# An unifying framework for compacting Petri nets behaviors[⋆]

Giovanni Casu and G. Michele Pinna

Dipartimento di Matematica e Informatica, Università di Cagliari, Cagliari, Italy
giovanni.casu@unica.it, gmpinna@unica.it

**Abstract.** Compacting Petri nets behaviors means to develop a more *succinct* representation of all the possible executions of a net, still giving the capability to *reason* on properties fulfilled by the computations of the net. To do so suitable equivalences on alternative executions have to be engineered. We introduce a general notion of *merging relation*, covering the existing approaches to compact behaviors of nets, and we state some properties this kind of relations may satisfy. The classical merging relations, defined on unfoldings, do not in general satisfy the properties one may be interested in, and we propose how to add information to the executions in order to enforce some of these properties.

The behavior of a Petri net can be described in many ways, *e.g* using the *marking graph*, or the set of *firing sequences*, or its *unfolding* (see [1,2] among many others). The notion of unfolding of a net $N$, a net where places (called conditions) and transitions (called events) are labeled with the places and transitions of $N$ ([3,4]), is particularly relevant as it allows to record conflicts and dependencies among the activities modeled with the Petri net $N$. Furthermore, the possibility of finding a finite representation of it (the prefix), has given profitability to the notion, otherwise confined to the purely theoretical modeling realm ([5,6]). However the size of a finite unfolding, even of the prefix, can be too large, hence manageable only with big efforts. Prefixes are obtained *cutting* the unfolding in such a way that each execution represented in the unfolding can be recovered in the prefix itself. The cutting procedure allows to eliminate unnecessary duplications. Still some information may be redundant, for instance the existence of conflicting components leading to isomorphic futures, but with alternative pasts, forces to represent all the possible futures, introducing in this way some avoidable duplications.

The identification of conflicting conditions seems to be a good basis for compacting nets' behaviors. Following this idea some approaches have been

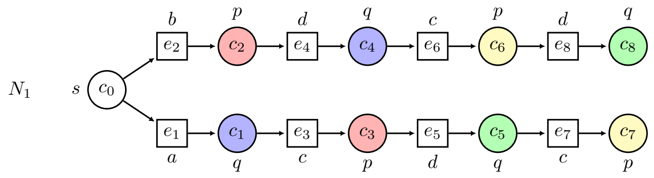
[⋆] Work partially supported by Aut. Region of Sardinia under grant P.I.A. 2013 "NOMAD" and by MIUR PRIN 2010-11 "Security Horizons"

V. Biló, A. Caruso (Eds.): ICTCS 2016, Proceedings of the 17th Italian Conference on Theoretical Computer Science, 73100 Lecce, Italy, September 7–9 2016, pp. 245–250 published in CEUR Workshop Proceedins Vol-1720 at http://ceur-ws.org/Vol-1720

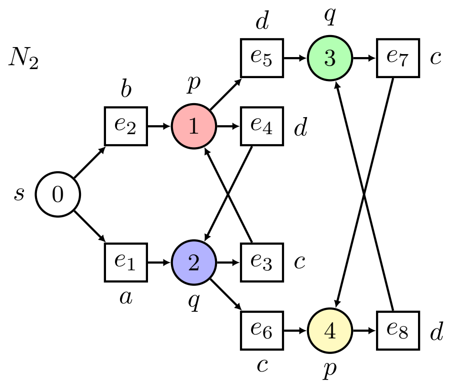
proposed, and these are based on giving precise criteria to identify conflicting conditions in nets which are *acyclic*, *i.e.* the transitive and reflexive closure of the flow relation is a partial order. In the case of *merged process* ([7]) the criterion is that the conditions must be equally labeled and have the same token occurrence (*i.e.* they represent the same token, in the collective token philosophy of [8]) whereas in the case of *trellis processes* ([9]) the criterion is the distance of the equally labeled conditions from the initial conditions (measuring the *time*). Once conditions have been identified, isomorphic *futures* can be identified as well. The identification of conflicting conditions has a *semantics* counterpart: the identification induces an equivalence relation on the different computations leading to these conditions, equivalence driven by the common futures of these computations.

We pursue this idea further, casting it in a general framework. We start choosing a representation of nets behaviors less constrained with respect to the usual notion of *causal net* on which unfoldings are based. Causal nets are acyclic safe nets where conditions may have at most one incoming arc. The uniqueness of incoming arcs, together with the safeness, guarantee that dependencies can be uniquely identified. Conflicts are deduced from conditions having more than one outgoing arcs (implying that various alternatives use that condition). We drop the assumption that each condition has at most one incoming arc, and we add the requirements that each transition in the net can be executed at most once (which is syntactically enforceable) and that restricting the net to all the transitions in a execution we obtain an acyclic net, where each condition has at most one incoming and one outgoing arc. Dependencies can be captured by looking at executions, and some conflicts may still be retrieved by looking at multiple outgoing arcs. We call these nets *unravel* nets. This notion covers the one of causal nets, as these are indeed unravel nets, whereas unravel nets may not be causal ones. Together with the notion of unravel net, we introduce a notion of conflict that it is not based on the syntax, like in causal nets, but on the semantics (the executions of the net), simply stipulating that two conditions are in conflict if they never appear together in an execution.

We can now put forward the general framework, that consists in taking a representation of the behaviors of a given net (in our case a labeled unravel net) and an equivalence relation defined on conditions of the chosen representation of the behaviors. The minimal requirement we put on this relation, which is called *merging relation* is that two different conditions in the relation should be *equally labeled* and in *conflict.*

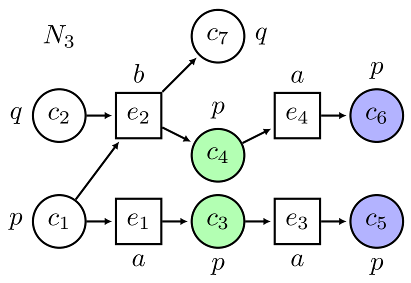


Consider the unravel net $N_1$. Conditions $c_2$ and $c_3$ are in conflict and they have the same label $p$, and similarly for conditions $c_1$ and $c_4$ (here the label is $q$). In the net above the merging relation (denoted with $\sim$) stipulates that $c_2 \sim c_3$, $c_1 \sim c_4$, $c_6 \sim c_7$ and $c_5 \sim c_8$ (reflexive pairs omitted). The relation is identified pictorially with different colors. This is not the unique merging relation definable on this net, we could have chosen this other relation: $c_2 \sim c_7$, $c_1 \sim c_4$, $c_6 \sim c_3$ and $c_5 \sim c_8$ (again reflexive pairs omitted), and clearly the identity relation is a merging relation. Once that a merging relation is fixed, we can *compact* the behavior by merging the conditions in the same equivalence classes and identifying the equally labeled transitions having the same preset and postset.



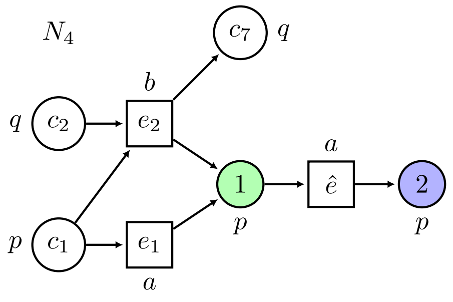
The result of this procedure is the net shown on the left. The condition 0 is the equivalence class of $c_0$, 1 is the equivalence class of $c_2$ and $c_3$, 2 the one of $c_1$ and $c_4$, 3 of $c_6$ and $c_7$ and finally 4 the one of $c_5$ and $c_8$. Transitions with the same labels are not identified as none of them has the same preset and postset. We observe that the net obtained identifying equivalent conditions is not any longer an unravel net. In the execution $e_1$ followed by $e_3$ and $e_4$ the condition 1 is marked twice violating the requirement of being acyclic. The fact that $e_3$ should be followed by $e_5$ and not $e_4$ has been lost in the compaction process.

The notion of merging relation covers the criteria used in merged and trellises processes. In the case of merged and trellises processes the starting point is always a branching process, hence a labeled causal net where dependencies and conflicts can be found syntactically. The criterion to use in case of merged processes is to consider two equally labeled conflicting conditions $c_i$ and $c_j$ as equivalent is that they have the same *token occurrence*, which is defined as the number of conditions labeled as $c_i$ and $c_j$ that are encountered going back to the initial conditions, comprising $c_i$ and $c_j$. In the net $N_1$ the conditions $c_6$ and the condition $c_7$ have both one condition in their past which has the same label, namely $c_2$ and $c_3$ respectively, hence their token occurrence is 2. In the case of trellises processes the starting point is not only a branching processes, but here the nets considered are called *multi-clocks* nets. Multi-clocks nets are the product of various automata where only one place is initially marked and each reachable marking is such that each component has



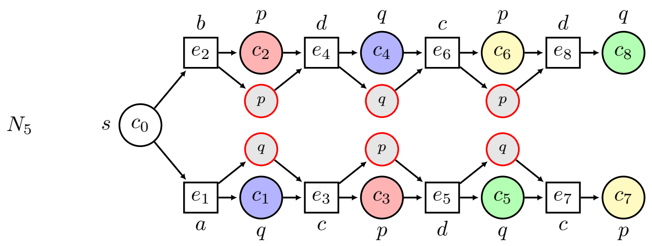
just one place marked. Due to this feature it is possible to identify, for each condition and each execution, the exact time in which the condition holds. The criterion is then the one of considering two equally labeled conflicting conditions $c_i$ and $c_j$ as equivalent is that they have the same *time*, which is defined as the number of conditions that are encountered going back to the initial conditions, comprising $c_i$ and $c_j$. In the unravel net $N_3$, the conditions $c_3$ and $c_4$ have the same label $p$ and have the same distance from the initial conditions, and similarly $c_5$ and $c_6$. By identifying these conditions also the transitions $e_5$ and $e_6$ have to be identified, resulting in the net $N_4$. 1 is the equivalence class containing $c_3$ and $c_4$, 2 the one with $c_5$ and $c_6$ and finally $\hat{e}$ is the transition obtained fusing $e_3$ and $e_4$, has these two transitions have the same preset, the same postset and are equally labelled, thus they *share* the same future.

The criteria used to obtain merged and trellises processes may be generalized equipping the unravel net with a mapping that associate to each condition a unique number, which we can call the *measure*. Thus a merging relation is obtained making equivalent all the conditions having the same labels, the same measure and being pairwise conflicting.

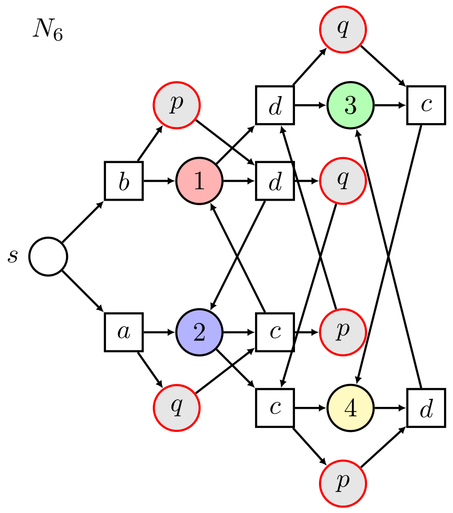


For the compaction process to be of real interest one would like to obtain a net which is possibly an unravel one, or that has strong relations with the unravel net we started with. In fact we devise two characteristic the compaction process may have. The first one is that to each execution in the compact net, at least an execution in the original one should correspond. The merging relation used to obtain $N_2$ out of $N_1$ does not fulfil this property, whereas it does the merging relation used to obtain $N_4$ out of $N_3$. When a merging relation fulfil this property we say that it is a *reflecting merging relation*. Clearly a reflecting merging relation always exists, as the identity relation is reflecting.

The property of being reflecting, adopting as the measure the token occurrence, can be enforced by enriching the starting unravel net. In the net $N_1$ conditions are used both to represent dependencies and conflicts, and by fusing some of them the dependencies may be lost. Thus the idea is to add some conditions that captures the dependencies. These conditions are easily obtainable by considering the whole token count for each transition of net. The net $N_1$ can be enriched as shown in the net $N_5$, and the added conditions are labeled with the condition representing the dependency.



Among the added conditions, in this case, there is no equivalence, as all of them have a different measure, the measure in this case being the one represented by the whole token count for the transitions (details on how to determine this measure can be found in [10], where the theory is applied to multi-clock nets). The result of the compaction process is the net $N_6$. Now the execution $e_1$ followed by $e_3$ and $e_4$ is no longer possible and $e_3$ is followed by $e_5$ only.



Beside looking for reflecting merging relation, one could be interested in preserving some characteristic of the net. For instance, one may be interested in preserving the fact that the resulting net is still an unravel one (and the measure induced by the time in the compaction done with trellis processes has this characteristic) or being acyclic when restricted to a certain subset of conditions (again, when considering the conditions belonging to an automata this is the case in trellis processes). When properties fulfilled by the net we start with are preserved by the compaction process we say that the merging relation is *preserving*. The merging relation giving the net $N_6$ preserves the property that, when only the added conditions are considered, the whole net is acyclic, and verification can be performed easily.

## References

1. Desel, J., Reisig, W.: The concepts of Petri nets. Software and System Modeling **14**(2) (2015) 669–683
2. Reisig, W.: Understanding Petri Nets - Modeling Techniques, Analysis Methods, Case Studies. Springer (2013)
3. Winskel, G.: Event Structures. In: Petri Nets: Central Models and Their Properties. LNCS 255 (1987) 325–392
4. Engelfriet, J.: Branching processes of Petri nets. Acta Informatica **28**(6) (1991) 575–591
5. McMillan, K.L.: Using unfoldings to avoid the state explosion problem in the verification of asynchronous circuits. In: CAV '92. LNCS 663 (1993) 164–177

6. Esparza, J., Römer, S., Vogler, W.: An Improvement of McMillan's Unfolding Algorithm. Formal Methods in System Design **20**(3) (2002) 285–310
7. Khomenko, V., Kondratyev, A., Koutny, M., Vogler, W.: Merged Processes: a new condensed representation of Petri net behaviour. Acta Informatica **43**(5) (2006) 307–330
8. van Glabbeek, R.J.: The individual and collective token interpretation of Petri nets. In: CONCUR 2005. LNCS 3653 (2005) 323–337
9. Fabre, E.: Trellis processes : A compact representation for runs of concurrent systems. Discrete Event Dynamic Systems **17**(3) (2007) 267–306
10. Casu, G., Pinna, G.M.: Flow unfolding of multi-clock nets. In: PETRI NETS 2014. LNCS 8489 (2014) 170–189