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-⋆ and P2P Systems using Refinement

Manamiary Bruno Andriamiarina, Dominique Méry, and Neeraj Kumar Singh

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-⋆ systems. The self-⋆ 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-⋆ systems along with an interesting case study related to the P2P-based self-healing protocol.

Keywords: Distributed systems, self-⋆, 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-⋆ 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-⋆ 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 analyzing a similar class of systems for years, namely distributed algorithms.

In this study, we use the correct by construction approach [12] for modeling 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 $CL$ of possible events of the system $S$. The events of $CL$ 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 $F$ 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 $ST$ 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 $CV(\subseteq ST)$ 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 $M = CL \cup ST \cup F$, where the model $M$ contains a set ($CL$) of events for representing the computation steps of the system $S$. When a fault occurs, a set ($ST$) 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 characterize 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 $ST$ modelling the stabilization of the system from illegal/failed states. We ensure that the convergence is always possible: a subset $CV$ of $ST$ 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 $ST$ 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 \textit{what}) or a local update of a variable (the \textit{how}). We formalise the system \(S\) using the \textsc{EVENT B} modelling language \cite{1}, dealing with \textit{events} and \textit{invariant} properties including \textit{convergence} properties by using a temporal framework. The \textit{service-as-event} paradigm \cite{3} helps to express this \textit{concretisation} process: the procedures (1) \textit{leading} from the \textit{illegal states} to the \textit{recovery states}, and (2) \textit{leading} from the \textit{recovery states} to the \textit{legal states} are stated by (abstract) events, during the first stages of the \textsc{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 \textsc{EVENT B} modelling framework including \textit{service-as-event} paradigm and a formal definition of self-\(\ast\) systems. Section 4 presents the formal verification approach and illustrates the proposed methodology with the study of the self-healing P2P-based protocol \cite{14}. Section 5 discusses on approaches for studying temporal properties for \textsc{EVENT B} models. Finally, Section 6 concludes the paper along with future work.

\section{Related Works on Formal Modelling for Self-\(\ast\) Systems}

Systems usually run in intricate environments, with frequent and unexpected changes. This feature increases interest towards autonomous and self-\(\ast\) 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-\(\ast\) systems originates from the needs of understanding how these systems behave and how they meet their specifications. A self-\(\ast\) system relies on \textit{emergent behaviours}, resulting from interactions between components of the system \cite{21}.

Traditionally, the correctness of self-\(\ast\) and autonomous systems is validated through the simulation and testing \cite{20,22}. However, simulation and testing are not sufficient to cover the whole set of possible states of a system \cite{2}. Therefore, formal methods appear as a promising land for validating self-\(\ast\) 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 \cite{6,8,24}.

Smith et al. \cite{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. \cite{6} have used Alloy to demonstrate the correctness of the autonomic computing policies (ACP). However, Alloy does not provide a mechanism for expressing the \textit{correct-by-construction} paradigm. Méry et al. \cite{2} have also investigated a self-reconfiguring system (Network-on-Chip: adaptative XY routing) using the \textsc{EVENT B} framework and the \textit{correct-by-construction} approach.

State exploration approaches such as model-checking are also used to study self-\(\ast\) systems. Model-checkers like \textsc{SPIN}, \textsc{PRISM}, \textsc{SMV}, \textsc{UPPAAL} are used for properties specification and getting evidences that properties, such as flexibility, robustness of the self-\(\ast\) systems hold \cite{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 toolset 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 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: $\text{ANY } t \text{ WHERE } G(t,x) \text{ THEN } x : P(t,x,x') \text{ END}$ 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: $\text{INV1}$ and $\text{INV2}$ 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 $e$; $\text{FIS}$ 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>$\text{INV1}$</th>
<th>$\text{INV2}$</th>
<th>$\text{FIS}$</th>
</tr>
</thead>
<tbody>
<tr>
<td>$\text{Init}(x) \Rightarrow I(x)$</td>
<td>$I(x) \land BA(e)(x,x') \Rightarrow I(x')$</td>
<td>$I(x) \land grd(e)(x) \Rightarrow \exists z : BA(e)(x,z)$</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(ae)(x, x') and BA(ce)(y, y') are predicates of events ae and ce respectively; we have to discharge the following proof obligation:

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

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-\(\star\) systems, we use the service description and decomposition that is provided by the service-as-event [3] 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-\(\star\) 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-\(\star\) systems.

A Definition of Self-\(\star\) Mechanism We characterize a self-stabilizing system \(S\) (more generally a self-\(\star\) 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: \((\neg P) \rightarrow P\). 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: \(Spec(M): Init(y) \land \Box [Next(y) \land L\), where \(Init(y)\) is the predicate specifying initial states; \(Next \equiv \exists e \in E.BA(e)(y, y')\) is an action formula representing the next-state relation; and \(L\) is a conjunction of formulas \(WF_{\gamma}(e)\): 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.iti.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 \( M \) satisfies \( P \rightarrow Q \) and \( Q \rightarrow R \), it satisfies \( P \rightarrow R \).
2. If \( M \) satisfies \( P \rightarrow Q \) and \( R \rightarrow Q \), it satisfies \( (P \lor R) \rightarrow Q \).
3. If \( I \) is invariant for \( M \) and if \( M \) satisfies \( P \land I \rightarrow Q \), then \( M \) satisfies \( P \rightarrow Q \).
4. If \( I \) is invariant for \( M \) and if \( M \) satisfies \( P \land I \Rightarrow Q \), then \( M \) satisfies \( P \Rightarrow Q \).
5. If \( P \not\rightarrow Q \) is a link of \( D \) for the machine \( M \), then \( M \) satisfies \( P \not\rightarrow Q \).
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 \( P \rightarrow Q \).
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 \( P \rightarrow Q \).

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 \( (\neg P) \leadsto P \), 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: \( (\neg P) \leadsto P \). An abstract event \( (e) \) is used for describing the service/procedure represented by \( (\neg P) \leadsto P \): \( (\neg P) \stackrel{\epsilon}{\leadsto} P \); 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:

\[
\begin{align*}
&\vdots \quad \vdots \quad \text{trans}_3 \quad \vdots \quad \text{trans}_4 \quad \vdots \quad \text{trans}_5 \\
&\quad \quad R_0 \leadsto R_1 \quad \quad R_0 \leadsto P \\
\hline
&\quad \quad \vdots \quad \text{trans}_1 \\
&\quad \quad \vdots \\
&\quad \quad \vdots \\
&\quad \quad \vdots \\
&(-P) \leadsto P
\end{align*}
\]

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 \((-P) \leadsto P\) is decomposed, until the identification of the stabilization steps is satisfactory. The stabilization phase is expressed by the property \((\neg P) \leadsto R_0 \wedge R_0 \leadsto R_1 \wedge \ldots \wedge R_{n-1} \leadsto R_n \wedge R_n \leadsto P\), 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 M0 abstracts the self-healing approach. The refinements M1, M2, M3 introduce step-by-step the self-detection, self-activation and self-configuration phases, respectively. The remaining refinements, from M4 to M20, are used for localisation of the system: each step of the algorithm is made local to a node. The last refinement M21 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: RUN (legal/running state) and FAIL (illegal/faulty state). A variable serviceState is defined as $s \mapsto st \in serviceState$, 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 $P \equiv (s \mapsto RUN \in serviceState)$. An event FAILURE models a faulty behaviour, where service (s) enters into a faulty state (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 HEAL ($\neg P \Rightarrow P$), where service (s) recovers from an illegal faulty state (FAIL) to a legal running state (RUN). The refinement diagram (see Fig.7) and events sum up the abstraction of a recovery procedure.

**Introducing the Self-Detection (M1)** The variable serviceState is replaced, by refinement, with a new variable serviceState_1, since new states are introduced. The states RUN, FAIL are refined into RUN_1, FAIL_1, and a new state (FL_DT_1) is defined. A service (s) can suspect and identify a failure state (FAIL_1) before triggering the recovery (HEAL). We introduce a property $R_0 \equiv (s \mapsto FL_DT_1 \in serviceState_1)$ and a new event FAILURE_DETECT in this self-detection mechanism. Let P and $\neg P$ be redefined as follows: $P \equiv (s \mapsto RUN_1 \in serviceState_1)$ and $\neg P \equiv (s \mapsto FAIL_1 \in serviceState_1)$.

1. The assertions $(s \mapsto st \in serviceState)$, describing the state (st) of a service (s), are shorten into $(st)$, in the nodes of the refinement diagrams, for practical purposes.
2. $\oplus$: to add elements to a model, $\ominus$: 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.

\[
\begin{align*}
\neg P \Rightarrow R_0 & \Rightarrow P \quad \text{trans} \\
\neg P \Rightarrow P &
\end{align*}
\]

The event FAIL_DETECT is introduced to express the self-detection: the failure state (FAIL_1) of a service (s) is detected (state FL_DT_1).

The property \((\neg P) \Rightarrow R_0\) is expressed by the event FAIL_DETECT, where the failure (FAIL_1) of service (s) is identified (FL_DT_1). \(R_0 \Rightarrow P\) is defined by the event HEAL, where the service (s) is restored to a legal running state (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 web³.

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 FL_ACT_2 and an event 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 FL_CONF_3), otherwise, the failure will be ignored (state 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 FAIL_DETECT and IS_OK. The event FAIL_DETECT models the failure detection; and the event 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 (RUN_4). Self-Activation process is used to evaluate when actual failures are identified, using

³ http://eb2all.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 (sus_p_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 3. 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 (sus) 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 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_AVAIL: 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_AVAIL: 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.

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
  aam1: partition(STATES_1, {RUN_1},{FAIL_1},{FAIL_DETECT_1}) not theorem
  aam2: InitState_1 ∈ SERVICES ↔ STATES_1 not theorem
  aam3: ∀ s : s ∈ SERVICES ⇒ s ↦ RUN_1 ∈ InitState_1 not theorem
  aam4: ∀ 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: \( \text{partition}(\text{STATES}_2, \{\text{RUN}_2\}, \{\text{FAIL}_2\}, \{\text{FAIL\_DETECT}_2\}, \{\text{FAIL\_ACTIV}_2\}) \) not theorem
axm2: \( \text{InitState}_2 \in \text{SERVICES} \leftrightarrow \text{STATES}_2 \) not theorem
axm3: \( \forall s \cdot s \in \text{SERVICES} \Rightarrow s \mapsto \text{RUN}_2 \in \text{InitState}_2 \) not theorem
axm4: \( \forall s, st_1, st_2 \cdot s \in \text{SERVICES} \land st_1 \in \text{STATES}_2 \land st_2 \in \text{STATES}_2 \land s \mapsto st_1 \in \text{InitState}_2 \land s \mapsto st_2 \in \text{InitState}_2 \Rightarrow st_1 = st_2 \) not theorem
END
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 ∈ SERVICES ⇒ s ∈ InitState_4 not theorem
  axm4: ∀ s, st1, st2 ∈ SERVICES ∧ st1 ∈ STATES_4 ∧ st2 ∈ STATES_4 ∧
  s ∈ InitState_4 ∧ s ∈ 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  \rightarrow Set of PEERS
CONSTANTS
  InitSrvcPeers  \rightarrow Initial set of peers / instances per service
AXIOMS
  axm1:  InitSrvcPeers ∈ SERVICES \rightarrow ℙ1(PEERS) not theorem \rightarrow each service is provided by a non empty set of peers/instances
  axm2:  ∀ s · s ∈ SERVICES ⇒ finite(InitSrvcPeers(s)) not theorem \rightarrow each service is provided by a finite set of peers/instances
  axm3:  ∀ s · s ∈ SERVICES ⇒ card(InitSrvcPeers(s)) = init_inst(s) not theorem \rightarrow 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

 EXTENDS

 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 ⩤ {s1 ↦ p})[{s2}] = set[{s2}] theorem

 axm4: ∀ set, s1, s2, p · set ⊆ SERVICES×PEERS ∧ s1 ∈ SERVICES ∧ s2 ∈ SERVICES ∧ p ∈ PEERS ∧ s1 ≠ s2 ⇒ (set ⩤ {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
  InitState_srv
  InitSuspPrs
  InitRunPeers
AXIOMS
  axm1: InitState_srv ∈ PEERS × SERVICES ↔ STATES.4 not theorem
  axm2: ∀ s, p · p ∈ PEERS ∧ s ∈ SERVICES ∧ p ∈ InitSrvcPeers(s) ⇒ (p ↦ s) ↦ RUN.4 ∈ InitState_srv not theorem
  axm3: ∀ s, p, stt · p ∈ PEERS ∧ s ∈ SERVICES ∧ (p ↦ s) ↦ stt ∈ InitStatesrv ⇒ 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
MACHINE

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
  gluing_run1: ∀ s · s ∈ SERVICES ∧ s ↦ RUN ∈ serviceState ⇒ s ↦ RUN_1 ∈ serviceState_1
  gluing_run2: ∀ s · s ∈ SERVICES ∧ s ↦ RUN_1 ∈ serviceState_1 ⇒ s ↦ RUN ∈ serviceState
  gluing_fail1: ∀ s · s ∈ SERVICES ∧ s ↦ FAIL ∈ serviceState ⇒ (s ↦ FAIL_1 ∈ serviceState_1 ∨ s ↦ FAIL_DETECT_1 ∈ serviceState_1)
  gluing_fail2: ∀ s, st · s ∈ SERVICES ∧ st ∈ STATES_1 ∧ st ∈ {FAIL_1,FAIL_DETECT_1} ∧ st ∈ serviceState_1 ⇒ s ↦ FAIL ∈ serviceState
  gluing_state3: ∀ s, st1, st2 · s ∈ SERVICES ∧ st1 ∈ STATES_1 ∧ st2 ∈ STATES_1 ∧ s ↦ st1 ∈ serviceState_1 ∧ s ↦ st2 ∈ serviceState_1 ⇒ st1 = st2
EVENTS
  INITIALISATION: not extended ordinary
    THEN
      act1: serviceState_1 ≔ InitState_1
    END
FAIL: not extended ordinary
  REFINES FAIL
  ANY
    s
    WHERE
      grd1: s ∈ SERVICES
      grd2: s ↦ RUN_1 ∈ serviceState_1
    THEN
      act1: serviceState_1 ≔ (serviceState_1∖{s ↦ RUN_1}) u {s ↦
    END
FAIL_DETECT: not extended ordinary
  REFINES FAIL
  ANY
    s
    WHERE
      grd1: s ∈ SERVICES
M01

\text{grd2: } s \Rightarrow \text{FAIL}_1 \in \text{serviceState}_1 \text{ not theorem } \rangle
\begin{align*}
\text{THEN} & \quad \text{act1: } \text{serviceState}_1 = (\text{serviceState}_1 \setminus \{s \Rightarrow \text{FAIL}_1\}) \cup \{s \Rightarrow \\
& \text{FAIL\_DETECT\_1}\} \rangle
\end{align*}
\text{END}

\text{HEAL: } \text{not extended ordinary } \rangle
\text{REFINES }
\text{HEAL}
\text{ANY }
\begin{align*}
\text{s } & \rangle
\text{WHERE }
\text{grd1: } s \in \text{SERVICES} \text{ not theorem } \rangle
\text{grd2: } s \Rightarrow \text{FAIL\_DETECT\_1} \in \text{serviceState}_1 \text{ not theorem } \rangle
\text{THEN} & \quad \text{act1: } \text{serviceState}_1 = (\text{serviceState}_1 \setminus \{s \Rightarrow \text{FAIL\_DETECT\_1}\})
\cup \{s \Rightarrow \text{RUN}_1\} \rangle
\end{align*}
\text{END}
\text{END}
MACHINE M02
REFINES M01
SEES C02

INVIARS

INITIALISATION: 

FAILURE: 
REFINES FAI

BRE

FAILURE_DETECT: 
REFINES
\begin{verbatim}
M02

IF AクトCT
\end{verbatim}
MACHINE M03
  REFINES M03
  SEES C03
  RIAES

  serv_initialisation

  INVAR
  \begin{align*}
  & L_3 \in \text{SERVICES} \land S \in \text{STATES} \\
  & \forall \text{INITIALISATION:} \quad & \text{a} \in \text{INITIALISATION_3} \land I \in \text{INITIALISATION_3}
  \end{align*}

  EENTS
  \begin{align*}
  & \text{FAIL:} \quad \text{FAIL} \land \text{INITIALISATION_3} \land I \text{INITIALISATION_3}
  \end{align*}

  THEN
  \begin{align*}
  & \text{FAIL} \land \text{INITIALISATION_3} \land I \text{INITIALISATION_3}
  \end{align*}

  END

  \begin{align*}
  & \text{FAIL} \land \text{SERVICES} \land \text{INITIALISATION_3} \land I \text{INITIALISATION_3}
  \end{align*}

  THEN
  \begin{align*}
  & \text{FAIL} \land \text{SERVICES_3} \land \text{INITIALISATION_3} \land I \text{INITIALISATION_3}
  \end{align*}

  \begin{align*}
  & \text{FAIL} \land \text{SERVICES_3} \land \text{INITIALISATION_3} \land I \text{INITIALISATION_3}
  \end{align*}
FAIL_DETECT: not extended ordinary
   REFINES
      FAIL_DETECT
   ANY
      \s
   \BRE
      \s in SERVICES not theorem
      serviceState_3 \subseteq \servicestate_3 not theorem
      THEN
      a \subseteq \servicestate_3 \setminus \ser{\{FAIL\_DETECT_3\}}
   FAI\_DETECT/3
   END

IS_OK: not extended ordinary
   REFINES
      IS_OK
   ANY
      \s
   \BRE
      \s in SERVICES not theorem
      serviceState_3 \subseteq \servicestate_3 not theorem
      THEN
      a \subseteq \servicestate_3 \setminus \ser{\{FAIL\_DETECT_3\}}
   FAI\_DETECT/3
   END

FAIL\_ACTI\_N: not extended ordinary
   ANY
   \s
   \s
   \BRE
   \s in SERVICES not theorem
   serviceState_3 \subseteq \servicestate_3 not theorem
   THEN
   a \subseteq \servicestate_3 \setminus \ser{\{FAIL\_DETECT_3\}}
   FAI\_DETECT/3
   END

FAIL\_CONF\_IGN: not extended ordinary
   ANY
   \s
   \s
   \BRE
   \s in SERVICES not theorem
   serviceState_3 \subseteq \servicestate_3 not theorem
   THEN
   a \subseteq \servicestate_3 \setminus \ser{\{FAIL\_DETECT_3\}}
   FAI\_DETECT/3
   END
M03

\[\text{grd2: } s \mapsto \text{FAIL_ACTIV}_3 \in \text{serviceState}_3 \text{ not theorem } \]
\[\text{THEN } \]
\[\text{act1: } \text{serviceState}_3 = (\text{serviceState}_3 \setminus \{s \mapsto \text{FAIL_ACTIV}_3\}) \cup \{s \mapsto \text{st}\} \]
\[\text{END}\]

\[\text{HEAL: } \text{not extended ordinary } \]
\[\text{REFINES } \]
\[\text{HEAL } \]
\[\text{ANY }\]
\[\text{s } \]
\[\text{st } \]
\[\text{WHERE }\]
\[\text{grd1: } s \in \text{SERVICES} \text{ not theorem } \]
\[\text{grd3: } \text{st} \in \{\text{FAIL_CONFIG}_3, \text{FAIL_IGN}_3\} \text{ not theorem } \]
\[\text{grd2: } s \mapsto \text{st} \in \text{serviceState}_3 \text{ not theorem } \]
\[\text{THEN } \]
\[\text{act1: } \text{serviceState}_3 = (\text{serviceState}_3 \setminus \{s \mapsto \text{st}\}) \cup \{s \mapsto \text{RUN}_3\} \]
\[\text{END}\]

\[\text{END}\]
MACHINE M04
REFINES M03
SEES C04
VARIABLES serviceState_4

INVARIANTS
inv1: serviceState_4 ∈ SERVICES ↔ STATES_4
inv2: serviceState_4 ∈ SERVICES ↔ serviceState_4 ∈ STATES_4

EVENTS
INITIALISATION:
THEN
a
EN

FAIL
REFINES FAI
\[ M_0 \]

\[ \text{AN} \]

\[ \text{HERE} \]

\[ WIBES \text{ not theorem} \]

\[ \text{gr} \]

\[ R\text{EN}_4 \text{ not theorem} \]

\[ \text{THEN} \]

\[ a \text{FAIL}_{\text{DETECT}_4} \]

\[ 4 \text{FAIL}_{\text{DETECT}_4} \]

\[ 4 \text{FAIL}_{\text{ACTIV}_4} \]

\[ \text{FAIL}_{\text{ACTIV}_4} \]

\[ \text{FAIL}_{\text{DETECT}_4} \]

\[ \text{REFINES} \]

\[ \text{FAIL}_{\text{DETECT}_4} \]

\[ \text{AN} \]

\[ \text{HERE} \]

\[ WIBES \text{ not theorem} \]

\[ \text{gr} \]

\[ \text{FAIL}_{\text{DETECT}_4} \text{ not theorem} \]

\[ \text{THEN} \]

\[ a \text{FAIL}_{\text{DETECT}_4} \]

\[ 4 \text{FAIL}_{\text{DETECT}_4} \]

\[ 4 \text{FAIL}_{\text{ACTIV}_4} \]

\[ \text{FAIL}_{\text{ACTIV}_4} \]

\[ \text{IS}_{\text{OK}_4} \]

\[ \text{REFINES} \]

\[ \text{IS}_{\text{OK}_4} \]

\[ \text{AN} \]

\[ \text{HERE} \]

\[ WIBES \text{ not theorem} \]

\[ \text{gr} \]

\[ \text{FAIL}_{\text{DETECT}_4} \text{ not theorem} \]

\[ \text{THEN} \]

\[ a \text{FAIL}_{\text{DETECT}_4} \]

\[ 4 \text{FAIL}_{\text{DETECT}_4} \]

\[ 4 \text{FAIL}_{\text{ACTIV}_4} \]

\[ \text{FAIL}_{\text{ACTIV}_4} \]

\[ \text{FAIL}_{\text{DETECT}_4} \]

\[ \text{REFINES} \]

\[ \text{FAIL}_{\text{DETECT}_4} \]

\[ \text{AN} \]

\[ \text{HERE} \]

\[ WIBES \text{ not theorem} \]

\[ \text{gr} \]

\[ \text{FAIL}_{\text{ACTIV}_4} \text{ not theorem} \]

\[ \text{THEN} \]

\[ a \text{FAIL}_{\text{ACTIV}_4} \]

\[ 4 \text{FAIL}_{\text{ACTIV}_4} \]

\[ 4 \text{FAIL}_{\text{ACTIV}_4} \]

\[ \text{FAIL}_{\text{ACTIV}_4} \]
FAIL_CONFIGURE: Refines FAIL_CONFIGURATION

FAIL_CONFIGURATION

s >>

s SUBCES not theorem

FAIL_AC_4 serviceState_4

st = FAIL_CONFIG_3

act1: serviceState_4 ≔ (serviceState_4 ∖ {s ↦ FAIL_AC_4}) ∪ {s ↦ FAILCONFIG_4}

END

FAIL_IGNORE: Refines FAIL_CONFIGURATION

FAIL_CONFIGURATION

s >>

s SUBCES not theorem

FAIL_AC_4 serviceState_4

st = FAIL_CONFIG_3

act1: serviceState_4 ≔ (serviceState_4 ∖ {s ↦ FAIL_AC_4}) ∪ {s ↦ FAIL_IGNORE_4}

END

IGNORE: Refines HEAL

HEAL

s >>

s WIRCES not theorem

FAIL IGNORE_4

st = FAIL IGNORE_3

act1: serviceState_4 ≔ (serviceState_4 ∖ {s ↦ FAIL_IGNORE_4}) ∪ {s ↦ RUN_4}

END

REDEPLOY: Refines HEAL

HEAL

s >>

s RON_4

END
WHERE \( gr_1 \): \( Y \in \text{SERVICES} \) not theorem

\( gr_2 \): \( Y \mapsto \text{FAIL_CONFIG_4} \in \text{YHrvicHStaH_4} \) not thHorHm

\( ct_1 \): \( \text{YHrvicHStatH_4} := (\text{YHrvicHStatH_4} \setminus \{Y \mapsto \text{FAIL_CONFIG_4}\}) \cup \{Y \mapsto \text{DPL_4}\} \)

\( \text{HEAL_4} \): \( \text{not HxtHndHd ordinary} \)

REFINES HEAL

WHERE \( gr_1 \): \( Y \in \text{SERVICES} \) not theorem

\( gr_2 \): \( Y \mapsto \text{DPL_4} \in \text{YHrvicHStatH_4} \) not thHorHm

\( Yt \): \( \text{Yt=FAIL_CONFIG_3} \)

HEN

\( \text{ct_1} \): \( \text{YHrvicHStatH_4} := (\text{YHrvicHStatH_4} \setminus \{Y \mapsto \text{DPL_4}\}) \cup \{Y \mapsto \text{RUN_4}\} \)

END
MACHINE M05
REFINES M04
SEES C05
VARIABLES
  serviceState_4
  num_run
  num_susp
INVIARNTS
  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:
  THEN
    act1: serviceState_4 ≔ InitState_4
    act2: num_run ≔ init_inst
    act3: num_susp ≔ SERVICES×{0}
  END

  FAIL:
  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:
  REFINES FAIL_DETECT

Page 1
M05

\[\text{ANY} \quad s \quad \text{num_safe} \quad \text{WHERE}
\]

grd1: \(s \in \text{SERVICES} \not\text{theorem}\) 
grd2: \(s \rightarrow \text{FAIL} \in \text{serviceState} \not\text{theorem}\) 
grd3: \(\text{num_safe} \in \mathbb{N} \not\text{theorem}\) 
grd4: \(\text{num_safe} \leq \text{num_susp}(s) \not\text{theorem}\) 

\[\text{THEN}
\]

act1: \(\text{serviceState} = (\text{serviceState} \setminus \{s \rightarrow \text{FAIL}\}) \cup \{s \rightarrow \text{FAIL_DETECT}\}\) 
act2: \(\text{num_susp}(s) = \text{num_susp}(s) - \text{num_safe}\) 

\[\text{END}\]

\[\text{IS_OK: extended ordinary} \quad \text{REFINES} \quad \text{IS_OK}\]

\[\text{ANY} \quad s \quad \text{WHERE}
\]

grd1: \(s \in \text{SERVICES} \not\text{theorem}\) 
grd2: \(s \rightarrow \text{FAIL_DETECT} \in \text{serviceState} \not\text{theorem}\) 
grd3: \(\text{num_susp}(s) = 0 \not\text{theorem}\) 

\[\text{THEN}
\]

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

\[\text{END}\]

\[\text{FAIL_ACTIV: extended ordinary} \quad \text{REFINES} \quad \text{FAIL_ACTIV}\]

\[\text{ANY} \quad s \quad \text{WHERE}
\]

grd1: \(s \in \text{SERVICES} \not\text{theorem}\) 
grd2: \(s \rightarrow \text{FAIL_DETECT} \in \text{serviceState} \not\text{theorem}\) 
grd3: \(\text{num_susp}(s) > 0 \not\text{theorem}\) 

\[\text{THEN}
\]

act1: \(\text{serviceState} = (\text{serviceState} \setminus \{s \rightarrow \text{FAIL_DETECT}\}) \cup \{s \rightarrow \text{FAIL_ACTIV}\}\) 
act2: \(\text{num_run}(s) = \text{num_run}(s) - \text{num_susp}(s)\) 
act3: \(\text{num_susp}(s) = 0\) 

\[\text{END}\]

\[\text{FAIL_CONFIGURE: extended ordinary} \quad \text{REFINES} \quad \text{FAIL_CONFIGURE}\]

\[\text{ANY}\]

Page 2
\[ s \rightarrow \]

WHERE
\[ \begin{align*}
& \text{grd1: } s \in \text{SERVICES} \not\text{ theorem} \\
& \text{grd2: } s \mapsto \text{FAIL_ACTIV}_4 \in \text{serviceState}_4 \not\text{ theorem} \\
& \text{grd3: } \text{num\_run}(s) < \text{min\_inst}(s) \not\text{ theorem}
\end{align*} \]

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: \text{ extended ordinary } \not\text{ theorem}

\text{REFINES}

FAIL_IGNORE

\text{ANY}

\[ s \rightarrow \]

WHERE
\[ \begin{align*}
& \text{grd1: } s \in \text{SERVICES} \not\text{ theorem} \\
& \text{grd2: } s \mapsto \text{FAIL_ACTIV}_4 \in \text{serviceState}_4 \not\text{ theorem} \\
& \text{grd3: } \text{num\_run}(s) \geq \text{min\_inst}(s) \not\text{ theorem}
\end{align*} \]

THEN

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

END

IGNORE: \text{ extended ordinary } \not\text{ theorem}

\text{REFINES}

IGNORE

\text{ANY}

\[ s \rightarrow \]

WHERE
\[ \begin{align*}
& \text{grd1: } s \in \text{SERVICES} \not\text{ theorem} \\
& \text{grd2: } s \mapsto \text{FAIL_IGN}_4 \in \text{serviceState}_4 \not\text{ theorem} \\
& \text{grd3: } \text{num\_run}(s) \geq \text{min\_inst}(s) \not\text{ theorem}
\end{align*} \]

THEN

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

END

REDEPLOY: \text{ extended ordinary } \not\text{ theorem}

\text{REFINES}

REDEPLOY

\text{ANY}

\[ s \rightarrow \]

WHERE
\[ \begin{align*}
& \text{grd1: } s \in \text{SERVICES} \not\text{ theorem} \\
& \text{grd2: } s \mapsto \text{FAIL_CONFIG}_4 \in \text{serviceState}_4 \not\text{ theorem} \\
& \text{grd3: } \text{new\_run} \in N_1 \not\text{ theorem} \\
& \text{grd4: } \text{new\_run} \geq \text{min\_inst}(s) \not\text{ theorem}
\end{align*} \]
THEN
   act1: serviceState_4 = (serviceState_4 \ {s ↦ FAIL_CONFIG_4})
       u {s ↦ DPL_4} >
   act2: num_run(s) = new_run >
END

EAL: 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}) u {s ↦ RUN_4} >
END

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

INVARIANTS
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 > an instance of a service s is either failed or providing the service s
inv5: ∀ s · s ∈ SERVICES ∧ s ∈ dom(susp_peers) ⇒ susp_peers(s) ⊆ run_peers(s) not theorem > suspicious instances of s are a subset of the instances providing s
inv6: ∀ s, st · s ∈ SERVICES ∧ st ∈ STATES_4 ∧ st ∈ {FAIL_4, FAIL_DETECT_4} ∧ s ↦ st ∈ serviceState_4 ⇒ s ∈ dom(susp_peers) not theorem
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
\begin{verbatim}

M06

s > fp >
WHERE
grd1: s ∈ SERVICES not theorem >
grd2: s ↦ RUN_4 ∈ serviceState_4 not theorem >
grd5: fp ⊆ PEERS not theorem >
grd3: fp ≠ ø not theorem >
grd4: fp ⊂ run_peers(s) not theorem >
WITH
  nb_fail: nb_fail=card(fp) >
THEN
  act1: serviceState_4 ≔ (serviceState_4∖{s ↦ RUN_4}) ∪ {s ↦ FAIL_4} >
  act2: susp_peers(s) ≔ fp >
END
FAIL_DETECT: not 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) ≠ ø not theorem >
grd6: sf ⊆ PEERS not theorem >
grd7: sf ⊆ susp_peers(s) not theorem >
WITH
  num_safe: num_safe=card(sf) >
THEN
  act1: serviceState_4 ≔ (serviceState_4∖{s ↦ FAIL_4}) ∪ {s ↦ FAIL_DETECT_4} >
  act2: susp_peers(s) ≔ susp_peers(s) ∖ sf >
END
IS_OK: not 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) = ø not theorem >
THEN
  act1: serviceState_4 ≔ (serviceState_4 \ {s ↦ FAIL_DETECT_4}) \\ {s ↦ RUN_4} >
\end{verbatim}
FAIL_ACTIV:

REFINES
FAIL_ACTIV

ANY
s

WHERE

grd1: s ∈ SERVICES not theorem
grd2: s ↦ FAIL_DETECT_4 ∈ serviceState_4 not theorem
grd5: susp_peers(s) ≠ Ø 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) = Ø
act4: fail_peers = fail_peers \ ({s}×susp_peers(s))

END

FAIL_CONFIGURE:

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:

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 ∈ SERVICES not theorem
p ∈ PEERS not theorem

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
MACHINE M07
REFINES M06
SEES C07

VARIABLES
  serviceState_4
  run_peers
  susp_peers
  fail_peers
  dep_inst

INVARIANTS
inv1: dep_inst ∈ SERVICES ↔ PEERS not theorem
inv2: ∀ s · s ∈ SERVICES ⇒ dep_inst[{s}] ∩ fail_peers[{s}] = ø not theorem
inv3: ∀ s, st · s ∈ SERVICES ∧ st ∈ STATES_4 ∧ s ↦ st ∈ serviceState_4 ∧ st ≠ FAIL_CONFIG_4 ⇒ dep_inst[{s}] = ø not theorem
inv4: ∀ s · s ∈ SERVICES ⇒ finite(dep_inst[{s}]) not theorem
inv5: ∀ s · s ∈ SERVICES ⇒ dep_inst[{s}] ∩ run_peers(s) = ø 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 ≠ ø not theorem
  grd4: fp ⊂ run_peers(s) not theorem

THEN
  act1: serviceState_4 ≔ (serviceState_4{s ↦ RUN_4}) ∪ {s ↦ FAIL_4}
  act2: susp_peers(s) ≔ fp

END
FAIL_DETECT:  extended ordinary  
  \text{REFINES} 
  \text{FAIL\_DETECT} 
  \text{ANY} 
  s  
  sf  
  \text{WHERE} 
  \text{grd1: } s \in \text{SERVICES} \not\Rightarrow \text{theorem} 
  \text{grd2: } s \mapsto \text{FAIL\_DETECT} \notin \text{serviceState\_4} \not\Rightarrow \text{theorem} 
  \text{grd5: } \text{susp\_peers}(s) \neq \emptyset \not\Rightarrow \text{theorem} 
  \text{grd6: } sf \subseteq \text{PEERS} \not\Rightarrow \text{theorem} 
  \text{grd7: } sf \subseteq \text{susp\_peers}(s) \not\Rightarrow \text{theorem} 
  \text{THEN} 
  \text{act1: } \text{serviceState\_4} = (\text{serviceState\_4} \setminus \{s \mapsto \text{FAIL\_DETECT}\}) \cup \{s \mapsto \text{FAIL\_DETECT\_4}\} 
  \text{act2: } \text{susp\_peers}(s) = \text{susp\_peers}(s) \setminus sf 
  \text{END} 

IS_OK:  extended ordinary  
  \text{REFINES} 
  \text{IS\_OK} 
  \text{ANY} 
  s  
  \text{WHERE} 
  \text{grd1: } s \in \text{SERVICES} \not\Rightarrow \text{theorem} 
  \text{grd2: } s \mapsto \text{FAIL\_DETECT\_4} \notin \text{serviceState\_4} \not\Rightarrow \text{theorem} 
  \text{grd5: } \text{susp\_peers}(s) = \emptyset \not\Rightarrow \text{theorem} 
  \text{THEN} 
  \text{act1: } \text{serviceState\_4} = (\text{serviceState\_4} \setminus \{s \mapsto \text{FAIL\_DETECT\_4}\}) 
  \cup \{s \mapsto \text{RUN\_4}\} 
  \text{END} 

FAIL\_ACTIV:  extended ordinary  
  \text{REFINES} 
  \text{FAIL\_ACTIV} 
  \text{ANY} 
  s  
  \text{WHERE} 
  \text{grd1: } s \in \text{SERVICES} \not\Rightarrow \text{theorem} 
  \text{grd2: } s \mapsto \text{FAIL\_DETECT\_4} \notin \text{serviceState\_4} \not\Rightarrow \text{theorem} 
  \text{grd5: } \text{susp\_peers}(s) \neq \emptyset \not\Rightarrow \text{theorem} 
  \text{THEN} 
  \text{act1: } \text{serviceState\_4} = (\text{serviceState\_4} \setminus \{s \mapsto \text{FAIL\_DETECT\_4}\}) 
  \cup \{s \mapsto \text{FAIL\_ACTIV\_4}\} 
  \text{act2: } \text{run\_peers}(s) = \text{run\_peers}(s) \setminus \text{susp\_peers}(s) 
  \text{act3: } \text{susp\_peers}(s) = \emptyset 
  \text{act4: } \text{fail\_peers} = \text{fail\_peers} \cup \{(s)\times\text{susp\_peers}(s)\}
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}) ∪ {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: 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_INST: not extended ordinary
ANY
  s
  dep
WHERE
grd1: \( s \in \text{SERVICES} \) not theorem
grd2: \( \text{dep} \in \text{PEERS} \) 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: \( \text{card}(\text{dep}) = \text{deplo_inst}(s) \) not theorem
grd7: \( \text{card}(\text{dep_inst}[^{s}]) + \text{card}(\text{run_peers}(s)) < \text{min_{inst}(s)} \)

\( s \mapsto \text{FAIL_CONFIG}_4 \in \text{service}_4 \) not theorem

THEN

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

END

REFINES

REDEPLOY

ANY

\( s \)

WHERE

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

WITH

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

THEN

act1: \( \text{service}_4 = (\text{service}_4 \setminus \{s \mapsto \text{FAIL_CONFIG}_4\}) \cup \{s \mapsto \text{DPL}_4\} \)

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

\[\text{HEAL}: \text{extended ordinary} \]

REFINES

HEAL

ANY

\( s \)

WHERE

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

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

END

\[\text{UNFAIL_PEER}: \text{extended ordinary} \]

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

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(suspe_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(suspe_inst[{ld}]) ⇒ ld = token_owner(s) not theorem →
  inv7: ∀ ld, s · ld ∈ PEERS ∧ s ∈ SERVICES ∧ s ∈ dom(suspe_inst[{ld}]) ∧ ld = token_owner(s) ⇒ ld ∈ susp_inst[{ld}][{s}] not theorem →
  inv8: ∀ ld, s · ld ∈ PEERS ∧ s ∈ SERVICES ∧ s ∈ dom(suspe_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 ↦ 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 →
new values for token owner per service if needed

\[ \text{WHERE} \]

\[ \text{grd1: } \text{prs} \not\in \text{PEERS} \not\text{ theorem} \]
\[ \text{grd2: } \text{prs} \subseteq \text{unav_peers} \not\text{ theorem} \]
\[ \text{grd3: } E \in \text{SERVICES} \mapsto \text{PEERS} \not\text{ theorem} \]

new value for token owner per service if needed

\[ \text{grd4: } \forall \text{srv} \cdot \text{srv} \in \text{SERVICES} \land \text{token_owner(srv)} \notin \text{prs} \Rightarrow E(srv) = \text{token_owner(srv)} \not\text{ theorem} \]

If the token owner of a service srv does not belong to prs, the token owner is not changed

\[ \text{grd5: } \forall \text{srv} \cdot \text{srv} \in \text{SERVICES} \land \text{token_owner(srv)} \in \text{prs} \land \text{srv} \notin \text{dom(susp_peers)} \Rightarrow E(srv) \in \text{run_peers(srv)} \setminus (\text{unav_peers} \cup \text{prs} \cup \text{fail_peers} \{\text{srv}\}) \not\text{ theorem} \]

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

\[ \forall \text{srv} \cdot \text{srv} \in \text{SERVICES} \land \text{token_owner(srv)} \in \text{prs} \land \text{srv} \in \text{dom(susp_peers)} \Rightarrow E(srv) \in \text{run_peers(srv)} \setminus (\text{unav_peers} \cup \text{prs} \cup \text{susp_peers(srv)} \cup \text{fail_peers} \{\text{srv}\}) \not\text{ theorem} \]

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

\[ \forall \text{srv} \cdot \text{srv} \in \text{SERVICES} \land \text{token_owner(srv)} \in \text{prs} \land \text{srv} \in \text{dom(susp_peers)} \Rightarrow E(srv) \in \text{run_peers(srv)} \setminus (\text{unav_peers} \cup \text{prs} \cup \text{susp_peers(srv)} \cup \text{fail_peers} \{\text{srv}\}) \not\text{ theorem} \]

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

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

\[ \text{act2: } \text{token_owner} = \text{token_owner} \hookrightarrow E \Rightarrow \text{new value for token owner per service is given if needed} \]

\[ \text{act3: } \text{susp_inst} = \text{prs} \setminus \text{susp_inst} \Rightarrow \text{the peers in prs can not suspect instances anymore} \]

\[ \text{END} \]

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

\[ \text{ANY} \]

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

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

\[ \text{WHERE} \]

\[ \text{grd1: } s \in \text{SERVICES} \not\text{ theorem} \]
\[ \text{grd2: } \text{susp} \subseteq \text{PEERS} \not\text{ theorem} \]
\[ \text{grd3: } \text{susp} = \text{run_peers(s)} \cap \text{unav_peers} \not\text{ theorem} \]

\[ \text{instances in susp are suspicious if the peers running them becomes unavailable} \]
\[ \text{grd4: } s \in \text{dom(susp_inst}\{\text{token_owner(s)}\}) \not\text{ theorem} \]

\[ \text{the member of susp have not yet been suspected for s by the token owner of s} \]
\[ \text{grd5: } s \hookrightarrow \text{RUN}_4 \in \text{serviceState}_4 \not\text{ theorem} \]

\[ \text{the state of s is OK} \]
THEN
act1: \[\text{ susp inst } = \text{ susp inst } \cup (\{\text{token owner}(s)\} \times \{s\} \times \text{ susp})\]
\(\triangleright\) 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
\hspace{1em} grd1: \(s \in \text{SERVICES}\) not theorem
\hspace{1em} grd2: \(s \mapsto \text{RUN 4} \in \text{serviceState_4}\) not theorem
\hspace{1em} grd3: \(\text{susp inst}[\{\text{token owner}(s)\}][\{s\}] \neq \emptyset\) not theorem

WITH
\hspace{1em} fp: \(fp = \text{susp inst}[\{\text{token owner}(s)\}][\{s\}]\)
THEN
act1: \(\text{serviceState}_4 = (\text{serviceState}_4 - \{s \mapsto \text{RUN 4}\}) \cup \{s \mapsto \text{FAIL}_4\}\)
act2: \(\text{susp peers}(s) = \text{susp inst}[\{\text{token owner}(s)\}][\{s\}]\)
act3: \(\text{susp inst } = \text{susp inst } \triangleright (\{s\} \sim \text{ran(susp inst)}\)
END

FAIL_DETECT: extended ordinary

REFINES
FAIL_DETECT
ANY s sf
WHERE
\hspace{1em} grd1: \(s \in \text{SERVICES}\) not theorem
\hspace{1em} grd2: \(s \mapsto \text{FAIL}_4 \in \text{serviceState}_4\) not theorem
\hspace{1em} grd5: \(\text{susp peers}(s) \neq \emptyset\) not theorem
\hspace{1em} grd6: \(sf \subseteq \text{PEERS}\) not theorem
\hspace{1em} grd7: \(sf \subseteq \text{susp peers}(s)\) not theorem

THEN
act1: \(\text{serviceState}_4 = (\text{serviceState}_4 - \{s \mapsto \text{FAIL}_4\}) \cup \{s \mapsto \text{FAIL}_4\}\)
act2: \(\text{susp peers}(s) = \text{susp peers}(s) \setminus sf\)
END

IS_OK: extended ordinary

REFINES
IS_OK
ANY s
WHERE
\hspace{1em} grd1: \(s \in \text{SERVICES}\) not theorem

FAIL_ACTIV: extended ordinary

\[ \text{REFINES} \]
FAIL_ACTIV
ANY
\[ s \]
WHERE
\[ \text{grd1}: \ s \in \text{SERVICES} \not\text{ theorem} \]
\[ \text{grd2}: \ s \mapsto \text{FAIL_DETECT}_4 \in \text{serviceState}_4 \not\text{ theorem} \]
\[ \text{grd5}: \ \text{susp_peers}(s) = \emptyset \not\text{ theorem} \]
THEN
\[ \text{act1}: \ \text{serviceState}_4 = (\text{serviceState}_4 \setminus \{s \mapsto \text{FAIL_DETECT}_4\}) \cup \{s \mapsto \text{RUN}_4\} \]
END

FAIL_CONFIGURE: extended ordinary

\[ \text{REFINES} \]
FAIL_CONFIGURE
ANY
\[ s \]
WHERE
\[ \text{grd1}: \ s \in \text{SERVICES} \not\text{ theorem} \]
\[ \text{grd2}: \ s \mapsto \text{FAIL_ACTIV}_4 \in \text{serviceState}_4 \not\text{ theorem} \]
\[ \text{grd3}: \ \text{card} (\text{run_peers}(s)) < \text{min_inst}(s) \not\text{ theorem} \]
THEN
\[ \text{act1}: \ \text{serviceState}_4 = (\text{serviceState}_4 \setminus \{s \mapsto \text{FAIL_ACTIV}_4\}) \cup \{s \mapsto \text{FAIL_CONFIG}_4\} \]
END

FAIL_IGNORE: extended ordinary

\[ \text{REFINES} \]
FAIL_IGNORE
ANY
\[ s \]
WHERE
\[ \text{grd1}: \ s \in \text{SERVICES} \not\text{ theorem} \]
\[ \text{grd2}: \ s \mapsto \text{FAIL_ACTIV}_4 \in \text{serviceState}_4 \not\text{ theorem} \]
\[ \text{grd3}: \ \text{card} (\text{run_peers}(s)) \geq \text{min_inst}(s) \not\text{ theorem} \]
THEN
act1:  \[ \text{serviceState}_4 = (\text{serviceState}_4 \setminus \{s \mapsto \text{FAIL}_4\}) \cup \{s \mapsto \text{FAIL}_4\} \]

**IGNORE:** extended ordinary

REFINES

IGNORE

ANY

s

WHERE

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

grd2:  \( s \mapsto \text{FAIL}_4 \in \text{serviceState}_4 \) not theorem

THEN

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

**REDEPLOY INST:** extended ordinary

REFINES

REDEPLOY INST

ANY

s

dep

WHERE

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

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

grd3:  \( \text{finite}(\text{dep}) \) not theorem

grd4:  \( \text{dep} \cap \text{run}_\text{peers}(s) = \emptyset \) not theorem

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

grd6:  \( \text{card}(\text{dep}) = \text{deplo}_\text{inst}(s) \) not theorem

grd7:  \( \text{card}(\text{dep}_\text{inst}[\{s\}]) + \text{card}(\text{run}_\text{peers}(s)) < \text{min}_\text{inst}(s) \) not theorem

grd8:  \( s \mapsto \text{FAIL}_4 \in \text{serviceState}_4 \) not theorem

THEN

act1:  \[ \text{dep}_\text{inst} = \text{dep}_\text{inst} \cup \{s\} \times \text{dep} \]

**REDEPLOY:** extended ordinary

REFINES

REDEPLOY

ANY

s

WHERE

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

grd2:  \( s \mapsto \text{FAIL}_4 \in \text{serviceState}_4 \) not theorem

grd6:  \( \text{dep}_\text{inst}[\{s\}] \neq \emptyset \) not theorem

grd4:  \( \text{card}(\text{run}_\text{peers}(s)) + \text{card}(\text{dep}_\text{inst}[\{s\}]) \geq \text{min}_\text{inst}(s) \) not theorem
THEN
  act1: serviceState_4 := (serviceState_4 \ {s ↦ FAIL_CONFIG_4}) u {s ↦ DPL_4}
  act2: run_peers(s) = run_peers(s) u dep_inst[{s}]
  act3: dep_inst = {s} u 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}) u {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   

  instances that are tried to be recontacted
  instances effectively recontacted after a try

INVIANTS
  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(rec_inst[ld]) ∧
         ld = token_owner(s) ⇒ ld ∈ rec_inst[ld][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 ∈ rct_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
\\rightarrow\text{new values for token owner per service if needed}

WHERE
\begin{align*}
\text{grd1: } & \text{prs} \subseteq \text{PEERS} \not\text{ theorem } \\
\text{grd2: } & \text{prs} \not\subseteq \text{unav_peers} \not\text{ theorem } \\
\text{grd3: } & \text{E} \in \text{SERVICES} \rightarrow \text{PEERS} \not\text{ theorem } \\
\text{grd4: } & \forall \text{srv} \cdot \text{srv} \in \text{SERVICES} \land \text{token_owner(srv)} \notin \text{prs} \Rightarrow \text{E(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}
\end{align*}

\begin{align*}
\text{grd5: } & \forall \text{srv} \cdot \text{srv} \in \text{SERVICES} \land \text{token_owner(srv)} \in \text{prs} \land \text{srv} \notin \text{dom(susp_peers)} \Rightarrow \text{E(srv)} \in \text{run_peers(srv)} \setminus (\text{unav_peers} \cup \text{prs} \cup \text{fail_peers}[[\text{srv}]])) \not\text{ theorem } \rightarrow \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}
\end{align*}

\begin{align*}
\text{grd6: } & \forall \text{srv} \cdot \text{srv} \in \text{SERVICES} \land \text{token_owner(srv)} \in \text{prs} \land \text{srv} \in \text{dom(susp_peers)} \Rightarrow \text{E(srv)} \in \text{run_peers(srv)} \setminus (\text{unav_peers} \cup \text{prs} \cup \text{susp_peers(srv)} \cup \text{fail_peers}[[\text{srv}]])) \not\text{ theorem } \rightarrow \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}
\end{align*}

\text{suspicious peers is chosen}

\text{THEN}
\begin{align*}
\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} \leftarrow \text{E} \rightarrow \text{new value for token owner per service is given if needed}
\text{act3: } & \text{susp_inst} = \text{prs} \leftarrow \text{susp_inst} \rightarrow \text{the peers in prs can not suspect instances anymore}
\text{act4: } & \text{rec_inst} = \text{prs} \leftarrow \text{rec_inst} \rightarrow \\
\text{act5: } & \text{rct_inst} = \text{prs} \leftarrow \text{rct_inst} \\
\end{align*}

\text{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 >

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

THEN
  act1: susp_inst = susp_inst u ({token_owner(s)} x ({s} x 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} c 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

peers for s is not empty

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

rec_inst not theorem >

Page 3
token owner of s has not yet tried to recontact i
grd6: token_owner(s) ↦ (s ↦ i) ∉ rec_inst
then
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: not extended ordinary

RECONTACT_INST_KO:

FAIL_DETECT: not extended ordinary

FAIL_DETECT

FAIL_DETECT

FAIL_DETECT

FAIL_DETECT
act1: \( \text{serviceState}_4 \leftarrow (\text{serviceState}_4 \setminus \{s \mapsto \text{FAIL}_4\}) \cup \{s \mapsto \text{RUN}_4\} \)

act2: \( \text{susp_peers}(s) = \text{susp_peers}(s) \setminus rct\_inst\{\text{token}\_owner(s)\}\{s\} \)

act3: \( \text{rec}\_inst = \text{rec}\_inst \setminus \{s\} \setminus \text{ran}(\text{rec}\_inst) \)

act4: \( rct\_inst = rct\_inst \setminus \{s\} \setminus \text{ran}(rct\_inst) \)

END

**IS_OK**: extended ordinary

REFINES IS_OK

ANY

s

WHERE

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

grd2: \( s \mapsto \text{FAIL}_4 \in \text{serviceState}_4 \) not theorem

grd5: \( \text{susp_peers}(s) = \emptyset \) not theorem

THEN

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

END

FAIL_ACTIV: extended ordinary

REFINES FAIL_ACTIV

ANY

s

WHERE

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

grd2: \( s \mapsto \text{FAIL}_4 \in \text{serviceState}_4 \) not theorem

grd5: \( \text{susp_peers}(s) \neq \emptyset \) not theorem

THEN

act1: \( \text{serviceState}_4 = (\text{serviceState}_4 \setminus \{s \mapsto \text{FAIL}_4\}) \cup \{s \mapsto \text{FAIL}\_ACTIV_4\} \)

act2: \( \text{run}\_peers(s) = \text{run}\_peers(s) \setminus \text{susp}\_peers(s) \)

act3: \( \text{susp}\_peers(s) = \emptyset \)

act4: \( \text{fail}\_peers = \text{fail}\_peers \cup \{s\} \times \text{susp}\_peers(s) \)

END

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: \[ serviceState_4 \equiv (serviceState_4 \setminus \{s \mapsto \text{FAIL_ACTIV}_4\}) \cup \{s \mapsto \text{FAIL_CONFIG}_4\} \]
END

FAIL_IGNORE: extended ordinary \>
REFINES
FAIL_IGNORE
ANY \( s \)
WHERE
grd1: \( s \in \text{SERVICES} \) \text{not theorem}
grd2: \( s \mapsto \text{FAIL_ACTIV}_4 \in \text{serviceState}_4 \) \text{not theorem}
grd3: \( \text{card}(\text{run_peers}(s)) \geq \text{min_inst}(s) \) \text{not theorem}
THEN
act1: \[ serviceState_4 \equiv (serviceState_4 \setminus \{s \mapsto \text{FAIL_ACTIV}_4\}) \cup \{s \mapsto \text{FAIL_IGN}_4\} \]
END

IGNORE: extended ordinary \>
REFINES
IGNORE
ANY \( s \)
WHERE
grd1: \( s \in \text{SERVICES} \) \text{not theorem}
grd2: \( s \mapsto \text{FAIL_IGN}_4 \in \text{serviceState}_4 \) \text{not theorem}
THEN
act1: \[ serviceState_4 \equiv (serviceState_4 \setminus \{s \mapsto \text{FAIL_IGN}_4\}) \cup \{s \mapsto \text{RUN}_4\} \]
END

REDEPLOY_INST: extended ordinary \>
REFINES
REDEPLOY_INST
ANY \( s \) \( \text{dep} \)
WHERE
grd1: \( s \in \text{SERVICES} \) \text{not theorem}
grd2: \( \text{dep} \subseteq \text{PEERS} \) \text{not theorem}
grd3: \( \text{finite}(\text{dep}) \) \text{not theorem}
grd4: \( \text{dep} \cap \text{run_peers}(s) = \emptyset \) \text{not theorem}
grd5: \( \text{dep} \cap \text{fail_peers}([s]) = \emptyset \) \text{not theorem}
grd6: \( \text{card}(\text{dep}) = \text{deplo}\_\text{inst}(s) \) \text{not theorem}
grd7: \( \text{card}(\text{dep}\_\text{inst}([s])) + \text{card}(\text{run_peers}(s)) < \text{min}\_\text{inst}(s) \) \text{not theorem}
grd8: \( s \mapsto \text{FAIL_CONFIG}_4 \in \text{serviceState}_4 \) \text{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} \text{ not theorem}
grd2: s \rightarrow \text{FAIL\_CONFIG\_4} \in \text{serviceState\_4} \text{ not theorem}
grd6: \text{dep}\_\text{inst}\[\{s\}\] \neq \emptyset \text{ not theorem}
grd4: \text{card}(\text{run}\_\text{peers}(s))+\text{card}(\text{dep}\_\text{inst}\[\{s\}\]) \geq \text{min}\_\text{inst}(s) \text{ not theorem}
THEN
act1: \[
\text{serviceState\_4} = (\text{serviceState\_4} \setminus \{s \rightarrow \text{FAIL\_CONFIG\_4}\}) \\
\cup \{s \rightarrow \text{DPL\_4}\}
\]
act2: \[
\text{run}\_\text{peers}(s) = \text{run}\_\text{peers}(s) \cup \text{dep}\_\text{inst}\[\{s\}\]
\]
act3: \[
\text{dep}\_\text{inst} = \{s\} \leftarrow \text{dep}\_\text{inst}
\]
END

HEAL: extended ordinary
REFINES HEAL
ANY
s
WHERE
grd1: s \in \text{SERVICES} \text{ not theorem}
grd2: s \rightarrow \text{DPL\_4} \in \text{serviceState\_4} \text{ not theorem}
THEN
act1: \[
\text{serviceState\_4} = (\text{serviceState\_4} \setminus \{s \rightarrow \text{DPL\_4}\}) \cup \{s \rightarrow \text{RUN\_4}\}
\]
END

UNFAIL\_PEER: extended ordinary
REFINES UNFAIL\_PEER
ANY
s
p
WHERE
grd1: s \in \text{SERVICES} \text{ not theorem}
grd2: p \in \text{PEERS} \text{ not theorem}
grd3: s \rightarrow p \in \text{fail}\_\text{peers} \text{ not theorem}
THEN
act1: \[
\text{fail}\_\text{peers} = \text{fail}\_\text{peers}\setminus\{s \rightarrow p\}
\]
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
M10

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 owns

INVARINTS
  inv1: actv_inst ∈ PEERS ↔ (SERVICES×PEERS) not theorem
  inv2: ∀ s, i · s ∈ SERVICES ∧ i ∈ PEERS ⇒ finite(actv_inst[{i}]{{s}})
  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 = ∅
  act9: rec_inst = ∅
  act10: rct_inst = ∅
  act11: actv_inst = ∅
MAKE_PEER_UNAVAIL: extended ordinary

REFINES

MAKE_PEER_UNAVAIL

ANY

prs →

E \rightarrow new values for token owner per service if needed

WHERE

grd1: prs \subseteq PEERS not theorem →

grd2: prs \not\subseteq unav_peers not theorem →

grd3: E \in SERVICES \rightarrow PEERS not theorem \rightarrow new value for token owner per service if needed

grd4: \forall srv \cdot srv \in SERVICES \land token_owner(srv) \not\in prs \Rightarrow E(srv) = token_owner(srv) not theorem \rightarrow If the token owner of a service srv does not belong to prs, the token owner is not changed

grd5: \forall srv \cdot srv \in SERVICES \land token_owner(srv) \in prs \land srv \not\in dom(susp_peers) \Rightarrow E(srv) \in run_peers(srv) \setminus (unav_peers \cup prs \cup fail_peers \setminus \{srv\}) not theorem \rightarrow 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: \forall srv \cdot srv \in SERVICES \land token_owner(srv) \in prs \land srv \in dom(susp_peers) \Rightarrow E(srv) \in run_peers(srv) \setminus (unav_peers \cup prs \cup susp_peers(srv) \cup fail_peers\{srv\}) not theorem \rightarrow 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 \cup prs \rightarrow the peers in prs become unavailable

act2: token_owner = token_owner \land E \rightarrow new value for token owner per service is given if needed

act3: susp_inst = prs \leftarrow susp_inst \rightarrow the peers in prs can not suspect instances anymore

act4: rec_inst = prs \leftarrow rec_inst

act5: rct_inst = prs \leftarrow rct_inst

act6: actv_inst = prs \leftarrow actv_inst

END

SUSPECT_INST: extended ordinary

REFINES

SUSPECT_INST
M10

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
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: 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}) ∪ {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

grd6: \( \text{token_owner(s)} \mapsto (s \mapsto i) \not\in \text{rec_inst} \) not theorem \( \Rightarrow \) the token owner of \( s \) has not yet tried to recontact \( i \)

grd7: \( \text{rec_inst}[\{\text{token_owner(s)}\}][\{s\}] \subseteq \text{susp_peers(s)} \) not theorem \( \Rightarrow \) the token owner of \( s \) has not yet tried to recontact all the suspecious instances of \( s \)

THEN

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

act2: \( \text{rct_inst} = \text{rct_inst} \cup \{\text{token_owner(s)} \mapsto (s \mapsto i)\} \) \( \Rightarrow \) \( i \) is recontacted by the token owner of \( s \) successfully

END

RECONTACT_INST_KO: extended ordinary \( \Rightarrow \)

REFINES

RECONTACT_INST_KO

ANY

s \( \mapsto \) a service \( s \)

i \( \mapsto \) an instance \( i \)

WHERE

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

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

grd3: \( s \mapsto \text{FAIL}_4 \in \text{serviceState}_4 \) not theorem \( \Rightarrow \) the state of \( s \) is SUSPICIOUS

grd4: \( \text{susp_peers(s)} \neq \emptyset \) not theorem \( \Rightarrow \) the set of suspicuous peers for \( s \) is not empty

grd5: \( i \in \text{susp_peers(s)\cup unav_peers} \) not theorem \( \Rightarrow \) \( i \) is a suspicuous instance of \( s \) and is unavialable (can not be contacted)

grd6: \( \text{token_owner(s)} \mapsto (s \mapsto i) \not\in \text{rec_inst} \) not theorem \( \Rightarrow \) the token owner of \( s \) has not yet tried to recontact \( i \)

grd7: \( \text{rec_inst}[\{\text{token_owner(s)}\}][\{s\}] \subseteq \text{susp_peers(s)} \) not theorem \( \Rightarrow \) the token owner of \( s \) has not yet tried to recontact all the suspicuous instances of \( s \)

THEN

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

END

FAIL_DETECT: extended ordinary \( \Rightarrow \)

REFINES

FAIL_DETECT

ANY

s \( \mapsto \)

WHERE

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

grd2: \( s \mapsto \text{FAIL}_4 \in \text{serviceState}_4 \) not theorem \( \Rightarrow \)

grd5: \( \text{susp_peers(s)} \neq \emptyset \) not theorem \( \Rightarrow \)

grd8: \( \text{rec_inst}[\{\text{token_owner(s)}\}][\{s\}] = \text{susp_peers(s)} \) not theorem \( \Rightarrow \)
THEN
  act1: serviceState_4 ≔ (serviceState_4\{s ↦ FAIL_4\}) u \{s ↦
  FAIL_DETECT_4\} )
  act2: susp_peers(s) = susp_peers(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: 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_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 u (\{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

**M10**

\[ \text{grd3: card(run_peers(s)) < min_inst(s) not theorem } \]
\[ \text{THEN} \]
\[ \text{act1: serviceState}_4 \equiv (\text{serviceState}_4 \setminus \{s \mapsto \text{FAIL_ACTIV}_4\}) \cup \{s \mapsto \text{FAIL_CONFIG}_4\} \]
\[ \text{END} \]

**FAIL_IGNORE:** extended ordinary

REFINES

FAIL_IGNORE

ANY

\[ s \to \]
\[ \text{WHERE} \]
\[ \text{grd1: } s \in \text{SERVICES not theorem } \]
\[ \text{grd2: } s \mapsto \text{FAIL_ACTIV}_4 \in \text{serviceState}_4 \text{ not theorem } \]
\[ \text{grd3: card(run_peers(s)) > min_inst(s) not theorem } \]
\[ \text{THEN} \]
\[ \text{act1: serviceState}_4 \equiv (\text{serviceState}_4 \setminus \{s \mapsto \text{FAIL_ACTIV}_4\}) \cup \{s \mapsto \text{FAIL IGN}_4\} \]
\[ \text{END} \]

**IGNORE:** extended ordinary

REFINES

IGNORE

ANY

\[ s \to \]
\[ \text{WHERE} \]
\[ \text{grd1: } s \in \text{SERVICES not theorem } \]
\[ \text{grd2: } s \mapsto \text{FAIL_IGN}_4 \in \text{serviceState}_4 \text{ not theorem } \]
\[ \text{THEN} \]
\[ \text{act1: serviceState}_4 \equiv (\text{serviceState}_4 \setminus \{s \mapsto \text{FAIL_IGN}_4\}) \cup \{s \mapsto \text{RUN}_4\} \]
\[ \text{END} \]

**REDEPLOY_INSTC:** not extended ordinary

ANY

\[ s \to \text{a service s} \]
\[ i \to \text{an instance i} \]
\[ \text{WHERE} \]
\[ \text{grd1: } s \in \text{SERVICES not theorem } \]
\[ \text{grd2: } i \in \text{PEERS not theorem } \]
\[ \text{grd3: } i \notin \text{run_peers}(s) \cup \text{fail_peers}\{s\} \cup \text{unav_peers} \cup \text{dep inst}\{\{i\} \} \text{ not theorem } \]
\[ \text{dep inst}\{\{s\}\} \text{ not theorem } \]
\[ \text{i does not run } s, \text{ is not failed for } s, \text{ is not unavailable and is not already activated for } s \]
\[ \text{grd4: token_owner}(s) \mapsto (s \mapsto i) \notin \text{actv inst} \text{ not theorem } \]
\[ \text{grd5: } s \mapsto \text{FAIL_CONFIG}_4 \in \text{serviceState}_4 \text{ not theorem } \]
\[ \text{grd6: card(actv inst}\{\{\text{token_owner}(s)\}\}\{\{s\}\}) < \text{deplo inst}(s) \]
\[ \text{not theorem } \]
\[ \text{grd7: card(dep inst}\{\{s\}\} \} + \text{card(run_peers}(s)) < \text{min_inst}(s) \]
not theorem
THEN
act1: \( \text{actv\_inst} = \text{actv\_inst} \cup \{\text{token\_owner}(s) \rightarrow (s \rightarrow i)\} \)
END

REDEPLOY\_INSTS: not extended ordinary
\[\text{REFINES} \]
REDEPLOY\_INST
\[\text{ANY} \]
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: \( s \rightarrow \text{FAIL\_CONFIG\_4} \in \text{serviceState\_4} \) 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: \( \text{actv\_inst} = \text{actv\_inst} \triangleright (\{s\} \setminus \text{ran}(\text{actv\_inst})) \)
END

REDEPLOY: not extended ordinary
\[\text{REFINES} \]
REDEPLOY
\[\text{ANY} \]
s
WHERE
grd1: \( s \in \text{SERVICES} \) not theorem
grd2: \( s \rightarrow \text{FAIL\_CONFIG\_4} \in \text{serviceState\_4} \) 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: \( \text{serviceState\_4} = (\text{serviceState\_4} \setminus \{s \rightarrow \text{FAIL\_CONFIG\_4}\}) \cup \{s \rightarrow \text{DPL\_4}\} \)
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
\[\text{REFINES} \]
HEAL
\[\text{ANY} \]

Page 7
s \rightarrow
WHERE
  grd1: s \in SERVICES not theorem
  grd2: s \rightarrow DPL_4 \in serviceState_4 not theorem
THEN
  act1: serviceState_4 = (serviceState_4 \setminus \{s \rightarrow DPL_4\}) \cup \{s \rightarrow RUN_4\}
END

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

MAKE_PEER_AVAIL: extended ordinary
  REFINES
  MAKE_PEER_AVAIL
  ANY
  p \rightarrow
WHERE
  grd1: p \in PEERS not theorem
  grd2: p \in unav_peers not theorem
THEN
  act1: unav_peers = unav_peers \setminus \{p\}
END
END
M11

MACHINE
M11 →
REFINES
M10
SEES
C08
VARIABLES
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 owns
i_state →
i_state ∈ (PEERS × SERVICES) ⇸ STATES_4 not theorem →
INVARIANTS
inv1:  i_state ∈ (PEERS × SERVICES) ⇸ STATES_4 not theorem →
inv2:  ∀ s · s ∈ SERVICES ⇒ token_owner(s) ⇝ s ∈ dom(i_state) not theorem →
theorem →
gluing_state1: ∀ s, stt · s ∈ SERVICES ∧ stt ∈ STATES_4 ∧ s ↦ stt ∈ serviceState_4 ⇒ (token_owner(s) ↦ s) ↦ stt ∈ i_state not theorem →
gluing_state2: ∀ s, stt · s ∈ SERVICES ∧ stt ∈ STATES_4 ∧ (token_owner(s) ↦ s) ↦ stt ∈ i_state ⇒ s ↦ stt ∈ serviceState_4 not theorem →
inj3:  ∀ 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 = Ø →
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 \triangleright
E \triangleright\text{new values for token owner per service if needed}
i_s \triangleright
\text{WHERE}
\begin{align*}
grd1: & \quad \text{prs} \in \text{PEERS} \not\text{ theorem} \\
grd2: & \quad \text{prs} \not\in \text{unav_peers} \not\text{ theorem} \\
grd3: & \quad \text{E} \in \text{SERVICES} \rightarrow \text{PEERS} \not\text{ theorem} \rightarrow \text{new value for token} \\text{owner per service if needed} \\
grd4: & \quad i_s \in (\text{PEERS} \times \text{SERVICES}) \leftrightarrow \text{STATES}_4 \not\text{ theorem} \\
grd5: & \quad \forall \text{srv} \cdot \text{srv} \in \text{SERVICES} \land \text{token_owner(srv)} \in \text{prs} \Rightarrow \text{E} \\
\end{align*}
\text{(srv) = token_owner(srv) not theorem} \rightarrow \text{If the token owner of a service srv does} \\
\text{not belong to prs, the token owner is not changed} \\
grd6: & \quad \forall \text{srv} \cdot \text{srv} \in \text{SERVICES} \land \text{token_owner(srv)} \in \text{prs} \land \text{srv} \not\in \text{dom(susp_peers)} \Rightarrow \text{E(srv) \in run_peers(srv) \setminus (unav_peers \cup \text{prs} \cup \text{fail_peers}} \\
\{\text{srv}\}) \not\text{ theorem} \rightarrow \text{if the owner of the token for a service becomes} \\
\text{unavailable and the service is not suspicious,} \\
\text{then a new token owner among available peers is chosen} \\
grd7: & \quad \forall \text{srv} \cdot \text{srv} \in \text{SERVICES} \land \text{token_owner(srv)} \in \text{prs} \land \text{srv} \not\in \text{dom(susp_peers)} \Rightarrow \text{E(srv) \in run_peers(srv) \setminus (unav_peers \cup \text{prs} \cup \text{fail_peers}} \\
\{\text{srv}\}) \not\text{ theorem} \rightarrow \text{if the owner of the token for a service becomes} \\
\text{unavailable, and the service} \\
\text{possess suspicious instances, then a new token owner among available and not} \\
\text{suspicious peers is chosen} \\
grd8: & \quad \forall p, s \cdot p \in \text{PEERS} \land s \in \text{SERVICES} \land p \equiv s \in \text{dom(i_s)} \Rightarrow \\
p = \text{E}(s) \not\text{ theorem} \\
grd9: & \quad \forall \text{srv} \cdot \text{srv} \in \text{SERVICES} \Rightarrow (\text{E(srv)} \rightarrow \text{srv}) \rightarrow \text{i_state} \\
\text{(token_owner(srv) \rightarrow \text{srv}) \in i_s \not\text{ theorem} \rightarrow} \\
\text{THEN} \\
\text{act1: unav_peers = unav_peers \cup \text{prs} \rightarrow \text{the peers in prs become}} \\
\text{unavailable} \\
\text{act2: token_owner = token_owner \leftarrow \text{E} \rightarrow \text{new value for token owner} per service is given if needed} \\
\text{act3: susp_inst = prs \leftarrow susp_inst \rightarrow \text{the peers in prs can not} suspect instances anymore} \\
\text{act4: rec_inst = prs \leftarrow rec_inst \rightarrow \text{the peers in prs can not try} to recontact instances anymore} \\
\text{act5: rct_inst = prs \leftarrow rct_inst \rightarrow \text{the peers in prs can not recontact instances anymore}} \\
\text{act6: actv_inst = prs \leftarrow actv_inst \rightarrow} \\
\text{act7: i_state = i_s} \\
\text{END}
M11

**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

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

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: 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

THEN
  act4: susp_peers(s) ≠ ∅ not theorem

the set of suspicious peers for s is not empty

END
```
M11

grd5: \( i \in \text{suspe}_\text{r}_\text{peers}(s) \setminus \text{unav}_\text{peers} \) not theorem \( \Rightarrow \) \( i \) is a

suspicous instance of \( s \) and is available (can be contacted)

grd6: \( \text{token}_\text{owner}(s) \not\to (s \not\to i) \notin \text{rec}_\text{inst} \) not theorem \( \Rightarrow \) the
token owner of \( s \) has not yet tried to recontact \( i \)

grd7: \( \text{rec}_\text{inst}[\{\text{token}_\text{owner}(s)\}][\{s\}] \subset \text{suspe}_\text{r}_\text{peers}(s) \) not

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

THEN

act1: \( \text{rec}_\text{inst} = \text{rec}_\text{inst} \cup \{\text{token}_\text{owner}(s) \not\to (s \not\to i)\} \) \( \Rightarrow \) the
token owner of \( s \) has tried to recontact \( i \)

act2: \( \text{rct}_\text{inst} = \text{rct}_\text{inst} \cup \{\text{token}_\text{owner}(s) \not\to (s \not\to i)\} \) \( \Rightarrow \) \( i \) 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 \( \Rightarrow \)
grd2: \( i \in \text{PEERS} \) not theorem \( \Rightarrow \)
grd3: \( \text{i}_\text{state}(\text{token}_\text{owner}(s) \not\to s) = \text{FAIL}_\text{4} \) not theorem \( \Rightarrow \)

the state of \( s \) is SUSPICIOUS

grd4: \( \text{suspe}_\text{r}_\text{peers}(s) \neq \emptyset \) not theorem \( \Rightarrow \)

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

grd5: \( i \in \text{suspe}_\text{r}_\text{peers}(s) \setminus \text{unav}_\text{peers} \) not theorem \( \Rightarrow \)

i is a

suspicous instance of \( s \) and is unavailable (can not be contacted)

grd6: \( \text{token}_\text{owner}(s) \not\to (s \not\to i) \notin \text{rec}_\text{inst} \) not theorem \( \Rightarrow \)

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

grd7: \( \text{rec}_\text{inst}[\{\text{token}_\text{owner}(s)\}][\{s\}] \subset \text{suspe}_\text{r}_\text{peers}(s) \) not

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

THEN

act1: \( \text{rec}_\text{inst} = \text{rec}_\text{inst} \cup \{\text{token}_\text{owner}(s) \not\to (s \not\to i)\} \) \( \Rightarrow \) the
token owner of \( s \) has tried to recontact \( i \)

END

FAIL_DETECT: not extended ordinary

REFINES

FAIL_DETECT

ANY

s \( \rightarrow \)

WHERE

grd1: \( s \in \text{SERVICES} \) not theorem \( \Rightarrow \)
grd2: \( \text{i}_\text{state}(\text{token}_\text{owner}(s) \not\to s) = \text{FAIL}_\text{4} \) not theorem \( \Rightarrow \)
grd5: \( \text{suspe}_\text{r}_\text{peers}(s) \neq \emptyset \) not theorem \( \Rightarrow \)
M11

\[ \text{grd8: } \text{rec\_inst[\{token\_owner(s)}\[\{s\}\] = susp\_peers(s)} \not\text{ theorem} \]

THEN

act1: \( i\_state(\text{token\_owner(s)} \gg s) = \text{FAIL\_DETECT\_4} \)
act2: susp\_peers(s) = susp\_peers(s) \setminus \text{rct\_inst[\{token\_owner(s)}\[\{s\}\]}

act3: \( \text{rec\_inst} = \text{rec\_inst} \cup \{\{\} \times \text{ran(rec\_inst)}\) \)
act4: \( \text{rct\_inst} = \text{rct\_inst} \cup \{\{\} \times \text{ran(rct\_inst)}\) \)

END

\[ \text{IS\_OK: } \not\text{ extended ordinary} \]

REFINES IS\_OK

ANY \( s \)

WHERE

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

grd5: susp\_peers(s) = \( \emptyset \not\text{ theorem} \)
THEN

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

END

\[ \text{FAIL\_ACTIV: } \not\text{ extended ordinary} \]

REFINES FAIL\_ACTIV

ANY \( s \)

WHERE

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

grd5: susp\_peers(s) \neq \( \emptyset \not\text{ theorem} \)
THEN

act1: \( i\_state(\text{token\_owner(s)} \gg s) = \text{FAIL\_ACTIV\_4} \)
act2: run\_peers(s) = run\_peers(s) \setminus susp\_peers(s)
act3: susp\_peers(s) = \( \emptyset \)
act4: fail\_peers = fail\_peers \cup \{\{s\}\times susp\_peers(s)} \)

END

\[ \text{FAIL\_CONFIGURE: } \not\text{ extended ordinary} \]

REFINES FAIL\_CONFIGURE

ANY \( s \)

WHERE

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

Page 5
M11

\[ \text{grd2: } i\_state(token\_owner(s) \Rightarrow s) = \text{FAIL\_ACTIV\_4 not theorem} \]
\[ \text{grd3: } \text{card}\text{(run\_peers(s))} < \text{min\_inst(s) not theorem} \]
\[ \text{THEN} \]
\[ \text{act1: } i\_state(token\_owner(s) \Rightarrow s) = \text{FAIL\_CONF\_4} \]
\[ \text{END} \]

\[ \text{FAIL\_IGNORE: not extended ordinary} \]
\[ \text{REFINES} \]
\[ \text{FAIL\_IGNORE} \]
\[ \text{ANY} \]
\[ s \]
\[ \text{WHERE} \]
\[ \text{grd1: } s \in \text{SERVICES not theorem} \]
\[ \text{grd2: } i\_state(token\_owner(s) \Rightarrow s) = \text{FAIL\_ACTIV\_4 not theorem} \]
\[ \text{grd3: } \text{card}\text{(run\_peers(s))} \geq \text{min\_inst(s) not theorem} \]
\[ \text{THEN} \]
\[ \text{act1: } i\_state(token\_owner(s) \Rightarrow s) = \text{FAIL\_IGN\_4} \]
\[ \text{END} \]

\[ \text{IGNORE: not extended ordinary} \]
\[ \text{REFINES} \]
\[ \text{IGNORE} \]
\[ \text{ANY} \]
\[ s \]
\[ \text{WHERE} \]
\[ \text{grd1: } s \in \text{SERVICES not theorem} \]
\[ \text{grd2: } i\_state(token\_owner(s) \Rightarrow s) = \text{FAIL\_IGN\_4} \]
\[ \text{THEN} \]
\[ \text{act1: } i\_state(token\_owner(s) \Rightarrow s) = \text{RUN\_4} \]
\[ \text{END} \]

\[ \text{REDEPLOY\_INSTC: not extended ordinary} \]
\[ \text{REFINES} \]
\[ \text{REDEPLOY\_INSTC} \]
\[ \text{ANY} \]
\[ s \rightarrow \text{a service s} \]
\[ i \rightarrow \text{an instance i} \]
\[ \text{WHERE} \]
\[ \text{grd1: } s \in \text{SERVICES not theorem} \]
\[ \text{grd2: } i \in \text{PEERS not theorem} \]
\[ \text{grd3: } i \notin \text{run\_peers(s) v fail\_peers\{s\} v unav\_peers v dep\_inst\{s\}} \text{ not theorem} \]
\[ \text{i does not run s, is not failed for s, is not unavailable and is not already activated for s} \]
\[ \text{grd4: } \text{token\_owner(s) \Rightarrow (s \Rightarrow i) \notin \text{actv\_inst not theorem}} \]
\[ \text{grd5: } i\_state(token\_owner(s) \Rightarrow s) = \text{FAIL\_CONF\_4 not theorem} \]
\[ \text{grd6: } \text{card}\text{(actv\_inst\{token\_owner(s)\}\{s\})} < \text{deplo\_inst(s)} \text{ not theorem}} \]
\[ \text{not theorem} \]

Page 6
GRD7: \( \text{card}(\text{dep}_\text{inst}[\{s\}]) + \text{card}(\text{run}_\text{peers}(s)) < \text{min}_\text{inst}(s) \)  

\text{THEN}  
\hspace{1em} \text{act}_1: \text{act}_\text{inst} = \text{act}_\text{inst} \cup \{\text{token}_\text{owner}(s) \mapsto (s \mapsto i)\}  
\text{END}  

\text{REDEPLOY INSTS:} \quad \text{not extended ordinary}  
\text{REFINES}  
\text{REDEPLOY INSTS}  
\text{ANY}  
\hspace{1em} s  
\text{WHERE}  
\hspace{1em} \text{grd}_1: \ s \in \text{SERVICES} \quad \text{not theorem}  
\hspace{1em} \text{grd}_6: \ \text{card}(\text{act}_\text{inst}[\{\text{token}_\text{owner}(s)\}][\{s\}]) = \text{deplo}_\text{inst}(s) \quad \text{not theorem}  
\hspace{1em} \text{grd}_7: \ \text{card}(\text{dep}_\text{inst}[\{s\}]) + \text{card}(\text{run}_\text{peers}(s)) < \text{min}_\text{inst}(s) \quad \text{not theorem}  
\hspace{1em} \text{grd}_8: \ \text{i}_\text{state}(\text{token}_\text{owner}(s) \mapsto s) = \text{FAIL}_\text{CONFIG}_4 \quad \text{not theorem}  
\hspace{1em} \text{THEN}  
\hspace{2em} \text{act}_1: \ \text{dep}_\text{inst} = \text{dep}_\text{inst} \cup \{\{s\} \times \text{act}_\text{inst}[\{\text{token}_\text{owner}(s)\}][\{s\}]\}  
\hspace{2em} \text{act}_2: \ \text{act}_\text{inst} = \text{act}_\text{inst} \cup \{\{s\} \leftarrow \text{ran}(\text{act}_\text{inst})\}  
\text{END}  

\text{REDEPLOY:} \quad \text{not extended ordinary}  
\text{REFINES}  
\text{REDEPLOY}  
\text{ANY}  
\hspace{1em} s  
\text{WHERE}  
\hspace{1em} \text{grd}_1: \ s \in \text{SERVICES} \quad \text{not theorem}  
\hspace{1em} \text{grd}_2: \ \text{i}_\text{state}(\text{token}_\text{owner}(s) \mapsto s) = \text{FAIL}_\text{CONFIG}_4 \quad \text{not theorem}  
\hspace{1em} \text{grd}_7: \ \text{act}_\text{inst}[\{\text{token}_\text{owner}(s)\}][\{s\}]=\emptyset \quad \text{not theorem}  
\hspace{1em} \text{grd}_6: \ \text{dep}_\text{inst}[\{s\}] \neq \emptyset \quad \text{not theorem}  
\hspace{1em} \text{grd}_4: \ \text{card}(\text{run}_\text{peers}(s)) + \text{card}(\text{dep}_\text{inst}[\{s\}]) \geq \text{min}_\text{inst}(s) \quad \text{not theorem}  
\hspace{1em} \text{THEN}  
\hspace{2em} \text{act}_1: \ \text{i}_\text{state}(\text{token}_\text{owner}(s) \mapsto s) = \text{DPL}_4  
\hspace{2em} \text{act}_2: \ \text{run}_\text{peers}(s) = \text{run}_\text{peers}(s) \cup \text{dep}_\text{inst}[\{s\}]  
\hspace{2em} \text{act}_3: \ \text{dep}_\text{inst} = \{s\} \leftarrow \text{dep}_\text{inst}  
\text{END}  

\text{HEAL:} \quad \text{not extended ordinary}  
\text{REFINES}  
\text{HEAL}  
\text{ANY}
M11

\[
\begin{align*}
\text{s} & > \\
\text{WHERE} & \\
\text{grd1: } & s \in \text{SERVICES not theorem} > \\
\text{grd2: } & \text{i_state(token_owner(s) }\rightarrow\text{ s) }= \text{DPL}_4 \text{ not theorem} > \\
\text{THEN} & \\
\text{act1: } & \text{i_state(token_owner(s) }\rightarrow\text{ s) }= \text{RUN}_4 > \\
\text{END} & \\
\text{UNFAIL\_PEER: extended ordinary}\ > \\
\text{REFINES} & \\
\text{UNFAIL\_PEER} & \\
\text{ANY} & \\
\text{s} & > \\
\text{p} & > \\
\text{WHERE} & \\
\text{grd1: } & s \in \text{SERVICES not theorem} > \\
\text{grd2: } & p \in \text{PEERS not theorem} > \\
\text{grd3: } & s \rightarrow p \in \text{fail\_peers not theorem} > \\
\text{THEN} & \\
\text{act1: } & \text{fail\_peers }= \text{fail\_peers}\backslash\{s \rightarrow p\} > \\
\text{END} & \\
\text{MAKE\_PEER\_AVAIL: extended ordinary}\ > \\
\text{REFINES} & \\
\text{MAKE\_PEER\_AVAIL} & \\
\text{ANY} & \\
\text{p} & > \\
\text{WHERE} & \\
\text{grd1: } & p \in \text{PEERS not theorem} > \\
\text{grd2: } & p \in \text{unav\_peers not theorem} > \\
\text{THEN} & \\
\text{act1: } & \text{unav\_peers }= \text{unav\_peers} \backslash \{p\} > \\
\text{END} & \\
\text{END} & \\
\end{align*}
\]
M12

MACHINE M12
REFINES M11
SEES C08

VARIABLES
  run_peers
  suspc_peers
  fail_peers
  dep_inst
  token_owner
  unav_peers
  susp_inst
  rec_inst
  rct_inst
  actv_inst
  instances that are tried to be recontacted
  instances effectively recontacted after a try
  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 ≔ ∅
    act5: dep_inst ≔ ∅
    act6: token_owner ≔ init_tok
    act7: unav_peers ≔ ∅
    act8: susp_inst ≔ ∅
    act9: rec_inst ≔ ∅
    act10: rct_inst ≔ ∅
    act11: actv_inst ≔ ∅
    act12: 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 > 
WHERE

grd1: \( \text{prs} \subseteq \text{PEERS} \) not theorem >
grd2: \( \text{prs} \not\subseteq \text{unav}_\text{peers} \) not theorem >
grd3: \( E \in \text{SERVICES} \rightarrow \text{PEERS} \) not theorem >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: \( \forall \text{srv} \cdot \text{srv} \in \text{SERVICES} \land \text{token}_\text{owner} (\text{srv}) \in \text{prs} \Rightarrow E (\text{srv}) = \text{token}_\text{owner} (\text{srv}) \) not theorem >If the token owner of a service srv does not belong to prs, the token owner is not changed

grd6: \( \forall \text{srv} \cdot \text{srv} \in \text{SERVICES} \land \text{token}_\text{owner} (\text{srv}) \in \text{prs} \land \text{token}_\text{owner} (\text{srv}) \Rightarrow \text{srv} \not\in \text{dom} (\text{suspc}_\text{peers}) \Rightarrow E (\text{srv}) \in \text{run}_\text{peers} (\text{srv}) \setminus (\text{unav}_\text{peers} \cup \text{prs} \cup \text{fail}_\text{peers} [\{\text{srv}\}] ) \) 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

grd7: \( \forall \text{srv} \cdot \text{srv} \in \text{SERVICES} \land \text{token}_\text{owner} (\text{srv}) \in \text{prs} \land \text{token}_\text{owner} (\text{srv}) \Rightarrow \text{srv} \in \text{dom} (\text{suspc}_\text{peers}) \Rightarrow E (\text{srv}) \in \text{run}_\text{peers} (\text{srv}) \setminus (\text{unav}_\text{peers} \cup \text{prs} \cup \text{fail}_\text{peers} [\{\text{srv}\}] \cup \text{suspc}_\text{peers} (\text{token}_\text{owner} (\text{srv}) \Rightarrow \text{srv})) \) 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: \( \forall p, s \cdot p \in \text{PEERS} \land s \in \text{SERVICES} \land p \ni s \in \text{dom} (i_s) \Rightarrow p = E(s) \) not theorem >
grd9: \( \forall \text{srv} \cdot \text{srv} \in \text{SERVICES} \Rightarrow (E (\text{srv}) \rightarrow \text{srv}) \rightarrow i_s \) not theorem >

(token_owner (srv) \Rightarrow srv) \in i_s not theorem >
grd10: \( \forall p, s \cdot p \in \text{PEERS} \land s \in \text{SERVICES} \land p \ni s \in \text{dom} (p_s) \Rightarrow p = E(s) \) not theorem >
grd11: \( \forall p, s \cdot p \in \text{PEERS} \land s \in \text{SERVICES} \land p \ni s \in \text{dom} (p_s) \Rightarrow (\text{token}_\text{owner} (\text{srv}) \Rightarrow \text{srv}) \in p_s \) not theorem >

THEN

act1: \( \text{unav}_\text{peers} \leftarrow \text{unav}_\text{peers} \cup \text{prs} \rightarrow \text{the peers in prs become unavailable} \)

act2: \( \text{token}_\text{owner} := \text{token}_\text{owner} \leftarrow E \rightarrow \text{new value for token owner per service if needed} \)

act3: \( \text{susp}_\text{inst} := \text{prs} \leftarrow \text{susp}_\text{inst} \rightarrow \text{the peers in prs can not suspect instances anymore} \)

act4: \( \text{rec}_\text{inst} := \text{prs} \leftarrow \text{rec}_\text{inst} \rightarrow \text{the peers in prs can not try to recontact instances anymore} \)

act5: \( \text{rct}_\text{inst} := \text{prs} \leftarrow \text{rct}_\text{inst} \rightarrow \text{the peers in prs can not recontact instances anymore} \)
act6: \( \text{actv\_inst} \leftarrow \text{prs} \leftarrow \text{actv\_inst} \)  
act7: \( \text{i\_state} \leftarrow \text{i\_s} \)  
act8: \( \text{susp\_peers} \leftarrow \text{p\_s} \) 

END

\text{SUSPECT\_INST: extended ordinary}  
\text{REFINES}  
\text{SUSPECT\_INST}  
\text{ANY}  
\text{s} \rightarrow \text{a service s}  
\text{susp} \rightarrow \text{suspicious instances}  
\text{WHERE}  
\text{grd1: } \text{s} \in \text{SERVICES} \text{ not theorem}  
\text{grd2: } \text{susp} \in \text{PEERS} \text{ not theorem}  
\text{grd3: } \text{susp} = \text{run\_peers(s)} \cap \text{unav\_peers} \text{ not theorem}  
\text{instances in susp are suspicious if the peers running them becomes unavailable}  
\text{grd4: } \text{s} \notin \text{dom(susp\_inst[{token\_owner(s)}])} \text{ not 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) ↦ s) = RUN\_4} \text{ not theorem}  
\text{the state of s is OK}  
\text{THEN}  
\text{act1: } \text{susp\_inst} = \text{susp\_inst} \cup \{\{\text{token\_owner(s)}\} \times \{\{\text{s}\}\times\text{susp}\} \}  
\text{the members of susp become suspected instances for s by the token owner of s}  
\text{END}

\text{FAIL: not extended ordinary}  
\text{REFINES}  
\text{FAIL}  
\text{ANY}  
\text{s} \rightarrow \text{a service s}  
\text{WHERE}  
\text{grd1: } \text{s} \in \text{SERVICES} \text{ not theorem}  
\text{grd2: } \text{i\_state(token\_owner(s) ↦ s) = RUN\_4} \text{ not theorem}  
\text{grd3: } \text{susp\_inst[\{token\_owner(s)\}][\{s\}] \neq \emptyset} \text{ not theorem}  
\text{THEN}  
\text{act1: } \text{i\_state(token\_owner(s) ↦ s) = FAIL\_4}  
\text{act2: } \text{susp\_peers(token\_owner(s) ↦ s) = susp\_inst[\{token\_owner(s)\}]}  
\text{act3: } \text{susp\_inst} = \text{susp\_inst} \leftarrow \{\{s\} \leftarrow \text{ran(susp\_inst)}\}  
\text{END}

\text{RECONTACT\_INST\_OK: not extended ordinary}  
\text{REFINES}  
\text{RECONTACT\_INST\_OK}  
\text{ANY}  
\text{s} \rightarrow \text{a service s}  
\text{i} \rightarrow \text{an instance i}  
\text{WHERE}
M12

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

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

grd3: \( i_{\text{state}}(\text{token\_owner}(s) \mapsto s) = \text{FAIL\_4} \) not theorem

the state of \( s \) is SUSPICIOUS

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

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

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

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

grd6: \( \text{token\_owner}(s) \mapsto (s \mapsto i) \notin \text{rec\_inst} \) not theorem

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

grd7: \( \text{rec\_inst}[^{\text{token\_owner}(s)}][^s] \subset \text{suspc\_peers}[^{\text{token\_owner}(s)}] \) not theorem

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

THEN

act1: \( \text{rec\_inst} = \text{rec\_inst} \cup \{\text{token\_owner}(s) \mapsto (s \mapsto i)\} \) not theorem

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

act2: \( \text{rct\_inst} = \text{rct\_inst} \cup \{\text{token\_owner}(s) \mapsto (s \mapsto i)\} \) not theorem

\( i \) 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: \( i_{\text{state}}(\text{token\_owner}(s) \mapsto s) = \text{FAIL\_4} \) not theorem

the state of \( s \) is SUSPICIOUS

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

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

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

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

grd6: \( \text{token\_owner}(s) \mapsto (s \mapsto i) \notin \text{rec\_inst} \) not theorem

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

grd7: \( \text{rec\_inst}[^{\text{token\_owner}(s)}][^s] \subset \text{suspc\_peers}[^{\text{token\_owner}(s)}] \) not theorem

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

THEN

act1: \( \text{rec\_inst} = \text{rec\_inst} \cup \{\text{token\_owner}(s) \mapsto (s \mapsto i)\} \) not theorem

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

END

FAIL\_DETECT: not 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

THEN
act1: i_state(token_owner(s) ↦ s) ≔ FAIL_DETECT_4
act2: suspc_peers(token_owner(s) ↦ s) = suspc_peers

act3: rec_inst = rec_inst \ ({s} × suspc_peers(token_owner(s) ↦ s))
act4: suspc_peers(token_owner(s) ↦ s) ≔ ∅

END

IS_OK: not 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: not 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

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: fail_peers = fail_peers ∪ ({s} × suspc_peers(token_owner(s) ↦ s))
act4: suspc_peers(token_owner(s) ↦ s) = ∅
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  

i  

WHERE  

grd1:  s ∈ SERVICES not theorem  

Page 6
grd2:  \( i \in \text{PEERS} \) not theorem
grd3:  \( i \not\in \text{run_peers}(s) \cup \text{fail_peers}(s) \cup \text{unav_peers} \cup \text{dep_inst}(s) \) not theorem \( \Rightarrow \) \( i \) does not run \( s \), is not failed for \( s \), is not unavailable and is not already activated for \( s \)

grd4:  \( \text{token_owner}(s) \Rightarrow (s \Rightarrow i) \not\in \text{actv_inst} \) not theorem

\( \text{grad5: } \text{i_state}(\text{token_owner}(s) \Rightarrow s) = \text{FAIL_CONFIG}_4 \) not theorem

\( \text{grad6: } \text{card}(\text{actv_inst}(\{\text{token_owner}(s)\})[s]) < \text{deplo_inst}(s) \) not theorem

\( \text{grad7: } \text{card}(\text{dep_inst}(s)) + \text{card}(\text{run_peers}(s)) < \text{min_inst}(s) \) not theorem

THEN

act1:  \text{actv_inst} = \text{actv_inst} \cup \{\text{token_owner}(s) \Rightarrow (s \Rightarrow i)\}

END

\text{REDEPLOY_INSTS: } \text{extended ordinary} \Rightarrow

\text{REFINES}

\text{REDEPLOY_INSTS}

\text{ANY}

\text{s} >

\text{WHERE}

\text{grad1: } \text{s} \in \text{SERVICES} \not\text{theorem} >

\text{grad2: } \text{i_state}(\text{token_owner}(s) \Rightarrow s) = \text{FAIL_CONFIG}_4 \not\text{theorem} >

\text{grad6: } \text{card}(\text{actv_inst}(\{\text{token_owner}(s)\})[s]) = \text{deplo_inst}(s) \not\text{theorem} >

\text{grad7: } \text{card}(\text{dep_inst}(s)) + \text{card}(\text{run_peers}(s)) < \text{min_inst}(s) \not\text{theorem} >

\text{grad8: } \text{i_state}(\text{token_owner}(s) \Rightarrow s) = \text{FAIL_CONFIG}_4 \not\text{theorem} >

\text{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} \Rightarrow \{s \leftarrow \text{ran}(\text{actv_inst})\}

END

\text{REDEPLOY: } \text{extended ordinary} \Rightarrow

\text{REFINES}

\text{REDEPLOY}

\text{ANY}

\text{s} >

\text{WHERE}

\text{grad1: } \text{s} \in \text{SERVICES} \not\text{theorem} >

\text{grad2: } \text{i_state}(\text{token_owner}(s) \Rightarrow s) = \text{FAIL_CONFIG}_4 \not\text{theorem} >

\text{grad7: } \text{actv_inst}(\{\text{token_owner}(s)\})[s] = \emptyset \not\text{theorem} >

\text{grad6: } \text{dep_inst}[s] \neq \emptyset \not\text{theorem} >

\text{grad4: } \text{card}(\text{run_peers}(s)) + \text{card}(\text{dep_inst}[s]) \geq \text{min_inst}(s) \not\text{theorem} >

\text{THEN}
act1: \[ \text{i\_state}(\text{token\_owner}(s) \leadsto s) = \text{DPL}\_4 \]
act2: \[ \text{run\_peers}(s) = \text{run\_peers}(s) \cup \text{dep\_inst}[\{s\}] \]
act3: \[ \text{dep\_inst} = \{s\} \land \text{dep\_inst} \]

\text{HEAL:} \quad \text{extended ordinary} \quad \text{REFINES} \quad \text{HEAL} \\
\text{ANY} \\
\quad s \quad \text{WHERE} \\
\quad \text{grd1: } s \in \text{SERVICES} \text{ not theorem} \\
\quad \text{grd2: } \text{i\_state}(\text{token\_owner}(s) \leadsto s) = \text{DPL}\_4 \text{ not theorem} \\
\quad \text{THEN} \\
\quad \text{act1: } \text{i\_state}(\text{token\_owner}(s) \leadsto s) = \text{RUN}\_4 \\
\quad \text{END} \\

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

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

\text{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
  i_state

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: not extended ordinary
  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
M13

\[ i_s \]
\[ p_s \]
\[ s_i \]

WHERE

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

grd2: \( prs \notin \text{unav_peers} \) not theorem

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

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

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

grd6: \( \forall \text{srv} \cdot \text{srv} \in \text{SERVICES} \land \text{token_owner(srv)} \in \text{prs} \land \text{token_owner(srv)} \rightarrow \text{srv} \in \text{dom(suspc_peers)} \rightarrow E(\text{srv}) \in \text{run_peers(\text{srv}) \setminus (\text{unav_peers u prs u fail_peers[\{\text{srv}\}]) not theorem} \) \( \forall \text{If the owner of the token for a service becomes unavailable and the service is not suspicious,} \)

then a new token owner among available peers is chosen

grd7: \( \forall \text{srv} \cdot \text{srv} \in \text{SERVICES} \land \text{token_owner(srv)} \in \text{prs} \land \text{token_owner(srv)} \rightarrow \text{srv} \in \text{dom(suspc_peers)} \rightarrow E(\text{srv}) \in \text{run_peers(\text{srv}) \setminus (\text{unav_peers u prs u fail_peers[\{\text{srv}\} u suspc_peers(token_owner(srv) \rightarrow \text{srv})]) not theorem} \) \( \forall \text{If the owner of the token for a service becomes unavailable and the service possesses suspicious instances, then a new token owner among available and not suspicious peers is chosen} \)

grd8: \( \forall \text{p, s} \cdot \text{p} \in \text{PEERS} \land \text{s} \in \text{SERVICES} \land \text{p} \mapsto \text{s} \in \text{dom(i_s)} \rightarrow p = E(\text{s}) \) not theorem

grd9: \( \forall \text{srv} \cdot \text{srv} \in \text{SERVICES} \rightarrow (E(\text{srv}) \mapsto \text{srv}) \mapsto \text{i_state} \) (token_owner(srv) \mapsto srv) \in i_s not theorem

grd10: \( \forall \text{p} \cdot \text{p} \in \text{PEERS} \land \text{s} \in \text{SERVICES} \land \text{p} \mapsto \text{s} \in \text{dom(p_s)} \rightarrow p = E(\text{s}) \) not theorem

grd11: \( \forall \text{p} \cdot \text{p} \in \text{PEERS} \land \text{s} \in \text{SERVICES} \land \text{p} \mapsto \text{s} \in \text{dom(s_i)} \rightarrow p = E(\text{s}) \) not theorem

grd12: \( \forall \text{srv} \cdot \text{srv} \in \text{SERVICES} \rightarrow (E(\text{srv}) \mapsto \text{srv}) \mapsto \text{suspc_peers} \) (token_owner(srv) \mapsto srv) \in p_s not theorem

grd13: \( \forall \text{p} \cdot \text{p} \in \text{PEERS} \land \text{s} \in \text{SERVICES} \land \text{p} \mapsto \text{s} \in \text{dom(s_i)} \rightarrow p = E(\text{s}) \) not theorem

grd14: \( \forall \text{srv} \cdot \text{srv} \in \text{SERVICES} \land \text{token_owner(srv)} \notin \text{prs} \rightarrow (E(\text{srv}) \mapsto \text{suspc.inst(E(srv) \mapsto srv)} \in s_i \) not theorem

grd15: \( \forall \text{srv} \cdot \text{srv} \in \text{SERVICES} \land \text{token_owner(srv)} \notin \text{prs} \rightarrow (E(\text{srv}) \mapsto \text{srv}) \rightarrow p = E(\text{s}) \) not theorem

THEN

act1: \( \text{unav_peers} = \text{unav_peers} \cup \text{prs} \rightarrow \text{the peers in prs become unavailable} \)
M13

act2: token_owner := token_owner ≔ E \new value for token owner

per service is given if needed

act3: rec_inst := prs ≔ rec_inst \the peers in prs can not try
to recontact instances anymore

act4: rct_inst := prs ≔ rct_inst \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: not 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

grd4: suspc_inst(token_owner(s) ↦ s) = ∅ not theorem \the

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

grd5: i_state(token_owner(s) ↦ s) = RUN_4 not theorem \the

state of s is OK

grd6: susp ≠ ∅ not theorem

THEN

act1: suspc_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: i_state(token_owner(s) ↦ s) = RUN_4 not theorem

grd3: suspc_inst(token_owner(s) ↦ s) ≠ ∅ not theorem

THEN

act1: i_state(token_owner(s) ↦ s) = FAIL_4

act2: suspc_peers(token_owner(s) ↦ s) = suspc_inst(token_owner

(s) ↦ s)

act3: suspc_inst(token_owner(s) ↦ s) = ∅

END
M13

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

THEN

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

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

THEN

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

END
recontact all the suspicious instances of $s$

THEN

act1: $\text{rec\_inst} = \text{rec\_inst} \cup \{\text{token\_owner}(s) \mapsto (s \mapsto i)\}$ \hspace{1em} \text{the token owner of } s \text{ has tried to recontact } i

END

FAIL\_DETECT: extended ordinary

REFINES

FAIL\_DETECT

ANY

$s \mapsto$

WHERE

$\text{grd1: } s \in \text{SERVICES not theorem}$

$\text{grd2: } \text{i\_state}(\text{token\_owner}(s) \mapsto s) = \text{FAIL\_4 not theorem}$

$\text{grd5: } \text{suscpc\_peers}(\text{token\_owner}(s) \mapsto s) \neq \emptyset \text{ not theorem}$

$\text{grd8: } \text{rec\_inst}[\{\text{token\_owner}(s)\}][\{s\}] = \text{suscpeers}$

$(\text{token\_owner}(s) \mapsto s) \text{ not theorem}$

THEN

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

act2: $\text{suscpc\_peers}(\text{token\_owner}(s) \mapsto s) = \text{suscpc\_peers}$

$(\text{token\_owner}(s) \mapsto s) \setminus \text{rct\_inst}[\{\text{token\_owner}(s)\}][\{s\}]$

act3: $\text{rec\_inst} = \text{rec\_inst} \cup \{\{s\} \setminus \text{ran}(\text{rec\_inst})\}$

act4: $\text{rct\_inst} = \text{rct\_inst} \cup \{\{s\} \setminus \text{ran}(\text{rct\_inst})\}$

END

IS\_OK: extended ordinary

REFINES

IS\_OK

ANY

$s \mapsto$

WHERE

$\text{grd1: } s \in \text{SERVICES not theorem}$

$\text{grd2: } \text{i\_state}(\text{token\_owner}(s) \mapsto s) = \text{FAIL\_DETECT\_4 not theorem}$

$\text{grd5: } \text{suscpc\_peers}(\text{token\_owner}(s) \mapsto s) = \emptyset \text{ not theorem}$

THEN

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

END

FAIL\_ACTIV: extended ordinary

REFINES

FAIL\_ACTIV

ANY

$s \mapsto$

WHERE

$\text{grd1: } s \in \text{SERVICES not theorem}$

$\text{grd2: } \text{i\_state}(\text{token\_owner}(s) \mapsto s) = \text{FAIL\_DETECT\_4 not theorem}$

Page 5
M13

grd5: \( \text{suspce}_{\text{ peers}}(\text{token}_{\text{ owner}}(s) \mapsto s) \neq \emptyset \) not theorem

\[ \text{THEN} \]

act1: \( \text{i}_{\text{ state}}(\text{token}_{\text{ owner}}(s) \mapsto s) = \text{FAIL}_{\text{ ACTIV}}.4 \)  

act2: \( \text{run}_{\text{ peers}}(s) = \text{run}_{\text{ peers}}(s) \setminus \text{suspce}_{\text{ peers}}(\text{token}_{\text{ owner}}(s) \mapsto s) \)  

act3: \( \text{fail}_{\text{ peers}} = \text{fail}_{\text{ peers}} \cup (\{s\} \times \text{suspce}_{\text{ peers}}(\text{token}_{\text{ owner}}(s) \mapsto s)) \)  

act4: \( \text{suspce}_{\text{ peers}}(\text{token}_{\text{ owner}}(s) \mapsto s) = \emptyset \)  

\[ \text{END} \]

FAIL_CONFIGURE: extended ordinary  
REFINES  
FAIL_CONFIGURE

\[ \text{ANY} \]

s  
WHERE

\[ \text{grd1: s \in SERVICES not theorem} \]  
\[ \text{grd2: } \text{i}_{\text{ state}}(\text{token}_{\text{ owner}}(s) \mapsto s) = \text{FAIL}_{\text{ ACTIV}}.4 \text{ not theorem} \]  
\[ \text{grd3: card}(\text{run}_{\text{ peers}}(s)) < \text{min}_{\text{ inst}}(s) \text{ not theorem} \]  

\[ \text{THEN} \]

act1: \( \text{i}_{\text{ state}}(\text{token}_{\text{ owner}}(s) \mapsto s) = \text{FAIL}_{\text{ CONFIG}}.4 \)  

\[ \text{END} \]

FAIL_IGNORE: extended ordinary  
REFINES  
FAIL_IGNORE

\[ \text{ANY} \]

s  
WHERE

\[ \text{grd1: s \in SERVICES not theorem} \]  
\[ \text{grd2: } \text{i}_{\text{ state}}(\text{token}_{\text{ owner}}(s) \mapsto s) = \text{FAIL}_{\text{ ACTIV}}.4 \text{ not theorem} \]  
\[ \text{grd3: card}(\text{run}_{\text{ peers}}(s)) \geq \text{min}_{\text{ inst}}(s) \text{ not theorem} \]  

\[ \text{THEN} \]

act1: \( \text{i}_{\text{ state}}(\text{token}_{\text{ owner}}(s) \mapsto s) = \text{FAIL}_{\text{ IGN}}.4 \)  

\[ \text{END} \]

IGNORE: extended ordinary  
REFINES  
IGNORE

\[ \text{ANY} \]

s  
WHERE

\[ \text{grd1: s \in SERVICES not theorem} \]  
\[ \text{grd2: } \text{i}_{\text{ state}}(\text{token}_{\text{ owner}}(s) \mapsto s) = \text{FAIL}_{\text{ IGN}}.4 \text{ not theorem} \]  

\[ \text{THEN} \]

act1: \( \text{i}_{\text{ state}}(\text{token}_{\text{ owner}}(s) \mapsto s) = \text{RUN}.4 \)  

\[ \text{END} \]
REDEPLOY_INSTC: extended ordinary

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) ∪ fail_peers[s] ∪ unav_peers ∪

dep_inst[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

THEN

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

END

REDEPLOY_INSTS: extended ordinary

REDEPLOY_INSTS

ANY
s →

WHERE

grd1: s ∈ SERVICES not theorem
grd6: card(actv_inst[{token_owner(s)}][{s}]) = deplo_inst(s)

grd7: card(dep_inst[{s}]) + card(run_peers(s)) < min_inst(s)

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

THEN

act1: dep_inst = dep_inst ∪ {{s}×actv_inst[{token_owner(s)}]

act2: actv_inst = actv_inst ∪ {{s} ↦ ran(actv_inst)}

END

REDEPLOY: extended ordinary

REDEPLOY

ANY
s →

WHERE
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}]=∅ not theorem >
grd6:  dep_inst[{s}] ≠ ∅ not theorem >
grd4:  card(run_peers(s))+card(dep_inst[{s}]) ≥ min_inst(s) not theorem >

THEN
act1:  i_state(token_owner(s) ↦ 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:  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
INVARIANTS
  inv1: rect_inst ∈ (PEERS×SERVICES) ⇸ ℙ(PEERS) not theorem
  p = token_owner(s) ⇒ (p ↦ s) ∈ dom(rect_inst)
  inv2: ∀ p, s · p ∈ PEERS ∧ s ∈ SERVICES ∧ (p ↦ s) ∈ dom(rect_inst) ⇒
        p = token_owner(s) ⇒ (p ↦ s) ∈ dom(rect_inst) not theorem
  inv3: ∀ p, s · p ∈ PEERS ∧ s ∈ SERVICES ∧ (p ↦ s) ∈ dom(rect_inst) ⇒
        p = token_owner(s) ⇒ (p ↦ s) ∈ dom(rect_inst) not theorem
  gluing_tok_own_rec1: ∀ p, s · p ∈ PEERS ∧ s ∈ SERVICES ∧ (p ↦ s) ∈ dom(rect_inst) ⇒
                      rect_inst[p][s] = rect_inst(p ↦ s) not theorem
  inv4: rctt_inst ∈ (PEERS×SERVICES) ⇸ ℙ(PEERS) not theorem
  inv5: ∀ p, s · p ∈ PEERS ∧ s ∈ SERVICES ∧ (p ↦ s) ∈ dom(rctt_inst) ⇒
        p = token_owner(s) ⇒ (p ↦ s) ∈ dom(rctt_inst) not theorem
  inv6: ∀ p, s · p ∈ PEERS ∧ s ∈ SERVICES ∧ (p ↦ s) ∈ dom(rctt_inst) ⇒
        p = token_owner(s) ⇒ (p ↦ s) ∈ dom(rctt_inst) not theorem
  gluing_tok_own_rct1: ∀ p, s · p ∈ PEERS ∧ s ∈ SERVICES ∧ (p ↦ s) ∈ dom(rctt_inst) ⇒
                       rctt_inst[p][s] = rctt_inst(p ↦ s) not theorem
  inv7: failr_peers ∈ SERVICES ⇸ ℙ(PEERS) not theorem
  inv8: dep_instc ∈ SERVICES ⇸ ℙ(PEERS) not theorem
  dep_instc = dep_instc(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 = ∅
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
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
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
grd10: ∀ srv · srv ∈ SERVICES ∧ token_owner(srv) ∈ prs ⇒ E
(srv) ∈ run_peers(srv)\(unav_peers u prs u failr_peers(srv) u suspc_peers
(token_owner(srv) ↦ srv)) not theorem

If the token owner of a service srv does not belong to prs, the token owner is not changed.

If the token owner of a service srv becomes unavailable, and the service possesses 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

color: #0f0; text-align: justify; font-size: 0.8em; display: block;
\[(\text{srv} \mapsto \text{srv}) \in \text{rt}_s \land ((\text{E(srv)} \mapsto \text{srv}) \mapsto \text{rect}_\text{inst} (\text{E(srv)} \mapsto \text{srv}) \in \text{rc}_s)\] not theorem

grd14: \(\forall \text{ srv} \in \text{SERVICES} \land \text{token}_\text{owner}(\text{srv}) \in \text{prs} \Rightarrow ((\text{E(srv)} \mapsto \text{srv}) \mapsto \emptyset \in \text{s}_i) \land ((\text{E(srv)} \mapsto \text{srv}) \mapsto \emptyset \in \text{rt}_s) \land ((\text{E(srv)} \mapsto \text{srv}) \mapsto \emptyset \in \text{rc}_s)\) not theorem

 THEN
unavailable
act1: \(\text{unav}_\text{peers} = \text{unav}_\text{peers} \cup \text{prs}\) the peers in prs become unavailable
act2: \(\text{token}_\text{owner} = \text{token}_\text{owner} \odot \text{E}\) new value for token owner per service is given if needed
act3: \(\text{rect}_\text{inst} = \text{rc}_s\) the peers in prs can not try to recontact instances anymore
act4: \(\text{rctt}_\text{inst} = \text{rt}_s\) the peers in prs can not recontact instances anymore
act5: \(\text{actv}_\text{inst} = \text{prs} \odot \text{actv}_\text{inst}\) the members of susp become suspected instances for s by the token owner of s
act6: \(\text{i}_\text{state} = \text{i}_s\) the peers in prs can not try to recontact instances anymore
act7: \(\text{suspc}_\text{peers} = \text{p}_s\) the members of susp have not yet been suspected for s by the token owner of s
act8: \(\text{suspc}_\text{inst} = \text{s}_i\) the state of s is OK

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
grd2: \(\text{susp} \subseteq \text{PEERS}\) not theorem
grd3: \(\text{susp} = \text{run}_\text{peers}(s) \cap \text{unav}_\text{peers}\) not theorem instances in susp are suspicious if the peers running them becomes unavailable
grd4: \(\text{suspc}_\text{inst}(\text{token}_\text{owner}(s) \mapsto s) = \emptyset\) not theorem the member of susp have not yet been suspected for s by the token owner of s
grd5: \(\text{i}_\text{state}(\text{token}_\text{owner}(s) \mapsto s) = \text{RUN}_4\) not theorem the state of s is OK
grd6: \(\text{susp} \neq \emptyset\) not theorem

 THEN
act1: \(\text{suspc}_\text{inst}(\text{token}_\text{owner}(s) \mapsto s) = \text{susp}\) the members of susp become suspected instances for s by the token owner of s
act2: \(\text{token}_\text{owner} = \text{token}_\text{owner} \odot \text{E}\) new value for token owner per service is given if needed
act3: \(\text{rect}_\text{inst} = \text{rc}_s\) the peers in prs can not try to recontact instances anymore
act4: \(\text{rctt}_\text{inst} = \text{rt}_s\) the peers in prs can not recontact instances anymore
act5: \(\text{actv}_\text{inst} = \text{prs} \odot \text{actv}_\text{inst}\) the members of susp become suspected instances for s by the token owner of s
act6: \(\text{i}_\text{state} = \text{i}_s\) the peers in prs can not try to recontact instances anymore
act7: \(\text{suspc}_\text{peers} = \text{p}_s\) the members of susp have not yet been suspected for s by the token owner of s
act8: \(\text{suspc}_\text{inst} = \text{s}_i\) the state of s is OK

END

FAIL: extended ordinary
REFINES FAIL
ANY
s \(\rightarrow\)
WHERE
grd1: \(s \in \text{SERVICES}\) not theorem

Page 3
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
  the state of s is SUSPICIOUS
  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 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) ⊂ suspc_peers(token_owner(s) ↦ s)
  the token owner of s has not yet tried to recontact all the suspensive 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: not 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: i_state(token_owner(s) ↦ s) = FAIL_4 not theorem
  the state of s is SUSPICIOUS
  grd4: suspc_peers(token_owner(s) ↦ s) ≠ ∅ not theorem
  the set
of suspicious peers for s is not empty

\( \text{grd5: } i \in \text{suspc.peers(token_owner(s) ↦ s)} \) \( \text{nunav.peers not}
\)
theorem \( \triangleright i \) is a suspicious instance of s and i is unavailable (can not be contacted)

\( \text{grd6: } i \in \text{rect.inst(token_owner(s) ↦ s)} \) \( \text{not theorem} \) \( \triangleright \) the token owner of s has not yet tried to recontact i

\( \text{grd7: } \text{rect.inst(token_owner(s) ↦ s)} \subseteq \text{suspc.peers(token_owner(s) ↦ s)} \) \( \text{not theorem} \) \( \triangleright \) the token owner of s has not yet tried to recontact all the suspicious instances of s

\begin{align*}
\text{THEN} \\
\text{act1: } & \text{rect.inst(token_owner(s) ↦ s)} = \text{rect.inst(token_owner(s) ↦ s)} \\
& s \cup \{ i \} \quad \triangleright \text{the token owner of s has tried to recontact i}
\end{align*}

\begin{align*}
\text{FAIL_DETECT: } & \text{not extended ordinary } \\
\text{REFINES } & \text{FAIL_DETECT} \\
\text{ANY } & \text{s } \\
\text{WHERE} \\
\text{grd1: } & s \in \text{SERVICES not theorem} \\
\text{grd2: } & \text{i_state(token_owner(s) ↦ s)} = \text{FAIL}_4 \text{ not theorem} \\
\text{grd5: } & \text{suspc.peers(token_owner(s) ↦ s)} \neq \emptyset \text{ not theorem} \\
\text{grd8: } & \text{rect.inst(token_owner(s) ↦ s)} = \text{suspc.peers(token_owner(s) ↦ s)} \\
\text{(token_owner(s) ↦ s)} \text{ not theorem} \\
\text{THEN} \\
\text{act1: } & \text{i_state(token_owner(s) ↦ s)} = \text{FAIL_DETECT}_4 \\
\text{act2: } & \text{suspc.peers(token_owner(s) ↦ s)} = \text{suspc.peers} \\
\text{act3: } & \text{rect.inst(token_owner(s) ↦ s)} = \emptyset \\
\text{act4: } & \text{rctt.inst(token_owner(s) ↦ s)} = \emptyset
\end{align*}

\begin{align*}
\text{END} \\
\text{END}
\end{align*}

\begin{align*}
\text{IS_OK: } & \text{extended ordinary } \\
\text{REFINES } & \text{IS_OK} \\
\text{ANY } & \text{s } \\
\text{WHERE} \\
\text{grd1: } & s \in \text{SERVICES not theorem} \\
\text{grd2: } & \text{i_state(token_owner(s) ↦ s)} = \text{FAIL_DETECT}_4 \text{ not theorem} \\
\text{grd5: } & \text{suspc.peers(token_owner(s) ↦ s)} = \emptyset \text{ not theorem} \\
\text{THEN} \\
\text{act1: } & \text{i_state(token_owner(s) ↦ s)} = \text{RUN}_4
\end{align*}

\begin{align*}
\text{END} \\
\text{FAIL_ACTIV: } & \text{not extended ordinary }
\end{align*}
M14

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_DETECT_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_IGNORE_4 >
END

IGNORE: extended ordinary >
REFINES
  IGNORE
ANY
M14

\[ s \to \]

WHERE
\begin{align*}
\text{grd1: } & s \in \text{SERVICES} \quad \text{not theorem} \\
\text{grd2: } & i_{\text{state}}(\text{token_owner}(s) \to s) = \text{FAIL}_\text{IGN}_4 \quad \text{not theorem} \\
\end{align*}

THEN
\begin{align*}
\text{act1: } & i_{\text{state}}(\text{token_owner}(s) \to s) = \text{RUN}_4 \\
\end{align*}

END

REDEPLOY\_INSTC: \text{not extended ordinary} \to

REFINES

REDEPLOY\_INSTC

ANY
\begin{align*}
  s & \to \text{a service } s \\
  i & \to \text{an instance } i \\
\end{align*}

WHERE
\begin{align*}
\text{grd1: } & s \in \text{SERVICES} \quad \text{not theorem} \\
\text{grd2: } & i \in \text{PEERS} \quad \text{not theorem} \\
\text{grd3: } & i \not\in \text{run_peers}(s) \cup \text{failr_peers}(s) \cup \text{unav_peers} \cup \text{dep_instc}(s) \quad \text{not theorem} \\
\text{grd4: } & \text{token_owner}(s) \to (s \to i) \not\in \text{actv}_\text{inst} \quad \text{not theorem} \\
\text{grd5: } & i_{\text{state}}(\text{token_owner}(s) \to s) = \text{FAIL}_\text{CONFIG}_4 \quad \text{not theorem} \\
\text{grd6: } & \text{card(\text{actv}_\text{inst}[\{\text{token_owner}(s)\}][\{s\}])} < \text{deplo}_\text{inst}(s) \quad \text{not theorem} \\
\text{grd7: } & \text{card(\text{dep}_\text{instc}(s))} + \text{card(\text{run_peers}(s))} < \text{min}_\text{inst}(s) \quad \text{not theorem} \\
\end{align*}

THEN
\begin{align*}
\text{act1: } & \text{actv}_\text{inst} = \text{actv}_\text{inst} \cup \{\text{token_owner}(s) \to (s \to i)\} \\
\end{align*}

END

REDEPLOY\_INSTS: \text{not extended ordinary} \to

REFINES

REDEPLOY\_INSTS

ANY
\begin{align*}
  s & \to \\
\end{align*}

WHERE
\begin{align*}
\text{grd1: } & s \in \text{SERVICES} \quad \text{not theorem} \\
\text{grd6: } & \text{card(\text{actv}_\text{inst}[\{\text{token_owner}(s)\}][\{s\}])} = \text{deplo}_\text{inst}(s) \quad \text{not theorem} \\
\text{grd7: } & \text{card(\text{dep}_\text{instc}(s))} + \text{card(\text{run_peers}(s))} < \text{min}_\text{inst}(s) \quad \text{not theorem} \\
\text{grd8: } & i_{\text{state}}(\text{token_owner}(s) \to s) = \text{FAIL}_\text{CONFIG}_4 \quad \text{not theorem} \\
\end{align*}

THEN
\begin{align*}
\text{act1: } & \text{dep}_\text{instc}(s) = \text{dep}_\text{instc}(s) \cup \text{actv}_\text{inst}[\{\text{token_owner}(s)\}][\{s\}] \\
\text{act2: } & \text{actv}_\text{inst} = \text{actv}_\text{inst} \cup \{\{s\} \leftarrow \text{ran(\text{actv}_\text{inst})}\} \\
\end{align*}
REDEPLOY: not extended ordinary
REFINES
REDEPLOY
ANY
s
WHERE
  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_instc(s) ≠ ø not theorem
  grd4: card(run_peers(s)) + card(dep_instc(s)) ≥ min_inst(s) 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) ≔ ø
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: not 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
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

END

Page 9
M15

MACHINE M15
REFINES M14
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_instc > instances activated by token owne
  i_state

INVARIANTS
  inv1:  actv_instc ∈ (PEERS×SERVICES) ⇸ ℙ(PEERS) not theorem
  inv2:  ∀ p, s · p ∈ PEERS ∧ s ∈ SERVICES ∧ (p ↦ s) ∈ dom(actv_instc) ⇒
         p = token_owner(s) not theorem
  inv3:  ∀ p, s · p ∈ PEERS ∧ s ∈ SERVICES ∧ p = token_owner(s) ⇒ (p ↦ s) ∈
         dom(actv_instc) not theorem
  gluing_tok_own_rec1: ∀ p, s · p ∈ PEERS ∧ s ∈ SERVICES ∧ (p ↦ s) ∈
                      dom(actv_instc) ⇒ actv_instc[p][s] = actv_instc[p ↦ 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 ≔ ∅
  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  > new values for token owner per service if needed
M15

\[\begin{align*}
i_s & \triangleright \\
p_s & \triangleright \\
s_i & \triangleright \\
rc_s & \triangleright \\
rt_s & \triangleright \\
ac_i & \triangleright \\
\end{align*}\]

WHERE

- **grd1**: \(prs \subseteq \text{PEERS}\) not theorem
- **grd2**: \(prs \not\subseteq \text{unav_peers}\) not theorem
- **grd3**: \(E \in \text{SERVICES} \rightarrow \text{PEERS}\) not theorem
- **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**: \(p_s \in (\text{PEERS} \times \text{SERVICES}) \leftrightarrow \text{P(PEERS)}\) not theorem
- **grd6**: \(s_i \in (\text{PEERS} \times \text{SERVICES}) \leftrightarrow \text{P(PEERS)}\) not theorem
- **grd7**: \(rt_s \in (\text{PEERS} \times \text{SERVICES}) \leftrightarrow \text{P(PEERS)}\) not theorem
- **grd8**: \(rc_s \in (\text{PEERS} \times \text{SERVICES}) \leftrightarrow \text{P(PEERS)}\) not theorem
- **grd9**: \(ac_i \in (\text{PEERS} \times \text{SERVICES}) \leftrightarrow \text{P(PEERS)}\) not theorem
- **grd10**: \(\text{dom}(i_s) = E \triangleright \text{dom}(p_s) = \text{dom}(i_s) \wedge \text{dom}(s_i) = \text{dom}(i_s) \wedge \text{dom}(rc_s) = \text{dom}(i_s) \wedge \text{dom}(rt_s) = \text{dom}(i_s) \wedge \text{dom}(ac_i) = \text{dom}(i_s)\) not theorem

**NEW VALUE FOR TOKEN OWNER PER SERVICE**

- **grd11**: \(\forall \text{srv} \cdot \text{srv} \in \text{SERVICES} \wedge \text{token_owner(srv)} \in \text{prs} \Rightarrow \)
  \(E(\text{srv}) = \text{token_owner(srv)} \wedge s_i(E(\text{srv}) \triangleright \text{srv}) = \text{suspcc_peers}(E(\text{srv}) \triangleright \text{srv}) \wedge rt_s(E(\text{srv}) \triangleright \text{srv}) = \text{rctt_peers}(E(\text{srv}) \triangleright \text{srv}) \wedge rc_s(E(\text{srv}) \triangleright \text{srv}) = \text{rect_peers}(E(\text{srv}) \triangleright \text{srv}) \wedge ac_i(E(\text{srv}) \triangleright \text{srv}) = \text{actv_instc}(E(\text{srv}) \triangleright \text{srv})\) not theorem

**NEW VALUE FOR TOKEN OWNER PER SERVICE**

- **grd12**: \(\forall \text{srv} \cdot \text{srv} \in \text{SERVICES} \wedge \text{token_owner(srv)} \in \text{prs} \Rightarrow \)
  \(E(\text{srv}) \in \text{run_peers}(\text{srv}) \setminus (\text{unav_peers} \cup \text{prs} \cup \text{failr_peers}(\text{srv}) \cup \text{suspcc_peers}(\text{token_owner(srv)} \triangleright \text{srv})) \wedge s_i(E(\text{srv}) \triangleright \text{srv}) = \emptyset \wedge rt_s(E(\text{srv}) \triangleright \text{srv}) = \emptyset \wedge rc_s(E(\text{srv}) \triangleright \text{srv}) = \emptyset \wedge ac_i(E(\text{srv}) \triangleright \text{srv}) = \emptyset\) not theorem

**NEW VALUE FOR TOKEN OWNER PER SERVICE**

- **grd13**: \(\forall \text{srv} \cdot \text{srv} \in \text{SERVICES} \Rightarrow i_s(E(\text{srv}) \triangleright \text{srv}) = \text{i_state(token_owner(srv)} \triangleright \text{srv}) \wedge p_s(E(\text{srv}) \triangleright \text{srv}) = \text{suspcc_peers(token_owner(srv)} \triangleright \text{srv})\) not theorem
  THEN
  \(\begin{align*}
  \text{act1: unav_peers} & = \text{unav_peers} \cup \text{prs} \rightarrow \text{the peers in prs become unavailable}
  \end{align*}\)
act2: \( \text{token}_\text{owner} := \text{token}_\text{owner} \leftarrow E \) \( \triangleright \) new value for \( \text{token}_\text{owner} \) per service is given if needed

act3: \( \text{rect}_\text{inst} = \text{rc}_s \) \( \triangleright \) the peers in \( \text{prs} \) can not try to recontact instances anymore

act4: \( \text{rcrt}_\text{inst} = \text{rt}_s \) \( \triangleright \) the peers in \( \text{prs} \) can not recontact instances anymore

act5: \( \text{actv}_\text{instc} = \text{ac}_i \) 

act6: \( \text{i}_\text{state} = \text{i}_s \) 

act7: \( \text{suspc}_\text{peers} = \text{p}_s \) 

act8: \( \text{suspc}_\text{inst} = \text{s}_i \) 

END

\textbf{SUSPECT}_\text{INST:} \text{ extended ordinary } \triangleright \text{ REFINES } \text{ SUSPECT}_\text{INST} 

\text{ ANY } 

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

\( \text{suspc} \) \( \triangleright \) suspicious instances

\text{ WHERE } 

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

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

\( \text{grd3: } \text{suspc} = \text{run.peers}(s) \cap \text{unav.peers} \not\text{ theorem } \triangleright \)

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

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

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

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

the state of \( s \) is OK

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

THEN

\( \text{act1: } \text{suspc}_\text{inst}(\text{token}_\text{owner}(s) \leftarrow s) = \text{suspc} \triangleright \)

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

END

\textbf{FAIL:} \text{ extended ordinary } \triangleright \text{ REFINES } \text{ FAIL} 

\text{ ANY } 

\( s \) \( \triangleright \)

\text{ WHERE } 

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

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

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

THEN

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

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

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

END
RECONTACT_INST_OK: extended ordinary
RECONTACT_INST_OK
RECONTACT_INST_OK
RECONTACT_INST_OK

RECONTACT_INST_KO: extended ordinary
RECONTACT_INST_KO
RECONTACT_INST_KO
RECONTACT_INST_KO

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: 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) 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
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: 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) \ rectt_inst(token_owner(s) ↦ s) >
  act3: rect_inst(token_owner(s) ↦ s) = ∅ >
  act4: rectt_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 >
  grd5: suspc_peers(token_owner(s) ↦ s) = ∅ not theorem >
M15

grd5: \text{suspc\_peers(token\_owner(s) ↦ s) \neq \emptyset \text{ not theorem} →}

THEN
act1: \text{i\_state(token\_owner(s) ↦ s) = FAIL\_ACTIV\_4} →
act2: \text{run\_peers(s) = run\_peers(s) \setminus suspc\_peers(token\_owner(s) ↦ s)} →
act3: \text{failr\_peers(s) = failr\_peers(s) \cup suspc\_peers(token\_owner(s) ↦ s)} →
act4: \text{suspc\_peers(token\_owner(s) ↦ s) = \emptyset} →

END

FAIL\_CONFIGURE: \text{extended ordinary} →
REFINES
FAIL\_CONFIGURE
ANY
s →
WHERE
grd1: \text{s ∈ SERVICES \text{ not theorem} →}
grd2: \text{i\_state(token\_owner(s) ↦ s) = FAIL\_ACTIV\_4 \text{ not theorem} →}
grd3: \text{card(run\_peers(s)) < min\_inst(s) \text{ not theorem} →}
THEN
act1: \text{i\_state(token\_owner(s) ↦ s) = FAIL\_CONF\_4} →
END

FAIL\_IGNORE: \text{extended ordinary} →
REFINES
FAIL\_IGNORE
ANY
s →
WHERE
grd1: \text{s ∈ SERVICES \text{ not theorem} →}
grd2: \text{i\_state(token\_owner(s) ↦ s) = FAIL\_ACTIV\_4 \text{ not theorem} →}
grd3: \text{card(run\_peers(s)) ≥ min\_inst(s) \text{ not theorem} →}
THEN
act1: \text{i\_state(token\_owner(s) ↦ s) = FAIL\_IGN\_4} →
END

IGNORE: \text{extended ordinary} →
REFINES
IGNORE
ANY
s →
WHERE
grd1: \text{s ∈ SERVICES \text{ not theorem} →}
grd2: \text{i\_state(token\_owner(s) ↦ s) = FAIL\_IGN\_4 \text{ not theorem} →}
THEN
act1: \text{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) ∪ fail_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: i ∉ actv_instc(token_owner(s) ↦ s) not theorem
grd5: i_state(token_owner(s) ↦ s) = FAIL_CONFIG_4 not theorem
grd6: card(actv_instc(token_owner(s) ↦ s)) < deplo_inst(s) not theorem
i does not run s, is not failed for s, is not unavailable and is not already activated for s
grd7: card(dep_instc(s)) + card(run_peers(s)) < min_inst(s)
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 →

WHERE
grd1: s ∈ SERVICES not theorem
grd6: card(actv_instc(token_owner(s) ↦ s)) = deplo_inst(s) not theorem
grd7: card(dep_instc(s)) + card(run_peers(s)) < min_inst(s)
THEN
act1: dep_instc(s) = dep_instc(s) u actv_instc(token_owner(s) ↦ s)
END

REDEPLOY: not extended ordinary
REFINES
REDEPLOY

ANY
s →
WHERE

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

>  

grd7:  actv_instc(token_owner(s) ↦ s)=Ø not theorem
grd6:  dep_instc(s) ≠ Ø not theorem
grd4:  card(run_peers(s))+card(dep_instc(s)) ≥ min_inst(s) 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

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:  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
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 owns
  inst_state

INVARINTS
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 ⇒ i_state(token_owner(s) ↦ s) = inst_state(token_owner(s) ↦ s) not theorem ▹
inv4: ∀ s · s ∈ SERVICES ⇒ rctt_inst(token_owner(s) ↦ s) ⊆ run_peers(s) not theorem ▹
inv5: ∀ s · s ∈ SERVICES ⇒ suspc_peers(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 ▹
EVENTS

INITIALISATION:

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:

REFINES

\[
\text{MAKE_PEER_UNAVAIL}
\]

ANY

prs
E

p_s
s_i
rc_s
rt_s
ac_i

WHERE

\[
\begin{align*}
grd1: & \quad \text{prs} \subseteq \text{PEERS} \not\implies \text{theorem} \\
grd2: & \quad \text{prs} \not\subseteq \text{unav_peers} \not\implies \text{theorem} \\
grd3: & \quad \forall \text{srv} \cdot \text{srv} \in \text{SERVICES} \implies \text{dom(dom(inst_state)} \triangleright \{\text{srv}\}) \\
grd4: & \quad E \in \text{SERVICES} \rightarrow \text{PEERS} \not\implies \text{theorem} \\
grd5: & \quad \text{prs} \subseteq (\text{PEERS} \times \text{SERVICES}) \rightarrow \mathcal{P}(\text{PEERS}) \not\implies \text{theorem} \\
grd6: & \quad \text{s_i} \subseteq (\text{PEERS} \times \text{SERVICES}) \rightarrow \mathcal{P}(\text{PEERS}) \not\implies \text{theorem} \\
grd7: & \quad \text{rt_s} \subseteq (\text{PEERS} \times \text{SERVICES}) \rightarrow \mathcal{P}(\text{PEERS}) \not\implies \text{theorem} \\
grd8: & \quad \text{rc_s} \subseteq (\text{PEERS} \times \text{SERVICES}) \rightarrow \mathcal{P}(\text{PEERS}) \not\implies \text{theorem} \\
grd9: & \quad \text{ac_i} \subseteq (\text{PEERS} \times \text{SERVICES}) \rightarrow \mathcal{P}(\text{PEERS}) \not\implies \text{theorem} \\
grd10: & \quad \text{dom(p_s)} = E \wedge \text{dom(s_i)} = E \wedge \text{dom(rc_s)} = E \wedge \text{dom(rt_s)} = E \wedge \text{dom(ac_i)} = E \\
\end{align*}
\]

(\text{rt_s}) = E \wedge \text{dom(ac_i)} = E \not\implies \text{theorem} \\

\[
\begin{align*}
grd11: & \quad \forall \text{srv} \cdot \text{srv} \in \text{SERVICES} \wedge \text{token_owner}(\text{srv}) \notin \text{prs} \\
& \implies \quad E(\text{srv}) = \text{token_owner}(\text{srv}) \wedge \\
& \quad \text{s_i}(E(\text{srv}) \triangleright \text{srv}) = \text{suspce}\_\text{inst}(E(\text{srv}) \triangleright \text{srv}) \wedge \\
& \quad \text{rt_s}(E(\text{srv}) \triangleright \text{srv}) = \text{rctt}\_\text{inst}(E(\text{srv}) \triangleright \text{srv}) \wedge \\
& \quad \text{rc_s}(E(\text{srv}) \triangleright \text{srv}) = \text{rect}\_\text{inst}(E(\text{srv}) \triangleright \text{srv}) \wedge \\
& \quad \text{ac_i}(E(\text{srv}) \triangleright \text{srv}) = \text{actv}\_\text{instc}(E(\text{srv}) \triangleright \text{srv}) \not\implies \text{theorem}
\end{align*}
\]
M16

Theorem >If the token owner of a service srv does not belong to prs, the token owner is not changed.

\[ \forall \text{srv} \cdot \text{srv} \in \text{SERVICES} \land \text{token_owner(srv)} \in \text{prs} \implies \]
\[ \text{E(srv)} \in \text{run_peers(srv)} \setminus (\text{unav_peers} \cup \text{prs} \cup \text{failr_peers} \cup \text{suspc_peers(token_owner(srv))}) \land \]
\[ \text{E(srv) \cap srv \in dom(inst_state)} \land \]
\[ \text{inst_state(E(srv) \cap srv) = inst_state(token_owner(srv) \cap srv)} \land \]
\[ \text{s_i(E(srv) \cap srv) = \emptyset \land} \]
\[ \text{rt_s(E(srv) \cap srv) = \emptyset \land} \]
\[ \text{rc_s(E(srv) \cap srv) = \emptyset \land} \]
\[ \text{ac_i(E(srv) \cap srv) = \emptyset} \notag \]

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.

\[ \forall \text{srv} \cdot \text{srv} \in \text{SERVICES} \implies \text{p_s(E(srv)) = suspc_peers(token_owner(srv))} \notag \]

WITH
\[ \text{i_s: i_s = E∼ \triangleleft inst_state} \]

THEN
\[ \text{act1: unav_peers = unav_peers \cup prs} \notag \]
new value for token owner per service is given if needed
\[ \text{act2: token_owner = token_owner \oplus E} \notag \]
the peers in prs become unavailable
\[ \text{act3: rect_inst = rc_s} \notag \]
the peers in prs can not try to recontact instances anymore
\[ \text{act4: rctt_inst = rt_s} \notag \]
the peers in prs can not recontact instances anymore
\[ \text{act5: activ_instc = ac_i} \notag \]
\[ \text{act6: suspc_peers = p_s} \notag \]
\[ \text{act7: suspc_inst = s_i} \notag \]
\[ \text{act8: inst_state = (prs \times SERVICES) \oplus inst_state} \notag \]

END

SUSPECT_INST: not extended ordinary

REFINES

SUSPECT_INST

ANY
\[ \text{s \notag} \]
a service s
\[ \text{susp \notag} \]
suspicious instances

WHERE
\[ \text{grd1: s \in SERVICES} \notag \]
\[ \text{grd2: susp \subseteq PEERS} \notag \]
\[ \text{grd3: susp = run_peers(s) \cap unav_peers} \notag \]
instances in susp are suspicious if the peers running them becomes unavailable.
M16

**grd4:** \( \text{suspc}_{\text{inst}}(\text{token}_{\text{owner}}(s) \mapsto s) = \emptyset \) not theorem \( \triangleright \) the 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 \( \triangleright \) the state of \( s \) is OK

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

**THEN**

**act1:** \( \text{suspc}_{\text{inst}}(\text{token}_{\text{owner}}(s) \mapsto s) = \text{suspc} \) \( \triangleright \) the members of susp become suspected instances for \( s \) by the token owner of \( s \)

**END**

**FAIL:** not extended ordinary \( \triangleright \)

**REFINES**

**FAIL**

**ANY**

**s** \( \triangleright \)

**prop** \( \triangleright \)

**WHERE**

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

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

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

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

**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 \( \triangleright \)

**THEN**

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

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

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

**END**

**RECONTACT INST_OK:** not extended ordinary \( \triangleright \)

**REFINES**

**RECONTACT INST_OK**

**ANY**

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

**i** \( \triangleright \) an instance \( i \)

**WHERE**

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

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

**grd3:** \( \text{inst}_{\text{state}}(\text{token}_{\text{owner}}(s) \mapsto s) = \text{FAIL}_4 \) not theorem \( \triangleright \) the state of \( s \) is SUSPICIOUS

**grd4:** \( \text{suspc}_{\text{peers}}(\text{token}_{\text{owner}}(s) \mapsto s) \neq \emptyset \) not theorem \( \triangleright \) 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 \( \triangleright \) \( i \) 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 \( \triangleright \) 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 \( \triangleright \)
(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 →
grd2: i ∈ PEERS not theorem →
grd3: inst_state(token_owner(s) ↦ s) = FAIL_4 not theorem → the state of s is SUSPICIOUS

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 (cannot 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) 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 →

(s → s) not theorem →
M16

\[\text{grd6: prop} = (\text{run_peers(s)} \setminus \text{suspc_peers(token_owner(s) \mapsto s)}) \cup \text{rctt_inst(token_owner(s) \mapsto s)} \setminus \text{unav_peers}\]

\(\text{THEN}\)

\[\begin{align*}
\text{act1: inst_state} & = \text{inst_state} \setminus ((\text{prop} \times \{s\}) \times \{\text{FAIL_DETECT}_4\}) \\
\text{act2: suspc_peers(token_owner(s) \mapsto s)} & = \text{suspc_peers} (\text{token_owner(s) \mapsto s}) \setminus \text{rctt_inst(token_owner(s) \mapsto s)} \\
\text{act3: rect_inst(token_owner(s) \mapsto s)} & = \emptyset \\
\text{act4: rctt_inst(token_owner(s) \mapsto s)} & = \emptyset
\end{align*}\]

\(\text{END}\)

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

\(\text{REFINES}\)

\[\text{IS_OK}\]

\(\text{ANY}\)

\[\begin{align*}
s \mapsto \text{prop}
\end{align*}\]

\(\text{WHERE}\)

\[\begin{align*}
\text{grd1: s} & \in \text{SERVICES not theorem} \\
\text{grd2: prop} & \subseteq \text{PEERS not theorem} \\
\text{grd3: inst_state(token_owner(s) \mapsto s)} & = \text{FAIL_DETECT}_4 \text{ not theorem} \\
\text{grd4: suspc_peers(token_owner(s) \mapsto s)} & = \emptyset \text{ not theorem} \\
\text{grd5: prop} & = \text{run_peers(s)} \setminus \text{unav_peers} \text{ not theorem}
\end{align*}\]

\(\text{THEN}\)

\[\begin{align*}
\text{act1: inst_state} & = \text{inst_state} \setminus ((\text{prop} \times \{s\}) \times \{\text{RUN}_4\})
\end{align*}\]

\(\text{END}\)

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

\(\text{REFINES}\)

\[\text{FAIL_ACTIV}\]

\(\text{ANY}\)

\[\begin{align*}
s \mapsto \text{prop}
\end{align*}\]

\(\text{WHERE}\)

\[\begin{align*}
\text{grd1: s} & \in \text{SERVICES not theorem} \\
\text{grd2: prop} & \subseteq \text{PEERS not theorem} \\
\text{grd3: inst_state(token_owner(s) \mapsto s)} & = \text{FAIL_DETECT}_4 \text{ not theorem} \\
\text{grd4: suspc_peers(token_owner(s) \mapsto s)} & \neq \emptyset \text{ not theorem} \\
\text{grd5: prop} & = \text{run_peers(s)} \setminus (\text{unav_peers} \cup \text{suspc_peers}) \text{ not theorem}
\end{align*}\]

\(\text{THEN}\)

\[\begin{align*}
\text{act1: inst_state} & = \text{inst_state} \setminus ((\text{prop} \times \{s\}) \times \{\text{FAIL_ACTIV}_4\}) \\
\text{act2: run_peers(s)} & = \text{run_peers(s)} \setminus \text{suspc_peers(token_owner(s) \mapsto s)} \\
\text{act3: failr_peers(s)} & = \text{failr_peers(s)} \cup \text{suspc_peers}
\end{align*}\]
act4: \( \text{suspc}_{\text{peers}}(\text{token}_{\text{owner}}(s) \mapsto s) = \emptyset \) 

END

FAIL_CONFIGURE:  not extended ordinary

REFINES

FAIL_CONFIGURE

ANY

s  

prop  

WHERE

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

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

grd3:  \( \text{inst}_{\text{state}}(\text{token}_{\text{owner}}(s) \mapsto s) = \text{FAIL}_{\text{ACTIV}}_{\text{4}} \) not theorem

grd4:  \( \text{card}((\text{run}_{\text{peers}}(s)) < \text{min}_{\text{inst}}(s) \) not theorem

grd5:  \( \text{prop} = \text{run}_{\text{peers}}(s) \setminus \text{unav}_{\text{peers}} \) not theorem

THEN

act1:  \( \text{inst}_{\text{state}} = \text{inst}_{\text{state}} \triangleright (\text{prop} \times \{s\} \times \{\text{FAIL}_{\text{CONFIG}}_{\text{4}}\} \) 

END

FAIL_IGNORE:  not extended ordinary

REFINES

FAIL_IGNORE

ANY

s  

prop  

WHERE

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

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

grd3:  \( \text{inst}_{\text{state}}(\text{token}_{\text{owner}}(s) \mapsto s) = \text{FAIL}_{\text{ACTIV}}_{\text{4}} \) not theorem

grd4:  \( \text{card}((\text{run}_{\text{peers}}(s)) \geq \text{min}_{\text{inst}}(s) \) not theorem

grd5:  \( \text{prop} = \text{run}_{\text{peers}}(s) \setminus \text{unav}_{\text{peers}} \) not theorem

THEN

act1:  \( \text{inst}_{\text{state}} = \text{inst}_{\text{state}} \triangleright (\text{prop} \times \{s\} \times \{\text{FAIL}_{\text{IGN}}_{\text{4}}\} \) 

END

IGNORE:  not extended ordinary

REFINES

IGNORE

ANY

s  

prop  

WHERE

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

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

grd3:  \( \text{inst}_{\text{state}}(\text{token}_{\text{owner}}(s) \mapsto s) = \text{FAIL}_{\text{IGN}}_{\text{4}} \) not theorem
**M16**

```
grd4: prop = run_peers(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_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: i ∉ actv_instc(token_owner(s) ↦ s) not 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_instc(s)) + card(run_peers(s)) < min_inst(s) not theorem >
THEN
act1: actv_instc(token_owner(s) ↦ s) = actv_instc(token_owner(s) ↦ s) ∪ \{i\} >
END

REDEPLOY_INSTS: not extended ordinary >
REFINES
REDEPLOY_INSTS
ANY
s >
WHERE
grd1: s ∈ SERVICES not theorem >
grd2: card(actv_instc(token_owner(s) ↦ s)) = deplo_inst(s) not theorem >
grd3: card(dep_instc(s)) + card(run_peers(s)) < min_inst(s) not theorem >
grd4: inst_state(token_owner(s) ↦ s) = FAIL_CONFIG_4 not theorem >
THEN
act1: dep_instc(s) = dep_instc(s) ∪ actv_instc(token_owner(s) ↦ s) >
act2: actv_instc(token_owner(s) ↦ s) = ∅ >
END
```
REDEPLOY: not extended ordinary

REDEPLOY

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\_CONFIG\_4} not theorem  >
grd4:  \text{actv\_instc}(\text{token\_owner}(s) \mapsto s) = \emptyset not theorem  >
grd5:  \text{dep\_instc}(s) \neq \emptyset not theorem  >
grd6:  |\text{run\_peers}(s)| + |\text{dep\_instc}(s)| \geq \text{min\_inst}(s) not theorem  >
grd7:  prop = \text{run\_peers}(s) \setminus \text{unav\_peers} not theorem  >

THEN

act1:  \text{inst\_state}\assign \text{inst\_state} \circ ((prop \times \{s\}) \times \{DPL\_4\})  >
act2:  \text{run\_peers}(s) = \text{run\_peers}(s) \cup \text{dep\_instc}(s) not theorem  >
act3:  \text{dep\_instc}(s) = \emptyset  >

END

HEAL: not extended ordinary

HEAL

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{DPL\_4} not theorem  >
grd4:  prop = \text{run\_peers}(s) \setminus \text{unav\_peers} not theorem  >

THEN

act1:  \text{inst\_state}\assign \text{inst\_state} \circ ((prop \times \{s\}) \times \{RUN\_4\})  >

END

UNFAIL\_PEER: extended ordinary

UNFAIL\_PEER

ANY

s  >

p  >

WHERE

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

THEN
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 M17
REFINES M16
SEES C09
VARIABLES
run_peers
suspct_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: suspct_peers ∈ (PEERS×SERVICES) ⇸ ℙ(PEERS) not theorem >
inv2: ∀ s · s ∈ SERVICES ⇒ token_owner(s) ↦ s ∈ dom(suspct_peers) not theorem >

> gluing_susp_1: ∀ s · s ∈ SERVICES ⇒ suspct_peers(token_owner(s) ↦ s) = suspct_peers(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 > Peers that will become unavailable
E > Values for token owner per service
WHERE

> grd1: prs ⊆ PEERS not theorem >
grd2: \( \text{prs} \not\subseteq \text{unav_peers} \) not theorem \( \rightarrow \) the peers in prs are not yet unavailable

\( \forall \text{srv} \cdot \text{srv} \in \text{SERVICES} \Rightarrow \text{dom}(\text{dom}(\text{inst_state}) \triangleright \{\text{srv}\}) \not= \emptyset \)

\( \text{prs} \not= \emptyset \) not theorem \( \rightarrow \) for each service srv, there must always be at least 1 peer available

grd4: \( E \in \text{SERVICES} \rightarrow \text{PEERS} \) not theorem \( \rightarrow \) Value for token owner per service

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

grd6: \( \forall \text{srv} \cdot \text{srv} \in \text{SERVICES} \land \text{token_owner}(\text{srv}) \in \text{prs} \Rightarrow E(\text{srv}) \in \text{run_peers}(\text{srv}) \setminus (\text{unav_peers} \cup \text{prs} \cup \text{failr_peers}(\text{srv}) \cup \text{suspct_peers}(\text{token_owner}(\text{srv}) \triangleright \text{srv})) \land E(\text{srv}) \triangleright \text{srv} \in \text{dom}(\text{inst_state}) \land E(\text{srv}) \triangleright \text{srv} \in \text{dom}(\text{suspct_peers}) \land \text{inst_state}(E(\text{srv}) \triangleright \text{srv}) = \text{inst_state}(\text{token_owner}(\text{srv}) \triangleright \text{srv}) \land \text{suspct_peers}(E(\text{srv}) \triangleright \text{srv}) = \text{suspct_peers}(\text{token_owner}(\text{srv}) \triangleright \text{srv}) \) not theorem \( \rightarrow \) 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

WITH

\( \text{p_s} = E \sim \langle \text{suspct_peers} \rangle \)

\( \text{rc_s} = ((\text{prs} \times \text{SERVICES}) \sim \text{rect_inst}) \sim ((E \setminus \text{token_owner}) \sim ) \times \emptyset) \)

\( \text{si} = ((\text{prs} \times \text{SERVICES}) \sim \text{suspc_inst}) \sim ((E \setminus \text{token_owner}) \sim ) \times \emptyset) \)

\( \text{rt_s} = ((\text{prs} \times \text{SERVICES}) \sim \text{rctt_inst}) \sim ((E \setminus \text{token_owner}) \sim ) \times \emptyset) \)

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

THEN

act1: \( \text{unav_peers} = \text{unav_peers} \cup \text{prs} \) \( \rightarrow \) the peers in prs become unavailable

act2: \( \text{token_owner} = \text{token_owner} \sim E \) \( \rightarrow \) new values for token owner per service

act3: \( \text{rect_inst} = ((\text{prs} \times \text{SERVICES}) \sim \text{rect_inst}) \sim ((E \setminus \text{token_owner}) \sim ) \times \emptyset) \) \( \rightarrow \) the peers in prs can not try to recontact instances

Page 2
M17

anymore (1)
act4: \text{rctt\_inst} = ((\text{prs} \times \text{SERVICES}) \rhd \text{rctt\_inst}) \rhd
((\text{E} \setminus \text{token\_owner}) \times \emptyset) \text{the peers in prs can not try to recontact instances}

anymore (2)
act5: \text{actv\_instc} = ((\text{prs} \times \text{SERVICES}) \rhd \text{actv\_instc}) \rhd
((\text{E} \setminus \text{token\_owner}) \times \emptyset) \text{the peers in prs can not activate instances anymore}
act6: \text{suspc\_peers} = ((\text{prs} \times \text{SERVICES}) \rhd \text{suspc\_peers}) \text{the peers in prs can not suspect instances anymore (1)}
act7: \text{suspc\_inst} = ((\text{prs} \times \text{SERVICES}) \rhd \text{suspc\_inst}) \rhd
((\text{E} \setminus \text{token\_owner}) \times \emptyset) \text{the peers in prs can not suspect instances anymore (2)}
act8: \text{inst\_state} = (\text{prs} \times \text{SERVICES}) \rhd \text{inst\_state} \text{the peers in prs can not monitor the state of the services provided anymore}

END

SUSPECT\_INST: \text{extended ordinary \rhd}
REFINES
\text{SUSPECT\_INST}
ANY
s \rightarrow \text{a service s}
susp \rightarrow \text{suspicious instances}
WHERE
grd1: s \in \text{SERVICES} \text{ not theorem \rhd}
grd2: susp \subseteq \text{PEERS} \text{ not theorem \rhd}
grd3: susp = \text{run\_peers(s) \cap unav\_peers} \text{ not theorem \rhd} \text{instances in susp are suspicious if the peers running them becomes unavailable}
grd4: \text{suspc\_inst(token\_owner(s) \rhd s)} = \emptyset \text{ not theorem \rhd \text{the member of susp have not yet been suspected for s by the token owner of s}}
grd5: \text{inst\_state(token\_owner(s) \rhd s)} = \text{RUN\_4} \text{ not theorem \rhd \text{the state of s is OK}}
grd6: susp \neq \emptyset \text{ not theorem \rhd}
THEN
act1: \text{suspc\_inst(token\_owner(s) \rhd s)} = susp \text{ the members of susp become suspected instances for s by the token owner of s}
END

FAIL: \text{not extended ordinary \rhd}
REFINES
FAIL
ANY
s \rightarrow
prop \rightarrow
WHERE
grd1: s \in \text{SERVICES} \text{ not theorem \rhd}
grd2: prop \subseteq \text{PEERS} \text{ not theorem \rhd}
grd3: \text{inst\_state(token\_owner(s) \rhd s)} = \text{RUN\_4} \text{ not theorem \rhd}
grd4: \text{suspc\_inst(token\_owner(s) \rhd s)} \neq \emptyset \text{ not theorem \rhd}
grd5: prop = \text{run\_peers(s) \setminus (suspc\_inst(token\_owner(s) \rhd s) \cup unav\_peers)} \text{ not theorem \rhd}
THEN

act1: \[ \text{inst}_\text{state} = \text{inst}_\text{state} \triangleright ((\text{prop}\times\{s\})\times\{\text{FAIL}_4\}) \]
act2: \[ \text{suspct}_\text{peers} = \text{suspct}_\text{peers} \triangleright ((\text{prop}\times\{s\})\times\{\text{suspct}_\text{inst}(\text{token}_\text{owner}(s) \mapsto s)\}) \]
act3: \[ \text{suspct}_\text{inst}(\text{token}_\text{owner}(s) \mapsto s) = \emptyset \]

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

grd1: \[ s \in \text{SERVICES} \text{ not theorem } \]
grd2: \[ i \in \text{PEERS} \text{ not theorem } \]
grd3: \[ \text{inst}_\text{state}(\text{token}_\text{owner}(s) \mapsto s) = \text{FAIL}_4 \text{ not theorem } \]

state of s is SUSPICIOUS

grd4: \[ \text{suspct}_\text{peers}(\text{token}_\text{owner}(s) \mapsto s) \neq \emptyset \text{ not theorem } \]
set of suspicious peers for s is not empty

grd5: \[ i \in \text{suspct}_\text{peers}(\text{token}_\text{owner}(s) \mapsto s) \setminus \text{unav}_\text{peers} \text{ not theorem } \]
i is a suspicious instance of s and is available (can be contacted)

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

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

grd7: \[ \text{rect}_\text{inst}(\text{token}_\text{owner}(s) \mapsto s) \subseteq \text{suspct}_\text{peers}(\text{token}_\text{owner}(s) \mapsto s) \text{ not theorem } \]

the token owner of s has not yet tried to recontact all the suspicious instances of s

THEN

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

i is recontacted by the token owner of s successfully

act2: \[ \text{rctt}_\text{inst}(\text{token}_\text{owner}(s) \mapsto s) = \text{rctt}_\text{inst}(\text{token}_\text{owner}(s) \mapsto s) \cup \{i\} \]

END

RECONTACT_INST_KO: not extended ordinary

REFINES
RECONTACT_INST_KO

ANY
s \rightarrow \text{a service } s
i \rightarrow \text{an instance } i

WHERE

grd1: \[ s \in \text{SERVICES} \text{ not theorem } \]
grd2: \[ i \in \text{PEERS} \text{ not theorem } \]
grd3: \[ \text{inst}_\text{state}(\text{token}_\text{owner}(s) \mapsto s) = \text{FAIL}_4 \text{ not theorem } \]

state of s is SUSPICIOUS

grd4: \[ \text{suspct}_\text{peers}(\text{token}_\text{owner}(s) \mapsto s) \neq \emptyset \text{ not theorem } \]
set of suspicious peers for s is not empty

grd5: \[ i \in \text{suspct}_\text{peers}(\text{token}_\text{owner}(s) \mapsto s) \setminus \text{unav}_\text{peers} \text{ not theorem } \]
theorem \( i \) is a suspicious instance of \( s \) and \( i \) is unavailable (can not be contacted)

\[ \text{grd6:} \quad i \notin \text{rect\_inst(token\_owner(s) ↦ s)} \] not theorem \( \rightarrow \) the token owner of \( s \) has not yet tried to recontact \( i \)

\[ \text{grd7:} \quad \text{rect\_inst(token\_owner(s) ↦ s)} \subseteq \text{suspct\_peers(token\_owner(s) ↦ s)} \] not theorem \( \rightarrow \) the token owner of \( s \) has not yet tried to recontact all the suspicious instances of \( s \)

THEN

act1: \( \text{rect\_inst(token\_owner(s) ↦ s)} = \text{rect\_inst(token\_owner(s) ↦ s)} \cup \{i\} \rightarrow \) the token owner of \( s \) has tried to recontact \( i \)

END

FAIL\_DETECT: \( \not \) extended ordinary

REFINES FAIL\_DETECT

ANY

\( s \rightarrow \)

prop \( \rightarrow \)

susp \( \rightarrow \)

WHERE

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

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

\[ \text{grd7:} \quad \text{susp} \subseteq \text{PEERS} \] not theorem \( \rightarrow \)

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

\[ \text{grd4:} \quad \text{suspct\_peers(token\_owner(s) ↦ s)} \neq \emptyset \] not theorem \( \rightarrow \)

\[ \text{grd5:} \quad \text{rect\_inst(token\_owner(s) ↦ s)} = \text{suspct\_peers(token\_owner(s) ↦ s)} \] not theorem \( \rightarrow \)

\[ \text{grd6:} \quad \text{prop} = \left(\left(\text{run\_peers(s)} \setminus \text{suspct\_peers(token\_owner(s) ↦ s)}\right) \cup \text{rect\_inst(token\_owner(s) ↦ s)}) \setminus \text{unav\_peers} \right) \] not theorem \( \rightarrow \)

THEN

act1: \( \text{inst\_state} = \text{inst\_state} \setminus \left(\left(\text{prop} \times \{s\}\right) \times \{\text{FAIL\_DETECT}_4\}\right) \rightarrow \)

act2: \( \text{suspct\_peers} = \text{suspct\_peers} \setminus \left(\left(\text{prop} \times \{s\}\right) \times \{\text{susp}\}\right) \rightarrow \)

act3: \( \text{rect\_inst(token\_owner(s) ↦ s)} = \emptyset \rightarrow \)

act4: \( \text{rect\_inst(token\_owner(s) ↦ s)} = \emptyset \rightarrow \)

END

IS\_OK: \( \not \) extended ordinary

REFINES IS\_OK

ANY

\( s \rightarrow \)

prop \( \rightarrow \)

WHERE

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

\[ \text{grd2:} \quad \text{prop} \subseteq \text{PEERS} \] not theorem \( \rightarrow \)
M17

\[ \text{grd3: } \text{inst\_state(token\_owner(s) ↦ s)} = \text{FAIL\_DETECT\_4} \text{ not theorem} \]

\[ \text{grd4: } \text{suspct\_peers(token\_owner(s) ↦ s)} = \emptyset \text{ not theorem} \]
\[ \text{grd5: } \text{prop} = \text{run\_peers(s)} \setminus \text{unav\_peers} \text{ not theorem} \]

\[ \text{THEN} \]
\[ \text{act1: } \text{inst\_state ≔ inst\_state × ((prop×\{s\})×\{RUN\_4\})} \]
\[ \text{END} \]

\[ \text{FAIL\_ACTIV: } \text{not extended ordinary} \]

\[ \text{REFINES} \]

\[ \text{FAIL\_ACTIV} \]

\[ \text{ANY} \]
\[ s \]
\[ \text{prop} \]

\[ \text{WHERE} \]
\[ \text{grd1: } s \in \text{SERVICES} \text{ not theorem} \]
\[ \text{grd2: } \text{prop} \subseteq \text{PEERS} \text{ not theorem} \]
\[ \text{grd3: } \text{inst\_state(token\_owner(s) ↦ s)} = \text{FAIL\_DETECT\_4} \text{ not theorem} \]

\[ \text{THEN} \]
\[ \text{grd4: } \text{suspct\_peers(token\_owner(s) ↦ s)} \neq \emptyset \text{ not theorem} \]
\[ \text{grd5: } \text{prop} = \text{run\_peers(s)} \setminus (\text{unav\_peers} \cup \text{suspct\_peers (token\_owner(s) ↦ s)}) \text{ not theorem} \]

\[ \text{THEN} \]
\[ \text{act1: } \text{inst\_state ≔ inst\_state × ((prop×\{s\})×\{FAIL\_ACTIV\_4\})} \]
\[ \text{act2: } \text{run\_peers(s) ≔ run\_peers(s) \setminus \text{suspct\_peers(token\_owner(s) ↦ s)}} \]
\[ \text{act3: } \text{failr\_peers(s) ≔ failr\_peers(s) \cup \text{suspct\_peers (token\_owner(s) ↦ s)}} \]
\[ \text{act4: } \text{suspct\_peers ≔ suspct\_peers × ((prop×\{s\})×\{∅\})} \]
\[ \text{END} \]

\[ \text{FAIL\_CONFIGURE: } \text{extended ordinary} \]

\[ \text{REFINES} \]

\[ \text{FAIL\_CONFIGURE} \]

\[ \text{ANY} \]
\[ s \]
\[ \text{prop} \]

\[ \text{WHERE} \]
\[ \text{grd1: } s \in \text{SERVICES} \text{ not theorem} \]
\[ \text{grd2: } \text{prop} \subseteq \text{PEERS} \text{ not theorem} \]
\[ \text{grd3: } \text{inst\_state(token\_owner(s) ↦ s)} = \text{FAIL\_ACTIV\_4} \text{ not theorem} \]

\[ \text{THEN} \]
\[ \text{grd4: } \text{card(run\_peers(s))} < \text{min\_inst(s)} \text{ not theorem} \]
\[ \text{grd5: } \text{prop} = \text{run\_peers(s)} \setminus \text{unav\_peers} \text{ not theorem} \]

\[ \text{THEN} \]
\[ \text{act1: } \text{inst\_state ≔ inst\_state × ((prop×\{s\})×\{FAIL\_CONFIG\_4\})} \]
\[ \text{END} \]

\[ \text{FAIL\_ACTIV: } \text{extended ordinary} \]

\[ \text{REFINES} \]

\[ \text{FAIL\_ACTIV} \]

\[ \text{ANY} \]
\[ s \]
\[ \text{prop} \]

\[ \text{WHERE} \]
\[ \text{grd1: } s \in \text{SERVICES} \text{ not theorem} \]
\[ \text{grd2: } \text{prop} \subseteq \text{PEERS} \text{ not theorem} \]
\[ \text{grd3: } \text{inst\_state(token\_owner(s) ↦ s)} = \text{FAIL\_ACTIV\_4} \text{ not theorem} \]

\[ \text{THEN} \]
\[ \text{grd4: } \text{card(run\_peers(s))} < \text{min\_inst(s)} \text{ not theorem} \]
\[ \text{grd5: } \text{prop} = \text{run\_peers(s)} \setminus \text{unav\_peers} \text{ not theorem} \]

\[ \text{THEN} \]
\[ \text{act1: } \text{inst\_state ≔ inst\_state × ((prop×\{s\})×\{FAIL\_ACTIV\_4\})} \]
\[ \text{END} \]

Page 6
\textbf{FAIL\_IGNORE:} extended ordinary \>
\textbf{REFINES} FAIL\_IGNORE \>
\textbf{ANY} \>
\>
s \>
prop \>
\textbf{WHERE} \>

\quad \textbf{grd1:} s \in \text{SERVICES} \text{ not theorem} \>
\quad \textbf{grd2:} \text{prop} \subseteq \text{PEERS} \text{ not theorem} \>
\quad \textbf{grd3:} \text{inst\_state}(\text{token\_owner(s) } \mapsto s) = \text{FAIL\_ACTIV\_4} \text{ not theorem} \>

\quad \text{card(\text{run\_peers(s)})} \geq \text{min\_inst(s)} \text{ not theorem} \>
\quad \text{prop} = \text{run\_peers(s)}\setminus \text{unav\_peers} \text{ not theorem} \>
\textbf{THEN} \>

\quad \text{act1: inst\_state} = \text{inst\_state} \setminus ((\text{prop}\times\{s\})\times\{\text{FAIL\_IGN\_4}\}) \>
\textbf{END} \>

\textbf{IGNORE:} extended ordinary \>
\textbf{REFINES} IGNORE \>
\textbf{ANY} \>
\>
s \>
prop \>
\textbf{WHERE} \>

\quad \textbf{grd1:} s \in \text{SERVICES} \text{ not theorem} \>
\quad \textbf{grd2:} \text{prop} \subseteq \text{PEERS} \text{ not theorem} \>
\quad \textbf{grd3:} \text{inst\_state}(\text{token\_owner(s) } \mapsto s) = \text{FAIL\_IGN\_4} \text{ not theorem} \>

\quad \text{prop} = \text{run\_peers(s)}\setminus \text{unav\_peers} \text{ not theorem} \>
\textbf{THEN} \>

\quad \text{act1: inst\_state} = \text{inst\_state} \setminus ((\text{prop}\times\{s\})\times\{\text{RUN\_4}\}) \>
\textbf{END} \>

\textbf{REDEPLOY\_INSTC:} extended ordinary \>
\textbf{REFINES} REDEPLOY\_INSTC \>
\textbf{ANY} \>
\>
s \rightarrow a \text{ service s} \>
\>
i \rightarrow a \text{ instance i} \>
\textbf{WHERE} \>

\quad \textbf{grd1:} s \in \text{SERVICES} \text{ not theorem} \>
\quad \textbf{grd2:} i \in \text{PEERS} \text{ not theorem} \>
\quad \textbf{grd3:} i \notin \text{run\_peers(s)} \cup \text{failr\_peers(s)} \cup \text{unav\_peers} \cup \text{dep\_instc(s)} \text{ not theorem} \>
\quad \text{\textit{i does not run s, is not failed for s, is not unavailable and is not already activated for s}} \>
\quad \textbf{grd4:} i \notin \text{actv\_instc(token\_owner(s) } \mapsto s) \text{ not theorem} \>
\quad \textbf{grd5:} \text{inst\_state}(\text{token\_owner(s) } \mapsto s) = \text{FAIL\_CONFIG\_4} \text{ not theorem} \>
M17

\[ \text{grd6: card(\text{actv\_instc(token\_owner(s) \mapsto s)) < deplo\_inst(s) \not{\text{theorem}}} \]
\[ \text{not theorem} \]
\[ \text{THEN} \]
\[ \text{act1: actv\_instc(token\_owner(s) \mapsto s) = actv\_instc(token\_owner(s) \mapsto s) \cup \{i\}} \]
\[ \text{END} \]

\text{REDEPLOY\_INSTS: extended ordinary} \]
\text{REFINES}
\text{REDEPLOY\_INSTS}
\text{ANY}
\text{s} \]
\text{WHERE}
\[ \text{grd1: s \in SERVICES \not{\text{theorem}}} \]
\[ \text{grd2: card(\text{actv\_instc(token\_owner(s) \mapsto s)) = deplo\_inst(s) \not{\text{theorem}}} \]
\[ \text{THEN} \]
\[ \text{act1: dep\_instc(s) = dep\_instc(s) \cup actv\_instc(token\_owner(s) \mapsto s)} \]
\[ \text{END} \]

\text{REDEPLOY: extended ordinary} \]
\text{REFINES}
\text{REDEPLOY}
\text{ANY}
\text{s} \]
\text{prop} \]
\text{WHERE}
\[ \text{grd1: s \in SERVICES \not{\text{theorem}}} \]
\[ \text{grd2: prop \in PEERS \not{\text{theorem}}} \]
\[ \text{grd3: inst\_state(token\_owner(s) \mapsto s) = FAIL\_CONFIG\_4 \not{\text{theorem}}} \]
\[ \text{THEN} \]
\[ \text{act1: actv\_instc(token\_owner(s) \mapsto s) = \emptyset} \]
\[ \text{act2: dep\_instc(s) = \emptyset} \]
\[ \text{act3: run\_peers(s) = run\_peers(s) \cup dep\_instc(s)} \]

Page 8
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_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  
  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: run_inst ∈ (PEERS×SERVICES) ↦ ℙ(PEERS) not theorem
  inv2: ∀ s ∈ SERVICES ⇒ token_owner(s) ↦ s ∈ dom(run_inst) not theorem
  gluing_run_1: ∀ 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 = 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: not extended ordinary
  REFINES MAKE_PEER_UNAVAIL
  ANY
  prs  
  E  
  WHERE
  grd1: prs ⊆ PEERS not theorem
M18

\( \text{grd2: } \text{prs} \not\in \text{unav_peers} \) not theorem \( \rightarrow \) the peers in prs are not yet unavailable

\( \text{grd3: } \forall \text{srv} \cdot \text{srv} \in \text{SERVICES} \implies \text{dom(dom(inst_state))} \supset \{ \text{srv} \} \)
\( \\text{prs} \neq \emptyset \) not theorem \( \rightarrow \) for each service srv, there must always be at least 1 peer available

\( \text{grd4: } E \in \text{SERVICES} \rightarrow \text{PEERS} \) not theorem \( \rightarrow \) Value for token owner per service

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

\( \text{grd6: } \forall \text{srv} \cdot \text{srv} \in \text{SERVICES} \land \text{token_owner(srv)} \in \text{prs} \implies \)
\( E(srv) \in \text{run_inst(token_owner(srv) ↦ srv)} \setminus \text{(unav_peers u prs u failr_peers(srv) u suspct_peers(token_owner(srv) ↦ srv))} \land \)
\( \text{run_inst(E(srv) ↦ srv) = run_inst(token_owner(srv) ↦ srv)} \land \)
\( \text{inst_state(E(srv) ↦ srv) = inst_state(token_owner(srv) ↦ srv)} \land \)
\( \text{suspct_peers(E(srv) ↦ srv) = suspct_peers(token_owner(srv) ↦ srv)} \) not theorem \( \rightarrow \) 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

\( \text{THEN} \)

\( \text{act1: } \text{unav_peers} = \text{unav_peers} \cup \text{prs} \) \( \rightarrow \) the peers in prs become unavailable

\( \text{act2: } \text{token_owner} = \text{token_owner} \circ E \) \( \rightarrow \) new values for token owner per service

\( \text{act3: } \text{rect_inst} = (\text{((prs×SERVICES) ↦ rect_inst)} \circ \)
\( (((E\text{⋯token_owner)~})×\{\emptyset}) \) \( \rightarrow \) the peers in prs can not try to recontact instances anymore (1)

\( \text{act4: } \text{rctt_inst} = (\text{((prs×SERVICES) ↦ rctt_inst)} \circ \)
\( (((E\text{⋯token_owner)~})×\{\emptyset}) \) \( \rightarrow \) the peers in prs can not try to recontact instances anymore (2)

\( \text{act5: } \text{actv_instc} = (\text{((prs×SERVICES) ↦ actv_instc)} \circ \)
\( (((E\text{⋯token_owner)~})×\{\emptyset}) \) \( \rightarrow \) the peers in prs can not activate instances anymore (1)

\( \text{act6: } \text{suspc_inst} = (\text{((prs×SERVICES) ↦ suspc_inst)} \circ \)
\( (((E\text{⋯token_owner)~})×\{\emptyset}) \) \( \rightarrow \) the peers in prs can not suspect instances anymore (1)
M18

\[(((E\{\text{token\_owner}\})\setminus\{\emptyset\}) \times \{\emptyset\}) \triangleright \text{the peers in prs can not suspect instances anymore (2)}\]

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

act9: \[\text{run\_inst} = (prs \times \text{SERVICES}) < \text{run\_inst} \]

END

\textbf{SUSPECT\_INST}: not extended ordinary

\textbf{REFINES}

\textbf{SUSPECT\_INST}

\textbf{ANY}

\begin{itemize}
  \item \(s\) \triangleright \text{a service } s
  \item \(\text{susp}\) \triangleright \text{suspicious instances}
\end{itemize}

\textbf{WHERE}

\begin{itemize}
  \item \(\text{grd1}: s \in \text{SERVICES}\) not theorem
  \item \(\text{grd2}: \text{susp} \subseteq \text{PEERS}\) not theorem
  \item \(\text{grd3}: \text{susp} = \text{run\_inst}(\text{token\_owner}(s) \mapsto s) \cap \text{unav\_peers}\) not theorem
\end{itemize}

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

\begin{itemize}
  \item \(\text{grd4}: \text{suspc\_inst}(\text{token\_owner}(s) \mapsto s) = \emptyset\) not theorem
  \item \(\text{grd5}: \text{inst\_state}(\text{token\_owner}(s) \mapsto s) = \text{RUN\_4}\) not theorem
\end{itemize}

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

\begin{itemize}
  \item \(\text{grd6}: \text{susp} \neq \emptyset\) not theorem
  \item \(\text{THEN}\)
  \item \(\text{act1}: \text{suspc\_inst}(\text{token\_owner}(s) \mapsto s) = \text{susp}\) \triangleright \text{the members of susp become suspected instances for } s \text{ by the token owner of } s\)
\end{itemize}

END

\textbf{FAIL}: not extended ordinary

\textbf{REFINES}

\textbf{FAIL}

\textbf{ANY}

\begin{itemize}
  \item \(s\) \triangleright \(\)\)
  \item \(\text{prop}\) \triangleright \(\)
\end{itemize}

\textbf{WHERE}

\begin{itemize}
  \item \(\text{grd1}: s \in \text{SERVICES}\) not theorem
  \item \(\text{grd2}: \text{prop} \subseteq \text{PEERS}\) not theorem
  \item \(\text{grd3}: \text{inst\_state}(\text{token\_owner}(s) \mapsto s) = \text{RUN\_4}\) not theorem
  \item \(\text{grd4}: \text{suspc\_inst}(\text{token\_owner}(s) \mapsto s) \neq \emptyset\) not theorem
  \item \(\text{grd5}: \text{prop} = \text{run\_inst}(\text{token\_owner}(s) \mapsto s) \setminus (\text{suspc\_inst}(\text{token\_owner}(s) \mapsto s) \cup \text{unav\_peers})\) not theorem
\end{itemize}

\textbf{(token\_owner}(s) \mapsto s) \cup \text{unav\_peers))\) not theorem

\textbf{THEN}

\begin{itemize}
  \item \(\text{act1}: \text{inst\_state} = \text{inst\_state} \odot ((\text{prop} \times \{s\}) \times \{\text{FAIL\_4}\})\)
  \item \(\text{act2}: \text{suspc\_peers} = \text{suspc\_peers} \odot ((\text{prop} \times \{s\}) \times \{\text{suspc\_inst}(\text{token\_owner}(s) \mapsto s)\})\)
  \item \(\text{act3}: \text{suspc\_inst}(\text{token\_owner}(s) \mapsto s) = \emptyset\)
\end{itemize}

END

Page 3
RECONTACT_INST_OK: extended ordinary
REFINES
  RECONTACT_INST_OK
ANY
  s \rightarrow \text{a service } s
  i \rightarrow \text{an instance } i
WHERE
  grd1: s \in \text{SERVICES} \not\text{ theorem } \rightarrow \text{the state of } s \text{ is SUSPICIOUS}
  grd2: i \in \text{PEERS} \not\text{ theorem } \rightarrow \text{the set of suspicious peers for } s \text{ is not empty}
  grd3: \text{inst_state}(\text{token_owner}(s) \mapsto s) = \text{FAIL}_4 \not\text{ theorem } \rightarrow \text{the token owner of } s \text{ has not yet tried to recontact } i
  grd4: \text{suspct_peers}(\text{token_owner}(s) \mapsto s) \neq \emptyset \not\text{ theorem } \rightarrow \text{the set of suspicious peers for } s \text{ is not empty}
  grd5: i \in \text{suspct_peers}(\text{token_owner}(s) \mapsto s) \setminus \text{unav_peers} \not\text{ theorem } \rightarrow \text{i is a suspicious instance of } s \text{ and is unavailable (can not be contacted)}
  grd6: i \notin \text{rect_inst}(\text{token_owner}(s) \mapsto s) \not\text{ theorem } \rightarrow \text{the token owner of } s \text{ has not yet tried to recontact } i
  grd7: \text{rect_inst}(\text{token_owner}(s) \mapsto s) \subset \text{suspct_peers}(\text{token_owner}(s) \mapsto s) \not\text{ theorem } \rightarrow \text{the token owner of } s \text{ 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\} \rightarrow \text{the token owner of } s \text{ has tried to recontact } i
  act2: \text{rctt_inst}(\text{token_owner}(s) \mapsto s) = \text{rctt_inst}(\text{token_owner}(s) \mapsto s) \cup \{i\} \rightarrow \text{i is recontacted by the token owner of } s \text{ successfully}
END

RECONTACT_INST_KO: extended ordinary
REFINES
  RECONTACT_INST_KO
ANY
  s \rightarrow \text{a service } s
  i \rightarrow \text{an instance } i
WHERE
  grd1: s \in \text{SERVICES} \not\text{ theorem } \rightarrow \text{the state of } s \text{ is SUSPICIOUS}
  grd2: i \in \text{PEERS} \not\text{ theorem } \rightarrow \text{the set of suspicious peers for } s \text{ is not empty}
  grd3: \text{inst_state}(\text{token_owner}(s) \mapsto s) = \text{FAIL}_4 \not\text{ theorem } \rightarrow \text{the token owner of } s \text{ has not yet tried to recontact } i
  grd4: \text{suspct_peers}(\text{token_owner}(s) \mapsto s) \neq \emptyset \not\text{ theorem } \rightarrow \text{the set of suspicious peers for } s \text{ is not empty}
  grd5: i \in \text{suspct_peers}(\text{token_owner}(s) \mapsto s) \setminus \text{unav_peers} \not\text{ theorem } \rightarrow \text{i is a suspicious instance of } s \text{ and is unavailable (can not be contacted)}
  grd6: i \notin \text{rect_inst}(\text{token_owner}(s) \mapsto s) \not\text{ theorem } \rightarrow \text{the token owner of } s \text{ has not yet tried to recontact } i
  grd7: \text{rect_inst}(\text{token_owner}(s) \mapsto s) \subset \text{suspct_peers}(\text{token_owner}(s) \mapsto s) \not\text{ theorem } \rightarrow \text{the token owner of } s \text{ has not yet tried to recontact all the suspicious instances of } s
THEN
act1: \text{rect\_inst}(\text{token\_owner(s)} \rightarrow s) := \text{rect\_inst}(\text{token\_owner(s)} \rightarrow s) \cup \{i\} \quad \text{the token owner of s has tried to recontact i}
END

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

\text{ANY} \\
\quad s \quad > \\
\quad \text{prop} \quad > \\
\quad \text{susp} \quad > \\
\text{WHERE} \\
\quad \text{grd1: } s \in \text{SERVICES} \quad \text{not theorem} \\
\quad \text{grd2: } \text{prop} \subseteq \text{PEERS} \quad \text{not theorem} \\
\quad \text{grd7: } \text{susp} \subseteq \text{PEERS} \quad \text{not theorem} \\
\quad \text{grd3: } \text{inst\_state}(\text{token\_owner}(s) \rightarrow s) = \text{FAIL\_4} \quad \text{not theorem} \\
\quad \text{grd4: } \text{suspct\_peers}(\text{token\_owner}(s) \rightarrow s) \neq \emptyset \quad \text{not theorem} \\
\quad \text{grd5: } \text{rect\_inst}(\text{token\_owner}(s) \rightarrow s) = \text{suspct\_peers}(\text{token\_owner}(s) \rightarrow s) \quad \text{not theorem} \\
\quad \text{(s} \rightarrow s) \quad \text{not theorem} \\
\quad \text{grd6: } \text{prop} = ((\text{run\_inst}(\text{token\_owner}(s) \rightarrow s) \setminus \text{suspct\_peers}(\text{token\_owner}(s) \rightarrow s)) \cup \text{unav\_peers} \quad \text{not theorem} \\
\quad \text{grd8: } \text{susp} = \text{suspct\_peers}(\text{token\_owner}(s) \rightarrow s) \setminus \text{rctt\_inst}(\text{token\_owner}(s) \rightarrow s) \quad \text{not theorem} \\
\text{THEN} \\
\quad \text{act1: } \text{inst\_state} = \text{inst\_state} \setminus ((\text{prop} \times \{s\}) \times \{\text{FAIL\_DETECT\_4}\}) \\
\quad \text{act2: } \text{suspct\_peers} = \text{suspct\_peers} \setminus ((\text{prop} \times \{s\}) \times \{\text{susp}\}) \\
\quad \text{act3: } \text{rect\_inst}(\text{token\_owner}(s) \rightarrow s) = \emptyset \\
\quad \text{act4: } \text{rctt\_inst}(\text{token\_owner}(s) \rightarrow s) = \emptyset \\
\text{END}

\text{IS\_OK:} \quad \text{not extended ordinary} \\
\text{REFINES} \quad \text{IS\_OK}

\text{ANY} \\
\quad s \quad > \\
\quad \text{prop} \quad > \\
\text{WHERE} \\
\quad \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}(\text{token\_owner}(s) \rightarrow s) = \text{FAIL\_DETECT\_4} \quad \text{not theorem} \\
\quad \text{grd4: } \text{suspct\_peers}(\text{token\_owner}(s) \rightarrow s) = \emptyset \quad \text{not theorem} \\
\quad \text{grd5: } \text{prop} = \text{run\_inst}(\text{token\_owner}(s) \rightarrow s) \setminus \text{unav\_peers} \quad \text{not theorem} \\
\text{THEN} \\
\quad \text{act1: } \text{inst\_state} = \text{inst\_state} \setminus ((\text{prop} \times \{s\}) \times \{\text{RUN\_4}\}) \\

Page 5
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_inst ≔ run_inst  \((prop × \{s\}) × (run_inst(token_owner(s) ↦ s) \setminus \text{suspct_peers(token_owner(s) ↦ s)})\)
act3: failr_peers(s) ≔ failr_peers(s) ∪ suspct_peers(token_owner(s) ↦ s)
act4: suspct_peers ≔ suspct_peers  \((prop × \{s\}) × \{∅\})

END

FAIL_CONFIGURE: not 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_DETECT_4 not theorem

THEN

act1: inst_state = inst_state  \((prop × \{s\}) × \{FAIL_CONFIG_4\})

END

FAIL_IGNORE: not extended ordinary

REFINES
FAIL_IGNORE

ANY
\[
\begin{align*}
\text{s} &\quad \text{prop} \\
\text{WHERE} \\
\text{grd1: } &\quad s \in \text{SERVICES} \quad \text{not theorem} \\
\text{grd2: } &\quad \text{prop} \subseteq \text{PEERS} \quad \text{not theorem} \\
\text{grd3: } &\quad \text{inst\_state}(\text{token\_owner}(s) \rightarrow s) = \text{FAIL\_ACTIV\_4} \quad \text{not theorem} \\
\text{grd4: } &\quad \text{card}(\text{run\_inst}(\text{token\_owner}(s) \rightarrow s)) \geq \text{min\_inst}(s) \quad \text{not theorem} \\
\text{grd5: } &\quad \text{prop} = \text{run\_inst}(\text{token\_owner}(s) \rightarrow s) \setminus \text{unav\_peers} \quad \text{not theorem} \\
\text{THEN} \\
\text{act1: } &\quad \text{inst\_state} = \text{inst\_state} \bowtie ((\text{prop} \times \{s\}) \times \{\text{FAIL\_IGN\_4}\}) \\
\text{END} \\
\text{IGNORE: } &\quad \text{not extended ordinary} \\
\text{REFINES} \\
\quad \text{IGNORE} \\
\quad \text{ANY} \\
\quad s &\quad \text{prop} \\
\text{WHERE} \\
\quad \text{grd1: } &\quad s \in \text{SERVICES} \quad \text{not theorem} \\
\quad \text{grd2: } &\quad \text{prop} \subseteq \text{PEERS} \quad \text{not theorem} \\
\quad \text{grd3: } &\quad \text{inst\_state}(\text{token\_owner}(s) \rightarrow s) = \text{FAIL\_IGN\_4} \quad \text{not theorem} \\
\quad \text{grd4: } &\quad \text{prop} = \text{run\_inst}(\text{token\_owner}(s) \rightarrow s) \setminus \text{unav\_peers} \quad \text{not theorem} \\
\quad \text{THEN} \\
\quad \text{act1: } &\quad \text{inst\_state} = \text{inst\_state} \bowtie ((\text{prop} \times \{s\}) \times \{\text{RUN\_4}\}) \\
\quad \text{END} \\
\text{REDEPLOY\_INSTC: } &\quad \text{not extended ordinary} \\
\text{REFINES} \\
\quad \text{REDEPLOY\_INSTC} \\
\quad \text{ANY} \\
\quad s &\quad \text{a service s} \\
\quad i &\quad \text{an instance i} \\
\text{WHERE} \\
\quad \text{grd1: } &\quad s \in \text{SERVICES} \quad \text{not theorem} \\
\quad \text{grd2: } &\quad i \in \text{PEERS} \quad \text{not theorem} \\
\quad \text{grd3: } &\quad i \not\in \text{run\_inst}(\text{token\_owner}(s) \rightarrow s) \cup \text{failr\_peers}(s) \cup \text{unav\_peers} \cup \text{deplo\_instc}(s) \quad \text{not theorem} \\
\quad \text{grd4: } &\quad i \not\in \text{actv\_instc}(\text{token\_owner}(s) \rightarrow s) \quad \text{not theorem} \\
\quad \text{grd5: } &\quad \text{inst\_state}(\text{token\_owner}(s) \rightarrow s) = \text{FAIL\_CONFIG\_4} \quad \text{not theorem} \\
\quad \text{grd6: } &\quad \text{card}(\text{actv\_instc}(\text{token\_owner}(s) \rightarrow s)) < \text{deplo\_instc}(s) \quad \text{not theorem} \\
\end{align*}
\]
M18

\begin{align*}
\text{theorem} & > \\
\text{grd7: card(dep-instc(s)) + card(run-inst(token-owner(s) \to s))} & < \text{min-inst(s)} \not\text{not theorem} > \\
\text{THEN} & > \\
\text{act1: actv-instc(token-owner(s) \to s) = actv-instc(token-owner(s) \to s) \cup \{i\}} & > \\
\text{END} & > \\
\text{REDEPLOY INSTS: not extended ordinary >}
\text{REFINES REDEPLOY INSTS}
\text{ANY} & > \\
\text{s} & > \\
\text{WHERE} & > \\
\text{grd1: s \in SERVICES not theorem} & > \\
\text{grd2: card(actv-instc(token-owner(s) \to s)) = deplo-inst(s)} & \not\text{not theorem} > \\
\text{THEN} & > \\
\text{grd3: card(dep-instc(s)) + card(run-inst(token-owner(s) \to s))} & < \text{min-inst(s)} \not\text{not theorem} > \\
\text{grd4: inst-state(token-owner(s) \to s) = FAIL_CONFIG_4 not}\ & \text{theorem} > \\
\text{THEN} & > \\
\text{act1: dep-instc(s) = dep-instc(s) \cup actv-instc(token-owner(s) \to s)} & > \\
\text{act2: actv-instc(token-owner(s) \to s) = \emptyset} & > \\
\text{END} & > \\
\text{REDEPLOY: not extended ordinary >}
\text{REFINES REDEPLOY}
\text{ANY} & > \\
\text{s} & > \\
\text{prop} & > \\
\text{WHERE} & > \\
\text{grd1: s \in SERVICES not theorem} & > \\
\text{grd2: prop \subseteq PEERS not theorem} & > \\
\text{grd3: inst-state(token-owner(s) \to s) = FAIL_CONFIG_4 not}\ & \text{theorem} > \\
\text{grd4: actv-instc(token-owner(s) \to s) = \emptyset not theorem} & > \\
\text{grd5: dep-instc(s) \neq \emptyset not theorem} & > \\
\text{grd6: card(run-inst(token-owner(s) \to s)) + card(dep-instc(s)) \geq}\ & \text{min-inst(s)} \not\text{not theorem} > \\
\text{grd7: prop = run-inst(token-owner(s) \to s) \setminus unav_peers not}\ & \text{theorem} > \\
\text{THEN} & > \\
\text{act1: inst-state = inst-state \setminus ((prop \times \{s\}) \times \{DPL_4\})} & > \\
\text{act2: run-inst = run-inst \setminus ((prop \times \{s\}) \times \{run-inst(token-owner(s) \to s) \cup dep-instc(s)\})} & >
\end{align*}
act3: \[
\text{dep_instc}(s) = \emptyset
\]

M18

HEAL: not extended ordinary
REFINES
ANY
s
prop
WHERE
grd1: s \in \text{SERVICES} \text{ not theorem} \\
grd2: prop \subseteq \text{PEERS} \text{ not theorem} \\
grd3: \text{inst_state}(\text{token_owner}(s) \rightarrow s) = \text{DPL}_4 \text{ not theorem} \\
grd4: \text{prop} = \text{run_inst}(\text{token_owner}(s) \rightarrow s)\backslash\text{unav_peers} \text{ not theorem} \\
THEN
act1: \text{inst_state} = \text{inst_state} \leftarrow ((\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} \text{ not theorem} \\
grd2: p \in \text{PEERS} \text{ not theorem} \\
grd3: p \in \text{failr_peers}(s) \text{ not theorem} \\
THEN
act1: \text{failr_peers}(s) = \text{failr_peers}(s)\backslash\{p\}
END

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

END
MACHINE M19
REFINES M18
SEES C09
VARIABLES
  run_inst
  suspct_peers
  failr_inst
  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 owns
  inst_state

INVARIANTS
  inv1: failr_inst Œ (PEERS×SERVICES) ↷ ℙ(PEERS) not theorem >
  inv2: \> ∀ s · s Œ SERVICES ⇒ token_owner(s) ↷ s Œ dom(failr_inst) not theorem >
  gluing_fail_1: \> ∀ s · s Œ SERVICES ⇒ failr_inst(token_owner(s) ↷ s) = failr_peers(s) not theorem >

EVENTS

  INITIALISATION: not extended ordinary >
  THEN
  act1: run_inst ≔ InitRunPeers >
  act2: suspct_peers ≔ InitSuspPrs >
  act3: failr_inst ≔ InitSuspPeers >
  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 Œ PEERS not theorem >
M19

grd2: \( \text{prs} \not\in \text{unav_peers} \) not theorem \( \Rightarrow \) the peers in prs are not yet unavailable

\( \forall \text{srv} \cdot \text{srv} \in \text{SERVICES} \Rightarrow \text{dom}(\text{dom}(\text{inst_state}) \setminus \{\text{srv}\}) \)

\( \\text{\( \setminus \)} \text{prs} \not= \emptyset \) not theorem \( \Rightarrow \) for each service srv, there must always be at least 1 peer available

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

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

grd6: \( \forall \text{srv} \cdot \text{srv} \in \text{SERVICES} \wedge \text{token_owner(srv)} \in \text{prs} \Rightarrow \)

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: \( \text{unav_peers} = \text{unav_peers} \cup \text{prs} \) \( \Rightarrow \) the peers in prs become unavailable

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

act3: \( \text{rect_inst} = (\{\text{prs} \times \text{SERVICES}\} \setminus \text{rect_inst}) \setminus (\{(\text{E} \setminus \text{token_owner}) \times \{\emptyset\}\}) \) \( \Rightarrow \) the peers in prs can not try to recontact instances anymore (1)

act4: \( \text{rctt_inst} = (\{\text{prs} \times \text{SERVICES}\} \setminus \text{rctt_inst}) \setminus (\{(\text{E} \setminus \text{token_owner}) \times \{\emptyset\}\}) \) \( \Rightarrow \) the peers in prs can not try to recontact instances anymore (2)

act5: \( \text{actv_instc} = (\{\text{prs} \times \text{SERVICES}\} \setminus \text{actv_instc}) \setminus (\{(\text{E} \setminus \text{token_owner}) \times \{\emptyset\}\}) \) \( \Rightarrow \) the peers in prs can not activate instances anymore

act6: \( \text{suspect_peers} = (\{\text{prs} \times \text{SERVICES}\} \setminus \text{suspect_peers} \rightarrow \) the peers
in prs can not suspect instances anymore (1)

\[ \text{act7: } \text{suspc\_inst} = (\text{prs} \times \text{SERVICES}) \triangleq \text{suspc\_inst} \triangleq ((E \backslash \text{token\_owner} - \sim) \times \{\emptyset\}) \]  

the peers in prs can not suspect instances anymore (2)

\[ \text{act8: } \text{inst\_state} = (\text{prs} \times \text{SERVICES}) \triangleq \text{inst\_state} \]  

\[ \text{act9: } \text{run\_inst} = (\text{prs} \times \text{SERVICES}) \triangleq \text{run\_inst} \]  

\[ \text{act10: } \text{failr\_inst} = (\text{prs} \times \text{SERVICES}) \triangleq \text{failr\_inst} \]  

END

**SUSPECT\_INST:** extended ordinary

REFINES SUSPECT\_INST

ANY 

\[ s \]  

\[ \text{a service } s \]

\[ \text{susp} \]  

\[ \text{suspicious instances} \]

WHERE

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

\[ \text{not theorem} \]  

\[ \text{grd2: } \text{susp} \subseteq \text{PEERS} \]  

\[ \text{not theorem} \]  

\[ \text{grd3: } \text{susp} = \text{run\_inst}(\text{token\_owner}(s) \mapsto s) \cap \text{unav\_peers} \]  

\[ \text{not theorem} \]  

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

\[ \text{grd4: } \text{suspc\_inst}(\text{token\_owner}(s) \mapsto s) = \emptyset \]  

\[ \text{not theorem} \]  

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

\[ \text{grd5: } \text{inst\_state}(\text{token\_owner}(s) \mapsto s) = \text{RUN\_4} \]  

\[ \text{not theorem} \]  

\[ \text{the state of } s \text{ is OK} \]

\[ \text{grd6: } \text{susp} \neq \emptyset \]  

\[ \text{not theorem} \]  

THEN

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

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

END

**FAIL:** extended ordinary

REFINES FAIL

ANY

\[ s \]  

\[ \text{a service } s \]

\[ \text{prop} \]  

\[ \text{prop} \]

WHERE

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

\[ \text{not theorem} \]  

\[ \text{grd2: } \text{prop} \subseteq \text{PEERS} \]  

\[ \text{not theorem} \]  

\[ \text{grd3: } \text{inst\_state}(\text{token\_owner}(s) \mapsto s) = \text{RUN\_4} \]  

\[ \text{not theorem} \]  

\[ \text{grd4: } \text{suspc\_inst}(\text{token\_owner}(s) \mapsto s) \neq \emptyset \]  

\[ \text{not theorem} \]  

\[ \text{grd5: } \text{prop} = \text{run\_inst}(\text{token\_owner}(s) \mapsto s) \setminus (\text{suspc\_inst}

\[ \text{token\_owner}(s) \mapsto s) \cup \text{unav\_peers}) \]  

\[ \text{not theorem} \]  

THEN

\[ \text{act1: } \text{inst\_state} = \text{inst\_state} \triangleq ((\text{prop} \times \{s\}) \times \{\text{FAIL\_4}\}) \]  

\[ \text{act2: } \text{suspc\_peers} = \text{suspc\_peers} \triangleq ((\text{prop} \times \{s\}) \times \text{suspc\_inst}

\[ \text{token\_owner}(s) \mapsto s)) \]  

\[ \text{not theorem} \]  

\[ \text{END} \]
M19

act3:  suspc\_inst(token\_owner(s) ↦ s) = ∅ >

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) ≠ ∅ 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) ≠ ∅ 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
M19

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

THEN

act1: \( \text{rect\_inst}(\text{token\_owner}(s) \mapsto s) = \text{rect\_inst}(\text{token\_owner}(s) \mapsto s) \cup \{i\} \) \( \Rightarrow \) the token owner of \( s \) has tried to recontact \( i \)

END

FAIL\_DETECT: extended ordinary \>

REFINES

FAIL\_DETECT

ANY

s \>

prop \>

susp \>

WHERE

grd1: \( s \in \text{SERVICES} \) not theorem \( \Rightarrow \)
grd2: \( \text{prop} \subseteq \text{PEERS} \) not theorem \( \Rightarrow \)
grd7: \( \text{susp} \subseteq \text{PEERS} \) not theorem \( \Rightarrow \)
grd3: \( \text{inst\_state}(\text{token\_owner}(s) \mapsto s) = \text{FAIL\_4} \) not theorem \( \Rightarrow \)
grd4: \( \text{suspct\_peers}(\text{token\_owner}(s) \mapsto s) \neq \emptyset \) not theorem \( \Rightarrow \)
grd5: \( \text{rect\_inst}(\text{token\_owner}(s) \mapsto s) = \text{suspct\_peers}(\text{token\_owner}(s) \mapsto s) \) not theorem \( \Rightarrow \)
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 \( \Rightarrow \)
grd8: \( \text{susp} = \text{suspct\_peers}(\text{token\_owner}(s) \mapsto s) \setminus \text{rctt\_inst}(\text{token\_owner}(s) \mapsto s) \) not theorem \( \Rightarrow \)

THEN

act1: \( \text{inst\_state} = \text{inst\_state} \cup ((\text{prop} \times \{s\}) \times \{\text{FAIL\_DETECT\_4}\}) \) \( \Rightarrow \)

act2: \( \text{suspct\_peers} = \text{suspct\_peers} \cup ((\text{prop} \times \{s\}) \times \{\text{susp}\}) \) \( \Rightarrow \)
act3: \( \text{rect\_inst}(\text{token\_owner}(s) \mapsto s) = \emptyset \) \( \Rightarrow \)
act4: \( \text{rctt\_inst}(\text{token\_owner}(s) \mapsto s) = \emptyset \) \( \Rightarrow \)

END

IS\_OK: extended ordinary \>

REFINES

IS\_OK

ANY

s \>

prop \>

WHERE

grd1: \( s \in \text{SERVICES} \) not theorem \( \Rightarrow \)
grd2: \( \text{prop} \subseteq \text{PEERS} \) not theorem \( \Rightarrow \)
grd3: \( \text{inst\_state}(\text{token\_owner}(s) \mapsto s) = \text{FAIL\_DETECT\_4} \) not theorem \( \Rightarrow \)
grd4: \( \text{suspct\_peers}(\text{token\_owner}(s) \mapsto s) = \emptyset \) not theorem \( \Rightarrow \)
grd5: \( \text{prop} = \text{run\_inst}(\text{token\_owner}(s) \mapsto s) \setminus \text{unav\_peers} \) not theorem \( \Rightarrow \)
\begin{verbatim}
 theorem >
 THEN
 act1: inst_state = inst_state \ (\langle \text{prop} \times \{s\} \rangle \times \{\text{RUN_4}\}) >
 END

 FAIL_ACTIVE: not extended ordinary >
 REFINES
 FAIL_ACTIVE
 ANY
 s >
 prop >
 WHERE
 grd1: s \in \text{SERVICES} not theorem >
 grd2: \text{prop} \subseteq \text{PEERS} not theorem >
 grd3: inst_state(token_owner(s) \Rightarrow s) = \text{FAIL_DETECT_4} not theorem >
 theorem >
 grd4: \text{suspct_peers}(token_owner(s) \Rightarrow s) \neq \emptyset not theorem >
 grd5: \text{prop} = \text{run_inst}(token_owner(s) \Rightarrow s) \setminus (\text{unav_peers} \cup \text{suspct_peers}(token_owner(s) \Rightarrow s)) not theorem >
 THEN
 act1: inst_state = inst_state \ (\langle \text{prop} \times \{s\} \rangle \times \{\text{FAIL_ACTIV_4}\}) >
 act2: run_inst = run_inst \ (\langle \text{prop} \times \{s\} \rangle \times \text{run_inst}(token_owner(s) \Rightarrow s) \setminus \text{suspct_peers}(token_owner(s) \Rightarrow s)) >
 act3: failr_inst = failr_inst \ (\langle \text{prop} \times \{s\} \rangle \times \text{failr_inst}(token_owner(s) \Rightarrow s) \setminus \text{suspct_peers}(token_owner(s) \Rightarrow s)) >
 act4: suspct_peers = suspct_peers \ (\langle \text{prop} \times \{s\} \rangle \times \{\emptyset\}) >
 END

 FAIL_CONFIGURE: extended ordinary >
 REFINES
 FAIL_CONFIGURE
 ANY
 s >
 prop >
 WHERE
 grd1: s \in \text{SERVICES} not theorem >
 grd2: \text{prop} \subseteq \text{PEERS} not theorem >
 grd3: inst_state(token_owner(s) \Rightarrow s) = \text{FAIL_ACTIV_4} not theorem >
 theorem >
 grd4: \text{card}(\text{run_inst}(token_owner(s) \Rightarrow s)) < \text{min_inst}(s) not theorem >
 grd5: \text{prop} = \text{run_inst}(token_owner(s) \Rightarrow s) \setminus \text{unav_peers} not theorem >
 THEN
 act1: inst_state = inst_state \ (\langle \text{prop} \times \{s\} \rangle \times \{\text{FAIL_CONFIG_4}\}) >
 END

 FAIL_IGNORE: extended ordinary >
\end{verbatim}
M19

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) u failr_inst (token_owner(s) ↦ s) u unav_peers u 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

\text{grd5: \ inst\_state(token\_owner(s) \mapsto s) = FAIL\_CONFIG\_4 \ not theorem} >

\text{grd6: \ card(actv\_instc(token\_owner(s) \mapsto s)) < deplo\_inst(s) \ not theorem} >

\text{grd7: \ card(dep\_instc(s)) + card(run\_inst(token\_owner(s) \mapsto s)) < min\_inst(s) \ not theorem} >

\text{THEN}

\text{act1: \ actv\_instc(token\_owner(s) \mapsto s) = actv\_instc(token\_owner(s) \mapsto s) u \{i\}} >

\text{END}

\text{REDEPLOY\_INSTS: extended ordinary} >

\text{REFINES}

\text{REDEPLOY\_INSTS}

\text{ANY}

\text{s} >

\text{WHERE}

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

\text{grd2: \ card(actv\_instc(token\_owner(s) \mapsto s)) = deplo\_inst(s) \ not theorem} >

\text{grd3: \ card(dep\_instc(s)) + card(run\_inst(token\_owner(s) \mapsto s)) < min\_inst(s) \ not theorem} >

\text{THEN}

\text{act1: \ dep\_instc(s) = dep\_instc(s) u actv\_instc(token\_owner(s) \mapsto s)} >

\text{act2: \ actv\_instc(token\_owner(s) \mapsto s) = \emptyset} >

\text{END}

\text{REDEPLOY: extended ordinary} >

\text{REFINES}

\text{REDEPLOY}

\text{ANY}

\text{s} >

\text{prop} >

\text{WHERE}

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

\text{grd2: \ prop \in PEERS \ not theorem} >

\text{grd3: \ inst\_state(token\_owner(s) \mapsto s) = FAIL\_CONFIG\_4 \ not theorem} >

\text{grd4: \ actv\_instc(token\_owner(s) \mapsto s) = \emptyset \ not theorem} >

\text{grd5: \ dep\_instc(s) \neq \emptyset \ not theorem} >

\text{grd6: \ card(run\_inst(token\_owner(s) \mapsto s)) + card(dep\_instc(s)) \geq min\_inst(s) \ not theorem} >

\text{THEN}

\text{grd7: \ prop = run\_inst(token\_owner(s) \mapsto s) \setminus unav\_peers \ not theorem} >

\text{THEN}
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
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

Page 9
M19

grd2:  p ∈ unav_peers not theorem

THEN

act1:  unav_peers = unav_peers \ \{p\}

END
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  
  actv_instc  
  instances activated by token owners  
  instances effectively recontacted after a try  
  inst_state  
  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_instc(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 = 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:  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: \( \text{prs} \not\in \text{unav_peers} \) not theorem \( \rightarrow \) the peers in prs are not yet unavailable

\( \forall \text{srv} \cdot \text{srv} \in \text{SERVICES} \Rightarrow \text{dom}(\text{dom(\text{inst_state})} \supset \{\text{srv}\}) \) \( \backslash \text{prs} \neq \emptyset \) not theorem \( \rightarrow \) for each service srv, there must always be at least 1 peer available

grd4: \( \text{E} \in \text{SERVICES} \rightarrow \text{PEERS} \) not theorem \( \rightarrow \) Value for token owner per service

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

grd6: \( \forall \text{srv} \cdot \text{srv} \in \text{SERVICES} \land \text{token_owner(srv)} \in \text{prs} \Rightarrow \) 
\( \text{E(srv)} \in \text{run_inst(token_owner(srv)} \rightarrow \text{srv}) \cup (\text{unav_peers} \cup \text{prs} \cup \text{failr_inst(token_owner(srv)} \rightarrow \text{srv}) \cup \text{suspct_peers(token_owner(srv)} \rightarrow \text{srv}) \) 
\( \land \) 
\( \text{E(srv)} \rightarrow \text{srv} \in \text{dom(\text{inst_state})} \cup \text{dom(\text{suspct_peers})} \cup \text{dom(\text{failr_inst})} \land \text{run_inst(}\text{E(srv)} \rightarrow \text{srv}) = \text{run_inst(token_owner(srv)} \rightarrow \text{srv}) \) 
\( \land \) 
\( \text{inst_state(}\text{E(srv)} \rightarrow \text{srv}) = \text{inst_state(token_owner(srv)} \rightarrow \text{srv}) \) 
\( \land \) 
\( \text{suspct_peers(}\text{E(srv)} \rightarrow \text{srv}) = \text{suspct_peers(token_owner(srv)} \rightarrow \text{srv}) \) 
\( \land \) 
\( \text{failr_inst(}\text{E(srv)} \rightarrow \text{srv}) = \text{failr_inst(token_owner(srv)} \rightarrow \text{srv}) \) 
\( \land \) 
\( \text{dep_instcs(}\text{E(srv)} \rightarrow \text{srv}) = \text{dep_instcs(token_owner(srv)} \rightarrow \text{srv}) \) 
not theorem \( \rightarrow \) 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: \( \text{unav_peers} = \text{unav_peers} \cup \text{prs} \) \( \rightarrow \) the peers in prs become unavailable

act2: \( \text{token_owner} = \text{token_owner} \cup \text{E} \) \( \rightarrow \) new values for token owner per service

act3: \( \text{rect_inst} = (((\text{prs} \times \text{SERVICES}) \circ \text{rect_inst}) \circ (((\text{E} \setminus \text{token_owner}) \setminus \{\emptyset\}) \circ \text{rect_inst})) \) \( \rightarrow \) the peers in prs can not try to recontact instances anymore (1)

act4: \( \text{rctt_inst} = (((\text{prs} \times \text{SERVICES}) \circ \text{rctt_inst}) \circ (((\text{E} \setminus \text{token_owner}) \setminus \{\emptyset\}) \circ \text{rctt_inst})) \) \( \rightarrow \) the peers in prs can not try to recontact instances anymore (2)

act5: \( \text{actv_instc} = (((\text{prs} \times \text{SERVICES}) \circ \text{actv_instc}) \circ (((\text{E} \setminus \text{token_owner}) \setminus \{\emptyset\}) \circ \text{actv_instc})) \)
(((E\token_owner\−) × \{∅\})  › the peers in prs can not activate instances anymore

act6: suspct_peers = (prs×SERVICES) ≔ suspct_peers  › the peers in prs can not suspect instances anymore (1)
act7: suspc_inst = ((prs×SERVICES) ≔ suspc_inst) ≔

(((E\− token_owner\−) × \{∅\})  › the peers in prs can not activate instances anymore (2)

act8: inst_state = (prs×SERVICES) ≕ inst_state  › the peers in prs can not monitor the state of the services provided anymore
act9: run_inst = (prs×SERVICES) ≕ run_inst
act10: failr_inst = (prs×SERVICES) ≕ failr_inst
act11: dep_instcs = (prs×SERVICES) ≕ dep_instcs

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_inst(token_owner(s) ↦ s) ∩ unav_peers not theorem

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

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

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

grd5: inst_state(token_owner(s) ↦ s) = RUN_4 not theorem

_the state of s is OK

grd6: susp ≠ ∅ not theorem

THEN

act1: suspc_inst(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  ›
prop  ›

WHERE

grd1: s ∈ SERVICES not theorem
grd2: prop ⊆ PEERS not theorem
grd3: inst_state(token_owner(s) ↦ s) = RUN_4 not theorem
grd4: suspc_inst(token_owner(s) ↦ s) ≠ ∅ not theorem
grd5: prop = run_inst(token_owner(s) ↦ s) ∖ (suspc_inst(token_owner(s) ↦ s) ∪ unav_peers) not theorem

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

END

RECONTACT\_INST\_OK: extended ordinary

REFINES
RECONTACT\_INST\_OK

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

WHERE

g1: \(s \in \text{SERVICES}\) not theorem \(\to\) the state of \(s\) is SUSPICIOUS

g2: \(i \in \text{PEERS}\) not theorem \(\to\) the set of suspicious peers for \(s\) is not empty

g3: \(\text{inst\_state}(\text{token\_owner}(s) \mapsto s) = \text{FAIL\_4}\) not theorem \(\to\) the state of \(s\) is SUSPICIOUS

g4: \(\text{suspct\_peers}(\text{token\_owner}(s) \mapsto s) \neq \emptyset\) not theorem \(\to\) the set of suspicious peers for \(s\) is not empty

g5: \(i \in \text{suspct\_peers}(\text{token\_owner}(s) \mapsto s)\backslash\text{unav\_peers}\) not theorem \(\to\) \(i\) is a suspicious instance of \(s\) and is available (can be contacted)

g6: \(i \not\in \text{rect\_inst}(\text{token\_owner}(s) \mapsto s)\) not theorem \(\to\) the token owner of \(s\) has not yet tried to recontact \(i\)

g7: \(\text{rect\_inst}(\text{token\_owner}(s) \mapsto s) \subset \text{suspct\_peers}(\text{token\_owner}(s) \mapsto s)\) not theorem \(\to\) the token owner of \(s\) has not yet tried to recontact all the suspicious instances of \(s\)

THEN

g1: \(\text{rect\_inst}(\text{token\_owner}(s) \mapsto s) = \text{rect\_inst}(\text{token\_owner}(s) \mapsto s) \cup \{i\}\) not theorem \(\to\) the token owner of \(s\) has tried to recontact \(i\)

g2: \(\text{rctt\_inst}(\text{token\_owner}(s) \mapsto s) = \text{rctt\_inst}(\text{token\_owner}(s) \mapsto s) \cup \{i\}\) not theorem \(\to\) \(i\) is recontacted by the token owner of \(s\) successfully

END

RECONTACT\_INST\_KO: extended ordinary

REFINES
RECONTACT\_INST\_KO

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

WHERE

g1: \(s \in \text{SERVICES}\) not theorem \(\to\) the state of \(s\) is SUSPICIOUS

g2: \(i \in \text{PEERS}\) not theorem \(\to\) the set of suspicious peers for \(s\) is not empty

g3: \(\text{inst\_state}(\text{token\_owner}(s) \mapsto s) = \text{FAIL\_4}\) not theorem \(\to\) the state of \(s\) is SUSPICIOUS

g4: \(\text{suspct\_peers}(\text{token\_owner}(s) \mapsto s) \neq \emptyset\) not theorem \(\to\) the set of suspicious peers for \(s\) is not empty

g5: \(i \in \text{suspct\_peers}(\text{token\_owner}(s) \mapsto s)\backslash\text{unav\_peers}\) not theorem \(\to\) \(i\) is a suspicious instance of \(s\) and is unavailable (can not be
\[ i \in \text{suspect\_peers}(\text{token\_owner}(s) \mapsto s) \cap \text{unav\_peers} \]

**FAIL\_DETECT:** extended ordinary

\[ \text{REFINES} \]

\[ \text{FAIL\_DETECT} \]

\[ \text{WHERE} \]

\[ \text{grd1: } s \in \text{SERVICES} \text{ not theorem} \]
\[ \text{grd2: } \text{prop} \subseteq \text{PEERS} \text{ not theorem} \]
\[ \text{grd7: } \text{susp} \subseteq \text{PEERS} \text{ not theorem} \]
\[ \text{grd3: } \text{inst\_state}(\text{token\_owner}(s) \mapsto s) = \text{FAIL\_4} \text{ not theorem} \]
\[ \text{grd4: } \text{suspect\_peers}(\text{token\_owner}(s) \mapsto s) \neq \emptyset \text{ not theorem} \]
\[ \text{grd5: } \text{rect\_inst}(\text{token\_owner}(s) \mapsto s) = \text{suspect\_peers}(\text{token\_owner}(s) \mapsto s) \text{ not theorem} \]

\[ \text{grd6: } \text{prop} = ((\text{run\_inst}(\text{token\_owner}(s) \mapsto s) \setminus \text{suspect\_peers}) \cup \text{rctt\_inst}(\text{token\_owner}(s) \mapsto s)) \setminus \text{unav\_peers} \text{ not theorem} \]

\[ \text{THEN} \]

\[ \text{act1: } \text{inst\_state} = \text{inst\_state} \setminus \{(\text{prop} \times \{s\}) \times \{\text{FAIL\_DETECT\_4}\}\} \]
\[ \text{act2: } \text{suspect\_peers} = \text{suspect\_peers} \setminus \{(\text{prop} \times \{s\}) \times \{\text{susp}\}\} \]
\[ \text{act3: } \text{rect\_inst}(\text{token\_owner}(s) \mapsto s) = \emptyset \]
\[ \text{act4: } \text{rctt\_inst}(\text{token\_owner}(s) \mapsto s) = \emptyset \]

**IS\_OK:** extended ordinary

\[ \text{REFINES} \]

\[ \text{IS\_OK} \]

\[ \text{WHERE} \]

\[ \text{grd1: } s \in \text{SERVICES} \text{ not theorem} \]
\[ \text{grd2: } \text{prop} \subseteq \text{PEERS} \text{ not theorem} \]
\[ \text{grd3: } \text{inst\_state}(\text{token\_owner}(s) \mapsto s) = \text{FAIL\_DETECT\_4} \text{ not theorem} \]
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)\suspect_peers(token_owner(s) ↦ s))) >
  act3: failr_inst ≔ failr_inst ≔ ((prop×{s})×{failr_inst(token_owner(s) ↦ s)∪suspect_peers(token_owner(s) ↦ s)}) >
  act4: suspect_peers ≔ suspect_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

  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_CONFIG_4}) >
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
(\text{token\_owner}(s) \mapsto s) \cup \text{unav\_peers} \cup \text{dep\_instcs}(\text{token\_owner}(s) \mapsto s)$ not theorem

\begin{align*}
\text{grd4: } i \notin \text{actv\_instc}(\text{token\_owner}(s) \mapsto s) & \text{ not theorem} \\
\text{grd5: } \text{inst\_state}(\text{token\_owner}(s) \mapsto s) = \text{FAIL\_CONFIG\_4} & \text{ not theorem}
\end{align*}

\begin{align*}
\text{then}
\text{grd6: } \text{card}(\text{actv\_instc}(\text{token\_owner}(s) \mapsto s)) < \text{deplo\_inst}(s) & \text{ not theorem} \\
\text{grd7: } \text{card}(\text{dep\_instcs}(\text{token\_owner}(s) \mapsto s)) + \text{card}(\text{run\_inst}(\text{token\_owner}(s) \mapsto s)) < \text{min\_inst}(s) & \text{ not theorem}
\end{align*}

\begin{align*}
\text{THEN}
\text{act1: } \text{actv\_instc}(\text{token\_owner}(s) \mapsto s) = \text{actv\_instc}(\text{token\_owner}(s) \mapsto s) \cup \{i\} & \text{ not theorem} \\
\text{act2: } \text{actv\_instc}(\text{token\_owner}(s) \mapsto s) = \emptyset & \text{ not theorem}
\end{align*}

\text{REDEPLOY\_INSTS: } \text{not extended ordinary}

\text{REFINES REDEPLOY\_INSTS ANY s \text{ prop}}

\text{WHERE}
\begin{align*}
\text{grd1: } s \in \text{SERVICES} & \text{ not theorem} \\
\text{grd2: } \text{prop} \subseteq \text{PEERS} & \text{ not theorem} \\
\text{grd3: } \text{card}(\text{actv\_instc}(\text{token\_owner}(s) \mapsto s)) = \text{deplo\_inst}(s) & \text{ not theorem}
\end{align*}

\begin{align*}
\text{then}
\text{grd4: } \text{card}(\text{dep\_instcs}(\text{token\_owner}(s) \mapsto s)) + \text{card}(\text{run\_inst}(\text{token\_owner}(s) \mapsto s)) < \text{min\_inst}(s) & \text{ not theorem} \\
\text{grd5: } \text{inst\_state}(\text{token\_owner}(s) \mapsto s) = \text{FAIL\_CONFIG\_4} & \text{ not theorem}
\end{align*}

\begin{align*}
\text{then}
\text{act1: } \text{dep\_instcs} = \text{dep\_instcs} \setminus \{(\text{prop} \times \{s\}) \times \{\text{dep\_instcs}(\text{token\_owner}(s) \mapsto s) \cup \text{actv\_instc}(\text{token\_owner}(s) \mapsto s)\}\} & \text{ not theorem} \\
\text{act2: } \text{actv\_instc}(\text{token\_owner}(s) \mapsto s) = \emptyset & \text{ not theorem}
\end{align*}

\text{REDEPLOY: } \text{not extended ordinary}

\text{REFINES REDEPLOY ANY s \text{ prop}}

\text{WHERE}
\begin{align*}
\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) \mapsto s) = \text{FAIL\_CONFIG\_4} & \text{ not theorem}
\end{align*}
M20

theorem

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

grd5: dep_instscs(token_owner(s) ↦ s) ≠ ∅ not theorem

grd6: card(run_inst(token_owner(s) ↦ s))+card(dep_instcs

(token_owner(s) ↦ s)) ≥ min_in(s) not theorem

grd7: prop = run_inst(token_owner(s) ↦ s)\unav_peers not

THEN

act1: inst_state= inst_state < ((prop×{s})×{DPL_4}) >

act2: run_inst = run_inst < ((prop×{s})× {run_inst(token _owner

(s) ↦ s) u dep_instcs(token_owner(s) ↦ s)}) >

act3: dep_instcs = dep_instcs < ((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

THEN

act1: inst_state= inst_state < ((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

THEN

act1: failr_inst = failr_inst < ((prop×{s})×{failr_inst

(token_owner(s) ↦ 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 M21
REFINES M20
SEES C09
VARIABLES
  run_inst
  suspct_peers
  failr_inst
  dep_instcs
  token_owner
  unav_peers
  suspc_inst
  rect_inst
  rctt_inst
  actv_instc
  inst_state
INVARIANTS
  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 unavaible
  grd3: ∀ srv · srv ∈ SERVICES ⇒ dom(dom(inst_state) ⊃ {srv})
        \prs ≠ ∅ not theorem for each service srv, there must always be at least 1
peer available

grd4: E ∈ SERVICES → PEERS not theorem › Value for token owner per service

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

⇒

E(srv) ∈ run_inst(token_owner(srv) ↦ srv) \ (unav_peers u prs u failr_inst(token_owner(srv) ↦ srv) \ svr) u suspct_peers(token_owner(srv) ↦ srv))

∧

E(srv) ↦ svr ∈ dom(run_inst) n dom(suspc_peers) n dom

(failr_inst) n dom(dep_instcs)

∧

run_inst(E(srv) ↦ svr) = run_inst(token_owner(srv) ↦

svr) ∧

inst_state(E(srv) ↦ svr) = inst_state(token_owner(srv) ↦

svr) ∧

suspct_peers(E(srv) ↦ svr) = suspct_peers(token_owner

(srv) ↦ svr) ∧

failr_inst(E(srv) ↦ svr) = failr_inst(token_owner(srv) ↦

svr) ∧

dep_instcs(E(srv) ↦ svr) = dep_instcs(token_owner(srv) ↦

svr) not theorem › 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: unav_peers = unav_peers u prs › the peers in prs become unavailable

act2: token_owner = token_owner E ◄ E › new values for token owner per service

act3: rect_inst = ((prs×SERVICES ◄ rect_inst) ◄

(((E\token_owner)¬)×{∅}) › the peers in prs can not try to recontact instances anymore (1)

act4: rctt_inst = ((prs×SERVICES ◄ rctt_inst) ◄

(((E\token_owner)¬)×{∅}) › the peers in prs can not try to recontact instances anymore (2)

act5: actv_instc = ((prs×SERVICES ◄ actv_instc) ◄

(((E\token_owner)¬)×{∅}) › the peers in prs can not activate instances anymore

act6: suspct_peers = (prs×SERVICES ◄ suspct_peers › the peers in prs can not suspect instances anymore (1)

act7: suspc_inst = ((prs×SERVICES ◄ suspc_inst) ◄

act7: suspc_inst ≔ ((prs×SERVICES) ⩤ suspc_inst) 
(((E∖token_owner)∼)×{∅})
† the peers in prs can not suspect instances anymore (2)

act8: inst_state ≔ (prs×SERVICES) ⩤ inst_state
† the peers in prs can not monitor the state of the services provided anymore

act9: run_inst = (prs×SERVICES) ⩤ run_inst
act10: failr_inst = (prs×SERVICES) ⩤ failr_inst
act11: dep_instcs = (prs×SERVICES) ⩤ dep_instcs

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_inst(token_owner(s) ↦ s) ∩ unav_peers not theorem
† instances in susp are suspicious if the peers running them becomes unavailable
grd4: susp = run_inst(token_owner(s) ↦ s) ∩ unav_peers not theorem
† the member of susp have not yet been suspected for s by the token owner of s
grd5: suspc_inst(token_owner(s) ↦ s) = ∅ not theorem
† the state of s is OK
grd6: susp ≠ ∅ not theorem
† the members of susp become suspected instances for s by the token owner of s

THEN
act1: suspc_inst(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 ↦ a service s
prop ↦ suspct_peers
WHERE
grd1: s ∈ SERVICES not theorem
grd2: prop ∈ PEERS not theorem
grd3: inst_state(token_owner(s) ↦ s) = RUN_4 not theorem
grd4: suspc_inst(token_owner(s) ↦ s) ≠ ∅ not theorem
grd5: prop = run_inst(token_owner(s) ↦ s) \ (suspc_inst (token_owner(s) ↦ s) ∪ unav_peers) not theorem
† (token_owner(s) ↦ s) u unav_peers)

THEN
act1: inst_state = inst_state  ((prop×{s})×{FAIL_4})
act2: suspct_peers = suspct_peers  ((prop×{s})×{suspc_inst (token_owner(s) ↦ s) )})
act3: suspc_inst(token_owner(s) ↦ s) = ∅
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 ➔
grd3: inst_state(token_owner(s) ↦ s) = FAIL_4 not theorem ➔ the state of s is SUSPICIOUS
grd4: suspct_peers(token_owner(s) ↦ s) ≠ ∅ 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: 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) ≠ ∅ 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 ↦ the token owner of \( s \) has not yet tried to recontact all the suspicious instances of \( s \)

\[ \text{THEN} \]

\[ \text{act1: } \text{rect}\_\text{inst}(\text{token}\_\text{owner}(s) \mapsto s) = \text{rect}\_\text{inst}(\text{token}\_\text{owner}(s) \mapsto s) \cup \{i\} \text{ ↦ the token owner of } s \text{ has tried to recontact } i \]

\[ \text{END} \]

\text{FAIL\_DETECT: } \text{extended ordinary ↦}

\text{REFINES}

\text{FAIL\_DETECT}

\text{ANY}

\( s \)

\( \text{prop} \)

\( \text{susp} \)

\text{WHERE}

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

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

\[ \text{grd7: } \text{susp} \subseteq \text{PEERS} \text{ not theorem } \]

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

\[ \text{grd4: } \text{suspct}\_\text{peers}(\text{token}\_\text{owner}(s) \mapsto s) \neq \emptyset \text{ not theorem } \]

\[ \text{grd5: } \text{rect}\_\text{inst}(\text{token}\_\text{owner}(s) \mapsto s) = \text{suspct}\_\text{peers}(\text{token}\_\text{owner}(s) \mapsto s) \]

\[ (s \mapsto s) \) not theorem ↦

\[ \text{grd6: } \text{prop} = (\text{run}\_\text{inst}(\text{token}\_\text{owner}(s) \mapsto s) \setminus \text{suspct}\_\text{peers} \left(\text{token}\_\text{owner}(s) \mapsto s\right)) \cup \text{rctt}\_\text{inst}(\text{token}\_\text{owner}(s) \mapsto s) \setminus \text{unav}\_\text{peers} \]

\[ (\text{token}\_\text{owner}(s) \mapsto s) \]

\[ \text{THEN} \]

\[ \text{act1: } \text{inst}\_\text{state} = \text{inst}\_\text{state} \left(\text{prop} \times \{s\}\right) \left(\text{FAIL}\_\text{DETECT}\_4\right) \]

\[ \text{act2: } \text{suspct}\_\text{peers} \left(\text{token}\_\text{owner}(s) \mapsto s\right) \left(\text{prop} \times \{s\}\right) \left(\text{susp}\right) \]

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

\[ \text{act4: } \text{rctt}\_\text{inst}(\text{token}\_\text{owner}(s) \mapsto s) = \emptyset \]

\[ \text{END} \]

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

\text{REFINES}

\text{IS\_OK}

\text{ANY}

\( s \)

\( \text{prop} \)

\text{WHERE}

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

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

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

\[ \text{grd4: } \text{suspct}\_\text{peers}(\text{token}\_\text{owner}(s) \mapsto s) = \emptyset \text{ not theorem } \]

\[ \text{grd5: } \text{prop} = \text{run}\_\text{inst}(\text{token}\_\text{owner}(s) \mapsto s) \setminus \text{unav}\_\text{peers} \]

\[ \text{not theorem } \]
THEN
act1: \(\text{inst}\_\text{state} = \text{inst}\_\text{state} \land ((\text{prop}\times\{s\})\times\{\text{RUN}\_4\})\) →
END

FAIL\_ACTIV: extended ordinary →
REFINES
FAIL\_ACTIV
ANY
s > 
prop >
WHERE
grd1: \(s \in \text{SERVICES} \not\text{ theorem}\) →
grd2: \(\text{prop} \subseteq \text{PEERS} \not\text{ theorem}\) →
grd3: \(\text{inst}\_\text{state}(\text{token}\_\text{owner}(s) \mapsto s) = \text{FAIL\_DETECT}\_4 \not\text{ theorem}\) →
grd4: \(\text{suspct}\_\text{peers}(\text{token}\_\text{owner}(s) \mapsto s) \neq \emptyset \not\text{ theorem}\) →
grd5: \(\text{prop} = \text{run}\_\text{inst}(\text{token}\_\text{owner}(s) \mapsto s) \setminus (\text{unav}\_\text{peers} \cup \text{suspct}\_\text{peers}(\text{token}\_\text{owner}(s) \mapsto s)) \not\text{ theorem}\) →
THEN
act1: \(\text{inst}\_\text{state} = \text{inst}\_\text{state} \land ((\text{prop}\times\{s\})\times\{\text{FAIL}\_\text{ACTIV}\_4\})\) →
act2: \(\text{run}\_\text{inst} = \text{run}\_\text{inst} \land ((\text{prop}\times\{s\})\times(\text{run}\_\text{inst}(\text{token}\_\text{owner}(s) \mapsto s) \setminus \text{suspct}\_\text{peers}(\text{token}\_\text{owner}(s) \mapsto s)))\) →
act3: \(\text{failr}\_\text{inst} = \text{failr}\_\text{inst} \land ((\text{prop}\times\{s\})\times(\text{failr}\_\text{inst}(\text{token}\_\text{owner}(s) \mapsto s) \cup \text{suspct}\_\text{peers}(\text{token}\_\text{owner}(s) \mapsto s)))\) →
act4: \(\text{suspct}\_\text{peers} = \text{suspct}\_\text{peers} \land ((\text{prop}\times\{s\})\times\emptyset)\) →
END

FAIL\_CONFIGURE: extended ordinary →
REFINES
FAIL\_CONFIGURE
ANY
s > 
prop >
WHERE
grd1: \(s \in \text{SERVICES} \not\text{ theorem}\) →
grd2: \(\text{prop} \subseteq \text{PEERS} \not\text{ theorem}\) →
grd3: \(\text{inst}\_\text{state}(\text{token}\_\text{owner}(s) \mapsto s) = \text{FAIL\_ACTIV}\_4 \not\text{ theorem}\) →
grd4: \(\text{card}(\text{run}\_\text{inst}(\text{token}\_\text{owner}(s) \mapsto s)) < \text{min}\_\text{inst}(s) \not\text{ theorem}\) →
grd5: \(\text{prop} = \text{run}\_\text{inst}(\text{token}\_\text{owner}(s) \mapsto s) \setminus \text{unav}\_\text{peers} \not\text{ theorem}\) →
THEN
act1: \(\text{inst}\_\text{state} = \text{inst}\_\text{state} \land ((\text{prop}\times\{s\})\times\{\text{FAIL}\_\text{CONFIG}\_4\})\) →
END

FAIL\_IGNORE: extended ordinary →
M21

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

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

THEN
    act1: inst_state = inst_state ≔ ((prop×{s})×{RUN.4})

END

REDEPLOY_INSTC: extended ordinary

REFINES
    REDEPLOY_INSTC

ANY
s
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_instcs(token_owner(s) ↦ s) not theorem
    grd4: i ∉ actv_instc(token_owner(s) ↦ s) not theorem
M21

\[ \text{grd5: } \text{inst} \_\text{state}(\text{token} \_\text{owner}(s) \mapsto s) = \text{FAIL} \_\text{CONFIG} \_4 \] not theorem

\[ \text{grd6: } \text{card}(\text{actv} \_\text{instc}(\text{token} \_\text{owner}(s) \mapsto s)) < \text{deplo} \_\text{inst}(s) \] not theorem

\[ \text{grd7: } \text{card}(\text{dep} \_\text{instcs}(\text{token} \_\text{owner}(s) \mapsto s)) + \text{card}(\text{run} \_\text{inst}\ (\text{token} \_\text{owner}(s) \mapsto s)) < \text{min} \_\text{inst}(s) \] not theorem

\[ \text{THEN} \]

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

\[ \text{END} \]

REDEPLOY \_\text{INSTS:} \text{ extended ordinary} \]

REFINES

REDEPLOY \_\text{INSTS}

\text{ANY} \]

\[ s \]

\[ \text{prop} \]

\[ \text{WHERE} \]

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

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

\[ \text{grd3: } \text{card}(\text{actv} \_\text{instc}(\text{token} \_\text{owner}(s) \mapsto s)) = \text{deplo} \_\text{inst}(s) \] not theorem

\[ \text{grd4: } \text{card}(\text{dep} \_\text{instcs}(\text{token} \_\text{owner}(s) \mapsto s)) + \text{card}(\text{run} \_\text{inst}\ (\text{token} \_\text{owner}(s) \mapsto s)) < \text{min} \_\text{inst}(s) \] not theorem

\[ \text{THEN} \]

\[ \text{act1: } \text{dep} \_\text{instcs} = \text{dep} \_\text{instcs} \cup (((\text{prop} \times \{s\}) \times \{\text{dep} \_\text{instcs}\ (\text{token} \_\text{owner}(s) \mapsto s) \} \cup \text{actv} \_\text{instc}(\text{token} \_\text{owner}(s) \mapsto s))) \]

\[ \text{act2: } \text{actv} \_\text{instc}(\text{token} \_\text{owner}(s) \mapsto s) = \emptyset \]

\[ \text{END} \]

REDEPLOY: \text{ extended ordinary} \]

REFINES

REDEPLOY

\text{ANY} \]

\[ s \]

\[ \text{prop} \]

\[ \text{WHERE} \]

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

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

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

\[ \text{grd4: } \text{actv} \_\text{instc}(\text{token} \_\text{owner}(s) \mapsto s) = \emptyset \] not theorem

\[ \text{grd5: } \text{dep} \_\text{instcs}(\text{token} \_\text{owner}(s) \mapsto s) \neq \emptyset \] not theorem

\[ \text{grd6: } \text{card}(\text{run} \_\text{inst}(\text{token} \_\text{owner}(s) \mapsto s)) + \text{card}(\text{dep} \_\text{instcs} \]
(token_owner(s) ↦ s)) ≥ \text{min}_\text{inst}(s) \not\text{ theorem} >

\text{grd7: } \text{prop} = \text{run}_\text{inst}((\text{token}_\text{owner}(s) ↦ s) \cup \text{unav}_\text{peers} \not\text{ theorem} >

\text{THEN} \hspace{1cm}
\text{act1: } \text{inst}_\text{state} = \text{inst}_\text{state} \setminus ((\text{prop} \times \{s\}) \times \{DPL_4\}) >
\text{act2: } \text{run}_\text{inst} = \text{run}_\text{inst} \setminus ((\text{prop} \times \{s\}) \times \{\text{run}_\text{inst}((\text{token}_\text{owner}(s) ↦ s) \cup \text{dep}_\text{instcs}(\text{token}_\text{owner}(s) ↦ s))\}) >
\text{act3: } \text{dep}_\text{instcs} = \text{dep}_\text{instcs} \setminus ((\text{prop} \times \{s\}) \times \{\emptyset\}) >
\text{END} \hspace{1cm}

\text{HEAL: } \text{extended ordinary} >
\text{REFINES} \hspace{1cm}
\text{HEAL} \hspace{1cm}
\text{ANY} \hspace{1cm}
\text{s} >
\text{prop} >
\text{WHERE} \hspace{1cm}
\text{grd1: } s \in \text{SERVICES} \not\text{ theorem} >
\text{grd2: } \text{prop} \subseteq \text{PEERS} \not\text{ theorem} >
\text{grd3: } \text{inst}_\text{state}((\text{token}_\text{owner}(s) ↦ s) = \text{DPL}_4 \not\text{ theorem} >
\text{grd4: } \text{prop} = \text{run}_\text{inst}((\text{token}_\text{owner}(s) ↦ s) \cup \text{unav}_\text{peers} \not\text{ theorem} >
\text{THEN} \hspace{1cm}
\text{act1: } \text{inst}_\text{state} = \text{inst}_\text{state} \setminus ((\text{prop} \times \{s\}) \times \{\text{RUN}_4\}) >
\text{END} \hspace{1cm}

\text{UNFAIL_PEER: } \text{extended ordinary} >
\text{REFINES} \hspace{1cm}
\text{UNFAIL_PEER} \hspace{1cm}
\text{ANY} \hspace{1cm}
\text{s} >
\text{p} >
\text{prop} >
\text{WHERE} \hspace{1cm}
\text{grd1: } s \in \text{SERVICES} \not\text{ theorem} >
\text{grd2: } \text{prop} \subseteq \text{PEERS} \not\text{ theorem} >
\text{grd3: } p \in \text{PEERS} \not\text{ theorem} >
\text{grd4: } p \in \text{failr}_\text{inst}((\text{token}_\text{owner}(s) ↦ s) \not\text{ theorem} >
\text{grd5: } \text{prop} = \text{run}_\text{inst}((\text{token}_\text{owner}(s) ↦ s) \cup \text{unav}_\text{peers} \not\text{ theorem} >
\text{THEN} \hspace{1cm}
\text{act1: } \text{failr}_\text{inst} = \text{failr}_\text{inst} \setminus ((\text{prop} \times \{s\}) \times \{\text{failr}_\text{inst}((\text{token}_\text{owner}(s) ↦ s) \setminus \{p\})\}) >
\text{END} \hspace{1cm}

\text{MAKE_PEER_AVAIL: } \text{extended ordinary} >
\text{REFINES} \hspace{1cm}
\text{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