Humboldt-Universität zu Berlin, Institut für Informatik,
Unter den Linden 6, 10099 Berlin, Germany

DFG-Graduiertenkolleg 1324 “METRIK”

Lehrstuhl Theorie der Programmierung
Group Theory of Programming

Oclets - a formal approach to adaptive systems using scenario-based concepts

Dirk Fahland

May 11, 2008

Usually, a component in a distributed system has assumptions about the remaining components of the system. A change in one component might require to change other components as well. It may happen that the change has to be performed in the running system. In this paper, we propose a formal model for systems that change their behavior at run-time: An *adaptive system* is denoted as a set of scenarios using a Petri net syntax. Our operational model provides an adaptation operator that synthesizes and adapts the system behavior as a Petri net branching-process at run-time based on the given scenarios. We show the feasibility of our approach by the help of an example.

Keywords: adaptive processes, scenario-based modeling, Petri nets, live-sequence charts

Contents

1. Introduction **5**

2. Concepts and notations **6**
2.1. Petri nets and basic notations 6
2.2. Introduction of the running example 6
2.3. Branching processes of Petri nets 7
2.3.1. Branching processes of merged nets. 9

3. A formal model for adaptive systems **11**
3.1. Adapting behavior by scenarios 11
3.2. Temperature-annotated Petri nets 11
3.3. Oclets – a Petri net syntax for scenarios 13
3.4. Oclets constitute an adaptive system 14
3.5. Dynamically constructed branching process 16
3.6. Semantics of oclets 16
3.6.1. Embedding of branching processes 16
3.6.2. Realized oclets 17
3.7. Semantics of adaptive systems 17
3.7.1. Firing transitions 18
3.7.2. Enabling oclets in a state 18
3.7.3. Adapting branching processes by oclets 19
3.7.4. Runs of an adaptive systems 20

4. Example revisited: adapting behavior to handle faults **21**

5. Conclusion **23**

A. Formal notations related to Petri nets **27**
A.1. Notations 27
A.2. Petri nets 27
A.3. Branching Processes 27

B. Formal notations related to Oclets **29**
B.1. Dynamically constructed branching processes 29
B.1.1. Dynamically constructed branching processes are sound 30
B.2. Correctness of adaptation steps 30
B.2.1. Properties of an enabling embedding 30

B.2.2. Properties of an adaptation step 33

1. Introduction

The need to formally specify systems the behavior of which must be changed at run-time is well-identified. Mostly, this concerns distributed systems which cannot be stopped, updated, and restarted, but must be continuously running. A system that changes its behavior at run-time is *adaptive*. Several formal models have been proposed to support specification and verification of systems and their changes to guarantee correctness, e.g. [14, 6, 11, 12]. The key idea is that modifying the behavior of a component is a system operation that is executed at runtime. Most approaches assume or require that the system to be changed is surrounded by a system that does the change to achieve a sound solution; their combination yields the adaptive system.

In this paper, we propose a formal model to directly specify distributed adaptive systems, based on Petri nets: We use *scenarios* to specify the behavior of a system. The system has some initial scenarios that specify the standard behavior. As the system is running, one may add scenarios specifying new behavior, or behavior that the system must not have. We contribute a generic *adaptation operator* that synthesizes the system behavior from the given scenarios at run-time as a Petri net *branching process*, by synchronization of smaller branching processes. The synthesis depends on each scenario's own *enabling condition*, and annotations that specify how different scenarios may synchronize in different situations. In each state, the actual branching process specifies the possible system behavior. We implement the adaptation operator as an *adaptation step* of the adaptive system. Thus, the system adapts its behavior by itself, based on its own current specification. A system modeler only has to declare the desired behavior of the system; the system dynamics then synthesizes an operational model at run-time.

The paper is organized as follows. We explain our approach with a running example of a locking-protocol. We explain Petri nets and their branching processes, and introduce the necessary notations and the running example in Sect. 2. We then define the syntax to denote scenarios in our setting, and introduce our model for an *adaptive system* together with the generic adaptation operator in Sect. 3. We then show the feasibility of our model by revisiting our example in Sect. 4. In Sect. 5 we discuss related work and conclude our approach.

2. Concepts and notations

2.1. Petri nets and basic notations

We reserve the symbol $\perp$ to be interpreted as *undefined*. We declare a *partial* function f from a domain A to a co-domain B by $f : A \to B^{\perp}$; a function $f : A \to B$ is *total*. If f is undefined for $a \in A$, we write $f(a) = \perp$. We use $\mathbb{N}$ to denote the set of natural numbers. We canonically lift a function $f : A \to B^{\perp}$ to a set of arguments, mapping to the *powerset* 2^B, and to multi-sets of arguments, mapping to the *multi-sets* $\mathbb{N}^B$.

Petri nets are a formalism for specifying concurrent, distributed systems in a graphical notation. For this paper, we assume the reader to be familiar with ordinary nets, and introduce notation only; we refer to [15] for a detailed introduction.

Let *Names* be a set of names. We denote a *labeled Petri net* as $N = \langle P, T, F, \ell \rangle$ with *places* P, *transitions* T, *arcs* $F \subseteq (T \times P) \cup (P \times T)$, and a partial *naming* function $\ell : P \cup T \to \mathit{Names}^{\perp}$. Let $\mathcal{P}$ and $\mathcal{T}$ denote the universe of all places and transitions, respectively.

A state of N is a *marking* $m : P \to \mathbb{N}$ of N that assigns each place a number of *tokens*; we conceive a marking as a multi-set. A *net system* $\Sigma = \langle N, m^0 \rangle$ is a labeled Petri net N with an initial marking m^0. We denote the *nodes* of N by $X := P \uplus T$, and write $\mathit{pre}(x)$ for the *pre-set* of x in N, and $\mathit{post}(x)$ for its *post-set*. The notion of enabling, the firing of a transition that yields a *step* of N, the successor marking, and the runs of a net system that produce the underlying transition system S_Σ of Σ are defined as usual.

2.2. Introduction of the running example

Figure 2.1 depicts a net system $\langle N_{ex}, m^0_{ex} \rangle$ in the standard graphical notation of Petri nets; we ignore the colors in Fig. 2.1 for now. The initial marking is $m^0_{ex} = [\![\mathsf{rdy}, \mathsf{av}]\!]$. Names are written next to transitions and places. Our example net N_{ex} specifies a lock-based access control mechanism restricting access to some operation op. In the initial marking m^0_{ex}, the access control is ready (rdy), and access to op is available (av).

In short, N_{ex} has the following behavior: A user requests access (rq) to operation op (for instance an operation on a database). The request is pending (pd) until it is granted (gr), or denied (dn).

Access is granted only if op is available (av): By gr, op becomes enabled (en), op becomes unavailable (un) for other users, and the access control returns from pending to rdy. The user can now execute op and, thus, finishes (fin) his work. By releasing the access rights (rl), op becomes accessible again.

While the first user has access rights to op, a next user may request access (rq is enabled). This request is denied: Instead of gr transition dn is enabled. By dn, op

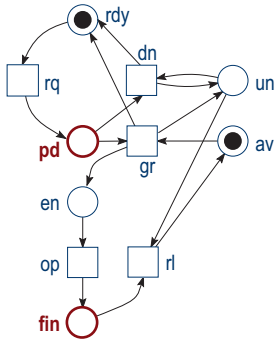

Figure 2.1.: A Petri net N_{ex} that initially marks places rdy and av.

remains unaccessible (un).

N_{ex} has some nice properties: No two users can access op at the same time. Every transition of N_{ex} can occur. The system always can return to its initial state (assuming weak fairness for rl). If op is not enabled or just finished (before releasing access rights), then a request is always denied.

These properties hold under the assumption that op is executed by a process that never fails. Assume that in the implementation of N_{ex}, op might fail s.t. it does not produce a token on fin. This is not expressed in N_{ex}, but could be caused by the environment of N_{ex}. If this happened, rl could not fire and access to op would be denied infinitely. In the next sections, we develop a model that allows us to express this failure, and to adapt the behavior of N_{ex} to recover from the failure.

2.3. Branching processes of Petri nets

Besides an underlying transition system, Petri nets have another semantical model called *branching processes*. We recall concepts and notations at an intuitive level; the reader is referred to [8] for a detailed introduction to branching processes.

The underlying transition system S_Σ of a net system Σ formalizes a run of Σ as a path (that may have cycles) in S_Σ that starts in m^0. S_Σ can structurally be unfolded into a tree S' with root m^0 where an acyclic path in S' represents an unfolded run s.t. any two runs diverge in S' iff they diverge in S_Σ, and two runs that diverge at some point, never meet again. S' is acyclic and may be infinite (if S_Σ has cycles).

Informally speaking, one can build a *branching process* (BP) of Σ by unfolding the structure of Σ along its arcs. A BP of Σ corresponds to its unfolded transition system, and is formalized as an acyclic Petri net. A BP contains subnets that correspond to the paths of the unfolded transition system; each such subnet is a *process* of Σ. Each process can be linearized into a run of S_Σ. We now explain how a given branching process corresponds to a net system at an intuitive level and define the necessary notations. The formal definitions are given in Appendix A.3.

Let $\Sigma = \langle N, m^0 \rangle$ be a net system with net $N = \langle P_N, T_N, F_N, \ell_N \rangle$. A branching process

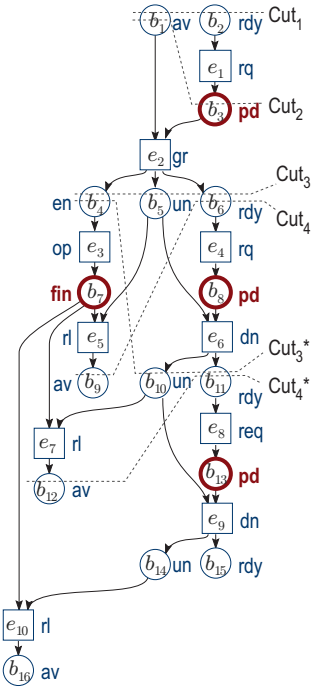

Figure 2.2.: A branching process β_{ex} of net N_{ex} of Fig. 2.1

$\beta = \langle B, E, F, \lambda \rangle$ of Σ is an acyclic, labeled Petri with some special properties.

The places B are called *conditions*, the transitions E are called *events*. Each node of β has only finitely many predecessors in β wrt. F: Let $x, y \in X$, x is a *causal predecessor* of y, $x \leq y$, iff there exists a path from x to y along F.

The *labeling* function $\lambda : X \to X_N$ labels each condition of β with exactly one place of N, and each event of β with exactly one transition of N; we use a different terminology and notation to avoid confusion with 'normal' nets. As an example, the BP β_{ex} of Fig. 2.2 is a branching process of the net system Σ_{ex} of Fig. 2.1; in β_{ex}, $b_1 \leq_{ex} b_{16}$.

The labeling λ is a homomorphism from β to N in the following sense: A condition b represents a specific token on place $p = \lambda(b)$. An event e represents an *occurrence*, i.e. a specific firing, of transition $t = \lambda(e)$. An arc (b, e) denotes that an occurrence of t will consume a specific token from p. An arc (e, b) denotes that an occurrence of t will produce a specific token on p.

Two events e_1, e_2 that a have a common pre-condition are in *direct conflict*. If e_1 and e_2 are in direct conflict, then each x_1 and each x_2 with $e_1 \leq x_1$, $e_2 \leq x_2$ are in *conflict*. Nodes that are neither conflicting, nor causally ordered, are *concurrent*. A set $X' \subseteq X$ is *causally closed* if X' contains all predecessor of its members: $\lfloor X' \rfloor := \{x \in X \mid \exists x' \in X' : x \leq x'\} \subseteq X'$. For example, in β_{ex}, e_3 and e_4 are concurrent, and e_5 and e_6 are in direct conflict.

Each maximal set $Cut \subseteq B$ of concurrent conditions, called *cut*, represents a *reachable* marking $\lambda(Cut)$ of Σ. Each causally closed, conflict-free set $C \subseteq E$ of events, called

configuration, represents occurrences of transitions $\lambda(C)$ that can be linearized into a run of Σ. Configurations and cuts of β correspond to each other: A configuration C induces a prefix of β, which we denote by $\beta(C)$. $\beta(C)$ is a *process* of Σ and describes a run of Σ by the events of C. Two configurations C_1, C_2 where $C_1 \cup C_2$ is not conflict-free define two diverging processes $\beta(C_1)$ and $\beta(C_2)$ of Σ, therefore β is called 'branching' process. The maximal conditions of $\beta(C)$ constitute the $Cut(C)$ that represents the marking which is reached in the run, $m(C) := \lambda(Cut(C))$. We explain these concepts by the help of our example.

The *initial* configuration is the empty set. The set of *minimal conditions* $Min(\beta)$ that have no pre-events is the *initial* cut, $Cut(\emptyset)$, which represents the initial marking. In our example, $Cut_1 = Min(\beta_{ex}) = \{b_1, b_2\}$, with $\lambda_{ex}(Cut_1) = m^0_{ex} = [\![\mathsf{rdy}, \mathsf{av}]\!]$.

Let C be a configuration of β. Let t a transition of N for which $Cut(C)$ contains conditions labeled with all pre-places of t. Then t is *enabled* in $Cut(C)$, because $pre_N(t) \subseteq m = \lambda(Cut(C))$. In our example, transition rq with $pre_{N_{ex}}(\mathsf{rq}) = \{\mathsf{av}\}$ is enabled in Cut_1.

If t is enabled in $m(C)$, then β has an event $e \notin C$, $\lambda(e) = t$, that represents the occurrence of t in $m(C)$. That is, e consumes a token from each pre-place of t, $\lambda(pre(e)) = pre_N(t), pre(e) \subseteq Cut(C)$, and produces a token on each post-place of t, $\lambda(post(e)) = post_N(t)$. We say that e *extends* C, denoted as $C \oplus e$. The *extension* of C by e is a configuration $(C \oplus e) := C \uplus \{e\}$.

Extending C by e yields the *successor* cut, $Cut(C \oplus e)$ that represents the successor marking $m(C \oplus e)$, and the step $m(C) \xrightarrow{t} m(C \oplus e)$ of Σ. In our example, event e_1 represents the occurrence of rq in Cut_1 yielding $Cut_2 = Cut(\{e_1\})$. From there $Cut_3 = Cut(\{e_1, e_2\})$ can be reached by the occurrence e_2 of gr. The process $\beta_{ex}(\{e_1, e_2\})$ that describes a run of Σ_{ex} by firing rq and gr is the subnet of β_{ex} that contains the nodes $b_1, \ldots, b_6, e_1, e_2$.

A Petri net that has the structural properties of a BP but an arbitrary labeling is called *occurrence net*. It is important to note that a configuration is a concept on top of BPs to express how a BP corresponds to the runs of a net system. That is, a configuration tells which behavior has been *observed* in a run. It is the structure of a BP alone that denotes the possible behavior of the system. For instance, the branching process β_{ex} does not describe the complete behavior of Σ, but only a finite part of it. The maximal BP $\beta(\Sigma)$ of Σ is called *unfolding*. In [8] is shown that one can construct a *complete, finite prefix* $\beta^{pre}(\Sigma)$ of $\beta(\Sigma)$ that equivalently describes the entire behavior of Σ, if Σ has not infinitely many reachable states.

2.3.1. Branching processes of merged nets.

Now assume that in $\Sigma = \langle N, m^0 \rangle$, the net N is structurally composed by merging pairwise disjoint nets $A = \langle N_1, \ldots, N_n \rangle$ at equally labeled transitions and places. Merging transitions t_1 and t_2 results in a transition t that consumes from the pre-places of t_1 and t_2 and produces on their post-places; merging sets of transitions and (sets of) places is correspondingly defined [3]. For each node $x \in X$, let $\phi(x)$ denote the set of nodes that was merged into x (while preserving sorts); we call ϕ the *synchronization mapping*.

Let $X_A = \biguplus_{i=1}^{n} X_i$ denote the set of all nodes of the nets $N_1, \ldots, N_n$. (We apply the index of a structure, e.g. N_i, implicitly to its parts, e.g. P_i and X_i.) As a notational convention, we write $[n] := \{1, \ldots, n\}$.

Definition 1 (Safe merging). Let $A = \langle N_1, \ldots, N_n \rangle$ be an enumeration of labeled Petri nets. And let N be the result of merging A by a synchronization mapping $\phi : X_N \to 2^{X_A}$.

The mapping ϕ is *safe* if no two nodes of the same net N_i are merged together, i.e. $\forall x \in X_N \forall i \in [n] : |\phi(x) \cap X_i| \leq 1$. Merging the nets A by ϕ is *safe* if ϕ is safe.. $\star$

A safe ϕ induces mappings $\phi = \langle \phi_1, \ldots, \phi_n \rangle$ with $\phi_i(x) := x'$ if $\{x'\} = \phi(x) \cap X_i$ and $\phi_i(x) :=\perp$ otherwise, for $i = 1, \ldots, n$.

Let β be a BPof N. An event $e \in X_\beta$ realizes a synchronous occurrence of transitions $\phi(\lambda(e))$. A condition b realizes a shared token on the places $\phi(\lambda(b))$ being produced (consumed) by synchronously occurring transitions. Because of the structural merging of A into N, each mapping ϕ_i is a structural homomorphism from β to N_i in the sense of the labeling of a BP.

Subsequently we propose an approach to construct branching processes by merging events and conditions. We thereby do not merge by a structural composition of nets as written above. We rather determine events, and conditions to be merged dynamically based on observed behavior of a net.

3. A formal model for adaptive systems

We will now present our approach for specifying systems which adapt their behavior at runtime. The central entity of our approach is the *scenario*: A scenario describes a possible course of (future) actions and the therein involved resources in the *context* of a larger system. 'Possible' means that the described course of actions is not necessarily part of the system, and might require some initiative to be realized: The current behavior of the system must be *adapted* if the scenario shall happen.

3.1. Adapting behavior by scenarios

We use Petri nets to specify *adaptive systems*: The initial behavior of an adaptive system A is a BP β^0 of a Petri net system Σ which realizes the 'normal' behavior of A. Σ is supplemented with further annotated BPes, called *oclets*, that specify behavioral *scenarios* of A different from β^0.

We adapt the initial BP β^0 while A is executed: A may fire enabled transitions of β^i to follow its behavior. And A may adapt β^i to β^{i+1} by *synchronizing* an oclet's BP with β^i in order to *realize* a scenario. It is necessarily the case that β^i might no longer be a BP of the starting point Σ.

A scenario is realized in a context, i.e. β^i, but not every context suits every scenario. Therefore, an oclet has to provide information whether, and how, a scenario fits into a given context.

To this end we provide expressive means to denote the importance of an action or a resource by a *temperature*, and means to specify *behavioral assumptions* of a scenario including a notion of *abstraction* in the subsequent sections. We then define our semantical model for adaptive processes, and define what it means to *realize* a scenario in a BP β^i which transforms it into β^{i+1}. This yield a generic *adaptation step* that becomes part of the system dynamics like the step of a Petri net that fires a transition.

3.2. Temperature-annotated Petri nets

We now define the temperature of a transition or place which we use when denoting a scenario. The semantics of a temperature affects BPes and their synchronization. Let $\mathit{Temp} := \{\mathit{cold}, \mathit{hot}, \neg\mathit{cold}, \neg\mathit{hot}\}$ be the set of *temperatures*.

Definition 2 (Temperature-annotated Petri net). A *temperature-annotated* Petri net $N = \langle P, T, F, \ell, \vartheta \rangle$, TA-net for short, is a labeled net $\langle P, T, F, \ell \rangle$ with a partial *temperature-annotation* $\vartheta : (P \cup T) \to \mathit{Temp}^{\perp}$. $\star$

Let $\Sigma = \langle N, m^0 \rangle$ be a net system of a TA-net N where *each* node has a temperature. The temperature carries over to each BP $\beta = \langle B_\beta, E_\beta, F_\beta, \lambda, \vartheta_\beta \rangle$ of Σ, where $\forall x \in X_\beta : \vartheta_\beta(x) = \vartheta_N(\lambda_\beta(x))$; we call β a *temperature-annotated* BP, TA-BP for short.

Definition 3 (Semantics of temperatures). Let $\Sigma = \langle N, m^0 \rangle$ be a TA-net system. The temperature of a node of N is *realized* in β of Σ as follows:

1. (**T-hot**) If a *hot* transition $t \in T_N$ is enabled, it *must fire*; t is *realized* in β as a *hot* event e if $\lambda(e) = t$ and e is not in direct conflict with another event of β.
2. (**T-cold**) A *cold* transition $t \in T_N$ *may fire* if it is enabled; t is *realized* in β as a *hot* or *cold* event e if $\lambda(e) = t$.
3. (**P-hot**) A token on a *hot* place $p \in P_N$ *must be consumed*; p is *realized* in β as a *hot* condition b if $\lambda(b) = p$ and $post_\beta(b) \neq \emptyset$.
4. (**P-cold**) A token on a *cold* place $p \in P_N$ *may be consumed*; p is *realized* in β as a *hot* or *cold* condition b if $\lambda(b) = p$.
5. (**not-T-hot**) A $\neg$*hot* transition $t \in T_N$ *must not fire*. t is *realized* in β if there is no event labeled with t.
6. (**not-T-cold**) A $\neg$*cold* transition $t \in T_N$ *may not fire* if its enabled; t is *realized* in β as a *cold* event e if $\lambda(e) = t$.
7. (**not-P-hot**) A token on a $\neg$*hot* place $p \in P_N$ *must not be consumed*; p is *realized* in β as a *cold* condition b if $\lambda(b) = p$ and $post_\beta(b) = \emptyset$.
8. (**not-P-cold**) A token on a $\neg$*cold* place $p \in P_N$ *may not be consumed* (but it is not forbidden to consume it); p is *realized* in β as a *cold* condition b if $\lambda(b) = b$. $\star$

We denote a *hot* node with bold (red) lines, and *cold* node with thin (blue) lines as shown in Fig. 2.1 and Fig. 2.2; $\neg$*hot* or $\neg$*cold* nodes will be depicted in a special context only.

Definition 4 (Consistency of TA-BPes). Let β be a TA-BP. β *realizes its temperatures* if each node $x \in X_\beta$ realizes the temperature of its label $\lambda(x)$. β *realizes* a TA-net-system Σ if β is a BP of Σ that realizes its temperatures. $\star$

It follows from the above definitions that if in Σ any two transitions with a common pre-place are *cold* and each initially marked place and each place without a post-transition is *cold* then $\beta(\Sigma)$ realizes Σ.

Observe that (T-cold), (P-cold) (and (not-T-cold), (not-P-cold)) correspond to the standard semantics of Petri nets. All other annotations have additional requirements on the BP. These requirements are defined *without* references to the structure of a net system Σ. This will become important, when synchronizing two different BP: the result of the synchronization has to realize the temperatures.

3.3. Oclets – a Petri net syntax for scenarios

We now introduce *oclets*, which are a special class of temperature-annotated Petri nets. The name originates from the idea to specify occurring behavior of a larger system by small snippets. Oclets provide the expressive means to specify a scenario. An oclet will be realized as a partial execution of a larger BP by synchronizing events and conditions.

Definition 5 (Oclet). An *oclet* $o = \langle P, T, F, \ell, \vartheta, con, mark, type \rangle$ is a TA-net with:

- The net $\langle P, T, F, \ell \rangle$ has the structure of an occurrence net, but $\ell : X \to (Names \setminus (\mathcal{T} \cup \mathcal{P}))^{\perp}$ does not label nodes of o with transitions or places.
- o has a causally closed *precondition* where nodes have no temperature; $con \subseteq X, \lfloor con \rfloor \subseteq con, \forall x \in con : \vartheta(x) = \perp$.
- o has set of places, that are a subset of the causally maximal places of con; $mark \subseteq (con \cap P), post(mark) \cap con = \emptyset$; $mark$ is the *enabling* marking of o.
- $type \in \{universal, existential\}$. ⋆

By labels and temperatures, the nodes of an oclet o (and of any other TA-net) are partitioned into *active* nodes, *passive* nodes (no temperature), and *abstract* nodes (no name and no temperature): $active := \{x \mid \vartheta(x) \neq \perp, \ell(x) \neq \perp\}$; $passive := \{x \mid \vartheta(x) = \perp, \ell(x) \neq \perp\}$; $abstr := \{x \mid \vartheta(x) = \perp \wedge \ell(x) = \perp\}$, i.e. each node with a temperature, must have a name. Observe that $con \subseteq passive \cup abstr$.

Figure 3.1 depicts some oclets: The precondition and the type of an oclet is drawn above the dashed line; the places of the enabling marking carry a token; passive nodes are grey; and abstract nodes have no label. If anti is written below the dashed line, an active bold (red) node is $\neg hot$, and an active thin (blue) node is $\neg cold$.

Because $o = \langle P, T, F, \ell, \vartheta, con, mark, type \rangle$ is an occurrence net, the labeling $\lambda_o(x) := x$ for all $x \in X_o$ constitutes the maximal BP $\langle P, T, F, \lambda_o \rangle$ of o. For mere technical reasons, we will work with an isomorphic copy that is disjoint from this BP; let β_o denote this copy.

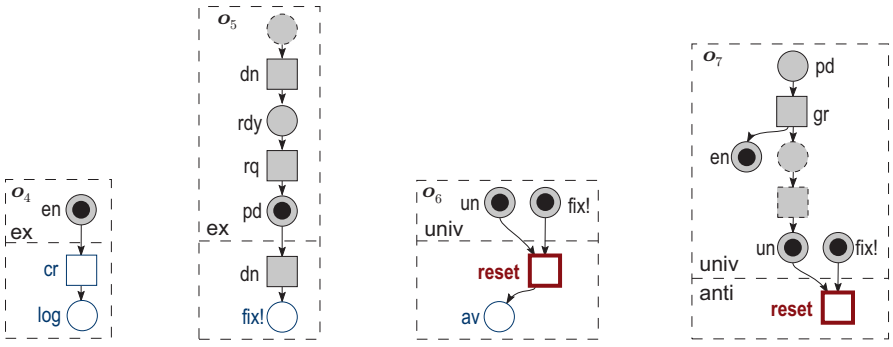

Figure 3.1.: Oclets o_4, o_5, o_6, o_7

An oclet o allows to specify a scenario in terms of Petri nets; β_o denotes a possible course of actions. Thereby a node of o is active if some initiative for its realization might be necessary; its temperature denotes how it shall be realized. A passive node of o is assumed to be realized by 'someone else'. Abstract nodes are similar to passive nodes; but we only assume *that* something is happening in a certain causal order without exactly specifying *what*.

An oclet's scenario will be realized in the context of another BP β by synchronization. Passive and abstract nodes together describes the *assumptions* that o has on β in order to be realized. The precondition of o specifies the behavior that must have been observed to enable the entire scenario. The enabling marking further restricts this precondition to a specific (partial) state by requiring that certain resources are still available. A universal oclet must be realized in a context as soon as all assumptions hold, for an existential oclet, one may choose to realize it; depending on the temperatures of nodes, behavior is added or removed.

For the remainder of this paper, we only work with a special class of oclets.

Definition 6 (Well-formed oclet, Normal oclet, Anti-oclet). An oclet o is *well-formed* if

1. o is connected and conflict-free;
2. β_o realizes the temperatures of o;
3. o has either temperatures $\bot, \mathit{hot}, \mathit{cold}$ – then o is called a *normal* oclet –, or temperatures $\bot, \neg\mathit{hot}, \neg\mathit{cold}$ – then o is called an *anti*-oclet –;
4. each active node has a non-empty pre-set of active and passive nodes;
5. no active node of o precedes a passive or an abstract node of o;
6. each *hot* transition of o has only active pre-places; and
7. if o is *existential* then all maximal nodes of *con* are places that are marked by *mark* (i.e. existential oclets are enabled in a specific state only). $\star$

Effectively, an anti-oclet denotes that the specified behavior may not, or must not be realized; o_7 of Fig. 3.1 is an anti-oclet. The BP β_o of a well-formed oclet o is has no conflicts and hence is the only process of o.

3.4. Oclets constitute an adaptive system

We now explain how an oclet o is related to a Petri net system Σ. From there, we define our notion of an adaptive system that specifies normal and exceptional behavior.

Each net system Σ can be translated into a set of oclets: For each transition $t\ in T_\Sigma$ define the oclet $o(t)$ with $T_o = \{t\}$, $P_o = \mathit{pre}_\Sigma(t) \cup \mathit{post}_\Sigma(t)$, $F_o = F_\Sigma \cap (X_o \times X_o)$, $\ell_o = \ell_\Sigma(x)$ for all $x \in X_o$, $\mathit{con}_o = \mathit{mark}_o = \mathit{pre}_o(t)$, $\vartheta_o(x) = \mathit{cold}$ for all $x \in X_o \setminus \mathit{con}_o$, $\mathit{type}_o = \mathit{universal}$. Let o_t denote an isomorphic, disjoint copy of $o(t)$; o_t is just a different

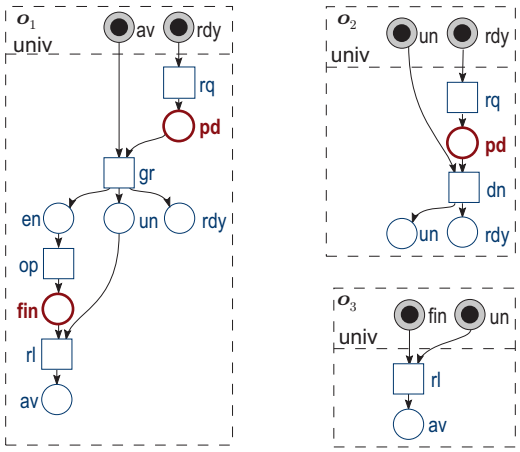

Figure 3.2.: Oclets o_1, o_2, o_3

encoding of a Petri net event for transition t. Its precondition denotes the (partial) marking where t is enabled. If $T_\Sigma = \{t_1, \ldots, t_n\}$, the oclets $O_{T_\Sigma} = \langle o_{t_1}, \ldots, o_{t_n} \rangle$ provide the basic building blocks to construct any branching process of Σ [8].

Intuitively, the oclets of O_{T_Σ} can be composed to larger, logical units of behavior. Assume Σ is a net system that has a complete finite prefix β^{pre} of its unfolding [8]; this prefix has only finitely many different processes: $\beta^{pre}(C_1), \ldots, \beta^{pre}(C_k)$. Merging and concatenating these processes yields the behavior of Σ. Each process can be conceived as a normal scenario of Σ.

Because a process is an occurrence net without conflicts, we can equivalently translate the processes $\beta^{pre}(C_i)$ into a set of well-formed oclets $O_\Sigma = \langle o_1, \ldots, o_k \rangle$ with the initial marking m^0_Σ. Just like $\beta^{pre}(C_i)$ is composed of Petri net events, each $o_i \in O_\Sigma$ can be conceived as a composition of oclets from O_{T_Σ}.

For example, consider oclets $o_1, \ldots, o_3$ in Fig. 3.2. Starting with o_1, and appending oclets at equally named conditions yields β_{ex} of Fig. 2.2.

By adding another well-formed oclet o to O_Σ we extend the behavior specified in O_Σ if o is used in constructing a BP β of $O_\Sigma \uplus \{o\}$. By constructing β during an execution, and by making the use of o depend on this execution (by its precondition), we get adaptive system behavior. Note that this idea distinguishes our approach from a BP of a structurally merged net, see Sect. 2.3.1.

Definition 7 (Adaptive system). We call $A = \langle O_A, m^0_A \rangle$ an *adaptive system* if $O_A = \langle o_1, \ldots, o_n \rangle$ is an enumeration of pairwise disjoint, well-formed oclets (usually sharing some names), and m^0_A is a multiset of names occurring in A. The set of places of A is $P_A := \biguplus_{i=1}^{n} P_i$ which contains no abstract places; let T_A, F_A, ℓ_A be correspondingly defined. $\star$

A state of A will be a branching process (in a liberalized notion) together with a configuration of the branching process. Extending the configuration fires transition of A, changing the branching process adapts the behavior of A.

3.5. Dynamically constructed branching process

To describe the behavior of an adaptive system, we use a liberalized notion of branching processes. In this section we only present an intuitive characterization. The technical details are rather subtle, and given formally in Appendix B.1. Let $A = \langle O_A, m_A^0 \rangle, O_A = \langle o_1, \ldots, o_n \rangle$ be an adaptive system in the following.

We call our liberalized notion of TA-BP a *dynamically constructed branching process* (Dbp) $\beta = \langle B, E, F, \lambda, \vartheta, \ell \rangle$. Like a TA-BP, β is structurally an occurrence net that realizes its temperatures.

Events (conditions) of β are labeled with sets of transitions (places) of A by a safe-labeling $\lambda : X \to 2^{X_A}$ of A, $\lambda = \langle \lambda_1, \ldots, \lambda_n \rangle$, each λ_i may be partial, see Sect. 2.3.1. Thereby each node of β must be labeled with equally named nodes of A. Thus the name $\ell(x)$ of a node $x \in X$ is canonically inherited from the name $\ell_{o_i}(x_i)$ of its labels $x_i \in \lambda(x) \cap X_{o_i}$. Like the labeling of a BP is not arbitrary, a Dbp has similar constraints on λ to give a sound correspondence to the nets of A. The notion of active, passive and abstract nodes carries over from oclets to Dbp by their definition.

An event $e \in E_\beta$ represents the synchronous occurrence of the transitions $\{t_1, \ldots, t_k\} = \lambda_\beta(e)$. A condition $b \in B_\beta$ represents a common token on the places $\{p_1, \ldots, p_l\} = \lambda_\beta(b)$. Put differently, an event e synchronizes k occurrences of the transitions $t_1, \ldots, t_k$, while a condition synchronizes l tokens on the places $p_1, \ldots, p_l$. Thereby, each transition and each place is contributed by a different oclet.[1] A Dbp β where $Min(\beta)$ is not named with m_A^0, that is $\ell(Min(\beta)) \neq m_A^0$, is called *partial* Dbp (p-Dbp). By definition, each net system Σ has a maximal Dbp $\beta(\Sigma)$, and each oclet o has a maximal p-Dbp $\beta(o)$ where labels are singleton sets of nodes of Σ or o, respectively.

3.6. Semantics of oclets

We sketched in Sect. 3.1 that a scenario specifies behavior in some context. In this section, we define what it means that an oclet's scenario is *realized* in the context of a Dbp. Summarized, an oclet is realized in a Dbp β if it is *embedded* as an isomorphic subgraph of β (except for abstract nodes), and if β realizes the temperatures of the oclet.

3.6.1. Embedding of branching processes

In the following, let A be an adaptive system, let β_1 be a p-Dbp of A, and β_2 a Dbp of A. Let $\beta_1^{emb} := \beta_1[active_1 \cup passive_1]$ be the restriction of β_1 to its active and passive nodes, that is the induced subgraph by the nodes $active_1 \cup passive_1$.

Definition 8 (Embedding). β_1 is *embedded* in β_2 if β_1^{emb} is an isomorphic subgraph of β_2 by isomorphism h where

1. $\forall x, y \in X_1^{emb} : x \leq_{\beta_1} y \Leftrightarrow h(x) \leq_{\beta_2} h(y)$;

[1] The main difference to BPes is that we allow that an event consumes (produces) tokens that are shared by overlapping sets of places. Thus the λ_i are only surjective wrt. an event. For a sound definition we require that λ is injective.

2. x and $h(x)$ have equal names; and

3. $\forall x \in X_1^{emb} \forall i \in [n] : (\lambda_{1,i}(x) \neq \perp \wedge \lambda_{2,i}(h(x)) \neq \perp) \Rightarrow \lambda_{1,i}(x) = \lambda_{2,i}(h(x))$.

We call isomorphism h an *embedding* of β_1 in β_2, that embeds X_1^{emb} into $h(X_1^{emb}) \subseteq X_2$. The inverse of h is h^{-1}. ⋆

The abstract nodes of β_1 contribute to an embedding by requiring further causal ordering in β_2. If β_1 is embedded in β_2 then the ordering of β_1 is also present in β_2, there realized by other nodes that are not embeddings of β_1. In this sense the abstract nodes of β_1 abstract from the concrete structure of β_2.

Definition 9 (Realizing embedding). Let β_1 be embedded in β_2 by embedding h. We say that a node $x \in X_1^{emb}$ is *realized* in $h(x)$ if $\lambda_1(x) \subseteq \lambda_2(h(x))$. β_1 is realized in β_2 if all nodes of X_1^{emb} are realized; in this case h is a *realizing* embedding. ⋆

3.6.2. Realized oclets

We now use the notion of a realizing embedding to define the semantics of an oclet. An oclet specifies the set of branching processes where it is *realized* as follows.

Definition 10 (Realized normal oclet). A normal oclet $o \in O_A$ is *realized* in DBP β_2 by the realizing embedding h if h realizes β_o in $h(X_o^{emb}) = Y \subseteq X_2$ s.t. each node $y \in Y$ realizes the temperature of $h^{-1}(y)$ as defined in Sect. 3.2. ⋆

Observe that this definition allows that a *hot* node of β_2 realizes a *cold* node of o. An anti-oclet is realized in β_2 if a largest prefix of its BP is realized in β_2 s.t. the temperatures are not violated.

Definition 11 (Prefix of an oclet). Let $o \in O_A$. Each (possibly empty), causally closed set $X^* \subseteq X_o$ of o together with $X^{**} = \mathit{passive} \cup \mathit{abstr}$ yields a *prefix* $\gamma(X^*) := \beta_o[X^* \cup X^{**}]$ of β_o. Let Γ_o be the set of all prefixes of β_o. ⋆

Definition 12 (Realized anti-oclet). An anti-oclet $o \in O_A$ is *realized* in DBP β_2 by a realizing embedding h if for the largest prefix $\gamma \in \Gamma_o$ of β_o that is embedded in β_2, h realizes X_γ in $h(X_\gamma) = Y \subseteq X_2$ s.t. each node $y \in Y$ realizes the temperature of $h^{-1}(y)$ as defined in Sect. 3.2. ⋆

3.7. Semantics of adaptive systems

We now define the semantics of an adaptive system A. An adaptive system A exhibits two kinds of dynamics: Firing transitions or adapting its behavior. Let $A = \langle O_A, m_A^0 \rangle$ be an adaptive system in the following.

Definition 13 (State of an adaptive system). A *state* $s = \langle \beta, C \rangle$ of an adaptive system A has a DBP β of A and a configuration C of β. The initial state of A is $s_A^0 = \langle \beta(m_A^0), \emptyset \rangle$ where $\beta(m_A^0)$ is the DBP that has a unique p-labeled event for each occurrence of $p \in m_A^0$. ⋆

3.7.1. Firing transitions

The dynamics of an adaptive system $A = \langle O_A, m_A^0 \rangle$ due to its transitions is essentially the dynamics of a net system as they are defined by branching processes. The difference are that an event realizes *sets* of transitions firing synchronously, and that that this firing may occur only, if it is already realized in a state s.

Definition 14 (Firing transition of an adaptive system). Let $T \subseteq T_A$ be a set of transitions of T_A. In a state $s_1\langle \beta, C \rangle$, the transitions T are *synchronously enabled* iff there exists an event $e_T \in E_\beta$ with $C \oplus e_T$, and $\lambda_\beta(e_T) = T$. If T is enabled in s_1 (by event e_T), then T may *fire synchronously* which yields the successor state $s_2 = \langle \beta, C \oplus e_T \rangle$ and the *transition-step* $s_1 \xrightarrow{T} s_2$ of A. The state s_2 is a state of A because β is not changed. $\star$

3.7.2. Enabling oclets in a state

Let $o \in O_A$ be an oclet, and let $s_1 = \langle \beta_1, C_1 \rangle$ be a state of A. In order to let o be enabled in s, we have two requirements: Firstly, a largest prefix $\gamma^{en} \in \Gamma_o$ of β_o can be embedded in s_1 (including con_o) by an embedding h^{en}. Secondly, o is not completely realized in β. When choosing the prefix γ^{en} we have to be careful.

If o is a normal oclet, the nodes $X_o \backslash \gamma^{en}$ shall be realized as new nodes by an adaptation step. If o is an anti-oclet $\gamma^{en} \cap active_o$ may or will be removed. This adding and removing must not be arbitrary in order to get a DBPthat realizes its temperatures. To this end γ^{en} has to satisfy some properties:

- The enabling embedding h^{en} must *respect realization* in β_1, that is h^{en} has to satisfy the following property: For every transition t in γ^{en}, for every (pre-) post-place p of t in γ^{en}, if $h^{en}(t)$ has a (pre-) post-condition b that already realizes $p \in \lambda_1(b)$, then $h^{en}(p) = b$.
- If o is a normal oclet the prefix γ^{en} must leave at least once active node of o that is not yet realized in β_1: If $active_o \setminus \gamma^{en} \neq \emptyset$ we call h^{en} *active*.
- Further, if o is a normal oclet, h^{en} must not embed a pre-place p of a hot transition t of o as a pre-condition $h^{en}(p) = b$ of a differently named hot event e of β_1 (otherwise we would violate temperatures for b in the adaptation step); we call h^{en} *aware of temperatures* in this case.
- For an anti-oclet, h^{en} has to embed at least one active node to allow its removal. Then h^{en} is called *active*.

Formal definitions for these properties are given in App. B.2.1.

Definition 15 (Enabled oclet). An oclet $o \in O_A$ is *enabled* in $s_1 = \langle \beta_1, C_1 \rangle$ at $Y^{en} \subseteq X_1$ iff there exists a largest prefix $\gamma^{en} \in \Gamma_o$ of β_o and an embedding h^{en} of γ^{en} that respects realization (with $Y^{en} = h^{en}(X_\gamma^{en})$) in β_1 s.t.

1. the precondition of o has already been observed in s_1: $h^{en}(con_o) \subseteq \beta_1(C_1)$;
2. marked conditions of o are in the current cut: $h^{en}(mark_o) \subseteq Cut_{\beta_1}(C_1)$, and
3. if o is a normal oclet then h^{en} is active and aware of temperatures;
4. if o is an anti oclet then h^{en} is active; ⋆

We call h^{en} an *enabling embedding*. It can be proven that our definition of enabling an oclet corresponds to the requirements we gave in the beginning of this section; see Prop. 1 in App. B.2.

3.7.3. Adapting branching processes by oclets

Let $s_1 = \langle \beta_1, C_1 \rangle, s_2 = \langle \beta_2, C_2 \rangle$ be states of A and let $o \in O_A$ be enabled in s_1 at $Y^{en} \subseteq X_1$ via enabling embedding h^{en}. To perform an *adaptation step*, $s_1 \xrightarrow{o} s_2$ of A based on h^{en}, proceed as follows.

Definition 16 (Adaptation step of a normal oclet). Let $o \in O_A$ be a normal oclet that is enabled in state $\langle \beta_1, C_1 \rangle$.

- Merge a copy β^* of $\beta_o[active_o \cup passive_o]$ without abstract nodes with β_1 along the enabling embedding h^{en}; resulting in β_2.
- The isomorphism h from $\beta_o[active_o \cup passive_o]$ to its merged copy β^* is the embedding of o in β_2.
- Then realize the temperatures of o in β_2: For each hot transition $t \in T_o$, remove each directly conflicting event e' of $e = h(t)$, and all causally depending nodes $y, e' \leq_{\beta_2} y$.
- Finally update labels, names, and temperatures according to Dbp.

This yields the adaptation step $\langle \beta_1, C_1 \rangle \xrightarrow{o} \langle \beta_2, C_2 \rangle$. ⋆

Definition 17 (Adaptation step of an anti-oclet). Let $o \in O_A$ be an anti-oclet that is enabled in state $\langle \beta_1, C_1 \rangle$.

- For each hot transition $t \in T_o$ that is embedded in β_1, remove from β_1 the event $e = h^{en}(t)$, and all nodes that causally depend on e.
- For each hot place $p \in P_o$, remove from β_1 all nodes that causally depend on $b = h^{en}(p)$, except b itself. This results in β_2.
- Restrict the embedding h^{en} to h as the nodes of β_1 are removed.
- Update labels, names, and temperatures according to Dbp.

This yields the adaptation step $\langle \beta_1, C_1 \rangle \xrightarrow{o} \langle \beta_2, C_2 \rangle$. ⋆

One can show that an adaptation step $s_1 \xrightarrow{o} s_2$ of A yields a DBP β_2 of A, that does not modify β_1 before the $Cut(C_1)$, and that a node gets added or removed in an adaptation step only, if there is a 'reason' for this in o; see Prop. 6 and Thm. 3 in App. B.2. The adaptation step can be implemented using matching of isomorphic subgraphs, and the net addition and subtraction operations of [3].

3.7.4. Runs of an adaptive systems

A sequence $\langle \alpha_0, \alpha_1, \ldots \rangle$ of sets of transitions and oclets is a *possible* execution of A if there are states $s^0, s^1, \ldots$, $s^0 = s^0_A$ and for all $i = 0, 1, 2, \ldots$ holds: If $\alpha_i = T_i \subseteq T_A$ is a set of transition of A, then $s^i \xrightarrow{T_i} s^{i+1}$ is a transition step of A. If $\alpha_i = o_i \in O_A$ is an oclet of A then $s_i \xrightarrow{o_i} s^{i+1}$ is an adaptation step of A.

An oclets type (existential or universal) specifies that an oclet may or must participate in a run of the system. A possible execution $\langle \alpha_0, \alpha_1, \ldots \rangle$ of A is *valid* if the following property holds: Let s^i be the i-th state reached by the execution and that enables universal oclets $\{o_1, \ldots, o_k\}$ and existential oclets $\{o_{k+1}, \ldots, o_l\}$. Then the next k steps are adaptation steps of $\{o_1, \ldots, o_k\}$ followed by adaptation steps of some (or none) of the $\{o_{k+1}, \ldots, o_l\}$.

As in Life-sequence charts, we call an adaptive system A *live* if for each existential oclet $o \in O_A$ exists a valid execution $\langle \alpha_0, \alpha_1, \ldots \rangle$ of A that has a state s^i where o is realized [5].

Note that the entire system dynamic can furthermore be changed by adding or removing an entire oclet at run-time, to allow or prevent its use in the remainder of the execution. Thereby an oclet works as an isolated aspect of the system that is used or not used.

4. Example revisited: adapting behavior to handle faults

We revisit our example from Sect. 2.2. Let A_{ex} be an adaptive system with oclets $O_{ex} = \langle o_1, \ldots, o_4 \rangle$ from Fig. 3.1 and 3.2 and initial marking $m^0_{ex} = [\![\mathsf{rdy}, \mathsf{av}]\!]$. We will identify subnets of β^2_{ex} of Fig. 4.1 as states of A as we proceed.

Oclets o_1, o_2, o_3 reproduce our example system N_{ex} from Fig. 2.1, while o_4 specifies a failure in the execution: Oclet o_1 is enabled in s^0_{ex} and the adaptation step by o_1 is the only possible step; after adding o_1, rq is enabled. The execution $\langle o_1, \mathsf{rq}, \mathsf{gr}, o_2, \mathsf{rq} \rangle$ describes a granted request to access op, and one pending request. The resulting state is $\langle \beta_5, C_5 \rangle$ where β_5 is the restriction of β^2_{ex} of Fig. 4.1 to the nodes $X_5 = \{b_1, \ldots, b_{11}, e_1, \ldots, e_6\}$, and $C_5 = \{e_1, e_2, e_4\}, m(C_5) = [\![\mathsf{en}, \mathsf{un}, \mathsf{pd}]\!]$.

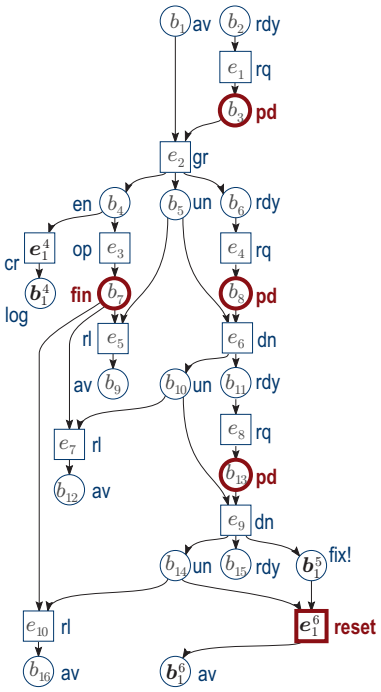

Figure 4.1.: β^2_{ex} depicting possible behavior of A_{ex}

In s_5, o_4 is enabled, because we can embed $\mathsf{en} \mapsto b_4$; o_4 denotes a possible failure of op. So we continue with $\langle o_4, \mathsf{cr}, \mathsf{dn}, o_3, o_2, \mathsf{rq} \rangle$ which results in a state s_{11} that with $X_{\beta_{11}} = \{b_1, \ldots, b_{15}, e_1, \ldots, e_9, e^4_1, b^4_1\}$, $C_{11} = \{e_1, e_2, e_3, e_4, e_6, e_8, e^4_1\}$, $m(C_{11}) = [\![\mathsf{log}, \mathsf{pd}]\!]$. From now on, by just using the oclets $o_1, \ldots, o_4$, the system will never grant a request, because op is not enabled and rl which depends on op cannot put a token back on av.

We therefore extend A by the oclets o_5, o_6, o_7 of Fig. 3.1. Oclets o_5 and o_6 stabilize the system to recover it from this fault: o_5 is enabled in s_{11}. Its precondition con_5 describes that the system has detected that a request has been denied, and that another request is pending. If the next transition should be a further deny dn (dn is passive in o_5), then adapt this occurrence of dn to produce a notification to fix the problem. That is, append condition fix! to dn only in this situation. o_5 is existential, and we choose to adapt $s_{11} \xrightarrow{o_5} s_{12}$. β_{12} extends β_{11} by condition b^5_1; the configuration stays put. Now the step $\beta_{12} \xrightarrow{\mathsf{dn}} \beta_{13}$ yields the marking $m(C_{13}) = [\![\mathsf{log}, \mathsf{un}, \mathsf{rdy}, \mathsf{fix!}]\!]$.

Universal oclet o_3 is enabled in s_{13}; $s_{13} \xrightarrow{o_3} s_{14}$ adds e_{10} and b_{16}. In s_{13} and s_{14}, oclet o_6 is enabled. It specifies that the problem can be fixed by forcefully resetting the state from unavailable to available by a hot transition reset. The step $s_{14} \xrightarrow{o_6} s_{15}$ adds e^6_1 and b^6_1 to s_{14}, and removes e_{10} and all subsequent nodes, i.e. b_{16}, because reset is hot which

requires that e_1^6 is guaranteed to occur once reset is enabled. Now, the step $s_{15} \xrightarrow{\mathsf{reset}} s_{16}$ yields the marking $m(C_{16}) = [\![\mathsf{rdy}, \mathsf{av}, \mathsf{log}]\!]$ which includes the initial state m_{ex}^0, hence the system is back into normal operation, and is now *fault-tolerant* wrt. cr.

Now, observe that the steps $\langle o_5, \mathsf{dn}, o_3 \rangle$ could have been applied in this order from s_5 if cr does not occur. Let s_{17} denote the thereby resulting state with $m(C_{17}) = [\![\mathsf{en}, \mathsf{rdy}, \mathsf{un}, \mathsf{fix!}]\!]$. Now firing reset would be disastrous, because the access to op is legally locked. The anti-oclet o_7 describes that this action is not allowed; o_7 is enabled in s_{17}: Embed $\mathsf{pd} \mapsto b_3$, $\mathsf{gr} \mapsto e_2$, $\mathsf{en} \mapsto b_4$, $\mathsf{un} \mapsto b_{14}$, $\mathsf{fix!} \mapsto b_1^5$, and $\mathsf{reset} \mapsto e_1^6$. This embedding enables o_7 in s_{17}. The step $s_{17} \xrightarrow{o_7} s_{18}$ removes e_1^6 and subsequent nodes, i.e. b_1^6 from s_{17}; $m(C_{18}) = [\![\mathsf{en}, \mathsf{rdy}, \mathsf{un}, \mathsf{fix!}]\!]$. By o_3 the system can continue.

This example also shows that one has to be careful in the design of the oclets: o_6 is enabled in s_{18} and *must* occur according to our specification. This immediately would enable o_7 again, leading to an infinite cycle of adaptations where no transition can occur. The reason is that con_6 is contained in con_7. Detecting such inconsistencies in adaptive systems is important future work for our approach. A possible solution for our example would be to make o_6 existential.

5. Conclusion

We have defined an approach to formally define systems that can adapt their behavior at run-time by a system-controlled adaptation operation. An adaptive system is specified in scenarios with an intuitively understandable Petri net syntax which we formalized as oclets. We contributed an operational adaptation step that synthesizes the behavior of the system from its oclets at run-time. The synthesis depends on the current behavior of the system. The synthesis may add, remove or modify existing behavior by a special adaptation step of the system which effectively leads to an adaptation of behavior at run-time. The formal model for this approach are Petri net branching processes. Thus, our model is operational, and may be analyzed with techniques that work on branching processes. Additionally, the system specification can be extended at run-time to allow a wider range of adaptation. In an example, we showed the feasibility of our approach by making a given system fault-tolerant.

Related Work

The problem of adapting system behavior has been researched from various angles. Many works consider the adaptation of workflows by transformation rules on the structure of the system specification to be applied at runtime [7, 2, 4, 14], or by a notion of relating an old system specification to a new system specification to guarantee a correct replacement [1]. Some of these approaches, e.g. [14], are more expressive than our approach. But the condition when to perform an adaptation step is not part of the operational system specification. Hence the system cannot adapt itself unlike in our approach.

The paradigm of nets in nets, or object nets, allows to change behavior of a Petri net as it is executed by a Petri net transition. In this model, a token of a net, can be a Petri net by itself. This produces a multi-level hierarchy of nets that can synchronize their behavior [12]. In extensions of this model, a transition can perform a change operation on a lower-level net (that is treated as a token to this changing transition). Operations to change a net have been formalized as direct operations on nets [9], or as graph transformations based on rules that allow constructing system behavior from scenarios [6]. In [10] graph transformation rules are used to construct a branching process of a system for fault diagnosis. While in hierarchical nets the modeler has to make adaptation steps explicit to achieve change, a modeler using our approach just has to write down the scenario and anti-scenarios of the system to achieve adaptive behavior. Further, our model performs adaptations in a flat model without hierarchies; the advantage or disadvantage of this fact depends on the concrete application or system

Dynamically adaptive systems can also be expressed with various sorts of process

algebras, for instance χ [11] or the π-calculus [13]. In comparison, our approach features an intuitive graphical notation, and an adaptation operator that works for any kind of denoted scenarios. In process algebras in turn, open systems, dynamic communication, and creation of processes can be expressed as first class citizens together with powerful analysis techniques; our approach presented so far in this paper does not cover these aspects.

The expressive means for our approach were highly influenced by live-sequence charts [5] which are an entirely declarative technique, while we provide an operational model. It is certainly worth to compare the expressiveness with our approach, but this question is exceeds the scope of this paper.

Future Work

We are currently implementing our algorithms in a proof-of-concept run-time environment for adaptive processes. This tool will help us in validating our approach in actual case studies for modeling processes in disaster management. We research the conditions for consistent sets of oclets the avoid infinite cycles of adaptation. We also work on generalizing the notion of well-formedness for oclets to achieve more expressivity in order to cover practically relevant adaptation patterns as identified in [16].

Bibliography

[1] W.M.P. v.d. Aalst and T. Basten. Inheritance of workflows: an approach to tackling problems related to change. *Theor. Comput. Sci.*, 270(1-2):125–203, Feb 2002.

[2] A. Agostini and G. De Michelis. Improving flexibility of workflow management systems. In *Business Process Management, Models, Techniques, and Empirical Studies*, pages 218–234, London, UK, 2000. Springer-Verlag.

[3] João Paulo Barros and Luís Gomes. Net model composition and modification by net operations: a pragmatic approach. In *Proceedings of INDIN'2004*, Berlin, Germany, June 2004.

[4] F. Casati, S. Ceri, B. Pernici, and G. Pozzi. Workflow evolution. In *International Conference on Conceptual Modeling / the Entity Relationship Approach*, pages 438–455, 1996.

[5] W. Damm and D. Harel. LSCs: Breathing life into message sequence charts. *Form. Methods Syst. Des.*, 19(1):45–80, 2001.

[6] H. Ehrig, K. Hoffmann, and J. Padberg. Transformations of Petri Nets. *Electr. Notes Theor. Comput. Sci.*, 148(1):151–172, 2006.

[7] C. Ellis, K. Keddara, and G. Rozenberg. Dynamic change within workflow systems. In *COCS '95: Proceedings of conference on Organizational computing systems*, pages 10–21. ACM Press, 1995.

[8] J. Esparza, S. Römer, and W. Vogler. An Improvement of McMillan's Unfolding Algorithm. In *Tools and Algorithms for Construction and Analysis of Systems*, pages 87–106, 1996.

[9] Berndt Farwer. Recovery and reset in object petri nets with process markings. In *Proceedings of CS&P 2006*, pages 47–57, Sept. 2005.

[10] Stefan Haar, Albert Benveniste, Eric Fabre, and Claude Jard. Fault diagnosis for distributed asynchronous dynamically reconfigured discrete event systems. In *IFAC World Congress Praha*, 2005.

[11] D.A. van Beek J.C.M. Baeten and J.E. Rooda. *CRC Handbook of Dynamic System Modeling*, chapter Chapter 19: Process algebra (for dynamic system modeling), pages 19.1–21. Chapman & Hall, 2007.

[12] Michael Köhler and Heiko Rölke. Reference and value semantics are equivalent for ordinary object Petri nets. In P. Darondeau and G. Ciardo, editors, *Proceedings of the ATPN 2005*, volume 3536, pages 309–328. Springer, June 2005.

[13] Robin Milner. *Communicating and Mobile Systems: the Pi-Calculus*. Cambridge University Press, June 1999.

[14] M. Reichert and P. Dadam. $\text{ADEPT}_{\text{flex}}$-Supporting Dynamic Changes of Workflows Without Losing Control. *J. Intell. Inf. Syst.*, 10(2):93–129, 1998.

[15] W. Reisig. *A Primer in Petri Net Design*. Springer Compass International, 1992.

[16] B. Weber, S. Rinderle, and M. Reichert. Change patterns and change support features in process-aware information systems. pages 574–588. 2007.

A. Formal notations related to Petri nets

A.1. Notations

We canonically lift a function $f : A \to B^{\perp}$ to a set of arguments: $f(S) := \{f(a) \mid a \in S, f(a) \neq \perp\}$ for any $S \subseteq A$. For a function $f : A \to 2^B$ with co-domain 2^B being the *powerset* of B, we define $f(S) := \bigcup\{f(a) \mid a \in S\}$ for any $S \subseteq A$. That is $f(S)$ yields the union.

We write $\mathbb{N}$ for the set of natural numbers. A *multiset* $m \in \mathbb{N}^A$ assigns to each $a \in A$, its number of occurrences $m(a)$ in m; i.e. $m : A \to \mathbb{N}$. For a function $f : A \to B$ and $m \in \mathbb{N}^A$, the resulting multiset $m' := f(m), m' \in \mathbb{N}^B$ preserves the number of occurrences $m(a)$ of a; $f(m) := [\![m(a) \cdot f(a) \mid a \in m]\!]$.

A.2. Petri nets

Let $N = \langle P, T, F, \ell \rangle$ be a labeled Petri net. A *state* of N is described by a marking of N; a *marking* m of N is a function $m : P \to \mathbb{N}$ that assigns each place p a number $m(p)$ of *tokens*. We conceive a marking as a multi-set. A *net system* $\Sigma = \langle N, m^0 \rangle$ is a labeled Petri net N with a marking of N, which we call *initial*.

We denote the *nodes* of a net N by $X := P \uplus T$. We write $pre(x) := \{y \mid (y, x) \in F\}$ for the *pre-set* of x in N, and $post(x) := \{y \mid (x, y) \in F\}$ for its *post-set*. A transition $t \in T$ is *enabled* in a marking m if each pre-place of t has a token, $pre(t) \subseteq m$. If t is enabled, it may *fire*. Firing t removes a token from each pre-place, and produces a token on each post-place of t. This gives the *successor marking* $m' = m - pre(t) + post(t)$ and describes the *step* $m \xrightarrow{t} m'$. From markings and steps, one can construct a *run* of Σ and the *underlying transition system* of Σ in the usual way.

A.3. Branching Processes

Let N be a Petri net, and let $x, y \in X_N$.

- x and y are in *causal* relation $x \leq_N y$ iff there exists a directed path from x to y along the arcs F_N. We write $x <_N y$ if $x \neq y$. Let $X' \subseteq X$. The set $\lfloor X' \rfloor_N := \{x \in X_N \mid \exists x' \in X' : x \leq_N x'\}$ is the set of nodes of N that precede X'; $X' \subseteq X$ is *causally closed* if $\lfloor X' \rfloor_N \subseteq X'$.

- Two different transitions t_1 and t_2 are in *direct conflict* if $pre_N(t_1) \cap pre_N(t_2) \neq \emptyset$. Conflict is inherited along $\leq_N$: Any two nodes x_1, x_2 are in *conflict*, $x \#_N y$, if there

exists a place p and two paths $\langle p, t_1, \ldots, x_1\rangle$ and $\langle p, t_2, \ldots, x_2\rangle$ in N where t_1 and t_2 are in direct conflict. A set $X' \subseteq X_N$ is called *conflict-free* if $\forall x, y \in X' : \neg\, x \#_N y$.

- x and y are *concurrent* if neither $x \leq_N y$ nor $y \leq_N x$ nor $x \#_N y$. A set of concurrent nodes is a *co-set*.

Definition 18 (Occurrence net). An *occurrence net* K is a Petri net where each place p of K has at most one pre-transition ($|pre_K(p)| \leq 1$), K is acyclic ($\leq_K$ is a partial order), K is finitely preceded (each $x \in X_K$ has only finitely many predecessors wrt. $\leq_K$), and no transition t of K is in conflict with itself (self-conflict would be inherited from two different conflicting nodes $x, y \leq_K t$).

If K is conflict-free then K is a *causal net*.

The set of minimal nodes of K is $Min(K) := \{x \in X_K \mid \nexists y \in X_K : y <_K x\}$. ⋆

Definition 19 (Branching process). A *branching process* β of a net system $\Sigma = \langle N, m^0\rangle$ is a labeled occurrence net $\beta = \langle B_\beta, E_\beta, F_\beta, \lambda_\beta\rangle$ (where places B_β are called *conditions* and transition E_β are called *events*) with a labeling $\lambda_\beta : X_\beta \to X_N$ where

1. λ_β is *sort-preservingly* mapping conditions to places, $\lambda_\beta(B_\beta) \subseteq P_N$, and events to transitions, $\lambda_\beta(E_\beta) \subseteq T_N$,
2. $\lambda_\beta(Min(\beta)) = m^0$,
3. λ_β bijectively maps for each event e of β $pre_\beta(e)$ to $pre_N(\lambda_\beta(e))$, and $post_\beta(e)$ to $post_N(\lambda_\beta(e))$ (we call λ_β an *event-local bijection*), and
4. λ_β guarantees that N_β contains no duplicate representations of transitions of N, $\forall e_1, e_2 \in E_\beta : (\lambda_\beta(e_1) = \lambda_\beta(e_2) \wedge pre_\beta(e_1) = pre_\beta(e_2)) \Rightarrow e_1 = e_2$.

If K is a causal net, then β is a *process* of Σ. ⋆

From property 3 follows that λ_β is a net homomorphism $\lambda_\beta : \beta \to N$.

Let β be a branching process. Let $C \subseteq E_\beta$. If C is conflict-free and causally closed, then C is a *configuration* of β.

Let C be a configuration, and let $Y = Min(\beta) \cup pre_\beta(C) \cup C \cup post_\beta(C)$. C induces the process $\beta(C) := \beta[Y]$, i.e. the restriction of β to the nodes Y. The $Cut(C) := (Min(\beta) \cup post_\beta(C)) \setminus pre_\beta(C)$ is a co-set of conditions and represents the reachable marking $m(C) := \lambda_\beta(Cut(C))$ of Σ.

B. Formal notations related to Oclets

B.1. Dynamically constructed branching processes

Definition 20 (Dynamically Constructed Branching Process).
Let $A = \langle O_A, m_A^0 \rangle$, $O_A = \langle o_1, \ldots, o_n \rangle$ be an adaptive system. A *dynamically constructed branching process*, DBP for short, $\beta = \langle B_\beta, E_\beta, F_\beta, \lambda_\beta, \vartheta_\beta, \ell_\beta \rangle$ of A is an occurrence net $\langle B_\beta, E_\beta, F_\beta, \lambda_\beta \rangle$ where the following properties hold:

1. Events (conditions) of β are labeled with sets of transitions (places) of A by a safe-labeling $\lambda : X_\beta \to 2^{X_A}$ of A, $\lambda_\beta = \langle \lambda_{\beta,1}, \ldots, \lambda_{\beta,n} \rangle$, see Sect. 2.3.1. We say that node x is $\lambda_{\beta,i}$*-labeled* if $\lambda_{\beta,i}(x) \neq \perp$.

2. Each event (condition) of β is labeled with equally named transitions (places) of A s.t. the naming function of β is well-defined, $\forall x \in X_\beta, \forall i \in [n] : \lambda_{\beta,i}(x) \neq \perp \Rightarrow \ell_\beta(x) = \ell_{o_i}(\lambda_{\beta,i}(x))$.

3. The minimal conditions represent the initial marking, $[\![\ell_\beta(b) \mid b \in Min(\beta)]\!] = m_A^0$.

4. For each $i \in [n]$, each $\lambda_{\beta,i}$ is for each event e a surjection from the its $\lambda_{\beta,i}$-labeled pre-set (post-set) onto the pre-set (post-set) of transition $t = \lambda_{\beta,i}(e)$ of oclet o_i:

$$\forall e \in E_\beta \forall p \in pre_{o_i}(t) \exists b \in pre_\beta(e) : \lambda_{\beta,i}(b) = p, \tag{B.1}$$

   and correspondingly for post-conditions; called *event-local surjection.*

5. Each event of β has no two equally labeled pre-conditions: $\forall e \in E_\beta \forall b, b' \in pre_\beta(e) : \lambda_\beta(b) = \lambda_\beta(b') \Rightarrow b = b'$; and no two equally labeled post-conditions:

$$\forall e \in E_\beta \forall b, b' \in post_\beta(e) : \lambda_\beta(b) = \lambda_\beta(b') \Rightarrow b = b'; \tag{B.2}$$

   called *event-local injection.*

6. Each condition of β has no two equally labeled post-events.

7. β realizes its temperatures by $\vartheta_\beta : X_\beta \to Temp$.

If β satisfies all properties except 3 we call β a *partial* DBP, shortly P-DBP. $\star$

B.1.1. Dynamically constructed branching processes are sound

Properties 1-3, 6 of Def. 20 follow canonically from Properties 1,2, and 4 for BP in Def. 19. Property 7 is the canonical requirement that follows from using temperature-annotated branching processes, see Sect. 3.2. Properties 4 and 5 soundly liberalize the labeling of BP s.t. any dynamically constructed branching process of A is actually built from the oclets O_A as follows.

For classical BPes of a net system Σ, the event-local bijection between events of $\beta(\Sigma)$ and transitions of Σ according to property 3 of Def. 19 guarantees:

- Each event e represents the firing of a single transition t that consumes a single token from each $p \in pre(t)$ and produces on each $p \in post(t)$ (surjection onto the pre- and postset).
- Furthermore e represents a firing that consumes *exactly one token* from each $p \in pre(t)$ and produces *exactly one token* on each $p \in post(t)$ (injection between the pre- and post-sets).

In a dynamically constructed branching process β, an event e represents the synchronous firing of a set of transitions $\lambda_\beta(e) = \{t_1, \ldots, t_k\}$ of A. A condition b represents a shared token on the places $\lambda_\beta(b) = \{p_1, \ldots, p_l\}$ of A. Thus e consumes shared tokens and produces shared tokens.

- This is sound if each shared token $b \in pre_\beta(e)$ is only consumed from (produced on) a place that actually has (gets) it: $\forall i \in [n] \forall t_i \in (\lambda_\beta(e) \cap T_i) \forall p_i \in pre_{o_i}(t_i) \exists b \in pre_\beta(e) : p_i = \lambda_{\beta,i}(b)$; correspondingly for post-places. The property 4 of Def. 20 satisfies this requirement.
- Each condition $b \in pre_\beta(e)$ is the only representative of a shared token on $\lambda_\beta(b) = \{p_1, \ldots, p_k\}$ that is consumed by e: $\forall b, b' \in pre_\beta(e) : \lambda_\beta(b) = \lambda_\beta(b') \Rightarrow b = b'$; correspondingly for post-places. The property 5 of Def. 20 satisfies this requirement

Our definition allows that an event e may consume two shared tokens $b, b' \in pre_\beta(e)$ both shared by one place $p \in \lambda_\beta(b) \cap \lambda_\beta(b')$. But by property 5 of Def. 20 it is guaranteed that, say, b' is shared by another place $p' \in \lambda_\beta(b') \setminus \lambda_\beta(b)$, which means tokens b and b' are different in A. The event-local surjection by property 4 makes sure that e legally consumes b, b' by requiring $\exists t, t' \in \lambda_\beta(e), i, j \in [n] : p \in pre_i(t) \wedge p' \in pre_j(t')$.

By definition, each 'normal' net system Σ has a maximal dynamically constructed branching process $\beta(\Sigma)$ where the labels are singleton sets of X_{N_Σ}; in this case properties 4 and 5 of Def. 20 are equivalent to property 3 of Def. 19.

B.2. Correctness of adaptation steps

B.2.1. Properties of an enabling embedding

Let A be an adaptive system, let $o \in O_A$ be an oclet, and let $s_1 = \langle \beta_1, C_1 \rangle$ be a state of A.

They key to realize o in s_1 by an adaptation step $s_1 \xrightarrow{o} s_2$ is to find an enabling embedding h^{en} of a prefix $\gamma^{en} \in \Gamma_o$ of β_o in β_1. It must be possible to extend or to reduce h^{en} to an embedding h of β_o in β_2 through extending or reducing β_1 to β_2 s.t. h realizes o in β_2. There are four requirements that additionally have to be satisfied by h^{en} (beyond embedding precondition and requirements) to allow the extension or reduction to h.

The first requirement is rather technical and postulates that if an oclet o has been realized by some embedding h' at a specific location Y' of β_1 already (in an earlier step), and we now try to embed it again around Y', we should use the earlier realizing embedding h' as much as possible. This requirement makes the repeated application of oclets consistent.

Definition 21 (Embedding respects realization). Let $o \in O_A$ be an oclet, let β_1 be a DBP, let $\gamma^{en} \in \Gamma_o$, and let $Y' \subseteq \beta_1$. A mapping $h : \gamma^{en} \to Y'$ *respects realization* if for every event e in γ^{en}, for every (pre-) post-condition b of e in γ^{en}, if $e' = h^{en}(e)$ has a (pre-) post-condition b' with $\lambda_{\beta_o}(b) \subseteq \lambda_{\beta_1}(b')$ then $h^{en}(b) = b'$. ⋆

The second requirement takes care of hot transitions. While β_o and β_1 each realize (T-hot) for their events by definition, extending β_1 by β_o could violate (T-hot) in the result. Let both, $e_1 \in E_{\beta_o}$ and $e_2 \in E_{\beta_1}$ with $\lambda_{\beta_o}(e_1) = t_1$ and $\lambda_{\beta_1}(e_2) = t_2$, be *hot*. If any $b_1 \in pre_{\beta_o}(e_1)$ gets realized by h as $b_2 \in pre_{\beta_1}(e_2)$ then t_1 must be realizable as e_2; otherwise t_1 and t_2 would be realized in direct conflict. But t_1 can be realized as e_2 only if t_1 and t_2 have equal names.

Definition 22 (Temperature-aware embedding). Let $o \in O_A$ be an oclet and let β_1 be a DBP of A. Let $X' \subseteq X_{\beta_o}$ and $Y' \subseteq X_1$. A mapping $h : X' \to Y'$, is *aware of temperatures* if $\forall b \in (X' \cap B_{\beta_o}) \exists e \in post_{\beta_o}(b) : (\vartheta_{\beta_o}(e) = hot \Rightarrow \forall e' \in post_1(h(b)) : \vartheta_1(e') = cold \vee \ell_1(e') = \ell_{\beta_o}(e))$. ⋆

Note that because β_1 is an DBP, the condition $h(b) \in B_1$ has at most one hot post-event $e' \in E_1$ which then must match names with e; symmetrically for β_o.

Definition 23 (Active embedding for normal oclets). Let $o \in O_A$ be a normal oclet, let β_1 be a DBP of A, and let $\gamma^{en} \in \Gamma_o$ be a prefix of o. An embedding $h^{en} : X^{en}_{\gamma} \to Y, Y \subseteq X_1$ is *active* if $\gamma^{en} \neq \beta_o[active_{\beta_o} \cup passive_{\beta_o}]$, or $\exists x \in X_{\beta_o} : \vartheta_{\beta_o}(x) = hot \wedge \vartheta_1(h^{en}(x)) = cold$. ⋆

Definition 24 (Active embedding for anti-oclets). Let $o \in O_A$ be an anti-oclet, let β_1 be a DBP of A, and let $\gamma^{en} \in \Gamma_o$ be a prefix of β_o. An embedding $h^{en} : X^{en}_{\gamma} \to Y, Y \subseteq X_1$ is *active* if $X^{en}_{\gamma} \cap active_{\beta_o} \neq \emptyset$. ⋆

Proposition 1: *Let o be an oclet and let $s_1 = \langle \beta_1, C_1 \rangle$ be such that there is no largest embeddable prefix $\gamma^{en} \in \Gamma_o$ of β_o s.t. o is enabled in s. Then the assumptions of o do not hold, or o is already realized.* ⋆

Lemma 2: *Let o be a normal oclet of A, and let h embed a largest embeddable prefix of $\gamma \in \Gamma_o$ in s_1 that satisfies all requirements except that h is aware of temperatures. Then γ and h correspondingly can be reduced to an enabling embedding.* $\star$

Proof. Firstly, $\gamma = \gamma(X^0)$ for some causally closed set $X^* \subseteq X_o$. Temperature awareness is violated only at conditions $b_1, \ldots, b_k \in B_\gamma$ each having a hot post-event in β_o. Because o is well-formed, each b_i is active; thus $b_i \in X^0$.

Hence there exists a causally closed set $X^1 \subseteq X^0 \setminus \{b_1, \ldots, b_k\}$ in β_o; X^1 might be empty. $\gamma(X^1)$ is an embeddable prefix of β_o by definition of $\gamma(.)$. The restriction h^1 of h to $\gamma(X^1)$ still embeds the precondition and the passive nodes of o.

Embedding h^1 satisfies all requirements for enabling normal oclets, thus o is enabled in s_1:

- Because h respects realization, h^1 respects realization (since h^1 is a restriction of h).
- The precondition consists of passive and abstract nodes only, thus $con_o \subseteq X^1$, satisfying properties (1.) and (2.) of enabled oclets.
- $\gamma(X^0)$ is the largest embeddable prefix of o; it only violates temperatures. Any set X^* with $X^1 \subset X^* \subset X^0$ is either contains a b_i or is not causally closed. Thus $\gamma(X^1)$ is the largest embeddable prefix s.t. h^1 is temperature-aware.
- From $X^1 \subset X^0 \subseteq X_o$ follows that h^1 is active, satifying property (3.) of enabled normal oclets. $\square$

Proof (Proposition 1). Assume the smallest prefix of β_o, $\gamma(\emptyset) = \beta_o[\mathit{passive}_{\beta_o} \cup \mathit{abstr}_{\beta_o}]$, can not be embedded in β. Then the assumption of o that its passive elements are realized in β in the order that is induced by abstr_o is not satisfied. Thus o should not be enabled.

Now, let $\gamma \in \Gamma_o$ be a prefix of β_o that is embedded in β by h^{en} but violates properties (1.) or (2.) of an enabled oclet. If the precondition of o has not been observed or mark_o is not in the current cut, then o should not be enabled.

Assume properties properties (1.) and (2.) of an enabled oclet hold, and o is a normal oclet. Then h is aware of temperatures (by Lemma 2). Then h cannot be active (otherwise o would be enabled) which means $X_\gamma = X_{\beta_o}$ and each node and its embedding have the same temperature. From the definition of embedding and of realizing temperatures of nodes follows that o is already realized in β at $h^{en}(\gamma_o) = h^{en}(X_o)$. Thus o should not be enabled.

Assume properties (1.) and (2.) of an enabled oclet hold, and o is an anti-oclet. Then h is not active which means $X_\gamma = \mathit{passive}_{\beta_o} \cup \mathit{abstr}_{\beta_o}$. But then o is already realized by in β by definition. $\square$

B.2.2. Properties of an adaptation step

Theorem 3 (Adaptation steps yield states). *Let A be an adaptive system, let s_1 be a state of A, let $o \in O_A$ be an oclet enabled in A, and let $s_1 \xrightarrow{o} s_2$ be an adaptation step. Then s_2 is a state of A.* ⋆

Lemma 4: *Let A be an adaptive system, let $\langle \beta_1, C_1 \rangle$ be a state of A, let $o \in O_A$ be an anti-oclet enabled in A, and let $\langle \beta_1, C_1 \rangle \xrightarrow{o} \langle \beta_2, C_2 \rangle$ be an adaptation step. Then β_2 is a* Dbp *of A.* ⋆

Proof. Let $\gamma^{en} \in \Gamma_o$ be the enabling prefix of β_o and let $h^{en} : \gamma^{en} \to Y^{en}, Y^{en} \subseteq X_1$ be the enabling embedding of o in β_1. The net β_2 is the result of the adaptation step defined in Def. 17. We now show that β_2 satisfies the properties of Dbp according to Def. 20.

First of all, β_2 is an occurrence net: The step $s_1 \xrightarrow{o} s_2$ only removed nodes from β_1 and all their causal successors. Thus β_2 is a prefix of β_1 and hence an occurrence net. It follows that embedding $h : \gamma^{en} \to (Y^{en})^{\perp}$ is a restriction of h^{en} ($h \subseteq h^{en}$). By Def. 17, β_2 realizes its temperature annotations.

1. γ^{en} is embedded in β_1 by an enabling embedding h^{en}. From Def. 8.3 and $h \subseteq h^{en}$ follows that λ_2 is a safe labeling: h embeds a node of β_o in a node of β_2 only if their labels are not contradicting wrt. oclet o. Property Def. 20.1 holds.

2. From Def. 8.2 follows that λ_2 labels events and conditions with equally named transitions and places. Property Def. 20.2 holds.

3. By assumption, o is well-formed. Thus the minimal nodes of o are not active. Thus the step cannot remove minimal nodes from β_1 or introduce new minimal nodes. Hence property Def. 20.3 holds.

4. The event-local surjection (Def. 20.4) would be violated if the step removed a condition from a pre-set or post-set of an event without removing the condition. (i) If the step removes a pre-condition of an event, the event gets removed as well (being causal successor). (ii) The step does not remove a post-condition b of an $e \in \beta_1 \cap \beta_2$.

   Property (ii) holds by the semantics of temperatures $\neg\, hot$ and $\neg\, cold$ in Def. 3. Events and conditions of β_o having temperature $\neg\, cold$ are only removed from β_1 if a causal predecessor was removed. A condition having temperature $\neg\, hot$ is not removed, but its causal successors only. Thus no post-condition of an event is removed and property Def. 20.4 holds.

5. The enabling embedding h^{en} embeds a maximally embeddable prefix that respects realization (Def. 21).

   Thus when embedding an event $e \in E_{\beta_o}$ in $h^{en}(e) = e_2 \in E_2$, a pre-condition b of e is embedded as a pre-condition $b_2 \in pre_2(e_2)$ with $\lambda_{\beta_o}(b) \notin \lambda_2(b_2)$ only if there

exists no pre-condition $b_2' \in pre_2(e_2), b_2 \neq b_2'$ that has $\lambda_{\beta_o}(b) \in \lambda_2(b)$. Respectively for post-conditions of e.

Therefore, extending the labeling of λ_1 by the labels of β_o does not break the injection-property for pre- and post-sets of events, and property Def 20.5 holds.

6. The adaptation step only removes events. Since β_2 is an occurrence net, property Def. 20.6 holds.

7. β_2 realizes its temperatures by Def. 17. Hence property Def. 20.7. □

Lemma 5: *Let A be an adaptive system, let $\langle \beta_1, C_1 \rangle$ be a state of A, let $o \in O_A$ be a normal oclet enabled in A, and let $\langle \beta_1, C_1 \rangle \xrightarrow{o} \langle \beta_2, C_2 \rangle$ be an adaptation step. Then β_2 is a* Dbp *of A.* ⋆

Proof. Let $\gamma^{en} \in \Gamma_o$ be the enabling prefix of β_o and let $h^{en} : \gamma^{en} \to Y^{en}, Y^{en} \subseteq X_1$ be the enabling embedding of o in β_1. The net β_2 is the result of the adaptation step defined in Def. 16. We now show that β_2 satisfies the properties of Dbp according to Def. 20.

First of all, β_2 is an occurrence net: By assumption (Def. 6) β_o is conflict-free. Therefore, γ^{en} is embedded in a conflict-free set Y^{en} of β_2 by Def. 8.
If we remove a node, we remove all its causal successors as well; this operation cannot violate the requirements of an occurrence net.
The nodes $Z \subseteq X_{\beta_o} \setminus X_\gamma^{en}$, that are added by the step, together constitute a conflict-free occurrence net that is attached to a co-set of nodes (by o being well-formed, Def. 6).
Again, by the well-formedness of oclets, there exists no arc $(x, y) \in \beta_o$ with $x \in active_o, y \in (passive_o \cup abstr_o)$. Thus the step adds no arc (x', y') to β_2 with $x' \in \beta_2 \setminus \beta_1$, $y \in \beta_1$.
Thus we do not introduce backward conflicts, that is no self-conflicts, in β_2.
Therefore β_2 is an occurrence net.

We now show that β_2 also satisfies the remaining properties of Def. 20. Let $h : X_{\beta_o} \to X_2$ be the realizing embedding of o in β_2 after the adaptation step. Since the step does not removes a node from $Y^{en} = h^{en}(X_\gamma^{en}$, $h^{en} \subseteq h$ holds.

1. γ^{en} is embedded in β_1 by an enabling embedding h^{en}. From Def. 8.3 and $h^{en} \subseteq h$ follows that λ_2 is a safe labeling: h embeds a node of β_o in a node of β_2 only if their labels are not contradicting wrt. oclet o. Thus λ_2 is safe for Y^{en} because h^{en} is an enabling embedding, λ_2 is safe for the added nodes Z by definition, and λ_2 is safe for all other nodes because λ_1 is safe for these nodes. Hence, property Def. 20.1 holds.

2. From Def. 8.2 and the preceding argument follows that λ_2 labels events and conditions with equally named transitions and places. Property Def. 20.2 holds.

3. By assumption, o is well-formed. Thus the minimal nodes of o are not active. Thus the step cannot remove minimal nodes from β_1 or introduce new minimal nodes. Hence property Def. 20.3 holds.

4. If we remove a pre-condition of an event, the event gets removed as well (being a causal successor). By the definition of the adaptation step, we remove a post-condition of an event only, if it is a causal successor of an event that is in conflict with a hot event of β_o. Thus removal of conditions cannot violate the event-local surjection.

   Each event $e \in E_{\beta_o}$ satisfies the event-local surjection because β_o is a P-DBP.

   Let $e \in E_{\beta_o}$ be embedded as an existing event $e_1 \in E_2 \cap E_1$. Then $e \in \gamma^{en}$ and because h^{en} respects realization (Def. 21). Hence the event-local surjection of e holds for the pre-conditions of e. By construction of β_2, each post-condition $b \in post_{\beta_o}(e)$ is embedded as a post-condition $h(b)$ of $h(e)$ with corresponding labels satisfying the event-local surjection of e (b is embedded as an existing condition only if the names are equal, see Def. 8). By the same argument all newly added events $h(e) = e_2 \in E_2 \setminus E_1$ satisfy event-local surjection and property Def 20.4 holds.

5. The enabling embedding h^{en} embeds a maximally embeddable prefix γ^{en}. Each event $e \in E_{\beta_o}$ satisfies the event-local injection because β_o is a P-DBP. Further each event $e_1 \in E_1$ satisfies the event-local injection because β_1 is a DBP.

   The event-local injection could therefore only be violated by adding a new pre-condition (post-condition) $b_2 \in B_2 \setminus B_1$ of an event e_2 to embed a condition $b \in B_{\beta_o}$ while there exists a pre-condition $b_1 \in B_1 \cap B_2$ of e_2 with the same labeling.

   If the step adds a new post-condition $b_2 \in B_2 \setminus B_1$ to an existing event $e_1 \in E_1 \cap E_2$, then b_2 must have a different labeling $\lambda_2(b_2)$ than all other post-conditions e_1, otherwise $b \in B_{\beta_o}, b_2 = h(b) = h^{en}(b)$ would have been embedded as an existing node $b_1 \in PNpostF_1(e_1)$. The step cannot add a new pre-condition b_2 to an existing event e_1 because this would violate that γ^{en} is a causally closed prefix of β_o (see Def. 11). Thus the step preserves the event-local injection and property Def 20.5 holds.

6. By the same argument, the step does not introduce two equally labeled post-events to a given set of conditions of β_2; property Def. 20.6 holds.

7. β_2 realizes its temperatures by Def. 16. Hence property Def. 20.7. □

Proposition 6: *Let $s_1 \xrightarrow{o} s_2$ be an adaptation step of A based on h^{en}. Let $\gamma^{en} \in \Gamma_o$ be the enabling prefix of β_o and let $h^{en} : \gamma^{en} \to Y^{en}, Y^{en} \subseteq X_1$ be the enabling embedding of o in β_1. Then the following properties hold:*

1. *Oclet o is realized in β_2 by the* realizing embedding *h in the set $Y \subseteq X_2$ s.t. if o is a normal oclet then $Y^{en} \subseteq Y$ and h conservatively extends h^{en}, and if o is an anti-oclet then $Y \subseteq Y^{en}$ and h is the restriction of h^{en} to Y.*

2. *Each new node of s_2 was introduced by realizing oclet o: for each new node $x^* \in X_2 \setminus X_1$ exists $x \in X_o$ that is realized in $h(x) = x^* \in Y$.*

3. *Each removed node of s_1 was removed by realizing oclet o: for each removed node $y \in X_1 \setminus X_2$ exists a node $x \in X_o$ that is realized in $h(x) \in Y$ with $h(x) \leq_{\beta_1} y$ in β_1. Removing nodes of s respects causality: if $y \in X_1 \setminus X_2$ then each $y' \in X_1$ with $y \leq_{\beta_1} y'$ is not in X_2.*

4. *The step does not change the past of s_1: $C_1 = C_2$ and $\beta_1(C_1) \setminus Cut_1(C_1) = \beta_2(C_2) \setminus Cut_2(C_2)$.* ⋆

Proof. 1. Let o be a normal oclet. The enabling embedding h^{en} and the embedded set Y^{en} grows by adding the new nodes to β_2, according to Def. 16, resulting in h and $Y = h(active_o \cup passive_o)$. The step only removes nodes y' which are in conflict with hot events of Y (hence $y' \notin Y \cup Y^{en}$). Thus $Y^{en} \subseteq Y$ and $h^{en} = h|_{\gamma^{en}}$. By construction and by Lemma 5, o is realized in β_2.

Let o be an anti-oclet. Because the step only remove nodes, the embedding h and the set Y may only get smaller, see Def. 17. By construction and by Lemma 5, o is realized in β_2.

2. Let o be a normal oclet. By definition of the adaptation step, a new node $x \in X_2 \setminus X_1$ is an embedded node $h^{-1}(x)$ of o.

We do not add nodes if o is an anti-oclet.

3. Let o be a normal oclet. We only remove an event $e_1 \in E_1 \setminus E_2$ (and its successors) from β_1 if it is in direct conflict with a newly added hot event e_2, $e_2 = h(e), e \in E_{\beta_o}$. From the well-formedness of o follows that the there exists an active condition $b \in pre_{\beta_o}(e)$ and the removed event is a post-event of $h(b)$.

Let o be an anti-oclet. By definition of the adaptation step, a node x_1 of β_1 is removed only if there is a $\neg hot$ node x of β_o that was embedded as a $h(x_1)$ with $h(x_1) \leq x$.

4. Observe that by definition of an adaptation step $C_1 = C_2$. It remains to show that $\beta_1(C_1) \setminus Cut_1(C_1) = \beta_2(C_2) \setminus Cut_2(C_2)$.

If o is an existential oclet, then each active nodes of β_o is embedded as a causal successor of $Cut_1(C_1)$ (by o being well-formed, see Def. 6, and the definition of an enabled oclet, see Def. 15). Thus the proposition holds.

Let o be a universal oclet. An active node x of β_o has no passive causal successors because o is well-formed, see Def. 6.

Because the enabling prefix is causally closed holds: If x is embedded as a new node $h(x)$ in β_2 then $h(x)$ has no causal successor in $\beta_1(C_1)$.

Thus a newly added condition is in $Cut_2(C_2) \setminus Cut_1(C_1)$ and a removed condition is in $Cut_1(C_1) \setminus Cut_2(C_2)$. A newly added event is causally after $\beta_1(C_1)$. Thus $\beta_2(C_2) \setminus Cut_2(C_2) \subseteq \beta_1(C_1) \setminus Cut_1(C_1)$.

A normal oclet removes an event $e_1 \in (E_1 \setminus E_2) \cap C_1$ (and its successors) if it is in conflict with a hot event $h(e), e \in E_{\beta_o}$. If $h(e) \in C_1$ then e_1 was not present

in s_1, contradiction. Thus $h(e)$ is a new node in conflict with e_1. But then there exists a configuration $C_1' \subseteq C_1 \setminus \{e_1\}$ and a state $\beta_1(C_1')$ where o was already enabled. Because o is a universal oclet by assumption, an adaptation step with o had to be performed. In the resulting step $h(e)$ was added while e_1 was removed, contradiction to $e_1 \in (E_1 \setminus E_2) \cap C_1$.

By the same argument, a universal oclet removes nodes as early as possible. Because o is well-formed, it cannot happen that o is enabled only after an active event of β_o was added to the configuration. Thus $\beta_1(C_1) \setminus Cut_1(C_1) \subseteq \beta_2(C_2) \setminus Cut_2(C_2)$.

Hence β_2 is a DBP of A that realizes o and satisfies the properties of Prop. 6. □

Proof (Theorem 3). The proof follows from Lemma 4, Lemma 5, and Prop. 6.4. □