Robustness: A natural Definition based on Nets-within-Nets

Michael Köhler-Bussmeier[1], Lorenzo Capra[2]

[1]Hamburg University of Applied Sciences, Berliner Tor 7, D-20099 Hamburg
[2]Dipartimento di Informatica, Università degli Studi di Milano - Italy

Abstract
In modern distributed systems robustness is a major requirement. In previous work on availability, the analysis required an additional model part that specifies the assumptions about where in the model there are areas of unreliability. So, there are two models: the system itself and the error model. The error model usually requires specific domain knowledge. Therefore, the approach is not applicable out-of-the-box.

Instead, we like to derive an error model directly from the system model. We will show that for Elementary Object Systems we have a natural candidate to describe such a localised area of failure: the net-tokens. They are clearly localised and can be understood as computational entities (like containers in Kubernetes).

Keywords
Robustness, nets within nets, nets as tokens, multi-agent systems

1. Defining Robustness for Models of Distributed Architectures

In distributed systems (like e.g. applications based on micro services or multi-agent system) robustness is a major research topic [1]. In some of our previous work on availability [2, 3, 4], we observed that the analysis required an additional model part where we have to specify our assumptions about 'sources' of unreliability. So, there are two models: the system itself and the error model. The error model usually requires specific domain knowledge. Therefore, the approach is not applicable out-of-the-box.

As a rule of thumb, in micro service or container-based architectures we assume that the unit of failure is a service/container. More generally, we assume that each 'computational unit' is a source of failure. Here, we follow the *let it crash* approach – as popularised by the programming language Erlang – and assume that our computational units are either operational or completely crashed, i.e., we have a crash failure model opposed to a Byzantine error model [5].

Petri nets [6] are a standard model for distributed systems. For traditional Petri net models, including coloured variants, it is hard to identify such computational units since any subnet of the net could represent one of them. Therefore, we need additional domain knowledge to

PNSE'23, International Workshop on Petri Nets and Software Engineering, 2023
michael.koehler-bussmeier@haw-hamburg.de (M. Köhler-Bussmeier); capra@di.unimi.it (L. Capra)
https://www.haw-hamburg.de/michael-koehler-bussmeier (M. Köhler-Bussmeier)
0000-0002-3074-4145 (M. Köhler-Bussmeier); 0000-0002-1029-1169 (L. Capra)

identify these sub-nets as the whole structure does not provide clear hints to identify them.[1]

Our main goal is to derive the computational units directly from the system model – without the use of domain knowledge. For Nets-within-Nets as proposed by Valk [7] we have a *natural* candidate to describe such a localised service: the net-tokens. They are clearly localized and can be understood as computational entities. This idea is also applicable to other models having an explicit notion of computational context, like the Ambient Calculus [8] or the π-calculus [9]. But, for most formalisms the borderlines of computational contexts are not that obvious and usually have to be identified with the modeller's help.

After we have identified net-tokens as computational units it is straightforward to define robustness: Robustness means that the system still provides its service even when one or more units are malfunctioning. Here, we model a malfunction by a complete deactivation of a net-token, which provides no observable inner activity any more. To formally express the crash-failure of computational units we define a state space extension where we have so called *crash* or *break-down* events.

Our main application area are multi-agent systems (MAS). The first author uses nets-within-nets to specify adaptivity of agents in the context MAS-organisations [10] where one considers large-scale and and agent-independent aspects of MAS, like roles, norms, positions, interaction protocols etc. His MAS-organisation framework Sonar [11] is based on Petri nets, which facilitates the definition of the semantics as well as the analysis. For these Sonar-MAS the agents are modelled as net-tokens within the overall MAS organisation net. The natural unit to break-down is an agent (i.e. a net-token), so our concept of robustness fits well here.

We like to mention some limitations of our idea: Obviously, our *let it crash* assumption is not fulfilled in general; but it matches with our main research area, i.e., adaptive agents, where agent is any active computational entity, like services, containers, mobile devices, etc. Additionally, we admit that for a quantitative analysis, like in [12], a fault-model is still needed.

The paper has the following structure. Section 2 introduces base nets-within-nets (Eos). Section 3 defines the concept of robustness for Eos: The definition extends the reachability graph with additional events that model a break-down. In Section 4 we show that this semantic extension can be characterized syntactically by a slight modification of the Eos structure. The work closes with a conclusion and outlook.

2. Nets-within-Nets, EOS

Object nets [13, 14, 15], follow the *nets-within-nets* paradigm as proposed by Valk [7]. Other approaches adapting the nets-within-nets approach are nested nets [16], mobile predicate/transition nets [17], Reference nets [18], PN^2 [19], hypernets [20], and adaptive workflow nets [21]. There are relationships to Rewritable Petri nets [22] and Reconfigurable Petri Nets [23]. Object Nets can be seen as the Petri net perspective on contextual change, in contrast to the Ambient Calculus [8] or the π-calculus [9], which form the process algebra perspective.

[1]Sometimes we consider Petri nets that have a compositional structure, like workflows with AND-splits or OR-choices; or we can identify an refinement structure. In these cases we can identify at least sub-structures, which are candidates for these computational units. However, there is no one-to-one correspondence between the sub-structures and the concept of a computational unit.

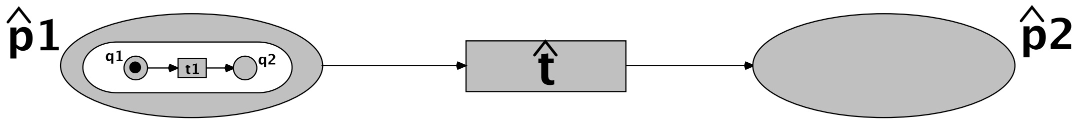


Figure 1: An Elementary Object Net System (Eos)

With nets-within-nets we study Petri nets where tokens are Petri nets in turn, i.e., we have nested markings. For the class of *elementary object net systems* (Eos), we restrict the nesting to two levels [13, 15]. Events are also nested: We have three different kinds of events – as illustrated by the example given in Figure 1:

1. System-autonomous: The system net transition $\widehat{t}$ fires autonomously, which moves the net-token from $\widehat{p}_1$ to $\widehat{p}_2$ without changing its marking.
2. Object autonomous: The object net fires transition t_1 by "moving" the black token from q_1 to q_2. The object net remains at its location $\widehat{p}_1$.
3. Synchronisation: Whenever we add matching synchronisation inscriptions at the system net transition $\widehat{t}$ and the object net transition t_1, then both must fire synchronously: The object net is moved to $\widehat{p}_2$ whereas the black token moves from q_1 to q_2 inside. Whenever synchronisation is specified, autonomous actions are forbidden.

An elementary object system (Eos) is composed of a system net, which is a p/t net $\widehat{N} = (\widehat{P}, \widehat{T}, \mathbf{pre}, \mathbf{post})$, and a set of object nets $\mathcal{N} = \{N_1, \dots, N_n\}$, which are p/t nets given as $N = (P_N, T_N, \mathbf{pre}_N, \mathbf{post}_N)$, where $N \in \mathcal{N}$. We assume $\widehat{N} \notin \mathcal{N}$ and the existence of the empty object net $\bullet \in \mathcal{N}$, which models black tokens. The system net places are typed by the mapping $d : \widehat{P} \to \mathcal{N}$ meaning that a place $\widehat{p} \in \widehat{P}$ of the system net may only contain net-tokens of the object net type $d(\widehat{p}) = N$. No place of the system net is mapped to the system net itself since $\widehat{N} \notin \mathcal{N}$.

Since the tokens of an Eos are instances of object nets, a *marking* of an Eos is a *nested* multiset. A marking of an Eos OS is denoted $\mu = \sum_{k=1}^{n}(\widehat{p}_k, M_k)$, where $\widehat{p}_k$ is a place of the system net and M_k is the marking of a net-token with type $d(\widehat{p}_k)$. To emphasise the nesting, markings are also denoted as $\mu = \sum_{k=1}^{n} \widehat{p}_k[M_k]$. For example, the marking show in Fig. 1 is $\mu = \widehat{p}_1[q_1]$.

The set of all markings which are syntactically consistent with the typing d is denoted $\mathcal{M}$, where $d^{-1}(N) \subseteq \widehat{P}$ is the set of system net places of the type N:

$$\mathcal{M} := MS\left(\bigcup\nolimits_{N \in \mathcal{N}} \left(d^{-1}(N) \times MS(P_N)\right)\right) \tag{1}$$

Analogously to markings, which are nested multisets μ, the events of an Eos are also nested. An Eos allows three different kinds of events: system-autonomous, synchronous, and object-autonomous events. The set of events is denoted Θ (cf. the appendix for details).

Definition 1 (Elementary Object System, EOS). *An elementary object system (Eos) is a tuple $OS = (\widehat{N}, \mathcal{N}, d, \Theta, \mu_0)$, where:*

1. $\widehat{N}$ *is a p/t net, called the* system net.

2. $\mathcal{N}$ *is a finite set of p/t nets, called* object nets*, with disjoint set of nodes.*
3. $d : \widehat{P} \to \mathcal{N}$ *is the* typing *of the system net places.*
4. Θ *is the set of* events*.*
5. $\mu_0 \in \mathcal{M}$ *is the initial marking.*

More details – e.g. the firing rule $\mu \xrightarrow[OS]{\theta} \mu'$ – are given in the appendix. For a in-depth presentation of Eos cf. [15]. Most problems for *safe Eos* (in which place markings $[M_k]$ are *sets*) are PSpace-complete [24, 25, 15]. More precisely: All problems that are expressible in LTL or CTL, which includes reachability and liveness, are PSpace-complete. This means that with respect to these problems *safe Eos* are no more complex than P/T nets.

For Hornets we extend object nets with algebraic concepts that allow to modify the structure of the net-tokens as a result of a firing transition. The complexity for *safe, elementary Hornets* is beyond PSpace: We have shown that "the reachability problem requires exponential space" for safe, elementary Hornets [26]. In [27] we give an algorithm that needs at most exponential space, which shows that lower and upper bound coincide.

3. Defining Robustness for EOS

We have a very simple idea: The most natural unit of computation in a Eos is easy to identify – it is a net-token. We assume that each computational unit may fail during the execution. This idea also applies to other models having an explicit notion of computational context, like the Ambient Calculus [8] or the π-calculus [9]. For other formalism the borderlines of computational contexts are not that obvious and usually have to be identified with the modeller's help.

Assume a given Eos $OS = (\widehat{N}, \mathcal{N}, d, \Theta, \mu_0)$. Its reachability graph $RG(OS)$ defines the behaviour in the absence of failures, i.e. the reachability graph is the state space for $k = 0$ break-downs.

In the following we assume break-downs. We tag markings μ with the number k of break-downs and denote this as the pair $\langle \mu, k \rangle$.

For technical convenience, we assume that transitions in object nets have a non-empty preset, so, the empty marking disables object-autonomous events and synchronisations. As a consequence, a break-down is like an enforced deadlock in the net-token.[2]

We have additional break-down edges that connect the level k with $k+1$; a break down edge models the fact that we have one more break-down in the system. A break-down picks some net-token $\widehat{p}[M]$ and disables it by setting the net-token's marking to $\mathbf{0}$:

$$BD_{k+1} := \{(\langle \mu, k \rangle, \langle \mu', k+1 \rangle) \mid \exists \mu'' : \mu = \widehat{p}[M] + \mu'' \wedge M \neq \mathbf{0} \wedge \mu' = \widehat{p}[\mathbf{0}] + \mu''\}$$

For technical convenience we define $BD_0 := \emptyset$.

We define V_k as the nodes of level k. For $k = 0$ we have the reachable markings. For $k > 0$ we have all markings reachable from a marking after the last break-down:

$$V_0 \quad = \quad \{\langle \mu, 0 \rangle \mid \mu_0 \xrightarrow{*} \mu\} \tag{2}$$

[2]If one wishes to distinguish between a deadlock and a break-down, we have to ensure that net-tokens do not deadlock unless they are unmarked. We can guarantee this by adding a side-transition to each place in the object net.

$$V_{k+1} \quad = \quad \{\langle \mu'', k+1\rangle \mid (\langle \mu, k\rangle, \langle \mu', k+1\rangle) \in BD_{k+1} \wedge \mu' \xrightarrow{*} \mu''\} \tag{3}$$

We define the edges of level k as:

$$E_k := \{(\langle \mu, k\rangle, \langle \mu', k\rangle) \mid \mu \xrightarrow{\theta} \mu'\} \tag{4}$$

Each sub-graph (V_i, E_i) contains all the node of the same level i and these sub-graphs are disjoint, since there are no edges from a higher level to a lower level, i.e. the level is increased monotonously.

Definition 2. *The graph* $RG_k(OS) = \left(\bigcup_{i=0}^{k} V_i, \bigcup_{i=0}^{k}(E_i \cup BD_i)\right)$ *is called the* failure extension *of* OS *up to level* k *or the* k-failure extension.

The graph $RG^*(OS) := (V^*, E^*)$ *where* $V^* := \bigcup_{i\geq 0} V_i$ *and* $E^* = \bigcup_{i\geq 0}(E_i \cup BD_i)$ *is called the* $*$-failure extension.

Note, that $RG(OS)$ is isomorphic to $RG_0(OS)$ as we only have to remove the tag 0 from the nodes.

Due to the explicit notion of levels in the marking $\langle \mu, k\rangle$ the structure of $RG_k(OS)$ is a chain of sub-graphs (V_i, E_i); these sub-graphs are linked by the breakdown edges BD_i. Schematically:

$$(V_0, E_0) \xrightarrow{BD_1} (V_1, E_1) \xrightarrow{BD_2} \cdots \xrightarrow{BD_k} (V_k, E_k)$$

Proposition 1. *Whenever a firing sequence that is enabled within the sub-graph* (V_{i+1}, E_{i+1}) *it is enabled within* (V_i, E_i)*, too.*

Proof The central observation is that we must have a break-down connection between the two sub-graph (V_i, E_i) and (V_{i+1}, E_{i+1}). In this case, we modify the marking $\mu = \widehat{p}[M_i] + \mu''$ into $\mu' = \widehat{p}[\mathbf{0}] + \mu''$, i.e. by a break-down of the net-token $\widehat{p}[M]$.

We use the following partial-order $\alpha \preceq \beta$ on nested multi-sets:

$$\begin{aligned}\textstyle\sum_{i=1}^{m} \widehat{a}_i[A_i] \preceq \sum_{j=1}^{n} \widehat{b}_j[B_j] \iff\ & \exists \text{ injection } f : \{1, ..., m\} \to \{1, ..., n\} : \\ & \forall 1 \leq i \leq m : \widehat{a}_i = \widehat{b}_{f(i)} \wedge A_i \leq B_{f(i)}\end{aligned}$$

As we have shown in [15, Lemma 5.1] that he firing rule is monotonous w.r.t. the order $\preceq$. Obviously, we have $\mu \in V_k$, $\mu' \in V_{k+1}$ and $\mu' \preceq \mu$ for the markings given above. Whenever μ' enables a sequence in (V_{i+1}, E_{i+1}) then μ enables this sequence in (V_{i+1}, E_{i+1}), too. Since we have obtained (V_{i+1}, E_{i+1}) from (V_i, E_i) by setting some net-token's marking to $\mathbf{0}$, the sequence is enabled in (V_i, E_i), too. qed.

We can also describe the behaviour in the limit, i.e. for $k \to \infty$: For each increase of the level we have one more break-down. So, if we consider the sub-graph (E_k, V_k) for very large k, we usually have that all net-tokens are down. In this case the net-tokens do not enable object-autonomous events any more and also no synchronous events are possible; only system-autonomous events are possible. Thus, in the limit the Eos behaves like the system net $\widehat{N}$ when considered as a p/t net. Technically we obtain this behaviour by the projection Π^1, which maps a marking μ to the multiset of system net places.

3.1. The semantic definition of robustness

The semantic definition of robustness uses our definition of the reachability graph extended with break-downs. We introduce a hierarchy for satisfiability by restricting the numbers of break-down events: The usual satisfiability of a property ϕ considers the normal reachability graph $RG_0(OS)$; whenever we extend the reachability graph towards $RG_1(OS)$ we check whether the property can withstand a break-down of at most one net-token. We can generalise this idea to an arbitrary number k of break-down events.

Definition 3 (Robustness). *A property ϕ is satisfied k-*robustly *by the Eos OS iff ϕ holds in $RG_k(OS)$. This is denoted by $OS, \mu_0 \models_k \phi$, or short: $\mu_0 \models_k \phi$ wheneverOS is clear from the context.*

*A property ϕ is satisfied $*$-*robustly *by the Eos OS iff ϕ holds in $RG^*(OS)$. This is denoted by $\mu_0 \models_* \phi$.*

Fact 1. *Note, that a property ϕ holds for an Eos OS iff ϕ is* 0*-robust, since $RG(OS) \simeq RG_0$.*

In general, a property ϕ that is satisfied k-robustly will not be satisfied $k+1$-robustly. Especially, a property ϕ that is satisfied by an Eos (0-robustly) is not satisfied 1-robustly, i.e., in general even one break-down destroys the property ϕ. Liveness of the Eos is a typical example. In other words: robustness does not come for free. Usually the Eos model has to take care to satisfy a property k-robustly, e.g., by having model parts that will restart crashed net-tokens in a well-defined state.

An example for a property ϕ that is satisfied even $*$-robustly is safeness [24], i.e., the property that there is at most one net-token on each place in the system net. This is due to the fact that a break-down modifies the marking μ in a way that does not change the projection $\Pi^1(\mu)$ to the system net and, consequently, does not enable any 'unseen' transition of the system net.

In our context of organisational multi-agent systems the notion of robustness allows us to specify two central classes of properties:

- Flexibility/Adaptivity: The MAS has the possibility to react to changes in the environment (here: break-downs of other agents) to re-establish a desired property ϕ is expressed as $OS, \mu_0 \models_* \text{AGEF}\phi$ – the usual liveness property in CTL [28].
- Resilience: The system always stays in a 'safe' region of the state space – even in the presence of break-downs of other agents – is expressed as $OS, \mu_0 \models_* \text{AG}\phi$ – a classical invariant property (aka as a safety property).

A structural property for the system net is derived from the net topology only. The classical example are net-invariants. A structural invariant holds for any initial marking and not only a given one (cf. [29] for the discussion between invariants and structural invariants).

Fact 2. *A structural property is independent from the net-tokens' markings and therefore $*$-robust.*

Another interesting question is whether we can tolerate break-downs using redundancy, i.e., multiple copies of the net-tokens. This question is closely related to the Petri net concept of monotonicity. A property is monotonous if it holds also for any greater initial marking. It is well known that reachability is monotonous, while deadlock-freedom or liveness are – in general – not (However, for the structural sub-class of free-choice nets liveness is monotonous).

For monotonous properties redundancy is useful, i.e. we can add additional net-tokens to survive knock-outs of net-tokens. Here we simply multiply the initial marking and consider $\mu_0 + k \cdot \mu_0$.

Proposition 2. *Let ϕ be a monotonous property. When ϕ holds in the initial marking μ_0, then ϕ is k-robust in the initial marking $\mu_0 + k \cdot \mu_0$:*

$$\mu_0 \models \phi \quad \Longrightarrow \quad (\mu_0 + k \cdot \mu_0) \models_k \phi$$

Proof (Sketch) We can execute at most k break-downs. So any marking μ reachable from $(\mu_0 + k \cdot \mu_0)$ is still greater than some marking reachable from μ_0 only. Since the property holds in μ_0 and is monotonous it holds in $RG_k(OS)$. qed.

4. The Syntactical Characterisation of Robustness

Our characterisation of robustness has the drawback that it is based on an extension of the usual Petri net semantics as we extend the reachability graph. In the following, we like to show that we can express this extension on a purely syntactical level, i.e., we modify the Eos OS into another Eos OS^* with the property that the 'normal' reachability graph of OS^* is isomorphic to the break-down extension $RG^*(OS)$. In this case, we can reduce the specialised analysis of OS to an 'normal' analysis of OS^* using existing standard tools.

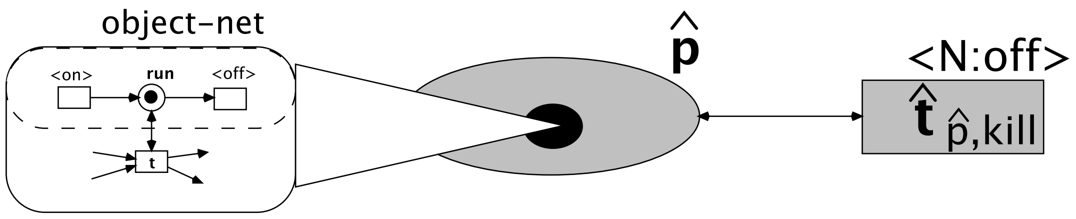


Figure 2: Construction of the Eos OS^*.

The main idea is quite simple (cf. Fig. 2): We apply the following construction to the Eos OS.

- We add a *run-place* to each object net and this run-place is attached as a side condition to each transition t.
- The run-place is initially marked with one token.
- We add a new transition t_{kill} to each object net that synchronises with the environment (via the channel $\langle$*off*$\rangle$) and removes the token of the run-place.
- In the system net we add for each place $\widehat{p}$ a new side transition $\widehat{t_{\widehat{p},kill}}$ which synchronises with t_{kill}.

The effect of the synchronous kill-event is that the net-token is dead afterwards (but usually contains useless tokens on the places other than the run-place). We like to establish that an empty run-place always indicates that the net-token has broken down. In this case we could drop the explicit tag k from the state because it is implicitly given by the number of net-tokens with an empty run-place. However, this is not true, since the Eos firing rule 'collects' the net-tokens markings from all incoming arcs and 'distributes' this sum over all net-tokens generated on the outgoing arcs. For a system net transition that has, e.g., one incoming arc and two outgoing arcs, one of the generated net-tokens must have an empty run-place. However, the net-token should not be regarded as a break-down.

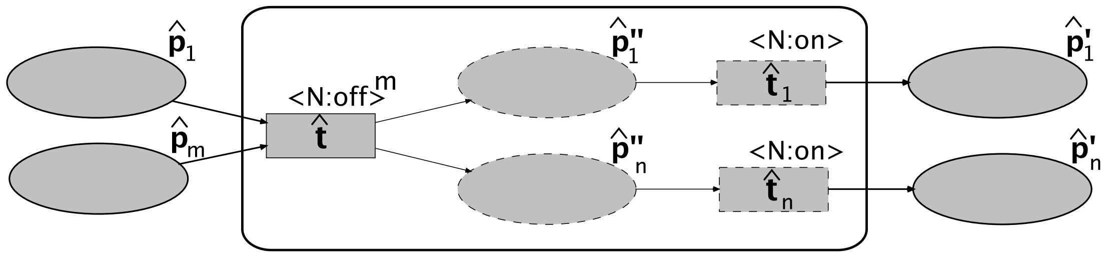


Figure 3: Construction for Run-Places in the Eos OS^*.

We can 'repair' this problem in the following way: We replace each system net transition $\widehat{t}$ by a simple subnet (cf. Fig. 3) that removes the tokens from the m incoming places $\widehat{p_i}$ (using the synchronisation channel $\langle$*off*$\rangle$ m times) and individually re-creates one token on each run-place in each net-token on the n outgoing places $\widehat{p'_i}$ (using the synchronisation channel $\langle$*on*$\rangle$ at the internal transitions $\widehat{t_i}$). We replace each system net transition with such a subnet.[3] The resulting Eos is denoted as OS^*.

Due to this construction, an empty run-place in a net-token (except for those net-tokens in the intermediate place $\widehat{p''_i}$ of the subnet) is equivalent to a break-down.

When we consider the n intermediate transitions $\widehat{t_i}$ as 'silent' events, then the constructed net OS^* is bisimilar to the original state space OS.

Proposition 3. *The extended Eos OS^* is the syntactical representation of the $*$-failure extension of OS, i.e. both state spaces are bisimilar: $RG^*(OS) \approx RG(OS^*)$*

With a small modification of the extended Eos OS^* we can also describe the k-failure extension of OS:

- We add a global pool-place to the system net that contains exactly k black tokens.
- Each kill-transition removes one token from the global pool-place.

We denote this modification of the Eos as OS_k.

Note that after k kill-events the pool place is empty and further kill-events are disabled. The behaviour of OS_k is similar to the extension OS^* except that we restrict the number of break-downs to k.

[3]We could skip this construction for system net transitions which have exactly one place in the preset and one in the postset.

Proposition 4. *The extended* Eos OS_k *is the syntactical representation of the* k*-failure extension of* OS*, i.e. both state spaces are bisimilar:* $RG_k(OS) \approx RG(OS_k)$

In the absence of break-downs, i.e., for $k = 0$, the normal semantics coincides with the extended semantics and the given Eos coincides with its extension:

Fact 3. *For* $k = 0$ *we obtain that all systems coincide:* $RG(OS) \simeq RG_0(OS) \approx RG(OS_0)$.

With these constructions we have reduced checking properties w.r.t. robustness to a usual state space based query.

5. Conclusion

In this paper, we have shown that Nets-within-Nets have an intuitive representation of the concept of a 'computational unit': the net-tokens. Based on this, we could define a break-down of net-tokens and define the satisfiability of properties in the context of break-downs.

Our concept of robustness relies on an extension of the state space. We provided a reduction of this semantical definition (which extends the Petri net semantics) to a syntactical one (where we use the usual semantics): We transformed the Eos OS into the Eos OS^* such that the 'normal' reachability graph of OS^* is bisimilar to the break-down extension $RG^*(OS)$. Therefore, we could still use existing tools to investigate robustness.

We plan to extend the simple – though popular – *let it crash*-based model, e.g., by allowing the computational units (net-tokens) to gracefully degrade.

In future work, we will apply this notion of robustness to our multi-agent system Sonar [11], especially with a focus on the analysis of adaptivity as we have started with our previous work on the analysis of adaption processes [30] and adapting agent architectures [31]. In [30] we used the set of different types of net-tokens that occur in the current marking to define the system's gene-pool. Our current research hypothesis is that a more diverse gene-pool in adapting agent architectures [31] leads to more robust systems in general.

References

[1] M. Baldoni, C. Baroglio, R. Micalizio, Fragility and robustness in multiagent systems, in: C. Baroglio, J. Hubner, M. Winikoff (Eds.), Engineering Multi-Agent Systems. EMAS 2020, volume 12589 of *Lecture Notes in Computer Science*, Springer-Verlag, 2020.

[2] M. Müller, M. Köhler-Bußmeier, Availability analysis of the ONOS architecture, in: E. Kindler, M. Köhler-Bußmeier, H. Rölke (Eds.), International Workshop on Petri Nets and Software Engineering 2021, volume 2907 of *CEUR Workshop Proceedings*, Aachen, 2021, pp. 41–64. URL: http://ceur-ws.org/Vol-2907/.

[3] L. Capra, M. Köhler-Bußmeier, A Maude formalization of object nets, in: S. Bonfanti, T. Kobayashi, D. Perez-Palacin (Eds.), 6th International Workshop on Formal Approaches for Advanced Computing Systems, FAACS'22, 2022.

[4] L. Capra., M. Köhler-Bußmeier., Modelling adaptive systems with nets-within-nets in maude, in: Proceedings of the 18th International Conference on Evaluation of Novel Approaches to Software Engineering - ENASE,, INSTICC, SciTePress, 2023, pp. 487–496. doi:10.5220/0011860000003464.

[5] L. Lamport, R. Shostak, M. Pease, The byzantine generals problem, ACM Transactions on Programming Languages and Systems (1982) 382–401.

[6] W. Reisig, Petri Nets: An Introduction, Springer-Verlag, Heidelberg, 1985.

[7] R. Valk, Object Petri nets: Using the nets-within-nets paradigm, in: J. Desel, W. Reisig, G. Rozenberg (Eds.), Advanced Course on Petri Nets 2003, volume 3098 of *Lecture Notes in Computer Science*, Springer-Verlag, 2003, pp. 819–848.

[8] L. Cardelli, A. D. Gordon, G. Ghelli, Mobility types for mobile ambients, in: Proceedings of the Conference on Automata, Languages, and Programming (ICALP'99), volume 1644 of *Lecture Notes in Computer Science*, Springer-Verlag, 1999, pp. 230–239.

[9] R. Milner, J. Parrow, D. Walker, A calculus of mobile processes, parts 1-2, Information and computation 100 (1992) 1–77.

[10] V. Dignum, J. Padget, Multiagent organizations, in: G. Weiss (Ed.), Multiagent Systems, 2nd ed., Intelligent Robotics; Autonomous Agents Series, MIT Press, 2013, pp. 51–98.

[11] M. Köhler-Bußmeier, M. Wester-Ebbinghaus, D. Moldt, A formal model for organisational structures behind process-aware information systems, Transactions on Petri Nets and Other Models of Concurrency. Special Issue on Concurrency in Process-Aware Information Systems 5460 (2009) 98–114.

[12] M. A. Marsan, G. Balbo, G. Conte, S. Donatelli, G. Franceschinis, Modelling with Generalized Stochastic Petri Nets, Wiley Series in Parallel Computing, John Wiley and Sons, 1995.

[13] M. Köhler, H. Rölke, Properties of Object Petri Nets, in: J. Cortadella, W. Reisig (Eds.), International Conference on Application and Theory of Petri Nets 2004, volume 3099 of *Lecture Notes in Computer Science*, Springer-Verlag, 2004, pp. 278–297.

[14] M. Köhler-Bußmeier, F. Heitmann, On the expressiveness of communication channels for object nets, Fundamenta Informaticae 93 (2009) 205–219.

[15] M. Köhler-Bußmeier, A survey on decidability results for elementary object systems, Fundamenta Informaticae 130 (2014) 99–123.

[16] I. A. Lomazova, Nested Petri nets – a formalism for specification of multi-agent distributed

systems, Fundamenta Informaticae 43 (2000) 195–214.
[17] D. Xu, Y. Deng, Modeling mobile agent systems with high level Petri nets, in: IEEE International Conference on Systems, Man, and Cybernetics'2000, 2000.
[18] O. Kummer, Referenznetze, Logos Verlag, 2002.
[19] K. Hiraishi, PN^2: An elementary model for design and analysis of multi-agent systems, in: F. Arbab, C. L. Talcott (Eds.), Coordination Models and Languages, COORDINATION 2002, volume 2315 of *Lecture Notes in Computer Science*, Springer-Verlag, 2002, pp. 220–235.
[20] M. A. Bednarczyk, L. Bernardinello, W. Pawlowski, L. Pomello, Modelling mobility with Petri hypernets, in: J. L. Fiadeiro, P. D. Mosses, F. Orejas (Eds.), Recent Trends in Algebraic Development Techniques (WADT 2004), volume 3423 of *Lecture Notes in Computer Science*, Springer-Verlag, 2004, pp. 28–44.
[21] I. A. Lomazova, K. M. van Hee, O. Oanea, A. Serebrenik, N. Sidorova, M. Voorhoeve, Nested nets for adaptive systems, in: Application and Theory of Petri Nets and Other Models of Concurrency, Lecture Notes in Computer Science, Springer-Verlag, 2006, pp. 241–260.
[22] L. Capra, A maude implementation of rewritable petri nets: a feasible model for dynamically reconfigurable systems, in: M. Gleirscher, J. v. d. Pol, J. Woodcock (Eds.), First Workshop on Applicable Formal Methods 2021, 2021.
[23] J. Padberg, L. Kahloul, Overview of reconfigurable petri nets, in: R. Heckel, G. Taentzer (Eds.), Graph Transformation, Specifications, and Nets, volume 10800, Springer-Verlag, 2018, pp. 201–222.
[24] M. Köhler-Bußmeier, F. Heitmann, Safeness for object nets, Fundamenta Informaticae 101 (2010) 29–43.
[25] M. Köhler-Bußmeier, F. Heitmann, Liveness of safe object nets, Fundamenta Informaticae 112 (2011) 73–87.
[26] M. Köhler-Bußmeier, On the complexity of the reachability problem for safe, elementary Hornets, Fundamenta Informaticae 129 (2014) 101–116. Dedicated to the Memory of Professor Manfred Kudlek.
[27] M. Köhler-Bußmeier, F. Heitmann, An upper bound for the reachability problem of safe, elementary hornets, Fundamenta Informaticae 143 (2016) 89–100.
[28] A. Emerson, Temporal and modal logic, volume B, MIT press, 1990.
[29] E. Kindler, Invariants, composition, and substitution, Acta Informatica 32 (1995) 299–312.
[30] M. Köhler-Bußmeier, H. Rölke, Analysing adaption processes of Hornets, in: M. Köhler-Bußmeier, D. Moldt, H. Rölke (Eds.), Petri Nets and Software Engineering 2022, PNSE'22, volume 3170, CEUR, 2022, pp. 80–98.
[31] M. Köhler-Bußmeier, J. Sudeikat, Balance vs. contingency: Adaption measures for organizational multi-agent systems, in: K. Jander, L. Braubach, C. Badica (Eds.), 15th International Symposium on Intelligent Distributed Computing (IDC'22), volume 1089 of *Studies in Computational Intelligence*, Springer-Verlag, 2023.

A. Petri Nets

The definition of Petri nets relies on the notion of multisets. A multiset $\mathbf{m}$ on the set D is a mapping $\mathbf{m} : D \to \mathbb{N}$. Multisets are generalisations of sets in the sense that every subset of D corresponds to a multiset $\mathbf{m}$ with $\mathbf{m}(d) \leq 1$ for all $d \in D$. The empty multiset $\mathbf{0}$ is defined as $\mathbf{0}(d) = 0$ for all $d \in D$. Multiset addition for $\mathbf{m}_1, \mathbf{m}_2 : D \to \mathbb{N}$ is defined component-wise: $(\mathbf{m}_1 + \mathbf{m}_2)(d) := \mathbf{m}_1(d) + \mathbf{m}_2(d)$. Multiset-difference $\mathbf{m}_1 - \mathbf{m}_2$ is defined by $(\mathbf{m}_1 - \mathbf{m}_2)(d) := \max(\mathbf{m}_1(d) - \mathbf{m}_2(d), 0)$. We use common notations for the cardinality of a multiset $|\mathbf{m}| := \sum_{d \in D} \mathbf{m}(d)$ and multiset ordering $\mathbf{m}_1 \leq \mathbf{m}_2$, where the partial order $\leq$ is defined by $\mathbf{m}_1 \leq \mathbf{m}_2 \iff \forall d \in D : \mathbf{m}_1(d) \leq \mathbf{m}_2(d)$.

A multiset $\mathbf{m}$ is finite if $|\mathbf{m}| < \infty$. The set of all finite multisets over the set D is denoted $MS(D)$. The set $MS(D)$ naturally forms a monoid with multiset addition $+$ and the empty multiset $\mathbf{0}$. Multisets can be identified with the commutative monoid structure $(MS(D), +, 0)$. Multisets are the free commutative monoid over D since every multiset has the unique representation in the form $\mathbf{m} = \sum_{d \in D} \mathbf{m}(d) \cdot d$, where $\mathbf{m}(d)$ denotes the multiplicity of d. Multisets can also be represented as a formal sum in the form $\mathbf{m} = \sum_{i=1}^{n} x_i$, where $x_i \in D$.

Any mapping $f : D \to D'$ is extended to a multiset homomorphism $f^\sharp : MS(D) \to MS(D')$: $f^\sharp\left(\sum_{i=1}^{n} x_i\right) = \sum_{i=1}^{n} f(x_i)$. This includes the special case $f^\sharp(\mathbf{0}) = \mathbf{0}$. We simply write f to denote the mapping $f^\sharp$. The notation is in accordance with the set-theoretic notation $f(A) = \{f(a) \mid a \in A\}$.

Definition 4. *A* p/t net *N is a tuple $N = (P, T, \mathbf{pre}, \mathbf{post})$, such that P is a set of places, T is a set of transitions, with $P \cap T = \emptyset$, and $\mathbf{pre}, \mathbf{post} : T \to MS(P)$ are the pre- and post-condition functions. A marking of N is a multiset of places: $\mathbf{m} \in MS(P)$. A p/t net with initial marking $\mathbf{m}_0$ is denoted $N = (P, T, \mathbf{pre}, \mathbf{post}, \mathbf{m}_0)$.*

We use the usual notations for nets like ${}^\bullet x$ for the set of predecessors and $x^\bullet$ for the set of successors for a node $x \in (P \cup T)$. For $t \in T$ we have ${}^\bullet t = \{p \in P \mid \mathbf{pre}(t)(p) > 0\}$ and $t^\bullet = \{p \in P \mid \mathbf{post}(t)(p) > 0\}$. For $p \in P$ we have ${}^\bullet p = \{t \in T \mid \mathbf{post}(t)(p) > 0\}$ and $p^\bullet = \{t \in T \mid \mathbf{pre}(t)(p) > 0\}$.

A transition $t \in T$ of a p/t net N is enabled in marking $\mathbf{m}$ iff $\forall p \in P : \mathbf{m}(p) \geq \mathbf{pre}(t)(p)$ holds. The successor marking when firing t is $\mathbf{m}'(p) = \mathbf{m}(p) - \mathbf{pre}(t)(p) + \mathbf{post}(t)(p)$ for all $p \in P$. Using multiset notation enabling is expressed by $\mathbf{m} \geq \mathbf{pre}(t)$ and the successor marking is $\mathbf{m}' = \mathbf{m} - \mathbf{pre}(t) + \mathbf{post}(t)$. We denote the enabling of t in marking $\mathbf{m}$ by $\mathbf{m} \xrightarrow[N]{t}$. Firing of t is denoted by $\mathbf{m} \xrightarrow[N]{t} \mathbf{m}'$. The net N is omitted if it is clear from the context.

Firing is extended to sequences $w \in T^*$ in the obvious way: (i) $\mathbf{m} \xrightarrow{\epsilon} \mathbf{m}$; (ii) If $\mathbf{m} \xrightarrow{w} \mathbf{m}'$ and $\mathbf{m}' \xrightarrow{t} \mathbf{m}''$ hold, then we have $\mathbf{m} \xrightarrow{wt} \mathbf{m}''$. We write $\mathbf{m} \xrightarrow{*} \mathbf{m}'$ whenever there is some $w \in T^*$ such that $\mathbf{m} \xrightarrow{w} \mathbf{m}'$ holds. The set of reachable markings is $RS(\mathbf{m}_0) := \{\mathbf{m} \mid \exists w \in T^* : \mathbf{m}_0 \xrightarrow{w} \mathbf{m}\}$.

B. Elementary Object Systems

An elementary object system (Eos) is composed of a system net, which is a p/t net $\widehat{N} = (\widehat{P}, \widehat{T}, \mathbf{pre}, \mathbf{post})$ and a set of object nets $\mathcal{N} = \{N_1, \dots, N_n\}$, which are p/t nets given as $N = (P_N, T_N, \mathbf{pre}_N, \mathbf{post}_N)$, where $N \in \mathcal{N}$. In extension we assume that all sets of nodes (places and transitions) are pairwise disjoint. Moreover we assume $\widehat{N} \notin \mathcal{N}$ and the existence of the object net $\bullet \in \mathcal{N}$, which has no places and no transitions and is used to model anonymous, so called black tokens.

Typing The system net places are typed by the mapping $d : \widehat{P} \to \mathcal{N}$ with the meaning, that a place $\widehat{p} \in \widehat{P}$ of the system net with $d(\widehat{p}) = N$ may contain only net-tokens of the object net type N.[4] No place of the system net is mapped to the system net itself since $\widehat{N} \notin \mathcal{N}$.

A typing is called *conservative* iff for each place $\widehat{p}$ in the preset of a system net transition $\widehat{t}$ such that $d(\widehat{p}) \neq \bullet$ there is place in the postset being of the same type: $(d({}^\bullet\widehat{t}) \cup \{\bullet\}) \subseteq (d(\widehat{t}^\bullet) \cup \{\bullet\})$. An Eos is *conservative* iff its typing d is.

An Eos is *p/t-like* iff it has only places for black tokens: $d(\widehat{P}) = \{\bullet\}$.

Nested Markings Since the tokens of an Eos are instances of object nets, a *marking* of an Eos is a *nested* multiset. A marking of an Eos OS is denoted $\mu = \sum_{k=1}^{|\mu|} (\widehat{p}_k, M_k)$, where $\widehat{p}_k$ is a place of the system net and M_k is the marking of a net-token with type $d(\widehat{p}_k)$. To emphasise the nesting, markings are also denoted as $\mu = \sum_{k=1}^{|\mu|} \widehat{p}_k[M_k]$. Markings of the form $\widehat{p}[\mathbf{0}]$ with $d(\widehat{p}) = \bullet$ are abbreviated as $\widehat{p}[]$.

The set of all markings which are syntactically consistent with the typing d is denoted $\mathcal{M}$, where $d^{-1}(N) \subseteq \widehat{P}$ is the set of system net places of the type N:

$$\mathcal{M} := MS\left(\bigcup_{N \in \mathcal{N}} \left(d^{-1}(N) \times MS(P_N)\right)\right) \tag{5}$$

We define the partial order $\sqsubseteq$ on nested multisets by setting $\mu_1 \sqsubseteq \mu_2$ iff $\exists \mu : \mu_2 = \mu_1 + \mu$.

A more liberal variant is the order $\preceq$ defined by:

$$\begin{array}{lll} \alpha \preceq \beta & \iff & \alpha = \sum_{i=1}^{m} \widehat{a}_i[A_i] \wedge \beta = \sum_{j=1}^{n} \widehat{b}_j[B_j] \wedge \\ & & \exists \text{ injection } f : \{1, ..., m\} \to \{1, ..., n\} : \\ & & \forall 1 \leq i \leq m : \widehat{a}_i = \widehat{b}_{f(i)} \wedge A_i \leq B_{f(i)} \end{array} \tag{6}$$

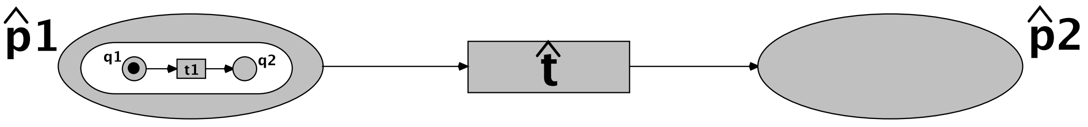
For $\alpha \preceq \beta$ the injection f generates a sub-marking of β which is denoted $f(\alpha) = \sum_{i=1}^{m} \widehat{a}_{f(i)}[A_{f(i)}]$.

Note that $\alpha \sqsubseteq \beta$ is a special case of $\alpha \preceq \beta$, where $A_i \leq B_{f(i)}$ is restricted to $A_i = B_{f(i)}$.

[4]In some sense, net-tokens are object nets with its own marking. However, net-tokens should not be considered as *instances* of an object net (as in object-oriented programming), since net-tokens do not have an identity. This is reflected by the fact that the firing rule joins and distributes the net-tokens' markings.

Instead, all net-tokes of an object net act as a *collective* entity, like a group. This group can be considered as an object with identy – an object with its state distributed over the net-tokens. For in in-depth discussion of this semantics cf. [7].

Events Analogously to markings, which are nested multisets μ, the events of an Eos are also nested. An Eos allows three different kinds of events – as illustrated by the following Eos:



1. System-autonomous: The system net transition t fires autonomously which moves the net-token from p_1 to p_2 without changing its marking.
2. Object-autonomous: The object net fires transition t_1, which "moves" the black token from q_1 to q_2. The object net itself remains at its location p_1.
3. Synchronisation: The system net transition t fires synchronously with t_1 in the object net. Whenever synchronisation is demanded, autonomous actions are forbidden.

The set of events is denoted Θ. Events are formalised as a pair $\widehat{\tau}[\vartheta]$, where $\widehat{\tau}$ is either the transition that fires in the system net or a special "idle" transition $id_{\widehat{p}}$ (cf. below); and ϑ is a function such that $\vartheta(N)$ is the multiset of transitions, which have to fire synchronously with $\widehat{\tau}$, (i.e. for each object net $N \in \mathcal{N}$ we have $\vartheta(N) \in MS(T_N)$).[5]

In general $\widehat{\tau}[\vartheta]$ describes a synchronisation, but autonomous events are special subcases: Obviously, a system-autonomous events is the special case, where $\vartheta = \mathbf{0}$ with $\mathbf{0}(N) = \mathbf{0}$ for all object nets N. To describe an object-autonomous event we assume the set of *idle transitions* $\{id_{\widehat{p}} \mid \widehat{p} \in \widehat{P}\}$, where $id_{\widehat{p}}$ formalises object-autonomous firing on the place $\widehat{p}$:

1. Each idle transition $id_{\widehat{p}}$ has $\widehat{p}$ as a side condition: $\mathbf{pre}(id_{\widehat{p}}) = \mathbf{post}(id_{\widehat{p}}) := \widehat{p}$.
2. Each idle transition $id_{\widehat{p}}$ synchronises only with transitions from $N = d(\widehat{p})$:

$$\forall \widehat{\tau}[\vartheta] \in \Theta : \ \widehat{\tau} = id_{\widehat{p}} \implies \forall N \in \mathcal{N} : \ (\vartheta(N) \neq \mathbf{0} \iff N = d(\widehat{p}))$$

Definition 5 (Elementary Object System, EOS). *An elementary object system (Eos) is a tuple $OS = (\widehat{N}, \mathcal{N}, d, \Theta, \mu_0)$, where:*

1. $\widehat{N}$ *is a p/t net, called the* system net.
2. $\mathcal{N}$ *is a finite set of disjoint p/t nets, called* object nets.
3. $d : \widehat{P} \to \mathcal{N}$ *is the* typing *of the system net places.*
4. Θ *is the set of* events.
5. $\mu_0 \in \mathcal{M}$ *is the initial marking.*

Example Figure 4 shows an Eos with the system net $\widehat{N}$ and the object nets $\mathcal{N} = \{N_1, N_2\}$. The system has four net-tokens: two on place $\widehat{p}_1$ and one on $\widehat{p}_2$ and $\widehat{p}_3$ each. The net-tokens on $\widehat{p}_1$ and $\widehat{p}_2$ share the same net structure, but have independent markings.

[5]In the graphical representation the events are generated by transition inscriptions. For each object net $N \in \mathcal{N}$ a system net transition $\widehat{t}$ is labelled with a multiset of channels $\widehat{l}(\widehat{t})(N) = ch_1 + \cdots + ch_n$, depicted as $\langle N{:}ch_1, N{:}ch_2, \ldots \rangle$. Similarily, an object net transition t may be labelled with a channel $l_N(t) = ch$ – depicted as $\langle {:}ch \rangle$ whenver there is such a label. We obtain an event $\widehat{t}[\vartheta]$ by setting $\vartheta(N) := t_1 + \cdots + t_n$ to be any transition multiset such that the labels match: $l_N(t_1) + \cdots + l_N(t_n) = \widehat{l}(\widehat{t})(N)$.

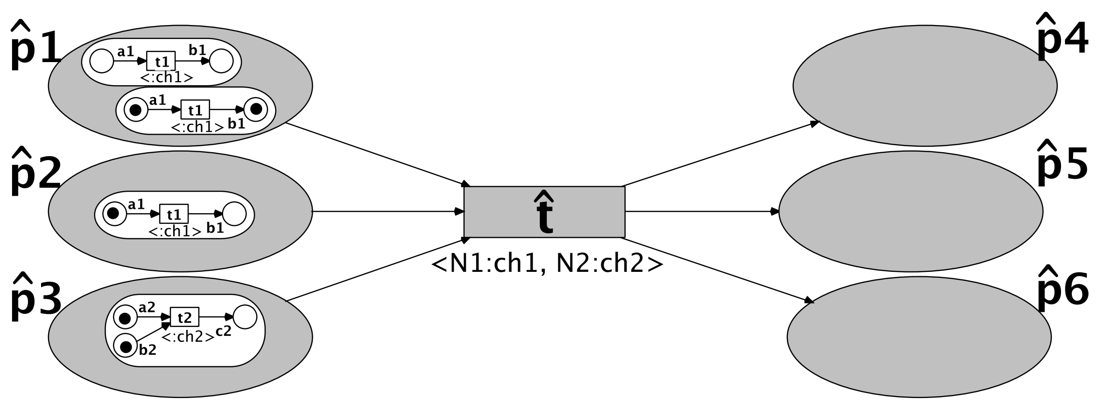


Figure 4: An Elementary Object Net System

The system net is $\widehat{N} = (\widehat{P}, \widehat{T}, \mathbf{pre}, \mathbf{post})$, where $\widehat{P} = \{\widehat{p}_1, \ldots, \widehat{p}_6\}$ and $\widehat{T} = \{\widehat{t}\}$.
One object net is $N_1 = (P_1, T_1, \mathbf{pre}_1, \mathbf{post}_1)$ with $P_1 = \{a_1, b_1\}$ and $T_1 = \{t_1\}$.
Another object net is $N_2 = (P_2, T_2, \mathbf{pre}_2, \mathbf{post}_2)$ with $P_2 = \{a_2, b_2, c_2\}$ and $T_2 = \{t_2\}$.
The typing is $d(\widehat{p}_1) = d(\widehat{p}_2) = d(\widehat{p}_4) = N_1$ and $d(\widehat{p}_3) = d(\widehat{p}_5) = d(\widehat{p}_6) = N_2$.
The labelling generates one event: $\Theta = \{\widehat{t}[N_1 \mapsto t_1, N_2 \mapsto t_2]\}$.
The initial marking has two net-tokens on $\widehat{p}_1$, one on $\widehat{p}_2$, and one on $\widehat{p}_3$:

$$\mu = \widehat{p}_1[a_1 + b_1] + \widehat{p}_1[\mathbf{0}] + \widehat{p}_2[a_1] + \widehat{p}_3[a_2 + b_2]$$

Note that for Figure 4 the structure is the same for the three net-tokens on $\widehat{p}_1$ and $\widehat{p}_2$ but the net-token markings are different.

B.1. Firing Rule

The projection Π^1 on the first component abstracts from the substructure of all net-tokens for a marking of an Eos:

$$\Pi^1\left(\sum\nolimits_{k=1}^{n} \widehat{p}_k[M_k]\right) := \sum\nolimits_{k=1}^{n} \widehat{p}_k \tag{7}$$

The projection Π^2_N on the second component is the sum of all net-token markings M_k of the type $N \in \mathcal{N}$, ignoring their local distribution within the system net:

$$\Pi^2_N\left(\sum\nolimits_{k=1}^{n} \widehat{p}_k[M_k]\right) := \sum\nolimits_{k=1}^{n} \mathbf{1}_N(\widehat{p}_k) \cdot M_k \tag{8}$$

where the indicator function $\mathbf{1}_N : \widehat{P} \to \{0, 1\}$ is $\mathbf{1}_N(\widehat{p}) = 1$ iff $d(\widehat{p}) = N$. Note that $\Pi^2_N(\mu)$ results in a marking of the object net N.

A system event $\widehat{\tau}[\vartheta]$ removes net-tokens together with their individual internal markings. Firing the event replaces a nested multiset $\lambda \in \mathcal{M}$ that is part of the current marking μ, i.e. $\lambda \sqsubseteq \mu$, by the nested multiset ρ. Therefore the successor marking is $\mu' := (\mu - \lambda) + \rho$. The enabling condition is expressed by the *enabling predicate* ϕ_{OS} (or just ϕ whenever OS is clear from the context):

$$\begin{aligned} \phi(\widehat{\tau}[\vartheta], \lambda, \rho) \iff & \Pi^1(\lambda) = \mathbf{pre}(\widehat{\tau}) \wedge \Pi^1(\rho) = \mathbf{post}(\widehat{\tau}) \wedge \\ & \forall N \in \mathcal{N} : \Pi^2_N(\lambda) \geq \mathbf{pre}_N(\vartheta(N)) \wedge \\ & \forall N \in \mathcal{N} : \Pi^2_N(\rho) = \Pi^2_N(\lambda) - \mathbf{pre}_N(\vartheta(N)) + \mathbf{post}_N(\vartheta(N)) \end{aligned} \tag{9}$$

With $\widehat{M} = \Pi^1(\lambda)$ and $\widehat{M'} = \Pi^1(\rho)$ as well as $M_N = \Pi^2_N(\lambda)$ and $M'_N = \Pi^2_N(\rho)$ for all $N \in \mathcal{N}$ the predicate ϕ has the following meaning:

1. The first conjunct expresses that the system net multiset $\widehat{M}$ corresponds to the pre-condition of the system net transition $\widehat{\tau}$, i.e. $\widehat{M} = \mathbf{pre}(\widehat{\tau})$.
2. In turn, a multiset $\widehat{M'}$ is produced, that corresponds to the post-set of $\widehat{\tau}$.
3. A multi-set $\vartheta(N)$ of object net transitions is enabled if the sum M_N of the net-token markings (of type N) enable it, i.e. $M_N \geq \mathbf{pre}_N(\vartheta(N))$.
4. The firing of $\widehat{\tau}[\vartheta]$ must also obey the *object marking distribution condition*: $M'_N = M_N - \mathbf{pre}_N(\vartheta(N)) + \mathbf{post}_N(\vartheta(N))$, where $\mathbf{post}_N(\vartheta(N)) - \mathbf{pre}_N(\vartheta(N))$ is the effect of the object net's transitions on the net-tokens.

Note that conditions 1. and 2. assure that only net-tokens relevant for the firing are included in λ and ρ. Conditions 3. and 4. allow for additional tokens in the net-tokens.

For system-autonomous events $\widehat{t}[\mathbf{0}]$ the enabling predicate ϕ can be simplified further. We have $\mathbf{pre}_N(\mathbf{0}(N)) = \mathbf{post}_N(\mathbf{0}(N)) = \mathbf{0}$. This ensures $\Pi^2_N(\lambda) = \Pi^2_N(\rho)$, i.e. the sum of markings in the copies of a net-token is preserved w.r.t. each type N. This condition ensures the existence of linear invariance properties

Analogously, for an object-autonomous event we have an idle-transition $\widehat{\tau} = id_{\widehat{p}}$ for the system net and the first and the second conjunct is: $\Pi^1(\lambda) = \mathbf{pre}(id_{\widehat{p}}) = \widehat{p} = \mathbf{post}(id_{\widehat{p}}) = \Pi^1(\rho)$. So, there is an addend $\lambda = \widehat{p}[M]$ in μ with $d(\widehat{p}) = N$ and M enables $\vartheta(N)$.

Definition 6 (Firing Rule). *Let OS be an Eos and $\mu, \mu' \in \mathcal{M}$ markings. The event $\widehat{\tau}[\vartheta]$ is enabled in μ for the mode $(\lambda, \rho) \in \mathcal{M}^2$ iff $\lambda \sqsubseteq \mu \wedge \phi(\widehat{\tau}[\vartheta], \lambda, \rho)$ holds.*

An event $\widehat{\tau}[\vartheta]$ that is enabled in μ for the mode (λ, ρ) can fire: $\mu \xrightarrow[OS]{\widehat{\tau}[\vartheta](\lambda,\rho)} \mu'$. The resulting successor marking is defined as $\mu' = \mu - \lambda + \rho$.

We write $\mu \xrightarrow[OS]{\widehat{\tau}[\vartheta]} \mu'$ whenever $\mu \xrightarrow[OS]{\widehat{\tau}[\vartheta](\lambda,\rho)} \mu'$ for some mode (λ, ρ).

Note that the firing rule makes no a-priori assumptions about how to distribute the object net markings onto the generated net-tokens. Therefore we need the mode (λ, ρ) to formulate the firing of $\widehat{\tau}[\vartheta]$ in a functional way.

Example Consider the Eos of Figure 4 again. The current marking μ of the Eos enables $\widehat{t}[N_1 \mapsto t_1, N_2 \mapsto t_2]$ in the mode (λ, ρ), where

$$\begin{aligned} \mu &= \widehat{p}_1[\mathbf{0}] + \widehat{p}_1[a_1 + b_1] + \widehat{p}_2[a_1] + \widehat{p}_3[a_2 + b_2] = \widehat{p}_1[\mathbf{0}] + \lambda \\ \lambda &= \widehat{p}_1[a_1 + b_1] + \widehat{p}_2[a_1] + \widehat{p}_3[a_2 + b_2] \\ \rho &= \widehat{p}_4[a_1 + b_1 + b_1] + \widehat{p}_5[\mathbf{0}] + \widehat{p}_6[c_2] \end{aligned}$$

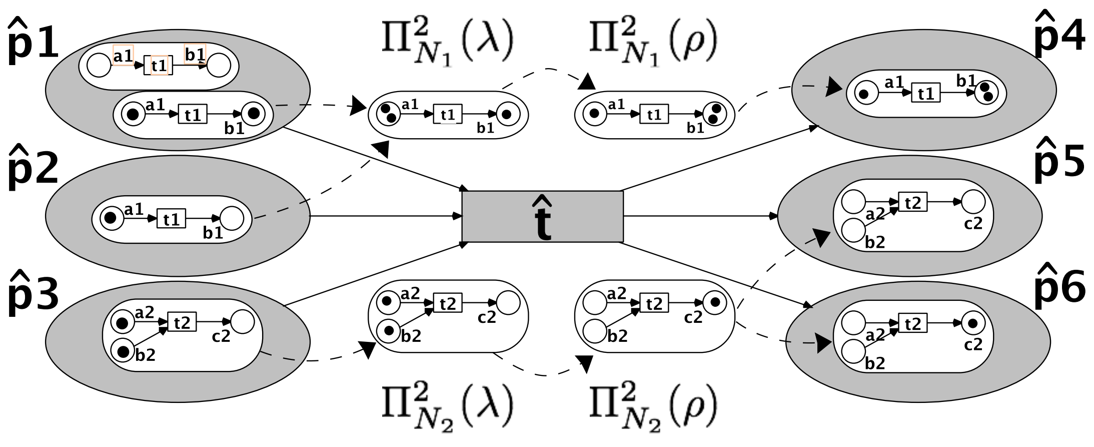


Figure 5: The EOS of Figure 4 illustrating the firing of $\widehat{t}[N_1 \mapsto t_1, N_2 \mapsto t_2]$ in the mode (λ, ρ)

The net-token markings are added by the projections Π^2_N resulting in the markings $\Pi^2_N(\lambda)$. The sub-synchronisation generates $\Pi^2_N(\rho)$. (The results are shown above and below the transition $\widehat{t}$.) After the synchronisation we obtain the successor marking μ' with net-tokens on $\widehat{p}_4$, $\widehat{p}_5$, and $\widehat{p}_6$ as shown in Figure 5:

$$\begin{aligned} \mu' &= (\mu - \lambda) + \rho = \widehat{p}_1[\mathbf{0}] + \rho \\ &= \widehat{p}_1[\mathbf{0}] + \widehat{p}_4[a_1 + b_1 + b_1] + \widehat{p}_5[\mathbf{0}] + \widehat{p}_6[c_2] \end{aligned}$$

Note, that we have only presented one mode (λ, ρ) and that other modes are possible, too.

B.2. Properties of the Firing Rule

In the following we relate those nested multisets, that coincide in their projections. The projection of a marking μ is defined as follows:

$$\Pi(\mu) := (\Pi^1(\mu), (\Pi^2_N(\mu))_{N \in \mathcal{N}}) \tag{10}$$

Obviously, there are several markings μ with the same projection, i.e. μ is not uniquely defined by $\Pi(\mu)$. The nested multisets, that coincide in their projections give rise to the equivalence $\cong \; \subseteq \mathcal{M}^2$, called *projection equivalence* defined by:

$$\begin{aligned} \alpha \cong \beta \quad :\Longleftrightarrow \quad & \Pi(\alpha) = \Pi(\beta) \\ \Longleftrightarrow \quad & \Pi^1(\alpha) = \Pi^1(\beta) \wedge \forall N \in \mathcal{N} : \Pi^2_N(\alpha) = \Pi^2_N(\beta) \end{aligned} \tag{11}$$

The relation $\alpha \cong \beta$ abstracts from the location, i.e. the concrete net-token, in which a object net's place p is marked as long as it is present in α and β. For example, for $d(\widehat{p}) = d(\widehat{p}')$ we have

$$\widehat{p}[p_1 + p_2] + \widehat{p}'[p_3] \cong \widehat{p}[p_3 + p_2] + \widehat{p}'[p_1]$$

which means that $\cong$ allows the tokens p_1 and p_3 to change their locations (i.e. between $\widehat{p}$ and $\widehat{p}'$).

Since an event collects all relevant object nets of the firing mode and combines them to one "virtual object net" that is only present at the moment of firing, the location of the object nets' tokens is irrelevant and can be ignored. These virtual object nets $\Pi^2_N(\lambda)$ are also shown in the example of Figure 4. This invariance can be expressed as follows:

Lemma 1. *The enabling predicate is invariant with respect to the relation* $\cong$*:*

$$\phi(\widehat{\tau}[\vartheta], \lambda, \rho) \iff (\forall \lambda', \rho' : \lambda' \cong \lambda \land \rho' \cong \rho \Longrightarrow \phi(\widehat{\tau}[\vartheta], \lambda', \rho'))$$

Proof From the definition of ϕ one can see that the firing mode (λ, ρ) is used only via its projection by Π. Since $\lambda' \cong \lambda, \rho' \cong \rho$ expresses equality modulo projection, the predicate ϕ cannot distinguish between λ' and λ, resp. ρ' and ρ. $\square$ qed.

As for p/t nets the firing rule has several nice properties

- Let OS be an Eos and μ a marking. The event $\widehat{\tau}[\vartheta]$ is enabled in the mode (λ, ρ) iff it is enabled in the mode (λ', ρ') such that $\lambda' \cong \lambda \land \rho' \cong \rho$. This is an immediate consequence of Lemma 1.
- Firing is reversible, i.e. for each Eos OS we obtain OS^{rev} by inverting the arcs and have: $\mu_1 \xrightarrow[OS]{\theta} \mu_2 \iff \mu_2 \xrightarrow[OS^{rev}]{\theta^{rev}} \mu_1$.
- The behaviour of an Eos is a conservative extension of p/t nets in the following sense: When one abstract from the net-tokens structure by the projection Π^1, then the behaviour of the system net in the Eos cannot be distinguished from the system net $\widehat{N}$ regarded as a p/t net, i.e. $\mu \xrightarrow[OS]{\widehat{t}[\vartheta]} \mu' \Longrightarrow \Pi^1(\mu) \xrightarrow[\widehat{N}]{\widehat{t}} \Pi^1(\mu')$.
- The invariance calculus for p/t nets can be extended to Eos in a compositional way, i.e. invariance equations can be obtained from the invariance equations of the constituent components separately.