

On the statistical thermodynamics of communicating processes

Bacci, Giorgio; Danos, Vincent; Kammar, Ohad

Published in:
Lecture Notes in Computer Science

DOI (link to publication from Publisher):
[10.1007/978-3-642-22944-2_1](https://doi.org/10.1007/978-3-642-22944-2_1)

Publication date:
2011

Document Version
Early version, also known as pre-print

[Link to publication from Aalborg University](#)

Citation for published version (APA):
Bacci, G., Danos, V., & Kammar, O. (2011). On the statistical thermodynamics of communicating processes. *Lecture Notes in Computer Science*, 6859, 1-18. https://doi.org/10.1007/978-3-642-22944-2_1

General rights

Copyright and moral rights for the publications made accessible in the public portal are retained by the authors and/or other copyright owners and it is a condition of accessing publications that users recognise and abide by the legal requirements associated with these rights.

- Users may download and print one copy of any publication from the public portal for the purpose of private study or research.
- You may not further distribute the material or use it for any profit-making activity or commercial gain
- You may freely distribute the URL identifying the publication in the public portal -

Take down policy

If you believe that this document breaches copyright please contact us at vbn@aub.aau.dk providing details, and we will remove access to the work immediately and investigate your claim.

On the Statistical Thermodynamics of Reversible Communicating Processes

Giorgio Bacci¹, Vincent Danos^{2*}, and Ohad Kammar²

¹ DiMI, University of Udine

² LFCS, School of Informatics, University of Edinburgh

Abstract. We propose a probabilistic interpretation of a class of reversible communicating processes. The rate of forward and backward computing steps, instead of being given explicitly, is derived from a set of formal energy parameters. This is similar to the Metropolis-Hastings algorithm. We find a lower bound on energy costs which guarantees that a process converges to a probabilistic equilibrium state (a grand canonical ensemble in statistical physics terms [19]). This implies that such processes hit a success state in finite average time, if there is one.

1 Introduction

Regardless of the task a distributed algorithm is trying to complete, it has to deal with the generic problem of escaping deadlocks. One can solve this problem in a general fashion for CCS, π -calculus, and similar message-passing models of concurrency by equipping communicating processes with local memories. The role of memories is to record what local information will allow processes to backtrack, if needed. Backtracking preserves the granularity of distribution and can be seen as a special kind of communication [1,2,8]. A somewhat similar idea, based on resets rather than memories, can be found in early work from K.V.S. Prasad [17], and more recently in the context of higher-order π -calculus [11].

Following this approach, it is enough for the programmer to design a process that *may* succeed according to its natural forward semantics, and the same process, wrapped in a reversible operational semantics, *must* succeed. Separating the concern of deadlocks from the fundamentally different one of advancing locally towards a distributed consensus - makes the code simpler to understand and amenable to efficient automated verification [3,9]. What this method does not do, however, is to generate efficient code - that the code must eventually succeed does not say anything about the time it will take to do so. In fact, the general framework of non-deterministic transition systems is ill-equipped to even discuss the issue, as the worst time of arrival will often be infinite.

In the following, we provide a reinterpretation of our earlier construction of a reversible CCS in probabilistic terms. This sets the question in a well-understood quantitative framework. In particular, we can define a notion of exhaustive search

* Corresponding author: vdanos@inf.ed.ac.uk

for our distributed reversible processes, namely that the search reaches a *probabilistic equilibrium*. Better, we can show that under suitable constraints on the parameters arbitrating between forward and backward moves, a process will reach such an equilibrium. In other words, when such constraints are met, a particular event that may happen in a basic forward process, not only must happen in its reversible form, but must do so in finite average time. Obviously, this is a stronger guarantee.

1.1 Energy landscaping

To achieve this, we build an energy function, or simply a *potential*, on the state space of a reversible process. This potential generates convergent search behaviours, on the transition graph that the process generates. The actual grammar of processes that we use slightly generalizes CCS [14] in that synchronizations can be many-way (more than two processes can synchronize together; eg, as in Petri nets), and can also be non-exclusive (a channel can be used to synchronize with many others, possibly at different rates; eg, as in Kwiatkowski-Stark’s continuous π -calculus [10]). Most importantly, the potential we zero in on constrains the continuous-time Markov chain semantics without altering the granularity of distribution.

Not any potential works, and some energy policies will be too liberal and allow some processes to undergo an explosive growth which never reaches an equilibrium (despite reversibility). Our key finding is that, in order to obtain the existence of a general equilibrium, processes must have a probability to fork that decreases exponentially as a function of the size of their local memory. We prove that any rate of decrease larger than $\alpha^2 \beta \log(4(\beta + 1))$, where α is the maximal number of processes that can synchronize at once, and β the maximal number of successive forks that a process can undergo before communicating again, is sufficient.

This lower bound is reasonably sharp.

1.2 Related work

Beyond the technical result, this paper borrows from a fundamental physical intuition, that of driving and deriving the dynamics by shaping the limit probability distribution it should converge to. A stochastic semantics is simpler in this respect, but the idea can be adapted to deterministic semantics as well. This is, in essence, the celebrated Metropolis algorithm which rules supreme in search and sample problems [13,7]: one first describes the energetic landscape of the state space, equivalently the probabilistic equilibrium, and then one defines the probabilistic transitions used to travel this landscape. Transition probabilities are chosen in a way that guarantees that the dynamics converge to the said equilibrium (this is explained in more details in §2).

Our work also borrows from Ref. [4], where the question of the existence of an equilibrium for a recursive Markov chain is proved to be undecidable. The proof uses a sequential reduction of an archetypical search problem, namely the

Post correspondence problem, into the equilibrium existence problem. To obtain an equilibrium, even in the absence of a solution to the underlying Post problem instance, one introduces an energy penalty on searches. The class of potential that we use in the present paper is a distributed version of the former (although, here, we only deal with finding a potential that converges, as opposed to checking whether a given dynamics admits of one).

Note that the simple syntax of reversible CCS is merely used here for the sake of a simple reduction of the idea to practice. As for our earlier work on the qualitative aspects of reversible CCS, other syntaxes could do just as well.

Finally, and despite its concurrent-theoretic nature, the present work also draws inspirations from practical applications. In a modeling context, Ollivier’s et al. recent paper [16] proposes a method to design models of biomolecular networks of allosteric interactions that is entirely based on the systematic usage of a certain grammar of local potentials. Among other things, this guarantees the thermodynamic consistency of the models. This has been a major source of inspiration for this work, as for the earlier Ref [5] where we prove that the thermodynamic consistency of mass action Petri nets is decidable (and find a definite shape for the associated potentials).

1.3 Outline

The paper is organised as follows. The next section (§2) is a reminder of the basics of discrete-space/continuous-time Markov chains, and the (in the context of this paper) central notion of equilibrium - in essence, a special kind of fixed point for the action of the Markov chain. Next, in §3, we turn to the (slightly generalized) syntax of reversible CCS and discuss the important property of simplicity which the transition graph underlying a reversible process enjoys. Roughly, this means that the transition graph of a reversibilized process is acyclic because it incorporates its own history, and this has consequences on the construction of equilibria. As in both cases, we are dealing here with simple or well-understood objects, §2-3 will be a bit concise, but still, hopefully, reasonably self-contained.

In §4, we introduce and compare different candidate potentials which one could think of using to landscape the reversible CCS state space. The §5 investigates an example of explosive growth which shows that no equilibrium can be obtained if energy penalties are purely based on the number of synchronizations of processes; this prepares the ground for a general approach. Finally, in §6, which is the technical core of the paper, we obtain our convergence result, which gives sufficient lower bounds on energy costs for communication to ensure the existence of a probabilistic equilibrium. The main technicality has to do with finding lower bounds for the potential of a process as a function of its number of synchronizations (Lemma 4). The conclusion returns to some of the issues discussed above, and touches on likely future research and intersections with more traditional concurrent-theoretic work on rewriting and termination.

2 Probabilistic reminders

Throughout the paper we write $\log(x)$ for the *natural* logarithm; we manipulate multisets as vectors, that is to say additively, and write $|\mathbf{a}|$ for the size of a multiset (equivalently its L_1 -norm); eg $\mathbf{a} = a + 2b$, and $|\mathbf{a}| = 3$.

We start with a quick reminder on CTMCs.

2.1 Timers and chains

A *random exponential time* of parameter $\lambda > 0$ is an $[0, +\infty)$ -valued random variable T such that $p(T > t) = \exp(-\lambda t)$. Thus, the density of T is $\lambda \exp(-\lambda t)$, for $t \geq 0$; and T 's mean is $\int_0^{+\infty} \lambda \exp(-\lambda t) t dt = \lambda^{-1}$.

Suppose given a set X which is at most countably infinite, and a rate function $q(x, y) \in \mathbb{R}^+$, for x, y in X , and $x \neq y$.

The *transition graph* or the support of q , written $|q|$, is the binary relation, or the directed graph, on X which contains (x, y) iff $q(x, y) > 0$.

We suppose $|q|$ has finite out-degree (this also called being image-finite).

We can define a continuous-time Markov chain over X in the following way. When the chain is at x in X , for each of the finitely many y s such that $q(x, y) > 0$, draw a random exponential time $\tau(x, y)$ with parameter $q(x, y)$; advance time by $\tau = \min \tau(x, y)$, and jump to the (almost surely) unique y such that $\tau(x, y) = \tau$.

The idea is that all possible next states compete, and the higher the rate of $q(x, y)$, the more likely it is that y will be the next state. It is easy to calculate that the probability to jump to y is actually $q(x, y) / \sum_z q(x, z)$; and that for small t s, the probability to jump to x within t is equivalent to $q(x, y)t$, hence one can think of $q(x, y)$ as the rate at which one jumps from x to y .

Note that for the above definition to make sense it is important to suppose as we have done that $|q|$ is image-finite. We will also suppose thereafter that $|q|$ is symmetric (not to be confused with the much stronger assumption that q is a symmetric function, ie $q(x, y) = q(y, x)$), and define $\rho(x, y) = q(y, x)/q(x, y)$ when either (equivalently both) of $q(x, y)$ and $q(y, x)$ are > 0 .

2.2 Equilibrium

Now, on to the definition of an equilibrium that will be our central concern here.

Consider a function p defined on X and with values in \mathbb{R}^+ . One says p is an *equilibrium* for q if p is not everywhere zero, and:

- [detailed balance] for all $(x, y) \in |q|$, $p(x)q(x, y) = q(y, x)p(y)$
- [convergence] $Z = \sum_X p(x) < +\infty$

If such a p exists, we can obtain a probability on X by normalizing p as p/Z . Naturally, if X is finite the second condition always holds.

The detailed balance condition implies that p , construed as a probabilistic state of the system, is a fixed point of the action of the chain q , and as $|q|$ is symmetric, regardless of the initial state, the chain will converge to p (see Ref. [5, §2] for more details, or for a comprehensive textbook explanation, see Ref. [15]).

2.3 Potentials

Suppose given a real-valued function V on X (the energy landscape or the potential), together with a symmetric graph G on X (the moves one can make to travel the landscape). One can always define a rate function $q(x, y)$ over G for which $p_V(x) = \exp(-V(x))/Z$ is an equilibrium, when X is finite.

For instance, set $q(x, y) = 1$ if $V(x) \geq V(y)$ (one is always willing to travel downhill), $q(x, y) = \exp(V(x) - V(y))$ else (one is increasingly reluctant to travel uphill).

We can readily see that, with these settings, we have detailed balance:

$$p_V(x)/p_V(y) = e^{V(y)-V(x)} = q(y, x)/q(x, y) \quad (1)$$

When X is finite, this is enough to define an equilibrium, and this particular choice of a rate function q together with the choice of G is the *Metropolis algorithm*. In the case which interests us, when X can be countably infinite, the idea still applies, but one has to make sure that the potential V defines a finite $Z_V = \sum_X \exp(-V(x))$.

The converse problem of, given q , finding a potential for q , is also interesting. In general, we can pick an origin x_0 arbitrarily and within the connected component of x_0 in $|q|$, define the potential V as:

$$V(x) = \sum_{(x,y) \in \gamma} \log \rho(x, y) \quad (2)$$

for some path γ leading from x_0 to x .

Such an assignment is correct, meaning that detailed balance holds for the pair (q, V) , iff the assignment does not depend on the choice of γ . In this case, V is defined uniquely on x_0 's component up to an additive constant. If there is a dependency, then there is no solution, ie the dynamics on x_0 's component is not describable by a potential. One says then that the chain is *dissipative*. Even for simple CTMCs this property is undecidable [4].

If the transition graph $|q|$ is acyclic, seen as an *undirected* graph, there is only one choice for γ (up to trivial backtracks - which the formula above is always happy with as $\log \rho(x, y) = -\log \rho(y, x)$), so independence trivially holds.

In the next section, X will be the state space of some reversible communicating process p_0 . Reversibility will be obtained by equipping threads with memories which collectively capture the history of the computation (up to causal equivalence of computation paths, as we will see). Hence, the set of states reachable from the initial state p_0 is nearly equivalent to the space of its computations, and in particular, the underlying transition graph is (nearly) acyclic. This means that detailed balance will be for free, and only convergence will be an issue.

We mention in passing that in the case of stochastic mass action Petri nets, we find the opposite situation. Namely, verifying detailed balance might be involved as one needs to compute reactions invariants (the Petri net “loops”), whereas the convergence automatically follows [5].

3 Qualitative semantics

We start with a reminder of CCS and its reversible form. In this minimalistic model, communication is devoid of any content, that is to say no value or name changes hands in a communication event; hence we will talk rather about synchronizations (synchs for short).

We assume a countable set of *channels* A , and a finite set of *synchronizable* (non-empty) multisets of channels A^* .

A process p can be a product p_1, \dots, p_n , or a guarded sum $a_1p_1 + \dots + a_np_n$ with coefficients $a_i \in A$.

Products and sums are considered associative and commutative.

When $p = (p_1, \dots, p_n)$, p forks into the siblings p_i s which then run in parallel. When $p = (a_1p_1 + \dots + a_np_n)$, p waits for an opportunity to synch on any of the channels a_i with a set of processes willing to synchronize on a tuple \mathbf{a} such that $a + \mathbf{a} \in A^*$. When that happens, p runs p_i .

Recursive definitions are allowed only if *guarded*, meaning that a recursively defined process variable only appears prefixed (aka guarded) by a channel; such definitions are considered to unfold silently.

As the reader may have noticed, to enhance readability (as we need to compute some examples in §4-5) we use lightfooted notations. Specifically, we use the comma to denote the product, and juxtaposition for prefixing.

An example of process (which we analyze from close in §5) is $p_0 = p, p'$, with $p = a(p, p)$, $p' = a'(p', p')$. Assuming $a + a' \in A^*$, the two top threads p, p' can synchronize, after what they will fork into two copies of themselves, which can synchronize too, etc. Clearly p_0 has countably many computation traces, therefore we do need to deal with countable state spaces.

3.1 Memories and Transitions

Reversibility is obtained by adjoining memory stacks to processes in order to record transitions. One pushes a sibling identifier on the memory, when forking, and information about the synch partners, when synching.

Thus we have *fork* transitions (with $n > 0$):

$$\Gamma \cdot (p_1, \dots, p_n) \rightarrow^f \Gamma 1 \cdot p_1, \dots, \Gamma n \cdot p_n$$

where the memory Γ is copied over to each sibling, with a unique integer identifier for each one of them.

And we also have *synch* transitions (with $m > 0$):

$$\begin{aligned} \Gamma_1 \cdot (a_1p_1 + q_1), \dots, \Gamma_m \cdot (a_mp_m + q_m) &\rightarrow_{\mathbf{a}}^s \\ \Gamma_1(\mathbf{\Gamma}, a_1, q_1) \cdot p_1, \dots, \Gamma_m(\mathbf{\Gamma}, a_m, q_m) \cdot p_m \end{aligned}$$

where $\mathbf{\Gamma}$ is short for $\Gamma_1a_1, \dots, \Gamma_ma_m$. This means that each of the threads taking part in the synch records the memory and channel of all participants, its own channel a_i , and its sum remainder q_i (preempted by the synch).

We have labelled the transition arrows for convenience, where $\rightarrow_{\mathbf{a}}^s$ means synch on a multiset $\mathbf{a} \in A^*$. Naturally, the above transitions can happen in any product context.

(NB: Memories are reversed in this notation compared to Ref. [1].)

Communicating processes commonly offer additional constructs: name creation, restrictions, value- or name-passing, etc, which are not considered here. None should make a difference to our main argument, but this has to be verified.

Consider a process with an empty memory $\emptyset \cdot p_0$, and define $p \in \Omega(p_0)$ if p is reachable from $\emptyset \cdot p_0$ by a sequence of transitions, also known as a computation trace, as defined above. It is easy to see that within any $p \in \Omega(p_0)$, memories uniquely identify their respective processes. Thus, both types of transitions store enough information to be reversed unambiguously. In particular, adding the symmetric backward transitions leaves $\Omega(p_0)$ unchanged.

Hereafter, we will suppose that transitions are effectively symmetric.

3.2 Near acyclicity and Simplicity

Now that we have our symmetric transition graph in place, we can return to the acyclicity property that we alluded to in §2.

Consider a computation trace γ , taking place in some $\Omega(p_0)$. If in γ we find a forward move followed immediately by its symmetric backward move, then we can cancel both and obtain a new (shorter) trace γ' with the same end points. Likewise, if in γ we find two synch moves in immediate succession which are triggered by entirely disjoint set of threads (one says the synchs are concurrent), then we can commute the two steps and obtain a new trace γ' (of equal length) with the same end points.

This defines a notion of *causal equivalence* on traces with common end points. We know from Ref. [1] that any two computation traces γ, γ' with the same end points are causally equivalent. We will refer to this as the *labeling property*. (The name is chosen in relation to Lévy's labeling for λ -calculus [12]; indeed, forward reversible CCS is a Lévy-labeling of CCS.) Essentially, this means that the transition graph on $\Omega(p_0)$ is nearly acyclic.

The labeling property implies a convenient property of *simplicity*, namely that, for p, p' in $\Omega(p_0)$, there is *at most one* transition from p to p' .

3.3 An aside on degenerate sums

Actually, for the labeling and simplicity properties to be strictly true, one needs to be precise in the management of degeneracy in sums (as noticed in Ref. [8]). Consider a simple binary synch, with $\mathbf{a} = a + a' \in A^*$:

$$x := \Gamma \cdot ap + r, \Gamma' \cdot a'p' + r' \rightarrow_{\mathbf{a}}^s \Gamma(a, \mathbf{\Gamma}, r) \cdot p, \Gamma'(a', \mathbf{\Gamma}, r') \cdot p' =: y$$

Suppose ap and $a'p'$ occur with multiplicities $\mu(ap)$, $\mu(a'p')$, then there are $\mu(ap)\mu(a'p')$ distinct ways to jump from x to y .

To handle this additional cyclicity/lack of simplicity in the transition graph, one can forbid $p + p$ in sums altogether; or one can recover simplicity by memorising the particular term in the sum that was used (which introduces non-commutative sums that are a bit awkward); or simply incorporate such parallel edges in the causal equivalence - and handle the induced local symmetry factor in the quantitative part of the development (next section and onwards). The latter solution seems preferable, as it is more general.

Then, anticipating somewhat on the next section, the compound rate ratio $\rho(x, y)$ for the above parallel jumps on $a + a'$ is:

$$\rho(x, y) = \frac{k_a^-}{k_a^+} \cdot \frac{1}{\mu(ap)} \cdot \frac{1}{\mu(a'p')} \quad (3)$$

where k_a^+ , k_a^- are the forward and backward rates for a synch on a (which can depend on x and y in general).

The multiplicity factors appearing above are perfectly manageable, but they do make the treatment a little less smooth. Henceforth, we will assume such degeneracies do not happen, and simplicity holds as is.

3.4 Which potential to look for?

Returning to the main thrust, we are now looking for a quantitative version of the above calculus.

As the labeling property guarantees near-acyclicity of the underlying symmetrized transition graph, we know from §2 that any rate function q will lead to a potential V definable as in (2), provided that it respects the causal equivalence, ie the little cyclicity left in reversible CCS.

By which we mean, specifically, that the ratios ρ should verify: 1) $\rho(x, y)\rho(y, z) = \rho(x, y')\rho(y', z)$ when y, y' are intermediate forms obtained by interleaving concurrent synchs in either order, and 2) should be equal when coming from degenerate sums as above. There is no need to say anything for trivial forward/backward loops, as by definition $\rho(x, y) = \rho(y, x)^{-1}$.

The fact that (almost) any rate assignment works notwithstanding, we would like to derive the rates from some potential, for reasons explained in the introduction. This raises the question of what a good potential is. Two requirements stand out. Firstly, the potential should be such that the implied dynamics is implementable concurrently and does not require any global knowledge of the state of the system. Secondly, it should converge. Besides, one might also want a potential that is invariant under natural syntactic isomorphisms (eg the numbering of siblings), and such that disjoint sums of processes implies independent dynamics.

The solution we will eventually home in on, ticks all of the above boxes (but we are not going to be sure until §6).

4 Concurrent potentials

We examine now two potentials that seem natural in the light of the discussion right above, and establish that they are implementable concurrently (within the abstract model of concurrency on which CCS relies, that is). Convergence issues are investigated in the next two sections.

Throughout this section we fix an initial process $\emptyset \cdot p_0$, and a real-valued *energy vector* indexed over A^\star , and written ϵ .

4.1 Total stack size potential

The first potential we consider is defined inductively on the syntax of a reversible process in $\Omega(p_0)$:

$$\begin{aligned} V_1(p_1, \dots, p_n) &= V_1(p_1) + \dots + V_1(p_n) \\ V_1(\Gamma \cdot p) &= V_1(\Gamma i) = V_1(\Gamma) \\ V_1(\Gamma(\mathbf{I}, a, q)) &= V_1(\Gamma) + \epsilon_a \end{aligned}$$

with $\mathbf{I} = \Gamma_1 a_1, \dots, \Gamma_m a_m$ and $\mathbf{a} = a_1 + \dots + a_m$.

Equivalently, $V_1(p)$ is the inner product $\langle \epsilon, \tilde{\Gamma}(p) \rangle$, where $\tilde{\Gamma}(p)(\mathbf{a})$ is the number of occurrences of \mathbf{a} in p ; $\tilde{\Gamma}(p)$ can be seen as a forgetful and commutative projection of the memory structure of p .

Note that $V_1(\emptyset \cdot p_0) = 0$ with this definition; one can always choose a zero energy point in the (strongly) connected component of the initial state, and it is natural to choose the initial state itself.

For each of the two types of transitions, we can easily compute the energy balance:

$$\begin{aligned} \Delta V_1 &= (n-1)V_1(\Gamma) && n\text{-ary fork with memory } \Gamma \\ \Delta V_1 &= m\epsilon_a && \text{synch on } \mathbf{a} \end{aligned}$$

Now, we need to understand how these constrain the rate function. This is analogous to what we have done earlier with (1) in §2.3.

Let us write k_f^-, k_f^+ for backward and forward forking rates, and k_a^-, k_a^+ for backward and forward synching rates. For a fork, and by the simplicity property, the constraint translates into $\log \rho(x, y) = \log(k_f^-/k_f^+) = (n-1)V_1(\Gamma)$. A possible solution is:

$$\begin{aligned} k_f^- &= 1 \\ k_f^+ &= e^{-(n-1)V_1(\Gamma)} \end{aligned}$$

This is an entirely local solution, as the increasing reluctance to fork only depends on the local memory of the process of interest (and the number of siblings, but that can be statically controlled). Similarly, for a synch, the constraint is $\log \rho(x, y) = \log(k_a^-/k_a^+) = m\epsilon_a$. A possible solution is:

$$\begin{aligned} k_a^- &= 1 \\ k_a^+ &= e^{-m\epsilon_a} \end{aligned}$$

not only this is local, but in contrast with the fork case, the assignment does not depend on the memories of the synching processes.

Note that there are many other solutions compatible with V_1 .

4.2 Total synch potential

Perhaps the most natural potential is the following.

Given a path γ from $\emptyset \cdot p_0$ to p :

$$V_0(p) = \sum_{\mathbf{a} \in A^*} \sum_{x \rightarrow_{\mathbf{a}}^s y \in \gamma} (-1)^{v(s)} \epsilon_{\mathbf{a}}$$

where $v(s) = \pm 1$ depending on whether the synch is forward or backward. This V_0 is based on the idea that only communication costs, and forking is done at constant potential. As for V_1 , $V_0(\emptyset \cdot p_0) = 0$.

Clearly, this definition is independent of the choice of γ . Indeed, by the labeling property, any γ, γ' linking $\emptyset \cdot p_0$ to p are convertible by swaps of concurrent synchs, and trivial cancellations, both of which leave V_0 invariant. The corresponding constraints can be met by a locally implementable rate function, eg, $k_f^- = k_f^+$, and $k_{\mathbf{a}}^-/k_{\mathbf{a}}^+ = \exp(\epsilon_{\mathbf{a}})$. (We could add a term to V_0 to count the forks as well.)

Differently from V_1 , there is no straightforward inductive formula for $V_0(p)$, as to compute it one essentially needs to replay a reduction to p .

4.3 V_1 vs. V_0

Let us compare the potentials on two simple examples. Below, we suppose $\mathbf{a} = a + a'$, $\mathbf{b} = b + b'$ in A^* , and $\epsilon_{\mathbf{a}} > 0$; we do not represent the entirety of the memory elements, just what we need to compute the V s.

Here is a first example:

$$\begin{aligned} \emptyset \cdot a(a, b, a', b'), a' &\rightarrow 0\mathbf{a} \cdot (a, b, a', b'), 1\mathbf{a} \cdot - \\ &\rightarrow 0\mathbf{a}0 \cdot a, 0\mathbf{a}1 \cdot b, 0\mathbf{a}2 \cdot a', 0\mathbf{a}3 \cdot b', 1\mathbf{a} \cdot - \\ &\rightarrow 0\mathbf{a}0\mathbf{a} \cdot -, 0\mathbf{a}1\mathbf{b} \cdot -, 0\mathbf{a}2\mathbf{a} \cdot -, 0\mathbf{a}3\mathbf{b} \cdot -, 1\mathbf{a} \cdot - = p \end{aligned}$$

and we get:

$$V_0(p) = 2\epsilon_{\mathbf{a}} + \epsilon_{\mathbf{b}} < 7\epsilon_{\mathbf{a}} + \epsilon_{\mathbf{b}} = V_1(p)$$

We can use the expansion law, replacing a, b with $ab + ba$, and similarly for a', b' in p_0 , and get a variant of the above trace:

$$\begin{aligned} \emptyset \cdot a(ab + ba, a'b' + b'a'), a' &\rightarrow 0\mathbf{a} \cdot (ab + ba, a'b' + b'a'), 1\mathbf{a} \cdot - \\ &\rightarrow 0\mathbf{a}0 \cdot (ab + ba), 0\mathbf{a}1 \cdot (a'b' + b'a'), 1\mathbf{a} \cdot - \\ &\rightarrow 0\mathbf{a}0\mathbf{a}\mathbf{b} \cdot -, 0\mathbf{a}1\mathbf{a}\mathbf{b} \cdot -, 1\mathbf{a} \cdot - = p' \end{aligned}$$

with:

$$V_0(p) = V_0(p') = 2\epsilon_{\mathbf{a}} + \epsilon_{\mathbf{b}} < 4\epsilon_{\mathbf{a}} + 2\epsilon_{\mathbf{b}} = V_1(p') < V_1(p)$$

We see that V_1 , unlike V_0 , is truly concurrent in the sense that it is sensitive to sequential expansions. In fact, according to V_1 , an expanded form using a sum is cheaper by an amount of $V_1(I)$; a sequentialized version is bolder in its search (and the backward options are fewer).

In general, $V_0 \leq V_1$, as a synch performed on the way to p is visible at least once in a memory in p (in fact, at least $|\mathbf{a}|$ for a synch on \mathbf{a}); and $V_0(p) = V_1(p)$ if p has only forks with $n \leq 1$.

So which potential should one prefer? Both seem equally good, but the next section will tell a very different story.

5 Explosive growth

Any potential V partitions $\Omega(p_0)$ into its level sets $\Omega_v(p_0)$. That is to say, $\Omega_v(p_0)$, sometimes simply written Ω_v , is the (finite) set of ps reachable from $\emptyset \cdot p_0$, and such that $V(p) = v$.

Among other things, to address the convergence issue, we will need to control the cardinality of $\Omega_v(p_0)$, which by the labeling property, is the number of traces (up to causal equivalence) γ leading to $\Omega_v(p_0)$.

Let us try to see how this plays out with our earlier example, $p_0 = p, p'$, with $p = a(p, p)$, $p' = a'(p', p')$ (or isomorphically $q_0 = aq_0, a'q_0$).

Call ϕ_k the following trace (synch partners not represented in memories):

$$\begin{aligned} \emptyset \cdot p_0 &\xrightarrow{f} 0 \cdot p, 1 \cdot p' \\ &\xrightarrow{fs} 0a0 \cdot p, 0a1 \cdot p, 1a0 \cdot p', 1a1 \cdot p' \\ &= 0a0 \cdot a(p, p), 0a1 \cdot a(p, p), \\ &\quad 1a0 \cdot a'(p', p'), 1a1 \cdot a'(p', p') \\ &\xrightarrow{fs} 0a0a0 \cdot p, 0a0a1 \cdot p, 0a1a0 \cdot p, 0a1a1 \cdot p, \\ &\quad 1a0a0 \cdot p', 1a0a1 \cdot p', 1a1a0 \cdot p', 1a1a1 \cdot p' \\ &\dots \\ &\xrightarrow{fs} \prod_{w \in \{0,1\}^k} 0w(\mathbf{a}) \cdot p, \prod_{w \in \{0,1\}^k} 1w(\mathbf{a}) \cdot p' = p_k \end{aligned}$$

where $w(x)$, for $w \in \{0,1\}^k$, is defined as the fair interleaving of w and x^k - where x begins, eg $01(\mathbf{a}) = a0a1$.

Note that ϕ_k is maximally synchronous, in the sense that synchronizations are all intra-generational. In this respect it is a very peculiar trace, and one which is easy to compute with.

As the computation unfolds symmetrically, ϕ_k has $2^k - 1$ synchs, and its end process p_k has 2^{k+1} threads, and each has a memory where \mathbf{a} occurs k times.

Hence process p_k has respective potentials:

$$V_0(p_k) = (2^k - 1)\epsilon_{\mathbf{a}} \leq k2^{k+1}\epsilon_{\mathbf{a}} = V_1(p_k)$$

Non-causally equivalent realizations of ϕ_k (where synch partners are chosen differently), can be obtained by picking different intra-generational matchings. Each choice leads to distinct end processes p_k , all with the same V_1 and V_0 potentials - and thus all are in the intersection of $\Omega_{V_0(p_k)}$ and $\Omega_{V_1(p_k)}$.

There are $\prod_{0 \leq h < k} 2^h!$ distinct such ϕ_k s, hence, using $n^n e^{-n} \leq n!$, we get the following lower bound on the cardinality of $\Omega_{V_0(p_k)}$ and $\Omega_{V_1(p_k)}$:

$$(2^{k-1})^{2^{k-1}} e^{-2^{k-1}} \leq 2^{k-1}! \leq \prod_{0 \leq h < k} 2^h! \quad (4)$$

Thus $\log |\Omega_{V_0(p_k)}|$ and $|\log \Omega_{V_1(p_k)}|$ grow asymptotically faster than $k2^{k-1} \log 2$.

This entropic term will trump the term opposed by V_0 which is asymptotically equivalent to $-2^k \epsilon_{\mathbf{a}}$. The inescapable conclusion is that, no matter how costly a synch on $a + a'$ is made to be, V_0 will diverge, and, concretely, the process p_0

will undergo an infinite growth if it follows this potential. So, we can forget V_0 for infinite state spaces.

On the other hand, V_1 's term is $-k2^{k+1}\epsilon_{\mathbf{a}}$ which can control our lower bound of the entropic term, if $4\epsilon_{\mathbf{a}} > \log 2$. So V_1 might still work.

Now (4) only provides a lower bound. As there are many other traces, using extra-generational matchings, that might end up in the same level set, it is hard to know how sharp it is. And, anyway, this is just an example. We have yet to prove that for all p_0 , there are suitable choices of ϵ , that will make V_1 converge. This is what we do in the next section.

6 Main statement

We need a couple of combinatorial lemmas.

Lemma 1. *Suppose $k > 0$ and $\sum_{i=1}^{i=k} n_i = n$, then $\sum_i n_i \log n_i \geq n \log(n/k)$.*

Proof.

$$\begin{aligned} \sum_i n_i \log n_i &= -n \sum_i -(n_i/n) \log n_i/n + n \log n \\ &= -n S(n_i/n) + n \log n \\ &\geq -n \log k + n \log n \end{aligned}$$

where, in the last step, we use $S(n_i/n) \leq \log k$, the usual upper bound on the entropy over a finite set $\{1, \dots, k\}$. \square

Lemma 2 (lower bound on tree depth). *Consider the set of trees t with maximal branching β , then for some $c > 0$, $\|t\| := \sum_{u \in t^\circ} d(u) \geq c \cdot n \log n$ with $n = |t^\circ|$ the number of internal nodes of t . Specifically, $c = 1/\log 4\beta$ works.*

Proof. For $n = 0, 1$, the lower bound holds for any c as the rhs is 0.

Suppose $n \geq 2$. We partition t into its k immediate subtrees with non-zero internal nodes, $n_i > 0$; as $n \geq 2$, we know that $k > 0$.

$$\begin{aligned} \|t\| &= \sum_i \sum_{u_i \in t_i^\circ} d_i(u) + 1 \\ &= \sum_i n_i + \|t_i\| \\ &= n - 1 + \sum_i \|t_i\| \\ &\geq n - 1 + c \sum_i n_i \log n_i && \text{by induction} \\ &\geq n - 1 + c(n - 1) \log((n - 1)/k) && \text{by Lemma 1} \\ &= (n - 1)(1 - c \log k) + c(n - 1) \log(n - 1) \end{aligned}$$

So we need to find c such that for $n \geq 2$:

$$(n - 1)(1 - c \log k) + c(n - 1) \log(n - 1) \geq cn \log n$$

Set for $x \geq 1$:

$$g(x) = (x - 1)(1 - c \log k) + c(x - 1) \log(x - 1) - cx \log x$$

We have $g(1^+) = 0$, $g'(x) = 1 - c \log(xk/(x-1)) \geq 0$ as soon as $c \leq 1/\log(xk/(x-1)) =: h(x)$. $(1/h)'(x) = -1/(x(x-1)) \leq 0$ for $x \geq 1$, so h is increasing on $(1, \infty)$, and if we take $c \leq 1/\log 2k$, $c \leq h(x)$ for $x \geq 2$, and g increases for $x \geq 2$.

Now $g(2) = 1 - c \log 4k \geq 0$ as soon as $c \leq 1/\log 4k$.

Set $c = 1/\log 4\beta$ where β is the maximum branching of t .

Clearly $\beta \geq k$, so we have $g(x) \geq 0$ for $x \geq 2$. \square

Note that the proof gives explicit control in terms of the maximal branching degree β , namely $\|t\| \geq n \log n / \log 4\beta$. If one allows arbitrary branching, any tree t with all $n-1$ nodes right below the root of t verifies $\|t\| = n-1$. While the (best) inequality says $n-1 \geq n \log n / (2 \log 2 + \log(n-1))$, which is true indeed. Clearly the inequality is more interesting if one imposes a maximal branching degree.

It is possible to specialize the inequality (which is central to the main convergence result below) to the case of balanced trees. As these minimize depth for a given number of internal nodes (else one can always move groups of sibling leaves upwards, and in so doing decrease the potential), they should be a good test of the sharpness of our lower bound. We consider only binary trees to keep computations simpler.

Lemma 3 (balanced binary case). *Let t_k be the balanced binary tree with 2^k leaves, equivalently $n = 2^k - 1$ internal nodes, $k \geq 0$:*

$$\|t_k\| = \sum_{1 \leq i < k} i 2^i = (n+1) \log(n+1) / \log 2 - 2n$$

Proof. The formula holds for $k = 0, 1$, and $\|t_0\| = \|t_1\| = 0$.

Suppose k , the number of ‘generations’ in t_k , is strictly positive.

As t_k has 2^k leaves, $2^k - 1$ internal nodes, we have by induction on the last generation of the tree:

$$\begin{aligned} \|t_1\| &= 0 \\ \|t_{k+1}\| &= k 2^k + \|t_k\| \end{aligned}$$

Therefore $\|t_k\| = \sum_{1 \leq i < k} i 2^i$.

Set $\phi_k(x) = (x^k - 1)/(x - 1) = \sum_{0 \leq i < k} x^i$. We have:

$$\phi'_k(x) = \sum_{1 \leq i < k} i x^{i-1} = ((k-1)x^k - kx^{k-1} + 1)/(x-1)^2$$

Hence $\sum_{1 \leq i < k} i 2^i = 2\phi'(2) = 2((k-1)2^k - k2^{k-1} + 1) = k2^k - 2^{k+1} + 2$. \square

Therefore, in this case the inequality of Lemma 2 amounts to saying that $(n+1) \log(n+1) / \log 2 - 2n \geq n \log n / 3 \log 2$ (with $\beta = 2$) or equivalently:

$$(n+1) \log(n+1) \geq 1/3 \cdot n \log n + (2 \log 2)n$$

which is indeed true for $n \geq 0$ and a rather sharp estimate for small values of n . This means that the lower bound provided by Lemma 2 is good.

6.1 Lower bound on the potential

With Lemma 2 in place, we can bound below the energy of a process/trace with a given number of synchs. But first, we need to fix some notations.

As in §4-5, we suppose given a process p_0 , and consider only computation traces starting from the initial state $\emptyset \cdot p_0$.

We write $T(n)$ for the set of traces containing n synchs, considered up to causal equivalence, and set $\epsilon_m := \min_{\mathbf{a} \in A^*} \epsilon_{\mathbf{a}}$, for the minimal energetic cost of a synch. We also write $\Omega_n(p_0)$ for the set of processes reachable in n synchs.

As traces originating from the initial state are isomorphic to their end points, ie $T(n) \sim \Omega_n(p_0)$, we will treat traces from $\emptyset \cdot p_0$ and processes in $\Omega(p_0)$ as nearly synonymous.

We suppose given an upper bound α on the number of processes that can synchronize at once during an execution of $\emptyset \cdot p_0$. Eg $\alpha = \max_{\mathbf{a} \in A^*} |\mathbf{a}|$, the maximal synch size in A^* .

We write δ for the thread *increment* of a particular forking event in a trace. This means that one replaces one thread with $\delta + 1$ ones. We suppose also that we are given two numbers (eg obtained from a trivial syntactic analysis) such that $\beta_- \leq \delta + 1$ and $\delta \leq \beta_+$ always hold. If $\beta_- > 1$, then $\delta > 0$, which amounts to saying that the number of threads always increase under fork.

We can establish the following lower bound on the potential:

Lemma 4. *Suppose $\beta_- > 1$, $\epsilon_m > 0$, $p \in \Omega_n(p_0)$:*

$$\frac{\epsilon_m}{\log 4 + \log(\beta_+ + 1)} \cdot n \log n \leq V_1(p)$$

Proof. Consider the set $U(n)$ of trees with n internal nodes labeled in A^* . It clearly makes sense to extend the definition of V_1 to such labeled trees.

Consider $t \in U(n)$, $n > 0$, and u an internal node in t with label \mathbf{a} , all the children of which are leaves (so that u is on the boundary of the set of internal nodes). Define $t \setminus u \in U(n-1)$ as the tree obtained from t by erasing the $\delta(u) + 1$ leaves right below u (as well as u 's label).

Write $\tilde{\Gamma}(u)$ for the multiset of occurrences of labels above u , and $d(u)$ for the depth of u in t (as we have already done above). We can bound below the difference of potential incurred by erasing the $\delta(u) + 1$ children of u :

$$\begin{aligned} V_1(t) - V_1(t \setminus u) &= (\delta(u) + 1)\epsilon_{\mathbf{a}} + \delta(u)\langle \epsilon, \tilde{\Gamma}(u) \rangle \\ &\geq \epsilon_m \delta(u) d(u) \\ &\geq \epsilon_m d(u) \end{aligned}$$

We have used $\delta(u) > 0$.

It follows that V/ϵ_m decreases by chunks of at least $d(u)$ for each deletion of a node on the internal boundary, therefore $V(t)/\epsilon_m \geq \sum_i d(u_i) =: \|t\|$, and we can apply Lemma 2, to obtain $V(t)/\epsilon_m \geq n \log n / \log 4(\beta_+ + 1)$.

As any p in $\Omega_n(p_0)$ projects to a labeled tree in $U(n)$, by forgetting the information on communication partners and remainders, and this forgetful operation leaves V_1 invariant, the statement follows. \square

With the same notations as in the proof above, consider a leaf $v \in t$, and define $t(u, v)$ as the new tree obtained by moving the leaves below u , to below v ; clearly, if $d(v) < d(u)$, $d(t(u, v)) < d(t)$. If no such move exists, by definition t is balanced. So, as alluded to earlier, the lower bound we get for the potential is obtained for balanced concurrent structures of execution - and as they have lower energies, they will be highly favoured by the dynamics. In other words, our potential V_1 penalizes depth - one could call it a *breadth-first* potential - and different threads will tend to stay synchronous.

We turn now to the other pending question, namely that of binding above the entropy (that is to say the logarithm of the cardinality) of the set of traces of a given number of synchs.

6.2 Upper bound on the number of traces

Dually to Lemma 4, which a lower bound on potentials, we can derive an upper bound on entropies:

Lemma 5. *For large ns , $\log |T(n)| \leq \beta_+ \alpha^2 O(n \log n)$*

Proof. By induction on n , there are at most $\delta_0 + n\beta_+\alpha$ threads in the end process of a trace in $T(n)$, as each synch adds at most $\delta\alpha$ new threads, where we have written δ_0 for the initial number of threads in p_0 .

Any trace with $n+1$ synchs can be obtained (perhaps in many ways but we are looking for an upper bound) by synching one of length n , so $|T(n+1)| \leq |T(n)|(\delta_0 + n\beta_+\alpha)^\alpha$. As $T(0) = 1$, we get $\log |T(n)| \leq \alpha \log(\delta_0 + n\beta_+\alpha)!$.

Since:

$$1 - n + n \log n \leq \log n! \leq 1 - n + (n+1) \log n$$

it follows that $\log(\delta_0 + n\beta_+\alpha)! \sim \beta_+\alpha O(n \log n)$. \square

The first inequality is sharp if all synchs are possible, and one has the maximal thread count, and no sums (as they decrease the number of matches), which is exactly the situation of the explosive example of §5.

As the arithmetic progression that gives rise to the factorial, samples the factorial only with frequency $1/\delta\alpha$ (this is sometimes called a shifted j -factorial [18, p.46], where $j = \alpha\delta$, and the shift is δ_0 in our example), it seems the upper bound above could be improved. But, if we return to the maximal synchronous traces computed in §5, we see that the bound above is quite sharp, so it seems unlikely.

6.3 Convergence

Now we can put both bounds to work to get the convergence of our potential.

Proposition 1. *Suppose $1 < \beta_-$, and $\beta_+\alpha^2 \log(4(\beta_+ + 1)) < \epsilon_m$, then:*

$$Z(p_0) := \sum_{p \in \Omega(p_0)} e^{-V_1(p)} < +\infty$$

Proof. We can partition $Z(p_0)$ by number of synchs:

$$\begin{aligned} Z(p_0) &:= \sum_n \sum_{p \in \Omega_n(p_0)} e^{-V_1(p)} \\ &\leq \sum_n e^{-\epsilon_m n \log n / \log 4(\beta_+ + 1)} \cdot |T(n)| \quad \text{by Lemma 4} \end{aligned}$$

By Lemma 5, the logarithm of the general term of the upper series is equivalent to $-\epsilon_m n \log n / \log 4(\beta_+ + 1) + \beta_+ \alpha^2 O(n \log n)$, so both series converge if $\epsilon_m > \delta \alpha^2 \log(4(\beta_+ + 1))$. \square

We can summarise our findings:

Theorem 1. *Consider a reversible process $\emptyset \cdot p_0$ equipped with a rate function q which satisfies the §4.1-constraints of the V_1 potential; suppose that for any fork event in $\Omega(p_0)$, the thread increment δ verifies $0 < \delta \leq \beta_+$ for some constant β_+ , and assume that for $\mathbf{a} \in A^*$, V_1 stipulates a synch cost $\epsilon_{\mathbf{a}}$ which is at least $\max_{\mathbf{a} \in A^*} |\mathbf{a}|^2 \cdot \beta_+ \log(4(\beta_+ + 1))$.*

Then q has an equilibrium on $\Omega(p_0)$ defined as $\pi(p) \propto e^{-V_1(p)}$.

6.4 Discussion

This concludes the comparison of the potentials introduced in §4. Unlike the potential V_0 which is not enough to control growth (as we have seen in §5), V_1 which forces forking costs to increase with the size of the local memory will, when parametrized suitably, lead to an equilibrium.

Note that 1) the condition given above on the minimal energy cost is a sufficient one, and might not be necessary (we don't know at the time of writing), 2) in particular, to obtain refined effects on the equilibrium population of certain level sets, and therefore modulate the search, one might need more flexibility. Whether this is possible and useful remains to be seen.

As said in §2, for general reasons in the theory of continuous-time Markov chains, the symmetry of the underlying transition graph guarantees that the probabilistic state of the system converges to the invariant probability $\pi(p)$ defined above. This avoids co-Zenoid situations where the probability of return to a p is 1, while the mean return time is actually infinite. Here we are guaranteed finite mean return times. This form of probabilistic termination, which does not lose itself in infinite branches, is the technical definition of exhaustivity. It is easy to construct examples (and we have seen one earlier in §5) where one has an invariant measure, and almost certain returns, but infinite mean return times. This is why it is fundamental to prove the convergence of $Z(p_0)$.

The restriction to a minimum branching degree $\beta^- > 0$ is not very strong, as one can always use some padding with null processes to make the minimal forking degree higher. Nevertheless, it would be nice to have a more elegant way to deal with non-expansive forks, as surely they cannot seriously stand in the way of convergence.

7 Conclusion

There has been a lingering desire in concurrency theory for a metaphor of computation as a physical process. We present here evidence that one can promote this metaphor to an operational conceptualization of a certain, arguably rather abstract, type of distributed programming. Specifically, we have shown how a slightly generalized version of reversible CCS can be equipped with a distributed potential. This potential is parametrized by costs for different types of synchronizations, in a way that any process eventually reaches a probabilistic equilibrium over its reachable state space - provided that the rate at which processes fork decreases exponentially as a function of the size of the local history.

Much remains to be done.

It would be interesting to see how our findings can be adjusted to a less abstract model of distributed computing, and whether one can find examples where this technique solves problems. We intend to seriously pursue such examples in the future, perhaps in the field of multi-party and simultaneous transactions. To talk about efficiency, solutions, and examples, one needs to make room for the inclusion of irreversible synchronizations expressing success states. This, we have done already in the qualitative case [2], and the extension should be straightforward.

Another natural companion question is to optimize parameters for efficiency of the search mechanism. Instead of minimizing the time to irreversible synchs (aka commits), which begs the question, one could use as a proxy the following objective. Namely to maximize the *equilibrium residency time* of the search reflected on success states ps which live on the boundary ∂X of the fully reversible state space:

$$\operatorname{argmax} \epsilon. \sum_{p \in \partial X} \pi(\epsilon, p) = \int \mathbf{1}_{\partial X} d\pi$$

(by reflected search we mean that irreversible synchs are de-activated, and, hence, the process bounces off the success boundary) where π is the equilibrium probability, and ϵ its energy vector. Such quantities are nice optimization targets as they can be estimated via the ergodic theorem by the averages $\frac{1}{n} \sum \mathbf{1}_{\partial X}(X_k)$, ie the empirical time of residence in a success state (under reflective regime). From there on, it seems one might be in a good position to interface with machine learning techniques to discover efficient parametrizations.

One can also think of this result as a termination one, more in line with the tradition of rewriting and proof-theory. Of course, it is a kind of termination, just as in actual physical systems, which does not mean the complete disappearance of any activity in the system, but rather the appearance of a steady or stable form of activity. As such, it introduces a discourse on resources which is not the one commonly offered in relation to termination proofs in the context of programming languages and rewriting systems, where one tries to limit copies, sizes, and iterations. There has been a thread of research studying termination by various typing systems in process languages (for a recent example, see Ref. [6]) - here we propose what seems a fundamentally different way to achieve the same, in a probabilistic setting, and one wonders if perhaps there is a fruitful relationship to be established.

Finally, it seems that one could squeeze more out of the statistical physical metaphor, and start thinking about the concepts of temperature (which here is degenerate, as it only measures the energy scale) as they are used in the context of sequential simulated annealing algorithms, where the potential can change over time.

References

1. Danos, V., Krivine, J.: Reversible communicating systems. In: Gardner, P., Yoshida, N. (eds.) CONCUR 2004, 15th International Conference, London. Lecture Notes in Computer Science, vol. 3170, pp. 292–307. Springer (2004)
2. Danos, V., Krivine, J.: Transactions in RCCS. In: Abadi, M., de Alfaro, L. (eds.) CONCUR 2005, 16th International Conference, San Francisco. Lecture Notes in Computer Science, vol. 3653, pp. 398–412. Springer (2005)
3. Danos, V., Krivine, J., Tarissan, F.: Self-assembling trees. *Electr. Notes Theor. Comput. Sci.* 175(1), 19–32 (2007)
4. Danos, V., Oury, N.: Equilibrium and termination. In: Cooper, S.B., Panangaden, P., Kashefi, E. (eds.) *Proceedings Sixth Workshop on Developments in Computational Models: Causality, Computation, and Physics*. EPTCS, vol. 26, pp. 75–84 (2010)
5. Danos, V., Oury, N.: Equilibrium and termination II: the case of *Petri Nets*. *Mathematical Structures in Computer Science* (2011), to appear
6. Demangeon, R., Hirschhoff, D., Kobayashi, N., Sangiorgi, D.: On the complexity of termination inference for processes. *Trustworthy Global Computing* pp. 140–155 (2008)
7. Diaconis, P.: The Markov chain Monte-Carlo revolution. *AMS* 46(2), 179–205 (2009)
8. Krivine, J.: *Algèbres de Processus Réversible - Programmation Concurrente Déclarative*. Ph.D. thesis, Université Paris 6 & INRIA Rocquencourt (Nov 2006)
9. Krivine, J.: A verification algorithm for declarative concurrent programming. *CoRR* abs/cs/0606095 (2006)
10. Kwiatkowski, M., Stark, I.: The continuous π -calculus: A process algebra for biochemical modelling. In: *Computational Methods in Systems Biology: Process of the Sixth International Conference CMSB 2008*. pp. 103–122. No. 5307 in *Lecture Notes in Computer Science*, Springer-Verlag (2008)
11. Lanese, I., Mezzina, C.A., Stefani, J.B.: Reversing higher-order pi. In: Gastin, P., Laroussinie, F. (eds.) CONCUR. *Lecture Notes in Computer Science*, vol. 6269, pp. 478–493. Springer (2010)
12. Lévy, J.J.: *Réductions correctes et optimales dans le λ -calcul*. Ph.D. thesis, Thèse de doctorat d’État, Université Paris 7 (1978)
13. Metropolis, N., Rosenbluth, A., Rosenbluth, M., Teller, A., Teller, E., et al.: Equation of state calculations by fast computing machines. *The journal of chemical physics* 21(6), 1087 (1953)
14. Milner, R.: *Communication and concurrency*. International Series on Computer Science, Prentice Hall (1989)
15. Norris, J.: *Markov chains*. Cambridge University Press (1998)
16. Ollivier, J., Shahrezaei, V., Swain, P.: Scalable rule-based modelling of allosteric proteins and biochemical networks. *PLoS Computational Biology* 6(11) (2010)

17. Prasad, K.V.S.: Combinators and bisimulation proofs for restartable systems. Ph.D. thesis, University of Edinburgh (1987)
18. Schmidt, M.: Generalized j -factorial functions, polynomials, and applications. Journal of Integer Sequences 13(2), 3 (2010)
19. Streater, R.: Statistical dynamics. Imperial College Press (1995)