NAPI Project Cambridge's Workshop, June 2001

A zero-safe's agenda

Roberto Bruni and Ugo Montanari

Dipartimento di Informatica, Università di Pisa
{bruni,ugo}@di.unipi.it

Join calculus, transactions and zero-safe nets

Many different proposals (e.g., Petri nets, Linda and other tuple-space languages, CHAM, Join calculus) for applying formal methods to distributed systems rely on two fundamental design postulates: (1) states are multisets; and (2) elementary actions can atomically fetch (or release) several state components, thus synchronizing them at the event level. However, they often miss a third feature which is instead very useful for programming reliable services, namely the definition of *transactions*, grouping events into work units that either completely succeed or have no effect. Note that we are interested in transactions denoting atomic computations *in a concurrent and causal scenario* (as opposed to a sequential one).

Though ad-hoc transaction mechanisms are integrated in languages such as BizTalk Orchestration and JavaSpaces, we think the issue deserves a uniform treatment along the many calculi proposed in the literature. Building on points (1) and (2) above, it is in fact possible to define a taxonomy of models with increasing expressiveness.[1] Starting from PT Petri nets, where tokens are seen as non structured data, one may move to *colored* or *high level* nets, whose pre- and postsets are based on domains of token values. In *reconfigurable* nets, a postset may depend on the values got from the preset (i.e., presets are static, but postsets can change), and thus network reconfigurability can be accounted for. In *dynamic* nets, not only a firing can modify the current marking, but can also increase the set of transitions (i.e., the control). In Join the situation is analogous to dynamic nets, because new agents can be generated at run-time by the application of reduction steps.

When adding transactions, one has two main issues to handle: *(i) semantics*, a sound theoretical characterization of transactions, making it possible to study their properties and the way in which they can be combined together (e.g., in parallel, sequentially, serializability matters); *(ii) algorithms*, the development of distributed interpreters, which are able to implement transactions in a consistent way with the semantic level. Hence, it is convenient to select a formal language where these issues can be easily dealt with also at the syntax level. *Zero-safe nets* have been proposed to make these intuitions concrete for several flavors of PT Petri nets.

[1] We use the term *expressiveness* with a different meaning from just computational power: the presence of language constructs for a direct modeling and studying of certain aspects.

This document overviews the zero-safe approach and contains a complete list of references to related literature. Most papers are available at: http://www.di.unipi.it/~bruni

The simplest way to synchronize the execution of transitions in PT Petri nets is via token exchanging over a suitable subset of places. Zero-safe nets exploit this idea by distinguishing between *stable* places (the ordinary repositories for distributed resources, which define observable system states) and *zero safe* places that cannot contain tokens in any observable state. The firing of a transition will possibly put tokens in zero-safe places, beginning a transaction; these tokens (called *zero tokens*) can be used to coordinate the transaction. All zero tokens must be removed to commit the transaction. Moreover, all the stable tokens produced during the transaction are effectively released only when the transaction ends, so to avoid interference between causally dependent transactions. Thus, all the stable resources fetched by the firings involved in the transaction must be present in the initial stable marking.

What we described above is, to some extent, the low-level view of the model. At the abstract level, transactions can be seen as ordinary transitions. This viewpoint yields a PT Petri net N, which is the abstract counterpart of the zero-safe net B: the places on N are the stable places of B, and each transition of N corresponds to an elementary transaction of B (i.e., a transaction that cannot be decomposed in two smaller disjoint transactions). Note that the net N can become infinite also when B is finite, and that transactions retain all the causal and concurrent information about the synchronized evolution of B.

Zero-safe nets can be used to give a modular presentation of distributed decision making, as any net can be modeled as the abstract counterpart of a free choice zero-safe net. For example, suppose that the presets of two transitions t_1 and t_2 intersect on the place a, then the choice of whether assigning a token in a to t_1 or t_2 can be carried out by introducing two zero safe places a_1 and a_2 that replace a for t_1 and t_2 respectively, together with two transitions s_1 from a to a_1 and s_2 from a to a_2 that, given a token in a, nondeterministically enable t_1 or t_2. Then, the definition of transaction at the semantic level (or the interpreter at the algorithm level) enforces the choice on a followed by the firing of either t_1 or t_2 to be executed atomically. Note that this view does not violate the locality principle, because the interpreter can be as much distributed as before. When viewed in the other direction, this means that the system can be as much distributed as the abstract counterpart of the zero-safe net.

Zero-safe nets have been introduced in [2], where their operational and abstract semantics are defined under the collective token philosophy (which does not distinguish between tokens with different histories in the same place and roughly corresponds to take Best and Deviller's commutative processes as computational entities). Moreover, it is shown that the two semantics can be formulated as universal constructions between a category of zero-safe nets and suitable categories of models. As a running example, the modeling of a multicasting system is illustrated, where a finite zero-safe net originates an abstract net with infinitely many transitions (one for each 1-to-n communication).

In [3] the results of [2] have been extended to deal with the individual token philosophy (which gives a more precise account of causality threads in concurrent computations and roughly corresponds to take Goltz and Reisig's processes as computational entities). The case study of multicasting system is again used as a running example, and this time the abstract net has a transition for each copy policy (e.g. sequential, with maximal parallelism) of the message to be multicast.

The journal paper [6] collects and extends [2,3] by giving a precise comparison of the two approaches (collective vs individual) on the basis of the multicasting system and of the modeling of a simple CCS-like language. A previous presentation of the material in [6] can be found in the Part I of the Ph.D. Thesis of the first author [1], while a tutorial presentation of zero-safe nets under the collective and individual token philosophy is in [4], and a distributed interpreter for zero-safe nets has been proposed in [5], which is based on the ordinary net unfolding. We refer the interested reader to [6,1] also for a discussion on related approaches to net refinement/abstraction.

Two recent papers focus on the mixing of zero-safe nets with *read arcs* (see [7], where the distributed interpreter is extended to deal with read arcs) and with *inhibitor arcs* (cf. [8], where it is also shown how to extend the idea of zero-safe resource to other languages, introducing the possibility of expressing transactions; the case study of a Linda-like language is considered and it is shown that zero-safe nets can provide such language with a straightforward concurrent semantics).

Building on the mentioned analogies between Petri nets and Join calculus, we are currently investigating how well the zero-safe approach and Join can fit together to define a more interesting distributed formal framework with mechanisms of refinement/abstraction, transactions, dynamic network reconfigurations and mobility. The idea is to distinguish two kinds of names: stable and zero. Then, a stable Join term must contain stable messages only, i.e., messages without zero names. The application of a reduction rule whose pattern matches with a subset of the currently available messages can produce: (a) new stable messages; (b) new messages on stable channels but whose data contain zero names; (c) new messages on zero channels (either with stable or zero data); (d) new reduction rules. Since spawned messages on stable channels must be frozen until the commit of the transaction, we forbid the case (b). (Otherwise, after the commit, we will end up in a non-stable situation.) The production of messages on zero channels opens a transaction, where other reduction rules can be used to consume such messages. Several transactions can take place concurrently. To see whether two messages belong to the same ongoing transaction, it is sufficient to check if the events in their histories are connected (in general, it is not necessarily the case that their histories intersect on some event). The commit of a transaction involve the consumption of all the zero messages involved. These assumptions lift the zero-safe approach to a setting where tokens can be structured data, transitions postsets can be reconfigured upon the input data tokens, and the network can dynamically grow. Moreover, since reduction rules can be nested inside Join terms, one can also impose a hierarchical structure on transactions, in such a way that the zero names of, say, level n are stable names for the level $n+1$.

The distributed interpreter of [5] should then be extended to deal with the above issues: in the case of zero-safe nets, the interpreter just applies transitions nondeterministically in a simulation environment, which is committed whenever all the zero tokens involved in a transaction can be consumed. (A possible alternative solution, not discussed in [5], would be to consider backtracking mechanisms based on reversing transition application, to guarantee that all transactions can be either terminated successfully or aborted without garbage).

We plan to investigate both the 'flat' and the 'hierarchical' alternatives and compare their expressiveness.

References

[1] R. Bruni. *Tile Logic for Synchronized Rewriting of Concurrent Systems*. PhD thesis, Computer Science Department, University of Pisa, 1999. `http://www.di.unipi.it/~bruni/publications/PhDThesis.ps.gz`

[2] R. Bruni and U. Montanari. Zero-safe nets, or transition synchronization made simple. In C. Palamidessi and J. Parrow, editors, *Proceedings of EXPRESS'97, 4th workshop on Expressiveness in Concurrency*, volume 7 of *Elect. Notes in Th. Comput. Sci.* Elsevier Science, 1997. `http://www.elsevier.nl/locate/entcs/volume7.html`

[3] R. Bruni and U. Montanari. Zero-safe nets: The individual token approach. In F. Parisi-Presicce, editor, *Proceedings of WADT'97, 12th workshop on Recent Trends in Algebraic Development Techniques*, volume 1376 of *Lect. Notes in Comput. Sci.*, pages 122–140. Springer Verlag, 1998. `http://www.di.unipi.it/~bruni/publications/wadt97.ps.gz`

[4] R. Bruni and U. Montanari. Zero-safe nets: Composing nets via transition synchronization. In H. Weber, H. Ehrig, and W. Reisig, editors, *Int. Colloquium on Petri Net Technologies for Modelling Communication Based Systems*, pages 43–80. Fraunhofer Gesellschaft ISST, 1999. `http://www.di.unipi.it/~bruni/publications/cpnt99.ps.gz`

[5] R. Bruni and U. Montanari. Executing transactions in zero-safe nets. In M. Nielsen and D. Simpson, editors, *Proceedings of ICATPN 2000, 21st Int. Conf. on Application and Theory of Petri Nets*, volume 1825 of *Lect. Notes in Comput. Sci.*, pages 83–102. Springer Verlag, 2000. `http://link.springer.de/link/service/series/0558/tocs/t1825.htm`

[6] R. Bruni and U. Montanari. Zero-safe nets: Comparing the collective and individual token approaches. *Inform. and Comput.*, 156:46–89, 2000. `http://www.idealibrary.com/links/toc/inco/156/1/0`

[7] R. Bruni and U. Montanari. Transactions and zero-safe nets, 2001. To appear in *Advances in Petri nets: Unifying Petri Nets*. `http://www.di.unipi.it/~bruni/publications/uapnzs.ps.gz`

[8] R. Bruni and U. Montanari. Zero-safe net models for transactions in Linda, 2001. To appear in *Proceeding of ConCoord 2001, Concurrency and Coordination*. `http://www.di.unipi.it/~bruni/publications/concoord2001.ps.gz`