallel and what is sequential. However, if the user wants such a rule to be sequential also w.r.t. the outer membrane, then it can be written as [[]]→[[][]] .
- Unfortunately, this convention alone does not suffice for automatic deduction of parallelism for rules like (b_0), (c_0). Because the context is not outside. (Yes, they could be written as boundary rules, but this syntax is neither universal nor compatible with traditional syntax for active membranes). But of course something can always be invented, e.g., when specifying rule types, write "[p" vs "[s" (parallel vs sequential) or "[r" vs "[c" (resource vs context).
- Moreover, I was told that there is some problem with templates without external membrane, except the standard types ☹

PLingua 3

- 3. Dynamic membranes
- Clearly, without explicitly indicated semantics []→[[]] would be a seqential membrane creation, while [a→[b]] would be a parallel membrane creation. Then,
- [[a]→b] is a membrane dissolution, where the external membrane is a context.
- But what is the behavior of other objects, those not specified in the rules explicitly?
- The main variant is of course, upon creation the new membrane will only contain b, and upon dissolution all the contents of the old membrane is released in the outer membrane. But of course there are other rules, although less studied. Last year I suggested to use some wildcard, or mask, e.g., $1, to represent other objects (similarly, something like #1 can represent other membranes, and for technical reasons different characters may be chosen; I use these ones to explain the idea how to describe semantics different from the main one).
- [a $1→$1 [b]] usual parallel membrane creation
- a $1→$1 [b] same without the outer context
- [a $1]→[b [$1]] create a new membrane around the existing one and send b there
- [[a $1]→b $1] usual membrane dissolution
- [[a $1]→b] lose contents of the dissolved membrane
- [a $1]→[b $1][c $1] usual membrane division
- [a $1]→[b $1][c] create a sibling membrane, without replicating contents
- [a $1]→[$1(O)][$1(O')] membrane separation
- [a $1[$2]]→[$2 [a $1]] exchange objects in two membranes if the first one contains a
- Then, there may be different kinds of non-elementary membrane division
- [a]→[b][c] same syntax as for elementary membranes, replicate objects and membranes.
- Can be written as [a $1 #1]→[b $1 #1][c $1 #1]
- [[][]]→[[]][[]] separating submembranes. But what exactly happens to other submembranes if there are more than these two?
- [$1 #1[][]]→[$1 #1[]][$1 #1[]] replicating other objects and membranes
- [$1 #1[][]]→[$1(O) #1[]][$1(O') #1[]] separating objects and replicating membranes
- [$1 #1[][]]→[$1 #1(H)[]][$1 #1(H')[]] replicating objects and separating membranes
- [$1 #1[][]]→[$1(O) #1(H)[]][$1(O') #1(H')[]] separating objects and membranes
- Overall, I think there may be some reasonable consistent universal way how to describe the precise evolution not only of dedicated objects and membranes, but also related objects and membranes, because mass action is needed (the first classical example of the mass action is dissolution, of course currently programmed explicitly).

PLingua 4

- 4. Tests
- From time to time, researchers consider rules that were not considered in the original model.
- I believe many (though not all) of these issues can be captured by the thoughts above.
- a) sequential (a), parallel (b),(c),(d),(e),…
- b) where other objects and membranes go - division vs separation, outside vs delete, …
- c) external rules: a[]→u[], a[]→b, a[]→b[][], …
- d) …

PLingua 5

- 5. Other models besides active membranes
- With suitable choice of parallel/sequential semantics made clear, $r \in Ri$ can be written as [r]i. In most cases, membrane i must be treated as context, hence, rules are parallel with respect to it.
- Antiport: u[v]→ v[u]
- Evolutional antiport or boundary rules: u[v]→u'[v']
- Transitional: pretty standard, except multiple targets would be represented as multi-membrane context, and dissolution semantics is normally assumed parallel (multiple δ = one dissolution), not sequential.
- Spiking: mostly similar, the main difference are additional regular expressions.
- Promoters, inhibitors - how much is already captured by PLingua??
- Priorities - is there already a well-established syntax for them?
- Notice that strong and week priorities can co-exist: these are just additional filters for the set of the next configurations (see part 1: Simulator) besides the derivation mode.
- As discussed with Rudi a few days after BWMC19, filters like priorities should be applied BEFORE the derivation mode filter.

PLingua 6

- 6. Other derivation modes.
- A new (mostly studied in the last few years) important derivation mode for many models is **set maximally-parallel**. Same as maximally parallel, but in each step each rule may be only applied once. Technically similar to having a dedicated catalyst for each rule.
- Some of the classical modes that would be most important to also have are sequential and asynchronous. Asyn is even easier than maxpar - just remove the maximality filter. Sequential is of course the easiest to implement.
- 7. New ways of rule control.
- Activation and blocking (I hope to soon finish formalizing the concept also for zero-delay).

PLingua 7

- 8. One of the "worst" things that could happen.
- "Denying"
- This is how we call the situation where there exists at least one applicable rule, but there is no valid multiset of rules. An example is ">1 mode" in the situation where only one rule is applicable. This situation has been carefully avoided in the first years of membrane computing, but it does not present a problem (except it is unusual), e.g., this is similar to what happens to partially blind register machines when they try to decrement a register containing zero, which is not allowed by the model.
- Finally, a question is - can all of this co-exist in the same context?
- I still think it could. If anyone has an example of ANY membrane features that seem incompatible, please let me know, and _maybe_ I will be able to convince you that there is no problem. Reminder - a universal look at P systems models: network of cells, see a few publications on the Formal Framework for a)static structures, b)dynamic structures, c)spiking.
- R. Freund, S. Verlan: **A Formal Framework for Static (Tissue) P Systems**. In: Eleftherakis G., Kefalas P., Păun G., Rozenberg G., Salomaa A. (eds) Membrane Computing. WMC 2007. Lecture Notes in Computer Science 4860. Springer, Berlin, Heidelberg, 2007, 271-284.
- https://link.springer.com/chapter/10.1007%2F978-3-540-77312-2_17
- R. Freund, I. Pérez-Hurtado, A. Riscos-Núñez, S. Verlan: **A Formalization of Membrane Systems with Dynamically Evolving Structures**. International Journal of Computer Mathematics 90(4), 801–815 (2013)
- https://doi.org/10.1080/00207160.2012.748899
- S. Verlan, A. Alhazov, R. Freund, S. Ivanov: **A Formal Framework for Spiking Neural P Systems**. In Proceedings of the 20th International Conference on Membrane Computing, CMC20, Curtea de Argeș (Păun, Gh., Ed.). Bibliostar, Râmnicu Vâlcea, 2019, pp. 523–535. http://membranecomputing.net/cmc20/pdf/procCMC20.pdf#page=250

Rule Forms in Rewriting. Catalytic P Systems

Artiom Alhazov

Vladimir Andrunachievici Institute of Mathematics and Computer Science

Previous publication on this topic:

http://www.math.md/imcs55/Proceedings_IMCS_55.pdf#page=283

In collaboration with R. Freund and S. Ivanov

6th Annual Rogozhin Lectures. 15.11.2019

Intro

- A rewriting rule: $u \rightarrow v$ $(u,v \in V^*)$
- In strings: $sut \Rightarrow svt$
- Multiset: an unordered string
- Parallelism: apply rule(s) to independent objects
- Maximality requirement: nothing is applicable to idle objects
- Inspiration: normal forms in grammars
  - ex.: $\{A \rightarrow BC, A \rightarrow a\}$: Chomsky NF for $\{L \in CF | \lambda \notin L\}$

Computational completeness

- Capability to generate/accept all recursively enumerable sets of
  - strings or
  - vectors of non-negative integers
- Allow ignoring a bounded amount
  - of extra symbols (ex: state, catalyst) and/or
  - strings/multisets (ex: λ)
- Models: sequential/parallel string/multiset rewriting
- Focus: number of rule forms, their length

Rewriting formally

- (V,R), where finite $R \in V^* \times V^*$
- Additional specifications:
  - Initial string/multiset w
  - Sub-alphabets, e.g., terminals T or states Q
  - Special sub-alphabets (for additional restrictions/normal forms), e.g., catalysts C
  - Derivation relation, e.g., $\{sut \Rightarrow svt | s,t \in V^*, (u,v) \in R\}$ for sequ. Grammars
  - Halting condition, e.g.,
    - absence of non-terminals (grammars)
    - appearance of the final state (machines)
    - inapplicability of any rule (P systems)

More?

- In principle rules may be more complicated (control mechanisms or distributed systems)
- Notation may need special symbols for additional ingredients
  - e.g., permitting/forbidden context, membrane labels/polarizations, etc.,
- We mainly restrict ourselves to the basic rule forms

Rule form

- A pattern of the form u→v, where u,v are strings of variables (and, possibly, special symbols)
- For a computing model M, we say that a system G of type M is constructed from a set F of rule forms if each rule of G is obtained from some rule form in F by replacing each variable by an element from an alphabet specified for this variable.
- Usual sub-alphabets: $T \subseteq V$, $N = V \setminus T$

Rule form examples

- $A \rightarrow BC$ $(A,B,C \in V)$ is a rule form
- $A \rightarrow u$ $(A \in V, u \in V^*)$ is not a rule form
  - Can be written as infinite set of rule forms, one for each $|u|$
- A set F of rule forms is sufficient for computational completeness of a formal computing model M if the family of string/multiset languages generated/accepted by systems of M constructed from F equals RE/PsRE, possibly ignoring a bounded amount of symbols or strings/multisets

Sequential rewriting

- {A→BC,AB→C}: for Turing completeness cooperation plus lengthening and shortening the sentential form is enough
  - Obtained from type-0 grammars or TMs with intermediate symbols
  - Also need A→λ to generate λ (obvious)
- Accepting (by halting): {AB→CDE} is enough
  - AB→CGG for AB→C with AG→GGA, GA→AGG
  - AE→BCE for A→BC
- Other remarkable examples can easily be derived from well-known normal forms

From normal forms & ins/del

- [Pentonnen]: {A→BC,AB→AD,A→a,E→λ}
- ➔: {A→BC,AB→AD,A→λ} (A,B,C,D∈V)
  - A→aE, E→λ for A→a
- [Geffert]: {S→uSv,S→u,AB→λ,CD→λ}
- ➔: {X→YZ,XY→λ}
- [MargensternPăunRogozhinVerlan]:
- {λ→AB,ABC→λ} and {λ→ABC,AB→λ}

Sequential multiset grammars

- Not computationally complete: power of partially blind register machines
- Possible with additional mechanisms, e.g., inhibitors: {$AB \rightarrow C, A \rightarrow BC|_{\neg F}$}
  - p:(A(i),q) by $p \rightarrow a_i q|_{\neg h}$,
  - p:(S(i),q,r) by $pa_i \rightarrow q, p \rightarrow p'p''|_{\neg a_i}$, $p'p'' \rightarrow q$
- Accepting (by halting): {$AB \rightarrow CDE|_{\neg F}$}
- Parallelism: later

Infinite tape

- [Turing]: {(p,A)→(q,B,R), (p,A)→(q,B,L)} (p,q$\in$Q, A,B$\in\Gamma$)
- Rewriting: (p,A)→(q,B,R) by pA→Bq,
- (p,A)→(q,B,R) by CpA→qCB $\forall C\in\Gamma$
- Split into not moving and not reading/writing:
- ➔ {(p,A)→(q,B,S),(p,C)→(q,C,d)$\forall$C,d=L/R}
- Proof: (p,A)→(q,B,d) by (p,A)→(r,B,S) and (r,B)→(q,B,d)
- Rewriting: (p,A)→(q,B,S) by pA→qB
- (p,A)→(q,A,R) by pA→Aq
- (p,A)→(q,A,L) by Cp→pC$\forall$C
- ➔ {PA→RB,RB→BR,BR→RB} (P,R$\in$Q, A,B$\in\Gamma$)

Two-stack machine & circular tape

- ➔ {P→AR,AP→R,P→RA,PA→R} (P,R∈Q, A,B∈Γ)
- push(left), pop(left), push(right), pop(right)

- [KudlekRogozhin]: {(p,A)→(B,q,R), (p,A)→(λ,q,R), (p,A)→(BC,q,R)}
- “Circular Post Machine”
- Rewriting: pA→Bq, pA→q, pA→BCq
- “Variant 5”: (p,λ)→(A,q,R), (p,A)→(λ,q,R)
- ➔ {pA→q,p→Aq} (p,q∈Q, A∈Γ)
- On linear tape: left deletion, right insertion
- Queue automaton

P systems

- Maximally parallel multiset rewriting
- Non-extendable multisets of rules acting on multisets; halt when no rules are applicable
- One-region systems
- {A→BC,AB→C}
  - p:(A(i),q) by p→a_iq, p:(S(i),q,r) by p→p'D_i, p'→p''p''', $D_i a_i \rightarrow Z_i$, p''p'''→p'''', p''''D_i→q, p''''Z_i→r
- Accepting (by halting): {AB→CDE}
- Antimatter ➔{A→BC,AB→λ}

Purely catalytic

- $\{cA \to cB_1 \ldots B_k | k \in F\}$ $(c \in C, A, B_1, \ldots, B_k \in V \setminus C)$
- [FreundKariOswaldSosík]: $F=\{0,1,2,3\}$ √
- $cA \to cB$ by cBE, $cA \to cBD$ by $cA \to cBDE$ with $dE \to d$ give us $F=\{0,3\}$, i.e.,
- ➔$\{cA \to c, cA \to cB_1B_2B_3\}$
- Previously open: $F=\{0,2\}$

Purely catalytic model: another view

- Put all rules with the same catalyst in a group
- Remove all catalysts
- $cA \rightarrow cB_1 \ldots B_k$ becomes "$A \rightarrow cB_1 \ldots B_k$ in group c"
- We have groups of non-cooperative rules
  - Possibly with overlapping LHSs
- Choose exactly 1 rule non-deterministically from each group or until no remaining groups have rules applicable to remaining objects
- Studied as "min_1", an ugly name for a beautiful concept. (pcat,max)=(ncoo,min_1)
- Like multiple multiset grammars working on the same sentential form, competing for resources
- Catalytic: also ncoo rules, = +1 maxpar. group

Improved rule forms

- Goal: computational completeness by rule forms {$cA \to c, cA \to cBD$}
- Start: $c_1 p_0 c_2 d_2$
- Note: $ca \to cb$ by $ca \to cbe$ and $c_3 e \to c_3$
- Note: $p \to \#$ by $c_4 p \to c_4 \#$
- General rules:
  $c_i o_i \to c_i d_{3-i}$, $c_i d_i \to c_i$, ($d_i \to \#$, $\# \to \#$)
- Simulating p:A(j),q
- $c_1 p \to c_1 p_1 a_j$ $c_2 d_2 \to c_2$ ($p \to \#$)
- $c_1 p_1 \to c_1 q d_2$ $c_2 a_j \to c_2 o_j$ ($a_j \to \#$, $p_1 \to \#$)

Improved rule forms - II

- Assumption: registers 1,2 are never (0,0) for S - instructions
- Simulating p:S(1),q
- $c_1 p \rightarrow c_1 p_2$ $\quad c_2 d_2 \rightarrow c_2$ $\quad (p \rightarrow \#)$
- $c_1 o_1 \rightarrow c_1 d_2$ (or idle) $\quad c_2 p_2 \rightarrow c_2 p_3$ $\quad (p_2 \rightarrow \#)$
- $c_1 p_3 \rightarrow c_1 q d_2$ $\quad c_2 d_2 \rightarrow c_2$ $\quad (p_3 \rightarrow \#)$

  (or $c_2 o_2 \rightarrow c_2 d_1$)

- Simulating p:S(2),q
- $c_1 p \rightarrow c_1 p_1$ $\quad c_2 d_2 \rightarrow c_2$ $\quad (p \rightarrow \#)$
- $c_1 p_1 \rightarrow c_1 p_2$ $\quad c_2 o_2 \rightarrow c_2 d_1$ (or idle) $\quad (p_1 \rightarrow \#)$
- $c_1 d_1 \rightarrow c_1$ $\quad c_2 p_2 \rightarrow c_2 q d_2$ $\quad (p_2 \rightarrow \#)$

  (or $c_1 o_1 \rightarrow c_1 d_2$)

Improved rule forms - III

- Note: we treat decrement and zero-test as separate instructions
- Simulating p:Z(1),q
- $c_1p \rightarrow c_1p_2$ $c_2d_2 \rightarrow c_2$ $(p \rightarrow \#)$
- c_1 idle (or $c_1o_1 \rightarrow c_1d_2$) $c_2p_2 \rightarrow c_2qd_2$ $(p_2 \rightarrow \#)$
- Simulating p:Z(2),q
- $c_1p \rightarrow c_1p_1$ $c_2d_2 \rightarrow c_2$ $(p \rightarrow \#)$
- $c_1p_1 \rightarrow c_1qd_2$ c_2 idle (or $c_2o_2 \rightarrow c_2d_1$) $(p_1 \rightarrow \#)$
- If instruction choice is wrong, at least one of red symbols produces #

Improved rule forms - IV

- Proved: {cA→c,cA→cBD} is computationally complete (4 catalysts)
- Replace cA→c (A≠e) by cA→cee and e→λ
- Proved: {A→λ,cA→cBD} is computationally complete (2 catalysts; c_3 and c_4 not needed)
- Open: what about {cA→c,A→BD}
- Conjecture: not universal
- Reason: checking decrement needs a witness (non-catalyst in rhs of a cat. rule)
- Question: what is the power of RMs with unconditional transfer?

Improved rule forms - V

- Accepting Ps(d)RE
- Need a catalyst for each input registers + a small constant number of working registers
- Do not need erasing, produce dummy-symbols instead; only halting matters
- [FreundKariOswaldSosík]:➔{cA→cBDE} with d+3 catalysts
- Conjecture: {cA→cBD} should suffice for computational completeness, maybe even with the same number of catalysts

Research directions. Conclusions

- Sequential string grammars: further variants of rule forms, e.g., Geffert-based
- Parallel multiset rewriting: further variants, e.g., with mcre/mdiv, active membranes
- Generalize new cat. results for accepting vectors

- Presented: a number of generating&accepting devices and corresponding sets of rule forms sufficient for obtaining computational completeness
- Discussed: sequential grammars & Turing machines, circular Post machines, 2-stack machines as specific models for computations on strings
- Described: several sets of rule forms sufficient for obtaining computational completeness for sequential string grammars, especially using specific well-known normal forms as the Penttonen NF and the Geffert normal form
- Considered: model of 1-region membrane systems, where multisets are evolving by applying non-extendable multisets of rules in parallel
- Investigated: rule forms necessary for several variants of P systems
- Proved comput.compl. for {cA→c,cA→cBD} and {A→λ,cA→cBD}.
- Open: is the number of catalysts optimal?
- Open: the challenging open question about {cA→c,A→BD}.
- Exercise: {cA→cB,A→BD,A→λ}.
- Discussed: further variants to be investigated in the future.
- Thank you for your attention

Direction list

- Counting
- Anti-membranes
- Channels
- Global
- Concistency
- cP
- PLingua
- RuleForms

- Thank you for your attention