Analysis of Self-* and P2P Systems using Refinement (Full Report)
Manamiary Bruno Andriamiarina, Dominique Méry, Neeraj Kumar Singh

To cite this version:

HAL Id: hal-01018162
https://inria.hal.science/hal-01018162
Submitted on 3 Jul 2014

HAL is a multi-disciplinary open access archive for the deposit and dissemination of scientific research documents, whether they are published or not. The documents may come from teaching and research institutions in France or abroad, or from public or private research centers.

L’archive ouverte pluridisciplinaire HAL, est destinée au dépôt et à la diffusion de documents scientifiques de niveau recherche, publiés ou non, émanant des établissements d’enseignement et de recherche français ou étrangers, des laboratoires publics ou privés.
Analysis of Self-\(\ast\) and P2P Systems using Refinement

Manamiary Bruno Andriamiarina\(^1\), Dominique Mery\(^1\), and Neeraj Kumar Singh\(^2\)

Université de Lorraine, LORIA, BP 239, 54506 Vandœuvre-lès-Nancy, France
{Manamiary.Andriamiarina, Dominique.Mery}@loria.fr
McMaster Centre for Software Certification, McMaster University, Hamilton, Ontario, Canada
singhn10@mcmaster.ca, Neerajkumar.Singh@loria.fr

Abstract. Distributed systems and applications require efficient and effective techniques (e.g. self-(re)configuration, self-healing, etc.) for ensuring safety, security and more generally dependability properties, as well as convergence. The complexity of these systems is increased by features like dynamic (changing) topology, interconnection of heterogeneous components or failures detection. This paper presents a methodology for verifying protocols and satisfying safety and convergence requirements of the distributed self-\(\ast\) systems. The self-\(\ast\) systems are based on the idea of managing complex infrastructures, software, and distributed systems, with or without minimal user interactions. Correct-by-construction and service-as-event paradigms are used for formalizing the system requirements, where the formalization process is based on incremental refinement in EVENT B. Moreover, this paper describes a fully mechanized proof of correctness of the self-\(\ast\) systems along with an interesting case study related to the P2P-based self-healing protocol.

Keywords: Distributed systems, self-\(\ast\), self-healing, self-stabilization, P2P, EVENT B, liveness, service-as-event

1 Introduction

Nowadays, our daily lives are affected by various advanced technologies including computers, chips, and smart-phones. These technologies are integrated into distributed systems with different types of complexities like mobility, heterogeneity, security, fault-tolerance, and dependability. Distributed systems are largely used in many applications and provide required functionalities from the interactions between a large collection of possibly heterogeneous and mobile components (nodes and/or agents). Within the domain of distributed computing, there is an increasing interest in the self-stabilizing systems, which are able to autonomically recover from occurring the faults [7]. The autonomous property of the self-\(\ast\) systems tends to take a growing importance in the analysis and development of distributed systems. It is an imperative that we need to get a better understanding of the self-\(\ast\) systems (emergent behaviours, interactions between agents, etc.), if we want to reason about their security, correctness and trustworthiness.

Fortunately, the formal methods community has been analysing a similar class of systems for years, namely distributed algorithms.

In this study, we use the *correct by construction* approach [12] for modelling the distributed self-* systems. Moreover, we also emphasize the use of the *service-as-event* [3] paradigm, that identifies the phases of self-stabilization mechanism, which can be simplify into more stable and simple coordinated steps.

We consider that a given system \( S \) (see in Fig.1) is characterized by a set of events (procedures modelling either phases or basic actions according to an abstraction level) that modifies the state of the system. Legal states (correct states) satisfying a safety property \( P \) are defined by a subset \( \mathcal{CL} \) of possible events of the system \( S \). The events of \( \mathcal{CL} \) represent the possible big or small computation steps of the system \( S \) and introduce the notion of *closure* [5], where any computation starting from a legal state satisfying the property \( P \) leads to another legal state that also satisfies the property \( P \). The occurrence of a fault \( f \) leads the system \( S \) into an illegal state (incorrect state), which violates the property \( P \). The fault \( f \) is defined as an event \( f \) that belongs to a subset \( \mathcal{F} \) of events. When considering the hypothesis of having a self-* system, we assume that there are procedures (protocols or actions) which implement the identification of current illegal states and recovery for legal states. There is a subset \( \mathcal{ST} \) of events modelling recovery phases for demonstrating the stabilization process. The system recovers using a finite number of stabilization steps \( r \). The process is modelled as an event \( r \) of \( \mathcal{CV} \) (\( \subseteq \mathcal{ST} \)) eventually leading to the legal states (convergence property) from recovery states. During the recovery phase, a fault may occur (see dotted transitions in Fig.1).

The system \( S \) can be represented by a set of events \( \mathcal{M} = \mathcal{CL} \cup \mathcal{ST} \cup \mathcal{F} \), where the model \( \mathcal{M} \) contains a set \( \mathcal{CL} \) of events for representing the computation steps of the system \( S \). When a fault occurs, a set \( \mathcal{ST} \) of events simulates the stabilization process that is performed by \( S \). The formal representation expresses a closed model but we do not know what is the complete set of events modelling faults/failures. We characterise the fault model in a very abstract way and it may be possible to develop the fault model according to the assumptions on the environment, but we do not consider this in the current study. We restrict our study by making explicit the events of \( \mathcal{ST} \) modelling the stabilization of the system from illegal/failed states. We ensure that the convergence is always possible: a subset \( \mathcal{CV} \) of \( \mathcal{ST} \) eventually leads \( S \) into the legal states satisfying the invariant \( P \) of the system. Whenever the system \( S \) is in a legal state, we consider that the events of \( \mathcal{ST} \) are either not operative or simply preserve the invariant \( P \) of the system.

In the previous paragraph, we name procedures (protocols or actions) by the term events. An event is modelling a process which is defined by its pre and post specifications or a state transformation belonging to a larger process. It means that we need to play with abstraction levels to develop a self-* system. For instance, one can state that an event called stabilise is ensuring the functionality of getting a stable system (the what) without giving details of the detailed process itself (the how). Hence, the notion of
event is identified to an abstraction level and can be either modelling a global process (the *what*) or a local update of a variable (the *how*). We formalise the system $S$ using the EVENT B modelling language [1], dealing with *events* and *invariant* properties including convergence properties by using a temporal framework. The service-as-event paradigm [3] helps to express this concretisation process: the procedures (1) leading from the illegal states to the recovery states, and (2) leading from the recovery states to the legal states are stated by (abstract) events, during the first stages of the EVENT B development. The next step is to unfold each (abstract procedure) event, by refinement, to a set of coordinated and concrete events, which form the body of the procedure.

This paper is organised as follows. Section 2 presents related works. Section 3 introduces the EVENT B modelling framework including service-as-event paradigm and a formal definition of self-$\star$ systems. Section 4 presents the formal verification approach and illustrates the proposed methodology with the study of the self-healing P2P-based protocol [14]. Section 5 discusses on approaches for studying temporal properties for EVENT B models. Finally, Section 6 concludes the paper along with future work.

2 Related Works on Formal Modelling for Self-$\star$ Systems

Systems usually run in intricate environments, with frequent and unexpected changes. This feature increases interest towards autonomous and self-$\star$ architectures, as they are able to adapt themselves according to the changes that may occur in the systems (faults, etc.) or in the environment. Applying formal methods to self-$\star$ systems originates from the needs of understanding how these systems behave and how they meet their specifications. A self-$\star$ system relies on emergent behaviours, resulting from interactions between components of the system [21].

Traditionally, the correctness of self-$\star$ and autonomous systems is validated through the simulation and testing [20, 22]. However, simulation and testing are not sufficient to cover the whole set of possible states of a system [2]. Therefore, formal methods appear as a promising land for validating self-$\star$ systems, as long as formal techniques can assert the correctness of these systems and certify target properties, like trustworthiness, security, efficiency, etc. under the rigorous mathematical reasoning [6, 8, 24].

Smith et al. [21] have applied the stepwise refinement using Z to study a case of self-reconfiguration, where a set of autonomous robotic agents is able to assemble and to reach a global shape. They do not validate models using an adequate tool (e.g. proof checker, proof assistant, etc.) and models are not localized. Calinescu et al. [6] have used Alloy to demonstrate the correctness of the autonomic computing policies (ACP). However, Alloy does not provide a mechanism for expressing the correct-by-construction paradigm. Méry et al. [2] have also investigated a self-reconfiguring system (Network-on-Chip: adaptative XY routing) using the EVENT B framework and the correct-by-construction approach.

State exploration approaches such as model-checking are also used to study self-$\star$ systems. Model-checkers like SPIN, PRISM, SMV, UPPAAL are used for properties specification and getting evidences that properties, such as flexibility, robustness of the self-$\star$ systems hold [6, 8, 10, 24]. Moreover, these tools allow users to obtain the metrics...
for the self-* systems, such as performance, and quantitative evaluations [6, 8, 10, 24]. Model-checking and state-space evaluation can be used during the conception of self-* systems, but they are especially used for runtime verification [10, 24]. The limit of model checking is clearly the size of models.

Other formal techniques like static analysis and design by contract are also applied for the formal specification of self-* systems [23]. These techniques are mainly used for runtime verification. Graphical approaches, such as Petri Nets, are used to model the temporal aspects and communication flows between different components of a self-* system, and helped to study the cases like self-reconfiguration (replacement of a component, removal of a link between two components, etc.) [24].

Finally, graphical notations (e.g. UML) help to represent self-* systems with understandable figures [25]. Their general purpose is to provide users an insight of a self-* system by describing its architecture, the relationships between agents of the system (OperA methodology [17], ADELFE [20]) or by presenting the system as a composition of extendable/instantiable primitives (FORMS [25]). These notations are generally graphical front-ends for the more complex representations of self-* systems, where the source code [20], and formal models [25] can be generated from the notations.

Our proposed methodology integrates the EVENT B method and elements of temporal logics. Using the refinement technique, we gradually build models of self-* systems in the EVENT B framework. Moreover, we use the service-as-event paradigm to describe the stabilization and convergence from illegal states to legal ones. Self-* systems require the expression of traces properties like liveness properties and we borrow a minimal set of inference rules for deriving liveness properties. The concept of refinement diagrams intends to capture the intuition of the designer for deriving progressively the target self-* system. The RODIN platform provides a laboratory for checking, animating and validating the formal models.

3 Modelling Framework

3.1 EVENT B

We advocate the use of correct-by-construction paradigm for modelling the self-* systems. The key concept is the incremental refinement (simulation) which provides link between discrete models by preserving properties. The EVENT B modelling language designed by Abrial [1] is based on set theory and the refinement of models: an abstract model expressing the requirements of a given system can be verified and validated easily; a concrete model corresponding to the actual system is constructed progressively by refining the abstraction. EVENT B is supported by a complete toset RODIN [19] providing features like refinement, proof obligations generation, proof assistants and model-checking.

Modelling Actions over States The EVENT B modelling language can express safety properties, which are either invariants or theorems in a model corresponding to the system. Two main structures are available in EVENT B:

1. Contexts express static
informations about the model (for instance, graph properties as connectivity); (2) Machines express dynamic informations about the model, safety properties, and events. An EVENT B model is defined by a context and a machine. A machine organises events (or actions) modifying state variables and uses static informations defined in a context. An EVENT B model is characterised by a (finite) list $x$ of state variables possibly modified by a (finite) list of events. An invariant $I(x)$ states properties that must always be satisfied by the variables $x$ and maintained by the activation of the events. The general form of an event $e$ is as follows: ANY $t$ WHERE $G(t, x)$ THEN $x : |P(t, x, x')$ END and corresponds to the transformation of the state of the variable $x$, which is described by a before-after predicate $BA(e)(x, x')$: the predicate is semantically equivalent to $\exists t \cdot G(t, x) \land P(t, x, x')$ and expresses the relationship linking the values of the state variables before ($x$) and just after ($x'$) the execution of the event $e$. Proof obligations are produced by RODIN, from events: INV1 and INV2 state that an invariant condition $I(x)$ is preserved; their general form follows immediately from the definition of the before-after predicate $BA(e)(x, x')$ of each event $e$; FIS expresses the feasibility of an event $e$, with respect to the invariant $I$. By proving feasibility, we achieve that $BA(e)(x, z)$ provides a next state whenever the guard $grd(e)(x)$ holds: the guard is the enabling condition of the event.

<table>
<thead>
<tr>
<th>INV1</th>
<th>INV2</th>
<th>FIS</th>
</tr>
</thead>
<tbody>
<tr>
<td>$\text{Init}(x) \Rightarrow I(x) \land BA(e)(x, x') \Rightarrow I(x') \land \text{BA}(e)(x)$</td>
<td>$\exists z : BA(e)(x, z)$</td>
<td></td>
</tr>
</tbody>
</table>

**Model Refinement** The refinement of models extends the structures described previously, and relates an abstract model and a concrete model. This feature allows us to develop EVENT B models of the self-* approach gradually and validate each decision step using the proof tool. The refinement relationship is expressed as follows: a model $AM$ is refined by a model $CM$, when $CM$ simulates $AM$ (i.e. when a concrete event $ce$ occurs in $CM$, there must be a corresponding enabling abstract event $ae$ in $AM$). The final concrete model is closer to the behaviour of a real system that observes events using real source code. The relationships between contexts, machines and events are illustrated by the following diagrams (Fig. 2), which consider refinements of events and machines.

![Fig. 2: Machines and Contexts relationships](image)

The refinement of a formal model allows us to enrich the model via a step-by-step approach and is the foundation of our correct-by-construction approach [12]. Refinement provides a way to strengthen invariants and to add details to a model. It is also used to transform an abstract model to a more concrete version by modifying the state description. This is done by extending the list of state variables (possibly suppressing some of them), by refining each abstract event to a set of possible concrete versions, and by adding new events. We suppose (see Fig. 2) that an abstract model $AM$ with variables $x$ and an invariant $I(x)$ is refined by a concrete model $CM$ with variables $y$. The abstract state variables, $x$, and the concrete ones, $y$, are linked together by means of a, so-called, gluing invariant $J(x, y)$. Event $ae$ is in abstract model $AM$ and event $ce$ is in
concrete model \(CM\). Event \(ce\) refines event \(ae\). \(BA(ce)(y, y')\) are predicates of events \(ae\) and \(ce\) respectively; we have to discharge the following proof obligation:

\[
I(x) \land J(x, y) \land BA(ce)(y, y') \Rightarrow \exists x' \cdot (BA(ce)(y, y') \land J(x', y'))
\]

Due to limitations on the number of pages, we have briefly introduced the EVENT-B modelling language and the structures proposed for organising the formal development. However, more details are available in [1] and on the Internet\(^1\). In fact, the refinement-based development of EVENT-B requires a very careful derivation process, integrating possible tough interactive proofs. For assisting the development of the self-* systems, we use the service description and decomposition that is provided by the service-as-event paradigm (derived from the call-as-event approach [15]).

### 3.2 The Service-as-Event Paradigm

This section introduces the refinement diagrams [3, 15] and presents the service-as-event paradigm. A brief overview on the usage of these formalisms for modelling the self-* systems is given.

**Objectives** The service-as-event paradigm [3, 15] is a semantical extension of EVENT-B and introduces a way to deal with liveness properties and traces, for modelling the self-* systems.

**A Definition of Self-* Mechanism** We characterize a self-stabilizing system \(S\) (more generally a self-* system) by its ability to recover autonomously from an illegal (faulty) state (violating the invariant \(P\) of the system) to a legal (correct) state satisfying the invariant property \(P\) of system \(S\). Temporal logic [3, 11, 15, 18] can be used to describe such mechanism, using the liveness properties: we represent the stabilization (especially the convergence) property as a service where a system \(S\), in an illegal state (characterized by \(\neg P\)), reaches eventually a legal state (satisfying \(P\)). This service is expressed, with the leads to (\(\Rightarrow\)) operator, as follows: \((\neg P) \Leftrightarrow P\). This leads to property (equivalently \((\neg P) \Rightarrow \diamond P\)) states that every illegal state (satisfying \(\neg P\)) will eventually (at some point in the future) lead to a legal state (satisfying \(P\)).

We define a temporal framework for the EVENT-B model \(M\) of the studied system \(S\) by the following TLA specification: \(\text{Spec}(M): \text{Init}(y) \land \Box[\text{Next}, y] \land L\), where \(\text{Init}(y)\) is the predicate specifying initial states; \(\text{Next} \equiv \exists e \in E.BA(e)(y, y')\) is an action formula representing the next-state relation; and \(L\) is a conjunction of formulas \(WF_y(e)\) : we express a weak fairness assumption over each event \(e\) modelling a step of the recovery process (we do not add any fairness on events leading to illegal states (faults)).

---

\(^1\)http://lfm.itl.kit.edu/download/EventB_Summary.pdf
Refinement Diagrams  We express the self-\* mechanism using Event B, together with liveness properties under fairness assumptions. Refinement diagrams (see in Fig.3), introduced by Méry et al in [3, 15], allow to develop Event B models and add control inside these models. They are also used for stating (proofs of) liveness properties (under fairness assumptions), and for supporting refinement. Therefore, these diagrams are suitable for representing the models of self-\* systems. A refinement diagram $D \equiv PD(M)$ for a machine $M$ is defined as follows: $PD(M) = (A, M, G, E)$, where $A$ is a set of assertions, $G$ a set of assertions for $M$ called conditions/guards of the form $g(x)$, $E$ is the set of events of $M$. The diagram $PD(M)$ is a labelled directed graph over $A$, with labels from $G$ or $E$, satisfying the following rules:

1. If an assertion $R$ is related to another assertion $S$, by an unique non-dotted arrow labelled $e \in E$ (where $e$ does not model a fault), then the property $R \rightsquigarrow S$ is satisfied; (2) if $R$ is related to $S_1, \ldots, S_p$, then each arrow from $R$ to $S_i$ is labelled by a guard $g_i \in G$.

The diagram $D$ possesses proved properties:

1. If $M$ satisfies $P \rightsquigarrow Q$ and $Q \rightsquigarrow R$, it satisfies $P \rightsquigarrow R$.
2. If $M$ satisfies $P \rightsquigarrow Q$ and $R \rightsquigarrow Q$, it satisfies $(P \lor R) \rightsquigarrow Q$.
3. If $I$ is invariant for $M$ and if $M$ satisfies $P \land I \rightsquigarrow Q$, then $M$ satisfies $P \rightsquigarrow Q$.
4. If $I$ is invariant for $M$ and if $M$ satisfies $P \land I \Rightarrow Q$, then $M$ satisfies $P \rightsquigarrow Q$.
5. If $P \rightsquigarrow Q$ is a link of $D$ for the machine $M$, then $M$ satisfies $P \rightsquigarrow Q$.
6. If $P$ and $Q$ are two nodes of $D$ such that there is a path in $D$ from $P$ to $Q$ and any path from $P$ can be extended in a path containing $Q$, then $M$ satisfies $P \rightsquigarrow Q$.
7. If $I, U, V, P, Q$ are assertions such that $I$ is the invariant of $M$; $P \land I \Rightarrow U$; $V \Rightarrow Q$; and there is a path from $U$ to $V$ and each path from $U$ leads to $V$; then $M$ satisfies $P \rightsquigarrow Q$.

These refinement diagrams are attached to Event B models and are used for deriving liveness properties. As an example, the diagram in Fig.3 represents a model of a self-stabilizing system: the diagram relates a pair of assertions $(\neg P, P)$, where $\neg P$ is a precondition stating that the studied system is in an illegal state ($P$ does not hold); and $P$ is the post-condition, describing the desired legal state. We observe that the leads to property $(\neg P) \rightsquigarrow P$, demonstrating the stabilization and convergence, is satisfied by the diagram and the model linked to it.

Applying the Service-as-Event Paradigm [3] We apply the service-as-event paradigm, for formalizing the self-\* systems.

1. Describing stabilization and convergence as a service. We express the stabilization and convergence properties of a self-\* system $S$, where service is stated by the following property: $(\neg P) \rightsquigarrow P$. An abstract event $(e)$ is used for describing the service/procedure represented by $(\neg P) \rightsquigarrow P$: $(\neg P) \overset{\epsilon}{\rightarrow} P$; where $(\neg P)$ is a pre-condition for triggering event $(e)$; and $P$ is a post-condition defined by the actions of event $(e)$, which should be satisfied by the "execution" of event.
2. **Decomposing stabilization and convergence into simple steps.** We decompose the abstract service stated by \((\neg P) \leadsto P\) into simple sub-procedures/steps, using the inference rules [11] related to the leads to properties:

\[
\begin{array}{c}
\vdots \vdots \vdots \text{trans}_3 \\
\text{trans}_4 \\
\text{trans}_5 \\
\hline
\neg P \leadsto R_0 \leadsto R_1 \leadsto P \\
\end{array}
\]

This process is similar to refinement (see Fig. 5), since we add, at each level of the proof tree, a new state \(R_k\) \((0 \leq k \leq n)\) leading from \((\neg P)\) to \(P\). The initial property \((\neg P) \leadsto P\) is decomposed, until the identification of the stabilization steps is satisfactory. The stabilization phase is expressed by the property \((\neg P) \leadsto R_0 \land \ldots \land R_{n-1} \leadsto R_n \leadsto P\), which states the convergence leading to the desired legal states. Each level of the proof tree corresponds to a level of refinement (see Fig. 5) in the formal development. Each leads to property demonstrates a service of stabilization, which is defined by an event in the model.

4 **Stepwise Design of the Self-Healing Approach**

4.1 **Introduction to the Self-Healing P2P-Based Approach**

The development of self-healing P2P-based approach is proposed by Marquezan et al. [14], where system reliability is the main concern. The self-healing process ensures the maintenance of proper functioning of the system services. If a service fails then it switches from a legal state to a faulty state. The self-healing/recovery procedure ensures that the service switches back to the legal state. The services run in a distributed (P2P) system composed of agents/peers executing instances of tasks. The services and peers notions are introduced as: (1) **Management Services**: Tasks/Services are executed by the peers; (2) **Instances of Management Services**: Peers executing a certain type of management service; (3) **Management Peer Group (MPG)**: Instances of the same management service. The self-healing property can be described as follows: (1) **Self-identification** triggers to detect the failure of service. This mechanism identifies running or failed instances of a management service. (2) **Self-activation** is started, whenever a management service will be detected fail by the self-identification. Self-activation evaluates if the management service needs a recovery, based on the criticality of the failure: if there are still enough instances for running the service, the recovery procedure is not started; otherwise, the self-configuration mechanism is triggered for repairing the service. (3) **Self-configuration** is activated if the failure of service is critical: the role of this mechanism is to instantiate the failed management service, and to return the service into a legal state.
4.2 The Formal Design

Figure 6 depicts the formal design of self-healing P2P-based approach. The model \(M_0\) abstracts the self-healing approach. The refinements \(M_1, M_2, M_3\) introduce step-by-step the self-detection, self-activation and self-configuration phases, respectively. The remaining refinements, from \(M_4\) to \(M_{20}\), are used for localisation of the system: each step of the algorithm is made local to a node. The last refinement \(M_{21}\) presents a local model that describes a set of procedures for recovering process of P2P system.

**Abstracting the Self-Healing Approach (M0)** This section presents an abstraction of the self-healing procedure for a failed service. Each service \(s\) is described by two states: \(R\) (legal/running state) and \(FAIL\) (illegal/faulty state). A variable \(\text{serviceState}\) is defined as \(s \mapsto st \in \text{serviceState}\), where \(s\) denotes a service and \(st\) denotes a possible state. A property \(P\) expresses that a service \((s)\) is in a legal running state that is formalised as \(P \equiv (s \mapsto \text{RUN} \in \text{serviceState})\). An event \(\text{FAILURE}\) models a faulty behaviour, where service \((s)\) enters into a faulty state \((\text{FAIL})\), satisfying \(\neg P\). The self-healing of management service \((s)\) is expressed as \((\neg P) \Rightarrow P\). The recovery procedure is stated by an event \(\text{HEAL}\) \((\neg P) \Rightarrow (P \cup \{s \mapsto \text{RUN}\})\), where service \((s)\) recovers from an illegal faulty state \((\text{FAIL})\) to a legal running state \((\text{RUN})\). The refinement diagram\(^1\) (see Fig.7) and events sum up the abstraction of a recovery procedure.

**Introducing the Self-Detection (M1)** The variable \(\text{serviceState}\) is replaced, by refinement, with a new variable \(\text{serviceState}_1\), since new states are introduced. The states \(\text{RUN}, \text{FAIL}\) are refined into \(\text{RUN}_1, \text{FAIL}_1\), and a new state \((\text{FL}_\text{DT}_1)\) is defined. A service \((s)\) can suspect and identify a failure state \((\text{FAIL}_1)\) before triggering the recovery (HEAL). We introduce a property \(R_0 \equiv (s \mapsto \text{FL}_\text{DT}_1 \in \text{serviceState}_1)\) and a new event \(\text{FAIL}_\text{DT}\) DETECT in this self-detection mechanism. Let \(P\) and \(\neg P\) be redefined as follows: \(P \equiv (s \mapsto \text{RUN}_1 \in \text{serviceState}_1)\) and \(\neg P \equiv (s \mapsto \text{FAIL}_1 \in \text{serviceState}_1)\).

---

\(^1\) The assertions \((s \mapsto st \in \text{serviceState})\), describing the state \((st)\) of a service \((s)\), are shorten into \((st)\), in the nodes of the refinement diagrams, for practical purposes.

\(^2\) ⊕: to add elements to a model, ⊖: to remove elements from a model
The intermediate steps of self-detection are introduced according to the refinement diagram (see Fig. 8) and proof tree.

\[
\begin{align*}
\text{FAILURE} & \quad \xrightarrow{\text{trans}} \quad \text{FAIL DETECT} \\
\text{HEAL} & \quad \xrightarrow{} \quad \text{RUN} \\
\text{FAIL DETECT} & \quad \xrightarrow{} \quad \text{FL DT}_1 \\
\text{FAIL}_1 & \quad \xrightarrow{} \quad \text{R}_0 \\
\text{R}_0 & \quad \xrightarrow{} \quad \text{P} \\
\end{align*}
\]

The event \(\text{FAIL DETECT}\) is introduced to express the self-detection: the failure state \((\text{FAIL}_1)\) of a service \((s)\) is detected (state \(\text{FL DT}_1\)).

The property \((-\text{P}) \xrightarrow{} R_0 \xrightarrow{} \text{P}\) is expressed by the event \(\text{FAIL DETECT}\), where the failure \((\text{FAIL}_1)\) of service \((s)\) is identified \((\text{FL DT}_1)\). \(R_0 \xrightarrow{} \text{P}\) is defined by the event \(\text{HEAL}\), where the service \((s)\) is restored to a legal running state \((\text{RUN}_1)\) after failure detection. The same method is applied to identify all the phases of self-healing algorithm. Due to limited space, we focus on the interesting parts of models and liveness properties. The complete formal development of models can be downloaded from web.

Introducing the Self-Activation (M2) and Self-Configuration (M3) The self-activation is introduced in this refinement M2 (see Fig. 9), where a failure of a service \((s)\) is evaluated in terms of critical or non-critical using a new state \(\text{FL ACT}_2\) and an event \(\text{FAIL ACTIV}\). The self-configuration step is introduced in the next refinement M3 (see Fig. 10), which expresses that if the failure of service \((s)\) is critical, then the self-configuration procedure for a service \((s)\) will be triggered (state \(\text{FL CONF}_3\)), otherwise, the failure will be ignored (state \(\text{FL IGN}_3\)).

Introducing the Self-Activation (M2) and Self-Configuration (M3) The self-activation is introduced in this refinement M2 (see Fig. 9), where a failure of a service \((s)\) is evaluated in terms of critical or non-critical using a new state \(\text{FL ACT}_2\) and an event \(\text{FAIL ACTIV}\). The self-configuration step is introduced in the next refinement M3 (see Fig. 10), which expresses that if the failure of service \((s)\) is critical, then the self-configuration procedure for a service \((s)\) will be triggered (state \(\text{FL CONF}_3\)), otherwise, the failure will be ignored (state \(\text{FL IGN}_3\)).

The Global Behaviour (M4) The developed models are refined and decomposed into several steps (see Fig. 11) [14]. These steps are: (1) Self-Detection, (2) Self-Activation, and (3) Self-Configuration. Self-Detection phase is used to detect any failure in the autonomous system using two events \(\text{FAIL DETECT}\) and \(\text{IS OK}\). The event \(\text{FAIL DETECT}\) models the failure detection; and the event \(\text{IS OK}\) states that if a detected failure of a service \((s)\) is a false alarm, then the service \((s)\) returns to a legal state \((\text{RUN}_4)\). Self-Activation process is used to evaluate when actual failures are identified, using

\[ \text{http://ebzall.loria.fr/html_files/files/selfhealing/self-healing.zip} \]
the following events: FAIL_ACTIV, FAIL_IGN, IGNORE, and FAIL_CONF. The events FAIL_IGN and IGNORE are used to ignore the failure of service (s) when failure is not in critical state (FL_IGN_4). The event FAIL_CONF is used to evaluate the failure of service (s) when failure is critical (FL_CONF_4). The last phase **Self-Configuration** presents the healing procedure of a failed service using an event REDEPLOY.

From model M5 to M20, we localise the events (we switch from a service point of view to the instances/peers point of view) and detail the macro (global) steps by adding new events, variables, and constraints. The refinements M5, M6, M7 introduce the running (run_peers(s)), faulty (fail_peers(s)), suspicious (susp_peers(s)) and deployed peers/instances (dep_inst(s)) for a service (s). A function (min_inst) associates each service (s) with the minimal number of instances that is required for running service (s), and helps to detail the self-activation phase: if the number of running instances of service (s) is below than minimum, then the failure is critical. The models M8, M9, M10 detail the self-detection and self-configuration phases to introduce the token owners for the services. Models from M11 to M20 localise gradually the events (to switch from a service point of view to the instances/peers point of view). The detailed formal development of various steps from M5 to M20 are given in the archive. Due to limited space, in the following section, we present only the local model M21.

**The Local Model (M21)** This model details locally the self-healing procedure of a service (s). The peers instantiating management service (s) are introduced, as well as the notion of token owner. The token owner is a peer instance of service (s) that is marked as a token owner for the Management Peer Group (MPG). It can perform the self-healing procedure using self-detection, self-activation, and self-configuration steps.

1. **Self-Detection** introduces an event SUSPECT_INST that states that the token owner for service (s) is able to suspect a set (susp) of unavailable peers instances of service (s). Other events RECONTACT_INST_OK and RECONTACT_INST_KO are used to specify the successful recontact, and failed recontact, respectively, of the unavailable instances for ensuring the failed states. Moreover, the token owner is able to monitor the status of service (s) using two events FAIL_DETECT, and IS_OK. If there are unavailable instances after the recontacting procedure, the token owner informs the safe members of MPG of failed instances using the event FAIL_DETECT, otherwise, the token owner indicates that service is running properly.

2. **Self-Activation** introduces an event FAIL_ACTIV that states that if there are failed instances of service (s), then the token owner evaluates if the failure is critical. Another event FAIL_IGNORE specifies that the failure is not critical. An event IGNORE can ignore the failure if several instances (more than minimum) are running correctly. If the number of instances for the running service (s) will be less than the minimum required services, then the failure will be declared critical, and the self-healing process will be triggered using an event FAIL_CONFIGURE.

3. **Self-Configuration** presents the healing procedure of a failed service using an event REDEPLOY.
**Configuration** introduces three events `REDEPLOY_INSTC`, `REDEPLOY_INSTS` and `REDEPLOY` that specify that if the failure of service \((s)\) is critical, then new instances of running service \((s)\) can be deployed until to reach the minimal number of instances, and after the event `HEAL` can be triggered corresponding to the convergence of the self-healing process.

It is noticeable that the architectural decomposition of the self-healing process is emphasized in this model, by the events related to the algorithm. There is also a set of events describing actions related to the environment. `MAKE_PEER_UNAVAIL`: a set of peers \((prs)\) becomes unavailable (can not be contacted); `MAKE_PEER_AVAL`: a formerly unavailable instance \((p)\) becomes available; `UNFAIL_PEER`: a failed instance re-enters a legal running state.

This model M21 describes locally the Self-Healing P2P-Based Approach, where we have formulated hypotheses for ensuring the correct functioning of the self-healing process: (1) Event `MAKE_PEER_AVAL`: If the token owner of a service \((s)\) becomes unavailable, at least one peer, with the same characteristics as the disabled token owner (state, local informations about running, failed peers, etc.) can become the new token owner of service \((s)\); (2) Event `REDEPLOY_INSTC`: There is always a sufficient number of available peers that can be deployed to reach the legal running state of a service \((s)\).

In a nutshell, we say that our methodology allows users to understand the self-⋆ mechanisms and to gain insight into their architectures (components, coordination, etc.); and gives evidences of the correctness of self-⋆ systems under some assumptions/hypotheses.

### 5 Analysis of Temporal Properties for Event-B Models

Leuschel et al. [13] developed a tool ProB for animating, model-checking, and verifying the consistency of Event-B models. ProB provides two ways for analysing Event-B models: constraint-based checking and temporal model-checking. We focus on temporal model-checking, since we are interested in liveness properties. Temporal model-checking [13] allows ProB to detect problems with a model (invariants violation, deadlocks, etc.) and to verify if the model satisfies LTL properties: ProB explores the state space of the model and tries to find a counter-example (i.e. a sequence of events) leading to the violation of invariants or LTL properties.

A difference with TLC (model-checker for TLA⁺) is that ProB does not support fairness [9], allowing unfair traces to be analysed during model-checking. Therefore, the TLA⁺ framework is more suited to our work, since we are verifying liveness properties, in Event-B models, under fairness assumptions.
6 Discussion, Conclusion and Future Work

We present a methodology based on liveness properties and refinement diagrams for modelling the self-* systems using EVENT B. We characterize the self-* systems by three modes (abstract states): 1) legal (correct) state, 2) illegal (faulty) state, and 3) recovery state. We have proposed a generic pattern for deriving correct self-* systems (see Fig.1). The service-as-event and call-as-event paradigms provide a way to express the relationships between modes for ensuring required properties as convergence. The correct-by-construction principle gives us the possibility to refine procedures from events and to link modes. The key idea is to identify the modes (considered as abstract states) and the required abstract steps to allow the navigation between modes, and then to gradually enrich abstract models, using refinement to introduce the concrete states and events. We have illustrated our methodology by the self-healing approach [14].

The complexity of the development is measured by the number of proof obligations (PO) which are automatically/manually discharged (see Table 1). It should be noted that a large majority (∼ 70%) of the 1177 manual proofs is solved by simply running the provers. The actual summary of proof obligations is given by Table 2. The manually discharged POs (327) require analysis and skills: searching and adding premises, simplifying the complex predicates, and even transforming goals are needed to discharge these POs. Examples of difficult POs are related to proving the finiteness of Management Peer Groups (MPG), during the redeployment operation of the self-configuration phase.

<table>
<thead>
<tr>
<th>Model</th>
<th>Total</th>
<th>Auto</th>
<th>Interactive</th>
</tr>
</thead>
<tbody>
<tr>
<td>CONTEXTS</td>
<td>30</td>
<td>26</td>
<td>66.67%</td>
</tr>
<tr>
<td>M0</td>
<td>3</td>
<td>3</td>
<td>100%</td>
</tr>
<tr>
<td>M1</td>
<td>21</td>
<td>15</td>
<td>71.4%</td>
</tr>
<tr>
<td>M2</td>
<td>46</td>
<td>39</td>
<td>84.8%</td>
</tr>
<tr>
<td>M3</td>
<td>68</td>
<td>0</td>
<td>0%</td>
</tr>
<tr>
<td>M4</td>
<td>142</td>
<td>16</td>
<td>11.27%</td>
</tr>
<tr>
<td>M5</td>
<td>46</td>
<td>17</td>
<td>39.5%</td>
</tr>
<tr>
<td>OTHER MACHINES</td>
<td>1083</td>
<td>141</td>
<td>12.44%</td>
</tr>
<tr>
<td>M21</td>
<td>13</td>
<td>0</td>
<td>0%</td>
</tr>
<tr>
<td>TOTAL</td>
<td>1434</td>
<td>257</td>
<td>17.9%</td>
</tr>
</tbody>
</table>

Table 1: Summary of Proof Obligations

Furthermore, our refinement-based formalization allows us to produce final local models close to the source code. Our future works include the development of techniques for generating applications from the resulting model extending tools like EB2ALL [16]. Moreover, further case studies will help us to discover new patterns; these patterns will be added to a catalogue of patterns that could be implemented in the Rodin platform. Finally, another point would be to take into account dependability properties in our methodology.

References


A Appendix: EVENT-B models
CONTEXT
  C00  
SETS
  SERVICES  
  STATES  
CONSTANTS
  RUN  
  FAIL  
  InitState  
AXIOMS
  axm1:  SERVICES ≠ ∅ not theorem  
  axm2:  STATES = {RUN, FAIL} not theorem  
  axm3:  RUN ≠ FAIL not theorem  
  axm4:  InitState ∈ SERVICES ↔ STATES not theorem  
  axm5:  ∀ s · s ∈ SERVICES ⇒ s ↦ RUN ∈ InitState not theorem  
  axm6:  ∀ s, st1, st2 · s ∈ SERVICES ∧ st1 ∈ STATES ∧ st2 ∈ STATES ∧ s ↦ 
        st1 ∈ InitState ∧ s ↦ st2 ∈ InitState ⇒ st1 = st2 not theorem  
END
CONTEXT
  C01
EXTENDS
  C00
SETS
  STATES_1
CONSTANTS
  RUN_1
  FAIL_1
  FAIL_DETECT_1
  InitState_1
AXIOMS
  axm1: partition(STATES_1, {RUN_1},{FAIL_1},{FAIL_DETECT_1}) not theorem
  axm2: InitState_1 ∈ SERVICES ↔ STATES_1 not theorem
  axm3: ∀ s · s ∈ SERVICES ⇒ s ↦ RUN_1 ∈ InitState_1 not theorem
  axm4: ∀ s, st1, st2 · s ∈ SERVICES ∧ st1 ∈ STATES_1 ∧ st2 ∈ STATES_1 ∧ s ↦ st1 ∈ InitState_1 ∧ s ↦ st2 ∈ InitState_1 ⇒ st1 = st2 not theorem
END
CONTEXT
  C02
EXTENDS
  C01
SETS
  STATES_2
CONSTANTS
  RUN_2
  FAIL_2
  FAIL_DETECT_2
  FAIL_ACTIV_2
  InitState_2
AXIOMS
  axm1: partition(STATES_2, {RUN_2},{FAIL_2},{FAIL_DETECT_2},
  {FAIL_ACTIV_2}) not theorem
  axm2: InitState_2 ∈ SERVICES ↔ STATES_2 not theorem
  axm3: ∀ s · s ∈ SERVICES ⇒ s ↦ RUN_2 ∈ InitState_2 not theorem
  axm4: ∀ s, st1, st2 · s ∈ SERVICES ∧ st1 ∈ STATES_2 ∧ st2 ∈ STATES_2 ∧
  s ↦ st1 ∈ InitState_2 ∧ s ↦ st2 ∈ InitState_2 ⇒ st1 = st2 not theorem
END
C03

CONTEXT
   C03
EXTENDS
   C02
SETS
   STATES_3
CONSTANTS
   RUN_3
   FAIL_3
   FAIL_DETECT_3
   FAIL_ACTIV_3
   FAIL_CONFIG_3
   FAIL_IGN_3
   InitState_3
AXIOMS
   axm1: partition(STATES_3, {RUN_3},{FAIL_3},{FAIL_DETECT_3},
   {FAIL_ACTIV_3},{FAIL_CONFIG_3},{FAIL_IGN_3}) not theorem
   axm2: InitState_3 ∈ SERVICES ↔ STATES_3 not theorem
   axm3: ∀ s · s ∈ SERVICES ⇒ s ↦ RUN_3 ∈ InitState_3 not theorem
   axm4: ∀ s, st1, st2 · s ∈ SERVICES ∧ st1 ∈ STATES_3 ∧ st2 ∈ STATES_3 ∧
   s ↦ st1 ∈ InitState_3 ∧ s ↦ st2 ∈ InitState_3 ⇒ st1 = st2 not theorem
END
CONTEXT
   C04
EXTENDS
   C03
SETS
   STATES_4
CONSTANTS
   RUN_4
   FAIL_4
   FAIL_DETECT_4
   FAIL_ACTIV_4
   FAIL_CONFIG_4
   FAIL_IGN_4
   DPL_4
   InitState_4
AXIOMS
   axm1: partition(STATES_4, {RUN_4},{FAIL_4},{FAIL_DETECT_4},
   {FAIL_ACTIV_4},{FAIL_CONFIG_4},{FAIL_IGN_4},{DPL_4}) not theorem
   axm2: InitState_4 ∈ SERVICES ↔ STATES_4 not theorem
   axm3: ∀ s · s ∈ SERVICES ⇒ s ↦ RUN_4 ∈ InitState_4 not theorem
   axm4: ∀ s, st1, st2 · s ∈ SERVICES ∧ st1 ∈ STATES_4 ∧ st2 ∈ STATES_4 ∧
   s ↦ st1 ∈ InitState_4 ∧ s ↦ st2 ∈ InitState_4 ⇒ st1 = st2 not theorem
END
CONTEXT  
  C05  
EXTENDS  
  C04  
CONSTANTS  
  min_inst  
  init_inst  
AXIOMS  
  axm1:  min_inst ∈ SERVICES → N1 not theorem  
  axm2:  init_inst ∈ SERVICES → N1 not theorem  
  axm3:  ∀ s · s ∈ SERVICES ⇒ min_inst(s) ≥ 2 not theorem  
  axm4:  ∀ s · s ∈ SERVICES ⇒ init_inst(s) ≥ min_inst(s) not theorem  
  axm5:  ∀ s · s ∈ SERVICES ⇒ init_inst(s) ≥ 2 theorem  
END
CONTEXT
C06
EXTENDS
C05
SETS
PEERS ▶ Set of PEERS
CONSTANTS
InitSrvcPeers ▶ Initial set of peers / instances per service
AXIOMS
axm1: InitSrvcPeers ∈ SERVICES → ℙ1(PEERS) not theorem ▶ each service is provided by a non empty set of peers/instances
axm2: ∀ s · s ∈ SERVICES ⇒ finite(InitSrvcPeers(s)) not theorem ▶ each service is provided by a finite set of peers/instances
axm3: ∀ s · s ∈ SERVICES ⇒ card(InitSrvcPeers(s)) = init_inst(s) not theorem ▶ each service s is provided by peers/instances, whose number is init_inst(s)
axm4: ∀ s1, s2 · s1 ⊆ PEERS ∧ s2 ⊆ PEERS ∧ s1 ≠ ∅ ∧ s2 ≠ ∅ ∧ finite(s1) ∧ finite(s2) ∧ s1 ⊆ s2 ⇒ card(s1) ≤ card(s2)−1 not theorem ▶
axm5: ∀ s1 · s1 ⊆ PEERS ∧ s1 ≠ ∅ ∧ finite(s1) ⇒ card(s1) > 0 theorem ▶
axm6: ∀ s1, s2 · s1 ⊆ PEERS ∧ s2 ⊆ PEERS ∧ finite(s1) ∧ finite(s2) ∧ s1 ⊆ s2 ⇒ card(s2) − card(s1) = card(s2\s1) not theorem ▶
END
CONTEXT

C07

EXTENDS

C06

CONSTANTS

deplo_inst

AXIOMS

axm1: ∀ set, s1, s2 · set ⊆ SERVICES×PEERS ∧ s1 ∈ SERVICES ∧ s2 ∈ SERVICES ∧ s1 = s2 ⇒ ({s1} ⩤ set)[{s2}] = ∅ theorem

axm2: ∀ set, s1, s2 · set ⊆ SERVICES×PEERS ∧ s1 ∈ SERVICES ∧ s2 ∈ SERVICES ∧ s1 ≠ s2 ⇒ ({s1} ⩤ set)[{s2}] = set[{s2}] theorem

axm3: ∀ set, s1, s2, p · set ⊆ SERVICES×PEERS ∧ s1 ∈ SERVICES ∧ s2 ∈ SERVICES ∧ p ∈ PEERS ∧ s1 = s2 ⇒ (set u {s1 ↦ p})[{s2}] = set[{s2}] u {p} theorem

axm4: ∀ set, s1, s2, p · set ⊆ SERVICES×PEERS ∧ s1 ∈ SERVICES ∧ s2 ∈ SERVICES ∧ p ∈ PEERS ∧ s1 ≠ s2 ⇒ (set u {s1 ↦ p})[{s2}] = set[{s2}] theorem

axm5: deplo_inst ∈ SERVICES → ℕ1 not theorem

END
CONTEXT
C08

EXTENDS
C07

CONSTANTS
init_tok
InitStatus
InitSuspPeers
InitFail

AXIOMS
axm1: init_tok ∈ SERVICES → PEERS not theorem
axm2: ∀ s · s ∈ SERVICES ⇒ init_tok(s) ∈ InitSrvcPeers(s) not theorem
axm3: ∀ a1, a2 · a1 ∈ PEERS ↔ (SERVICES×PEERS) ∧ a2 ∈ PEERS ↔ (SERVICES×PEERS) ∧ finite(a1) ∧ a2 ⊆ a1 ⇒ finite(a2) not theorem
axm4: InitStatus ∈ (PEERS × SERVICES) ↔ STATES_4 not theorem
axm5: ∀ s, p · s ∈ SERVICES ∧ p ∈ PEERS ∧ p = init_tok(s) ⇒ (p ↦ s) ↦ RUN_4 ∈ InitStatus not theorem
axm6: ∀ s, p, stt · s ∈ SERVICES ∧ p ∈ PEERS ∧ stt ∈ STATES_4 ∧ (p ↦ s) ↦ stt ∈ InitStatus ⇒ p = init_tok(s) ∧ stt = RUN_4 not theorem
axm7: InitSuspPeers ∈ (PEERS×SERVICES) ↔ ℙ(PEERS) not theorem
axm8: ∀ p, s, sp · p ∈ PEERS ∧ s ∈ SERVICES ∧ sp ⊆ PEERS ∧ (p ↦ s) ↦ sp ∈ InitSuspPeers ⇒ p = init_tok(s) ∧ sp = ∅ not theorem
axm9: ∀ p, s · p ∈ PEERS ∧ s ∈ SERVICES ∧ p = init_tok(s) ⇒ (p ↦ s) ↦ ∅ ∈ InitSuspPeers not theorem
axm10: InitFail ∈ SERVICES → ℙ(PEERS) not theorem
axm11: ∀ s · s ∈ SERVICES ⇒ InitFail(s) = ∅ not theorem

END
CONTEXT
  C09  
EXTENDS
  C08
CONSTANTS
  InitStateSvc
  InitSuspPrs
  InitRunPeers
AXIOMS
  axm1:  InitStateSvc ∈ PEERS × SERVICES ↔ STATES_4 not theorem >
  axm2:  ∀ s, p · p ∈ PEERS ∧ s ∈ SERVICES ∧ p ∈ InitSrvcPeers(s) ⇒ (p ↦ s) ↦ RUN_4 ∈ InitStateSvc not theorem >
  axm3:  ∀ s, p, stt · p ∈ PEERS ∧ s ∈ SERVICES ∧ (p ↦ s) ↦ stt ∈ InitStateSvc ⇒ p ∈ InitSrvcPeers(s) ∧ stt = RUN_4 not theorem >
  axm4:  InitSuspPrs ∈ PEERS × SERVICES ↔ ℙ(PEERS) not theorem >
  axm5:  ∀ s, p · p ∈ PEERS ∧ s ∈ SERVICES ∧ p ∈ InitSrvcPeers(s) ⇒ (p ↦ s) ↦ ∅ ∈ InitSuspPrs not theorem >
  axm6:  ∀ s, p, stt · p ∈ PEERS ∧ s ∈ SERVICES ∧ (p ↦ s) ↦ stt ∈ InitSuspPrs ⇒ p ∈ InitSrvcPeers(s) ∧ stt = ∅ not theorem >
  axm7:  InitRunPeers ∈ PEERS × SERVICES ↔ ℙ(PEERS) not theorem >
  axm8:  ∀ s, p · p ∈ PEERS ∧ s ∈ SERVICES ∧ p ∈ InitSrvcPeers(s) ⇒ (p ↦ s) ↦ InitSrvcPeers(s) ∈ InitRunPeers not theorem >
  axm9:  ∀ s, p, stt · p ∈ PEERS ∧ s ∈ SERVICES ∧ (p ↦ s) ↦ stt ∈ InitRunPeers ⇒ p ∈ InitSrvcPeers(s) ∧ stt = InitSrvcPeers(s) not theorem >
END
M00

M00

SEES

c00

VARIABLES

serviceState

INVARIANTS

inv1: serviceState ∈ SERVICES ↔ STATES not theorem

inv2: ∀ s, st1, st2 · s ∈ SERVICES ∧ st1 ∈ STATES ∧ st2 ∈ STATES ∧ s ↦ st1 ∈ serviceState ∧ s ↦ st2 ∈ serviceState ⇒ st1 = st2

NOT THEOREM

EVENTS

INITIALISATION: not extended ordinary

THEN

act1: serviceState = InitState

END

FAIL: not extended ordinary

ANY

s

WHERE

grd1: s ∈ SERVICES not theorem

THEN

act1: serviceState = (\{s\} ↔ serviceState) u \{s ↦ FAIL\}

END

HEAL: not extended ordinary

ANY

s

WHERE

grd1: s ∈ SERVICES not theorem

grd2: s ↦ FAIL ∈ serviceState not theorem

THEN

act1: serviceState = (serviceState \ {s ↦ FAIL}) u \{s ↦ RUN\}

END

END
MACHINE
M01
REFINES
M00
SEES
C01
VARIABLES
serviceState_1
INVARIANTS
inv1: serviceState_1 ∈ SERVICES ↔ STATES_1 not theorem
  gluing_run1: ∀ s · s ∈ SERVICES ∧ s ↦ RUN ∈ serviceState ⇒ s ↦ RUN_1 ∈ serviceState_1 not theorem
  gluing_run2: ∀ s · s ∈ SERVICES ∧ s ↦ RUN_1 ∈ serviceState_1 ⇒ s ↦ RUN ∈ serviceState not theorem
  gluing_fail1: ∀ s · s ∈ SERVICES ∧ s ↦ FAIL ∈ serviceState ⇒ (s ↦ FAIL_1 ∈ serviceState_1 ∨ s ↦ FAIL_DETECT_1 ∈ serviceState_1) not theorem
  gluing_fail2: ∀ s, st · s ∈ SERVICES ∧ st ∈ STATES_1 ∧ st ∈ {FAIL_1,FAIL_DETECT_1} ∧ s ↦ st ∈ serviceState_1 ⇒ s ↦ FAIL ∈ serviceState not theorem
  gluing_state3: ∀ s, st1, st2 · s ∈ SERVICES ∧ st1 ∈ STATES_1 ∧ st2 ∈ STATES_1 ∧ s ↦ st1 ∈ serviceState_1 ∧ s ↦ st2 ∈ serviceState_1 ⇒ st1 = st2 not theorem
EVENTS
INITIALISATION: not extended ordinary
  THEN
  act1: serviceState_1 ≔ InitState_1
END
FAIL: not extended ordinary
  REFINES
  FAIL
  ANY
  s
  WHERE
  grd1: s ∈ SERVICES not theorem
  grd2: s ↦ RUN_1 ∈ serviceState_1 not theorem
  THEN
  act1: serviceState_1 = (serviceState_1\{s ↦ RUN_1\}) ∪ {s ↦
FAIL_1}
END
FAIL_DETECT: not extended ordinary
  REFINES
  FAIL
  ANY
  s
  WHERE
  grd1: s ∈ SERVICES not theorem
M01

\[ \text{grd2: } s \rightarrow \text{FAIL}_1 \in \text{serviceState}_1 \text{ not theorem } \]

THEN

act1: \[ \text{serviceState}_1 = (\text{serviceState}_1 \setminus \{s \rightarrow \text{FAIL}_1\}) \cup \{s \rightarrow \text{FAIL_DETECT}_1\} \]

END

\[ \text{END} \]

\[ \text{EAL: } \text{not extended ordinary } \]

REFINES

HEAL

ANY

s

WHERE

\[ \text{grd1: } s \in \text{SERVICES} \text{ not theorem } \]

\[ \text{grd2: } s \rightarrow \text{FAIL_DETECT}_1 \in \text{serviceState}_1 \text{ not theorem } \]

THEN

act1: \[ \text{serviceState}_1 = (\text{serviceState}_1 \setminus \{s \rightarrow \text{FAIL_DETECT}_1\}) \]

u \{s \rightarrow \text{RUN}_1\}

END

END
MACHINE
M02
REFINES
M01
SEES
C02
\[\text{RFAI}\]
\[\text{RIA}\]
servi2\[\text{\_state}\] \>

INVIARNTS
\[\text{\_ prepares serviceState}_2 \text{SERIES}_2 \text{S TATES}_2 \text{serviceState}_2 \Rightarrow \text{s}_1 \]
\[\text{\_ prepares s SERIES}_2 s \text{S RON}_2 \text{serviceState}_2 \Rightarrow s \]
\[\text{\_ prepares serviceState}_2 \text{SERIES}_2 s \text{S FAI}_2 \text{serviceState}_2 \Rightarrow s \]
\[\text{\_ prepares serviceState}_2 \text{SERIES}_2 s \text{S FAI}_2 \text{TEC}_2 \text{serviceState}_2 \Rightarrow (s \text{FAI}_2 \text{C1}_2 \text{serviceState}_2 \) \text{SERIES}_2 s \text{S S TATE}_2 \text{STATE}_2 \rightarrow \text{s}_1 \]
\[\text{\_ prepares serviceState}_2 \text{SERIES}_2 s \text{S S TATE}_2 \text{STATE}_2 \rightarrow \text{s}_1 \]
\[\text{\_ prepares serviceState}_2 \text{SERIES}_2 s \text{S S TATE}_2 \text{STATE}_2 \rightarrow \text{s}_2 \]
\[\text{\_ prepares serviceState}_2 \text{SERIES}_2 s \text{S S TATE}_2 \text{STATE}_2 \rightarrow \text{s}_2 \]
\[\text{\_ prepares serviceState}_2 \text{SERIES}_2 s \text{S S TATE}_2 \text{STATE}_2 \rightarrow \text{s}_2 \]

EVENTS
INITIALISATION: \[\text{\_ prepares or}_2 \text{prepare serviceState}_2 \text{SERIES}_2 s \text{S TATE}_2 \text{STATE}_2 \rightarrow \text{s}_2 \]
\[\text{\_ prepares or}_2 \text{prepare serviceState}_2 \text{SERIES}_2 s \text{S TATE}_2 \text{STATE}_2 \rightarrow \text{s}_2 \]
\[\text{\_ prepares or}_2 \text{prepare serviceState}_2 \text{SERIES}_2 s \text{S TATE}_2 \text{STATE}_2 \rightarrow \text{s}_2 \]
\[\text{\_ prepares or}_2 \text{prepare serviceState}_2 \text{SERIES}_2 s \text{S TATE}_2 \text{STATE}_2 \rightarrow \text{s}_2 \]

FAIL \[\text{\_ prepares or}_2 \text{prepare serviceState}_2 \text{SERIES}_2 s \text{S TATE}_2 \text{STATE}_2 \rightarrow \text{s}_2 \]
REFINES
\[\text{\_ prepares FAI}_2 \text{prepare serviceState}_2 \text{SERIES}_2 s \text{S TATE}_2 \text{STATE}_2 \rightarrow \text{s}_2 \]
\[\text{\_ prepares FAI}_2 \text{prepare serviceState}_2 \text{SERIES}_2 s \text{S TATE}_2 \text{STATE}_2 \rightarrow \text{s}_2 \]
\[\text{\_ prepares FAI}_2 \text{prepare serviceState}_2 \text{SERIES}_2 s \text{S TATE}_2 \text{STATE}_2 \rightarrow \text{s}_2 \]
\[\text{\_ prepares FAI}_2 \text{prepare serviceState}_2 \text{SERIES}_2 s \text{S TATE}_2 \text{STATE}_2 \rightarrow \text{s}_2 \]

FAIL \[\text{\_ prepares or}_2 \text{prepare serviceState}_2 \text{SERIES}_2 s \text{S TATE}_2 \text{STATE}_2 \rightarrow \text{s}_2 \]
REFINES
\[\text{\_ prepares FAI}_2 \text{prepare serviceState}_2 \text{SERIES}_2 s \text{S TATE}_2 \text{STATE}_2 \rightarrow \text{s}_2 \]
\[\text{\_ prepares FAI}_2 \text{prepare serviceState}_2 \text{SERIES}_2 s \text{S TATE}_2 \text{STATE}_2 \rightarrow \text{s}_2 \]
\[\text{\_ prepares FAI}_2 \text{prepare serviceState}_2 \text{SERIES}_2 s \text{S TATE}_2 \text{STATE}_2 \rightarrow \text{s}_2 \]
\[\text{\_ prepares FAI}_2 \text{prepare serviceState}_2 \text{SERIES}_2 s \text{S TATE}_2 \text{STATE}_2 \rightarrow \text{s}_2 \]

FAIL \[\text{\_ prepares or}_2 \text{prepare serviceState}_2 \text{SERIES}_2 s \text{S TATE}_2 \text{STATE}_2 \rightarrow \text{s}_2 \]
REFINES
\[\text{\_ prepares FAI}_2 \text{prepare serviceState}_2 \text{SERIES}_2 s \text{S TATE}_2 \text{STATE}_2 \rightarrow \text{s}_2 \]
\[\text{\_ prepares FAI}_2 \text{prepare serviceState}_2 \text{SERIES}_2 s \text{S TATE}_2 \text{STATE}_2 \rightarrow \text{s}_2 \]
\[\text{\_ prepares FAI}_2 \text{prepare serviceState}_2 \text{SERIES}_2 s \text{S TATE}_2 \text{STATE}_2 \rightarrow \text{s}_2 \]
\[\text{\_ prepares FAI}_2 \text{prepare serviceState}_2 \text{SERIES}_2 s \text{S TATE}_2 \text{STATE}_2 \rightarrow \text{s}_2 \]

FAIL \[\text{\_ prepares or}_2 \text{prepare serviceState}_2 \text{SERIES}_2 s \text{S TATE}_2 \text{STATE}_2 \rightarrow \text{s}_2 \]
REFINES
\[\text{\_ prepares FAI}_2 \text{prepare serviceState}_2 \text{SERIES}_2 s \text{S TATE}_2 \text{STATE}_2 \rightarrow \text{s}_2 \]
\[\text{\_ prepares FAI}_2 \text{prepare serviceState}_2 \text{SERIES}_2 s \text{S TATE}_2 \text{STATE}_2 \rightarrow \text{s}_2 \]
\[\text{\_ prepares FAI}_2 \text{prepare serviceState}_2 \text{SERIES}_2 s \text{S TATE}_2 \text{STATE}_2 \rightarrow \text{s}_2 \]
\[\text{\_ prepares FAI}_2 \text{prepare serviceState}_2 \text{SERIES}_2 s \text{S TATE}_2 \text{STATE}_2 \rightarrow \text{s}_2 \]
FAIECT

AN

□ >

BRE

AN\)

□ >

BRE

THEN

an\) \text{ serviceState} \text{2} \text{FAI2}\) 20\)

FAIECT2\)

EN\)

IS_OK\)

\text{anye}\ or\text{enate}\ >

REFINES

HEA\)

AN\)

□ >

BRE

THEN

\text{serviceState} \text{2} \text{FAI2}\) 20\)

\text{serviceState} \text{2} \text{FAI2}\)

FAIL_ACTI\)

\text{anye}\ or\text{enate}\ >

AN\)

□ >

BRE

THEN

\text{serviceState} \text{2} \text{FAI2}\) 20\)

\text{serviceState} \text{2} \text{FAI2}\)

EAL\)

\text{anye}\ or\text{enate}\ >

REFINES

HEA\)

AN\)

□ >

BRE

THEN

\text{serviceState} \text{2} \text{FAI2}\) 20\)

\text{serviceState} \text{2} \text{FAI2}\)

FAIL\)

□ >

EN\)
MACHINE M03
REFINES M03
SEES C03
RIASE servic3

INVARIANTS
<table>
<thead>
<tr>
<th>I L G</th>
<th>A N E</th>
</tr>
</thead>
</table>

RUN servic3
<table>
<thead>
<tr>
<th>I L G</th>
<th>A N E</th>
</tr>
</thead>
</table>

FAIL
<table>
<thead>
<tr>
<th>I L G</th>
<th>A N E</th>
</tr>
</thead>
</table>

EVENTS
<table>
<thead>
<tr>
<th>I L G</th>
<th>A N E</th>
</tr>
</thead>
</table>

INITIALISATION: servic3

FAIL: servic3

BRE

THEN

FAIL:
FAIL_DETECT: not extended ordinary

REFINES
   FAIL_DETECT

AND
   G

BRE
   grd1 : s ∈ SERVICES not theorem
   grd3 : st ∈ {FAIL_CONFIG_3, FAIL_IGN_3} not theorem

THEN
   a → grd1 : s ∈ SERVICES not theorem
   a → grd3 : st ∈ {FAIL_CONFIG_3, FAIL_IGN_3} not theorem

END

IS_OK: not extended ordinary

REFINES
   IS_OK

AND
   G

BRE
   grd1 : s ∈ SERVICES not theorem
   grd3 : st ∈ {FAIL_CONFIG_3, FAIL_IGN_3} not theorem

THEN
   a → grd1 : s ∈ SERVICES not theorem
   a → grd3 : st ∈ {FAIL_CONFIG_3, FAIL_IGN_3} not theorem

END

FAIL_ACTION: not extended ordinary

REFINES
   FAIL_ACTION

AND
   G

BRE
   grd1 : s ∈ SERVICES not theorem
   grd3 : st ∈ {FAIL_CONFIG_3, FAIL_IGN_3} not theorem

THEN
   a → grd1 : s ∈ SERVICES not theorem
   a → grd3 : st ∈ {FAIL_CONFIG_3, FAIL_IGN_3} not theorem

END

FAIL_CONFIG: not extended ordinary

ANY
   s
   st

WHERE
   grd1: s ∈ SERVICES not theorem
   grd3: st ∈ {FAIL_CONFIG_3, FAIL_IGN_3} not theorem
M03

grd2:  s ↦ FAIL_ACTIV_3 ∈ serviceState_3 not theorem > 
THEN
act1: serviceState_3 = (serviceState_3 \ {s ↦ FAIL_ACTIV_3}) ∪ 
{s ↦ st} > 
END

HEAL:  not extended ordinary >
REFINES
HEAL
ANY
s >
st >
WHERE
grd1:  s ∈ SERVICES not theorem >
grd3:  st ∈ {FAIL_CONFIG_3,FAIL_IGN_3} not theorem >
grd2:  s ↦ st ∈ serviceState_3 not theorem >
THEN
act1: serviceState_3 = (serviceState_3 \ {s ↦ st}) ∪ {s ↦ 
RUN_3} > 
END

END
MACHINE M04
REFINES M03
SEES C04
VARIABLES serviceState_4

INVARIANTS
inv1: serviceState_4 ∈ SERVICES ↔ S TATES_4
inv2: serviceState_4 ∈ SERVICES ↔ S TATES_4
inv3: serviceState_4 ∈ SERVICES ↔ S TATES_4

EVENTS
INITIALISATION: not handling ordinary
HEN ct1: YHrvicHStatH_4 ≔ InitStatH_4
END

FAIL: not handled ordinary
REFINES FAIL
WHERE \( \text{grd1} : Y \in \text{SERVICES} \rightarrow \) theorem
\( \text{grd2} : Y \mapsto \text{RUN}_4 \in Y \text{ } \text{HrvicHStat}_4 \rightarrow \) theorem
\( \text{HEN} \)
\( \text{ct1} : Y \text{ } \text{HrvicHStat}_4 \equiv (Y \text{ } \text{HrvicHStat}_4 \setminus \{Y \mapsto \text{RUN}_4\}) \cup \{Y \mapsto \text{FAIL}_4\} \)
\( \text{END} \)
\( \text{FAIL\_DETECT} \rightarrow \) theorem
\( \text{REFINES} \)
\( \text{FAIL} \)
\( \text{AN} \)
\( \text{HERE} \)
\( \text{IF} \) theorem
\( \text{THEN} \)
\( a_{\text{FAIL\_DETECT}} \rightarrow \) theorem
\( \text{END} \)
\( \text{FAIL\_DETECT} \rightarrow \) theorem
\( \text{REFINES} \)
\( \text{FAIL\_DETECT} \)
\( \text{AN} \)
\( \text{HERE} \)
\( \text{IF} \) theorem
\( \text{THEN} \)
\( a_{\text{FAIL\_DETECT}} \rightarrow \) theorem
\( \text{END} \)
\( \text{IS\_OK} \rightarrow \) theorem
\( \text{REFINES} \)
\( \text{IS\_OK} \)
\( \text{AN} \)
\( \text{HERE} \)
\( \text{IF} \) theorem
\( \text{THEN} \)
\( a_{\text{FAIL\_DETECT}} \rightarrow \) theorem
\( \text{END} \)
\( \text{FAIL\_ACTIV} \rightarrow \) theorem
\( \text{REFINES} \)
\( \text{FAIL\_ACTIV} \)
\( \text{AN} \)
\( \text{HERE} \)
\( \text{IF} \) theorem
\( \text{THEN} \)
\( a_{\text{FAIL\_ACTIV}} \rightarrow \) theorem
\( \text{END} \)
FAIL CONFIGURE:  

REFINES 
FAIL CONFIGURE: 

AN: s > 

HERE 

s ▷ SERVICES not theorem > 
gr 4: s ▷ FAIL CONFIG_4 ∈ serviceState_4 

THEN 

s ∈ serviceState_4 > 4 

WHERE 

grd1: s ∈ SERVICES not theorem > 
gr 4: s ▷ FAIL CONFIG_4 ∈ serviceState_4 

THEN 

WHERE 

s ∈ serviceState_4 > 4 

END 

FAIL IGNORE:  

REFINES 
FAIL IGNORE: 

AN: s > 

HERE 

s ▷ SERVICES not theorem > 
gr 4: s ▷ FAIL CONFIG_4 ∈ serviceState_4 

THEN 

s ∈ serviceState_4 > 4 

FAIL IGNORE:  

THEN 

WHERE 

s ∈ serviceState_4 > 4 

END 

REDEPLOY:  

AN: 

END
WHERE
\begin{align*}
\text{grd1} & : Y \in \text{SERVICES} \\
\text{grd2} & : Y \mapsto \text{FAIL\_CONFIG\_4} \in \text{YHrvichState}_4
\end{align*}
\not\text{theorem}
\text{END}

\begin{align*}
\text{ct1} & : \text{YHrvichState}_4 \equiv (\text{YHrvichState}_4 \setminus \{Y \mapsto \text{FAIL\_CONFIG\_4}\}) \\
& \cup \{Y \mapsto \text{DPL\_4}\}
\end{align*}
\not\text{handled ordinary}
\not\text{REFINES HEAL}
\not\text{END}

\begin{align*}
\text{ct1} & : \text{YHrvichState}_4 \equiv (\text{YHrvichState}_4 \setminus \{Y \mapsto \text{DPL\_4}\}) \\
& \cup \{Y \mapsto \text{RUN\_4}\}
\end{align*}
\not\text{handled ordinary}
\not\text{REFINES HEAL}
\not\text{END}

\begin{align*}
\text{ct1} & : \text{YHrvichState}_4 \equiv (\text{YHrvichState}_4 \setminus \{Y \mapsto \text{DPL\_4}\}) \\
& \cup \{Y \mapsto \text{RUN\_4}\}
\end{align*}
\not\text{handled ordinary}
\not\text{REFINES HEAL}
\not\text{END}
MACHINE M05 
REFINES M04 
SEES C05 
VARIABLES 
  serviceState_4 > 
  num_run > 
  num_susp > 
INVARIANTS 
  inv1: num_run ∈ SERVICES → N not theorem > 
  inv2: num_susp ∈ SERVICES → N not theorem > 
  inv3: ∀ s, st · s ∈ SERVICES ∧ st ∈ STATES_4 ∧ st ∉ {FAIL_4, FAIL_DETECT_4} ∧ s ↦ st ∈ serviceState_4 ⇒ num_susp(s) = 0 not theorem > 
  inv4: ∀ s · s ∈ SERVICES ∧ s ↦ RUN_4 ∈ serviceState_4 ⇒ num_susp(s) = 0 theorem > 
  inv5: ∀ s · s ∈ SERVICES ∧ s ↦ FAIL_CONFIG_4 ∈ serviceState_4 ⇒ num_run(s) < min_inst(s) not theorem > 
  inv6: ∀ s · s ∈ SERVICES ⇒ num_susp(s) < num_run(s) not theorem > 
EVENTS 
INITIALISATION: extended ordinary > 
  THEN 
  act1: serviceState_4 = InitState_4 > 
  act2: num_run = init_inst > 
  act3: num_susp = SERVICES×{0} > 
  END 
FAIL: extended ordinary > 
  REFINES FAIL ANY s > 
  nb_fail > 
  WHERE 
  grd1: s ∈ SERVICES not theorem > 
  grd2: s ↦ RUN_4 ∈ serviceState_4 not theorem > 
  grd3: nb_fail ∈ N1 not theorem > 
  grd4: nb_fail < num_run(s) not theorem > 
  THEN 
  act1: serviceState_4 = (serviceState_4\{s ↦ RUN_4}) ∪ {s ↦ FAIL_4} > 
  act2: num_susp(s) = nb_fail > 
  END 
FAIL_DETECT: extended ordinary > 
  REFINES FAIL_DETECT
M05

\( \forall s \in \text{SERVICES} \) not theorem

\( \forall s \rightarrow \text{FAIL}_4 \in \text{serviceState}_4 \) not theorem

\( \forall \text{num_safe} \in \mathbb{N} \) not theorem

\( \forall \text{num_safe} \leq \text{num_susp}(s) \) not theorem

THEN

\( \forall \text{act1}: \text{serviceState}_4 := (\text{serviceState}_4 \setminus \{s \rightarrow \text{FAIL}_4\}) \cup \{s \rightarrow \text{FAIL_DETECT}_4\} \)

\( \forall \text{act2}: \text{num_susp}(s) := \text{num_susp}(s) - \text{num_safe} \)

END

\( \text{IS_OK:} \) extended ordinary

REFINES

\( \text{IS_OK} \)

\( \forall s \in \text{SERVICES} \) not theorem

\( \forall s \rightarrow \text{FAIL_DETECT}_4 \in \text{serviceState}_4 \) not theorem

\( \forall \text{num_susp}(s) = 0 \) not theorem

THEN

\( \forall \text{act1}: \text{serviceState}_4 := (\text{serviceState}_4 \setminus \{s \rightarrow \text{FAIL_DETECT}_4\}) \cup \{s \rightarrow \text{RUN}_4\} \)

END

\( \text{FAIL_ACTIV:} \) extended ordinary

REFINES

\( \text{FAIL_ACTIV} \)

\( \forall s \in \text{SERVICES} \) not theorem

\( \forall s \rightarrow \text{FAIL_DETECT}_4 \in \text{serviceState}_4 \) not theorem

\( \forall \text{num_susp}(s) > 0 \) not theorem

THEN

\( \forall \text{act1}: \text{serviceState}_4 := (\text{serviceState}_4 \setminus \{s \rightarrow \text{FAIL_DETECT}_4\}) \cup \{s \rightarrow \text{FAIL_ACTIV}_4\} \)

\( \forall \text{act2}: \text{num_run}(s) := \text{num_run}(s) - \text{num_susp}(s) \)

\( \forall \text{act3}: \text{num_susp}(s) := 0 \)

END

\( \text{FAIL_CONFIGURE:} \) extended ordinary

REFINES

\( \text{FAIL_CONFIGURE} \)

\( \forall s \in \text{SERVICES} \) not theorem

\( \forall s \rightarrow \text{FAIL_DETECT}_4 \in \text{serviceState}_4 \) not theorem

\( \forall \text{num_susp}(s) > 0 \) not theorem

THEN

\( \forall \text{act1}: \text{serviceState}_4 := (\text{serviceState}_4 \setminus \{s \rightarrow \text{FAIL_DETECT}_4\}) \cup \{s \rightarrow \text{FAIL_ACTIV}_4\} \)

\( \forall \text{act2}: \text{num_run}(s) := \text{num_run}(s) - \text{num_susp}(s) \)

\( \forall \text{act3}: \text{num_susp}(s) := 0 \)

END

Page 2
M05

s >

WHERE
grd1: s ∈ SERVICES not theorem
grd2: s ↦ FAIL_ACTIV_4 ∈ serviceState_4 not theorem
grd3: num_run(s) < min_inst(s) not theorem

THEN
act1: serviceState_4 ≔ (serviceState_4 \ {s ↦ FAIL_ACTIV_4}) ∪ 
{s ↦ FAIL_CONFIG_4} >

END

FAIL_IGNORE: extended ordinary
REFINES
FAIL_IGNORE

ANY

s >

WHERE
grd1: s ∈ SERVICES not theorem
grd2: s ↦ FAIL_ACTIV_4 ∈ serviceState_4 not theorem
grd3: num_run(s) ≥ min_inst(s) not theorem

THEN
act1: serviceState_4 ≔ (serviceState_4 \ {s ↦ FAIL_ACTIV_4}) ∪ 
{s ↦ FAIL_IGN_4} >

END

IGNORE: extended ordinary
REFINES
IGNORE

ANY

s >

WHERE
grd1: s ∈ SERVICES not theorem
grd2: s ↦ FAIL_IGN_4 ∈ serviceState_4 not theorem

THEN
act1: serviceState_4 ≔ (serviceState_4 \ {s ↦ FAIL_IGN_4}) ∪ 
{s ↦ RUN_4} >

END

REDEPLOY: extended ordinary
REFINES
REDEPLOY

ANY

s >

WHERE
grd1: s ∈ SERVICES not theorem
grd2: s ↦ FAIL_CONFIG_4 ∈ serviceState_4 not theorem
grd3: new_run ∈ N1 not theorem
grd4: new_run ≥ min_inst(s) not theorem

}
THEN
  act1:  serviceState_4 = (serviceState_4 ∖ {s ↦ FAIL_CONFIG_4})
         ∪ {s ↦ DPL_4}
  act2:  num_run(s) = new_run
END

THEN
  act1:  serviceState_4 = (serviceState_4 ∖ {s ↦ DPL_4}) ∪ {s ↦ RUN_4}
END

END
MACHINE M06
REFINES M05
SEES C06
VARIABLES
  serviceState_4
  run_peers
  susp_peers
  fail_peers

INTEGRANTS
  inv1: run_peers ∈ SERVICES → ℙ1(PEERS) not theorem
  inv2: susp_peers ∈ SERVICES → ℙ(PEERS) not theorem
  inv3: fail_peers ∈ SERVICES ↔ PEERS not theorem

  gluing_run1: ∀ s · s ∈ SERVICES ⇒ finite(run_peers(s)) not theorem
              → the number of instances providing a service s is finite
  gluing_run2: ∀ s · s ∈ SERVICES ⇒ num_run(s) = card(run_peers(s))
              not theorem → the number of instances providing a service s is num_run_peers(s)
  gluing_susp1: ∀ s · s ∈ SERVICES ∧ s ∈ dom(susp_peers) ⇒ finite
              (susp_peers(s)) not theorem → the number of suspect instances of a service s is finite
  gluing_susp2: ∀ s · s ∈ SERVICES ∧ s ∈ dom(susp_peers) ⇒ num_susp(s)
              = card(susp_peers(s)) not theorem → the number of suspect instances of a service s is num_susp_peers(s)
  inv4: ∀ s · s ∈ SERVICES ⇒ run_peers(s) ∩ fail_peers[{s}] = ∅ not theorem
  inv5: ∀ s · s ∈ SERVICES ∧ s ∈ dom(susp_peers) ⇒ susp_peers(s) ⊆ run_peers(s) not theorem
  inv6: ∀ s, st · s ∈ SERVICES ∧ st ∈ STATES_4 ∧ st ∈ \{FAIL_4, FAIL_DETECT_4\} ∧ s ↦ st ∈ serviceState_4 ⇒ s ∈ dom(susp_peers)
  inv7: ∀ s, st · s ∈ SERVICES ∧ st ∈ STATES_4 ∧ st ∈ \{FAIL_4, FAIL_DETECT_4\} ∧ s ↦ st ∈ serviceState_4 ⇒ susp_peers(s) ⊂ run_peers(s) not theorem

EVENTS
INITIALISATION: not extended ordinary
  THEN
    act1: serviceState_4 ≔ InitState_4
    act2: run_peers ≔ InitSrvcPeers
    act3: susp_peers ≔ ∅
    act4: fail_peers ≔ ∅

END

FAIL: not extended ordinary
  REFINES FAIL
  ANY
\[ s \rightarrow fp \]

\textbf{WHERE}

\[ \text{grd1: } s \in \text{SERVICES} \not\text{ theorem} \rightarrow \]
\[ \text{grd2: } s \rightarrow \text{RUN}_4 \in \text{serviceState}_4 \not\text{ theorem} \rightarrow \]
\[ \text{grd5: } fp \subseteq \text{PEERS} \not\text{ theorem} \rightarrow \]
\[ \text{grd3: } fp \neq \emptyset \not\text{ theorem} \rightarrow \]
\[ \text{grd4: } fp \subset \text{run_peers}(s) \not\text{ theorem} \rightarrow \]

\textbf{WITH}

\[ \text{nb_fail: } \text{nb_fail} = \text{card}(fp) \rightarrow \]

\textbf{THEN}

\[ \text{act1: } \text{serviceState}_4 \equiv (\text{serviceState}_4 \setminus \{s \mapsto \text{RUN}_4\}) \cup \{s \mapsto \text{FAIL}_4\} \rightarrow \]
\[ \text{act2: } \text{susp_peers}(s) \equiv fp \rightarrow \]

\textbf{END}

\textbf{FAIL\_DETECT: not extended ordinary } \rightarrow \textbf{REFINES FAIL\_DETECT ANY}

\[ s \rightarrow sf \rightarrow \]

\textbf{WHERE}

\[ \text{grd1: } s \in \text{SERVICES} \not\text{ theorem} \rightarrow \]
\[ \text{grd2: } s \mapsto \text{FAIL}_4 \in \text{serviceState}_4 \not\text{ theorem} \rightarrow \]
\[ \text{grd5: } \text{susp_peers}(s) \neq \emptyset \not\text{ theorem} \rightarrow \]
\[ \text{grd6: } sf \subseteq \text{PEERS} \not\text{ theorem} \rightarrow \]
\[ \text{grd7: } sf \subseteq \text{susp_peers}(s) \not\text{ theorem} \rightarrow \]

\textbf{WITH}

\[ \text{num_safe: } \text{num_safe} = \text{card}(sf) \rightarrow \]

\textbf{THEN}

\[ \text{act1: } \text{serviceState}_4 \equiv (\text{serviceState}_4 \setminus \{s \mapsto \text{FAIL}_4\}) \cup \{s \mapsto \text{FAIL\_DETECT}_4\} \rightarrow \]
\[ \text{act2: } \text{susp_peers}(s) \equiv \text{susp_peers}(s) \setminus sf \rightarrow \]

\textbf{END}

\textbf{IS\_OK: not extended ordinary } \rightarrow \textbf{REFINES IS\_OK ANY}

\[ s \rightarrow \]

\textbf{WHERE}

\[ \text{grd1: } s \in \text{SERVICES} \not\text{ theorem} \rightarrow \]
\[ \text{grd2: } s \mapsto \text{FAIL\_DETECT}_4 \in \text{serviceState}_4 \not\text{ theorem} \rightarrow \]
\[ \text{grd5: } \text{susp_peers}(s) = \emptyset \not\text{ theorem} \rightarrow \]

\textbf{THEN}

\[ \text{act1: } \text{serviceState}_4 \equiv (\text{serviceState}_4 \setminus \{s \mapsto \text{FAIL\_DETECT}_4\}) \]
\[ u \{s \mapsto \text{RUN}_4\} \rightarrow \]
FAIL_ACTIV: not extended ordinary
  REFINES FAIL_ACTIV
  ANY s
  WHERE
    grd1: s ∈ SERVICES not theorem
    grd2: s ↦ FAIL_DETECT_4 ∈ serviceState_4 not theorem
    grd5: susp_peers(s) ≠ ∅ not theorem
  THEN
    act1: serviceState_4 ≔ (serviceState_4 \ {s ↦ FAIL_DETECT_4})
    u {s ↦ FAIL_ACTIV_4}
    act2: run_peers(s) = run_peers(s) \ susp_peers(s)
    act3: susp_peers(s) = ∅
    act4: fail_peers = fail_peers u (s)×suspeers(s)
  END

FAIL_CONFIGURE: not extended ordinary
  REFINES FAIL_CONFIGURE
  ANY s
  WHERE
    grd1: s ∈ SERVICES not theorem
    grd2: s ↦ FAIL_ACTIV_4 ∈ serviceState_4 not theorem
    grd3: card(run_peers(s)) < min_inst(s) not theorem
  THEN
    act1: serviceState_4 ≔ (serviceState_4 \ {s ↦ FAIL_ACTIV_4}) \{s ↦ FAIL_CONFIG_4}
  END

FAIL_IGNORE: not extended ordinary
  REFINES FAIL_IGNORE
  ANY s
  WHERE
    grd1: s ∈ SERVICES not theorem
    grd2: s ↦ FAIL_ACTIV_4 ∈ serviceState_4 not theorem
    grd3: card(run_peers(s)) ≥ min_inst(s) not theorem
  THEN
    act1: serviceState_4 ≔ (serviceState_4 \ {s ↦ FAIL_ACTIV_4}) \{s ↦ FAIL_IGN_4}
  END

IGNORE: extended ordinary
REFINES
  IGNORE
  ANY
    s  >
WHERE
  grd1: s ∈ SERVICES not theorem  >
  grd2: s ↦ FAIL_IGN_4 ∈ serviceState_4 not theorem  >
THEN
  act1: serviceState_4 = (serviceState_4 \ {s ↦ FAIL_IGN_4}) ∪ {s ↦ RUN_4}  >
END

REDEPLOY: not extended ordinary  >
REFINES
  REDEPLOY
  ANY
    s  >
new_inst  >
WHERE
  grd1: s ∈ SERVICES not theorem  >
  grd2: s ↦ FAIL_CONFIG_4 ∈ serviceState_4 not theorem  >
  grd3: new_inst ⊆ PEERS not theorem  >
  grd5: new_inst ≠ ∅ not theorem  >
  grd6: finite(new_inst) not theorem  >
  grd7: run_peers(s) ∩ new_inst = ∅ not theorem  >
  grd8: fail_peers[{s}] ∩ new_inst = ∅ not theorem  >
  grd4: card(run_peers(s))+card(new_inst) ≥ min_inst(s) not theorem  >
WITH
  new_run: new_run=card(run_peers(s))+card(new_inst)  >
THEN
  act1: serviceState_4 = (serviceState_4 \ {s ↦ FAIL_CONFIG_4}) ∪ {s ↦ DPL_4}  >
  act2: run_peers(s) = run_peers(s) ∪ new_inst  >
END

HEAL: extended ordinary  >
REFINES
  HEAL
  ANY
    s  >
WHERE
  grd1: s ∈ SERVICES not theorem  >
  grd2: s ↦ DPL_4 ∈ serviceState_4 not theorem  >
THEN
  act1: serviceState_4 = (serviceState_4 \ {s ↦ DPL_4}) ∪ {s ↦ RUN_4}  >
END
UNFAIL_PEER:  not extended ordinary

ANY
s
p

WHERE
grd1:  s ∈ SERVICES not theorem
grd2:  p ∈ PEERS not theorem
grd3:  s ↦ p ∈ fail_peers not theorem

THEN
act1:  fail_peers = fail_peers\{s ↦ p\}

END

END
MACHINE
M07
REFINES
M06
SEES
C07
VARIABLES
serviceState_4  
run_peers  
susp_peers  
fail_peers  
dep_inst  

INVARINTS
inv1: dep_inst ∈ SERVICES ↔ PEERS not theorem  
inv2: ∀ s · s ∈ SERVICES ⇒ dep_inst[{s}] ∩ fail_peers[{s}] = ∅ not theorem  
inv3: ∀ s, st · s ∈ SERVICES ∧ st ∈ STATES_4 ∧ s ↦ st ∈ serviceState_4 
∧ st ≠ FAIL_CONFIG_4 ⇒ dep_inst[{s}] = ∅ not theorem  
inv4: ∀ s · s ∈ SERVICES ⇒ finite(dep_inst[{s}]) not theorem  
inv5: ∀ s · s ∈ SERVICES ⇒ dep_inst[{s}] ∩ run_peers(s) = ∅ not theorem  

EVENTS
INITIALISATION: extended ordinary  
THEN
act1: serviceState_4 ≔ InitState_4  
act2: run_peers ≔ InitSrvcPeers  
act3: susp_peers ≔ ∅  
act4: fail_peers ≔ ∅  
act5: dep_inst ≔ ∅  
END

FAIL: extended ordinary  
REFINES
FAIL
ANY
s  
fp  
WHERE
grd1: s ∈ SERVICES not theorem  
grd2: s ↦ RUN_4 ∈ serviceState_4 not theorem  
grd5: fp ≤ PEERS not theorem  
grd3: fp ≠ ∅ not theorem  
grd4: fp ⊂ run_peers(s) not theorem  
THEN
act1: serviceState_4 ≔ (serviceState_4\{s ↦ RUN_4\}) u {s ↦ 
FAIL_4}  
act2: susp_peers(s) ≔ fp  
END
FAIL_DETECT: extended ordinary
REFINES
FAIL_DETECT

ANY
s >
sf >

WHERE
grd1: s ∈ SERVICES not theorem
grd2: s ↦ FAIL_4 ∈ serviceState_4 not theorem
grd5: susp_peers(s) ≠ ∅ not theorem
grd6: sf ⊆ PEERS not theorem
grd7: sf ⊆ susp_peers(s) not theorem

THEN
act1: serviceState_4 = (serviceState_4\{s ↦ FAIL_4\}) ∪ \{s ↦ FAIL_DETECT_4\}
act2: susp_peers(s) = susp_peers(s) \ sf

END

IS_OK: extended ordinary
REFINES
IS_OK

ANY
s >

WHERE
grd1: s ∈ SERVICES not theorem
grd2: s ↦ FAIL_DETECT_4 ∈ serviceState_4 not theorem
grd5: susp_peers(s) = ∅ not theorem

THEN
act1: serviceState_4 = (serviceState_4 \ \{s ↦ FAIL_DETECT_4\})

END

FAIL_ACTIV: extended ordinary
REFINES
FAIL_ACTIV

ANY
s >

WHERE
grd1: s ∈ SERVICES not theorem
grd2: s ↦ FAIL_DETECT_4 ∈ serviceState_4 not theorem
grd5: susp_peers(s) ≠ ∅ not theorem

THEN
act1: serviceState_4 = (serviceState_4 \ \{s ↦ FAIL_DETECT_4\})
act2: run_peers(s) = run_peers(s) \ susp_peers(s)
act3: susp_peers(s) = ∅
act4: fail_peers = fail_peers ∪ (\{s\}×susp_peers(s))

FAIL ACTIV
FAIL\_CONFIGURE:  extended ordinary \>
REFINES  
FAIL\_CONFIGURE  
ANY  
s \>
WHERE  
grd1:  \ s \in \text{SERVICES} \ not \ theorem \>  
grd2:  \ s \mapsto \text{FAIL\_ACTIV}_4 \in \text{serviceState}_4 \ not \ theorem \>  
grd3:  \ \text{card}(\text{run\_peers}(s)) < \text{min\_inst}(s) \ not \ theorem \>
THEN  
act1:  \ \text{serviceState}_4 := (\text{serviceState}_4 \setminus \{s \mapsto \text{FAIL\_ACTIV}_4\}) \cup \{s \mapsto \text{FAIL\_CONFIG}_4\} \>
END  

FAIL\_IGNORE:  extended ordinary \>
REFINES  
FAIL\_IGNORE  
ANY  
s \>
WHERE  
grd1:  \ s \in \text{SERVICES} \ not \ theorem \>  
grd2:  \ s \mapsto \text{FAIL\_ACTIV}_4 \in \text{serviceState}_4 \ not \ theorem \>  
grd3:  \ \text{card}(\text{run\_peers}(s)) \geq \text{min\_inst}(s) \ not \ theorem \>
THEN  
act1:  \ \text{serviceState}_4 := (\text{serviceState}_4 \setminus \{s \mapsto \text{FAIL\_ACTIV}_4\}) \cup \{s \mapsto \text{FAIL\_IGN}_4\} \>
END  

IGNORE:  extended ordinary \>
REFINES  
IGNORE  
ANY  
s \>
WHERE  
grd1:  \ s \in \text{SERVICES} \ not \ theorem \>  
grd2:  \ s \mapsto \text{FAIL\_IGN}_4 \in \text{serviceState}_4 \ not \ theorem \>  
THEN  
act1:  \ \text{serviceState}_4 := (\text{serviceState}_4 \setminus \{s \mapsto \text{FAIL\_IGN}_4\}) \cup \{s \mapsto \text{RUN}_4\} \>
END  

REDEPLOY\_INST :  not extended ordinary \>
ANY  
s \>
 dep \>
WHERE  

Page 3
grd1:  \( s \in \text{SERVICES} \) not theorem
grd2:  \( \text{dep} \subseteq \text{PEERS} \) not theorem
grd3:  \( \text{finite}(\text{dep}) \) not theorem
grd4:  \( \text{dep} \cap \text{run_peers}(s) = \emptyset \) not theorem
grd5:  \( \text{dep} \cap \text{fail_peers}[^s] = \emptyset \) not theorem
grd6:  \( \text{card}(\text{dep}) = \text{deplo}_\text{inst}(s) \) not theorem
grd7:  \( \text{card}(\text{dep}_\text{inst}[^s]) + \text{card}(\text{run_peers}(s)) < \text{min}_\text{inst}(s) \) not theorem
grd8:  \( s \mapsto \text{FAIL\_CONFIG\_4} \in \text{serviceState}_4 \) not theorem

THEN

act1:  \( \text{dep}_\text{inst} \equiv \text{dep}_\text{inst} \cup \{s\times\text{dep} \} \)

END

REDEPLOY:: not extended ordinary
REFINES REDEPLOY
ANY
s
WHERE

grd1:  \( s \in \text{SERVICES} \) not theorem
grd2:  \( s \mapsto \text{FAIL\_CONFIG\_4} \in \text{serviceState}_4 \) not theorem
grd6:  \( \text{dep}_\text{inst}[^s] \neq \emptyset \) not theorem
grd4:  \( \text{card}(\text{run_peers}(s)) + \text{card}(\text{dep}_\text{inst}[^s]) \geq \text{min}_\text{inst}(s) \) not theorem

WITH

new\_inst:  \( \text{new\_inst} = \text{dep}_\text{inst}[^s] \)

THEN

act1:  \( \text{serviceState}_4 = (\text{serviceState}_4 \setminus \{s \mapsto \text{FAIL\_CONFIG\_4}\}) \)
act2:  \( \text{run_peers}(s) = \text{run_peers}(s) \cup \text{dep}_\text{inst}[^s] \)
act3:  \( \text{dep}_\text{inst} = \{s\} \,<\, \text{dep}_\text{inst} \)

END

HEAL:: extended ordinary
REFINES HEAL
ANY
s
WHERE

grd1:  \( s \in \text{SERVICES} \) not theorem
grd2:  \( s \mapsto \text{DPL\_4} \in \text{serviceState}_4 \) not theorem

THEN

act1:  \( \text{serviceState}_4 = (\text{serviceState}_4 \setminus \{s \mapsto \text{DPL\_4}\}) \cup \{s \mapsto \text{RUN\_4}\} \)

END

UNFAIL\_PEER:: extended ordinary
REFINES
M07

UNFAIL_PEEER

ANY
s ∈ SERVICES not theorem
p ∈ PEERS not theorem
THEN
act1: fail_peers = fail_peers\{s ↦ p}\ >
END

END
MACHINE
M08
REFINES
M07
SEES
C08
VARIABLES
serviceState_4
run_peers
susp_peers
fail_peers
dep_inst
token_owner
unav_peers
susp_inst

INvariants
inv1: token_owner ∈ SERVICES → PEERS not theorem
inv2: unav_peers ⊆ PEERS not theorem
inv3: ∀ s · s ∈ SERVICES ⇒ token_owner(s) ∈ run_peers(s) \ unav_peers not theorem
inv4: ∀ s · s ∈ SERVICES ∧ s ∈ dom(susp_peers) ⇒ token_owner(s) ∈ susp_peers(s) not theorem
inv5: susp_inst ∈ PEERS ↔ (SERVICES × PEERS) not theorem
inv6: ∀ ld, s · ld ∈ PEERS ∧ s ∈ SERVICES ∧ s ∈ dom(susp_inst[ld]) ⇒ ld = token_owner(s) not theorem
inv7: ∀ ld, s · ld ∈ PEERS ∧ s ∈ SERVICES ∧ s ∈ dom(susp_inst[ld]) ∧ ld = token_owner(s) ⇒ ld ∈ susp_inst[ld][s] not theorem
inv8: ∀ ld, s · ld ∈ PEERS ∧ s ∈ SERVICES ∧ s ∈ dom(susp_inst[ld]) ∧ ld = token_owner(s) ⇒ susp_inst[ld][s] ⊂ run_peers(s) not theorem
inv9: ∀ ld, s, stt · ld ∈ PEERS ∧ s ∈ SERVICES ∧ stt ∈ STATES_4 ∧ s ↦ stt ∈ serviceState_4 ∧ ld = token_owner(s) ∧ stt ≠ RUN_4 ⇒ susp_inst[ld][s] = ∅ not theorem

EVENTS
INITIALISATION: extended ordinary
THEN
act1: serviceState_4 ≔ InitState_4
act2: run_peers ≔ InitSrvcPeers
act3: susp_peers = ∅
act4: fail_peers = ∅
act5: dep_inst = ∅
act6: token_owner = init_tok
act7: unav_peers = ∅
act8: susp_inst = ∅

END

MAKE_PEER_UNAVAIL: not extended ordinary
ANY
prs

Page 1
E new values for token owner per service if needed

WHERE

grd1: prs ⊆ PEERS not theorem →
grd2: prs ∉ unav_peers not theorem →
grd3: E ∈ SERVICES → PEERS not theorem → new value for token owner per service if needed

grd4: ∀ srv · srv ∈ SERVICES ∧ token_owner(srv) ∉ prs ⇒ E(srv) = token_owner(srv) not theorem → if the token owner of a service srv does not belong to prs, the token owner is not changed

grd5: ∀ srv · srv ∈ SERVICES ∧ token_owner(srv) ∈ prs ∧ srv ∉ dom(susp_peers) ⇒ E(srv) ∈ run_peers(srv) \ (unav_peers ∪ prs ∪ fail_peers {{srv}}) not theorem → if the owner of the token for a service becomes unavailable, and the service is not suspicious,

then a new token owner among available peers is chosen

grd6: ∀ srv · srv ∈ SERVICES ∧ token_owner(srv) ∈ prs ∧ srv ∉ dom(susp_peers) ⇒ E(srv) ∈ run_peers(srv) \ (unav_peers ∪ prs ∪ susp_peers(srv) ∪ fail_peers {{srv}}) not theorem → if the owner of the token for a service becomes unavailable, and the service possess suspicious instances, then a new token owner among available and not suspicious peers is chosen

THEN

act1: unav_peers = unav_peers ∪ prs → the peers in prs become unavailable

act2: token_owner := token_owner ∧ E → new value for token owner per service is given if needed

act3: susp_inst := prs ∧ susp_inst → the peers in prs can not suspect instances anymore

END

SUSPECT_INST: not extended ordinary

ANY

s → a service s

suspend suspended instances

WHERE

grd1: s ∈ SERVICES not theorem →
grd2: susp ⊆ PEERS not theorem →
grd3: susp = run_peers(s) ∩ unav_peers not theorem → instances in susp are suspicious if the peers running them becomes unavailable

grd4: s ∈ dom(susp_inst{token_owner(s)}) not theorem → the member of susp have not yet been suspected for s by the token owner of s

grd5: s ⇒ RUN_4 ∈ serviceState_4 not theorem → the state of s is OK
THEN
  act1: susp_inst := susp_inst ∪ ({token_owner(s)} × ({s} × susp))
  ▶ the members of susp become suspected instances for s by the token owner of s
END

FAIL: not extended ordinary ▶
REFINES FAIL
ANY s ▶
WHERE
  grd1: s ∈ SERVICES not theorem ▶
  grd2: s ↦ RUN 4 ∈ serviceState_4 not theorem ▶
  grd3: susp_inst[{{token_owner(s)}}][{s}] ≠ ∅ not theorem ▶
WITH
  fp: fp = susp_inst[{{token_owner(s)}}][{s}] ▶
THEN
  act1: serviceState_4 = (serviceState_4 \ {s ↦ RUN_4}) ∪ {s ↦ \ FAIL_4} ▶
  act2: susp_peers(s) = susp_inst[{{token_owner(s)}}][{s}] ▶
  act3: susp_inst = susp_inst \ ({s} ↦ ran(susp_inst)) ▶
END

FAIL_DETECT: extended ordinary ▶
REFINES FAIL_DETECT
ANY s, sf ▶
WHERE
  grd1: s ∈ SERVICES not theorem ▶
  grd2: s ↦ FAIL_4 ∈ serviceState_4 not theorem ▶
  grd3: susp_peers(s) ≠ ∅ not theorem ▶
  grd4: sf ⊆ PEERS not theorem ▶
  grd5: sf ⊆ susp_peers(s) not theorem ▶
THEN
  act1: serviceState_4 = (serviceState_4 \ {s ↦ FAIL_4}) ∪ {s ↦ \ FAIL_DETECT_4} ▶
  act2: susp_peers(s) = susp_peers(s) \ sf ▶
END

IS_OK: extended ordinary ▶
REFINES IS_OK
ANY s ▶
WHERE
  grd1: s ∈ SERVICES not theorem ▶
  grd2: s ↦ RUN 4 ∈ serviceState_4 not theorem ▶
  grd3: susp_peers(s) ≠ ∅ not theorem ▶
  grd4: sf ⊆ PEERS not theorem ▶
  grd5: sf ⊆ susp_peers(s) not theorem ▶
THEN
  act1: serviceState_4 = (serviceState_4 \ {s ↦ RUN_4}) ∪ {s ↦ \ RUN_DETECT_4} ▶
  act2: susp_peers(s) = susp_peers(s) \ sf ▶
END
FAIL_ACTIV: extended ordinary
REFINES FAIL_ACTIV
ANY
s →
WHERE
  grd1: s ∈ SERVICES not theorem →
  grd2: s ↦ FAIL_DETECT_4 ∈ serviceState_4 not theorem →
  grd5: susp_peers(s) ≠ ∅ not theorem →
THEN
  act1: serviceState_4 = (serviceState_4 \ {s ↦ FAIL_DETECT_4}) u {s ↦ RUN_4}
END

FAIL_CONFIGURE: extended ordinary
REFINES FAIL_CONFIGURE
ANY
s →
WHERE
  grd1: s ∈ SERVICES not theorem →
  grd2: s ↦ FAIL_ACTIV_4 ∈ serviceState_4 not theorem →
  grd3: card(run_peers(s)) < min_inst(s) not theorem →
THEN
  act1: serviceState_4 = (serviceState_4 \ {s ↦ FAIL_ACTIV_4}) u {s ↦ FAIL_CONFIG_4}
END

FAIL_IGNORE: extended ordinary
REFINES FAIL_IGNORE
ANY
s →
WHERE
  grd1: s ∈ SERVICES not theorem →
  grd2: s ↦ FAIL_ACTIV_4 ∈ serviceState_4 not theorem →
  grd3: card(run_peers(s)) ≥ min_inst(s) not theorem →
THEN

act1: \( \text{serviceState}_4 = (\text{serviceState}_4 \setminus \{s \mapsto \text{FAIL_ACTIVITY}_4\}) \cup \{s \mapsto \text{FAIL_IGNORE}_4\} \) \>
END

**IGNORE**: extended ordinary
> REFINES
> IGNORE
> ANY
> \( s \) >
> WHERE
> \( \text{grd1}: \ s \in \text{SERVICES} \) not theorem
> \( \text{grd2}: \ s \mapsto \text{FAIL_IGNORE}_4 \in \text{serviceState}_4 \) not theorem
> THEN
> act1: \( \text{serviceState}_4 = (\text{serviceState}_4 \setminus \{s \mapsto \text{FAIL_IGNORE}_4\}) \cup \{s \mapsto \text{RUN}_4\} \) \>
END

**REDEPLOY INST**: extended ordinary
> REFINES
> REDEPLOY INST
> ANY
> \( s \) >
> dep >
> WHERE
> \( \text{grd1}: \ s \in \text{SERVICES} \) not theorem
> \( \text{grd2}: \ \text{dep }\subseteq \text{PEERS} \) not theorem
> \( \text{grd3}: \ \text{finite}(\text{dep}) \) not theorem
> \( \text{grd4}: \ \text{dep }\cap \text{run_peers}(s) = \emptyset \) not theorem
> \( \text{grd5}: \ \text{dep }\cap \text{fail_peers}([s]) = \emptyset \) not theorem
> \( \text{grd6}: \ \text{card}(\text{dep}) = \text{deplo INST}(s) \) not theorem
> \( \text{grd7}: \ \text{card}(\text{dep INST}([s])) + \text{card}(\text{run_peers}(s)) < \text{min_inst}(s) \) not theorem
> \( \text{grd8}: \ s \mapsto \text{FAIL_CONFIG}_4 \in \text{serviceState}_4 \) not theorem
> THEN
> act1: \( \text{dep INST} = \text{dep INST} \cup \{s\} \times \text{dep} \) \>
END

**REDEPLOY**: extended ordinary
> REFINES
> REDEPLOY
> ANY
> \( s \) >
> WHERE
> \( \text{grd1}: \ s \in \text{SERVICES} \) not theorem
> \( \text{grd2}: \ s \mapsto \text{FAIL_CONFIG}_4 \in \text{serviceState}_4 \) not theorem
> \( \text{grd6}: \ \text{dep INST}([s]) \neq \emptyset \) not theorem
> \( \text{grd4}: \ \text{card}(\text{run_peers}(s)) + \text{card}(\text{dep INST}([s])) \geq \text{min INST}(s) \) not theorem
THEN
  act1: serviceState_4 = (serviceState_4 \ {s ↦ FAIL_CONFIG_4}) ∪ {s ↦ DPL_4}
  act2: run_peers(s) = run_peers(s) ∪ dep_inst[{s}]
  act3: dep_inst = {s} ← dep_inst
END

HEAL: extended ordinary
  REFINES HEAL
  ANY s
    WHERE
      grd1: s ∈ SERVICES not theorem
      grd2: s ↦ DPL_4 ∈ serviceState_4 not theorem
    THEN
      act1: serviceState_4 = (serviceState_4 \ {s ↦ DPL_4}) ∪ {s ↦ RUN_4}
END

UNFAIL_PEER: extended ordinary
  REFINES UNFAIL_PEER
  ANY s p
    WHERE
      grd1: s ∈ SERVICES not theorem
      grd2: p ∈ PEERS not theorem
      grd3: s ↦ p ∈ fail_peers not theorem
    THEN
      act1: fail_peers = fail_peers \ {s ↦ p}
END

MAKE_PEER_AVAIL: not extended ordinary
  ANY p
    WHERE
      grd1: p ∈ PEERS not theorem
      grd2: p ∈ unav_peers not theorem
    THEN
      act1: unav_peers = unav_peers \ {p}
END

END
MACHINE M09
REFINES M08
SEES C08

VARIABLES
  serviceState_4
  run_peers
  susp_peers
  fail_peers
  dep_inst
  token_owner
  unav_peers
  susp_inst
  rec_inst
  rct_inst

INvariants
  inv1: rec_inst ∈ PEERS ↔ (SERVICES×PEERS) not theorem
  inv2: rct_inst ∈ PEERS ↔ (SERVICES×PEERS) not theorem
  inv3: ∀ ld, s · ld ∈ PEERS ∧ s ∈ SERVICES ∧ rct_inst[ld][s] ≠ ∅ ⇒ rec_inst[ld][s] ≠ ∅ not theorem
  inv4: ∀ ld, s · ld ∈ PEERS ∧ s ∈ SERVICES ∧ rct_inst[ld][s] ≠ ∅ ⇒ rct_inst[ld][s] ⊆ rec_inst[ld][s] not theorem
  inv5: ∀ ld, s · ld ∈ PEERS ∧ s ∈ SERVICES ∧ s ∈ dom(rec_inst[ld]) ⇒ ld = token_owner(s) not theorem
  inv6: ∀ ld, s · ld ∈ PEERS ∧ s ∈ SERVICES ∧ s ∈ dom(rct_inst[ld]) ⇒ ld = token_owner(s) not theorem
  inv7: ∀ ld, s · ld ∈ PEERS ∧ s ∈ SERVICES ∧ s ∈ dom(rct_inst[ld]) ⇒ ld = token_owner(s) not theorem
  inv8: ∀ ld, s · ld ∈ PEERS ∧ s ∈ SERVICES ∧ s ∈ dom(rct_inst[ld]) ∧ ld = token_owner(s) ⇒ ld ∈ rec_inst[ld][s] not theorem
  inv9: dom(rct_inst) ⊆ dom(rec_inst) not theorem
  inv10: ∀ ld, ld ∈ PEERS ∧ ld ∈ dom(rct_inst) ⇒ rct_inst[ld] ⊆ rec_inst[ld] theorem
  inv11: ∀ s · s ∈ SERVICES ∧ s ∈ dom(susp_peers) ⇒ token_owner(s) ≠ susp_peers(s) not theorem

EVENTS

INITIALISATION: extended ordinary

THEN
  act1: serviceState_4 ≔ InitState_4
  act2: run_peers ≔ InitSrvcPeers
  act3: susp_peers ≔ ∅
  act4: fail_peers ≔ ∅
  act5: dep_inst ≔ ∅
  act6: token_owner ≔ init_tok
  act7: unav_peers ≔ ∅
  act8: susp_inst ≔ ∅
MAKE_PEER_UNAVAIL:  extended ordinary
REFINES
MAKE_PEER_UNAVAIL
ANY
prs >
E ➔ new values for token owner per service if needed
WHERE
grd1:  prs ⊆ PEERS not theorem ➔
grd2:  prs ⊈ unav_peers not theorem ➔
grd3:  E ∈ SERVICES ➔ PEERS not theorem ➔ new value for token owner per service if needed
grd4:  ∀ srv · srv ∈ SERVICES ∧ token_owner(srv) ∉ prs ⇒ E(srv) = token_owner(srv) not theorem ➔ If the token owner of a service srv does not belong to prs, the token owner is not changed
grd5:  ∀ srv · srv ∈ SERVICES ∧ token_owner(srv) ∈ prs ∧ srv ∈ dom(susp_peers) ⇒ E(srv) ∈ run_peers(srv)(unav_peers ∪ prs ∪ fail_peers[srv]) not theorem ➔ if the owner of the token for a service becomes unavailable and the service is not suspicious,
then a new token owner among available peers is chosen
grd6:  ∀ srv · srv ∈ SERVICES ∧ token_owner(srv) ∈ prs ∧ srv ∈ dom(susp_peers) ⇒ E(srv) ∈ run_peers(srv)(unav_peers ∪ prs ∪ susp_peers[srv] ∪ fail_peers[srv]) not theorem ➔ if the owner of the token for a service becomes unavailable, and the service possess suspicious instances, then a new token owner among available and not suspicious peers is chosen
THEN
act1:  unav_peers = unav_peers ∪ prs ➔ the peers in prs become unavailable
act2:  token_owner = token_owner  E ➔ new value for token owner per service is given if needed
act3:  susp_inst = prs  susp_inst ➔ the peers in prs can not suspect instances anymore
act4:  rec_inst = prs  rec_inst ➔
act5:  rct_inst = prs  rct_inst ➔
END
SUSPECT_INST:  extended ordinary
REFINES
SUSPECT_INST

ANY
s ➞ a service s
susp ➞ suspicious instances

WHERE
grd1: s ∈ SERVICES not theorem ➞
grd2: susp ∈ PEERS not theorem ➞
grd3: susp = run_peers(s) n unav_peers not theorem ➞ instances

in susp are suspicious if the peers running them becomes unavailable

grd4: s ∈ dom(susp_inst[{token_owner(s)}]) not theorem ➞ the
member of susp have not yet been suspected for s by the token owner of s

grd5: s ↦ RUN_4 ∈ serviceState_4 not theorem ➞ the state of s is OK

THEN
act1: susp_inst = susp_inst u ({token_owner(s)} × ({s} × susp))

the members of susp become suspected instances for s by the token owner of s

END

FAIL: extended ordinary

REFINES
FAIL

ANY
s ➞

WHERE
grd1: s ∈ SERVICES not theorem ➞
grd2: s ↦ RUN_4 ∈ serviceState_4 not theorem ➞
grd3: susp_inst[{token_owner(s)}][{s}] ≠ ø not theorem ➞

THEN
act1: serviceState_4 = (serviceState_4\{s ↦ RUN_4\}) u {s ↦
FAIL_4}

act2: susp_peers(s) = susp_inst[{token_owner(s)}][{s}] ➞
act3: susp_inst = susp_inst ⩥ ({s} ◁ ran(susp_inst))

END

RECONTACT_INST_OK: not extended ordinary

ANY
s ➞ a service s
i ➞ an instance i

WHERE
grd1: s ∈ SERVICES not theorem ➞
grd2: i ∈ PEERS not theorem ➞
grd3: s ↦ FAIL_4 ∈ serviceState_4 not theorem ➞ the state of s is SUSPICIOUS

grd4: susp_peers(s) ≠ ø not theorem ➞ the set of suspicious
peers for s is not empty

grd5: i ∈ susp_peers(s)\unav_peers not theorem ➞ i is a
suspicous instance of s and is available (can be contacted)

grd6: token_owner(s) ↦ (s ↦ i) ∉ rec_inst not theorem ➞ the
RECONTACT_INST_KO: not extended ordinary

RENECESS

FAIL_DETECT: not extended ordinary

REFINES

FAIL_DETECT

ANY

s

WHERE

grd1: s ∈ SERVICES not theorem

grd2: s ↦ FAIL_4 ∈ serviceState_4 not theorem

grd5: susp_peers(s) ≠ ∅ not theorem

WITH

sf: sf=rct_inst[\{token_owner(s)\}][\{s\}]

THEN

FAIL_DETECT

ANY

s

WHERE

grd1: s ∈ SERVICES not theorem

grd2: s ↦ FAIL_4 ∈ serviceState_4 not theorem

grd5: susp_peers(s) ≠ ∅ not theorem

grd8: rec_inst[\{token_owner(s)\}][\{s\}] = susp_peers(s) not theorem

THEN

RENECESS
act1: \( \text{serviceState}_4 \triangleq (\text{serviceState}_4 \setminus \{s \mapsto \text{FAIL}_4\}) \cup \{s \mapsto \text{RUN}_4\} \) \\
act2: \( \text{suspPeers}(s) \triangleq \text{suspPeers}(s) \setminus \text{rctInst}[\{\text{tokenOwner}(s)\}][s] \) \\
act3: \( \text{recInst} = \text{recInst} \setminus \{s : \text{ran(recInst)}\} \) \\
act4: \( \text{rctInst} = \text{rctInst} \setminus \{s : \text{ran(rctInst)}\} \)

END

**IS_OK:** extended ordinary \\
\text{REFINES} \\
**IS_OK** \\
\text{ANY} \\
\text{s} \rightarrow \text{} \\
\text{WHERE} \\
\text{grd1: } s \in \text{SERVICES} \not\text{ theorem} \\
\text{grd2: } s \mapsto \text{FAIL}_4 \in \text{serviceState}_4 \not\text{ theorem} \\
\text{grd5: } \text{suspPeers}(s) = \emptyset \not\text{ theorem} \\
\text{THEN} \\
\text{act1: } \text{serviceState}_4 \triangleq (\text{serviceState}_4 \setminus \{s \mapsto \text{FAIL}_4\}) \cup \{s \mapsto \text{RUN}_4\} \\
\text{END}

**FAIL_ACTIV:** extended ordinary \\
\text{REFINES} \\
**FAIL_ACTIV** \\
\text{ANY} \\
\text{s} \rightarrow \text{} \\
\text{WHERE} \\
\text{grd1: } s \in \text{SERVICES} \not\text{ theorem} \\
\text{grd2: } s \mapsto \text{FAIL}_4 \in \text{serviceState}_4 \not\text{ theorem} \\
\text{grd5: } \text{suspPeers}(s) \neq \emptyset \not\text{ theorem} \\
\text{THEN} \\
\text{act1: } \text{serviceState}_4 \triangleq (\text{serviceState}_4 \setminus \{s \mapsto \text{FAIL}_4\}) \\
\text{act2: } \text{runPeers}(s) = \text{runPeers}(s) \setminus \text{suspPeers}(s) \\
\text{act3: } \text{suspPeers}(s) = \emptyset \\
\text{act4: } \text{failPeers} = \text{failPeers} \cup \{s \times \text{suspPeers}(s)\} \\
\text{END}

**FAIL_CONFIGURE:** extended ordinary \\
\text{REFINES} \\
**FAIL_CONFIGURE** \\
\text{ANY} \\
\text{s} \rightarrow \text{} \\
\text{WHERE} \\
\text{grd1: } s \in \text{SERVICES} \not\text{ theorem} \\
\text{grd2: } s \mapsto \text{FAIL}_4 \in \text{serviceState}_4 \not\text{ theorem} \\
\text{grd3: } \text{card(runPeers(s)) < minInst(s)} \not\text{ theorem} \\
\text{END}
THEN
act1: \( \text{serviceState}_4 = (\text{serviceState}_4 \setminus \{s \mapsto \text{FAIL_ACTIV}_4\}) \cup \{s \mapsto \text{FAIL_CONFIG}_4\} \)
END

\textbf{FAIL_IGNORE}: extended ordinary
\texttt{REFINES FAIL_IGNORE ANY s \} WHERE
\texttt{grd1: s \in SERVICES not theorem}
\texttt{grd2: s \mapsto \text{FAIL_ACTIV}_4 \in \text{serviceState}_4 not theorem}
\texttt{grd3: \text{card} (\text{run_peers}(s)) \geq \text{min_inst}(s) not theorem}

THEN
act1: \( \text{serviceState}_4 = (\text{serviceState}_4 \setminus \{s \mapsto \text{FAIL_ACTIV}_4\}) \cup \{s \mapsto \text{FAIL_IGN}_4\} \)
END

\textbf{IGNORE}: extended ordinary
\texttt{REFINES IGNORE ANY s \} WHERE
\texttt{grd1: s \in SERVICES not theorem}
\texttt{grd2: s \mapsto \text{FAIL_IGN}_4 \in \text{serviceState}_4 not theorem}

THEN
act1: \( \text{serviceState}_4 = (\text{serviceState}_4 \setminus \{s \mapsto \text{FAIL_IGN}_4\}) \cup \{s \mapsto \text{RUN}_4\} \)
END

\textbf{REDEPLOYINST}: extended ordinary
\texttt{REFINES REDEPLOYINST ANY s \} dep \} WHERE
\texttt{grd1: s \in SERVICES not theorem}
\texttt{grd2: \text{dep} \subseteq \text{PEERS} not theorem}
\texttt{grd3: \text{finite}(\text{dep}) not theorem}
\texttt{grd4: \text{dep} \cap \text{run_peers}(s) = \emptyset not theorem}
\texttt{grd5: \text{dep} \cap \text{fail_peers}([s]) = \emptyset not theorem}
\texttt{grd6: \text{card}(\text{dep}) = \text{deplo Inst}(s) not theorem}
\texttt{grd7: \text{card}(\text{depInst}([s])) + \text{card}(\text{run_peers}(s)) < \text{min_inst}(s)}
\texttt{not theorem}
\texttt{grd8: s \mapsto \text{FAIL_CONFIG}_4 \in \text{serviceState}_4 not theorem}
THEN
  act1: \( \text{dep}_\text{inst} = \text{dep}_\text{inst} \cup \{s\} \times \text{dep} \) >
END

REDEPLOY: extended ordinary >
  REFINES REDEPLOY
  ANY
    s >
  WHERE
    grd1: \( s \in \text{SERVICES} \) not theorem >
    grd2: \( s \mapsto \text{FAIL\_CONFIG\_4} \in \text{serviceState\_4} \) not theorem >
    grd6: \( \text{dep}_\text{inst}[\{s\}] \not= \emptyset \) not theorem >
    grd4: \( \text{card} (\text{run\_peers}(s)) + \text{card} (\text{dep}_\text{inst}[\{s\}]) \geq \text{min\_inst}(s) \) not theorem >
THEN
  act1: \( \text{serviceState\_4} = (\text{serviceState\_4} \setminus \{s \mapsto \text{FAIL\_CONFIG\_4}\}) \) u {s \mapsto DPL\_4} >
  act2: \( \text{run\_peers}(s) = \text{run\_peers}(s) \cup \text{dep}_\text{inst}[\{s\}] \) >
  act3: \( \text{dep}_\text{inst} = \{s\} \setminus \text{dep}_\text{inst} \)>
END

HEAL: extended ordinary >
  REFINES HEAL
  ANY
    s >
  WHERE
    grd1: \( s \in \text{SERVICES} \) not theorem >
    grd2: \( s \mapsto DPL\_4 \in \text{serviceState\_4} \) not theorem >
    grd3: \( \text{run\_peers}(s) = \text{run\_peers}(s) \cup \text{dep}_\text{inst}[\{s\}] \) >
    grd4: \( \text{card} (\text{run\_peers}(s)) + \text{card} (\text{dep}_\text{inst}[\{s\}]) \geq \text{min\_inst}(s) \) not theorem >
 THEN
  act1: \( \text{serviceState\_4} = (\text{serviceState\_4} \setminus \{s \mapsto DPL\_4\}) \cup \{s \mapsto \text{RUN\_4}\} \) >
END

UNFAIL\_PEER: extended ordinary >
  REFINES UNFAIL\_PEER
  ANY
    s >
    p >
  WHERE
    grd1: \( s \in \text{SERVICES} \) not theorem >
    grd2: \( p \in \text{PEERS} \) not theorem >
    grd3: \( s \mapsto p \in \text{fail\_peers} \) not theorem >
 THEN
  act1: \( \text{fail\_peers} = \text{fail\_peers}\setminus\{s \mapsto p\} \) >
END
MAKE_PEER_AVAIL: extended ordinary
REFINES
  MAKE_PEER_AVAIL
ANY
  p
WHERE
  grd1: p ∈ PEERS not theorem
  grd2: p ∈ unav_peers not theorem
THEN
  act1: unav_peers = unav_peers \ {p}
END
END
MACHINE M10 →
REFINES M09
SEES C08
VARIABLES
  serviceState_4 →
  run_peers →
  susp_peers →
  fail_peers →
  dep_inst →
  token_owner →
  unav_peers →
  susp_inst →
  rec_inst → instances that are tried to be recontacted
  rct_inst → instances effectively recontacted after a try
  actv_inst → instances activated by token owners

INVARIANTS
  inv1: actv_inst ∈ PEERS ↔ (SERVICES×PEERS) not theorem →
  inv2: ∀ s, i · s ∈ SERVICES ∧ i ∈ PEERS ⇒ finite(actv_inst[i][s])
    not theorem →
  inv3: ∀ ld, s · ld ∈ PEERS ∧ s ∈ SERVICES ∧ s ∈ dom(actv_inst[ld])
    ⇒ ld = token_owner(s) not theorem →
  inv4: ∀ s, i · s ∈ SERVICES ∧ i ∈ PEERS ⇒ actv_inst[i][s] ∩
    run_peers(s) = ∅ not theorem →
  inv5: ∀ s, i · s ∈ SERVICES ∧ i ∈ PEERS ⇒ actv_inst[i][s] ∩
    dep_inst[s] = ∅ not theorem →
  inv6: ∀ s, i · s ∈ SERVICES ∧ i ∈ PEERS ⇒ actv_inst[i][s] ∩
    fail_peers[s] = ∅ not theorem →
  inv7: ∀ ld, s, stt · ld ∈ PEERS ∧ s ∈ SERVICES ∧ stt ∈ STATES_4 ∧ s ↦
    stt ∈ serviceState_4 ∧ ld = token_owner(s) ∧ stt ≠ FAIL_CONFIG_4 ⇒ actv_inst
    [ld][s] = ∅ not theorem →
  inv8: finite(actv_inst) not theorem →

EVENTS
  INITIALISATION: extended ordinary →
  THEN
    act1: serviceState_4 ≔ InitState_4 →
    act2: run_peers = InitSrvcPeers →
    act3: susp_peers = ∅ →
    act4: fail_peers = ∅ →
    act5: dep_inst = ∅ →
    act6: token_owner = init_tok →
    act7: unav_peers = ∅ →
    act8: susp_inst = ∅ →
    act10: rec_inst = ∅ →
    act11: rct_inst = ∅ →
    act12: actv_inst = ∅ →
MAKE_PEER_UNAVAIL: extended ordinary

REFINES

MAKE_PEER_UNAVAIL

ANY

prs ›
E ›new values for token owner per service if needed

WHERE

grd1: prs ⊆ PEERS not theorem ›
grd2: prs ∉ unav_peers not theorem ›
grd3: E ∈ SERVICES → PEERS not theorem ›new value for token
owner per service if needed
grd4: ∀ srv · srv ∈ SERVICES ∧ token_owner(srv) ∉ prs ⇒ E
(srv) = token_owner(srv) not theorem ›If the token owner of a service srv does
not belong to prs, the token owner is not changed
grd5: ∀ srv · srv ∈ SERVICES ∧ token_owner(srv) ∈ prs ∧ srv ∉
dom(susp_peers) ⇒ E(srv) ∈ run_peers(srv)\(\text{unav_peers} ∪ \text{prs} ∪ \text{fail_peers}
\{\text{srv}\}) not theorem ›if the owner of the token for a service becomes
unavailable and the service is not suspicious,

then a new token owner among available peers is chosen
grd6: ∀ srv · srv ∈ SERVICES ∧ token_owner(srv) ∈ prs ∧ srv ∉
dom(susp_peers) ⇒ E(srv) ∈ run_peers(srv)\(\text{unav_peers} ∪ \text{prs} ∪ \text{fail_peers}
\{\text{srv}\} ∪ \text{dom}(\text{susp_peers}) ∪ \text{run_peers(srv)}\) not theorem ›if the owner of the token for a service becomes
unavailable, and the service

possess suspicious instances, then a new token owner among available and not

suspicious peers is chosen

THEN

act1: unav_peers = unav_peers ∪ prs ›the peers in prs become
unavailable
act2: token_owner = token_owner ∪ E ›new value for token owner
per service is given if needed
act3: susp_inst = prs ∩ susp_inst ›the peers in prs can not
suspect instances anymore
act4: rec_inst = prs ∩ rec_inst ›
act5: rct_inst = prs ∩ rct_inst ›
act6: actv_inst = prs ∩ actv_inst ›

END

SUSPECT_INST: extended ordinary

REFINES

SUSPECT_INST

Page 2
ANY
  s  >a service s
susp >suspicious instances
WHERE
  grd1:  s ∈ SERVICES not theorem >
  grd2:  susp ⊆ PEERS not theorem >
  grd3:  susp = run_peers(s) n unav_peers not theorem >instances
in susp are suspicious if the peers running them becomes unavailable
  grd4:  s ∉ dom(susp_inst[{token_owner(s)}]) not theorem >the member of susp have not yet been suspected for s by the token owner of s
  grd5:  s ↦ RUN_4 ∈ serviceState_4 not theorem >the state of s is OK

THEN
  act1:  susp_inst = susp_inst u ({token_owner(s)} × (s×susp)) >the members of susp become suspected instances for s by the token owner of s
END

FAIL:  extended ordinary >
REFINES FAIL
ANY
  s  >
WHERE
  grd1:  s ∈ SERVICES not theorem >
  grd2:  s ↦ RUN_4 ∈ serviceState_4 not theorem >
  grd3:  susp_inst[{token_owner(s)}][s] ≠ ∅ not theorem >
THEN
  act1:  serviceState_4 = (serviceState_4∖{s ↦ RUN_4}) u {s ↦ FAIL_4} >
  act2:  susp_peers(s) = susp_inst[{token_owner(s)}][s] >
  act3:  susp_inst = susp_inst ⩥ (s ↦ ran(susp_inst)) >
END

RECONTACT_INST_OK:  extended ordinary >
REFINES RECONTACT_INST_OK
ANY
  s  >a service s
i  >an instance i
WHERE
  grd1:  s ∈ SERVICES not theorem >
  grd2:  i ∈ PEERS not theorem >
  grd3:  s ↦ FAIL_4 ∈ serviceState_4 not theorem >the state of s is SUSPICIOUS
  grd4:  susp_peers(s) ≠ ∅ not theorem >the set of suspicious peers for s is not empty
  grd5:  i ∈ susp_peers(s)\unav_peers not theorem >i is a suspicious instance of s and is available (can be contacted)
M10

\[ \text{grd6: } \, \text{token owner of } \, s \text{ has not yet tried to recontact } \, i \]

\[ \text{not theorem } \rightarrow \text{the token owner of } \, s \text{ has not yet tried to recontact all the suspicious instances of } \, s \]

\[ \text{THEN} \]

\[ \text{act1: } \, \text{rec inst} \, = \, \text{rec inst} \, \cup \{ \text{token owner} \, (s \rightarrow i) \} \]

\[ \text{not theorem } \rightarrow \text{the token owner of } \, s \text{ has tried to recontact } \, i \]

\[ \text{act2: } \, \text{rct inst} \, = \, \text{rct inst} \, \cup \{ \text{token owner} \, (s \rightarrow i) \} \]

\[ i \text{ is recontacted by the token owner of } \, s \text{ successfully} \]

\[ \text{END} \]

\[
\text{RECONTACT_INST_KO: extended ordinary } \rightarrow \\
\quad \text{REFINES RECONTACT_INST_KO} \]

\[ \quad \text{ANY} \]

\[ \text{WHERE} \]

\[ \text{grd1: } \, s \in \, \text{SERVICES not theorem } \rightarrow \]

\[ \text{grd2: } \, i \in \, \text{PEERS not theorem } \rightarrow \]

\[ \text{grd3: } \, s \rightarrow \, \text{FAIL}_4 \in \, \text{serviceState}_4 \text{ not theorem } \rightarrow \text{the state of } \, s \]

\[ \text{is SUSPICIOUS} \]

\[ \text{grd4: } \, \text{susp peers}(s) \neq \emptyset \text{ not theorem } \rightarrow \text{the set of suspicious peers for } \, s \text{ is not empty} \]

\[ \text{grd5: } \, i \in \, \text{susp peers}(s) \cap \text{unav peers} \text{ not theorem } \rightarrow \text{i is a suspicious instance of } \, s \text{ and is unavailable (can not be contacted)} \]

\[ \text{grd6: } \, \text{token owner}(s) \rightarrow (s \rightarrow i) \in \, \text{rec inst} \text{ not theorem } \rightarrow \text{the token owner of } \, s \text{ has not yet tried to recontact } \, i \]

\[ \text{grd7: } \, \text{rec inst} \, \rightarrow \{ \text{token owner}(s) \} \in \, \text{rec inst} \text{ not theorem } \rightarrow \text{the token owner of } \, s \text{ has not yet tried to recontact all the suspicious instances of } \, s \]

\[ \text{THEN} \]

\[ \text{act1: } \, \text{rec inst} \, = \, \text{rec inst} \, \cup \{ \text{token owner} \, (s \rightarrow i) \} \]

\[ \text{not theorem } \rightarrow \text{the token owner of } \, s \text{ has tried to recontact } \, i \]

\[ \text{END} \]

\[
\text{FAIL_DETECT: extended ordinary } \rightarrow \\
\quad \text{REFINES FAIL_DETECT} \]

\[ \quad \text{ANY} \]

\[ \quad \text{WHERE} \]

\[ \text{grd1: } \, s \in \, \text{SERVICES not theorem } \rightarrow \]

\[ \text{grd2: } \, s \rightarrow \, \text{FAIL}_4 \in \, \text{serviceState}_4 \text{ not theorem } \rightarrow \]

\[ \text{grd5: } \, \text{susp peers}(s) \neq \emptyset \text{ not theorem } \rightarrow \]

\[ \text{grd6: } \, \text{rec inst} \, \rightarrow \{ \text{token owner}(s) \} \in \, \text{rec inst} \text{ not theorem } \rightarrow \]

\[ \text{theorem } \rightarrow \]

\[ \text{END} \]
THEN
act1: serviceState_4 := (serviceState_4 \ {s ↦ FAIL_DETECT_4}) ∪ \{s ↦ RUN_4\} \\
act2: susp_peers(s) := susp_peers(s) \ \{s\} \{rct_inst\}[[s]] \\
act3: rec_inst = rec_inst \ \{s\} \{s\} \{ran(rec_inst)\} \\
act4: rct_inst = rct_inst \ \{s\} \{ran(rct_inst)\} \\
END

IS_OK: extended ordinary
REFINES
IS_OK
ANY s >
WHERE
grd1: s ∈ SERVICES not theorem >
grd2: s ↦ FAIL_DETECT_4 ∈ serviceState_4 not theorem >
grd5: susp_peers(s) = ∅ not theorem >
THEN
act1: serviceState_4 := (serviceState_4 \ \{s ↦ FAIL_DETECT_4\}) \\
     \{s ↦ RUN_4\} >
END

FAIL_ACTIV: extended ordinary
REFINES
FAIL_ACTIV
ANY s >
WHERE
grd1: s ∈ SERVICES not theorem >
grd2: s ↦ FAIL_DETECT_4 ∈ serviceState_4 not theorem >
grd5: susp_peers(s) ≠ ∅ not theorem >
THEN
act1: serviceState_4 := (serviceState_4 \ \{s ↦ FAIL_DETECT_4\}) \\
act2: run_peers(s) := run_peers(s) \ susp_peers(s) >
act3: susp_peers(s) = ∅ >
act4: fail_peers = fail_peers ∪ \{s\} \{s\} susp_peers(s) >
END

FAIL_CONFIGURE: extended ordinary
REFINES
FAIL_CONFIGURE
ANY s >
WHERE
grd1: s ∈ SERVICES not theorem >
grd2: s ↦ FAIL_ACTIV_4 ∈ serviceState_4 not theorem >
THEN
act1: serviceState_4 := (serviceState_4 \ \{s ↦ FAIL_DETECT_4\}) \\ act2: run_peers(s) := run_peers(s) \ susp_peers(s) >
act3: susp_peers(s) = ∅ >
act4: fail_peers = fail_peers ∪ \{s\} \{s\} susp_peers(s) >
END
grd3: \[
\text{card}(\text{run}\_\text{peers}(s)) < \text{min}\_\text{inst}(s) \text{ not theorem } \\
\]
THEN
act1: \[
\text{serviceState}_4 := (\text{serviceState}_4 \setminus \{s \mapsto \text{FAIL\_ACTIV}_4\}) \cup \\{s \mapsto \text{FAIL\_CONFIG}_4\} \\
\]
END

**FAIL\_IGNORE:** extended ordinary

REFINES FAIL\_IGNORE

ANY \[
s \gg \\
\]
WHERE
grd1: \[
s \in \text{SERVICES} \text{ not theorem } \\
\]
grd2: \[
s \mapsto \text{FAIL\_ACTIV}_4 \in \text{serviceState}_4 \text{ not theorem } \\
\]
grd3: \[
\text{card}(\text{run}\_\text{peers}(s)) \geq \text{min}\_\text{inst}(s) \text{ not theorem } \\
\]
THEN
act1: \[
\text{serviceState}_4 := (\text{serviceState}_4 \setminus \{s \mapsto \text{FAIL\_ACTIV}_4\}) \cup \\{s \mapsto \text{FAIL\_IGN}_4\} \\
\]
END

**IGNORE:** extended ordinary

REFINES IGNORE

ANY \[
s \gg \\
\]
WHERE
grd1: \[
s \in \text{SERVICES} \text{ not theorem } \\
\]
grd2: \[
s \mapsto \text{FAIL\_IGN}_4 \in \text{serviceState}_4 \text{ not theorem } \\
\]
THEN
act1: \[
\text{serviceState}_4 := (\text{serviceState}_4 \setminus \{s \mapsto \text{FAIL\_IGN}_4\}) \cup \\{s \mapsto \text{RUN}_4\} \\
\]
END

**REDEPLOY\_INSTC:** not extended ordinary

ANY \[
s \gg \text{ a service } s \\
i \gg \text{ an instance } i \\
\]
WHERE
grd1: \[
s \in \text{SERVICES} \text{ not theorem } \\
\]
grd2: \[
i \in \text{PEERS} \text{ not theorem } \\
\]
grd3: \[
i \notin \text{run}\_\text{peers}(s) \cup \text{fail}\_\text{peers}\{s\} \cup \text{unav}\_\text{peers} \cup \text{dep}\_\text{inst}\{s\} \text{ not theorem } \rightarrow \text{i does not run } s, \text{ is not failed for } s, \text{ is not unavailable and is not already activated for } s \\
\]
grd4: \[
token\_\text{owner}(s) \mapsto (s \mapsto i) \notin \text{actv}\_\text{inst} \text{ not theorem } \\
\]
grd5: \[
s \mapsto \text{FAIL\_CONFIG}_4 \in \text{serviceState}_4 \text{ not theorem } \\
\]
grd6: \[
\text{card}(\text{actv}\_\text{inst}\{\{\text{token}\_\text{owner}(s)\}\}\{\{s\}\}) < \text{deplo}\_\text{inst}(s) \text{ not theorem } \\
\]
grd7: \[
\text{card}(\text{dep}\_\text{inst}\{\{s\}\}) + \text{card}(\text{run}\_\text{peers}(s)) < \text{min}\_\text{inst}(s) \\
\]
not theorem >
THEN
  act1: `actv_inst = actv_inst \cup \{\text{token\_owner}(s) \mapsto (s \mapsto i)\}\}`
END

**REDEPLOY\_INSTS**: not extended ordinary >
**REFINES**
  **REDEPLOY\_INST**
  **ANY**
  s >
  WHERE
  grd1: `s \in \text{SERVICES} not theorem` >
  grd6: `\text{card}(\text{actv\_inst}[\{\text{token\_owner}(s)\}][s]) = \text{deplo\_inst}(s)`
  not theorem >
  grd7: `\text{card}(\text{dep\_inst}[s]) + \text{card}(\text{run\_peers}(s)) < \text{min\_inst}(s)`
  not theorem >
  grd8: `s \mapsto \text{FAIL\_CONFIG\_4} \in \text{serviceState\_4} not theorem` >
  WITH
  dep: `\text{dep} = \text{actv\_inst}[\{\text{token\_owner}(s)\}][s]` >
  THEN
  act1: `\text{dep\_inst} = \text{dep\_inst} \cup \{s\} \times \text{actv\_inst}[\{\text{token\_owner}(s)\}][s]` >
  act2: `\text{actv\_inst} = \text{actv\_inst} \mapsto (s \mapsto \text{ran}(\text{actv\_inst}))` >
END

**REDEPLOY**: not extended ordinary >
**REFINES**
  **REDEPLOY**
  **ANY**
  s >
  WHERE
  grd1: `s \in \text{SERVICES} not theorem` >
  grd2: `s \mapsto \text{FAIL\_CONFIG\_4} \in \text{serviceState\_4} not theorem` >
  grd7: `\text{actv\_inst}[\{\text{token\_owner}(s)\}][s] = \emptyset` not theorem >
  grd6: `\text{dep\_inst}[s] \neq \emptyset` not theorem >
  grd4: `\text{card}(\text{run\_peers}(s)) + \text{card}(\text{dep\_inst}[s]) \geq \text{min\_inst}(s)` not theorem >
  THEN
  act1: `\text{serviceState\_4} = (\text{serviceState\_4} \setminus \{s \mapsto \text{FAIL\_CONFIG\_4}\})` >
  act2: `\text{run\_peers}(s) = \text{run\_peers}(s) \cup \text{dep\_inst}[s]` >
  act3: `\text{dep\_inst} = \{s\} \setminus \text{dep\_inst}` >
END

**HEAL**: extended ordinary >
**REFINES**
  **HEAL**
  **ANY**
M10

\begin{align*}
\text{WHERE} & \quad s \quad > \\
\text{grd1: } & \quad s \in \text{SERVICES} \text{ not theorem } > \\
\text{grd2: } & \quad s \mapsto \text{DPL}_4 \in \text{serviceState}_4 \text{ not theorem } > \\
\text{THEN} & \\
\text{act1: } & \quad \text{serviceState}_4 \leftarrow (\text{serviceState}_4 \setminus \{s \mapsto \text{DPL}_4\}) \cup \{s \mapsto \text{RUN}_4\} > \\
\end{align*}

\begin{align*}
\text{UNFAIL_PEER: } & \quad \text{extended ordinary } > \\
\text{REFINES} & \\
\text{UNFAIL_PEER} & \\
\text{ANY} & \\
\text{s } & \quad > \\
\text{p } & \quad > \\
\text{WHERE} & \\
\text{grd1: } & \quad s \in \text{SERVICES} \text{ not theorem } > \\
\text{grd2: } & \quad p \in \text{PEERS} \text{ not theorem } > \\
\text{grd3: } & \quad s \mapsto p \in \text{fail_peers} \text{ not theorem } > \\
\text{THEN} & \\
\text{act1: } & \quad \text{fail_peers} \leftarrow \text{fail_peers}{\setminus}\{s \mapsto p\} > \\
\end{align*}

\begin{align*}
\text{MAKE_PEER_AVAIL: } & \quad \text{extended ordinary } > \\
\text{REFINES} & \\
\text{MAKE_PEER_AVAIL} & \\
\text{ANY} & \\
\text{p } & \quad > \\
\text{WHERE} & \\
\text{grd1: } & \quad p \in \text{PEERS} \text{ not theorem } > \\
\text{grd2: } & \quad p \in \text{unav_peers} \text{ not theorem } > \\
\text{THEN} & \\
\text{act1: } & \quad \text{unav_peers} \leftarrow \text{unav_peers}{\setminus}\{p\} > \\
\end{align*}

\text{END}
M11

MACHINE M11
REFINES M10
SEES C08

VARIABLES
run_peers
susp_peers
fail_peers
dep_inst
token_owner
unav_peers
sus Inst
rec_inst
rct_inst
actv_inst
instances that are tried to be recontacted
instances effectively recontacted after a try
instances activated by token owners

i_state

INVARIANTS
inv1: i_state ∈ (PEERS × SERVICES) ⇸ STATES_4 not theorem
inv2: ∀ s · s ∈ SERVICES ⇒ token_owner(s) ⇒ s ∈ dom(i_state) not theorem

gluing_state1: ∀ s, stt · s ∈ SERVICES ∧ stt ∈ STATES_4 ∧ s ↦ stt ∈ serviceState_4 ⇒ (token_owner(s) ⇒ stt) ∈ i_state not theorem

gluing_state2: ∀ s, stt · s ∈ SERVICES ∧ stt ∈ STATES_4 ∧ (token_owner(s) ⇒ stt) ∈ serviceState_4 ⇒ s ↦ stt ∈ i_state ⇒ s ↦ stt ∈ serviceState_4 not theorem

inv3: ∀ p, s · p ∈ PEERS ∧ s ∈ SERVICES ∧ (p ↦ s) ∈ dom(i_state) ⇒ p = token_owner(s) not theorem

EVENTS
INITIALISATION: not extended ordinary

THEN
act2: run_peers = InitSrvcPeers
act3: susp_peers = ∅
act4: fail_peers = ∅
act5: dep_inst = ∅
act6: token_owner = init_tok
act7: unav_peers = ∅
act8: susp_inst = ∅
act10: rec_inst = ∅
act11: rct_inst = ∅
act12: actv_inst = ∅
act13: i_state = InitStatus

END

MAKE_PEER_UNAVAIL: not extended ordinary
REFINES MAKE_PEER_UNAVAIL
ANY
M11

prs →
E →new values for token owner per service if needed
i_s →
WHERE

grd1: prs ≤ PEERS not theorem →
grd2: prs ≤ unav_peers not theorem →
grd3: E ∈ SERVICES ⇒ PEERS not theorem →new value for token
owner per service if needed

grd4: i_s ∈ (PEERS×SERVICES) ↔ STATES_4 not theorem →
grd5: ∀ srv · srv ∈ SERVICES ∧ token_owner(srv) ∈ prs ⇒ E
(srv) = token_owner(srv) not theorem »If the token owner of a service srv does
not belong to prs, the token owner is not changed

grd6: ∀ srv · srv ∈ SERVICES ∧ token_owner(srv) ∈ prs ∧ srv ∈
dom(susp_peers) ⇒ E(srv) ∈ run_peers(srv)\{unav_peers u prs u fail_peers
[{srv}]\} not theorem »if the owner of the token for a service becomes
unavailable and the service is not suspicious,

then a new token owner among available peers is chosen

grd7: ∀ srv · srv ∈ SERVICES ∧ token_owner(srv) ∈ prs ∧ srv ∈
dom(susp_peers) ⇒ E(srv) ∈ run_peers(srv)\{unav_peers u prs u fail_peers[{srv}]
 u susp_peers(srv)\} not theorem »if the owner of the token for a service becomes
unavailable, and the service

possess suspicious instances, then a new token owner among available and not

suspicous peers is chosen

grd8: ∀ p, s · p ∈ PEERS ∧ s ∈ SERVICES ∧ p → s ∈ dom(i_s) ⇒
p = E(s) not theorem →
grd9: ∀ srv · srv ∈ SERVICES ⇒ (E(srv) → srv) → i_state
(token_owner(srv) → srv) ∈ i_s not theorem →
THEN

act1: unav_peers = unav_peers u prs »the peers in prs become
unavailable
act2: token_owner = token_owner ≔ E »new value for token owner
per service is given if needed
act3: susp_inst = prs ≔ susp_inst »the peers in prs can not
suspect instances anymore
act4: rec_inst = prs ≔ rec_inst »the peers in prs can not try
to recontact instances anymore
act5: rct_inst = prs ≔ rct_inst »the peers in prs can not
recontact instances anymore
act6: actv_inst = prs ≔ actv_inst
act7: i_state = i_s

END
SUSPECT_INST: not extended ordinary

REFINES
  SUSPECT_INST

ANY
  s  a service s
  susp suspicious instances

WHERE
  grd1:  s ∈ SERVICES not theorem
  grd2:  susp ⊆ PEERS not theorem
  grd3:  susp = run_peers(s) ∩ unav_peers not theorem

instances in susp are suspicious if the peers running them becomes unavailable

in susp have not yet been suspected for s by the token owner of s

state of s is OK

THEN

act1:  susp Inst ≔ susp Inst u ({token_owner(s)} × ({s} × susp))

END

FAIL: not extended ordinary

REFINES
  FAIL

ANY
  s

WHERE
  grd1:  s ∈ SERVICES not theorem
  grd2:  i_state(token_owner(s) ↦ s) = RUN_4 not theorem
  grd3:  susp Inst[{{token_owner(s)}}][{s}] ≠ ∅ not theorem

THEN

act1:  i_state(token_owner(s) ↦ s) = FAIL_4
act2:  susp peers(s) = susp Inst[{{token_owner(s)}}][{s}]
act3:  susp Inst ≔ susp Inst ⩥ ({s} ◁ ran(susp Inst))

END

RECONTACT_INST_OK: not extended ordinary

REFINES
  RECONTACT_INST_OK

ANY
  s  a service s
  i  an instance i

WHERE
  grd1:  s ∈ SERVICES not theorem
  grd2:  i ∈ PEERS not theorem
  grd3:  i_state(token_owner(s) ↦ s) = FAIL_4 not theorem

state of s is SUSPICIOUS

peers for s is not empty

Page 3
M11

\[ \text{grd5: } \ i \in \text{susp_peers}(s) \setminus \text{unav_peers} \text{ not theorem } \Rightarrow \text{i is a suspicious instance of s and is available (can be contacted)} \]
\[ \text{grd6: } \text{token_owner}(s) \mapsto (s \mapsto i) \notin \text{rec_inst} \text{ not theorem } \Rightarrow \text{the token owner of s has not yet tried to recontact i} \]
\[ \text{grd7: } \text{rec_inst}[[\text{token_owner}(s)]][[s]] \subseteq \text{susp_peers}(s) \text{ not theorem } \Rightarrow \text{the token owner of s has not yet tried to recontact all the suspicious instances of s} \]

\text{THEN}

\[ \text{act1: } \text{rec_inst} \mapsto \text{rec_inst} \cup \{\text{token_owner}(s) \mapsto (s \mapsto i)\} \Rightarrow \text{the token owner of s has tried to recontact i} \]
\[ \text{act2: } \text{rct_inst} \mapsto \text{rct_inst} \cup \{\text{token_owner}(s) \mapsto (s \mapsto i)\} \Rightarrow \text{i is recontacted by the token owner of s successfully} \]

\text{END}

\text{RECONTACT_INST_KO: not extended ordinary }$
\text{REFINES}

\text{RECONTACT_INST_KO

ANY}

\text{s} \mapsto \text{a service s}
\text{i} \mapsto \text{an instance i}

\text{WHERE}

\text{grd1: } \text{s} \in \text{SERVICES} \text{ not theorem }$
\text{grd2: } \text{i} \in \text{PEERS} \text{ not theorem }$
\text{grd3: } \text{i_state(token_owner(s) \mapsto s) = FAIL_4} \text{ not theorem }$

\text{state of s is SUSPICIOUS}
\text{grd4: } \text{susp_peers}(s) \neq \emptyset \text{ not theorem }$
\text{grd5: } \text{i} \in \text{susp_peers}(s) \setminus \text{unav_peers} \text{ not theorem }$
\text{grd6: } \text{token_owner}(s) \mapsto (s \mapsto i) \notin \text{rec_inst} \text{ not theorem }$
\text{grd7: } \text{rec_inst}[[\text{token_owner}(s)]][[s]] \subseteq \text{susp_peers}(s) \text{ not theorem }$

\text{THEN}

\text{act1: } \text{rec_inst} \mapsto \text{rec_inst} \cup \{\text{token_owner}(s) \mapsto (s \mapsto i)\} \Rightarrow \text{the token owner of s has tried to recontact i}

\text{END}

\text{FAIL_DETECT: not extended ordinary }$
\text{REFINES}

\text{FAIL_DETECT

ANY}

\text{s} \mapsto$

\text{WHERE}

\text{grd1: } \text{s} \in \text{SERVICES} \text{ not theorem }$
\text{grd2: } \text{i_state(token_owner(s) \mapsto s) = FAIL_4} \text{ not theorem }$
\text{grd5: } \text{susp_peers}(s) \neq \emptyset \text{ not theorem }$
M11

\[ \text{grd8: } \text{rec}_\text{inst}\{\{\text{token}_\text{owner}(s)\}\}[,s]\} = \text{susp}_\text{peers}(s) \text{ not theorem } \]

\[ \text{THEN } \]

\[ \text{act1: } \text{i}_\text{state}(\text{token}_\text{owner}(s) \mapsto s) = \text{FAIL}_\text{DETECT}_4 \]
\[ \text{act2: } \text{susp}_\text{peers}(s) = \text{susp}_\text{peers}(s) \setminus \text{rct}_\text{inst}\{\{\text{token}_\text{owner}(s)\}\}[,s]\} \]

\[ \text{act3: } \text{rec}_\text{inst} = \text{rec}_\text{inst} \cup (\{s\} \times \text{ran(rec}_\text{inst})\} \]
\[ \text{act4: } \text{rct}_\text{inst} = \text{rct}_\text{inst} \cup (\{s\} \times \text{ran(rct}_\text{inst})\} \]

\[ \text{END} \]

\[ \text{IS}_\text{OK: } \text{not extended ordinary } \]

\[ \text{REFINES } \]

\[ \text{IS}_\text{OK} \]

\[ \text{ANY } s \]

\[ \text{WHERE } \]

\[ \text{grd1: } s \in \text{SERVICES} \text{ not theorem } \]
\[ \text{grd2: } \text{i}_\text{state}(\text{token}_\text{owner}(s) \mapsto s) = \text{FAIL}_\text{DETECT}_4 \text{ not theorem } \]

\[ \text{grd5: } \text{susp}_\text{peers}(s) = \emptyset \text{ not theorem } \]

\[ \text{THEN } \]

\[ \text{act1: } \text{i}_\text{state}(\text{token}_\text{owner}(s) \mapsto s) = \text{RUN}_4 \]

\[ \text{END} \]

\[ \text{FAIL}_\text{ACTIV: } \text{not extended ordinary } \]

\[ \text{REFINES } \]

\[ \text{FAIL}_\text{ACTIV} \]

\[ \text{ANY } s \]

\[ \text{WHERE } \]

\[ \text{grd1: } s \in \text{SERVICES} \text{ not theorem } \]
\[ \text{grd2: } \text{i}_\text{state}(\text{token}_\text{owner}(s) \mapsto s) = \text{FAIL}_\text{DETECT}_4 \text{ not theorem } \]

\[ \text{grd5: } \text{susp}_\text{peers}(s) \neq \emptyset \text{ not theorem } \]

\[ \text{THEN } \]

\[ \text{act1: } \text{i}_\text{state}(\text{token}_\text{owner}(s) \mapsto s) = \text{FAIL}_\text{ACTIV}_4 \]
\[ \text{act2: } \text{run}_\text{peers}(s) = \text{run}_\text{peers}(s) \setminus \text{susp}_\text{peers}(s) \]
\[ \text{act3: } \text{susp}_\text{peers}(s) = \emptyset \]
\[ \text{act4: } \text{fail}_\text{peers} = \text{fail}_\text{peers} \cup (\{s\} \times \text{susp}_\text{peers}(s)) \]

\[ \text{END} \]

\[ \text{FAIL}_\text{CONFIGURE: } \text{not extended ordinary } \]

\[ \text{REFINES } \]

\[ \text{FAIL}_\text{CONFIGURE} \]

\[ \text{ANY } s \]

\[ \text{WHERE } \]

\[ \text{grd1: } s \in \text{SERVICES} \text{ not theorem } \]
M11

\[\text{grd2: } \text{i_state(token\_owner(s) ↦ s) = FAIL\_ACTIV\_4 not theorem }\]
\[\text{grd3: } \text{card(run\_peers(s)) < min\_inst(s) not theorem }\]

\[\text{THEN}\]
\[\text{act1: } \text{i_state(token\_owner(s) ↦ s) = FAIL\_CONFIG\_4 }\]

\[\text{END}\]

FAIL\_IGNORE: not extended ordinary

\[\text{REFINES}\]
\[\text{FAIL\_IGNORE}\]
\[\text{ANY}\]
\[\text{s }\]

\[\text{WHERE}\]
\[\text{grd1: } s \in \text{SERVICES not theorem }\]
\[\text{grd2: } \text{i_state(token\_owner(s) ↦ s) = FAIL\_ACTIV\_4 not theorem }\]
\[\text{grd3: } \text{card(run\_peers(s)) \geq min\_inst(s) not theorem }\]

\[\text{THEN}\]
\[\text{act1: } \text{i_state(token\_owner(s) ↦ s) = FAIL\_IGN\_4 }\]

\[\text{END}\]

IGNORE: not extended ordinary

\[\text{REFINES}\]
\[\text{IGNORE}\]
\[\text{ANY}\]
\[\text{s }\]

\[\text{WHERE}\]
\[\text{grd1: } s \in \text{SERVICES not theorem }\]
\[\text{grd2: } \text{i_state(token\_owner(s) ↦ s) = FAIL\_IGN\_4 not theorem }\]

\[\text{THEN}\]
\[\text{act1: } \text{i_state(token\_owner(s) ↦ s) = RUN\_4 }\]

\[\text{END}\]

REDEPLOY\_INSTC: not extended ordinary

\[\text{REFINES}\]
\[\text{REDEPLOY\_INSTC}\]
\[\text{ANY}\]
\[\text{s }\]
\[\text{i}\]

\[\text{WHERE}\]
\[\text{grd1: } s \in \text{SERVICES not theorem }\]
\[\text{grd2: } i \in \text{PEERS not theorem }\]
\[\text{grd3: } i \notin \text{run\_peers(s) }\cup \text{fail\_peers[\{s\}] }\cup \text{unav\_peers }\cup \text{dep\_inst[\{s\}] not theorem }\]
\[\text{i does not run } s, \text{is not failed for } s, \text{is not unavailable and is not already activated for } s\]
\[\text{grd4: } \text{token\_owner(s) ↦ (s ↦ i) }\notin \text{actv\_inst not theorem }\]
\[\text{grd5: } \text{i_state(token\_owner(s) ↦ s) = FAIL\_CONFIG\_4 not theorem }\]
\[\text{grd6: } \text{card(actv\_inst[\{token\_owner(s)\}][\{s\}]) < deplo\_inst(s) not theorem }\]
M11

grd7: \[\text{card}(\text{dep}\_\text{inst}[\{s\}]) + \text{card}(\text{run}\_\text{peers}(s)) < \text{min}\_\text{inst}(s)\]

\text{not theorem}\)

\text{THEN}\)

act1: \[\text{actv}\_\text{inst} = \text{actv}\_\text{inst} \cup \{\text{token}\_\text{owner}(s) \mapsto (s \mapsto i)\}\]

\text{END}\)

REDEPLOY\_\text{INSTS}: \text{not extended ordinary}\)

\text{REFINES}\)

REDEPLOY\_\text{INSTS}\)

ANY\)

\[s\]

\text{WHERE}\)

\begin{align*}
\text{grd1: } & s \in \text{SERVICES} \text{ not theorem} \\
\text{grd6: } & \text{card}(\text{actv}\_\text{inst}[\{\text{token}\_\text{owner}(s)\}][\{s\}]) = \text{deplo}\_\text{inst}(s) \\
\text{not theorem} & \text{grd7: } \text{card}(\text{dep}\_\text{inst}[\{s\}]) + \text{card}(\text{run}\_\text{peers}(s)) < \text{min}\_\text{inst}(s) \\
\text{not theorem} & \text{grd8: } \text{i}\_\text{state}(\text{token}\_\text{owner}(s) \mapsto s) = \text{FAIL}\_\text{CONFIG}\_\text{4} \text{ not theorem}
\end{align*}

\text{THEN}\)

act1: \[\text{dep}\_\text{inst} = \text{dep}\_\text{inst} \cup \{\{s\} \times \text{actv}\_\text{inst}[\{\text{token}\_\text{owner}(s)\}][\{s\}]\}\]

act2: \[\text{actv}\_\text{inst} = \text{actv}\_\text{inst} \cup (\{s\} \times \text{ran}(\text{actv}\_\text{inst}))\]

\text{END}\)

REDEPLOY: \text{not extended ordinary}\)

\text{REFINES}\)

REDEPLOY\)

ANY\)

\[s\]

\text{WHERE}\)

\begin{align*}
\text{grd1: } & s \in \text{SERVICES} \text{ not theorem} \\
\text{grd2: } & \text{i}\_\text{state}(\text{token}\_\text{owner}(s) \mapsto s) = \text{FAIL}\_\text{CONFIG}\_\text{4} \text{ not theorem} \\
\text{not theorem} & \text{grd7: } \text{actv}\_\text{inst}[\{\text{token}\_\text{owner}(s)\}][\{s\}] = \emptyset \text{ not theorem} \\
\text{not theorem} & \text{grd6: } \text{dep}\_\text{inst}[\{s\}] \neq \emptyset \text{ not theorem} \\
\text{not theorem} & \text{grd4: } \text{card}(\text{run}\_\text{peers}(s)) + \text{card}(\text{dep}\_\text{inst}[\{s\}]) \geq \text{min}\_\text{inst}(s) \text{ not theorem}
\end{align*}

\text{THEN}\)

act1: \[\text{i}\_\text{state}(\text{token}\_\text{owner}(s) \mapsto s) = \text{DPL}\_\text{4}\]

act2: \[\text{run}\_\text{peers}(s) = \text{run}\_\text{peers}(s) \cup \text{dep}\_\text{inst}[\{s\}]\]

act3: \[\text{dep}\_\text{inst} = \{s\} \times \text{dep}\_\text{inst}\]

\text{END}\)

HEAL: \text{not extended ordinary}\)

\text{REFINES}\)

HEAL\)

ANY\)

Page 7
M11

\[ s > \]
WHERE
\[ \text{grd1: } s \in \text{SERVICES not theorem} \]
\[ \text{grd2: } i_{\text{state}}(\text{token_owner}(s) \Rightarrow s) = \text{DPL}_4 \text{ not theorem} \]
THEN
\[ \text{act1: } i_{\text{state}}(\text{token_owner}(s) \Rightarrow s) = \text{RUN}_4 \]
END

\text{UNFAIL\_PEER: extended ordinary} >
\text{REFINES}
\quad \text{UNFAIL\_PEER}
\text{ANY}
\quad s >
\quad p >
WHERE
\quad \text{grd1: } s \in \text{SERVICES not theorem} >
\quad \text{grd2: } p \in \text{PEERS not theorem} >
\quad \text{grd3: } s \Rightarrow p \in \text{fail\_peers not theorem} >
THEN
\quad \text{act1: } \text{fail\_peers} = \text{fail\_peers}\\{s \Rightarrow p\} >
END

\text{MAKE\_PEER\_AVAIL: extended ordinary} >
\text{REFINES}
\quad \text{MAKE\_PEER\_AVAIL}
\text{ANY}
\quad p >
WHERE
\quad \text{grd1: } p \in \text{PEERS not theorem} >
\quad \text{grd2: } p \in \text{unav\_peers not theorem} >
THEN
\quad \text{act1: } \text{unav\_peers} = \text{unav\_peers} \setminus \{p\} >
END
END
M12

MACHINEM12→
REFINESM11
SEESC08
VARIABLES
  run_peers→
  suspc_peers→
  fail_peers→
  dep_inst→
  token_owner→
  unav_peers→
  susp_inst→
  rec_inst→instances that are tried to be recontacted
  rct_inst→instances effectively recontacted after a try
  actv_inst→instances activated by token owner
  i_state→
INVARIANTS
  inv1: suspc_peers ∈ (PEERS×SERVICES) → ℙ(PEERS) not theorem→
  inv2: ∀ p, s · p ∈ PEERS ∧ s ∈ SERVICES ∧ (p ↦ s) ∈ dom(suspc_peers)
      ⇒ p = token_owner(s) not theorem→
  inv3: ∀ p, s · p ∈ PEERS ∧ s ∈ SERVICES ∧ p = token_owner(s) ⇒ (p ↦ s) ∈ dom(suspc_peers) not theorem→
  gluing_tok_own1: ∀ s · s ∈ SERVICES ∧ s ∈ dom(suspc_peers) ⇒
                 susp_peers(s) = suspc_peers(token_owner(s) ↦ s) not theorem→
EVENTS
  INITIALISATION: not extended ordinary→
    THEN
      act2: run_peers := InitSrvcPeers→
      act3: suspc_peers := InitSuspPeers→
      act4: fail_peers := Ø→
      act5: dep_inst := Ø→
      act6: token_owner := init_tok→
      act7: unav_peers := Ø→
      act8: susp_inst := Ø→
      act10: rec_inst := Ø→
      act11: rct_inst := Ø→
      act12: actv_inst := Ø→
      act13: i_state := InitStatus→
  END

MAKE_PEER_UNAVAIL: not extended ordinary→
REFINES MAKE_PEER_UNAVAIL
  ANY
    prs→
    E→new values for token owner per service if needed
WHERE

\[ \text{grd1: } \text{prs} \subseteq \text{PEERS} \text{ not theorem } \]
\[ \text{grd2: } \text{prs} \not\subseteq \text{unav_peers} \text{ not theorem } \]
\[ \text{grd3: } \text{E} \in \text{SERVICES} \rightarrow \text{PEERS} \text{ not theorem } \text{ new value for token owner per service if needed } \]
\[ \text{grd4: } i_s \in (\text{PEERS} \times \text{SERVICES}) \leftrightarrow \text{STATES}_4 \text{ not theorem } \]
\[ \text{grd5: } \forall \text{srv} \cdot \text{srv} \in \text{SERVICES} \land \text{token_owner(srv)} \in \text{prs} \Rightarrow \text{E(srv)} = \text{token_owner(srv)} \text{ not theorem } \text{ if the token owner of a service srv does not belong to prs, the token owner is not changed } \]
\[ \text{grd6: } \forall \text{srv} \cdot \text{srv} \in \text{SERVICES} \land \text{token_owner(srv)} \in \text{prs} \land \text{token_owner(srv)} \rightarrow \text{srv} \not\in \text{dom(suspc_peers)} \Rightarrow \text{E(srv)} \in \text{run_peers(srv)} \setminus (\text{unav_peers u prs u fail_peers[{srv}]}) \text{ not theorem } \text{ if the owner of the token for a service becomes unavailable and the service is not suspicious, then a new token owner among available peers is chosen } \]
\[ \text{grd7: } \forall \text{srv} \cdot \text{srv} \in \text{SERVICES} \land \text{token_owner(srv)} \in \text{prs} \land \text{token_owner(srv)} \rightarrow \text{srv} \in \text{dom(suspc_peers)} \Rightarrow \text{E(srv)} \in \text{run_peers(srv)} \setminus (\text{unav_peers u prs u fail_peers[{srv}]} \cup \text{suspc_peers(token_owner(srv) \rightarrow srv)}) \text{ not theorem } \text{ if the owner of the token for a service becomes unavailable, and the service possess suspicious instances, then a new token owner among available and not suspicious peers is chosen } \]
\[ \text{grd8: } \forall p, s \cdot p \in \text{PEERS} \land s \in \text{SERVICES} \land p \rightarrow s \in \text{dom(i_s)} \Rightarrow p = \text{E(s)} \text{ not theorem } \]
\[ \text{grd9: } \forall \text{srv} \cdot \text{srv} \in \text{SERVICES} \Rightarrow (\text{E(srv)} \rightarrow \text{srv}) \rightarrow i_s \text{ not theorem } \]
\[ (\text{token_owner(srv)} \rightarrow \text{srv}) \in i_s \text{ not theorem } \]
\[ \text{grd10: } p_s \in (\text{PEERS} \times \text{SERVICES}) \leftrightarrow \mathbb{P}(\text{PEERS}) \text{ not theorem } \]
\[ \text{grd11: } \forall p, s \cdot p \in \text{PEERS} \land s \in \text{SERVICES} \land p \rightarrow s \in \text{dom(p_s)} \Rightarrow p = \text{E(s)} \text{ not theorem } \]
\[ \text{grd12: } \forall \text{srv} \cdot \text{srv} \in \text{SERVICES} \Rightarrow (\text{E(srv)} \rightarrow \text{srv}) \rightarrow \text{suspc_peers} \text{ not theorem } \]

THEN

\[ \text{act1: } \text{unav_peers} \triangleq \text{unav_peers u prs } \text{ the peers in prs become unavailable } \]
\[ \text{act2: } \text{token_owner} \triangleq \text{token_owner } \rightarrow \text{E } \text{ new value for token owner per service is given if needed } \]
\[ \text{act3: } \text{susp_inst} = \text{prs} \triangleq \text{susp_inst } \text{ the peers in prs can not suspect instances anymore } \]
\[ \text{act4: } \text{rec_inst} = \text{prs} \triangleq \text{rec_inst } \text{ the peers in prs can not try to recontact instances anymore } \]
\[ \text{act5: } \text{rct_inst} = \text{prs} \triangleq \text{rct_inst } \text{ the peers in prs can not recontact instances anymore } \]
act6: \[ \text{actv\_inst} \leftarrow \text{prs} \leftarrow \text{actv\_inst} \]
act7: \[ \text{i\_state} \leftarrow \text{i\_s} \]
act8: \[ \text{suspc\_peers} \leftarrow \text{p\_s} \]

END

**SUSPECT\_INST**: extended ordinary

**REFINES**

**SUSPECT\_INST**

**ANY**

\[ s \rightarrow \text{a service s} \]

\[ \text{susp} \rightarrow \text{suspicious instances} \]

**WHERE**

\[ \text{grd1: } s \in \text{SERVICES} \not \text{ theorem} \]

\[ \text{grd2: } \text{susp} \subseteq \text{PEERS} \not \text{ theorem} \]

\[ \text{grd3: } \text{susp} = \text{run\_peers}(s) \cap \text{unav\_peers} \not \text{ theorem} \]

\text{suspicious instances are instances in susp are suspicious if the peers running them becomes unavailable}

\[ \text{grd4: } s \not \in \text{dom(susp\_inst[\{token\_owner(s)\}])} \not \text{ theorem} \]

\text{the member of susp have not yet been suspected for s by the token owner of s}

\[ \text{grd5: } \text{i\_state(token\_owner(s) \mapsto s) = RUN\_4} \not \text{ theorem} \]

\text{the state of s is OK}

**THEN**

\[ \text{act1: } \text{susp\_inst} = \text{susp\_inst} \cup (\{\text{token\_owner(s)}\} \times (\{s\} \times \text{susp})) \]

\text{the members of susp become suspected instances for s by the token owner of s}

**END**

**FAIL**: not extended ordinary

**REFINES**

**FAIL**

**ANY**

\[ s \rightarrow \]

**WHERE**

\[ \text{grd1: } s \in \text{SERVICES} \not \text{ theorem} \]

\[ \text{grd2: } \text{i\_state(token\_owner(s) \mapsto s) = RUN\_4} \not \text{ theorem} \]

\[ \text{grd3: } \text{susp\_inst[\{token\_owner(s)\}][\{s\}] \neq \emptyset} \not \text{ theorem} \]

**THEN**

\[ \text{act1: } \text{i\_state(token\_owner(s) \mapsto s) = FAIL\_4} \]

\[ \text{act2: } \text{suspc\_peers(token\_owner(s) \mapsto s) = susp\_inst[\{token\_owner(s)\}][\{s\}]} \]

\[ \text{act3: } \text{susp\_inst} = \text{susp\_inst} \leftarrow (\{s\} \leftarrow \text{ran(susp\_inst)}) \]

**END**

**RECONTACT\_INST\_OK**: not extended ordinary

**REFINES**

**RECONTACT\_INST\_OK**

**ANY**

\[ s \rightarrow \text{a service s} \]

\[ i \rightarrow \text{an instance i} \]

**WHERE**
M12

grd1: \ s \in \text{SERVICES} \not \text{theorem} \quad \text{the state of } s \text{ is SUSPICIOUS}

grd2: \ i \in \text{PEERS} \not \text{theorem} \quad \text{the set of suspicious peers for } s \text{ is not empty}

grd3: \ \text{i\textunderscore\text{state}(token\_owner(s) \mapsto s) = FAIL\_4} \not \text{theorem} \quad \text{the token owner of } s \text{ has not yet tried to recontact } i

grd4: \ \text{susp\_peers(token\_owner(s) \mapsto s) \neq \emptyset} \not \text{theorem} \quad \text{i is a suspicious instance of } s \text{ and is available (can be contacted)}

grd5: \ \text{token\_owner(s) \mapsto (s \mapsto i) \notin rec\_inst} \not \text{theorem} \quad \text{the token owner of } s \text{ has not yet tried to recontact } i

THEN

act1: \ \text{rec\_inst = rec\_inst u \{token\_owner(s) \mapsto (s \mapsto i)\}} \not \text{theorem} \quad \text{i is recontacted by the token owner of } s \text{ successfully}

END

RECONTACT\_INST\_KO: \not \text{extended ordinary} \quad \text{REFINES}

RECONTACT\_INST\_KO

ANY

s \to \text{a service } s

i \to \text{an instance } i

WHERE

grd1: \ s \in \text{SERVICES} \not \text{theorem} \quad \text{the state of } s \text{ is SUSPICIOUS}

grd2: \ i \in \text{PEERS} \not \text{theorem} \quad \text{the set of suspicious peers for } s \text{ is not empty}

grd3: \ \text{i\textunderscore\text{state}(token\_owner(s) \mapsto s) = FAIL\_4} \not \text{theorem} \quad \text{the token owner of } s \text{ has not yet tried to recontact } i

grd4: \ \text{susp\_peers(token\_owner(s) \mapsto s) \neq \emptyset} \not \text{theorem} \quad \text{i is a suspicious instance of } s \text{ and is unavailable (can not be contacted)}

grd5: \ \text{token\_owner(s) \mapsto (s \mapsto i) \notin rec\_inst} \not \text{theorem} \quad \text{the token owner of } s \text{ has not yet tried to recontact } i

THEN

act2: \ \text{rct\_inst = rct\_inst u \{token\_owner(s) \mapsto (s \mapsto i)\}} \not \text{theorem} \quad \text{i is recontacted by the token owner of } s \text{ successfully}

END

FAIL\_DETECT: \not \text{extended ordinary} \quad \text{REFINES}

Page 4
FAIL_DETECT

ANY
s

WHERE
grd1: s ∈ SERVICES not theorem
grd2: i_state(token_owner(s) ↦ s) = FAIL_4 not theorem
grd5: suspc_peers(token_owner(s) ↦ s) ≠ ∅ not theorem
grd8: rec_inst[[token_owner(s)]]{{s}} = suspc_peers(token_owner(s) ↦ s) not theorem

THEN
act1: i_state(token_owner(s) ↦ s) ≔ FAIL_DETECT_4
act2: suspc_peers(token_owner(s) ↦ s) ≔ suspc_peers(token_owner(s) ↦ s) \ rct_inst[[token_owner(s)]]{{s}}
act3: rec_inst = rec_inst \ {{s} < ran(rec_inst)}
act4: rct_inst = rct_inst \ {{s} < ran(rct_inst)}

END

IS_OK: not extended ordinary

REFINES
IS_OK

FAIL_ACTIV: not extended ordinary

REFINES
FAIL_ACTIV

FAIL_ACTIV
FAIL_CONFIGURE: extended ordinary
REFINES
FAIL_CONFIGURE
ANY
s
WHERE
  grd1: s ∈ SERVICES not theorem
  grd2: i_state(token_owner(s) → s) = FAIL_ACTIV_4 not theorem
  grd3: card(run_peers(s)) < min_inst(s) not theorem
THEN
  act1: i_state(token_owner(s) → s) = FAIL_CONFIG_4
END

FAIL_IGNORE: extended ordinary
REFINES
FAIL_IGNORE
ANY
s
WHERE
  grd1: s ∈ SERVICES not theorem
  grd2: i_state(token_owner(s) → s) = FAIL_ACTIV_4 not theorem
  grd3: card(run_peers(s)) ≥ min_inst(s) not theorem
THEN
  act1: i_state(token_owner(s) → s) = FAIL_IGN_4
END

IGNORE: extended ordinary
REFINES
IGNORE
ANY
s
WHERE
  grd1: s ∈ SERVICES not theorem
  grd2: i_state(token_owner(s) → s) = FAIL_IGN_4 not theorem
THEN
  act1: i_state(token_owner(s) → s) = RUN_4
END

REDEPLOY_INSTC: extended ordinary
REFINES
REDEPLOY_INSTC
ANY
s → a service s
i → an instance i
WHERE
  grd1: s ∈ SERVICES not theorem
M12

\[ \text{grd2: } i \in \text{PEERS not theorem } \]
\[ \text{grd3: } i \notin \text{run_peers}(s) \cup \text{fail_peers}(s) \cup \text{unav_peers} \cup \]
\[ \text{dep_inst}(s) \text{ not theorem } \] i does not run s, is not failed for s, is not
\[ \text{unavailable and is not already activated for s } \]
\[ \text{grd4: } \text{token_owner}(s) \mapsto (s \mapsto i) \notin \text{actv_inst not theorem } \]
\[ \text{grd5: } i \_\text{state}(\text{token_owner}(s) \mapsto s) = \text{FAIL_CONFIG}_4 \text{ not theorem } \]
\[ \rightarrow \]
\[ \text{grd6: } \text{card}(\text{actv_inst}(s)) < \text{deplo}_\text{inst}(s) \text{ not theorem } \]
\[ \rightarrow \]
\[ \text{grd7: } \text{card}(\text{dep}_\text{inst}(s)) + \text{card}(\text{run}_\text{peers}(s)) < \text{min}_\text{inst}(s) \text{ not theorem } \]
\[ \rightarrow \]
THEN
\[ \text{act1: } \text{actv}_\text{inst} = \text{actv}_\text{inst} \cup \{\text{token_owner}(s) \mapsto (s \mapsto i)\} \rightarrow \]
END

\[ \text{REDEPLOY}_\text{INSTS: } \text{extended ordinary } \rightarrow \]
\[ \text{REFINES} \]
\[ \text{REDEPLOY}_\text{INSTS} \]
\[ \text{ANY} \]
\[ s \rightarrow \]
WHERE
\[ \text{grd1: } s \in \text{SERVICES not theorem } \]
\[ \text{grd6: } \text{card}(\text{actv}_\text{inst}(s)) = \text{deplo}_\text{inst}(s) \text{ not theorem } \]
\[ \rightarrow \]
\[ \text{grd7: } \text{card}(\text{dep}_\text{inst}(s)) + \text{card}(\text{run}_\text{peers}(s)) < \text{min}_\text{inst}(s) \text{ not theorem } \]
\[ \rightarrow \]
\[ \text{grd8: } i \_\text{state}(\text{token_owner}(s) \mapsto s) = \text{FAIL_CONFIG}_4 \text{ not theorem } \]
\[ \rightarrow \]
THEN
\[ \text{act1: } \text{dep}_\text{inst} = \text{dep}_\text{inst} \cup \{s\times\text{actv}_\text{inst}(s)\} \rightarrow \]
\[ \text{act2: } \text{actv}_\text{inst} = \text{actv}_\text{inst} \mapsto \{s \leftarrow \text{ran}(\text{actv}_\text{inst})\} \rightarrow \]
END

\[ \text{REDEPLOY: } \text{extended ordinary } \rightarrow \]
\[ \text{REFINES} \]
\[ \text{REDEPLOY} \]
\[ \text{ANY} \]
\[ s \rightarrow \]
WHERE
\[ \text{grd1: } s \in \text{SERVICES not theorem } \]
\[ \text{grd2: } i \_\text{state}(\text{token_owner}(s) \mapsto s) = \text{FAIL_CONFIG}_4 \text{ not theorem } \]
\[ \rightarrow \]
\[ \text{grd7: } \text{actv}_\text{inst}(s) = \emptyset \text{ not theorem } \]
\[ \text{grd6: } \text{dep}_\text{inst}(s) \neq \emptyset \text{ not theorem } \]
\[ \text{grd4: } \text{card}(\text{run}_\text{peers}(s)) + \text{card}(\text{dep}_\text{inst}(s)) = \text{min}_\text{inst}(s) \text{ not} \]
\[ \text{theorem } \rightarrow \]
THEN

Page 7
M12

act1: \( \text{i\_state(token\_owner(s) \mapsto s) = DPL\_4} \)
act2: \( \text{run\_peers(s) = run\_peers(s) \cup \text{dep\_inst}\{s\}} \)
act3: \( \text{dep\_inst = \{s\} \triangleleft \text{dep\_inst}} \)

END

HEAL: extended ordinary

REFINES

HEAL

ANY

s >

WHERE

grd1: \( s \in \text{SERVICES not theorem} \)
grd2: \( \text{i\_state(token\_owner(s) \mapsto s) = DPL\_4 not theorem} \)

THEN

act1: \( \text{i\_state(token\_owner(s) \mapsto s) = RUN\_4} \)

END

UNFAIL_PEER: extended ordinary

REFINES

UNFAIL_PEER

ANY

s > p >

WHERE

grd1: \( s \in \text{SERVICES not theorem} \)
grd2: \( p \in \text{PEERS not theorem} \)
grd3: \( s \mapsto p \in \text{fail\_peers not theorem} \)

THEN

act1: \( \text{fail\_peers = fail\_peers\{s \mapsto p\}} \)

END

MAKE_PEER_AVAIL: extended ordinary

REFINES

MAKE_PEER_AVAIL

ANY

p >

WHERE

grd1: \( p \in \text{PEERS not theorem} \)
grd2: \( p \in \text{unav\_peers not theorem} \)

THEN

act1: \( \text{unav\_peers = unav\_peers \setminus \{p\}} \)

END

END
MACHINE
M13
REFINES
M12
SEES
C08

VARIABLES
run_peers
suspc_peers
fail_peers
dep_inst
token_owner
unav_peers
suspc_inst
rec_inst
rct_inst
actv_inst

INVARIANTS
inv1: suspc_inst ∈ (PEERS×SERVICES) ⇸ ℙ(PEERS) not theorem
inv2: ∀ p, s · p ∈ PEERS ∧ s ∈ SERVICES ∧ (p ↦ s) ∈ dom(suspc_inst) ⇒ p = token_owner(s) not theorem
inv3: ∀ p, s · p ∈ PEERS ∧ s ∈ SERVICES ∧ p = token_owner(s) ⇒ (p ↦ s) ∈ dom(suspc_inst) not theorem

gluing_tok_own1: ∀ p, s · p ∈ PEERS ∧ s ∈ SERVICES ∧ (p ↦ s) ∈ dom(suspc_inst) ⇒ susp_inst[p][s] = suspc_inst(p ↦ s) not theorem

EVENTS
INITIALISATION:
THEN
act2: run_peers ≔ InitSrvcPeers
act3: suspc_peers ≔ InitSuspPeers
act4: fail_peers ≔ ∅
act5: dep_inst ≔ ∅
act6: token_owner ≔ init_tok
act7: unav_peers ≔ ∅
act8: suspc_inst ≔ InitSuspPeers
act10: rec_inst ≔ ∅
act11: rct_inst ≔ ∅
act12: actv_inst ≔ ∅
act13: i_state ≔ InitStatus

END

MAKE_PEER_UNAVAIL: not extended ordinary
REFINES
MAKE_PEER_UNAVAIL
ANY
prs
E

new values for token owner per service if needed
WHERE

grd1: \( prs \subseteq PEERS \) not theorem
grd2: \( prs \notin unav_peers \) not theorem
grd3: \( E \in SERVICES \rightarrow PEERS \) not theorem
new value for token owner per service if needed
grd4: \( i_s \in (PEERS \times SERVICES) \leftrightarrow STATES_4 \) not theorem
grd5: \( \forall srv \cdot srv \in SERVICES \land token_owner(srv) \in prs \Rightarrow E(srv) = token_owner(srv) \) not theorem
If the token owner of a service srv does not belong to prs, the token owner is not changed
grd6: \( \forall srv \cdot srv \in SERVICES \land token_owner(srv) \in prs \land token_owner(srv) \rightarrow srv \notin dom(suspc_peers) \Rightarrow E(srv) \in run_peers(srv) \setminus (unav_peers u prs u fail_peers[{srv}]) \) not theorem
if the owner of the token for a service becomes unavailable and the service is not suspicious,

then a new token owner among available peers is chosen
grd7: \( \forall srv \cdot srv \in SERVICES \land token_owner(srv) \in prs \land token_owner(srv) \rightarrow srv \in dom(suspc_peers) \Rightarrow E(srv) \in run_peers(srv) \setminus (unav_peers u prs u fail_peers[{srv}] \cup suspc_peers(token_owner(srv) \rightarrow srv)) \) not theorem
if the owner of the token for a service becomes unavailable, and the service possess suspicious instances, then a new token owner among available and not suspicious peers is chosen
grd8: \( \forall p, s \cdot p \in PEERS \land s \in SERVICES \land p \rightarrow s \in dom(i_s) \Rightarrow p = E(s) \) not theorem
grd9: \( \forall srv \cdot srv \in SERVICES \Rightarrow (E(srv) \rightarrow srv) \rightarrow i_state \)
(grd10: \( (token_owner(srv) \rightarrow srv) \in i_s \) not theorem)
grd11: \( \forall p, s \cdot p \in PEERS \land s \in SERVICES \land p \rightarrow s \in dom(p_s) \Rightarrow p = E(s) \) not theorem
grd12: \( \forall srv \cdot srv \in SERVICES \Rightarrow (E(srv) \rightarrow srv) \rightarrow suspc_peers \)
(grd13: \( (token_owner(srv) \rightarrow srv) \in p_s \) not theorem)
grd14: \( \forall p, s \cdot p \in PEERS \land s \in SERVICES \land p \rightarrow s \in dom(s_i) \Rightarrow p = E(s) \) not theorem
grd15: \( \forall srv \cdot srv \in SERVICES \land token_owner(srv) \notin prs \Rightarrow (E(srv) \rightarrow srv) \rightarrow suspc_inst(E(srv) \rightarrow srv) \in s_i \) not theorem
grd16: \( \forall srv \cdot srv \in SERVICES \land token_owner(srv) \in prs \Rightarrow (E(srv) \rightarrow srv) \rightarrow \emptyset \in s_i \) not theorem

THEN

act1: \( unav_peers = unav_peers \cup prs \rightarrow \) the peers in prs become unavailable
act2: \( \text{token} \_\text{owner} \leftarrow \text{token} \_\text{owner} \leftarrow \text{E} \) \( \triangleright \) new value for \( \text{token} \_\text{owner} \)

per service is given if needed

act3: \( \text{rec} \_\text{inst} \leftarrow \text{prs} \triangleleft \text{rec} \_\text{inst} \) \( \triangleright \) the peers in \( \text{prs} \) can not try

to recontact instances anymore

act4: \( \text{rct} \_\text{inst} \leftarrow \text{prs} \triangleleft \text{rct} \_\text{inst} \) \( \triangleright \) the peers in \( \text{prs} \) can not

recontact instances anymore

act5: \( \text{actv} \_\text{inst} \leftarrow \text{prs} \triangleleft \text{actv} \_\text{inst} \) 

act6: \( \text{i} \_\text{state} \leftarrow \text{i} \_\text{s} \)

act7: \( \text{susp} \_\text{peers} \leftarrow \text{p} \_\text{s} \)

act8: \( \text{susp} \_\text{inst} \leftarrow \text{s} \_\text{i} \)

END

SUPECT_INST: not extended ordinary \( \triangleright \)

REFINES

SUPECT_INST

ANY

s \( \triangleright \) a service \( s \)

susp \( \triangleright \) suspicious instances

WHERE

grd1: \( s \in \text{SERVICES} \) not theorem \( \triangleright \)

grd2: \( \text{susp} \subseteq \text{PEERS} \) not theorem \( \triangleright \)

grd3: \( \text{susp} = \text{run} \_\text{peers}(s) \cap \text{unav} \_\text{peers} \) not theorem \( \triangleright \) instances

in \( \text{susp} \) are suspicious if the peers running them becomes unavailable

grd4: \( \text{susp} \_\text{inst}(\text{token} \_\text{owner}(s) \Rightarrow s) = \emptyset \) not theorem \( \triangleright \) the

member of \( \text{susp} \) have not yet been suspected for \( s \) by the \text{token} \_\text{owner} \ of \( s \)

grd5: \( \text{i} \_\text{state}(\text{token} \_\text{owner}(s) \Rightarrow s) = \text{RUN} \_\text{4} \) not theorem \( \triangleright \) the

state of \( s \) is OK

grd6: \( \text{susp} \neq \emptyset \) not theorem \( \triangleright \)

THEN

act1: \( \text{susp} \_\text{inst}(\text{token} \_\text{owner}(s) \Rightarrow s) = \text{susp} \) \( \triangleright \) the members of

\( \text{susp} \) become suspected instances for \( s \) by the \text{token} \_\text{owner} \ of \( s \)

END

FAIL: not extended ordinary \( \triangleright \)

REFINES

FAIL

ANY

s \( \triangleright \)

WHERE

grd1: \( s \in \text{SERVICES} \) not theorem \( \triangleright \)

grd2: \( \text{i} \_\text{state}(\text{token} \_\text{owner}(s) \Rightarrow s) = \text{RUN} \_\text{4} \) not theorem \( \triangleright \)

grd3: \( \text{susp} \_\text{inst}(\text{token} \_\text{owner}(s) \Rightarrow s) \neq \emptyset \) not theorem \( \triangleright \)

THEN

act1: \( \text{i} \_\text{state}(\text{token} \_\text{owner}(s) \Rightarrow s) = \text{FAIL} \_\text{4} \) \( \triangleright \)

act2: \( \text{susp} \_\text{peers}(\text{token} \_\text{owner}(s) \Rightarrow s) = \text{susp} \_\text{inst}(\text{token} \_\text{owner}

\( s \Rightarrow s \)) \( \triangleright \)

act3: \( \text{susp} \_\text{inst}(\text{token} \_\text{owner}(s) \Rightarrow s) = \emptyset \) \( \triangleright \)

END

Page 3
RECONTACT_INST_OK:  extended ordinary

RECONTACT_INST_OK

ANY
s  >a service s
i  >an instance i

WHERE
  grd1:  s ∈ SERVICES not theorem
  grd2:  i ∈ PEERS not theorem
  grd3:  i_state(token_owner(s) ↦ s) = FAIL_4 not theorem

state of s is SUSPICIOUS
  grd4:  suspc_peers(token_owner(s) ↦ s) ≠ ø not theorem

of suspicious peers for s is not empty
  grd5:  i ∈ suspc_peers(token_owner(s) ↦ s) \ unav_peers not

theorem

i is a suspicious instance of s and is available (can be contacted)
  grd6:  token_owner(s) ↦ (s ↦ i) ∉ rec_inst not theorem

the token owner of s has not yet tried to recontact i
  grd7:  rec_inst[{token_owner(s)}][{s}] ⊂ suspc_peers

THEN

act1:  rec_inst = rec_inst ∪ {token_owner(s) ↦ (s ↦ i)}
act2:  rct_inst = rct_inst ∪ {token_owner(s) ↦ (s ↦ i)}

i is recontacted by the token owner of s successfully

END

RECONTACT_INST_KO:  extended ordinary

RECONTACT_INST_KO

ANY
s  >a service s
i  >an instance i

WHERE
  grd1:  s ∈ SERVICES not theorem
  grd2:  i ∈ PEERS not theorem
  grd3:  i_state(token_owner(s) ↦ s) = FAIL_4 not theorem

state of s is SUSPICIOUS
  grd4:  suspc_peers(token_owner(s) ↦ s) ≠ ø not theorem

of suspicious peers for s is not empty
  grd5:  i ∈ suspc_peers(token_owner(s) ↦ s) \ unav_peers not

theorem

i is a suspicious instance of s and is unavailable (can not be contacted)
  grd6:  token_owner(s) ↦ (s ↦ i) ∉ rec_inst not theorem

the token owner of s has not yet tried to recontact i
  grd7:  rec_inst[{token_owner(s)}][{s}] ⊂ suspc_peers

THEN

END
recontact all the suspicious instances of s
THEN
act1: rec_inst = rec_inst ∪ \{token_owner(s) ↦ (s ↦ i)\} ›the
token owner of s has tried to recontact i
END

FAIL_DETECT: extended ordinary ›
REFINES
FAIL_DETECT
ANY
s ›
WHERE
grd1: s ∈ SERVICES not theorem ›
grd2: i_state(token_owner(s) ↦ s) = FAIL_4 not theorem ›
grd5: suspc_peers(token_owner(s) ↦ s) ≠ ∅ not theorem ›
grd8: rec_inst\{token_owner(s)\}\{s\} = suspc_peers
(token_owner(s) ↦ s) not theorem ›
THEN
act1: i_state(token_owner(s) ↦ s) = FAIL_DETECT_4 ›
act2: suspc_peers(token_owner(s) ↦ s) = suspc_peers
(token_owner(s) ↦ s) \ rct_inst\{token_owner(s)\}\{s\} ›
act3: rec_inst = rec_inst ∪ \{{s} < ran(rec_inst)\} ›
act4: rct_inst = rct_inst ∪ \{{s} < ran(rct_inst)\} ›
END

IS_OK: extended ordinary ›
REFINES
IS_OK
ANY
s ›
WHERE
grd1: s ∈ SERVICES not theorem ›
grd2: i_state(token_owner(s) ↦ s) = FAIL_DETECT_4 not theorem ›
grd5: suspc_peers(token_owner(s) ↦ s) = ∅ not theorem ›
THEN
act1: i_state(token_owner(s) ↦ s) = RUN_4 ›
END

FAIL_ACTIV: extended ordinary ›
REFINES
FAIL_ACTIV
ANY
s ›
WHERE
grd1: s ∈ SERVICES not theorem ›
grd2: i_state(token_owner(s) ↦ s) = FAIL_DETECT_4 not theorem ›
M13

\[
\text{grd5: suspc_peers(token_owner(s) ↦ s) ≠ ∅ not theorem} \\
\text{THEN} \\
\text{act1: i_state(token_owner(s) ↦ s) = FAIL_ACTIV_4} \\
\text{act2: run_peers(s) = run_peers(s) \ suspc_peers(token_owner(s) ↦ s)} \\
\text{act3: fail_peers = fail_peers u \{s\}×suspc_peers(token_owner(s) ↦ s)} \\
\text{act4: suspc_peers(token_owner(s) ↦ s) = ∅} \\
\text{END}
\]

\[
\text{FAIL_CONFIGURE: extended ordinary} \\
\text{REFINES} \\
\text{FAIL_CONFIGURE ANY} \\
\text{WHERE} \\
\text{grd1: s ∈ SERVICES not theorem} \\
\text{grd2: i_state(token_owner(s) ↦ s) = FAIL_ACTIV_4 not theorem} \\
\text{grd3: card(run_peers(s)) < min_inst(s) not theorem} \\
\text{THEN} \\
\text{act1: i_state(token_owner(s) ↦ s) = FAIL_CONFIG_4} \\
\text{END}
\]

\[
\text{FAIL_IGNORE: extended ordinary} \\
\text{REFINES} \\
\text{FAIL_IGNORE ANY} \\
\text{WHERE} \\
\text{grd1: s ∈ SERVICES not theorem} \\
\text{grd2: i_state(token_owner(s) ↦ s) = FAIL_ACTIV_4 not theorem} \\
\text{grd3: card(run_peers(s)) ≥ min_inst(s) not theorem} \\
\text{THEN} \\
\text{act1: i_state(token_owner(s) ↦ s) = FAIL_IGN_4} \\
\text{END}
\]

\[
\text{IGNORE: extended ordinary} \\
\text{REFINES} \\
\text{IGNORE ANY} \\
\text{WHERE} \\
\text{grd1: s ∈ SERVICES not theorem} \\
\text{grd2: i_state(token_owner(s) ↦ s) = FAIL_IGN_4 not theorem} \\
\text{THEN} \\
\text{act1: i_state(token_owner(s) ↦ s) = RUN_4} \\
\text{END}
\]
REDEPLOY_INSTC: extended ordinary
  \begin{align*}
  &\text{REFINES} \quad \text{REDEPLOY_INSTC} \\
  &\text{ANY} \\
  &\quad \begin{align*}
  &\text{s} \rightarrow \text{a service s} \\
  &\text{i} \rightarrow \text{an instance i}
  \end{align*} \\
  &\text{WHERE} \\
  &\quad \begin{align*}
  &\text{grd1: } s \in \text{SERVICES not theorem} \\
  &\text{grd2: } i \in \text{PEERS not theorem} \\
  &\text{grd3: } i \notin \text{run_peers(s)} \cup \text{fail_peers[s]} \cup \text{unav_peers[s]} \cup \text{dep_inst[s]} \text{ not theorem} \\
  &\text{grd4: } \text{token_owner(s) }\rightarrow (s \rightarrow i) \notin \text{actv_inst not theorem} \\
  &\text{grd5: } \text{i_state(token_owner(s) }\rightarrow s) = \text{FAIL_CONFIG_4 not theorem} \\
  &\text{grd6: } \text{card(actv_inst[[token_owner(s)]]{s})} < \text{deplo_inst(s)} \text{ not theorem} \\
  &\text{grd7: } \text{card(dep_inst{[s]}) + card(run_peers(s))} < \text{min_inst(s)} \text{ not theorem} \\
  &\text{THEN} \\
  &\quad \text{act1: } \text{actv_inst }= \text{actv_inst }\cup \{\text{token_owner(s) }\rightarrow (s \rightarrow i)\} \\
  &\text{END} \\
  \end{align*}
\end{align*}

REDEPLOY_INSTS: extended ordinary
  \begin{align*}
  &\text{REFINES} \quad \text{REDEPLOY_INSTS} \\
  &\text{ANY} \\
  &\quad \begin{align*}
  &\text{s} \rightarrow \\
  \end{align*} \\
  &\text{WHERE} \\
  &\quad \begin{align*}
  &\text{grd1: } s \in \text{SERVICES not theorem} \\
  &\text{grd6: } \text{card(actv_inst[[token_owner(s)]]{s})} = \text{deplo_inst(s)} \text{ not theorem} \\
  &\text{grd7: } \text{card(dep_inst{[s]}) + card(run_peers(s))} < \text{min_inst(s)} \text{ not theorem} \\
  &\text{grd8: } \text{i_state(token_owner(s) }\rightarrow s) = \text{FAIL_CONFIG_4 not theorem} \\
  &\text{THEN} \\
  &\quad \text{act1: } \text{dep_inst }= \text{dep_inst }\cup \{s\times\text{actv_inst[[token_owner(s)]]}{s}\} \\
  &\text{act2: } \text{actv_inst }= \text{actv_inst }\rightarrow \{s \leftarrow \text{ran(actv_inst)}\} \\
  &\text{END} \\
  \end{align*}
\end{align*}

REDEPLOY: extended ordinary
  \begin{align*}
  &\text{REFINES} \quad \text{REDEPLOY} \\
  &\text{ANY} \\
  &\quad \begin{align*}
  &\text{s} \\
  &\text{WHERE} \\
  &\text{Page 7}
  \end{align*}
\end{align*}
M13

grd1:  s ∈ SERVICES not theorem >
grd2:  i_state(token_owner(s) ↦ s) = FAIL_CONFIG_4 not theorem >
grd7:  actv_inst[{token_owner(s)}]{{s}}=ø not theorem >
grd6:  dep_inst[] ≠ ø not theorem >
grd4:  card(run_peers(s)) + card(dep_inst[][]) ≥ min_inst(s) not theorem >
THEN
act1:  i_state(token_owner(s) ↦ s) = DPL_4 >
act2:  run_peers(s) = run_peers(s) u dep_inst[] >
act3:  dep_inst = {s} < dep_inst >
END

HEAL: extended ordinary >
REFINES
  HEAL
  ANY
  s >
WHERE
  grd1:  s ∈ SERVICES not theorem >
grd2:  i_state(token_owner(s) ↦ s) = DPL_4 not theorem >
THEN
act1:  i_state(token_owner(s) ↦ s) = RUN_4 >
END

UNFAIL_PEER: extended ordinary >
REFINES
  UNFAIL_PEER
  ANY
  s >
p >
WHERE
  grd1:  s ∈ SERVICES not theorem >
grd2:  p ∈ PEERS not theorem >
grd3:  s ↦ p ∈ fail_peers not theorem >
THEN
act1:  fail_peers = fail_peers\{s ↦ p} >
END

MAKE_PEER_AVAIL: extended ordinary >
REFINES
  MAKE_PEER_AVAIL
  ANY
  p >
WHERE
  grd1:  p ∈ PEERS not theorem >
grd2:  p ∈ unav_peers not theorem >
THEN
M13

act1: unav_peers = unav_peers \ {p} → END

END
MACHINE M14
REFINES M13
SEES C08
VARIABLES
  run_peers
  suspc_peers
  failr_peers
  dep_instc
  token_owner
  unav_peers
  suspc_inst
  rect_inst ▷ instances that are tried to be recontacted
  rctt_inst ▷ instances effectively recontacted after a try
  actv_inst ▷ instances activated by token owners
i_state
INVARINTS
  inv1: rect_inst ∈ (PEERS×SERVICES) ⇸ P(PEERS) not theorem ▷
  inv2: ∀ p, s · p ∈ PEERS ∧ s ∈ SERVICES ∧ (p ↦ s) ∈ dom(rect_inst) ⇒
        p = token_owner(s) not theorem ▷
  inv3: ∀ p, s · p ∈ PEERS ∧ s ∈ SERVICES ∧ p = token_owner(s) ⇒ (p ↦
        s) ∈ dom(rect_inst) not theorem ▷
  inv4: rect_inst ∈ (PEERS×SERVICES) ⇸ P(PEERS) not theorem ▷
  inv5: ∀ p, s · p ∈ PEERS ∧ s ∈ SERVICES ∧ (p ↦ s) ∈ dom(rctt_inst) ⇒
        p = token_owner(s) not theorem ▷
  inv6: ∀ p, s · p ∈ PEERS ∧ s ∈ SERVICES ∧ p = token_owner(s) ⇒ (p ↦
        s) ∈ dom(rctt_inst) not theorem ▷
  inv7: failr_peers ∈ SERVICES → P(PEERS) not theorem ▷
  inv8: dep_instc ∈ SERVICES → P(PEERS) not theorem ▷
  dep_instc ∈ SERVICES → dep_instc[s] = dep_instc(s) not
EVENTS
  INITIALISATION: not extended ordinary ▷
  THEN
  act2: run_peers = InitSrvcPeers ▷
  act3: suspc_peers = InitSuspPeers ▷
  act4: failr_peers = InitFail ▷
  act5: dep_instc = InitFail ▷
  act6: token_owner = init_tok ▷
  act7: unav_peers = Ø ▷
act8: suspc_inst = InitSuspPeers »
act10: rect_inst = InitSuspPeers »
act11: rctt_inst = InitSuspPeers »
act12: actv_inst = ∅ »
act13: i_state = InitStatus »

END

MAKE_PEER_UNAVAIL: not extended ordinary »
REFINES
MAKE_PEER_UNAVAIL

ANY
prs »
E »new values for token owner per service if needed
i_s »
p_s »
s_i »
rc_s »
rt_s »
WHERE

grd1: prs ⊆ PEERS not theorem »
grd2: prs ⊈ unav_peers not theorem »
grd3: E ∈ SERVICES → PEERS not theorem »new value for token
owner per service if needed
grd4: i_s ∈ (PEERS×SERVICES) ↔ STATES_4 not theorem »
grd5: p_s ∈ (PEERS×SERVICES) ↔ ℙ(PEERS) not theorem »
grd6: s_i ∈ (PEERS×SERVICES) ↔ ℙ(PEERS) not theorem »
grd7: rt_s ∈ (PEERS×SERVICES) ↔ ℙ(PEERS) not theorem »
grd8: rc_s ∈ (PEERS×SERVICES) ↔ ℙ(PEERS) not theorem »
grd9: ∀ srv ·srv ∈ SERVICES ∧ token_owner(srv) ∉ prs ⇒ E
(srv) = token_owner(srv) not theorem »If the token owner of a service srv does
not belong to prs, the token owner is not changed
grd10: ∀ srv ·srv ∈ SERVICES ∧ token_owner(srv) ∈ prs ⇒ E
(srv) ∈ run_peers(srv)\(unav_peers v prs v failr_peers(srv) v suspc_peers
(token_owner(srv) ↦ srv)) not theorem »if the owner of the token for a service
becomes unavailable, and the service

possess suspicious instances, then a new token owner among available and not

suspicious peers is chosen

grd11: dom(i_s) = E→ dom(p_s) = dom(i_s) ∧ dom(s_i) = dom
(i_s) ∧ dom(rc_s) = dom(i_s) ∧ dom(rt_s) = dom(i_s) not theorem »
grd12: ∀ srv ·srv ∈ SERVICES ⇒ ((E(srv) ↦ srv) » i_state
(token_owner(srv) ↦ srv) ∈ i_s ∧ (E(srv) » srv) » suspc_peers(token_owner(srv) ↦ srv)
∈ p_s) not theorem »
grd13: ∀ srv ·srv ∈ SERVICES ∧ token_owner(srv) ∈ prs ⇒ ((E
(srv) » srv) » suspc_inst(E(srv) » srv) ∈ s_i) ∧ ((E(srv) » srv) » rctt_inst(E

Page 2
(srv) ↦ srv ∈ rt_s ∧ ((E(srv) ↦ srv) ↦ rect_inst(E(srv) ↦ srv) ∈ rc_s) not theorem ›

grd14: ∀ srv · srv ∈ SERVICES ∧ token_owner(srv) ∈ prs ⇒ ((E(srv) ↦ srv) ↦ ∅ ∈ s_i) ∧ ((E(srv) ↦ srv) ↦ ∅ ∈ rt_s) ∧ ((E(srv) ↦ srv) ↦ ∅ ∈ rc_s) not theorem ›

THEN

act1: unav_peers ≔ unav_peers ∪ prs ›
the peers in prs become unavailable

act2: token_owner ≔ token_owner ≋ E ›
new value for token owner per service is given if needed

act3: rect_inst ≔ rc_s ›
the peers in prs can not try to recontact instances anymore

act4: rctt_inst ≔ rt_s ›
the peers in prs can not recontact instances anymore

act5: actv_inst ≔ prs ≋ actv_inst ›

act6: i_state ≔ i_s ›

act7: suspc_peers ≔ p_s ›

act8: suspc_inst ≔ s_i ›

END

SUSPECT_INST: extended ordinary ›

REFINES

SUSPECT_INST

ANY
s ›a service s

susp ›suspicous instances

WHERE

grd1: s ∈ SERVICES not theorem ›

grd2: susp ∈ PEERS not theorem ›

grd3: susp = run_peers(s) ∩ unav_peers not theorem ›
instances in susp are suspicious if the peers running them becomes unavailable

member of susp have not yet been suspected for s by the token owner of s

state of s is OK

grd4: susp ⊆ (token_owner(s) ≋ s) = ∅ not theorem ›

the

grd5: i_state(token_owner(s) ≋ s) = RUN_4 not theorem ›

the

THEN

act1: susp ⊆ (token_owner(s) ≋ s) = susp ›
members of susp become suspected instances for s by the token owner of s

END

FAIL: extended ordinary ›

REFINES

FAIL

ANY
s ›

WHERE

grd1: s ∈ SERVICES not theorem ›

Page 3
M14

grd2: \( i\_state(token\_owner(s) \gg s) \Rightarrow RUN\_4 \) not theorem

grd3: \( \text{suspc\_inst(token\_owner(s) \gg s) \neq \emptyset} \) not theorem

THEN

act1: \( i\_state(token\_owner(s) \gg s) \Rightarrow FAIL\_4 \)

act2: \( \text{suspc\_peers(token\_owner(s) \gg s) = \text{suspc\_inst(token\_owner(s) \gg s)}} \)

act3: \( \text{suspc\_inst(token\_owner(s) \gg s) = \emptyset} \)

END

RECONTACT\_INST\_OK: not extended ordinary

REFINES

RECONTACT\_INST\_OK

ANY
s \( \Rightarrow \) a service s
i \( \Rightarrow \) an instance i

WHERE

grd1: \( s \in \text{SERVICES} \) not theorem

grd2: \( i \in \text{PEERS} \) not theorem

grd3: \( i\_state(token\_owner(s) \gg s) = FAIL\_4 \) not theorem

\( \text{the state of } s \text{ is SUSPICIOUS} \)

grd4: \( \text{suspc\_peers(token\_owner(s) \gg s) \neq \emptyset} \) not theorem

\( \text{the set of suspicious peers for } s \text{ is not empty} \)

grd5: \( i \in \text{suspc\_peers(token\_owner(s) \gg s) \setminus unav\_peers} \) not theorem

\( i \text{ is a suspicious instance of } s \) and \( i \text{ is available (can be contacted)} \)

grd6: \( i \in \text{rect\_inst(token\_owner(s) \gg s)} \) not theorem

\( \text{the token owner of } s \text{ has not yet tried to recontact } i \)

grd7: \( \text{rect\_inst(token\_owner(s) \gg s) \subset \text{suspc\_peers(token\_owner(s) \gg s)}} \) not theorem

\( \text{the token owner of } s \text{ has not yet tried to recontact all the suspesious instances of } s \)

THEN

act1: \( \text{rect\_inst(token\_owner(s) \gg s) = rect\_inst(token\_owner(s) \gg s)} \cup \{i\} \)

\( \text{the token owner of } s \text{ has tried to recontact } i \)

act2: \( \text{rctt\_inst(token\_owner(s) \gg s) = rctt\_inst(token\_owner(s) \gg s)} \cup \{i\} \)

\( i \text{ is recontacted by the token owner of } s \text{ successfully} \)

END

RECONTACT\_INST\_KO: not extended ordinary

REFINES

RECONTACT\_INST\_KO

ANY
s \( \Rightarrow \) a service s
i \( \Rightarrow \) an instance i

WHERE

grd1: \( s \in \text{SERVICES} \) not theorem

grd2: \( i \in \text{PEERS} \) not theorem

grd3: \( i\_state(token\_owner(s) \gg s) = FAIL\_4 \) not theorem

\( \text{the state of } s \text{ is SUSPICIOUS} \)

grd4: \( \text{suspc\_peers(token\_owner(s) \gg s) \neq \emptyset} \) not theorem

\( \text{the set of suspicious peers for } s \text{ is not empty} \)

Page 4
of suspicious peers for \( s \) is not empty

\[ \text{grd5: } i \in \text{susp}_\text{c peers}(\text{token}\_\text{owner}(s) \Rightarrow s) \cap \text{unav}_\text{peers} \]

Not theorem \( \Rightarrow i \) is a suspicious instance of \( s \) and \( i \) is unavailable (can not be contacted)

\[ \text{grd6: } i \notin \text{rect}_\text{inst}(\text{token}\_\text{owner}(s) \Rightarrow s) \]

Not theorem \( \Rightarrow \) the token owner of \( s \) has not yet tried to recontact \( i \)

\[ \text{grd7: } \text{rect}_\text{inst}(\text{token}\_\text{owner}(s) \Rightarrow s) \subseteq \text{susp}_\text{c peers}(\text{token}\_\text{owner}(s) \Rightarrow s) \]

Not theorem \( \Rightarrow \) the token owner of \( s \) has not yet tried to recontact all the suspicious instances of \( s \)

THEN

\[ \text{act1: } \text{rect}_\text{inst}(\text{token}\_\text{owner}(s) \Rightarrow s) = (\text{rect}_\text{inst}(\text{token}\_\text{owner}(s) \Rightarrow s) \cup \{i\}) \]

\( \Rightarrow \) the token owner of \( s \) has tried to recontact \( i \)

END

FAIL\_\text{DETECT: not extended ordinary } \Rightarrow

\text{REFINES}

FAIL\_\text{DETECT}

\text{ANY } s \Rightarrow

\text{WHERE}

\[ \text{grd1: } s \in \text{SERVICES} \]

\[ \text{grd2: } \text{i}\_\text{state}(\text{token}\_\text{owner}(s) \Rightarrow s) = \text{FAIL}_4 \]

\[ \text{grd5: } \text{susp}_\text{c peers}(\text{token}\_\text{owner}(s) \Rightarrow s) \neq \emptyset \]

\[ \text{grd8: } \text{rect}_\text{inst}(\text{token}\_\text{owner}(s) \Rightarrow s) = \text{susp}_\text{c peers}(\text{token}\_\text{owner}(s) \Rightarrow s) \]

Not theorem \( \Rightarrow \) \( s \) nottheorem \( \Rightarrow \)

THEN

\[ \text{act1: } \text{i}\_\text{state}(\text{token}\_\text{owner}(s) \Rightarrow s) = \text{FAIL\_DETECT}_4 \]

\[ \text{act2: } \text{susp}_\text{c peers}(\text{token}\_\text{owner}(s) \Rightarrow s) = \text{susp}_\text{c peers} \]

\( (\text{token}\_\text{owner}(s) \Rightarrow s) \setminus \text{rect}_\text{tnt}(\text{token}\_\text{owner}(s) \Rightarrow s) \)

\[ \text{act3: } \text{rect}_\text{inst}(\text{token}\_\text{owner}(s) \Rightarrow s) = \emptyset \]

\[ \text{act4: } \text{rect}_\text{tnt}(\text{token}\_\text{owner}(s) \Rightarrow s) = \emptyset \]

END

\text{IS\_OK: extended ordinary } \Rightarrow

\text{REFINES}

\text{IS\_OK}

\text{ANY } s \Rightarrow

\text{WHERE}

\[ \text{grd1: } s \in \text{SERVICES} \]

\[ \text{grd2: } \text{i}\_\text{state}(\text{token}\_\text{owner}(s) \Rightarrow s) = \text{FAIL\_DETECT}_4 \]

\[ \text{grd5: } \text{susp}_\text{c peers}(\text{token}\_\text{owner}(s) \Rightarrow s) = \emptyset \]

\( (\text{token}\_\text{owner}(s) \Rightarrow s) \Rightarrow \)

THEN

\[ \text{act1: } \text{i}\_\text{state}(\text{token}\_\text{owner}(s) \Rightarrow s) = \text{RUN}_4 \]

END

FAIL\_\text{ACTIV: not extended ordinary } \Rightarrow
REFINES

FAIL_ACTIV

ANY

s ➔

WHERE

grd1: s ∈ SERVICES not theorem ➔
grd2: i_state(token_owner(s) ↦ s) = FAIL_DETECT_4 not theorem ➔
grd5: suspc_peers(token_owner(s) ↦ s) ≠ ∅ not theorem ➔

THEN

act1: i_state(token_owner(s) ↦ s) ≔ FAIL_ACTIV_4 ➔
act2: run_peers(s) ≔ run_peers(s) \ suspc_peers(token_owner(s) ↦ s) ➔
act3: failr_peers(s) ≔ failr_peers(s) ∪ suspc_peers(token_owner(s) ↦ s) ➔
act4: suspc_peers(token_owner(s) ↦ s) ≔ ∅ ➔

END

FAIL_CONFIGURE: extended ordinary ➔

REFINES

FAIL_CONFIGURE

ANY

s ➔

WHERE

grd1: s ∈ SERVICES not theorem ➔
grd2: i_state(token_owner(s) ↦ s) = FAIL_ACTIV_4 not theorem ➔
grd3: card(run_peers(s)) < min_inst(s) not theorem ➔

THEN

act1: i_state(token_owner(s) ↦ s) ≔ FAIL_CONFIG_4 ➔

END

FAIL_IGNORE: extended ordinary ➔

REFINES

FAIL_IGNORE

ANY

s ➔

WHERE

grd1: s ∈ SERVICES not theorem ➔
grd2: i_state(token_owner(s) ↦ s) = FAIL_ACTIV_4 not theorem ➔
grd3: card(run_peers(s)) ≥ min_inst(s) not theorem ➔

THEN

act1: i_state(token_owner(s) ↦ s) ≔ FAIL_IGN_4 ➔

END

IGNORE: extended ordinary ➔

REFINES

IGNORE

ANY

Page 6
WHERE
grd1: s ∈ SERVICES not theorem
grd2: i_state(token_owner(s) ↦ s) = FAIL_IGN_4 not theorem

THEN
act1: i_state(token_owner(s) ↦ s) = RUN_4

END

REDEPLOY_INSTC: not extended ordinary
REFINES
REDEPLOY_INSTC

ANY
s → a service s
i → an instance i

WHERE
grd1: s ∈ SERVICES not theorem
grd2: i ∈ PEERS not theorem
grd3: i ∉ run_peers(s) ∪ failr_peers(s) ∪ unav_peers ∪ dep_instc(s) not theorem → i does not run s, is not failed for s, is not unavailable and is not already activated for s
grd4: token_owner(s) ↦ (s ↦ i) ∉ actv_inst not theorem
grd5: i_state(token_owner(s) ↦ s) = FAIL_CONFIG_4 not theorem
grd6: card(actv_inst[{token_owner(s)}]{{s}}) < deplo_ins(t(s)
not theorem
grd7: card(dep_instc(s)) + card(run_peers(s)) < min_inst(s)
not theorem

THEN
act1: actv_inst = actv_inst ∪ {token_owner(s) ↦ (s ↦ i)}

END

REDEPLOY_INSTS: not extended ordinary
REFINES
REDEPLOY_INSTS

ANY
s →

WHERE
grd1: s ∈ SERVICES not theorem
grd6: card(actv_inst[{token_owner(s)}]{{s}}) = deplo_ins(t(s)
not theorem
grd7: card(dep_instc(s)) + card(run_peers(s)) < min_inst(s)
not theorem
grd8: i_state(token_owner(s) ↦ s) = FAIL_CONFIG_4 not theorem

THEN
act1: dep_instc(s) = dep_instc(s) ∪ actv_inst[{token_owner (s)}]{{s}}

act2: actv_inst = actv_inst ∪ {{s} ∩ ran(actv_inst)}
REDEPLOY: not extended ordinary

REDEPLOY

ANY s

WHERE

grd1: s ∈ SERVICES not theorem
grd2: i_state(token_owner(s) ↦ s) = FAIL_CONFIG_4 not theorem

THEN

act1: i_state(token_owner(s) ↦ s) ≔ DPL_4
act2: run_peers(s) ≔ run_peers(s) ∪ dep_instc(s)
act3: dep_instc(s) ≔ ∅

END

HEAL: extended ordinary

HEAL

ANY s

WHERE

grd1: s ∈ SERVICES not theorem
grd2: i_state(token_owner(s) ↦ s) = DPL_4 not theorem

THEN

act1: i_state(token_owner(s) ↦ s) ≔ RUN_4

END

UNFAIL_PEER: not extended ordinary

UNFAIL_PEER

ANY s p

WHERE

grd1: s ∈ SERVICES not theorem
grd2: p ∈ PEERS not theorem
grd3: p ∈ failr_peers(s) not theorem

THEN

act1: failr_peers(s) = failr_peers(s) \ {p}

END

MAKE_PEER_AVAIL: extended ordinary
M14

REFINES
   MAKE_PEER_AVAIL
ANY
   p >
WHERE
   grd1:  p ∈ PEERS not theorem >
   grd2:  p ∈ unav_peers not theorem >
THEN
   act1:  unav_peers = unav_peers \ {p} >
END
MACHINE M15
REFINES M14
SEES C08
VARIABLES
  run_peers
  suspc_peers
  failr_peers
  dep_instc
  token_owner
  unav_peers
  suspc_inst
  rect_inst \(\triangleright\) instances that are tried to be recontacted
  rctt_inst \(\triangleright\) instances effectively recontacted after a try
  actv_instc \(\triangleright\) instances activated by token owner
  i_state

INVARINTS
  inv1: actv_instc \(\in\) (PEERS \times SERVICES) \(\leftrightarrow\) P(PEERS) not theorem
  inv2: \(\forall p, s \cdot p \in\) PEERS \(\land\) s \(\in\) SERVICES \(\land\) (p \(\mapsto\) s) \(\in\) dom(actv_instc) \(\Rightarrow\) p = token_owner(s) not theorem
  inv3: \(\forall p, s \cdot p \in\) PEERS \(\land\) s \(\in\) SERVICES \(\land\) p = token_owner(s) \(\Rightarrow\) (p \(\mapsto\) s) \(\in\) dom(actv_instc) not theorem
  gluing_tok_\text{own}\_rec1: \(\forall p, s \cdot p \in\) PEERS \(\land\) s \(\in\) SERVICES \(\land\) (p \(\mapsto\) s) \(\in\) dom(actv_instc) \(\Rightarrow\) actv_instc[p] = actv_instc[p \(\mapsto\) s] not theorem

EVENTS

INITIALISATION: not extended ordinary

THEN
  act2: run_peers := InitSrvcPeers
  act3: suspc_peers := InitSuspPeers
  act4: failr_peers := InitFail
  act5: dep_instc := InitFail
  act6: token_owner := init_tok
  act7: unav_peers := \(\emptyset\)
  act8: suspc_inst := InitSuspPeers
  act10: rect_inst := InitSuspPeers
  act11: rctt_inst := InitSuspPeers
  act12: actv_instc := InitSuspPeers
  act13: i_state := InitStatus

END

MAKE_PEER_UNAVAIL: not extended ordinary

REFINES MAKE_PEER_UNAVAIL

ANY
  prs
  E \(\triangleright\) new values for token owner per service if needed

Page 1
WHERE

grd1: \( p_s \subseteq \text{PEERS} \) not theorem

grd2: \( p_s \not\subseteq \text{unav_peers} \) not theorem

grd3: \( E \in \text{SERVICES} \rightarrow \text{PEERS} \) not theorem \( \text{new value for token owner per service if needed} \)

grd4: \( i_s \in (\text{PEERS} \times \text{SERVICES}) \leftrightarrow \text{STATES}_4 \) not theorem

grd5: \( p_s \in (\text{PEERS} \times \text{SERVICES}) \leftrightarrow \mathcal{P} (\text{PEERS}) \) not theorem

grd6: \( s_i \in (\text{PEERS} \times \text{SERVICES}) \leftrightarrow \mathcal{P} (\text{PEERS}) \) not theorem

grd7: \( rt_s \in (\text{PEERS} \times \text{SERVICES}) \leftrightarrow \mathcal{P} (\text{PEERS}) \) not theorem

grd8: \( rc_s \in (\text{PEERS} \times \text{SERVICES}) \leftrightarrow \mathcal{P} (\text{PEERS}) \) not theorem

grd9: \( ac_i \in (\text{PEERS} \times \text{SERVICES}) \leftrightarrow \mathcal{P} (\text{PEERS}) \) not theorem

grd10: \( \text{dom}(i_s) = E \land \text{dom}(p_s) = \text{dom}(i_s) \land \text{dom}(s_i) = \text{dom}(i_s) \land \text{dom}(rc_s) = \text{dom}(i_s) \land \text{dom}(rt_s) = \text{dom}(i_s) \land \text{dom}(ac_i) = \text{dom}(i_s) \) not theorem

grd11: \( \forall \text{srv} \cdot \text{srv} \in \text{SERVICES} \land \text{token_owner}(\text{srv}) \notin \text{prs} \)
\[ \Rightarrow \]
\[ E(\text{srv}) = \text{token_owner}(\text{srv}) \land s_i(E(\text{srv}) \mapsto \text{srv}) = \text{susp_inst}(E(\text{srv}) \mapsto \text{srv}) \land rt_s(E(\text{srv}) \mapsto \text{srv}) = \text{rctt_inst}(E(\text{srv}) \mapsto \text{srv}) \land rc_s(E(\text{srv}) \mapsto \text{srv}) = \text{rect_inst}(E(\text{srv}) \mapsto \text{srv}) \land ac_i(E(\text{srv}) \mapsto \text{srv}) = \text{actv_instc}(E(\text{srv}) \mapsto \text{srv}) \not \text{theorem} \]

If the token owner of a service \( \text{srv} \) does not belong to \( \text{prs} \), the token owner is not changed

\( \forall \text{srv} \cdot \text{srv} \in \text{SERVICES} \land \text{token_owner}(\text{srv}) \in \text{prs} \)
\[ \Rightarrow \]
\[ E(\text{srv}) \in \text{run_peers}(\text{srv}) \setminus (\text{unav_peers} \cup \text{prs} \cup \text{failr_peers}(\text{srv}) \cup \text{susp_peers}(\text{token_owner}(\text{srv}) \mapsto \text{srv})) \land s_i(E(\text{srv}) \mapsto \text{srv}) = \emptyset \land rt_s(E(\text{srv}) \mapsto \text{srv}) = \emptyset \land rc_s(E(\text{srv}) \mapsto \text{srv}) = \emptyset \land ac_i(E(\text{srv}) \mapsto \text{srv}) = \emptyset \not \text{not theorem} \]

if the owner of the token for a service becomes unavailable, and the service possess suspicious instances, then a new token owner among available and not suspicious peers is chosen

\( \forall \text{srv} \cdot \text{srv} \in \text{SERVICES} \Rightarrow i_s(E(\text{srv}) \mapsto \text{srv}) = \text{i_state}(\text{token_owner}(\text{srv}) \mapsto \text{srv}) \land p_s(E(\text{srv}) \mapsto \text{srv}) = \text{susp_peers}(\text{token_owner}(\text{srv}) \mapsto \text{srv}) \not \text{not theorem} \)

THEN

act1: \( \text{unav_peers} = \text{unav_peers} \cup \text{prs} \) \( \text{the peers in prs become unavailable} \)
act2:  \( \text{token}\_\text{owner} \equiv \text{token}\_\text{owner} \downarrow \text{E} \quad \text{new value for token owner} \)

per service is given if needed

act3:  \( \text{rect}\_\text{inst} \equiv \text{rc}\_\text{s} \quad \text{the peers in prs can not try to} \)
recontact instances anymore

act4:  \( \text{rctt}\_\text{inst} \equiv \text{rt}\_\text{s} \quad \text{the peers in prs can not recontact} \)
instances anymore

act5:  \( \text{actv}\_\text{instc} \equiv \text{ac}\_\text{i} \quad \)

act6:  \( \text{i}\_\text{state} \equiv \text{i}\_\text{s} \quad \)

act7:  \( \text{suspc}\_\text{peers} \equiv \text{p}\_\text{s} \quad \)

act8:  \( \text{suspc}\_\text{inst} \equiv \text{s}\_\text{i} \quad \)

END

**SUSPECT\_INST:** extended ordinary

**REFINES**

**SUSPECT\_INST**

**ANY**

\( s \quad \text{a service s} \)

\( \text{susp} \quad \text{suspicious instances} \)

**WHERE**

\( \text{grd1: } s \in \text{SERVICES not theorem} \)

\( \text{grd2: } \text{susp} \subseteq \text{PEERS not theorem} \)

\( \text{grd3: } \text{susp} = \text{run}\_\text{peers}(s) \cap \text{unav}\_\text{peers} \text{not theorem} \)

\( \text{instances in susp are suspicious if the peers running them becomes unavailable} \)

\( \text{grd4: } \text{suspc}\_\text{inst}(\text{token}\_\text{owner}(s) \Rightarrow s) = \emptyset \text{ not theorem} \)

\( \text{member of susp have not yet been suspected for s by the token owner of s} \)

\( \text{grd5: } \text{i}\_\text{state}(\text{token}\_\text{owner}(s) \Rightarrow s) = \text{RUN}\_4 \text{ not theorem} \)

\( \text{the state of s is OK} \)

\( \text{grd6: } \text{susp} \neq \emptyset \text{ not theorem} \)

**THEN**

\( \text{act1: } \text{suspc}\_\text{inst}(\text{token}\_\text{owner}(s) \Rightarrow s) = \text{susp} \quad \text{the members of} \)

\( \text{susp become suspected instances for s by the token owner of s} \)

END

**FAIL:** extended ordinary

**REFINES**

**FAIL**

**ANY**

\( s \quad \)

**WHERE**

\( \text{grd1: } s \in \text{SERVICES not theorem} \)

\( \text{grd2: } \text{i}\_\text{state}(\text{token}\_\text{owner}(s) \Rightarrow s) = \text{RUN}\_4 \text{ not theorem} \)

\( \text{grd3: } \text{suspc}\_\text{inst}(\text{token}\_\text{owner}(s) \Rightarrow s) \neq \emptyset \text{ not theorem} \)

**THEN**

\( \text{act1: } \text{i}\_\text{state}(\text{token}\_\text{owner}(s) \Rightarrow s) = \text{FAIL}\_4 \quad \)

\( \text{act2: } \text{suspc}\_\text{peers}(\text{token}\_\text{owner}(s) \Rightarrow s) = \text{suspc}\_\text{inst}(\text{token}\_\text{owner}(s) \Rightarrow s) \quad \)

\( \text{act3: } \text{suspc}\_\text{inst}(\text{token}\_\text{owner}(s) \Rightarrow s) = \emptyset \quad \)

END
RECONTACT_INST_OK: extended ordinary >
  REFINES
  RECONTACT_INST_OK
  ANY
  s ➔ a service s
  i ➔ an instance i
  WHERE
  grd1: s ∈ SERVICES not theorem ➔ the state of s is SUSPICIOUS
  grd2: i ∈ PEERS not theorem ➔ the set of suspicious peers for s is not empty
  grd3: i_state(token_owner(s) ↦ s) = FAIL_4 not theorem ➔ the token owner of s has not yet tried to recontact i
  grd4: suspc_peers(token_owner(s) ↦ s) ≠ ∅ not theorem ➔ the set of suspicious peers for s is not empty
  grd5: i ∈ suspc_peers(token_owner(s) ↦ s)∩unav_peers not theorem ➔ i is a suspicious instance of s and is unavailable (can not be contacted)
  grd6: i ∉ rect_inst(token_owner(s) ↦ s) not theorem ➔ the token owner of s has not yet tried to recontact i
  grd7: rect_inst(token_owner(s) ↦ s) ⊂ suspc_peers(token_owner(s) ↦ s) not theorem ➔ the token owner of s has not yet tried to recontact all the suspicious instances of s
  THEN
  act1: rect_inst(token_owner(s) ↦ s) := rect_inst(token_owner(s) ↦ s) ∪ {i} ➔ the token owner of s has tried to recontact i
  act2: rctt_inst(token_owner(s) ↦ s) := rctt_inst(token_owner(s) ↦ s) ∪ {i} ➔ i is recontacted by the token owner of s successfully
  END

RECONTACT_INST_KO: extended ordinary >
  REFINES
  RECONTACT_INST_KO
  ANY
  s ➔ a service s
  i ➔ an instance i
  WHERE
  grd1: s ∈ SERVICES not theorem ➔ the state of s is SUSPICIOUS
  grd2: i ∈ PEERS not theorem ➔ the set of suspicious peers for s is not empty
  grd3: i_state(token_owner(s) ↦ s) = FAIL_4 not theorem ➔ the token owner of s has not yet tried to recontact i
  grd4: suspc_peers(token_owner(s) ↦ s) ≠ ∅ not theorem ➔ the set of suspicious peers for s is not empty
  grd5: i ∈ suspc_peers(token_owner(s) ↦ s)∩unav_peers not theorem ➔ i is a suspicious instance of s and is unavailable (can not be contacted)
  grd6: i ∉ rect_inst(token_owner(s) ↦ s) not theorem ➔ the token owner of s has not yet tried to recontact i
  grd7: rect_inst(token_owner(s) ↦ s) ⊂ suspc_peers(token_owner(s) ↦ s) not theorem ➔ the token owner of s has not yet tried to recontact all the suspicious instances of s
the suspicious instances of s
THEN
act1: rect_inst(token_owner(s) ↦ s) = rect_inst(token_owner(s) ↦ s) ∪ {i}  
BEGIN
the token owner of s has tried to recontact i
END

FAIL_DETECT: extended ordinary
REFINES
FAIL_DETECT
ANY
s  
WHERE
grd1: s ∈ SERVICES not theorem
grd2: i_state(token_owner(s) ↦ s) = FAIL_4 not theorem
grd5: suspc_peers(token_owner(s) ↦ s) ≠ ∅ not theorem
grd8: rect_inst(token_owner(s) ↦ s) = suspc_peers(token_owner(s) ↦ s) not theorem
THEN
act1: i_state(token_owner(s) ↦ s) = FAIL_DETECT_4
act2: suspc_peers(token_owner(s) ↦ s) = suspc_peers(token_owner(s) ↦ s) \ rctt_inst(token_owner(s) ↦ s)
act3: rect_inst(token_owner(s) ↦ s) = ∅
act4: rctt_inst(token_owner(s) ↦ s) = ∅
END

IS_OK: extended ordinary
REFINES
IS_OK
ANY
s  
WHERE
grd1: s ∈ SERVICES not theorem
grd2: i_state(token_owner(s) ↦ s) = FAIL_DETECT_4 not theorem
grd5: suspc_peers(token_owner(s) ↦ s) = ∅ not theorem
THEN
act1: i_state(token_owner(s) ↦ s) = RUN_4
END

FAIL_ACTIV: extended ordinary
REFINES
FAIL_ACTIV
ANY
s  
WHERE
grd1: s ∈ SERVICES not theorem
grd2: i_state(token_owner(s) ↦ s) = FAIL_DETECT_4 not theorem
M15

grd5:  
 THEN  
 act1:  
 act2:  
 run_peers(s) = run_peers(s) \ suspic_peers(token_owner(s) \s)  
 END

FAIL_CONFIGURE:  extended ordinary  

FAIL_IGNORE:  extended ordinary  

IGNORE:  extended ordinary  

REDEPLOY_INSTC: not extended ordinary

REFINES

REDEPLOY_INSTC

ANY

s \rightarrow \text{a service s}

i \rightarrow \text{an instance i}

WHERE

grd1: s ∈ SERVICES not theorem

grd2: i ∈ PEERS not theorem

grd3: i ∉ run_peers(s) u failr_peers(s) u unav_peers u

dep_instc(s) not theorem \rightarrow i \text{ does not run s, is not failed for s, is not unavailable and is not already activated for s}

grd4: i ∉ actv_instc(token_owner(s) ↦ s) not theorem

grd5: i_state(token_owner(s) ↦ s) = FAIL_CONFIG_4 not theorem

\rightarrow

grd6: card(actv_instc(token_owner(s) ↦ s)) < deplo_inst(s) not theorem

not theorem

THEN

act1: actv_instc(token_owner(s) ↦ s) = actv_instc(token_owner(s) ↦ s) u \{i\}

END

REDEPLOY_INSTS: not extended ordinary

REFINES

REDEPLOY_INSTS

ANY

s \rightarrow

WHERE

grd1: s ∈ SERVICES not theorem

grd6: card(actv_instc(token_owner(s) ↦ s)) = deplo_inst(s) not theorem

not theorem

\rightarrow

grd7: card(dep_instc(s)) + card(run_peers(s)) < min_inst(s)

\rightarrow

grd8: i_state(token_owner(s) ↦ s) = FAIL_CONFIG_4 not theorem

\rightarrow

THEN

act1: dep_instc(s) = dep_instc(s) u actv_instc(token_owner(s) ↦ s)

\rightarrow

act2: actv_instc(token_owner(s) ↦ s) = \emptyset

END

REDEPLOY: not extended ordinary

REFINES

REDEPLOY

ANY

s \rightarrow
M15

WHERE

grd1: \( s \in \text{SERVICES} \) not theorem
grd2: \( i_{\text{state}}(\text{token\_owner}(s) \mapsto s) = \text{FAIL\_CONFIG\_4} \) not theorem

\( \therefore \)

grd7: \( \text{actv\_instc}(\text{token\_owner}(s) \mapsto s) = \emptyset \) not theorem
grd6: \( \text{dep\_instc}(s) \neq \emptyset \) not theorem
grd4: \( \text{card}(\text{run\_peers}(s)) + \text{card}(\text{dep\_instc}(s)) \geq \text{min\_inst}(s) \) not theorem

THEN

act1: \( i_{\text{state}}(\text{token\_owner}(s) \mapsto s) = \text{DPL\_4} \)
act2: \( \text{run\_peers}(s) = \text{run\_peers}(s) \cup \text{dep\_instc}(s) \)
act3: \( \text{dep\_instc}(s) = \emptyset \)

END

HEAL: extended ordinary

REFINES HEAL

ANY \( s \)

WHERE

grd1: \( s \in \text{SERVICES} \) not theorem
grd2: \( i_{\text{state}}(\text{token\_owner}(s) \mapsto s) = \text{DPL\_4} \) not theorem

THEN

act1: \( i_{\text{state}}(\text{token\_owner}(s) \mapsto s) = \text{RUN\_4} \)

END

UNFAIL\_PEER: extended ordinary

REFINES UNFAIL\_PEER

ANY \( s \)

WHERE

grd1: \( s \in \text{SERVICES} \) not theorem
grd2: \( p \in \text{PEERS} \) not theorem
grd3: \( p \in \text{failr\_peers}(s) \) not theorem

THEN

act1: \( \text{failr\_peers}(s) = \text{failr\_peers}(s) \setminus \{p\} \)

END

MAKE\_PEER\_AVAIL: extended ordinary

REFINES MAKE\_PEER\_AVAIL

ANY \( p \)

WHERE

grd1: \( p \in \text{PEERS} \) not theorem
grd2: \( p \in \text{unav\_peers} \) not theorem
M15

THEN

    act1: unav_peers = unav_peers \ {p} \\

END

END
MACHINE M16
REFINES M15
SEES C09
VARIABLES
  run_peers
  suspc_peers
  failr_peers
  dep_instc
  token_owner
  unav_peers
  suspc_inst
  rect_inst » instances that are tried to be recontacted
  rctt_inst » instances effectively recontacted after a try
  actv_instc » instances activated by token owners
  inst_state

INVARIANTS
  inv1: inst_state ∈ (PEERS×SERVICES) ⇸ STATES_4 not theorem
  inv2: ∀ s · s ∈ SERVICES ⇒ token_owner(s) ↦ s ∈ dom(inst_state) not theorem
  inv3: ∀ s · s ∈ SERVICES ⇒ rctt_inst(token_owner(s) ↦ s) ⊆ run_peers(s) not theorem
  inv4: ∀ s · s ∈ SERVICES ⇒ suspc_peers(token_owner(s) ↦ s) ⊆ run_peers(s) not theorem
  inv5: ∀ s · s ∈ SERVICES ⇒ suspc_inst(token_owner(s) ↦ s) ⊆ run_peers(s) not theorem
  inv6: ∀ s · s ∈ SERVICES ⇒ rect_inst(token_owner(s) ↦ s) ⊆ run_peers(s) not theorem
  inv7: ∀ s · s ∈ SERVICES ⇒ token_owner(s) ∉ suspc_inst(token_owner(s) ↦ s) not theorem
  inv8: ∀ s · s ∈ SERVICES ⇒ token_owner(s) ∉ suspc_peers(token_owner(s) ↦ s) not theorem
  inv9: ∀ s · s ∈ SERVICES ⇒ token_owner(s) ∉ rctt_inst(token_owner(s) ↦ s) not theorem
  inv10: ∀ s · s ∈ SERVICES ⇒ token_owner(s) ∉ rect_inst(token_owner(s) ↦ s) not theorem
  inv11: ∀ s · s ∈ SERVICES ⇒ suspc_inst(token_owner(s) ↦ s) ∩ suspc_peers(token_owner(s) ↦ s) = ∅ not theorem
  inv12: ∀ s · s ∈ SERVICES ∧ inst_state(token_owner(s) ↦ s) ∉ {FAIL_4,FAIL_DETECT_4} ⇒ suspc_peers(token_owner(s) ↦ s) = ∅ not theorem
  inv13: ∀ s · s ∈ SERVICES ∧ inst_state(token_owner(s) ↦ s) ≠ FAIL_4 ⇒ rctt_inst(token_owner(s) ↦ s) = ∅ not theorem
  inv14: ∀ s · s ∈ SERVICES ∧ inst_state(token_owner(s) ↦ s) ≠ FAIL_4 ⇒ rect_inst(token_owner(s) ↦ s) = ∅ not theorem

Page 1
EVENTS

INITIALISATION: not extended ordinary

THEN

act1: run_peers = InitSrvcPeers
act2: suspc_peers = InitSuspPeers
act3: failr_peers = InitFail
act4: dep_instc = InitFail
act5: token_owner = init_tok
act6: unav_peers = \emptyset
act7: suspc_inst = InitSuspPeers
act8: rect_inst = InitSuspPeers
act9: rctt_inst = InitSuspPeers
act10: actv_instc = InitSuspPeers
act11: inst_state = InitStateSrv

END

MAKE_PEER_UNAVAIL: not extended ordinary

REFINES

MAKE_PEER_UNAVAIL

ANY

prs → E
p_s
s_i
rc_s
rt_s
ac_i

WHERE

grd1: prs ⊆ PEERS not theorem
grd2: prs ⊈ unav_peers not theorem
grd3: ∀ srv · srv ∈ SERVICES ⇒ dom(dom(inst_state) ▷ {srv}) ≠ \emptyset not theorem
\prs ≠ \emptyset not theorem
grd4: E ∈ SERVICES → PEERS not theorem → new value for token owner per service if needed

grd5: p_s ∈ (PEERS×SERVICES) ↔ ℙ(PEERS) not theorem
grd6: s_i ∈ (PEERS×SERVICES) ↔ ℙ(PEERS) not theorem
grd7: rt_s ∈ (PEERS×SERVICES) ↔ ℙ(PEERS) not theorem
grd8: rc_s ∈ (PEERS×SERVICES) ↔ ℙ(PEERS) not theorem
grd9: ac_i ∈ (PEERS×SERVICES) ↔ ℙ(PEERS) not theorem

(grd10) dom(p_s) = E~ ∧ dom(s_i) = E~ ∧ dom(rc_s) = E~ ∧ dom(rt_s) = E~ ∧ dom(ac_i) = E~ not theorem

∀ srv · srv ∈ SERVICES ∧ token_owner(srv) ∈ prs
E(srv) = token_owner(srv) ∧
s_i(E(srv) ↦ srv) = suspc_inst(E(srv) ↦ srv) ∧
rt_s(E(srv) ↦ srv) = rctt_inst(E(srv) ↦ srv) ∧
rc_s(E(srv) ↦ srv) = rect_inst(E(srv) ↦ srv) ∧
ac_i(E(srv) ↦ srv) = actv_instc(E(srv) ↦ srv) not
If the token owner of a service \( \text{srv} \) does not belong to \( \text{prs} \), the token owner is not changed.

\[
\text{grd12: } \forall \text{srv} \cdot \text{srv} \in \text{SERVICES} \land \text{token\_owner(\text{srv})} \in \text{prs} \\
\implies \text{E(\text{srv})} \in \text{run\_peers(\text{srv})}\setminus(\text{unav\_peers} \cup \text{prs} \cup \text{failr\_peers}(
\text{srv}) \cup \text{suspc\_peers}(\text{token\_owner(\text{srv})} \mapsto \text{srv})) \land \\
\text{E(\text{srv})} \mapsto \text{srv} \in \text{dom(\text{inst\_state})} \land \\
\text{inst\_state(\text{E(\text{srv})}\mapsto \text{srv})} = \text{inst\_state(\text{token\_owner(\text{srv})} \mapsto \text{srv})} \land \\
\text{s\_i(\text{E(\text{srv})}\mapsto \text{srv})} = \emptyset \land \\
\text{rt\_s(\text{E(\text{srv})}\mapsto \text{srv})} = \emptyset \land \\
\text{rc\_s(\text{E(\text{srv})}\mapsto \text{srv})} = \emptyset \land \\
\text{ac\_i(\text{E(\text{srv})}\mapsto \text{srv})} = \emptyset
\]

\]
M16

grd4: \(\text{suspc}_{\text{inst}}(\text{token}_{\text{owner}}(s) \mapsto s) = \emptyset\) not theorem \>

member of susp have not yet been suspected for \(s\) by the token owner of \(s\)

grd5: \(\text{inst}_{\text{state}}(\text{token}_{\text{owner}}(s) \mapsto s) = \text{RUN}_4\) not theorem \>

the state of \(s\) is OK

grd6: \(\text{suspc} \neq \emptyset\) not theorem \>

THEN

act1: \(\text{suspc}_{\text{inst}}(\text{token}_{\text{owner}}(s) \mapsto s) = \text{suspc}\) \>

the members of susp become suspected instances for \(s\) by the token owner of \(s\)

END

FAIL: not extended ordinary \>

REFINES

FAIL

ANY

s \>

prop \>

WHERE

grd1: \(s \in \text{SERVICES}\) not theorem \>

grd2: \(\text{prop} \subseteq \text{PEERS}\) not theorem \>

grd3: \(\text{inst}_{\text{state}}(\text{token}_{\text{owner}}(s) \mapsto s) = \text{RUN}_4\) not theorem \>

grd4: \(\text{suspc}_{\text{inst}}(\text{token}_{\text{owner}}(s) \mapsto s) \neq \emptyset\) not theorem \>

grd5: \(\text{prop} = \text{run}_{\text{peers}}(s) \setminus (\text{suspc}_{\text{inst}}(\text{token}_{\text{owner}}(s) \mapsto s) \cup \text{unav}_{\text{peers}})\) not theorem \>

THEN

act1: \(\text{inst}_{\text{state}} = \text{inst}_{\text{state}} \setminus (\text{prop} \times \{s\} \times \{\text{FAIL}_4\})\) \>

act2: \(\text{suspc}_{\text{peers}}(\text{token}_{\text{owner}}(s) \mapsto s) = \text{suspc}_{\text{inst}}(\text{token}_{\text{owner}}(s) \mapsto s)\) \>

act3: \(\text{suspc}_{\text{inst}}(\text{token}_{\text{owner}}(s) \mapsto s) = \emptyset\) \>

END

RECONTACT_INST_OK: not extended ordinary \>

REFINES

RECONTACT_INST_OK

ANY

s \>
a service \(s\)

i \>
an instance \(i\)

WHERE

grd1: \(s \in \text{SERVICES}\) not theorem \>

grd2: \(i \in \text{PEERS}\) not theorem \>

grd3: \(\text{inst}_{\text{state}}(\text{token}_{\text{owner}}(s) \mapsto s) = \text{FAIL}_4\) not theorem \>

the state of \(s\) is SUSPICIOUS

grd4: \(\text{suspc}_{\text{peers}}(\text{token}_{\text{owner}}(s) \mapsto s) \neq \emptyset\) not theorem \>

the set of suspicious peers for \(s\) is not empty

grd5: \(i \in \text{suspc}_{\text{peers}}(\text{token}_{\text{owner}}(s) \mapsto s) \setminus \text{unav}_{\text{peers}}\) not theorem \>

\(i\) is a suspicious instance of \(s\) and \(i\) is available (can be contacted)

grd6: \(i \notin \text{rect}_{\text{inst}}(\text{token}_{\text{owner}}(s) \mapsto s)\) not theorem \>

the token owner of \(s\) has not yet tried to recontact \(i\)

grd7: \(\text{rect}_{\text{inst}}(\text{token}_{\text{owner}}(s) \mapsto s) \subseteq \text{suspc}_{\text{peers}}(\text{token}_{\text{owner}}(s) \mapsto s)\) not theorem \>

\(\text{rect}_{\text{inst}}(\text{token}_{\text{owner}}(s) \mapsto s)\) is the set of rectified instances for \(s\)
(s) ↦ s) not theorem  > the token owner of s has not yet tried to recontact all
the suspicious instances of s

THEN

act1: rect_inst(token_owner(s) ↦ s) = rect_inst(token_owner(s)

⇒ s) u {i}  > the token owner of s has tried to recontact i

act2: rctt_inst(token_owner(s) ↦ s) = rctt_inst(token_owner(s)

⇒ s) u {i}  > i is recontacted by the token owner of s successfully

END

RECONTACT_INST_KO:  not extended ordinary >

REFINES

RECONTACT_INST_KO

ANY

s  > a service s

i  > an instance i

WHERE

grd1: s ∈ SERVICES not theorem  > the state of s is SUSPICIOUS

grd2: i ∈ PEERS not theorem  > the set of suspicious peers for s is not empty

grd3: inst_state(token_owner(s) ↦ s) = FAIL_4 not theorem  > i is a suspicious instance of s and is unavailable (can not be

contacted)

grd4: suspc_peers(token_owner(s) ↦ s) ≠ ∅ not theorem  > rect_inst(token_owner(s) ↦ s) = suspc_peers(token_owner

(s) ⇒ s) not theorem  > the token owner of s has not yet tried to recontact all

the suspicious instances of s

THEN

act1: rect_inst(token_owner(s) ↦ s) = rect_inst(token_owner(s)

⇒ s) u {i}  > the token owner of s has tried to recontact i

END

FAIL_DETECT:  not extended ordinary >

REFINES

FAIL_DETECT

ANY

s  >

prop  >

WHERE

grd1: s ∈ SERVICES not theorem  >

grd2: prop ⊆ PEERS not theorem  >

grd3: inst_state(token_owner(s) ↦ s) = FAIL_4 not theorem  >

grd4: suspc_peers(token_owner(s) ↦ s) ≠ ∅ not theorem  >

grd5: rect_inst(token_owner(s) ↦ s) = suspc_peers(token_owner

(s) ⇒ s) not theorem  >
M16

\( \text{grd6: } \text{prop} = ((\text{run_peers}(s) \setminus \text{suspc_peers}(\text{token_owner}(s) \mapsto s)) \cup \text{rctt_inst}(\text{token_owner}(s) \mapsto s)) \setminus \text{unav_peers} \) not theorem \>

THEN

act1: \ \text{inst_state} = \text{inst_state} \leftarrow ((\text{prop} \times \{s\}) \times \{\text{FAIL_DETECT}_4\}) \>

act2: \ \text{suspc_peers}(\text{token_owner}(s) \mapsto s) = \text{suspc_peers} \setminus \text{rctt_inst}(\text{token_owner}(s) \mapsto s) \>

act3: \ \text{rect_inst}(\text{token_owner}(s) \mapsto s) = \emptyset \>

act4: \ \text{rctt_inst}(\text{token_owner}(s) \mapsto s) = \emptyset \>

END

\text{IS_OK: } \) not extended ordinary \>

REFINES \text{IS_OK} \>

ANY \ s \>

prop \>

WHERE

\( \text{grd1: } s \in \text{SERVICES} \) not theorem \>

\( \text{grd2: } \text{prop} \subseteq \text{PEERS} \) not theorem \>

\( \text{grd3: } \text{inst_state}(\text{token_owner}(s) \mapsto s) = \text{FAIL_DETECT}_4 \) not theorem \>

\( \text{grd4: } \text{suspc_peers}(\text{token_owner}(s) \mapsto s) = \emptyset \) not theorem \>

\( \text{grd5: } \text{prop} = \text{run_peers}(s) \setminus \text{unav_peers} \) not theorem \>

THEN

act1: \ \text{inst_state} = \text{inst_state} \leftarrow ((\text{prop} \times \{s\}) \times \{\text{RUN}_4\}) \>

END

\text{FAIL_ACTIV: } \) not extended ordinary \>

REFINES \text{FAIL_ACTIV} \>

ANY \ s \>

prop \>

WHERE

\( \text{grd1: } s \in \text{SERVICES} \) not theorem \>

\( \text{grd2: } \text{prop} \subseteq \text{PEERS} \) not theorem \>

\( \text{grd3: } \text{inst_state}(\text{token_owner}(s) \mapsto s) = \text{FAIL_DETECT}_4 \) not theorem \>

\( \text{grd4: } \text{suspc_peers}(\text{token_owner}(s) \mapsto s) \neq \emptyset \) not theorem \>

\( \text{grd5: } \text{prop} = \text{run_peers}(s) \setminus (\text{unav_peers} \cup \text{suspc_peers}(\text{token_owner}(s) \mapsto s)) \) not theorem \>

THEN

act1: \ \text{inst_state} = \text{inst_state} \leftarrow ((\text{prop} \times \{s\}) \times \{\text{FAIL_ACTIV}_4\}) \>

act2: \ \text{run_peers}(s) = \text{run_peers}(s) \setminus \text{suspc_peers}(\text{token_owner}(s) \mapsto s) \>

act3: \ \text{failr_peers}(s) = \text{failr_peers}(s) \cup \text{suspc_peers}(\text{token_owner}(s) \mapsto s) \>

Page 6
act4: \( \text{suspc_peers}(\text{token_owner}(s) \mapsto s) = \emptyset \) 

FAIL_CONFIGURE: not extended ordinary 

REFINES FAIL_CONFIGURE

ANY

\( s \) 

prop 

WHERE

\( s \in \text{SERVICES} \) not theorem

\( \text{prop} \subseteq \text{PEERS} \) not theorem

\( \text{inst_state}(\text{token_owner}(s) \mapsto s) = \text{FAIL_ACTIV}_4 \) not theorem

\( \text{card}(\text{run_peers}(s)) < \text{min_inst}(s) \) not theorem

\( \text{prop} = \text{run_peers}(s) \setminus \text{unav_peers} \) not theorem

THEN

act1: \( \text{inst_state} = \text{inst_state} \triangleright ((\text{prop}\times\{s\})\times\{\text{FAIL_CONFIG}_4\}) \)

FAIL_IGNORE: not extended ordinary 

REFINES FAIL_IGNORE

ANY

\( s \) 

prop 

WHERE

\( s \in \text{SERVICES} \) not theorem

\( \text{prop} \subseteq \text{PEERS} \) not theorem

\( \text{inst_state}(\text{token_owner}(s) \mapsto s) = \text{FAIL_ACTIV}_4 \) not theorem

\( \text{card}(\text{run_peers}(s)) \geq \text{min_inst}(s) \) not theorem

\( \text{prop} = \text{run_peers}(s) \setminus \text{unav_peers} \) not theorem

THEN

act1: \( \text{inst_state} = \text{inst_state} \triangleleft ((\text{prop}\times\{s\})\times\{\text{FAIL_IGN}_4\}) \)

IGNORE: not extended ordinary 

REFINES IGNORE

ANY

\( s \) 

prop 

WHERE

\( s \in \text{SERVICES} \) not theorem

\( \text{prop} \subseteq \text{PEERS} \) not theorem

\( \text{inst_state}(\text{token_owner}(s) \mapsto s) = \text{FAIL_IGN}_4 \) not theorem
M16

\[ \text{grd4: prop = run_peers(s) \setminus unav_peers not theorem} \]
THEN
\[ \text{act1: inst_state = inst_state \setminus ((prop \times \{s\}) \times \{\text{RUN}_4\})} \]
END

REDEPLOY_INSTC: not extended ordinary

REFINES
REDEPLOY_INSTC

ANY
\[ s \rightarrow \text{a service s} \]
\[ i \rightarrow \text{an instance i} \]
WHERE
\[ \text{grd1: } s \in \text{SERVICES not theorem} \]
\[ \text{grd2: } i \in \text{PEERS not theorem} \]
\[ \text{grd3: } i \notin \text{run_peers(s) \cup failr_peers(s) \cup unav_peers} \]
\[ \text{dep_instc(s) not theorem} \] i does not run s, is not failed for s, is not unavailable and is not already activated for s
\[ \text{grd4: } i \notin \text{actv_instc(token_owner(s) \mapsto s) not theorem} \]
\[ \text{grd5: } \text{inst_state(token_owner(s) \mapsto s) = FAIL\_CONFIG\_4 not theorem} \]
\[ \text{grd6: } \text{card(actv_instc(token_owner(s) \mapsto s)) < deplo_inst(s) not theorem} \]
\[ \text{grd7: } \text{card(dep_instc(s)) + card(run_peers(s)) < min_inst(s) not theorem} \]
THEN
\[ \text{act1: actv_instc(token_owner(s) \mapsto s) = actv_instc(token_owner(s) \mapsto s) \cup \{i\}} \]
END

REDEPLOY_INSTS: not extended ordinary

REFINES
REDEPLOY_INSTS

ANY
\[ s \rightarrow \]
WHERE
\[ \text{grd1: } s \in \text{SERVICES not theorem} \]
\[ \text{grd2: } \text{card(actv_instc(token_owner(s) \mapsto s)) = deplo_inst(s) not theorem} \]
\[ \text{grd3: } \text{card(dep_instc(s)) + card(run_peers(s)) < min_inst(s) not theorem} \]
\[ \text{grd4: } \text{inst_state(token_owner(s) \mapsto s) = FAIL\_CONFIG\_4 not theorem} \]
THEN
\[ \text{act1: dep_instc(s) = dep_instc(s) \cup actv_instc(token_owner(s) \mapsto s)} \]
\[ \text{act2: actv_instc(token_owner(s) \mapsto s) = \emptyset} \]
END
M16

REDEPLOY: not extended ordinary
REFINES REDEPLOY
ANY
  s
  prop
WHERE
  grd1: s ∈ SERVICES not theorem
  grd2: prop ⊆ PEERS not theorem
  grd3: inst_state(token_owner(s) ↦ s) = FAIL_CONFIG_4 not theorem
  grd4: actv_instc(token_owner(s) ↦ s)=∅ not theorem
  grd5: dep_instc(s) ≠ ∅ not theorem
  grd6: card(run_peers(s))+card(dep_instc(s)) ≥ min_inst(s) not theorem
THEN
  act1: inst_state≔ inst_state  ((prop×{s})×{DPL_4})
  act2: run_peers(s) ≔ run_peers(s) ∪ dep_instc(s)
  act3: dep_instc(s) ≔ ∅
END

HEAL: not extended ordinary
REFINES HEAL
ANY
  s
  prop
WHERE
  grd1: s ∈ SERVICES not theorem
  grd2: prop ⊆ PEERS not theorem
  grd3: inst_state(token_owner(s) ↦ s) = DPL_4 not theorem
  grd4: prop = run_peers(s)∖unav_peers not theorem
THEN
  act1: inst_state≔ inst_state  ((prop×{s})×{RUN_4})
END

UNFAIL_PEER: extended ordinary
REFINES UNFAIL_PEER
ANY
  s
  p
WHERE
  grd1: s ∈ SERVICES not theorem
  grd2: p ∈ PEERS not theorem
  grd3: p ∈ failr_peers(s) not theorem
THEN
act1: failr_peers(s) = failr_peers(s) \{p\} 

END

MAKE_PEER_AVAIL: extended ordinary 
REFINES MAKE_PEER_AVAIL
ANY 
p 
WHERE 
grd1: p \in PEERS \not \text{ theorem} 
grd2: p \in unav_peers \not \text{ theorem} 
THEN 
act1: unav_peers = unav_peers \{p\} 
END

END
MACHINE M17
REFINES M16
SEES C09
VARIABLES
  run_peers
  suspct_peers
  failr_peers
  dep_instc
  token_owner
  unav_peers
  suspc_inst
  rect_inst  \(\text{instances that are tried to be recontacted}\)
  rctt_inst  \(\text{instances effectively recontacted after a try}\)
  actv_instc  \(\text{instances activated by token owners}\)
  inst_state

INVARIANTS
inv1:  suspct_peers ∈ (PEERS×SERVICES) ⇸ ℙ(PEERS) not theorem
inv2:  \(\forall s \cdot s ∈ SERVICES \Rightarrow \text{token_owner}(s) ∈ \text{dom(suspc_peers)}\) not theorem
gluing_susp_1:  \(\forall s \cdot s ∈ SERVICES \Rightarrow \text{suspc_peers}(\text{token_owner}(s) ↦ s) =\)
               \(\text{suspc_peers}(\text{token_owner}(s) ↦ s)\) not theorem

EVENTS
INITIALISATION:  not extended ordinary
  THEN
    act1:  run_peers ≔ InitSrvcPeers
    act2:  suspct_peers ≔ InitSuspPrs
    act3:  failr_peers ≔ InitFail
    act4:  dep_instc ≔ InitFail
    act5:  token_owner ≔ init_tok
    act6:  unav_peers ≔ ∅
    act7:  suspc_inst ≔ InitSuspPeers
    act8:  rect_inst ≔ InitSuspPeers
    act9:  rctt_inst ≔ InitSuspPeers
    act10: actv_instc ≔ InitSuspPeers
    act11: inst_state ≔ InitStateSrv
  END

MAKE_PEER_UNAVAIL:  not extended ordinary
REFINES MAKE_PEER_UNAVAIL
ANY
  prs  \(\text{Peers that will become unavailable}\)
  E  \(\text{Values for token owner per service}\)
WHERE
  grd1:  prs ∈ PEERS not theorem
grd2: \[ \text{prs \not\subseteq \text{unav_peers}} \not\text{theorem} \Rightarrow \text{the peers in prs are not yet unavailable} \]

grd3: \[
\forall \text{srv} \cdot \text{srv} \in \text{SERVICES} \Rightarrow \text{dom(dom(inst_state) \setminus \{srv\})} \]

\[ \text{prs} \neq \emptyset \not\text{theorem} \Rightarrow \text{for each service srv, there must always be at least 1 peer available} \]

grd4: \[ E \in \text{SERVICES} \rightarrow \text{PEERS} \not\text{theorem} \Rightarrow \text{Value for token owner per service} \]

grd5: \[
\forall \text{srv} \cdot \text{srv} \in \text{SERVICES} \land \text{token_owner(srv)} \notin \text{prs} \Rightarrow
E(\text{srv}) = \text{token_owner(srv)} \not\text{theorem} \Rightarrow \text{If the token owner of a service srv does not belong to prs, the token owner is not changed} \]

grd6: \[
\forall \text{srv} \cdot \text{srv} \in \text{SERVICES} \land \text{token_owner(srv)} \in \text{prs} \Rightarrow
E(\text{srv}) \in \text{run_peers(srv) \setminus (unav_peers u prs u failr_peers (srv) \cup \text{suspct_peers(token_owner(srv) ↦ srv)) \setminus \text{dom(inst_state) \setminus E(srv) ↦ srv) \subseteq \text{dom}} (suspct_peers) \land \text{inst_state(E(srv) ↦ srv) = inst_state(token_owner(srv) ↦ srv)}
\]

A new token owner is chosen: the new token owner must have same characteristics as the previous one (state, list of suspicious neighbours, etc.), and it must not be an unavailable, suspicious, failed peer or a member of prs

WITH

\[ p_s := \text{E} \setminus \text{suspct_peers} \]

\[ \text{rc_s} := \text{E} \setminus \text{rect_inst} = ((\text{E} \setminus \text{token_owner}) \setminus \{\emptyset\}) \]

\[ \text{s_i} := ((\text{prs} \times \text{SERVICES}) \setminus \text{suspc_inst}) = ((\text{E} \setminus \text{token_owner}) \setminus \{\emptyset\}) \]

\[ \text{rt_s} := ((\text{prs} \times \text{SERVICES}) \setminus \text{rcct_inst}) = ((\text{E} \setminus \text{token_owner}) \setminus \{\emptyset\}) \]

\[ \text{ac_i} := ((\text{prs} \times \text{SERVICES}) \setminus \text{actv_instc}) = ((\text{E} \setminus \text{token_owner}) \setminus \{\emptyset\}) \]

THEN

\[ \text{act1: unav_peers = unav_peers u prs} \Rightarrow \text{the peers in prs become unavailable} \]

\[ \text{act2: token_owner := token_owner \setminus E} \Rightarrow \text{new values for token owner per service} \]

\[ \text{act3: rect_inst := ((prs \times \text{SERVICES}) \setminus \text{rect_inst}) \setminus ((\text{E} \setminus \text{token_owner}) \setminus \{\emptyset\})} \Rightarrow \text{the peers in prs can not try to recontact instances} \]
act4: \[ \text{rctt\_inst} = ((\text{prs} \times \text{SERVICES}) \leftarrow \text{rctt\_inst})\times ((E \setminus \text{token\_owner}) \sim) \times \{\emptyset\} \] theorem the peers in prs can not try to recontact instances anymore (2)

act5: \[ \text{actv\_instc} = ((\text{prs} \times \text{SERVICES}) \leftarrow \text{actv\_instc})\times ((E \setminus \text{token\_owner}) \sim) \times \{\emptyset\} \] theorem the peers in prs can not activate instances anymore (1)

act6: \[ \text{suspct\_peers} = ((\text{prs} \times \text{SERVICES}) \leftarrow \text{suspct\_peers}) \times \text{the peers in prs can not suspect instances anymore (1)} \]

act7: \[ \text{suspc\_inst} = ((\text{prs} \times \text{SERVICES}) \leftarrow \text{suspc\_inst})\times ((E \setminus \text{token\_owner}) \sim) \times \{\emptyset\} \] theorem the peers in prs can not suspect instances anymore (2)

act8: \[ \text{inst\_state} = (\text{prs} \times \text{SERVICES}) \leftarrow \text{inst\_state} \times \text{the peers in prs can not monitor the state of the services provided anymore} \]

END

SUSPECT\_INST: extended ordinary

REFINES SUSPECT\_INST

ANY

s \[\rightarrow\] a service \(s\)

susp \[\rightarrow\] suspicious instances

WHERE

grd1: \[ s \in \text{SERVICES} \] not theorem \(\rightarrow\)

grd2: \[ \text{susp} \subset \text{PEERS} \] not theorem \(\rightarrow\)

grd3: \[ \text{susp} = \text{run\_peers}(s) \cap \text{unav\_peers} \] not theorem \(\rightarrow\) instances in susp are suspicious if the peers running them becomes unavailable

grd4: \[ \text{suspc\_inst}(\text{token\_owner}(s) \mapsto s) = \emptyset \] not theorem \(\rightarrow\) the member of susp have not yet been suspected for \(s\) by the token owner of \(s\)

grd5: \[ \text{inst\_state}(\text{token\_owner}(s) \mapsto s) = \text{RUN\_4} \] not theorem \(\rightarrow\) the state of \(s\) is OK

THEN

act1: \[ \text{suspc\_inst}(\text{token\_owner}(s) \mapsto s) = \text{susp} \] theorem the members of susp become suspected instances for \(s\) by the token owner of \(s\)

END

FAIL: not extended ordinary

REFINES FAIL

ANY

s \[\rightarrow\] a service \(s\)

prop \[\rightarrow\]

WHERE

grd1: \[ s \in \text{SERVICES} \] not theorem \(\rightarrow\)

grd2: \[ \text{prop} \subset \text{PEERS} \] not theorem \(\rightarrow\)

grd3: \[ \text{inst\_state}(\text{token\_owner}(s) \mapsto s) = \text{RUN\_4} \] not theorem \(\rightarrow\)

grd4: \[ \text{suspc\_inst}(\text{token\_owner}(s) \mapsto s) \neq \emptyset \] not theorem \(\rightarrow\)

grd5: \[ \text{prop} = \text{run\_peers}(s) \setminus (\text{suspc\_inst}(\text{token\_owner}(s) \mapsto s) \cup \text{unav\_peers}) \] not theorem \(\rightarrow\)
THEN

act1: \( \text{inst\_state} := \text{inst\_state} \circ ((\text{prop}\times\{s\})\times\{\text{FAIL\_4}\}) \)
act2: \( \text{suspct\_peers} := \text{suspct\_peers} \circ ((\text{prop}\times\{s\})\times\{\text{susp\_inst}\}) \)
act3: \( \text{susp\_inst}(\text{token\_owner}(s) \mapsto s) = \emptyset \)

RECONTACT\_INST\_OK: 
END

RECONTACT\_INST\_OK: not extended ordinary
REFINES
RECONTACT\_INST\_OK
ANY
\( s \rightarrow \) a service\( s \)
\( i \rightarrow \) an instance\( i \)
WHERE

grd1: \( s \in \text{SERVICES} \) not theorem
grd2: \( i \in \text{PEERS} \) not theorem
grd3: \( \text{inst\_state}(\text{token\_owner}(s) \mapsto s) = \text{FAIL\_4} \) not theorem

state of\( s \) is SUSPICIOUS
grd4: \( \text{suspct\_peers}(\text{token\_owner}(s) \mapsto s) \neq \emptyset \) not theorem

set of suspicious peers for\( s \) is not empty
grd5: \( i \in \text{suspct\_peers}(\text{token\_owner}(s) \mapsto s)\setminus\text{unav\_peers} \) not theorem

\( i \) is a suspicious instance of\( s \) and is available (can be contacted)
grd6: \( i \in \text{rect\_inst}(\text{token\_owner}(s) \mapsto s) \) not theorem

the token owner of\( s \) has not yet tried to recontact\( i \)
grd7: \( \text{rect\_inst}(\text{token\_owner}(s) \mapsto s) \subseteq \text{suspct\_peers}(\text{token\_owner}(s) \mapsto s) \) not theorem

THEN

act1: \( \text{rect\_inst}(\text{token\_owner}(s) \mapsto s) = \text{rect\_inst}(\text{token\_owner}(s) \mapsto s) \cup \{i\} \)
act2: \( \text{rctt\_inst}(\text{token\_owner}(s) \mapsto s) = \text{rctt\_inst}(\text{token\_owner}(s) \mapsto s) \cup \{i\} \)

i is recontacted by the token owner of\( s \) successfully

END

RECONTACT\_INST\_KO: not extended ordinary
REFINES
RECONTACT\_INST\_KO
ANY
\( s \rightarrow \) a service\( s \)
\( i \rightarrow \) an instance\( i \)
WHERE

grd1: \( s \in \text{SERVICES} \) not theorem
grd2: \( i \in \text{PEERS} \) not theorem
grd3: \( \text{inst\_state}(\text{token\_owner}(s) \mapsto s) = \text{FAIL\_4} \) not theorem

state of\( s \) is SUSPICIOUS
grd4: \( \text{suspct\_peers}(\text{token\_owner}(s) \mapsto s) \neq \emptyset \) not theorem

set of suspicious peers for\( s \) is not empty
grd5: \( i \in \text{suspct\_peers}(\text{token\_owner}(s) \mapsto s)\setminus\text{unav\_peers} \) not theorem
theorem → i is a suspicious instance of s and is unavailable (can not be contacted)
  grd6:  i ∈ rect_inst(token_owner(s) ↦ s) not theorem → the token owner of s has not yet tried to recontact i
  grd7:  rect_inst(token_owner(s) ↦ s) ⊂ suspct_peers(token_owner(s) ↦ s) not theorem → the token owner of s has not yet tried to recontact all the suspicious instances of s
  THEN
    act1:  rect_inst(token_owner(s) ↦ s) = rect_inst(token_owner(s) ↦ s) ∪ {i} → the token owner of s has tried to recontact i
  END

FAIL_DETECT: not extended ordinary
REFINES FAIL_DETECT
ANY
  s
  prop
  susp
WHERE
  grd1:  s ∈ SERVICES not theorem
  grd2:  prop ⊆ PEERS not theorem
  grd7:  susp ⊆ PEERS not theorem
  grd3:  inst_state(token_owner(s) ↦ s) = FAIL_4 not theorem
  grd4:  suspct_peers(token_owner(s) ↦ s) ≠ ∅ not theorem
  grd5:  rect_inst(token_owner(s) ↦ s) = suspct_peers(token_owner(s) ↦ s)
  grd6:  prop = ((run_peers(s) \ suspct_peers(token_owner(s) ↦ s)) ∪ rctt_inst(token_owner(s) ↦ s) \ unav_peers not theorem
  grd8:  susp = suspct_peers(token_owner(s) ↦ s) \ rctt_inst(token_owner(s) ↦ s) not theorem
  THEN
    act1:  inst_state = inst_state ∪ ((prop × {s}) × {FAIL_DETECT_4})
    act2:  suspct_peers = suspct_peers ∪ ((prop × {s}) × {susp})
    act3:  rect_inst(token_owner(s) ↦ s) = ∅
    act4:  rctt_inst(token_owner(s) ↦ s) = ∅
  END

IS_OK: not extended ordinary
REFINES IS_OK
ANY
  s
  prop
WHERE
  grd1:  s ∈ SERVICES not theorem
  grd2:  prop ⊆ PEERS not theorem

Page 5
grd3: inst_state(token_owner(s) ↦ s) = FAIL_DETECT_4 not theorem

grd4: suspct_peers(token_owner(s) ↦ s) = ∅ not theorem

grd5: prop = run_peers(s)\unav_peers not theorem

THEN

act1: inst_state = inst_state × ((prop×{s})×{RUN_4})

END

FAIL_ACTIV: not extended ordinary

REFINES

FAIL_ACTIV

ANY

s

prop

WHERE

grd1: s ∈ SERVICES not theorem

grd2: prop ⊆ PEERS not theorem

grd3: inst_state(token_owner(s) ↦ s) = FAIL_DETECT_4 not theorem

THEN

act1: inst_state = inst_state × ((prop×{s})×{FAIL_ACTIV_4})

act2: run_peers(s) = run_peers(s) \ suspct_peers(token_owner(s) ↦ s)

act3: failr_peers(s) = failr_peers(s) ∪ suspct_peers(token_owner(s) ↦ s)

act4: suspct_peers = suspct_peers × ((prop×{s})×{∅})

END

FAIL_CONFIGURE: extended ordinary

REFINES

FAIL_CONFIGURE

ANY

s

prop

WHERE

grd1: s ∈ SERVICES not theorem

grd2: prop ⊆ PEERS not theorem

grd3: inst_state(token_owner(s) ↦ s) = FAIL_ACTIV_4 not theorem

THEN

act1: inst_state = inst_state × ((prop×{s})×{FAIL_CONFIG_4})

act2: card(run_peers(s)) < min_inst(s) not theorem

act3: prop = run_peers(s)\unav_peers not theorem

THEN

act1: inst_state = inst_state × ((prop×{s})×{FAIL_CONFIG_4})

END

Page 6
FAIL_IGNORE: extended ordinary
  \text{REFINES} \text{FAIL_IGNORE}
  \text{ANY} \ s \ \text{\\rightarrow} \ prop \ \text{WHERE}
  \text{grd1: } s \in \text{SERVICES not theorem} \ \text{\\rightarrow} \ prop \subseteq \text{PEERS not theorem} \ \text{\\rightarrow} \ \text{inst}_{\text{state}}(\text{token}_{\text{owner}}(s) \Rightarrow s) = \text{FAIL_ACTIV}_4 \ \text{not theorem} \ \text{\\rightarrow} \ \text{card}(\text{run}_{\text{peers}}(s)) \geq \text{min}_{\text{inst}}(s) \ \text{not theorem} \ \text{\\rightarrow} \ \text{prop} = \text{run}_{\text{peers}}(s) \setminus \text{unav}_{\text{peers}} \ \text{not theorem} \ \text{\\rightarrow} \ \text{THEN}
  \text{act1: } \text{inst}_{\text{state}} = \text{inst}_{\text{state}} \Rightarrow ((\text{prop} \times \{s\}) \times \{\text{FAIL_IGN}_4\}) \ \text{\\rightarrow} \ \text{END}

IGNORE: extended ordinary
  \text{REFINES} \text{IGNORE}
  \text{ANY} \ s \ \text{\\rightarrow} \ prop \ \text{WHERE}
  \text{grd1: } s \in \text{SERVICES not theorem} \ \text{\\rightarrow} \ prop \subseteq \text{PEERS not theorem} \ \text{\\rightarrow} \ \text{inst}_{\text{state}}(\text{token}_{\text{owner}}(s) \Rightarrow s) = \text{FAIL_IGN}_4 \ \text{not theorem} \ \text{\\rightarrow} \ \text{THEN}
  \text{act1: } \text{inst}_{\text{state}} = \text{inst}_{\text{state}} \Rightarrow ((\text{prop} \times \{s\}) \times \{\text{RUN}_4\}) \ \text{\\rightarrow} \ \text{END}

REDEPLOY_INSTC: extended ordinary
  \text{REFINES} \text{REDEPLOY_INSTC}
  \text{ANY} \ s \ \text{\\rightarrow} \text{a service s} \ i \ \text{\\rightarrow} \text{an instance i}
  \text{WHERE}
  \text{grd1: } s \in \text{SERVICES not theorem} \ \text{\\rightarrow} \ i \in \text{PEERS not theorem} \ \text{\\rightarrow} \ i \notin \text{run}_{\text{peers}}(s) \cup \text{failr}_{\text{peers}}(s) \cup \text{unav}_{\text{peers}} \cup \text{dep}_{\text{instc}}(s) \ \text{not theorem} \ \text{\\rightarrow} \ i \ does \ not \ run \ s, \ is \ not \ failed \ for \ s, \ is \ not \ unavailable \ and \ is \ not \ already \ activated \ for \ s \ \text{\\rightarrow} \ \text{inst}_{\text{state}}(\text{token}_{\text{owner}}(s) \Rightarrow s) = \text{FAIL_CONFIG}_4 \ \text{not theorem} \ \text{\\rightarrow} \ \text{END}
M17

\[ \text{grd6}: \quad \text{card}(\text{actv}\_\text{instc}(\text{token}\_\text{owner}(s) \mapsto s)) < \text{deplo}\_\text{inst}(s) \not\text{ theorem} \]

\[ \text{grd7}: \quad \text{card}(\text{dep}\_\text{inst}(s)) + \text{card}(\text{run}\_\text{peers}(s)) < \text{min}\_\text{inst}(s) \not\text{ theorem} \]

\[ \text{THEN} \]

\[ \text{act1}: \quad \text{actv}\_\text{instc}(\text{token}\_\text{owner}(s) \mapsto s) = \text{actv}\_\text{instc}(\text{token}\_\text{owner}(s) \mapsto s) \cup \{i\} \] END

\[ \text{REDEPLOY}\_\text{INSTS}: \quad \text{extended ordinary} \]
\[ \text{REFINES} \]
\[ \text{REDEPLOY}\_\text{INSTS} \]
\[ \text{ANY} \quad s \quad \text{WHERE} \]

\[ \text{grd1}: \quad s \in \text{SERVICES} \not\text{ theorem} \]

\[ \text{grd2}: \quad \text{card}(\text{actv}\_\text{instc}(\text{token}\_\text{owner}(s) \mapsto s)) = \text{deplo}\_\text{inst}(s) \not\text{ theorem} \]

\[ \text{grd3}: \quad \text{card}(\text{dep}\_\text{inst}(s)) + \text{card}(\text{run}\_\text{peers}(s)) < \text{min}\_\text{inst}(s) \not\text{ theorem} \]

\[ \text{grd4}: \quad \text{inst}\_\text{state}(\text{token}\_\text{owner}(s) \mapsto s) = \text{FAIL}\_\text{CONFIG}_4 \not\text{ theorem} \]

\[ \text{THEN} \]

\[ \text{act1}: \quad \text{dep}\_\text{instc}(s) = \text{dep}\_\text{instc}(s) \cup \text{actv}\_\text{instc}(\text{token}\_\text{owner}(s) \mapsto s) \]

\[ \text{act2}: \quad \text{actv}\_\text{instc}(\text{token}\_\text{owner}(s) \mapsto s) = \emptyset \] END

\[ \text{REDEPLOY}: \quad \text{extended ordinary} \]
\[ \text{REFINES} \]
\[ \text{REDEPLOY} \]
\[ \text{ANY} \quad s \quad \text{WHERE} \]

\[ \text{grd1}: \quad s \in \text{SERVICES} \not\text{ theorem} \]

\[ \text{grd2}: \quad \text{prop} \subseteq \text{PEERS} \not\text{ theorem} \]

\[ \text{grd3}: \quad \text{inst}\_\text{state}(\text{token}\_\text{owner}(s) \mapsto s) = \text{FAIL}\_\text{CONFIG}_4 \not\text{ theorem} \]

\[ \text{grd4}: \quad \text{actv}\_\text{instc}(\text{token}\_\text{owner}(s) \mapsto s) = \emptyset \not\text{ theorem} \]

\[ \text{grd5}: \quad \text{dep}\_\text{instc}(s) \neq \emptyset \not\text{ theorem} \]

\[ \text{grd6}: \quad \text{card}(\text{run}\_\text{peers}(s)) + \text{card}(\text{dep}\_\text{instc}(s)) \geq \text{min}\_\text{inst}(s) \not\text{ theorem} \]

\[ \text{grd7}: \quad \text{prop} = (\text{run}\_\text{peers}(s) \setminus \text{unav}\_\text{peers}) \not\text{ theorem} \]

\[ \text{THEN} \]

\[ \text{act1}: \quad \text{inst}\_\text{state}(s) = (\text{prop} \times s) \setminus \text{DPL}_4 \]

\[ \text{act2}: \quad \text{run}\_\text{peers}(s) = \text{run}\_\text{peers}(s) \cup \text{dep}\_\text{instc}(s) \]

\[ \text{act3}: \quad \text{dep}\_\text{instc}(s) = \emptyset \]

Page 8
HEAL: extended ordinary

REFINES
HEAL

ANY
s
prop

WHERE
grd1: s ∈ SERVICES not theorem
grd2: prop \subseteq PEERS not theorem
grd3: inst_state(token_owner(s) ↦ s) = DPL_4 not theorem
grd4: prop = run_peers(s)∖unav_peers not theorem

THEN
act1: inst_state≔ inst_state  ((prop×{s})×{RUN_4})

END

UNFAIL_PEER: extended ordinary

REFINES
UNFAIL_PEER

ANY
s
p

WHERE
grd1: s ∈ SERVICES not theorem
grd2: p ∈ PEERS not theorem
grd3: p ∈ failr_peers(s) not theorem

THEN
act1: failr_peers(s) ≔ failr_peers(s)∖{p}

END

MAKE_PEER_AVAIL: extended ordinary

REFINES
MAKE_PEER_AVAIL

ANY
p

WHERE
grd1: p ∈ PEERS not theorem
grd2: p ∈ unav_peers not theorem

THEN
act1: unav_peers ≔ unav_peers ∖ {p}

END

END
MACHINE M18
REFINES M17
SEES C09
VARIABLES
  run_inst
  suspct_peers
  failr_peers
  dep_instc
  token_owner
  unav_peers
  suspc_inst
  rctt_inst
  actv_instc
  inst_state

INVARINTS
inv1: run_inst ∈ (PEERS×SERVICES) ⇸ ℙ(PEERS) not theorem
inv2: ∀ s · s ∈ SERVICES ⇒ token_owner(s) ↦ s ∈ dom(run_inst) not theorem
  gluing_run_1: ∀ s · s ∈ SERVICES ⇒ run_inst(token_owner(s) ↦ s) = run_peers(s) not theorem

EVENTS
INITIALISATION: not extended ordinary
  THEN
    act1: run_inst = InitRunPeers
    act2: suspct_peers = InitSuspPrs
    act3: failr_peers = InitFail
    act4: dep_instc = InitFail
    act5: token_owner = init_tok
    act6: unav_peers = ∅
    act7: suspc_inst = InitSuspPeers
    act8: rect_inst = InitSuspPeers
    act9: rctt_inst = InitSuspPeers
    act10: actv_instc = InitSuspPeers
    act11: inst_state = InitStateSrv
  END

MAKE_PEER_UNAVAIL: not extended ordinary
  REFINES MAKE_PEER_UNAVAIL
  ANY
    prs >Peers that will become unavailable
    E >Values for token owner per service
  WHERE
    grd1: prs ⊆ PEERS not theorem
\begin{align*}
\text{grd2: } & \text{ prs} \not\subseteq \text{unav_peers} \not\equiv \text{the peers in prs are not yet unavailable} \\
\text{grd3: } & \forall \text{srv} \cdot \text{srv} \in \text{SERVICES} \Rightarrow \text{dom(}\text{dom(}\text{inst_state})\text{)} \not\subseteq \{\text{srv}\} \\
\text{\textbackslash prs} \not\subseteq \emptyset \not\equiv \text{for each service srv, there must always be at least 1 peer available} \\
\text{grd4: } & E \in \text{SERVICES} \rightarrow \text{PEERS} \not\equiv \text{Value for token owner per service} \\
\text{grd5: } & \forall \text{srv} \cdot \text{srv} \in \text{SERVICES} \land \text{token_owner(srv)} \not\in \text{prs} \Rightarrow E(\text{srv}) = \text{token_owner(srv)} \not\equiv \text{If the token owner of a service srv does not belong to prs, the token owner is not changed} \\
\text{grd6: } & \forall \text{srv} \cdot \text{srv} \in \text{SERVICES} \land \text{token_owner(srv)} \in \text{prs} \\
& \Rightarrow \quad E(\text{srv}) \in \text{run_inst}(\text{token_owner(srv)} \mapsto \text{srv}) \not\subseteq \{\text{unav_peers} \cup \text{prs} \cup \text{failr_peers(srv)} \cup \text{suspt_peers(token_owner(srv)} \mapsto \text{srv})\} \not\Rightarrow \quad E(\text{srv}) \not\subseteq \text{dom(}\text{inst_state}) \not\subseteq \text{dom(suspt_peers) \land dom(}\text{run_inst}) \\
& \Rightarrow \quad \text{run_inst}(E(\text{srv}) \mapsto \text{srv}) = \text{run_inst}(\text{token_owner(srv)} \mapsto \text{srv}) \\
& \Rightarrow \quad \text{inst_state}(E(\text{srv}) \mapsto \text{srv}) = \text{inst_state}(\text{token_owner(srv)} \mapsto \text{srv}) \\
& \Rightarrow \quad \text{suspt_peers}(E(\text{srv}) \mapsto \text{srv}) = \text{suspt_peers}(\text{token_owner(srv)} \mapsto \text{srv}) \\
\text{act1: } & \text{unav_peers} = \text{unav_peers} \cup \text{prs} \not\equiv \text{the peers in prs become unavailable} \\
\text{act2: } & \text{token_owner} = \text{token_owner} \not\subseteq E \not\equiv \text{new values for token owner per service} \\
\text{act3: } & \text{rect_inst} = (\text{prs} \times \text{SERVICES}) \not\subseteq \text{rect_inst} \not\equiv \text{the peers in prs can not try to recontact instances anymore (1)} \\
\text{act4: } & \text{rctt_inst} = (\text{prs} \times \text{SERVICES}) \not\subseteq \text{rctt_inst} \not\equiv \text{the peers in prs can not try to recontact instances anymore (2)} \\
\text{act5: } & \text{actv_instc} = (\text{prs} \times \text{SERVICES}) \not\subseteq \text{actv_instc} \not\equiv \text{the peers in prs can not activate_instances anymore} \\
\text{act6: } & \text{suspt_peers} = (\text{prs} \times \text{SERVICES}) \not\subseteq \text{suspt_peers} \not\equiv \text{the peers in prs can not suspect instances anymore (1)} \\
\text{act7: } & \text{suspc_inst} = (\text{prs} \times \text{SERVICES}) \not\subseteq \text{suspc_inst} \not\equiv \text{the peers in prs can not suspect instances anymore (2)}
\end{align*}
(((E\token_owner\¬)×{∅}) \rightarrow \text{the peers in prs can not suspect instances anymore (2)}

act8: \text{inst\_state} = (\text{prs}×\text{SERVICES}) \leftarrow \text{inst\_state} \rightarrow \text{the peers in prs can not monitor the state of the services provided anymore}

act9: \text{run\_inst} = (\text{prs}×\text{SERVICES}) \leftarrow \text{run\_inst} \rightarrow

\text{END}

\text{SUSPECT\_INST: not extended ordinary} \rightarrow
\text{REFINES}
\text{SUSPECT\_INST}
\text{ANY}
\text{s} \rightarrow \text{a service s}
\text{susp} \rightarrow \text{suspicious instances}
\text{WHERE}
\text{grd1: s \in \text{SERVICES} not theorem} \rightarrow
\text{grd2: susp \subseteq \text{PEERS} not theorem} \rightarrow
\text{grd3: susp = run\_inst(token\_owner(s) \mapsto s) \cap \text{unav\_peers} not theorem} \rightarrow \text{instances in susp are suspicious if the peers running them becomes unavailable}
\text{grd4: suspc\_inst(token\_owner(s) \mapsto s) = \emptyset not theorem} \rightarrow \text{the member of susp have not yet been suspected for s by the token owner of s}
\text{grd5: inst\_state(token\_owner(s) \mapsto s) = \text{RUN\_4} not theorem} \rightarrow \text{the state of s is OK}
\text{grd6: susp \neq \emptyset not theorem} \rightarrow
\text{THEN}
\text{act1: suspc\_inst(token\_owner(s) \mapsto s) = susp} \rightarrow \text{the members of susp become suspected instances for s by the token owner of s}
\text{END}

\text{FAIL: not extended ordinary} \rightarrow
\text{REFINES}
\text{FAIL}
\text{ANY}
\text{s} \rightarrow
\text{prop} \rightarrow
\text{WHERE}
\text{grd1: s \in \text{SERVICES} not theorem} \rightarrow
\text{grd2: prop \subseteq \text{PEERS} not theorem} \rightarrow
\text{grd3: inst\_state(token\_owner(s) \mapsto s) = \text{RUN\_4} not theorem} \rightarrow
\text{grd4: suspc\_inst(token\_owner(s) \mapsto s) \neq \emptyset not theorem} \rightarrow
\text{grd5: prop = run\_inst(token\_owner(s) \mapsto s) \setminus (suspc\_inst (token\_owner(s) \mapsto s) \cup \text{unav\_peers}) not theorem} \rightarrow
\text{THEN}
\text{act1: inst\_state = inst\_state \leftarrow ((prop×\{s\})×\{\text{FAIL\_4}\})} \rightarrow
\text{act2: suspct\_peers = suspct\_peers \leftarrow ((prop×\{s\})×\{suspc\_inst \text{(token\_owner(s) \mapsto s)})})} \rightarrow
\text{act3: suspc\_inst(token\_owner(s) \mapsto s) = \emptyset} \rightarrow
\text{END}
RECONTACT_INST_OK: extended ordinary
REFINES
RECONTACT_INST_OK

ANY
s → a service s
i → an instance i

WHERE
grd1: s ∈ SERVICES not theorem
grd2: i ∈ PEERS not theorem
grd3: inst_state(token_owner(s) ↦ s) = FAIL_4 not theorem
the state of s is SUSPICIOUS
grd4: suspct_peers(token_owner(s) ↦ s) ≠ ø not theorem
the set of suspicious peers for s is not empty
grd5: i ∈ suspct_peers(token_owner(s) ↦ s)\unav_peers not theorem
i is a suspicious instance of s and is available (can be contacted)
grd6: i ∉ rect_inst(token_owner(s) ↦ s) not theorem
the token owner of s has not yet tried to recontact i
grd7: rect_inst(token_owner(s) ↦ s) ⊂ suspct_peers(token_owner(s) ↦ s) not theorem
the token owner of s has not yet tried to recontact all the suspicious instances of s

THEN
act1: rect_inst(token_owner(s) ↦ s) = rect_inst(token_owner(s) ↦ s) ∪ {i}
the token owner of s has tried to recontact i
act2: rctt_inst(token_owner(s) ↦ s) = rctt_inst(token_owner(s) ↦ s) ∪ {i}
i is recontacted by the token owner of s successfully

END

RECONTACT_INST_KO: extended ordinary
REFINES
RECONTACT_INST_KO

ANY
s → a service s
i → an instance i

WHERE
grd1: s ∈ SERVICES not theorem
grd2: i ∈ PEERS not theorem
grd3: inst_state(token_owner(s) ↦ s) = FAIL_4 not theorem
the state of s is SUSPICIOUS
grd4: suspct_peers(token_owner(s) ↦ s) ≠ ø not theorem
the set of suspicious peers for s is not empty
grd5: i ∈ suspct_peers(token_owner(s) ↦ s)∩unav_peers not theorem
i is a suspicious instance of s and is unavailable (can not be contacted)
grd6: i ∉ rect_inst(token_owner(s) ↦ s) not theorem
the token owner of s has not yet tried to recontact i
grd7: rect_inst(token_owner(s) ↦ s) ⊂ suspct_peers(token_owner(s) ↦ s) not theorem
the token owner of s has not yet tried to recontact all the suspicious instances of s
THEN
act1: \(\text{rect\_inst}(\text{token\_owner}(s) \redir s) = \text{rect\_inst}(\text{token\_owner}(s) \redir s) \cup \{i\}\)  
> the token owner of \(s\) has tried to recontact \(i\)
END

FAIL_DETECT:  

\[\text{REFINES}\]

FAIL_DETECT

ANY
prop >
susp >

WHERE

\(\text{grd1}: s \in \text{SERVICES} \not\text{theorem}\)  
\(\text{grd2}: \text{prop} \subseteq \text{PEERS} \not\text{theorem}\)  
\(\text{grd7}: \text{susp} \subseteq \text{PEERS} \not\text{theorem}\)  
\(\text{grd3}: \text{inst\_state}(\text{token\_owner}(s) \redir s) = \text{FAIL\_4} \not\text{theorem}\)  
\(\text{grd4}: \text{suspect\_peers}(\text{token\_owner}(s) \redir s) \neq \emptyset \not\text{theorem}\)  
\(\text{grd5}: \text{rect\_inst}(\text{token\_owner}(s) \redir s) = \text{suspect\_peers}(\text{token\_owner}(s) \redir s) \not\text{theorem}\)  
\(\text{grd6}: \text{prop} = (((\text{run\_inst}(\text{token\_owner}(s) \redir s) \setminus \text{suspect\_peers} \text{(token\_owner}(s) \redir s))) \cup \text{rctt\_inst}(\text{token\_owner}(s) \redir s) \setminus \text{unav\_peers}) \not\text{theorem}\)  
\(\text{grd8}: \text{susp} = \text{suspect\_peers}(\text{token\_owner}(s) \redir s) \setminus \text{rctt\_inst}(\text{token\_owner}(s) \redir s) \not\text{theorem}\)

THEN
act1: \(\text{inst\_state} = \text{inst\_state} \setminus ((\text{prop} \times \{s\}) \times \{\text{FAIL\_DETECT\_4}\})\)  
act2: \(\text{suspect\_peers} = \text{suspect\_peers} \setminus ((\text{prop} \times \{s\}) \times \{\text{susp}\})\)  
act3: \(\text{rect\_inst}(\text{token\_owner}(s) \redir s) = \emptyset\)  
act4: \(\text{rctt\_inst}(\text{token\_owner}(s) \redir s) = \emptyset\)
END

IS_OK:  

\[\text{REFINES}\]

IS_OK

ANY
prop >

WHERE

\(\text{grd1}: s \in \text{SERVICES} \not\text{theorem}\)  
\(\text{grd2}: \text{prop} \subseteq \text{PEERS} \not\text{theorem}\)  
\(\text{grd3}: \text{inst\_state}(\text{token\_owner}(s) \redir s) = \text{FAIL\_DETECT\_4} \not\text{theorem}\)

\(\text{grd4}: \text{suspect\_peers}(\text{token\_owner}(s) \redir s) = \emptyset \not\text{theorem}\)  
\(\text{grd5}: \text{prop} = \text{run\_inst}(\text{token\_owner}(s) \redir s) \setminus \text{unav\_peers} \not\text{theorem}\)

THEN
act1: \(\text{inst\_state} = \text{inst\_state} \setminus ((\text{prop} \times \{s\}) \times \{\text{RUN\_4}\})\)
FAIL_ACTIV: not extended ordinary

\[
\text{REFINES FAIL_ACTIV}
\]

\[
\text{ANY} \\
\quad s \\
\quad \text{prop}
\]

\[
\text{WHERE} \\
\quad \text{grd1: } s \in \text{SERVICES not theorem} \\
\quad \text{grd2: prop }\subseteq\text{PEERS not theorem} \\
\quad \text{grd3: inst_state(token_owner(s) }\mapsto\text{ s) }=\text{ FAIL_DETECT_4 not theorem} \\
\quad \text{grd4: suspct_peers(token_owner(s) }\mapsto\text{ s) }\neq\emptyset\text{ not theorem} \\
\quad \text{grd5: prop }=\text{ run_inst(token_owner(s) }\mapsto\text{ s) }\setminus (\text{unav_peers }\cup\text{suspct_peers(token_owner(s) }\mapsto\text{ s)})\text{ not theorem}
\]

\[
\text{THEN} \\
\quad \text{act1: inst_state }=\text{ inst_state }\setminus ((\text{prop} \times \{s\}) \times \{\text{FAIL_ACTIV_4}\}) \\
\quad \text{act2: run_inst }=\text{ run_inst }\setminus ((\text{prop} \times \{s\}) \times \{\text{run_inst(token_owner(s) }\mapsto\text{ s)}\} \setminus \text{suspct_peers(token_owner(s) }\mapsto\text{ s)})} \\
\quad \text{act3: failr_peers(s) }=\text{ failr_peers(s) }\cup\text{suspct_peers(token_owner(s) }\mapsto\text{ s)} \\
\quad \text{act4: suspct_peers }=\text{suspct_peers }\setminus ((\text{prop} \times \{s\}) \times \emptyset)
\]

\[
\text{END}
\]

FAIL_CONFIGURE: not extended ordinary

\[
\text{REFINES FAIL_CONFIGURE}
\]

\[
\text{ANY} \\
\quad s \\
\quad \text{prop}
\]

\[
\text{WHERE} \\
\quad \text{grd1: } s \in \text{SERVICES not theorem} \\
\quad \text{grd2: prop }\subseteq\text{PEERS not theorem} \\
\quad \text{grd3: inst_state(token_owner(s) }\mapsto\text{ s) }=\text{ FAIL_ACTIV_4 not theorem} \\
\quad \text{grd4: card(run_inst(token_owner(s) }\mapsto\text{ s)) }<\text{ min_inst(s) not theorem} \\
\quad \text{grd5: prop }=\text{ run_inst(token_owner(s) }\mapsto\text{ s)}\setminus\text{unav_peers not theorem}
\]

\[
\text{THEN} \\
\quad \text{act1: inst_state }=\text{ inst_state }\setminus ((\text{prop} \times \{s\}) \times \{\text{FAIL_CONFIG_4}\})
\]

\[
\text{END}
\]

FAIL_IGNORE: not extended ordinary

\[
\text{REFINES FAIL_IGNORE}
\]

\[
\text{ANY}
\]
M18

\( s \to \text{prop} \to \)

WHERE

- \( \text{grd1: } s \in \text{SERVICES} \) not theorem
- \( \text{grd2: } \text{prop} \subseteq \text{PEERS} \) not theorem
- \( \text{grd3: } \text{inst\_state}(\text{token\_owner}(s) \mapsto s) = \text{FAIL\_ACTIV\_4} \) not theorem

THEN

- \( \text{act1: } \text{inst\_state} = \text{inst\_state} \bowtie ((\text{prop} \times \{s\}) \times \{\text{FAIL\_IGN\_4}\}) \)

END

\text{IGNORE: } \text{not extended ordinary} \to

\text{REFINES}

\text{IGNORE}

\text{ANY}

\( s \to \text{prop} \to \)

WHERE

- \( \text{grd1: } s \in \text{SERVICES} \) not theorem
- \( \text{grd2: } \text{prop} \subseteq \text{PEERS} \) not theorem
- \( \text{grd3: } \text{inst\_state}(\text{token\_owner}(s) \mapsto s) = \text{FAIL\_IGN\_4} \) not theorem

THEN

- \( \text{act1: } \text{inst\_state} = \text{inst\_state} \bowtie ((\text{prop} \times \{s\}) \times \{\text{RUN\_4}\}) \)

END

\text{REDEPLOY\_INSTC: } \text{not extended ordinary} \to

\text{REFINES}

\text{REDEPLOY\_INSTC}

\text{ANY}

\( s \to \text{a service s} \)

\( i \to \text{an instance i} \)

WHERE

- \( \text{grd1: } s \in \text{SERVICES} \) not theorem
- \( \text{grd2: } i \in \text{PEERS} \) not theorem
- \( \text{grd3: } i \notin \text{run\_inst}(\text{token\_owner}(s) \mapsto s) \cup \text{failr\_peers}(s) \cup \text{unav\_peers} \cup \text{dep\_instc}(s) \) not theorem
- \( \text{grd4: } i \notin \text{actv\_instc}(\text{token\_owner}(s) \mapsto s) \) not theorem
- \( \text{grd5: } \text{inst\_state}(\text{token\_owner}(s) \mapsto s) = \text{FAIL\_CONFIG\_4} \) not theorem
- \( \text{grd6: } \text{card}(\text{actv\_instc}(\text{token\_owner}(s) \mapsto s)) < \text{deplo\_inst}(s) \) not theorem
\begin{align*}
\text{REDEPLOY INSTS:} & \quad \text{not extended ordinary} \\
\text{REFINES} & \quad \text{REDEPLOY INSTS} \\
\text{ANY} & \quad s \\
\text{WHERE} & \\
\text{grd1:} & \quad s \in \text{SERVICES} \\
\text{grd2:} & \quad \text{card(actv\_instc(token\_owner(s) \Rightarrow s)) = deplo\_inst(s)} \\
\text{THEN} & \\
\text{act1:} & \quad \text{actv\_instc(token\_owner(s) \Rightarrow s) = actv\_instc(token\_owner (s) \Rightarrow s) \cup \{i\}} \\
\text{END} \\
\text{REDEPLOY:} & \quad \text{not extended ordinary} \\
\text{REFINES} & \quad \text{REDEPLOY} \\
\text{ANY} & \quad s \\
\text{prop} & \\
\text{WHERE} & \\
\text{grd1:} & \quad s \in \text{SERVICES} \\
\text{grd2:} & \quad \text{prop \subseteq \text{PEERS}} \\
\text{grd3:} & \quad \text{inst\_state(token\_owner(s) \Rightarrow s) = FAIL\_CONFIG\_4} \\
\text{THEN} & \\
\text{act1:} & \quad \text{dep\_instc(s) = dep\_instc(s) \cup \text{actv\_instc(token\_owner(s) \Rightarrow s)}} \\
\text{act2:} & \quad \text{actv\_instc(token\_owner(s) \Rightarrow s) = \emptyset} \\
\text{END} \\
\end{align*}
act3: \( \text{dep_instc}(s) = \emptyset \)

END

HEAL: not extended ordinary

REFINES
HEAL
ANY
s
prop
WHERE
  grd1: \( s \in \text{SERVICES} \) not theorem
  grd2: \( \text{prop} \subseteq \text{PEERS} \) not theorem
  grd3: \( \text{inst_state}(\text{token_owner}(s) \mapsto s) = \text{DPL}_4 \) not theorem
  grd4: \( \text{prop} = \text{run_inst}(\text{token_owner}(s) \mapsto s) \setminus \text{unav_peers} \) not theorem
THEN
act1: \( \text{inst_state} = \text{inst_state} \circ ((\text{prop} \times \{s\}) \times \{\text{RUN}_4\}) \)
END

UNFAIL_PEER: extended ordinary

REFINES
UNFAIL_PEER
ANY
s
p
WHERE
  grd1: \( s \in \text{SERVICES} \) not theorem
  grd2: \( p \in \text{PEERS} \) not theorem
  grd3: \( p \in \text{failr_peers}(s) \) not theorem
THEN
act1: \( \text{failr_peers}(s) = \text{failr_peers}(s) \setminus \{p\} \)
END

MAKE_PEER_AVAIL: extended ordinary

REFINES
MAKE_PEER_AVAIL
ANY
p
WHERE
  grd1: \( p \in \text{PEERS} \) not theorem
  grd2: \( p \in \text{unav_peers} \) not theorem
THEN
act1: \( \text{unav_peers} = \text{unav_peers} \setminus \{p\} \)
END

END
M19

MACHINE M19
REFINES M18
SEES C09

VARIABLES
run_inst
suspc_inst
failr_inst
dep_instc
token_owner
unav_peers
suspc_inst
rect_inst
rctt_inst
actv_instc
inst_state

INVARIANTS
inv1: failr_inst ∈ (PEERS×SERVICES) ⇸ ℙ(PEERS) not theorem
inv2: ∀ s · s ∈ SERVICES ⇒ token_owner(s) ↦ s ∈ dom(failr_inst) not theorem

EGLUING_fail_1: ∀ s · s ∈ SERVICES ⇒ failr_inst(token_owner(s) ↦ s) = failr_peers(s) not theorem

EVENTS
INITIALISATION:
act1: run_inst = InitRunPeers
act2: suspct_peers = InitSuspPrs
act3: failr_inst = InitSuspPeers
act4: dep_instc = InitFail
act5: token_owner = init_tok
act6: unav_peers = ø
act7: suspc_inst = InitSuspPeers
act8: rect_inst = InitSuspPeers
act9: rctt_inst = InitSuspPeers
act10: actv_instc = InitSuspPeers
act11: inst_state = InitStateSrv

END

MAKE_PEER_UNAVAIL:
REFINES MAKE_PEER_UNAVAIL

ANY
prs >Peers that will become unavailable
E >Values for token owner per service
WHERE
grd1: prs ⊆ PEERS not theorem

Page 1
grd2: \( \text{prs} \not\in \text{unav_peers} \not\text{ theorem} \Rightarrow \text{the peers in prs are not yet unavailable} \)

\( \forall \text{srv} \cdot \text{srv} \in \text{SERVICES} \Rightarrow \text{dom(dom(inst_state) ▷ \{srv\})} \not\text{ theorem} \Rightarrow \text{for each service srv, there must always be at least 1 peer available} \)

grd4: \( \text{E} \in \text{SERVICES} \rightarrow \text{PEERS} \not\text{ theorem} \Rightarrow \text{Value for token owner per service} \)

grd5: \( \forall \text{srv} \cdot \text{srv} \in \text{SERVICES} \wedge \text{token_owner} (\text{srv}) \not\in \text{prs} \Rightarrow \text{E} (\text{srv}) = \text{token_owner} (\text{srv}) \not\text{ theorem} \Rightarrow \text{If the token owner of a service srv does not belong to prs, the token owner is not changed} \)

grd6: \( \forall \text{srv} \cdot \text{srv} \in \text{SERVICES} \wedge \text{token_owner} (\text{srv}) \in \text{prs} \Rightarrow \text{E(srv) ∈ run_inst(token_owner(srv) ▷ srv) ∖ (unav_peers u prs u failr_inst(token_owner(srv) ▷ srv) u suspct_peers(token_owner(srv) ▷ srv))} \wedge \text{E(srv) ▷ srv ∈ dom(inst_state) ∩ dom(failr_inst) ∩ dom(suspct_peers) ∩ dom(run_inst)} \wedge \text{run_inst(E(srv) ▷ srv) = run_inst(token_owner(srv) ▷ srv)} \wedge \text{inst_state(E(srv) ▷ srv) = inst_state(token_owner(srv) ▷ srv)} \wedge \text{suspct_peers(E(srv) ▷ srv) = suspct_peers(token_owner(srv) ▷ srv)} \wedge \text{failr_inst(E(srv) ▷ srv) = failr_inst(token_owner(srv) ▷ srv)} \not\text{ theorem} \Rightarrow \text{if the owner of the token for a service becomes unavailable, A new token owner is chosen: the new token owner must have same characteristics as the previous one (state, list of suspicious neighbours, etc.), and it must not be an unavailable, suspicious, failed peer or a member of prs}

THEN

act1: \( \text{unav_peers} = \text{unav_peers} \cup \text{prs} \Rightarrow \text{the peers in prs become unavailable} \)

act2: \( \text{token_owner} = \text{token_owner} \sim \text{E} \Rightarrow \text{new values for token owner per service} \)

act3: \( \text{rect_inst} = ((\text{prs} \times \text{SERVICES}) \sim \text{rect_inst}) \sim ((\text{E} \sim \text{token_owner}) \times \{\emptyset\}) \Rightarrow \text{the peers in prs can not try to recontact instances anymore (1)} \)

act4: \( \text{rctt_inst} = ((\text{prs} \times \text{SERVICES}) \sim \text{rctt_inst}) \sim ((\text{E} \sim \text{token_owner}) \times \{\emptyset\}) \Rightarrow \text{the peers in prs can not try to recontact instances anymore (2)} \)

act5: \( \text{actv_instc} = ((\text{prs} \times \text{SERVICES}) \sim \text{actv_instc}) \sim ((\text{E} \sim \text{token_owner}) \times \{\emptyset\}) \Rightarrow \text{the peers in prs can not activate instances anymore} \)

act6: \( \text{suspct_peers} = (\text{prs} \times \text{SERVICES}) \sim \text{suspct_peers} \Rightarrow \text{the peers in prs can not suspect instances anymore} \)
in prs can not suspect instances anymore (1)
\[ \text{act7: } \text{susc}_{\text{inst}} = ((\text{prs} \times \text{SERVICES}) \triangleq \text{susc}_{\text{inst}}) \triangleq ((E\triangle \text{token}_{\text{owner}}) \setminus \{\emptyset\}) \] the peers in prs can not suspect instances anymore (2)
\[ \text{act8: } \text{inst}_{\text{state}} = (\text{prs} \times \text{SERVICES}) \triangleq \text{inst}_{\text{state}} \] the peers in prs can not monitor the state of the services provided anymore
\[ \text{act9: } \text{run}_{\text{inst}} = (\text{prs} \times \text{SERVICES}) \triangleq \text{run}_{\text{inst}} \] \[ \text{act10: } \text{fail}_{\text{inst}} = (\text{prs} \times \text{SERVICES}) \triangleq \text{fail}_{\text{inst}} \]
END

\[ \text{SUSPECT}_{\text{INST}}: \text{ extended ordinary } \triangleq \text{REFINES} \]
\[ \text{SUSPECT}_{\text{INST}} \]
\[ \text{ANY} \]
\[ \text{s } \rightarrow \text{ a service } s \]
\[ \text{suspc}_{\text{inst}} \rightarrow \text{suspicious instances} \]
\[ \text{WHERE} \]
\[ \begin{align*}
\text{grd1: } & \text{s } \in \text{SERVICES} \not \text{ theorem } \rightarrow \\
\text{grd2: } & \text{suspc}_{\text{inst}} \in \text{PEERS} \not \text{ theorem } \rightarrow \\
\text{grd3: } & \text{suspc}_{\text{inst}} = \text{run}_{\text{inst}}(\text{token}_{\text{owner}}(s) \Rightarrow s) \cap \text{unav}_{\text{peers}} \not \text{ theorem } \rightarrow \text{instances in suspc}_{\text{inst}} \text{ are suspicious if the peers running them becomes unavailable} \\
\text{grd4: } & \text{suspc}_{\text{inst}}(\text{token}_{\text{owner}}(s) \Rightarrow s) = \emptyset \not \text{ theorem } \rightarrow \text{the member of suspc}_{\text{inst}} \text{ have not yet been suspected for } s \text{ by the token owner of } s \\
\text{grd5: } & \text{inst}_{\text{state}}(\text{token}_{\text{owner}}(s) \Rightarrow s) = \text{RUN}_4 \not \text{ theorem } \rightarrow \text{the state of } s \text{ is OK} \\
\text{grd6: } & \text{suspc}_{\text{inst}} \neq \emptyset \not \text{ theorem } \rightarrow \\
\text{THEN} \]
\[ \begin{align*}
\text{act1: } & \text{suspc}_{\text{inst}}(\text{token}_{\text{owner}}(s) \Rightarrow s) = \text{suspc}_{\text{inst}} \rightarrow \text{the members of suspc}_{\text{inst}} \text{ become suspected instances for } s \text{ by the token owner of } s \\
\end{align*} \]
END

\[ \text{FAIL: } \text{ extended ordinary } \triangleq \text{REFINES} \]
\[ \text{FAIL} \]
\[ \text{ANY} \]
\[ \text{s } \rightarrow \text{ prop } \rightarrow \]
\[ \text{WHERE} \]
\[ \begin{align*}
\text{grd1: } & \text{s } \in \text{SERVICES} \not \text{ theorem } \rightarrow \\
\text{grd2: } & \text{prop } \in \text{PEERS} \not \text{ theorem } \rightarrow \\
\text{grd3: } & \text{inst}_{\text{state}}(\text{token}_{\text{owner}}(s) \Rightarrow s) = \text{RUN}_4 \not \text{ theorem } \rightarrow \\
\text{grd4: } & \text{suspc}_{\text{inst}}(\text{token}_{\text{owner}}(s) \Rightarrow s) \neq \emptyset \not \text{ theorem } \rightarrow \\
\text{grd5: } & \text{prop } = \text{run}_{\text{inst}}(\text{token}_{\text{owner}}(s) \Rightarrow s) \setminus (\text{suspc}_{\text{inst}}(\text{token}_{\text{owner}}(s) \Rightarrow s) \cup \text{unav}_{\text{peers}}) \not \text{ theorem } \rightarrow \text{prop become suspected for } s \text{ by the token owner of } s \\
\text{THEN} \]
\[ \begin{align*}
\text{act1: } & \text{inst}_{\text{state}} = \text{inst}_{\text{state}} \triangleq ((\text{prop} \times \{s\}) \cup \{\text{FAIL}_4\}) \rightarrow \\
\text{act2: } & \text{suspc}_{\text{inst}} = \text{suspc}_{\text{inst}} \triangleq ((\text{prop} \times \{s\}) \cup \{\text{suspc}_{\text{inst}}(\text{token}_{\text{owner}}(s) \Rightarrow s)\}) \rightarrow \\
\end{align*} \]
Page 3
act3: \[\text{suspc}\_\text{inst}(\text{token}\_\text{owner}(s) \mapsto s) = \emptyset \]

\text{RECONTACT}_\text{INST}_\text{OK}: \quad \text{extended ordinary} \quad \triangleright
\text{REFINES}
\text{RECONTACT}_\text{INST}_\text{OK}
\text{ANY}
\begin{align*}
s & \rightarrow \text{a service } s \\
i & \rightarrow \text{an instance } i
\end{align*}
\text{WHERE}
\begin{align*}
grd1: & \quad s \in \text{SERVICES not theorem} \quad \triangleright \\
grd2: & \quad i \in \text{PEERS not theorem} \quad \triangleright \\
grd3: & \quad \text{inst}_\text{state}(\text{token}\_\text{owner}(s) \mapsto s) = \text{FAIL}_4 \quad \text{not theorem} \quad \triangleright \text{the state of } s \text{ is SUSPICIOUS} \\
grd4: & \quad \text{suspc}_\text{peers}(\text{token}\_\text{owner}(s) \mapsto s) \neq \emptyset \quad \text{not theorem} \quad \triangleright \text{the set of suspicious peers for } s \text{ is not empty} \\
grd5: & \quad i \in \text{suspc}_\text{peers}(\text{token}\_\text{owner}(s) \mapsto s) \setminus \text{unav}_\text{peers} \quad \text{not theorem} \quad \triangleright i \text{ is a suspicious instance of } s \text{ and is available (can be contacted)} \\
grd6: & \quad i \notin \text{rect}_\text{inst}(\text{token}\_\text{owner}(s) \mapsto s) \quad \text{not theorem} \quad \triangleright \text{the token owner of } s \text{ has not yet tried to recontact } i \\
grd7: & \quad \text{rect}_\text{inst}(\text{token}\_\text{owner}(s) \mapsto s) \subset \text{suspc}_\text{peers}(\text{token}\_\text{owner}(s) \mapsto s) \quad \text{not theorem} \quad \triangleright \text{the token owner of } s \text{ has not yet tried to recontact all the suspicious instances of } s
\end{align*}
\text{THEN}
\begin{align*}
\text{act1:} & \quad \text{rect}_\text{inst}(\text{token}\_\text{owner}(s) \mapsto s) = \text{rect}_\text{inst}(\text{token}\_\text{owner}(s) \mapsto s) \cup \{i\} \quad \triangleright \text{the token owner of } s \text{ has tried to recontact } i \\
\text{act2:} & \quad \text{rctt}_\text{inst}(\text{token}\_\text{owner}(s) \mapsto s) = \text{rctt}_\text{inst}(\text{token}\_\text{owner}(s) \mapsto s) \cup \{i\} \quad \triangleright i \text{ is recontacted by the token owner of } s \text{ successfully}
\end{align*}

\text{RECONTACT}_\text{INST}_\text{KO}: \quad \text{extended ordinary} \quad \triangleright
\text{REFINES}
\text{RECONTACT}_\text{INST}_\text{KO}
\text{ANY}
\begin{align*}
s & \rightarrow \text{a service } s \\
i & \rightarrow \text{an instance } i
\end{align*}
\text{WHERE}
\begin{align*}
grd1: & \quad s \in \text{SERVICES not theorem} \quad \triangleright \\
grd2: & \quad i \in \text{PEERS not theorem} \quad \triangleright \\
grd3: & \quad \text{inst}_\text{state}(\text{token}\_\text{owner}(s) \mapsto s) = \text{FAIL}_4 \quad \text{not theorem} \quad \triangleright \text{the state of } s \text{ is SUSPICIOUS} \\
grd4: & \quad \text{suspc}_\text{peers}(\text{token}\_\text{owner}(s) \mapsto s) \neq \emptyset \quad \text{not theorem} \quad \triangleright \text{the set of suspicious peers for } s \text{ is not empty} \\
grd5: & \quad i \in \text{suspc}_\text{peers}(\text{token}\_\text{owner}(s) \mapsto s) \cap \text{unav}_\text{peers} \quad \text{not theorem} \quad \triangleright i \text{ is a suspicious instance of } s \text{ and is unavailable (can not be contacted)} \\
grd6: & \quad i \notin \text{rect}_\text{inst}(\text{token}\_\text{owner}(s) \mapsto s) \quad \text{not theorem} \quad \triangleright \text{the token owner of } s \text{ has not yet tried to recontact } i
\end{align*}
M19

grd7: rect_inst(token_owner(s) ↦ s) ⊂ suspct_peers(token_owner(s) ↦ s) 

THEN

act1: rect_inst(token_owner(s) ↦ s) = rect_inst(token_owner(s) ↦ s) \cup \{i\} 

END

FAIL_DETECT: extended ordinary

REFINES

FAIL_DETECT

ANY

s ↦

prop ↦

susp ↦

WHERE

grd1: s ∈ SERVICES not theorem

grd2: prop ⊆ PEERS not theorem

grd7: susp ⊆ PEERS not theorem

grd3: inst_state(token_owner(s) ↦ s) = FAIL_4 not theorem

grd4: suspct_peers(token_owner(s) ↦ s) ≠ ∅ not theorem

grd5: rect_inst(token_owner(s) ↦ s) = suspct_peers(token_owner(s) ↦ s)

grd6: prop = ((run_inst(token_owner(s) ↦ s) \ suspct_peers(token_owner(s) ↦ s)) \ unav_peers) not theorem

grd8: susp = suspct_peers(token_owner(s) ↦ s) \ rctt_inst(token_owner(s) ↦ s) not theorem

THEN

act1: inst_state ≔ inst_state  ((prop×{s})×{FAIL_DETECT_4})

act2: suspct_peers ≔ suspct_peers  ((prop×{s})×{susp})

act3: rect_inst(token_owner(s) ↦ s) = ∅

act4: rctt_inst(token_owner(s) ↦ s) = ∅

END

IS_OK: extended ordinary

REFINES

IS_OK

ANY

s ↦

prop ↦

WHERE

grd1: s ∈ SERVICES not theorem

grd2: prop ⊆ PEERS not theorem

grd3: inst_state(token_owner(s) ↦ s) = FAIL_DETECT_4 not theorem

grd4: suspct_peers(token_owner(s) ↦ s) = ∅ not theorem

grd5: prop = run_inst(token_owner(s) ↦ s) \ unav_peers not theorem
\[ M_{19} \]

**FAIL_ACTIV:** not extended ordinary

\[ \text{REFINES} \]

FAIL_ACTIV

\[ \text{ANY} \]

\[ s \quad \text{prop} \quad \]

\[ \text{WHERE} \]

\[ \text{grd1: } s \in \text{SERVICES} \quad \text{not theorem} \quad \]
\[ \text{grd2: } \text{prop} \subseteq \text{PEERS} \quad \text{not theorem} \quad \]
\[ \text{grd3: } \text{inst_state(token_owner(s) \rightarrow s)} = \text{FAIL_DETECT}_4 \quad \text{not theorem} \quad \]

\[ \text{THEN} \]

\[ \text{act1: } \text{inst_state} = \text{inst_state} \triangleq ((\text{prop} \times \{s\}) \times \{\text{RUN}_4\}) \quad \]

\[ \text{END} \]

**FAIL_CONFIGURE:** extended ordinary

\[ \text{REFINES} \]

FAIL_CONFIGURE

\[ \text{ANY} \]

\[ s \quad \text{prop} \quad \]

\[ \text{WHERE} \]

\[ \text{grd1: } s \in \text{SERVICES} \quad \text{not theorem} \quad \]
\[ \text{grd2: } \text{prop} \subseteq \text{PEERS} \quad \text{not theorem} \quad \]
\[ \text{grd3: } \text{inst_state(token_owner(s) \rightarrow s)} = \text{FAIL_ACTIV}_4 \quad \text{not theorem} \quad \]
\[ \text{grd4: } \text{card(run_inst(token_owner(s) \rightarrow s))} < \text{min_inst}(s) \quad \text{not theorem} \quad \]
\[ \text{grd5: } \text{prop} = \text{run_inst(token_owner(s) \rightarrow s) \setminus \text{unav_peers}} \quad \text{not theorem} \quad \]

\[ \text{THEN} \]

\[ \text{act1: } \text{inst_state} = \text{inst_state} \triangleq ((\text{prop} \times \{s\}) \times \{\text{FAIL_CONFIG}_4\}) \quad \]

\[ \text{END} \]

**FAIL_IGNORE:** extended ordinary
REFINES
   FAIL_IGNORE
ANY
   s  >
   prop  >
WHERE
   grd1:  s ∈ SERVICES  not theorem  >
   grd2:  prop ⊆ PEERS  not theorem  >
   grd3:  inst_state(token_owner(s) ↦ s) = FAIL_ACTIV_4  not theorem  >
   grd4:  card(run_inst(token_owner(s) ↦ s)) ≥ min_inst(s)  not theorem  >
   grd5:  prop = run_inst(token_owner(s) ↦ s)∖unav_peers  not theorem  >
THEN
   act1:  inst_state = inst_state ≔ ((prop×{s})×{FAIL_IGN_4})  >
END

IGNORE:  extended ordinary  >
REFINES
   IGNORE
ANY
   s  >
   prop  >
WHERE
   grd1:  s ∈ SERVICES  not theorem  >
   grd2:  prop ⊆ PEERS  not theorem  >
   grd3:  inst_state(token_owner(s) ↦ s) = FAIL_IGN_4  not theorem  >
   grd4:  prop = run_inst(token_owner(s) ↦ s)∖unav_peers  not theorem  >
THEN
   act1:  inst_state = inst_state ≔ ((prop×{s})×{RUN_4})  >
END

REDEPLOY_INSTC:  not extended ordinary  >
REFINES
   REDEPLOY_INSTC
ANY
   s  ↠ a service s
   i  ↠ an instance i
WHERE
   grd1:  s ∈ SERVICES  not theorem  >
   grd2:  i ∈ PEERS  not theorem  >
   grd3:  i ∉ run_inst(token_owner(s) ↦ s) ∪ failr_inst(token_owner(s) ↦ s) ∪ unav_peers ∪ dep_instc(s)  not theorem  >i does not run s, is not failed for s, is not unavailable and is not already activated for s
   grd4:  i ∉ actv_instc(token_owner(s) ↦ s)  not theorem  >
M19

theorem

grd5: \text{inst\_state(token\_owner(s) \mapsto s) = FAIL\_CONFIG\_4} \not\text{theorem} \>

grd6: \text{card(actv\_instc(token\_owner(s) \mapsto s)) < deplo\_inst(s)} \not\text{theorem} \>

grd7: \text{card(dep\_instc(s)) + card(run\_inst(token\_owner(s) \mapsto s)) < min\_inst(s)} \not\text{theorem} \>

THEN

act1: \text{actv\_instc(token\_owner(s) \mapsto s) = actv\_instc(token\_owner(s) \mapsto s) \cup \{i\} \not\text{END}}

REDEPLOY\_INSTS: \text{extended ordinary} \>

\text{REFINES}

REDEPLOY\_INSTS

ANY

s \>

WHERE

grd1: \text{s \in SERVICES \not\text{theorem} \>

grd2: \text{card(actv\_instc(token\_owner(s) \mapsto s)) = deplo\_inst(s) \not\text{theorem} \>

grd3: \text{card(dep\_instc(s)) + card(run\_inst(token\_owner(s) \mapsto s)) < min\_inst(s)} \not\text{theorem} \>

THEN

act1: \text{dep\_instc(s) = dep\_instc(s) \cup actv\_instc(token\_owner(s) \mapsto s)} \not\text{END}

act2: \text{actv\_instc(token\_owner(s) \mapsto s) = \emptyset} \not\text{END}

REDEPLOY: \text{extended ordinary} \>

\text{REFINES}

REDEPLOY

ANY

s \>

prop \>

WHERE

grd1: \text{s \in SERVICES \not\text{theorem} \>

grd2: \text{prop \subseteq PEERS \not\text{theorem} \>

grd3: \text{inst\_state(token\_owner(s) \mapsto s) = FAIL\_CONFIG\_4} \not\text{theorem} \>

grd4: \text{actv\_instc(token\_owner(s) \mapsto s) = \emptyset} \not\text{theorem} \>

grd5: \text{dep\_instc(s) \neq \emptyset \not\text{theorem} \>

grd6: \text{card(run\_inst(token\_owner(s) \mapsto s)) + card(dep\_instc(s)) \geq min\_inst(s)} \not\text{theorem} \>

min\_inst(s) \not\text{theorem} \>

grd7: \text{prop = run\_inst(token\_owner(s) \mapsto s)\setminus unav\_peers} \not\text{theorem} \>

THEN

Page 8
act1: \( \text{inst\_state} \leftarrow \text{inst\_state} \land \{(\text{prop} \times \{s\}) \times \{\text{DPL}_4\}\} \)
act2: \( \text{run\_inst} = \text{run\_inst} \land \{(\text{prop} \times \{s\}) \times \{\text{run\_inst}(\text{token\_owner}(s) \rightarrow s) \cup \text{dep\_instc}(s)\}\} \)
act3: \( \text{dep\_instc}(s) = \emptyset \)

END

HEAL: extended ordinary
REFINES HEAL
ANY
s >
prop >
WHERE
  grd1: \( s \in \text{SERVICES} \) not theorem
  grd2: \( \text{prop} \subseteq \text{PEERS} \) not theorem
  grd3: \( \text{inst\_state}(\text{token\_owner}(s) \rightarrow s) = \text{DPL}_4 \) not theorem
  grd4: \( \text{prop} = \text{run\_inst}(\text{token\_owner}(s) \rightarrow s) \setminus \text{unav\_peers} \) not theorem

THEN
  act1: \( \text{inst\_state} \leftarrow \{(\text{prop} \times \{s\}) \times \{\text{RUN}_4\}\} \)
END

UNFAIL\_PEER: not extended ordinary
REFINES UNFAIL\_PEER
ANY
s >
p >
prop >
WHERE
  grd1: \( s \in \text{SERVICES} \) not theorem
  grd2: \( \text{prop} \subseteq \text{PEERS} \) not theorem
  grd3: \( p \in \text{PEERS} \) not theorem
  grd4: \( p \in \text{failr\_inst}(\text{token\_owner}(s) \rightarrow s) \) not theorem
  grd5: \( \text{prop} = \text{run\_inst}(\text{token\_owner}(s) \rightarrow s) \setminus \text{unav\_peers} \) not theorem

THEN
  act1: \( \text{failr\_inst} = \text{failr\_inst} \land \{(\text{prop} \times \{s\}) \times \{\text{failr\_inst}(\text{token\_owner}(s) \rightarrow s) \setminus \{p\}\}\} \)
END

MAKE\_PEER\_AVAIL: extended ordinary
REFINES MAKE\_PEER\_AVAIL
ANY
p >
WHERE
  grd1: \( p \in \text{PEERS} \) not theorem
M19

\[ \text{grd2: } p \in \text{unav_peers} \land \text{not theorem} \rightarrow \]

\[ \text{THEN} \]

\[ \text{act1: } \text{unav_peers} = \text{unav_peers} \setminus \{p\} \rightarrow \]

\[ \text{END} \]

\[ \text{END} \]
M20

MACHINE M20
REFINES M19
SEES C09

VARIABLES
  run_inst
  suspct_peers
  failr_inst
  dep_instcs
  token_owner
  unav_peers
  suspc_inst
  rctt_inst
  rect_inst
  instances that are tried to be recontacted
  rctt_inst
  instances effectively recontacted after a try
  actv_instc
  instances activated by token owners
  inst_state

INVARIANTS
  inv1: dep_instcs ∈ (PEERS×SERVICES) ↔ ℙ(PEERS) not theorem
  inv2: ∀ s · s ∈ SERVICES ⇒ token_owner(s) ↦ s ∈ dom(dep_instcs) not theorem
  gluing_act_1: ∀ s · s ∈ SERVICES ⇒ dep_instcs(token_owner(s) ↦ s) = dep_instcs(s) not theorem

EVENTS

INITIALISATION: not extended ordinary
  THEN
  act1: run_inst ≔ InitRunPeers
  act2: suspct_peers ≔ InitSuspPrs
  act3: failr_inst ≔ InitSuspPeers
  act4: dep_instcs ≔ InitSuspPeers
  act5: token_owner ≔ init_tok
  act6: unav_peers ≔ ∅
  act7: suspc_inst ≔ InitSuspPeers
  act8: rect_inst ≔ InitSuspPeers
  act9: rctt_inst ≔ InitSuspPeers
  act10: actv_instc ≔ InitSuspPeers
  act11: inst_state ≔ InitStateSrv
  END

MAKE_PEER_UNAVAIL: not extended ordinary
  REFINES MAKE_PEER_UNAVAIL
  ANY
    prs "Peers that will become unavailable"
    E "Values for token owner per service"
  WHERE
    grd1: prs ⊆ PEERS not theorem
M20

\[ \text{grd2: } \text{prs} \not\subseteq \text{unav_peers} \not\text{ theorem} \Rightarrow \text{the peers in prs are not yet unavailable} \]

\[ \text{grd3: } \forall \text{srv} \cdot \text{srv} \in \text{SERVICES} \Rightarrow \text{dom(dom(inst_state) \supset \{srv\})} \]

\[ \\text{\textbackslash prs} \not= \text{\textemptyset} \not\text{ theorem} \Rightarrow \text{for each service srv, there must always be at least 1 peer available} \]

\[ \text{grd4: } E \in \text{SERVICES} \rightarrow \text{PEERS} \not\text{ theorem} \Rightarrow \text{Value for token owner per service} \]

\[ \text{grd5: } \forall \text{srv} \cdot \text{srv} \in \text{SERVICES} \land \text{token_owner(srv)} \in \text{prs} \Rightarrow E(srv) = \text{token_owner(srv)} \]

\[ \text{not theorem} \Rightarrow \text{If the token owner of a service srv does not belong to prs, the token owner is not changed} \]

\[ \text{grd6: } \forall \text{srv} \cdot \text{srv} \in \text{SERVICES} \land \text{token_owner(srv)} \in \text{prs} \Rightarrow \]

\[ E(srv) \in \text{run_inst(token_owner(srv) \rightarrow srv) \setminus (unav_peers} \cup \text{failr_inst(token_owner(srv) \rightarrow srv) \cup suspct_peers(token_owner(srv) \rightarrow srv)}) \]

\[ \land \]

\[ \begin{align*}
\text{E(srv)} & \rightarrow \text{srv} \in \text{dom(inst_state) \cap dom(suspct_peers) \cap dom(failr_inst) \cap dom(run_inst)} \supset \text{run_inst(E(srv) \rightarrow srv)} = \text{run_inst(token_owner(srv) \rightarrow srv)} \\
\text{inst_state(E(srv) \rightarrow srv)} & \rightarrow \text{inst_state(token_owner(srv) \rightarrow srv)} \\
\text{suspct_peers(E(srv) \rightarrow srv)} & \rightarrow \text{suspct_peers(token_owner(srv) \rightarrow srv)} \\
\text{failr_inst(E(srv) \rightarrow srv)} & \rightarrow \text{failr_inst(token_owner(srv) \rightarrow srv)} \\
\text{dep_instcs(E(srv) \rightarrow srv)} & \rightarrow \text{dep_instcs(token_owner(srv) \rightarrow srv)}
\end{align*} \]

\[ \text{not theorem} \Rightarrow \text{if the owner of the token for a service becomes unavailable,} \]

A new token owner is chosen: the new token owner must have same characteristics as the previous one (state, list of suspicious neighbours, etc.), and it must not be an unavailable, suspicious, failed peer or a member of prs

\[ \text{THEN} \]

\[ \text{act1: } \text{unav_peers} = \text{unav_peers} \cup \text{prs} \Rightarrow \text{the peers in prs become unavailable} \]

\[ \text{act2: } \text{token_owner} = \text{token_owner} \cup E \Rightarrow \text{new values for token owner per service} \]

\[ \text{act3: } \text{rect_inst} = ((\text{prs} \times \text{SERVICES}) \leftarrow \text{rect_inst}) \leftarrow ((\text{E} \setminus \text{token_owner}) \times \text{\{\emptyset\}}) \Rightarrow \text{the peers in prs can not try to recontact instances anymore (1)} \]

\[ \text{act4: } \text{rctt_inst} = ((\text{prs} \times \text{SERVICES}) \leftarrow \text{rctt_inst}) \leftarrow ((\text{E} \setminus \text{token_owner}) \times \text{\{\emptyset\}}) \Rightarrow \text{the peers in prs can not try to recontact instances anymore (2)} \]

\[ \text{act5: } \text{actv_instc} = ((\text{prs} \times \text{SERVICES}) \leftarrow \text{actv_instc}) \leftarrow \]

Page 2
(\{\text{\textbf{E}}\text{\textunderscore}token\textunderscore}owner\}) \times \{\varnothing\} \triangleright \text{the peers in prs can not activate instances anymore}

\text{act5}: \quad \text{actv\textunderscore}instc = ((prs \times \text{SERVICES}) \triangleright \text{actv\textunderscore}instc) \triangleright \text{the peers in prs can not activate instances anymore (1)}

\text{act6}: \quad \text{suspc\textunderscore}peers = (prs \times \text{SERVICES}) \triangleright \text{suspc\textunderscore}peers \triangleright \text{the peers in prs can not suspect instances anymore (1)}

\text{act7}: \quad \text{suspc\textunderscore}inst = ((prs \times \text{SERVICES}) \triangleright \text{suspc\textunderscore}inst) \triangleright ((\text{E} \setminus \text{token\textunderscore}owner) \sim) \times \{\varnothing\} \triangleright \text{the peers in prs can not activate instances anymore (2)}

\text{act8}: \quad \text{inst\textunderscore}state = (prs \times \text{SERVICES}) \triangleright \text{inst\textunderscore}state \triangleright \text{the peers in prs can not monitor the state of the services provided anymore}

\text{act9}: \quad \text{run\textunderscore}inst = (prs \times \text{SERVICES}) \triangleright \text{run\textunderscore}inst \triangleright \text{act10}: \quad \text{failr\textunderscore}inst = (prs \times \text{SERVICES}) \triangleright \text{failr\textunderscore}inst \triangleright \text{act11}: \quad \text{dep\textunderscore}instcs = (prs \times \text{SERVICES}) \triangleright \text{dep\textunderscore}instcs \triangleright \text{END}

\textbf{SUSPECT\textunderscore}INST: \quad \text{extended ordinary} \triangleright \text{REFINES} \triangleright \text{SUSPECT\textunderscore}INST

\textbf{ANY}\triangleright \text{a service s}
\text{suspc}\triangleright \text{suspicious instances}

\textbf{WHERE}\triangleright \text{instances in susp are suspicious if the peers running them becomes unavailable}

\text{grd1}: \quad s \in \text{SERVICES} \not\text{theorem} \triangleright \text{the member of susp have not yet been suspected for s by the token owner of s}
\text{grd2}: \quad \text{suspc\textunderscore}inst(token\textunderscore}owner(s) \mapsto s) = \varnothing \not\text{theorem} \triangleright \text{the state of s is OK}
\text{grd3}: \quad \text{inst\textunderscore}state(token\textunderscore}owner(s) \mapsto s) = \text{RUN\textunderscore}4 \not\text{theorem} \triangleright \text{suspc\textunderscore}inst(token\textunderscore}owner(s) \mapsto s) = \text{suspc}\not\text{theorem} \triangleright \text{THEN}\triangleright \text{the members of susp become suspected instances for s by the token owner of s}
\text{END}

\textbf{FAIL:} \quad \text{extended ordinary} \triangleright \text{REFINES} \triangleright \text{FAIL}

\textbf{ANY}\triangleright \text{a service s}
\text{prop}\triangleright \text{prop set}

\textbf{WHERE}\triangleright \text{instances in suspc are suspicious if the peers running them becomes unavailable}

\text{grd1}: \quad s \in \text{SERVICES} \not\text{theorem} \triangleright \text{inst\textunderscore}state(token\textunderscore}owner(s) \mapsto s) = \text{RUN\textunderscore}4 \not\text{theorem} \triangleright \text{prop = run\textunderscore}inst(token\textunderscore}owner(s) \mapsto s) \setminus \text{inst\textunderscore}state(token\textunderscore}owner(s) \mapsto s) \not\text{theorem} \triangleright \text{THEN}\triangleright \text{THEN}
\[
\begin{align*}
\text{act1: } \quad \text{inst\_state} \leftarrow \text{inst\_state} \circ ((\text{prop}\times\{s\})\times\{\text{FAIL}_4\}) \circ (\text{token\_owner}(s) \mapsto s)) \\
\text{act2: } \quad \text{suspct\_peers} \leftarrow \text{suspct\_peers} \circ ((\text{prop}\times\{s\})\times\{\text{suspc\_inst}(\text{token\_owner}(s) \mapsto s)\}) \\
\text{act3: } \quad \text{suspc\_inst}(\text{token\_owner}(s) \mapsto s) = \emptyset
\end{align*}
\]
grd6: $i \notin \text{rect\_inst}(\text{token\_owner}(s) \mapsto s)$ not theorem $\triangleright$ the token owner of $s$ has not yet tried to recontact $i$

grd7: $\text{rect\_inst}(\text{token\_owner}(s) \mapsto s) \subseteq \text{suspct\_peers}(\text{token\_owner}(s) \mapsto s)$ not theorem $\triangleright$ the token owner of $s$ has not yet tried to recontact all the suspicious instances of $s$

THEN

act1: $\text{rect\_inst}(\text{token\_owner}(s) \mapsto s) = \text{rect\_inst}(\text{token\_owner}(s) \mapsto s) \cup \{i\} \triangleright$ the token owner of $s$ has tried to recontact $i$

END

FAIL\_DETECT: extended ordinary $\triangleright$

REFINES

FAIL\_DETECT

ANY

$s \triangleright$

prop $\triangleright$

susp $\triangleright$

WHERE

grd1: $s \in \text{SERVICES}$ not theorem $\triangleright$

grd2: $\text{prop} \subseteq \text{PEERS}$ not theorem $\triangleright$

grd7: $\text{susp} \subseteq \text{PEERS}$ not theorem $\triangleright$

grd3: $\text{inst\_state}(\text{token\_owner}(s) \mapsto s) = \text{FAIL} 4$ not theorem $\triangleright$

grd4: $\text{suspct\_peers}(\text{token\_owner}(s) \mapsto s) \neq \emptyset$ not theorem $\triangleright$

grd5: $\text{rect\_inst}(\text{token\_owner}(s) \mapsto s) = \text{suspct\_peers}(\text{token\_owner}(s) \mapsto s)$ not theorem $\triangleright$

grd6: $\text{prop} = (\text{run\_inst}(\text{token\_owner}(s) \mapsto s) \setminus \text{suspct\_peers}(\text{token\_owner}(s) \mapsto s)) \cup \text{rctt\_inst}(\text{token\_owner}(s) \mapsto s) \setminus \text{unav\_peers}$ not theorem $\triangleright$

grd8: $\text{susp} = \text{suspct\_peers}(\text{token\_owner}(s) \mapsto s) \setminus \text{rctt\_inst}(\text{token\_owner}(s) \mapsto s)$ not theorem $\triangleright$

THEN

act1: $\text{inst\_state} = \text{inst\_state} \setminus ((\text{prop} \times \{s\}) \times \{\text{FAIL\_DETECT\_4}\})$

act2: $\text{suspct\_peers} = \text{suspct\_peers} \setminus ((\text{prop} \times \{s\}) \times \{\text{susp}\})$

act3: $\text{rect\_inst}(\text{token\_owner}(s) \mapsto s) = \emptyset$

act4: $\text{rctt\_inst}(\text{token\_owner}(s) \mapsto s) = \emptyset$

END

IS\_OK: extended ordinary $\triangleright$

REFINES

IS\_OK

ANY

$s \triangleright$

prop $\triangleright$

WHERE

grd1: $s \in \text{SERVICES}$ not theorem $\triangleright$

grd2: $\text{prop} \subseteq \text{PEERS}$ not theorem $\triangleright$

grd3: $\text{inst\_state}(\text{token\_owner}(s) \mapsto s) = \text{FAIL\_DETECT\_4}$ not theorem $\triangleright$
M20

theorem >
  grd4: suspct_peers(token_owner(s) ↦ s) = ∅ not theorem >
  grd5: prop = run_inst(token_owner(s) ↦ s) ∖ unav_peers not theorem >

THEN
  act1: inst_state = inst_state « ((prop×{s})×{RUN_4}) >
END

FAIL_ACTIV: extended ordinary >
REFINES
FAIL_ACTIV
ANY
s >
prop >
WHERE
  grd1: s ∈ SERVICES not theorem >
  grd2: prop ⊆ PEERS not theorem >
  grd3: inst_state(token_owner(s) ↦ s) = FAIL_DETECT_4 not theorem >

THEN
  act1: inst_state = inst_state « ((prop×{s})×{FAIL_ACTIV_4}) >
  act2: run_inst = run_inst « ((prop×{s})×{run_inst(token_owner(s) ↦ s) \ suspct_peers(token_owner(s) ↦ s)}) >
  act3: failr_inst = failr_inst « ((prop×{s})×{failr_inst(token_owner(s) ↦ s) \ suspct_peers(token_owner(s) ↦ s)}) >
  act4: suspct_peers = suspct_peers « ((prop×{s})×{∅}) >
END

FAIL_CONFIGURE: extended ordinary >
REFINES
FAIL_CONFIGURE
ANY
s >
prop >
WHERE
  grd1: s ∈ SERVICES not theorem >
  grd2: prop ⊆ PEERS not theorem >
  grd3: inst_state(token_owner(s) ↦ s) = FAIL_ACTIV_4 not theorem >

THEN
  act1: inst_state = inst_state « ((prop×{s})×{FAIL_CONFIG_4}) >
  act2: run_inst = run_inst « ((prop×{s})×{run_inst(token_owner(s) ↦ s) \ suspct_peers(token_owner(s) ↦ s)}) >
  act3: failr_inst = failr_inst « ((prop×{s})×{failr_inst(token_owner(s) ↦ s) \ suspct_peers(token_owner(s) ↦ s)}) >
  act4: suspct_peers = suspct_peers « ((prop×{s})×{∅}) >
END
FAIL_IGNORE: extended ordinary

REFINES
FAIL_IGNORE

ANY
s
  prop
WHERE
  grd1: s ∈ SERVICES not theorem
  grd2: prop ⊆ PEERS not theorem
  grd3: inst_state(token_owner(s) ↦ s) = FAIL_ACTIV_4 not theorem
  theorem
  grd4: card(run_inst(token_owner(s) ↦ s)) ≥ min_inst(s) not theorem
  theorem
  grd5: prop = run_inst(token_owner(s) ↦ s) \ unav_peers not theorem
  theorem
THEN
  act1: inst_state = inst_state ≔ ((prop×{s})×{FAIL_IGN_4})
END

IGNORE: extended ordinary

REFINES
IGNORE

ANY
s
  prop
WHERE
  grd1: s ∈ SERVICES not theorem
  grd2: prop ⊆ PEERS not theorem
  grd3: inst_state(token_owner(s) ↦ s) = FAIL_IGN_4 not theorem
  theorem
  grd4: prop = run_inst(token_owner(s) ↦ s) \ unav_peers not theorem
  theorem
THEN
  act1: inst_state = inst_state ≔ ((prop×{s})×{RUN_4})
END

REDEPLOY_INSTC: not extended ordinary

REFINES
REDEPLOY_INSTC

ANY
s → a service s
i → an instance i
WHERE
  grd1: s ∈ SERVICES not theorem
  grd2: i ∈ PEERS not theorem
  grd3: i ∉ run_inst(token_owner(s) ↦ s) ∪ failr_inst
M20

(token_owner(s) ↦ s) u unav_peers u dep_instcs(token_owner(s) ↦ s) not theorem
> i does not run s, is not failed for s, is not unavailable and is not already
activated for s

grd4: i ∉ actv_instc(token_owner(s) ↦ s) not theorem
> theorem

grd5: inst_state(token_owner(s) ↦ s) = FAIL_CONFIG_4 not
> theorem

grd6: card(actv_instc(token_owner(s) ↦ s)) < deplo_inst(s) not
> theorem

grd7: card(dep_instcs(token_owner(s) ↦ s)) + card(run_inst
(token_owner(s) ↦ s)) < min_inst(s) not theorem
> THEN

act1: actv_instc(token_owner(s) ↦ s) = actv_instc(token_owner
(s) ↦ s) u {i} >
END

REDEPLOY_INSTS: not extended ordinary
> REFINES

REDEPLOY_INSTS

ANY
s >
prop >
WHERE

grd1: s ∈ SERVICES not theorem
> theorem

grd2: prop ⊆ PEERS not theorem
> theorem

grd3: card(actv_instc(token_owner(s) ↦ s)) = deplo_inst(s) not
> theorem

grd4: card(dep_instcs(token_owner(s) ↦ s)) + card(run_inst
(token_owner(s) ↦ s)) < min_inst(s) not theorem
> theorem

grd5: inst_state(token_owner(s) ↦ s) = FAIL_CONFIG_4 not
> theorem

grd6: prop = run_inst(token_owner(s) ↦ s)\unav_peers not
> theorem
> THEN

act1: dep_instcs = dep_instcs ∪ (prop×{s})× {dep_instcs
(token_owner(s) ↦ s) u actv_instc(token_owner(s) ↦ s)}
> act2: actv_instc(token_owner(s) ↦ s) = ∅ >
END

REDEPLOY: not extended ordinary
> REFINES

REDEPLOY

ANY
s >
prop >
WHERE

grd1: s ∈ SERVICES not theorem
> theorem

grd2: prop ⊆ PEERS not theorem
> theorem

grd3: inst_state(token_owner(s) ↦ s) = FAIL_CONFIG_4 not
>
THEOREM: 

\text{grd4: } \text{actv\_instc(token\_owner(s) ↦ s)} = \emptyset \text{ not theorem} \\
\text{grd5: } \text{dep\_instcs(token\_owner(s) ↦ s)} \neq \emptyset \text{ not theorem} \\
\text{grd6: } \text{card(run\_inst(token\_owner(s) ↦ s))} + \text{card(dep\_instcs (token\_owner(s) ↦ s))} \geq \text{min\_inst(s)} \text{ not theorem} \\
\text{grd7: } \text{prop = run\_inst(token\_owner(s) ↦ s)} \setminus \text{unav\_peers} \text{ not theorem} \\

\text{THEN} \\
\text{act1: } \text{inst\_state} \leftarrow \text{inst\_state} \circ ((\text{prop} \times \{s\}) \times \{\text{DPL\_4}\}) \\
\text{act2: } \text{run\_inst} \leftarrow \text{run\_inst} \circ ((\text{prop} \times \{s\}) \times \{\text{run\_inst(token\_owner(s) ↦ s)} \} \cup \text{dep\_instcs(token\_owner(s) ↦ s)}) \\
\text{act3: } \text{dep\_instcs} \leftarrow \text{dep\_instcs} \circ ((\text{prop} \times \{s\}) \times \{\emptyset\}) \\
\text{END} \\

\text{HEAL:} \quad \text{extended ordinary} \\
\text{REFINES} \\
\text{HEAL} \\
\text{ANY} \\
\text{s} \quad \text{prop} \\
\text{WHERE} \\
\text{grd1: } s \in \text{SERVICES} \text{ not theorem} \\
\text{grd2: } \text{prop} \in \text{PEERS} \text{ not theorem} \\
\text{grd3: } \text{inst\_state(token\_owner(s) ↦ s)} = \text{DPL\_4} \text{ not theorem} \\
\text{grd4: } \text{prop = run\_inst(token\_owner(s) ↦ s)} \setminus \text{unav\_peers} \text{ not theorem} \\
\text{THEN} \\
\text{act1: } \text{inst\_state} \leftarrow \text{inst\_state} \circ ((\text{prop} \times \{s\}) \times \{\text{RUN\_4}\}) \\
\text{END} \\

\text{UNFAIL\_PEER:} \quad \text{extended ordinary} \\
\text{REFINES} \\
\text{UNFAIL\_PEER} \\
\text{ANY} \\
\text{s} \quad \text{p} \quad \text{prop} \\
\text{WHERE} \\
\text{grd1: } s \in \text{SERVICES} \text{ not theorem} \\
\text{grd2: } \text{prop} \in \text{PEERS} \text{ not theorem} \\
\text{grd3: } p \in \text{PEERS} \text{ not theorem} \\
\text{grd4: } p \in \text{failr\_inst(token\_owner(s) ↦ s)} \text{ not theorem} \\
\text{grd5: } \text{prop = run\_inst(token\_owner(s) ↦ s)} \setminus \text{unav\_peers} \text{ not theorem} \\
\text{THEN} \\
\text{act1: } \text{failr\_inst} \leftarrow \text{failr\_inst} \circ ((\text{prop} \times \{s\}) \times \{\text{failr\_inst (token\_owner(s) ↦ s)} \} \setminus \{p\}) \\
\text{END} \

Page 9
MAKE_PEER_AVAIL: extended ordinary

REFINES

MAKE_PEER_AVAIL

ANY

p

WHERE

grd1: p ∈ PEERS not theorem

grd2: p ∈ unav_peers not theorem

THEN

act1: unav_peers = unav_peers \ {p}

END

END
M21

MACHINE M21 ›
REFINES M20
SEES C09

VARIABLES
run_inst ›
suspct_peers ›
failr_inst ›
dep_instcs ›
token_owner ›
unav_peers ›
suspc_inst ›
rect_inst › instances that are tried to be recontacted
rctt_inst › instances effectively recontacted after a try
actv_instc › instances activated by token owner
inst_state ›

INVARINTANTS
inv1: dom(run_inst) ⊆ dom(inst_state) not theorem ›

EVENTS

INITIALISATION: extended ordinary ›
THEN
act1: run_inst ≔ InitRunPeers ›
act2: suspct_peers ≔ InitSuspPrs ›
act3: failr_inst ≔ InitSuspPeers ›
act4: dep_instcs ≔ InitSuspPeers ›
act5: token_owner ≔ init_tok ›
act6: unav_peers ≔ ∅ ›
act7: suspc_inst ≔ InitSuspPeers ›
act8: rect_inst ≔ InitSuspPeers ›
act9: rctt_inst ≔ InitSuspPeers ›
act10: actv_instc ≔ InitSuspPeers ›
act11: inst_state ≔ InitStateSrv ›

END

MAKE_PEER_UNAVAIL: not extended ordinary ›
REFINES MAKE_PEER_UNAVAIL
ANY
prs › Peers that will become unavailable
E › Values for token owner per service
WHERE
grd1: prs ∈ PEERS not theorem ›
grd2: prs \∈ unav_peers not theorem › the peers in prs are not yet unavailable
grd3: ∀ srv · srv ∈ SERVICES ⇒ dom(dom(inst_state) \{srv\}) \prṣ \≠ \ø not theorem › for each service srv, there must always be at least 1
peer available

\[ \text{grd4: } E \in \text{SERVICES} \rightarrow \text{PEERS} \]
not theorem

\( \forall \text{srv} \cdot \text{srv} \in \text{SERVICES} \Rightarrow \text{dom} \left( \text{dom} \left( \text{inst\_state} \right) \cap \{ \text{srv} \} \right) \neq \emptyset \)

for each service srv, there must always be at least 1 peer available

\[ \text{grd5: } E \in \text{SERVICES} \rightarrow \text{PEERS} \]
not theorem

\( \forall \text{srv} \cdot \text{srv} \in \text{SERVICES} \wedge \text{token\_owner}(\text{srv}) \notin \text{prs} \Rightarrow E(\text{srv}) = \text{token\_owner}(\text{srv}) \)

If the token owner of a service srv does not belong to prs, the token owner is not changed

\[ \text{grd6: } \forall \text{srv} \cdot \text{srv} \in \text{SERVICES} \wedge \text{token\_owner}(\text{srv}) \in \text{prs} \Rightarrow \]

\[ E(\text{srv}) \in \text{run\_inst}(\text{token\_owner}(\text{srv}) \mapsto \text{srv}) \setminus (\text{unav\_peers} \cup \text{failr\_inst}(\text{token\_owner}(\text{srv}) \mapsto \text{srv}) \cup \text{suspc\_peers}(\text{token\_owner}(\text{srv}) \mapsto \text{srv})) \]

\( \land \)

\[ E(\text{srv}) \mapsto \text{srv} \in \text{dom}(\text{run\_inst}) \cap \text{dom}(\text{suspc\_peers}) \cap \text{dom}(\text{failr\_inst}) \cap \text{dom}(\text{dep\_instcs}) \]

\( \land \)

\[ \text{run\_inst}(E(\text{srv}) \mapsto \text{srv}) = \text{run\_inst}(\text{token\_owner}(\text{srv}) \mapsto \text{srv}) \]

\( \land \)

\[ \text{inst\_state}(E(\text{srv}) \mapsto \text{srv}) = \text{inst\_state}(\text{token\_owner}(\text{srv}) \mapsto \text{srv}) \]

\( \land \)

\[ \text{suspc\_peers}(E(\text{srv}) \mapsto \text{srv}) = \text{suspc\_peers}(\text{token\_owner}(\text{srv}) \mapsto \text{srv}) \]

\( \land \)

\[ \text{failr\_inst}(E(\text{srv}) \mapsto \text{srv}) = \text{failr\_inst}(\text{token\_owner}(\text{srv}) \mapsto \text{srv}) \]

\( \land \)

\[ \text{dep\_instcs}(E(\text{srv}) \mapsto \text{srv}) = \text{dep\_instcs}(\text{token\_owner}(\text{srv}) \mapsto \text{srv}) \]

not theorem

\( \text{if the owner of the token for a service becomes unavailable,} \)

A new token owner is chosen: the new token owner must have same characteristics

as the previous one (state, list of suspicious neighbours, etc.), and it must

not be an unavailable, suspicious, failed peer or a member of prs

THEN

\[ \text{act1: } \text{unav\_peers} = \text{unav\_peers} \cup \text{prs} \rightarrow \text{the peers in prs become unavailable} \]

\[ \text{act2: } \text{token\_owner} = \text{token\_owner} \cup E \rightarrow \text{new values for token owner per service} \]

\[ \text{act3: } \text{rect\_inst} = ((\text{prs} \times \text{SERVICES}) \setminus \text{rect\_inst}) \setminus ((\text{E} \setminus \text{token\_owner}) \times \{ \emptyset \}) \rightarrow \text{the peers in prs can not try to recontact instances anymore (1)} \]

\[ \text{act4: } \text{rctt\_inst} = ((\text{prs} \times \text{SERVICES}) \setminus \text{rctt\_inst}) \setminus ((\text{E} \setminus \text{token\_owner}) \times \{ \emptyset \}) \rightarrow \text{the peers in prs can not try to recontact instances anymore (2)} \]

\[ \text{act5: } \text{actv\_instc} = ((\text{prs} \times \text{SERVICES}) \setminus \text{actv\_instc}) \setminus ((\text{E} \setminus \text{token\_owner}) \times \{ \emptyset \}) \rightarrow \text{the peers in prs can not activate instances anymore (1)} \]

\[ \text{act6: } \text{suspc\_inst} = ((\text{prs} \times \text{SERVICES}) \setminus \text{suspc\_inst}) \setminus ((\text{E} \setminus \text{token\_owner}) \times \{ \emptyset \}) \rightarrow \text{the peers in prs can not suspect instances anymore (1)} \]

\[ \text{act7: } \text{suspc\_inst} = ((\text{prs} \times \text{SERVICES}) \setminus \text{suspc\_inst}) \setminus ((\text{E} \setminus \text{token\_owner}) \times \{ \emptyset \}) \rightarrow \text{the peers in prs can not activate instances anymore (2)} \]
(((E∖token_owner)×{∅})) →\(the\ \ peers\ \ in\ \ prs\ \ can\ \ not\ \ suspect\ \ instances\ \ anymore\ \ (2)\)
act8: \(\text{inst\ state} = (prs×SERVICES) ≔ \text{inst\ state} →\(the\ \ peers\ \ in\ \ prs\ \ can\ \ not\ \ monitor\ \ the\ \ state\ \ of\ \ the\ \ services\ \ provided\ \ anymore\)
act9: \(\text{run\ inst} = (prs×SERVICES) ≔ \text{run\ inst} →\)
act10: \(\text{failr\ inst} = (prs×SERVICES) ≔ \text{failr\ inst} →\)
act11: \(\text{dep\ instcs} = (prs×SERVICES) ≔ \text{dep\ instcs} →\)

END

SUSPECT_INST: extended ordinary
REFINES
SUSPECT_INST
ANY
s → a service s
susp → suspicious instances
WHERE
grd1: \(s ∈ \text{SERVICES} \not\ \text{not\ theorem}\ →\)
grd2: \(\text{susp} ∈ \text{PEERS} \not\ \text{not\ theorem}\ →\)
grd3: \(\text{susp} = \text{run\ inst}(\text{token\ owner}(s) ↦ s) \cap \text{unav\ peers} \not\ \text{not\ theorem}\ →\)\(instances\ \ in\ \ susp\ \ are\ \ suspicious\ \ if\ \ the\ \ peers\ \ running\ \ them\ \ becomes\ \ unavailable\)
grd4: \(\text{suspc\ inst}(\text{token\ owner}(s) ↦ s) = \emptyset \not\ \text{not\ theorem}\ →\)\(the\ \ member\ \ of\ \ susp\ \ have\ \ not\ \ yet\ \ been\ \ suspected\ \ for\ \ s\ \ by\ \ the\ \ token\ \ owner\ \ of\ \ s\)
grd5: \(\text{inst\ state}(\text{token\ owner}(s) ↦ s) = \text{RUN}_4 \not\ \text{not\ theorem}\ →\)\(the\ \ state\ \ of\ \ s\ \ is\ \ OK\)
grd6: \(\text{susp} ≠ \emptyset \not\ \text{not\ theorem}\ →\)
THEN
act1: \(\text{suspc\ inst}(\text{token\ owner}(s) ↦ s) = \text{susp} \rightarrow\)\(the\ \ members\ \ of\ \ susp\ \ become\ \ suspected\ \ instances\ \ for\ \ s\ \ by\ \ the\ \ token\ \ owner\ \ of\ \ s\)

END

FAIL: extended ordinary
REFINES
FAIL
ANY
s → prop →
WHERE
grd1: \(s ∈ \text{SERVICES} \not\ \text{not\ theorem}\ →\)
grd2: \(\text{prop} ∈ \text{PEERS} \not\ \text{not\ theorem}\ →\)
grd3: \(\text{inst\ state}(\text{token\ owner}(s) ↦ s) = \text{RUN}_4 \not\ \text{not\ theorem}\ →\)
grd4: \(\text{suspc\ inst}(\text{token\ owner}(s) ↦ s) ≠ \emptyset \not\ \text{not\ theorem}\ →\)
grd5: \(\text{prop} = \text{run\ inst}(\text{token\ owner}(s) ↦ s)\ (\text{suspc\ inst}(\text{token\ owner}(s) ↦ s) \cup \text{unav\ peers}) \not\ \text{not\ theorem}\ →\)
THEN
act1: \(\text{inst\ state} = \text{inst\ state} ≔ (\text{prop}×{s})×{\text{FAIL}_4} \rightarrow\)
act2: \(\text{suspct\ peers} = \text{suspct\ peers} ≔ (\text{prop}×{s})×(\text{suspc\ inst}) \ (\text{token\ owner}(s) ↦ s)) \rightarrow\)
act3: \(\text{suspc\ inst}(\text{token\ owner}(s) ↦ s) = \emptyset \rightarrow\)
RECONTACT_INST_OK: extended ordinary
REFINES RECONTACT_INST_OK
ANY
s → a service s
i → an instance i
WHERE
grd1: s ∈ SERVICES not theorem → the state of s is SUSPICIOUS
grd2: i ∈ PEERS not theorem →
grd3: inst_state(token_owner(s) ↦ s) = FAIL_4 not theorem → 
the set of suspicious peers for s is not empty
grd4: suspct_peers(token_owner(s) ↦ s) ≠ ø not theorem → the set of suspicious peers for s is not empty
grd5: i ∈ suspct_peers(token_owner(s) ↦ s)\unav_peers not theorem → i is a suspicious instance of s and is available (can be contacted)
grd6: i ∉ rect_inst(token_owner(s) ↦ s) not theorem → the token owner of s has not yet tried to recontact i
grd7: rect_inst(token_owner(s) ↦ s) ⊂ suspct_peers(token_owner(s) ↦ s) not theorem → the token owner of s has not yet tried to recontact all the suspicious instances of s
THEN
act1: rect_inst(token_owner(s) ↦ s) = rect_inst(token_owner(s) ↦ s) ∪ {i} → the token owner of s has tried to recontact i
act2: rctt_inst(token_owner(s) ↦ s) = rctt_inst(token_owner(s) ↦ s) ∪ {i} → i is recontacted by the token owner of s successfully
END

RECONTACT_INST_KO: extended ordinary
REFINES RECONTACT_INST_KO
ANY
s → a service s
i → an instance i
WHERE
grd1: s ∈ SERVICES not theorem → the state of s is SUSPICIOUS
grd2: i ∈ PEERS not theorem →
grd3: inst_state(token_owner(s) ↦ s) = FAIL_4 not theorem → the state of s is SUSPICIOUS
grd4: suspct_peers(token_owner(s) ↦ s) ≠ ø not theorem → the set of suspicious peers for s is not empty
grd5: i ∈ suspct_peers(token_owner(s) ↦ s)\unav_peers not theorem → i is a suspicious instance of s and is unavailable (can not be contacted)
grd6: i ∉ rect_inst(token_owner(s) ↦ s) not theorem → the token owner of s has not yet tried to recontact i
grd7: rect_inst(token_owner(s) ↦ s) ⊂ suspct_peers(token_owner(s) ↦ s) not theorem → the token owner of s has not yet tried to recontact all the suspicious instances of s
\( s \mapsto s \) not theorem \( \Rightarrow \) the token owner of \( s \) has not yet tried to recontact all the suspicious instances of \( s \)

\[
\text{THEN}
\]
\[
\text{act1: rect\_inst(token\_owner(s) \mapsto s) = rect\_inst(token\_owner(s) \mapsto s) \cup \{i\} \Rightarrow the token owner of s has tried to recontact i}
\]
\[
\text{END}
\]

**FAIL\_DETECT:**

\[
\text{extended ordinary} \Rightarrow 
\]
\[
\text{REFINES}
\]
\[
\text{FAIL\_DETECT}
\]

**ANY**

\[
\begin{align*}
\text{s} & \quad \text{>} \\
\text{prop} & \quad \text{>} \\
\text{susp} & \quad \text{>}
\end{align*}
\]

**WHERE**

\[
\begin{align*}
\text{grd1: } s & \in \text{SERVICES not theorem} \\
\text{grd2: } \text{prop} & \in \text{PEERS not theorem} \\
\text{grd7: } \text{susp} & \in \text{PEERS not theorem} \\
\text{grd3: } \text{inst\_state(token\_owner(s) \mapsto s) = FAIL\_4 not theorem} \\
\text{grd4: } \text{suspct\_peers(token\_owner(s) \mapsto s) \neq \emptyset not theorem} \\
\text{grd5: } \text{rect\_inst(token\_owner(s) \mapsto s) = suspct\_peers(token\_owner(s) \mapsto s)}
\end{align*}
\]

\( s \mapsto s \) not theorem

\[
\text{THEN}
\]
\[
\text{act1: inst\_state = inst\_state \cup ((prop \times \{s\}) \times \{\text{FAIL\_DETECT\_4}\})} \\
\text{act2: suspct\_peers = suspct\_peers \cup ((prop \times \{s\}) \times \{\text{susp}\})} \\
\text{act3: rect\_inst(token\_owner(s) \mapsto s) = \emptyset} \\
\text{act4: rctt\_inst(token\_owner(s) \mapsto s) = \emptyset}
\]
\[
\text{END}
\]

**IS\_OK:**

\[
\text{extended ordinary} \Rightarrow 
\]
\[
\text{REFINES}
\]
\[
\text{IS\_OK}
\]

**ANY**

\[
\begin{align*}
\text{s} & \quad \text{>} \\
\text{prop} & \quad \text{>}
\end{align*}
\]

**WHERE**

\[
\begin{align*}
\text{grd1: } s & \in \text{SERVICES not theorem} \\
\text{grd2: } \text{prop} & \in \text{PEERS not theorem} \\
\text{grd3: } \text{inst\_state(token\_owner(s) \mapsto s) = FAIL\_DETECT\_4 not theorem} \\
\text{grd4: } \text{suspct\_peers(token\_owner(s) \mapsto s) = \emptyset not theorem} \\
\text{grd5: } \text{prop = run\_inst(token\_owner(s) \mapsto s) \setminus \text{unav\_peers not theorem}}
\end{align*}
\]
M21

THEN
act1: \text{inst\_state} := \text{inst\_state} \land ((\text{prop} \times \{s\}) \times \{\text{RUN}_4\}) \land

END

FAIL\_ACTIV: extended ordinary

\text{REFINES}

FAIL\_ACTIV

\text{ANY}
\begin{align*}
& s \ \triangleright \\
& \text{prop} \ \triangleright \\
\text{WHERE}
& \text{grd1}: \ s \in \text{SERVICES} \ \text{not theorem} \\
& \text{grd2}: \ \text{prop} \subseteq \text{PEERS} \ \text{not theorem} \\
& \text{grd3}: \ \text{inst\_state}(\text{token\_owner}(s) \Rightarrow s) = \text{FAIL\_DETECT}_4 \ \text{not theorem} \\
& \text{grd4}: \ \text{suspct\_peers}(\text{token\_owner}(s) \Rightarrow s) \neq \emptyset \ \text{not theorem} \\
& \text{grd5}: \ \text{prop} = \text{run\_inst}(\text{token\_owner}(s) \Rightarrow s) \setminus (\text{unav\_peers} \cup \\
& \text{suspct\_peers}(\text{token\_owner}(s) \Rightarrow s)) \ \text{not theorem} \\
\end{align*}

THEN
\begin{align*}
& \text{act1}: \ \text{inst\_state} = \text{inst\_state} \land ((\text{prop} \times \{s\}) \times \{\text{FAIL\_ACTIV}_4\}) \\
& \text{act2}: \ \text{run\_inst} = \text{run\_inst} \land ((\text{prop} \times \{s\}) \times \{\text{run\_inst}(\text{token\_owner} \\
& (s) \Rightarrow s) \setminus \text{suspct\_peers}(\text{token\_owner}(s) \Rightarrow s)) \\
& \text{act3}: \ \text{failr\_inst} = \text{failr\_inst} \land ((\text{prop} \times \{s\}) \times \{\text{failr\_inst} \\
& (\text{token\_owner}(s) \Rightarrow s) \cup \text{suspct\_peers}(\text{token\_owner}(s) \Rightarrow s)) \\
& \text{act4}: \ \text{suspct\_peers} = \text{suspct\_peers} \land ((\text{prop} \times \{s\}) \times \emptyset) \\
\end{align*}

END

FAIL\_CONFIGURE: extended ordinary

\text{REFINES}

FAIL\_CONFIGURE

\text{ANY}
\begin{align*}
& s \ \triangleright \\
& \text{prop} \ \triangleright \\
\text{WHERE}
& \text{grd1}: \ s \in \text{SERVICES} \ \text{not theorem} \\
& \text{grd2}: \ \text{prop} \subseteq \text{PEERS} \ \text{not theorem} \\
& \text{grd3}: \ \text{inst\_state}(\text{token\_owner}(s) \Rightarrow s) = \text{FAIL\_ACTIV}_4 \ \text{not theorem} \\
& \text{grd4}: \ \text{card}(\text{run\_inst}(\text{token\_owner}(s) \Rightarrow s)) < \text{min\_inst}(s) \ \text{not theorem} \\
& \text{grd5}: \ \text{prop} = \text{run\_inst}(\text{token\_owner}(s) \Rightarrow s) \setminus \text{unav\_peers} \ \text{not theorem} \\
\end{align*}

THEN
\begin{align*}
& \text{act1}: \ \text{inst\_state} = \text{inst\_state} \land ((\text{prop} \times \{s\}) \times \{\text{FAIL\_CONF}\_4\}) \\
\end{align*}

END

FAIL\_IGNORE: extended ordinary

\text{REFINES}

FAIL\_IGNORE
FAIL_IGNORE

ANY
  s →
  prop →
WHERE
  grd1: s ∈ SERVICES not theorem →
  grd2: prop ⊆ PEERS not theorem →
  grd3: \text{inst\_state}(\text{token\_owner}(s) \mapsto s) = \text{FAIL\_ACTIV\_4} not theorem →
  grd4: \text{card}(\text{run\_inst}(\text{token\_owner}(s) \mapsto s)) \geq \text{min\_inst}(s) not theorem →
  grd5: prop = run\_inst(\text{token\_owner}(s) \mapsto s)\backslash\text{unav\_peers} not theorem →
THEN
  act1: \text{inst\_state} ≔ \text{inst\_state} \triangleright ((prop \times \{s\}) \times \{\text{FAIL\_IGN\_4}\}) →
END

IGNORE: extended ordinary →
REFINES
  IGNORE
ANY
  s →
  prop →
WHERE
  grd1: s ∈ SERVICES not theorem →
  grd2: prop ⊆ PEERS not theorem →
  grd3: \text{inst\_state}(\text{token\_owner}(s) \mapsto s) = \text{FAIL\_IGN\_4} not theorem →
  grd4: prop = run\_inst(\text{token\_owner}(s) \mapsto s)\backslash\text{unav\_peers} not theorem →
THEN
  act1: \text{inst\_state} = \text{inst\_state} \triangleright ((prop \times \{s\}) \times \{\text{RUN\_4}\}) →
END

REDEPLOY\_INSTC: extended ordinary →
REFINES
  REDEPLOY\_INSTC
ANY
  s → a service s
  i → an instance i
WHERE
  grd1: s ∈ SERVICES not theorem →
  grd2: i ∈ PEERS not theorem →
  grd3: i \notin \text{run\_inst}(\text{token\_owner}(s) \mapsto s) \cup \text{failr\_inst}
  (\text{token\_owner}(s) \mapsto s) \cup \text{unav\_peers} \cup \text{dep\_instcs}(\text{token\_owner}(s) \mapsto s) not theorem
  → i does not run s, is not failed for s, is not unavailable and is not already activated for s
  grd4: i \notin \text{actv\_instc}(\text{token\_owner}(s) \mapsto s) not theorem →

M21

\[ \text{REDEPLOY INSTS: extended ordinary} \]
\[ \text{REDEPLOY INSTS} \]

\[ \text{REDEPLOY ANY} \]
\[ \text{REDEPLOY prop} \]

\[ \text{WHERE} \]
\[ \text{grd1: } s \in \text{SERVICES not theorem} \]
\[ \text{grd2: } \text{prop} \subseteq \text{PEERS not theorem} \]
\[ \text{grd3: } \text{card(\text{actv_instc(token_owner(s) }\mapsto\text{s})})} = \text{deplo_inst(s)} \text{ not theorem} \]
\[ \text{grd4: } \text{card(\text{dep_instcs(token_owner(s) }\mapsto\text{s})})} + \text{card(\text{run_inst(token_owner(s) }\mapsto\text{s})})} < \text{min_inst(s)} \text{ not theorem} \]
\[ \text{grd5: } \text{inst_state(token_owner(s) }\mapsto\text{s})} = \text{FAIL_CONFIG_4 not theorem} \]
\[ \text{grd6: } \text{prop = run_inst(token_owner(s) }\mapsto\text{s})\setminus\text{unav_peers not theorem} \]

\[ \text{THEN} \]
\[ \text{act1: } \text{dep_instcs = dep_instcs }\cup\text{(prop }\times\text{s})\times\{\text{dep_instcs(token_owner(s) }\mapsto\text{s})\})} \]
\[ \text{act2: } \text{actv_instc(token_owner(s) }\mapsto\text{s})} = \emptyset \]

\[ \text{END} \]

\[ \text{REDEPLOY: extended ordinary} \]
\[ \text{REDEPLOY ANY} \]
\[ \text{REDEPLOY prop} \]

\[ \text{WHERE} \]
\[ \text{grd1: } s \in \text{SERVICES not theorem} \]
\[ \text{grd2: } \text{prop} \subseteq \text{PEERS not theorem} \]
\[ \text{grd3: } \text{inst_state(token_owner(s) }\mapsto\text{s})} = \text{FAIL_CONFIG_4 not theorem} \]
\[ \text{grd4: } \text{actv_instc(token_owner(s) }\mapsto\text{s})} = \emptyset \text{ not theorem} \]
\[ \text{grd5: } \text{dep_instcs(token_owner(s) }\mapsto\text{s})} \neq \emptyset \text{ not theorem} \]
\[ \text{grd6: } \text{card(\text{run_inst(token_owner(s) }\mapsto\text{s})})} + \text{card(\text{dep_instcs(token_owner(s) }\mapsto\text{s})})} \]
(token_owner(s) ↦ s)) ≥ min_inst(s) not theorem >
  grd7: prop = run_inst(token_owner(s) ↦ s)\unav_peers not theorem >
  THEN
  act1: inst_state= inst_state ː ((prop×{s})×{DPL_4})
  act2: run_inst = run_inst ː ((prop×{s})× {run_inst(token_owner
  (s) ↦ s) ∪ dep_instcs(token_owner(s) ↦ s)})
  act3: dep_instcs = dep_instcs ː ((prop×{s})×{∅})
  END

HEAL:  extended ordinary >
REFINES
HEAL
ANY
  s >
  prop >
WHERE
  grd1: s ∈ SERVICES not theorem >
  grd2: prop ⊆ PEERS not theorem >
  grd3: inst_state(token_owner(s) ↦ s) = DPL_4 not theorem >
  grd4: prop = run_inst(token_owner(s) ↦ s)\unav_peers not theorem >
  THEN
  act1: inst_state= inst_state ː ((prop×{s})×{RUN_4})
  END

UNFAIL_PEER:  extended ordinary >
REFINES
UNFAIL_PEER
ANY
  s >
  p >
  prop >
WHERE
  grd1: s ∈ SERVICES not theorem >
  grd2: prop ⊆ PEERS not theorem >
  grd3: p ∈ PEERS not theorem >
  grd4: p ∈ failr_inst(token_owner(s) ↦ s) not theorem >
  grd5: prop = run_inst(token_owner(s) ↦ s)\unav_peers not theorem >
  THEN
  act1: failr_inst = failr_inst ː ((prop×{s})×{failr_inst
  (token_owner(s) ↦ s) \ {p}})
  END

MAKE_PEER_AVAIL:  extended ordinary >
REFINES
MAKE_PEER_AVAIL
M21

ANY
   p   >
WHERE
   grd1:  p ∈ PEERS not theorem >
   grd2:  p ∈ unav_peers not theorem >
THEN
   act1:  unav_peers = unav_peers \ {p} >
END

END