Analysis of Self-* and P2P Systems using Refinement  
(Full Report)  
Manamiary Bruno Andriamiharina, 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 Andriamiarina¹, Dominique Méry¹, and Neeraj Kumar Singh²

Université de Lorraine, LORIA, BP 239, 54506 Vandoeuvre-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 analysing a similar class of systems for years, namely distributed algorithms.

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

We consider that a given system $S$ (see in Fig.1) is characterized by a set of events (procedures modelling either phases or basic actions according to an abstraction level) that modifies the state of the system. Legal states (correct states) satisfying a safety property $P$ are defined by a subset $CL$ 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 characterise the fault model in a very abstract way and it may be possible to develop the fault model according to the assumptions on the environment, but we do not consider this in the current study. We restrict our study by making explicit the events of $ST$ 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 [1], dealing with \textit{events} and \textit{invariant} properties including \textit{convergence} properties by using a temporal framework. The \textit{service-as-event} paradigm [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-$\star$ systems. Section 4 presents the formal verification approach and illustrates the proposed methodology with the study of the self-healing P2P-based protocol [14]. Section 5 discusses on approaches for studying temporal properties for \textsc{Event} B models. Finally, Section 6 concludes the paper along with future work.

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

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

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

Smith et al. [21] have applied the stepwise refinement using Z to study a case of self-reconfiguration, where a set of autonomous robotic agents is able to assemble and to reach a global shape. They do not validate models using an adequate tool (e.g. proof checker, proof assistant, etc.) and models are not localized. Calinescu et al. [6] have used Alloy to demonstrate the correctness of the autonomic computing policies (ACP). However, Alloy does not provide a mechanism for expressing the \textit{correct-by-construction} paradigm. Méry et al. [2] have also investigated a self-reconfiguring system (Network-on-Chip: \textit{adaptive 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-$\star$ 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-$\star$ systems hold [6, 8, 10, 24]. Moreover, these tools allow users to obtain the metrics
for the self-⋆ systems, such as performance, and quantitative evaluations [6, 8, 10, 24]. Model-checking and state-space evaluation can be used during the conception of self-⋆ systems, but they are especially used for runtime verification [10, 24]. The limit of model checking is clearly the size of models.

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

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

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

3 Modelling Framework

3.1 EVENT B

We advocate the use of correct-by-construction paradigm for modelling the self-⋆ systems. The key concept is the incremental refinement (simulation) which provides link between discrete models by preserving properties. The EVENT B modelling language designed by Abrial [1] is based on set theory and the refinement of models: an abstract model expressing the requirements of a given system can be verified and validated easily; a concrete model corresponding to the actual system is constructed progressively by refining the abstraction. EVENT B is supported by a complete 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: ANY \( t \) WHERE \( G(t, x) \) THEN \( x : \{ P(t, x, x') \} \) END and corresponds to the transformation of the state of the variable \( x \), which is described by a before-after predicate \( BA(e)(x, x') \): the predicate is semantically equivalent to \( \exists t \cdot G(t, x) \land P(t, x, x') \) and expresses the relationship linking the values of the state variables before \( (x) \) and just after \( (x') \) the execution of the event \( e \). Proof obligations are produced by RODIN, from events: INV1 and INV2 state that an invariant condition \( I(x) \) is preserved; their general form follows immediately from the definition of the before-after predicate \( BA(e)(x, x') \) of each event \( e \); FIS expresses the feasibility of an event \( e \), with respect to the invariant \( I \). By proving feasibility, we achieve that \( BA(e)(x, z) \) provides a next state whenever the guard \( grd(e)(x) \) holds: the guard is the enabling condition of the event.

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

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

Fig. 2: Machines and Contexts relationships

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:

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

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 \textit{eventually} a legal state (satisfying \( P \)). This service is expressed, with the leads to \((\leadsto)\) operator, as follows: \((\neg P) \leadsto P\). This leads to property (equivalently \((\neg P) \implies \Diamond P\)) states that every illegal state (satisfying \( \neg P \)) will \textit{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 \( WF_y(e) \): we express a weak fairness assumption over each event \( e \) modelling a step of the recovery process (we do not add any fairness on events leading to illegal states (faults)).

---

\(^1\)http://lfm.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 \( P \iff Q \) and \( Q \iff R \), it satisfies \( P \iff R \).
2. If \( M \) satisfies \( P \iff Q \) and \( R \iff Q \), it satisfies \((P \lor R) \iff Q\).
3. If \( I \) is invariant for \( M \) and if \( M \) satisfies \( P \land I \iff Q \), then \( M \) satisfies \( P \iff Q \).
4. If \( I \) is invariant for \( M \) and if \( M \) satisfies \( P \land I \Rightarrow Q \), then \( M \) satisfies \( P \Rightarrow Q \).
5. If \( P \Rightarrow Q \) is a link of \( D \) for the machine \( M \), then \( M \) satisfies \( P \Rightarrow Q \).
6. If \( P \land Q \) 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 \( (\neg P) \Rightarrow P \), 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: \( (\neg P) \Rightarrow P \). An abstract event \( (e) \) is used for describing the service/procedure represented by \( (\neg P) \Rightarrow P; (\neg P) \xrightarrow{e} P \); 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 \((\neg P) \rightsquigarrow P\) into simple sub-procedures/steps, using the inference rules [11] related to the leads to properties:

\[
\begin{align*}
\cdots & \cdots \xrightarrow{\text{trans}_2} R_0 \xrightarrow{\text{trans}_4} R_1 \xrightarrow{\text{trans}_5} R_0 \xrightarrow{\text{trans}_1} P \\
\underbrace{\text{Fig. 4: Proof Tree - Usage of Inference Rules}}_{} & \\
\end{align*}
\]

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 \((\neg P) \rightsquigarrow P\) is decomposed, until the identification of the stabilization steps is satisfactory. The stabilization phase is expressed by the property

\[
(\neg P) \rightsquigarrow R_0 \land R_0 \rightsquigarrow R_1 \land \ldots \land R_{n-1} \rightsquigarrow R_n \land R_n \rightsquigarrow P,
\]

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 $s \mapsto st \in \text{serviceState}$, where $s$ denotes a service and $st$ denotes a possible state. A property $P$ expresses that a service (s) is in a legal running state that is formalised as $P \equiv (s \mapsto \text{RUN} \in \text{serviceState})$. An event 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 $R_0 \equiv (s \mapsto \text{FL}_DT_1 \in \text{serviceState}_1)$ and a new event FAIL_DETECT in this self-detection mechanism. Let $P$ and $\neg P$ be redefined as follows: $P \equiv (s \mapsto \text{RUN}_1 \in \text{serviceState}_1)$ and $\neg P \equiv (s \mapsto \text{FAIL}_1 \in \text{serviceState}_1)$.

This macro/abstract view of the self-healing is detailed by refinement$^2$, using intermediate steps. A set of new variables is introduced to capture the system requirements. The variables are denoted by NAME$_{\text{Refinement Level}}$.

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

2 $\oplus$: 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.

\[
\begin{align*}
\text{FAIL} & \overset{\text{FAIL\_DETECT}}{\rightarrow} \text{FL\_DT\_1} \\
\text{RUN\_1} & \overset{\text{HEAL}}{\rightarrow} \text{FL\_DT\_1} \\
\text{FAIL\_DETECT} & \overset{\text{FAIL\_DETECT}}{\rightarrow} \text{FL\_DT\_1}
\end{align*}
\]

Fig. 8: Self-Detection

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).

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

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 web\(^3\).

Introducing the Self-Activation (M2) and Self-Configuration (M3) The self-activation is introduced in this refinement M2 (see Fig. 9), where a failure of a service (s) is evaluated in terms of critical or non-critical using a new state 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).

\[
\begin{align*}
\text{EVENT \text{FAILURE} \text{REFINES} \text{FAILURE}} & \\
\text{WHERE} & \circ s \rightarrow RUN\_1 \in \text{serviceState\_1} \\
\text{THEN} & \circ \text{act1} : \ldots \\
& \circ \text{serviceState\_1} := \text{(serviceState\_1} \setminus \{s \rightarrow \text{RUN\_1}\}) \\
& \cup \{s \rightarrow \text{FAIL\_1}\}
\end{align*}
\]

\[
\begin{align*}
\text{EVENT \text{FAIL\_DETECT}} & \\
\text{ANY} & \\
\text{WHERE} & \circ \text{act2} : \ldots \\
& \circ s \rightarrow \text{FAIL\_1} \in \text{serviceState\_1} \\
\text{THEN} & \circ \text{serviceState\_1} := \text{(serviceState\_1} \setminus \{s \rightarrow \text{FAIL\_1}\}) \\
& \cup \{s \rightarrow \text{FL\_DT\_1}\}
\end{align*}
\]

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

\[^3\text{http://eb2all.loria.fr/html_files/files/selfhealing/self-healing.zip}\]
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_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 (susp) of unavailable peers instances of service (s). Other events RECONTACT_INST_OK and RECONTACT_INST_KO are used to specify the successful recontact, and failed recontact, respectively, of the unavailable instances for ensuring the failed states. Moreover, the token owner is able to monitor the status of service (s) using two events FAIL_DETECT, and IS_OK. If there are unavailable instances after the recontacting procedure, the token owner informs the safe members of MPG of failed instances using the event FAIL_DETECT, otherwise, the token owner indicates that service is running properly. (2) Self-Activation introduces an event FAIL_ACTIV that states that if there are failed instances of service (s), then the token owner evaluates if the failure is critical. Another event FAIL_IGNORE specifies that the failure is not critical. An event IGNORE can ignore the failure if several instances (more than minimum) are running correctly. If the number of instances for the running service (s) will be less than the minimum required services, then the failure will be declared critical, and the self-healing process will be triggered using an event FAIL_CONFIGURE. (3) Self-
Configuration 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_AVL: 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_AVL: If the token owner of a service \( s \) becomes unavailable, at least one peer, with the same characteristics as the disabled token owner (state, local informations about running, failed peers, etc.) can become the new token owner of service \( s \); (2) Event REDEPLOY_INSTC: There is always a sufficient number of available peers that can be deployed to reach the legal running state of a service \( s \).

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

5 Analysis of Temporal Properties for Event-B Models

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

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

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

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

<table>
<thead>
<tr>
<th>Model</th>
<th>Total</th>
<th>Auto</th>
<th>Interactive</th>
</tr>
</thead>
<tbody>
<tr>
<td>CONTEXTS</td>
<td>30</td>
<td>26</td>
<td>86.67%</td>
</tr>
<tr>
<td>M0</td>
<td>3</td>
<td>3</td>
<td>100%</td>
</tr>
<tr>
<td>M1</td>
<td>21</td>
<td>15</td>
<td>71.4%</td>
</tr>
<tr>
<td>M2</td>
<td>46</td>
<td>39</td>
<td>84.8%</td>
</tr>
<tr>
<td>M3</td>
<td>68</td>
<td>0</td>
<td>0%</td>
</tr>
<tr>
<td>M4</td>
<td>142</td>
<td>16</td>
<td>11.27%</td>
</tr>
<tr>
<td>M5</td>
<td>46</td>
<td>17</td>
<td>39.59%</td>
</tr>
<tr>
<td>OTHER MACHINES</td>
<td>1093</td>
<td>141</td>
<td>12.44%</td>
</tr>
<tr>
<td>M21</td>
<td>13</td>
<td>0</td>
<td>0%</td>
</tr>
<tr>
<td>TOTAL</td>
<td>1434</td>
<td>257</td>
<td>17.9%</td>
</tr>
</tbody>
</table>

Table 2: Synthesis of POs

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 ≠ Ø not theorem
  axm2: STATES = {RUN, FAIL} not theorem
  axm3: RUN ≠ FAIL not theorem
  axm4: InitState ∈ SERVICES ↔ STATES not theorem
  axm5: ∀ s · s ∈ SERVICES ⇒ s ↦ RUN ∈ InitState not theorem
  axm6: ∀ s, st1, st2 · s ∈ SERVICES ∧ st1 ∈ STATES ∧ st2 ∈ STATES ∧ s ↦ st1 ∈ InitState ∧ s ↦ st2 ∈ InitState ⇒ st1 = st2 not theorem
END
CONTEXT C01
EXTENDS C00
SETS STATES_1
CONSTANTS RUN_1 FAIL_1 FAIL_DETECT_1 InitState_1

AXIOMS
axm1: partition(STATES_1, {RUN_1},{FAIL_1},{FAIL_DETECT_1}) not theorem
axm2: InitState_1 ∈ SERVICES ↔ STATES_1 not theorem
axm3: ∀ s · s ∈ SERVICES ⇒ s ↦ RUN_1 ∈ InitState_1 not theorem
axm4: ∀ s, st1, st2 · s ∈ SERVICES ∧ st1 ∈ STATES_1 ∧ st2 ∈ STATES_1 ∧ s ↦ st1 ∈ InitState_1 ∧ s ↦ st2 ∈ InitState_1 ⇒ st1 = st2 not theorem

END
CONTEXT C02
EXTENDS C01
SETS STATES_2
CONSTANTS
  RUN_2
  FAIL_2
  FAIL_DETECT_2
  FAIL_ACTIV_2
  InitState_2
AXIOMS
  axm1: partition(STATES_2, {RUN_2},{FAIL_2},{FAIL_DETECT_2}, {FAIL_ACTIV_2})
  axm2: InitState_2 ∈ SERVICES ↔ STATES_2
  axm3: ∀ s · s ∈ SERVICES ⇒ s ↦ RUN_2 ∈ InitState_2
  axm4: ∀ s, st1, st2 · s ∈ SERVICES ∧ st1 ∈ STATES_2 ∧ st2 ∈ STATES_2 ∧ s ↦ st1 ∈ InitState_2 ∧ s ↦ st2 ∈ InitState_2 ⇒ st1 = st2
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 · s ∈ SERVICES ⇒ s ↦ RUN_4 ∈ InitState_4 not theorem
 axm4: ∀ s, st1, st2 · s ∈ SERVICES ∧ st1 ∈ STATES_4 ∧ st2 ∈ STATES_4 ∧
 s ↦ st1 ∈ InitState_4 ∧ s ↦ st2 ∈ InitState_4 ⇒ st1 = st2 not theorem
END
CONTEXT
  C05
EXTENDS
  C04
CONSTANTS
  min_inst
  init_inst
AXIOMS
  axm1: \text{min\_inst} \in \text{SERVICES} \rightarrow \mathbb{N}_1 \text{ not theorem}
  axm2: \text{init\_inst} \in \text{SERVICES} \rightarrow \mathbb{N}_1 \text{ not theorem}
  axm3: \forall s \cdot s \in \text{SERVICES} \Rightarrow \text{min\_inst}(s) \geq 2 \text{ not theorem}
  axm4: \forall s \cdot s \in \text{SERVICES} \Rightarrow \text{init\_inst}(s) \geq \text{min\_inst}(s) \text{ not theorem}
  axm5: \forall s \cdot s \in \text{SERVICES} \Rightarrow \text{init\_inst}(s) \geq 2 \text{ theorem}
END
CONTEXT
C06
EXTENDS
C05
SETS
PEERS  \text{Set of PEERS}
CONSTANTS
InitSrvcPeers  \text{Initial set of peers / instances per service}
AXIOMS
\textbf{axm1}: \text{InitSrvcPeers} \in \text{SERVICES} \rightarrow \mathcal{P}_1(\text{PEERS}) \text{ not theorem} \Rightarrow \text{each service is provided by a non empty set of peers/instances}
\textbf{axm2}: \forall s \cdot s \in \text{SERVICES} \Rightarrow \text{finite(InitSrvcPeers(s))} \text{ not theorem} \Rightarrow \text{each service is provided by a finite set of peers/instances}
\textbf{axm3}: \forall s \cdot s \in \text{SERVICES} \Rightarrow \text{card(InitSrvcPeers(s))} = \text{init\_inst(s)} \text{ not theorem} \Rightarrow \text{each service s is provided by peers/instances, whose number is init\_inst(s)}
\textbf{axm4}: \forall s_1, s_2 \cdot s_1 \subseteq \text{PEERS} \land s_2 \subseteq \text{PEERS} \land s_1 \neq \emptyset \land s_2 \neq \emptyset \land \text{finite}(s_1) \land \text{finite}(s_2) \land s_1 \subset s_2 \Rightarrow \text{card}(s_1) \leq \text{card}(s_2) - 1 \text{ not theorem} \Rightarrow \text{each service s is provided by peers/instances, whose number is init\_inst(s)}
\textbf{axm5}: \forall s_1 \cdot s_1 \subseteq \text{PEERS} \land s_1 \neq \emptyset \land \text{finite}(s_1) \Rightarrow \text{card}(s_1) > 0 \text{ theorem} \Rightarrow \text{each service s is provided by peers/instances, whose number is init\_inst(s)}
\textbf{axm6}: \forall s_1, s_2 \cdot s_1 \subseteq \text{PEERS} \land s_2 \subseteq \text{PEERS} \land \text{finite}(s_1) \land \text{finite}(s_2) \land s_1 \subseteq s_2 \Rightarrow \text{card}(s_2) - \text{card}(s_1) = \text{card}(s_2\backslash s_1) \text{ not theorem} \Rightarrow \text{each service s is provided by peers/instances, whose number is init\_inst(s)}
\text{END}
CONTEXT
    C07
EXTENDS
    C06
CONSTANTS
deplo_inst
AXIOMS
    axm1: ∀ set, s1, s2 · set ⊆ SERVICES×PEERS ∧ s1 ∈ SERVICES ∧ s2 ∈ SERVICES ∧ s1 = s2 ⇒ ({s1} ⩤ set)[{s2}] = ∅ theorem
    axm2: ∀ set, s1, s2 · set ⊆ SERVICES×PEERS ∧ s1 ∈ SERVICES ∧ s2 ∈ SERVICES ∧ s1 ≠ s2 ⇒ ({s1} ⩤ set)[{s2}] = set[{s2}] theorem
    axm3: ∀ set, s1, s2, p · set ⊆ SERVICES×PEERS ∧ s1 ∈ SERVICES ∧ s2 ∈ SERVICES ∧ p ∈ PEERS ∧ s1 = s2 ⇒ (set ∪ {s1 ↦ p})[{s2}] = set[{s2}] union{p} 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

InitStateSrv

InitSuspPrs

InitRunPeers

AXIOMS

axm1: InitStateSrv ∈ PEERS × SERVICES ↔ STATES_4 not theorem

axm2: ∀ s, p · p ∈ PEERS ∧ s ∈ SERVICES ∧ p ∈ InitSrvcPeers(s) ⇒ (p ↦ s) ↦ RUN_4 ∈ InitStateSrv not theorem

InitStateSrv ⇒ p ∈ InitSrvcPeers(s) ∧ stt = RUN_4 not theorem

axm3: ∀ s, p, stt · p ∈ PEERS ∧ s ∈ SERVICES ∧ (p ↦ s) ↦ stt ∈ InitSuspPrs 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 ∈ InitRunPeers 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 not theorem

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

EVENTS
INITIALISATION: not extended ordinary
  THEN
  act1: serviceState_1 := InitState_1
END

FAIL: not extended ordinary
  REFINES FAIL
  ANY
  s
  WHERE
  grd1: s ∈ SERVICES not theorem
  grd2: s ↦ RUN_1 ∈ serviceState_1 not theorem
  THEN
  act1: serviceState_1 := (serviceState_1∖{s ↦ RUN_1}) ∪ {s ↦
END

FAIL_DETECT: not extended ordinary
  REFINES FAIL
  ANY
  s
  WHERE
  grd1: s ∈ SERVICES not theorem
M01

\(手工2: \quad s \mapsto \text{FAIL}_1 \in \text{serviceState}_1 \) not theorem

THEN

act1: \quad \text{serviceState}_1 = (\text{serviceState}_1 \setminus \{s \mapsto \text{FAIL}_1\}) \cup \{s \mapsto \text{FAIL\_DETECT}_1\}

END

\text{HEAL}: \quad \text{not extended ordinary}

\text{REFINES}

\text{HEAL}

\text{ANY}

s

WHERE

\(手工1: \quad s \in \text{SERVICES} \) not theorem

\(手工2: \quad s \mapsto \text{FAIL\_DETECT}_1 \in \text{serviceState}_1 \) not theorem

THEN

act1: \quad \text{serviceState}_1 = (\text{serviceState}_1 \setminus \{s \mapsto \text{FAIL\_DETECT}_1\}) \cup \{s \mapsto \text{RUN}_1\}

END

END
MACHINE
M02
REFINES
M01
SEES
C02
 intra
servicie

INVAR
states ∈ S, TATES2 ∈ S, TATES1 ∈ S, TATES0 ∈ S

REFINES
FAIL
AND
FAIL_DETECT
REFINES
FAILECT

AND

BRE

THEN

IF \( s \text{ } ^{FAI\text{LTE}_2} \) serviceState_2 \leftrightarrow \text{reh}_2 \):

FAILECT2

END

IS_OK

REFINES HEA

AND

BRE

THEN

IF \( s \text{ } ^{FAI\text{LTE}_2} \) serviceState_2 \leftrightarrow \text{reh}_2 \):

FAILACT2

END

FAIL.ACTI

REFINES HEA

AND

BRE

THEN

IF \( s \text{ } ^{FAI\text{LTE}_2} \) serviceState_2 \leftrightarrow \text{reh}_2 \):

FAILACT2

END

REAL

REFINES HEA

AND

BRE

THEN

IF \( s \text{ } ^{FAI\text{LTE}_2} \) serviceState_2 \leftrightarrow \text{reh}_2 \):

FAILACT2

END
MACHINE M03
REFINES M0
SEES C03
RIADS

INVAR

RUN3 serv1 serv2 serv3 serv4 serv5 serv6 serv7 serv8 serv9 

INIT

FAIL:

THEN

END

FAI

IF

THEN

END

BRE

THEN

END

FAI

END

END
FAIL_DETECT: not extended ordinary

FAIL_CONF_IGN: not extended ordinary

FAIL_ACTI: not extended ordinary

FAIL_OK: not extended ordinary

END

\[\text{\texttt{M03}}\]
\[
\begin{align*}
\text{FAIL_DETECT:} & \quad \text{not extended ordinary} \\
\text{FAIL_OK:} & \quad \text{not extended ordinary} \\
\text{FAIL_ACTI:} & \quad \text{not extended ordinary} \\
\text{FAIL_CONF_IGN:} & \quad \text{not extended ordinary}
\end{align*}
\]
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 } \]

\[ \text{REFINES} \]

\[ \text{HEAL} \]

\[ \text{ANY} \]

\[ s \mapsto \]

\[ \text{st} \mapsto \]

\[ \text{WHERE} \]

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

\[ \text{grd3: } \text{st} \in \{\text{FAIL\_CONFIG\_3,FAIL\_IGN\_3}\} \text{ not theorem } \]

\[ \text{grd2: } s \mapsto \text{st} \in \text{serviceState\_3} \text{ not theorem } \]

\[ \text{THEN} \]

\[ \text{act1: } \text{serviceState\_3} = (\text{serviceState\_3} \setminus \{s \mapsto \text{st}\}) \cup \{s \mapsto \text{RUN\_3}\} \]

\[ \text{END} \]

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

INVARİANTS
  inv1: serviceState_4 $ SERVICES <-> STATES_4 $ serviceState_4 3 4
  gluing_run1: ∀ Y: Y $ SERVICES ∧ s ↦ RUN_3 $ Y $ RUN_4 4 4
  gluing_run2: ∀ Y: Y $ SERVICES ∧ s ↦ RUN_4 $ Y $ RUN_3 4 4
  gluing_fail1: ∀ Y: Y $ SERVICES ∧ s ↦ FAIL_3 $ Y $ FAIL_4 4 4
  gluing_fail2: ∀ Y: Y $ SERVICES ∧ s ↦ FAIL_4 $ Y $ FAIL_3 4 4
  gluing_dtct1: ∀ Y: Y $ SERVICES ∧ s ↞ FAIL_DETECT_3 $ Y $ FAIL_DETECT_4 4 4
  gluing_dtct2: ∀ Y: Y $ SERVICES ∧ s ↞ FAIL_DETECT_4 $ Y $ FAIL_DETECT_3 4 4
  gluing_act1: ∀ Y: Y $ SERVICES ∧ s ↞ FAIL_ACTIVE_3 $ Y $ FAIL_ACTIVE_4 4 4
  gluing_act2: ∀ Y: Y $ SERVICES ∧ s ↞ FAIL_ACTIVE_4 $ Y $ FAIL_ACTIVE_3 4 4
  gluing_ign1: ∀ Y: Y $ SERVICES ∧ s ↞ FAIL_IGN_3 $ Y $ FAIL_IGN_4 4 4
  gluing_ign2: ∀ Y: Y $ SERVICES ∧ s ↞ FAIL_IGN_4 $ Y $ FAIL_IGN_3 4 4
  gluing_conf1: ∀ Y: Y $ SERVICES ∧ s ↞ FAIL_CONFIG_3 $ Y $ Y $ FAIL_CONFIG_4 $ Y $ DPL_4 4 4
  gluing_conf2: ∀ Y, Yt: Y $ SERVICES ∧ s $ STATISTICS_4 ∧ Yt $ STATES_4 ∧ Y $ Yt $ Y $ Yt 4 4
  gluing_YtatH: ∀ Y, Yt1, Yt2: Y $ SERVICES ∧ s $ STATISTICS_4 ∧ Yt2 $ STATES_4 ∧ Y $ Yt1 $ Y $ Yt2 4 4

EVENTS
  INITIALISATION: not e
  THEN
    serviceState_4 $ SERVICES <-> STATES_4 4 4
  END

FAIL $ serviceState_4
REFINES FAI
\[ \begin{align*}
\text{FAIL}_4 & \quad \text{not theorem} \\
\text{FAIL}_4 & \quad \text{not theorem} \\
\text{FAIL}_{\text{DETECT}} & \quad \text{not theorem} \\
\text{FAIL}_{\text{ACTIV}} & \quad \text{not theorem}
\end{align*} \]
FAIL_CONFIGURE:  

```plaintext
FAIL_CONFIGURE s (FAIL_CONFIG 3) {s ↦ FAIL_CONFIG_4}:
  FAIL_CONFIG s (FAIL_CONFIG 3) {s ↦ FAIL_CONFIG_4}:
    s ∈ SERVICES not theorem:
      s ∈ SERVICES not theorem:
        s 4 (s 4):
          st = FAIL_CONFIG_3:
            st = FAIL_CONFIG_3:
              st = FAIL_CONFIG_3:
                act1: serviceState_4 = serviceState_4 \ {s ↦ FAIL_CONFIG_4} ∪ {s ↦ FAIL_CONFIG_4}:
```

FAIL_IGNORE:  

```plaintext
FAIL_IGNORE s (FAIL_IGNORE 4) {s ↦ FAIL_IGNORE_4}:
  FAIL_IGNORE s (FAIL_IGNORE 4) {s ↦ FAIL_IGNORE_4}:
    s ∈ SERVICES not theorem:
      s ∈ SERVICES not theorem:
        s 4 (s 4):
          st = FAIL_IGNORE_3:
            st = FAIL_IGNORE_3:
              st = FAIL_IGNORE_3:
                act1: serviceState_4 = serviceState_4 \ {s ↦ FAIL_IGNORE_4} ∪ {s ↦ FAIL_IGNORE_4}:
```

IGNORE:  

```plaintext
IGNORE s (FAIL_IGNORE 4) {s ↦ FAIL_IGNORE_4}:
  IGNORE s (FAIL_IGNORE 4) {s ↦ FAIL_IGNORE_4}:
    s ∈ SERVICES not theorem:
      s ∈ SERVICES not theorem:
        s 4 (s 4):
          st = FAIL_IGNORE_3:
            st = FAIL_IGNORE_3:
              st = FAIL_IGNORE_3:
                act1: serviceState_4 = serviceState_4 \ {s ↦ FAIL_IGNORE_4} ∪ {s ↦ FAIL_IGNORE_4}:
```

REDEPLOY:  

```plaintext
REDEPLOY:
```
\[ \text{HERE} \quad \begin{array}{c}
\text{IF} \quad Y \in \text{SERVICES} \quad \text{not theorem} \quad > \\
\text{THEN} \quad a(Y) \quad \text{meaningless} \\
\text{END} \\
\end{array} \]

\[ \text{HERE} \quad \begin{array}{c}
\text{IF} \quad Y \in \text{SERVICES} \quad \text{not theorem} \quad > \\
\text{THEN} \quad a(Y) \quad \text{meaningless} \\
\text{END} \\
\end{array} \]
MACHINE M05
REFINES M04
SEES C05
VARIABLES
  serviceState_4
  num_run
  num_susp
INVARINTS
  inv1: num_run ∈ SERVICES → N
  inv2: num_susp ∈ SERVICES → N
  inv3: ∀ s, st · s ∈ SERVICES ∧ st ∈ STATES_4 ∧ st ∉ {FAIL_4, FAIL_DETECT_4} ∧ s ↦ st ∈ serviceState_4 ⇒ num_susp(s) = 0
  inv4: ∀ s · s ∈ SERVICES ∧ s ↦ RUN_4 ∈ serviceState_4 ⇒ num_susp(s) = 0
  inv5: ∀ s · s ∈ SERVICES ∧ s ↦ FAIL_CONFIG_4 ∈ serviceState_4 ⇒ num_run(s) < min_inst(s)
  inv6: ∀ s ∈ SERVICES ⇒ num_susp(s) < num_run(s)
EVENTS
INITIALISATION: extended ordinary
  THEN
  act1: serviceState_4 ≔ InitState_4
  act2: num_run ≔ init_inst
  act3: num_susp ≔ SERVICES×{0}
END
FAIL: extended ordinary
REFINES FAIL
ANY
  s
nb_fail
WHERE
  grd1: s ∈ SERVICES
  grd2: s ↦ RUN_4 ∈ serviceState_4
  grd3: nb_fail ∈ N
  grd4: nb_fail < num_run(s)
THEN
  act1: serviceState_4 ≔ (serviceState_4∖{s ↦ RUN_4}) ∪ {s ↦ FAIL_4}
  act2: num_susp(s) ≔ nb_fail
END
FAIL_DETECT: extended ordinary
REFINES FAIL_DETECT
M05

\textbf{FAIL}\_\textbf{ACTI}'\textbf{V}: extended ordinary
  \textbf{REFINES} FAIL\_\textbf{ACTIV}
\begin{enumerate}
  \item \textbf{ANY}\textbf{ s >}
  \item \textbf{WHERE}
  \begin{enumerate}
    \item grd1: \textbf{s} \in \textbf{SERVICES} not \textbf{theorem} >
    \item grd2: \textbf{s} \mapsto \textbf{FAIL\_DETECT\_4} \in \textbf{serviceState\_4} not \textbf{theorem} >
    \item grd3: \textbf{num\_susp(s)} > 0 not \textbf{theorem} >
  \end{enumerate}
  \item \textbf{THEN}
  \begin{enumerate}
    \item act1: serviceState\_4 = (serviceState\_4 \\setminus \{s \mapsto \textbf{FAIL\_DETECT\_4}\}) \cup \{s \mapsto \textbf{FAIL\_ACTIV\_4}\}
    \item act2: num\_run(s) = num\_run(s) - num\_susp(s)
    \item act3: num\_susp(s) = 0
  \end{enumerate}
\end{enumerate}
\textbf{FAIL\_CONF}\textbf{IGURE}: extended ordinary
  \textbf{REFINES} FAIL\_\textbf{CONF\_IGURE}
\begin{enumerate}
  \item \textbf{FAIL\_ACTI}'\textbf{V}: extended ordinary
  \item \textbf{FAIL\_ACTIV}
\end{enumerate}
M05

s >
WHERE
  grd1: s ∈ SERVICES not theorem
  grd2: s ↦ FAIL_ACTIV_4 ∈ serviceState_4 not theorem
  grd3: num_run(s) < min_inst(s) not theorem
THEN
  act1: serviceState_4 ≔ (serviceState_4 \ {s ↦ FAIL_ACTIV_4}) ∪ {s ↦ FAIL_CONFIG_4}
END

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

IGNORE: extended ordinary
  REFINES
    IGNORE
  ANY
  s >
WHERE
  grd1: s ∈ SERVICES not theorem
  grd2: s ↦ FAIL_IGN_4 ∈ serviceState_4 not theorem
  grd3: new_run ∈ :1 not theorem
  grd4: new_run ≥ min_inst(s) not theorem
THEN
  act1: serviceState_4 ≔ (serviceState_4 \ {s ↦ FAIL_IGN_4}) ∪ {s ↦ RUN_4}
END

REDEPLOY: extended ordinary
  REFINES
    REDEPLOY
  ANY
    s >
new_run >
WHERE
  grd1: s ∈ SERVICES not theorem
  grd2: s ↦ FAIL_CONFIG_4 ∈ serviceState_4 not theorem
  grd3: new_run ∈ N1 not theorem
  grd4: new_run ≥ min_inst(s) not theorem
THEN
  act1: serviceState_4 ≝ (serviceState_4 \ \{s ↦ FAIL_CONFIG_4\}) \cup \{s ↦ DPL_4\}
  act2: num_run(s) = new_run
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\}) \cup \{s ↦ RUN_4\}
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
  gluing_run2: ∀ s · s ∈ SERVICES ⇒ num_run(s) = card(run_peers(s)) not theorem
  gluing_susp1: ∀ s · s ∈ SERVICES ∧ s ∈ dom(susp_peers) ⇒ finite(susp_peers(s)) not theorem
  gluing_susp2: ∀ s · s ∈ SERVICES ∧ s ∈ dom(susp_peers) ⇒ num_susp(s) = card(susp_peers(s)) not theorem
  inv4: ∀ s · s ∈ SERVICES ⇒ run_peers(s) ∩ fail_peers[s] = ∅ not theorem
  inv5: ∀ s · s ∈ SERVICES ∧ s ∈ dom(susp_peers) ⇒ susp_peers(s) ⊆ run_peers(s) not theorem
  inv6: ∀ s, st · s ∈ SERVICES ∧ st ∈ STATES_4 ∧ st ∈ {FAIL_4, FAIL_DETECT_4} ∧ s → st ∈ serviceState_4 ⇒ s ∈ dom(susp_peers) 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
\[ s \rightarrow fp \]

**WHERE**

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

**WITH**

- **nb_fail**: \( nb\_fail = \text{card}(fp) \)

**THEN**

- **act1**: \( \text{serviceState}_4 \leftarrow (\text{serviceState}_4 \setminus \{s \mapsto \text{RUN}_4\}) \cup \{s \mapsto \text{FAIL}_4\} \)
- **act2**: \( \text{susp_peers}(s) \leftarrow fp \)

**END**

**FAIL\_DETECT**: not extended ordinary

**REFINES**

**FAIL\_DETECT**

**ANY**

- **s**
- **sf**

**WHERE**

- **grd1**: \( s \in SERVICES \) not theorem
- **grd2**: \( s \mapsto \text{FAIL}_4 \in \text{serviceState}_4 \) not theorem
- **grd5**: \( \text{susp\_peers}(s) \neq \emptyset \) not theorem
- **grd6**: \( sf \subseteq \text{PEERS} \) not theorem
- **grd7**: \( sf \subseteq \text{susp\_peers}(s) \) not theorem

**WITH**

- **num\_safe**: \( num\_safe = \text{card}(sf) \)

**THEN**

- **act1**: \( \text{serviceState}_4 \leftarrow (\text{serviceState}_4 \setminus \{s \mapsto \text{FAIL\_DETECT}_4\}) \cup \{s \mapsto \text{FAIL\_DETECT}_4\} \)
- **act2**: \( \text{susp\_peers}(s) \leftarrow \text{susp\_peers}(s) \setminus sf \)

**END**

**IS\_OK**: not extended ordinary

**REFINES**

**IS\_OK**

**ANY**

- **s**

**WHERE**

- **grd1**: \( s \in SERVICES \) not theorem
- **grd2**: \( s \mapsto \text{FAIL\_DETECT}_4 \in \text{serviceState}_4 \) not theorem
- **grd5**: \( \text{susp\_peers}(s) = \emptyset \) not theorem

**THEN**

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

**Page 2**
FAIL_ACTIV: not extended ordinary
REFINES FAIL_ACTIV
ANY s
WHERE
  grd1: s ∈ SERVICES not theorem
  grd2: s ↦ FAIL_DETECT_4 ∈ serviceState_4 not theorem
  grd5: susp_peers(s) ≠ ∅ not theorem
THEN
  act1: serviceState_4 = (serviceState_4 \ {s ↦ FAIL_DETECT_4}) ∪ {s ↦ FAIL_ACTIV_4}
  act2: run_peers(s) = run_peers(s) \ susp_peers(s)
  act3: susp_peers(s) = ∅
  act4: fail_peers = fail_peers ∪ ({s}×susp_peers(s))
END

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

FAIL.Ignore: not extended ordinary
REFINES FAIL.Ignore
ANY s
WHERE
  grd1: s ∈ SERVICES not theorem
  grd2: s ↦ FAIL_ACTIV_4 ∈ serviceState_4 not theorem
  grd3: card(run_peers(s)) ≥ min_inst(s) not theorem
THEN
  act1: serviceState_4 = (serviceState_4 \ {s ↦ FAIL_ACTIV_4}) ∪ {s ↦ FAIL_IGNORE_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

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

HEAL: extended ordinary
REFINES
    HEAL
ANY
    s →
WHERE
    grd1: s ∈ SERVICES not theorem
    grd2: s ↦ DPL_4 ∈ serviceState_4 not theorem
THEN
    act1: serviceState_4 = (serviceState_4 \ {s ↦ DPL_4}) ∪ {s ↦ RUN_4}
END
UNFAIL_PEER:  not extended ordinary
  ANY
    s  >
    p  >
WHERE
  grd1:  s ∈ SERVICES not theorem
  grd2:  p ∈ PEERS not theorem
  grd3:  s ↦ p ∈ fail_peers not theorem
THEN
  act1:  fail_peers = fail_peers\{s ↦ p\}
END
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 ≔ ø
    act4: fail_peers ≔ ø
    act5: dep_inst ≔ ø
  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

REFINES
FAIL_DETECT

ANY
s
sf

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

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

END

IS_OK: extended ordinary

REFINES
IS_OK

ANY
s

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

THEN
act1: serviceState_4 = (serviceState_4 \ {s ↦ FAIL_DETECT_4}) ∪ {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}) ∪ {s ↦ FAIL_ACTIV_4}
act2: run_peers(s) = run_peers(s) \ susp_peers(s)
act3: susp_peers(s) = ∅
act4: fail_peers = fail_peers ∪ ({s}×susp_peers(s))
FAIL_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
M07

grd1: s ∈ SERVICES not theorem
grd2: dep ⊆ PEERS not theorem
grd3: finite(dep) not theorem
grd4: dep ∩ run_peers(s) = ∅ not theorem
grd5: dep ∩ fail_peers[{s}] = ∅ not theorem
grd6: card(dep) = deplo_inst(s) not theorem
grd7: card(dep_inst[{s}]) + card(run_peers(s)) < min_ins t(s) not theorem

grd8: s ↦ FAIL_CONFIG_4 ∈ serviceState_4 not theorem

THEN

act1: dep_inst = dep_inst u ({s}×dep)

END

REDEPLOY: not extended ordinary

REFINES
REDEPLOY

ANY

s

WHERE

grd1: s ∈ SERVICES not theorem
grd2: s ↦ FAIL_CONFIG_4 ∈ serviceState_4 not theorem
grd6: dep_inst[{s}] ≠ ∅ not theorem
grd4: card(run_peers(s))+card(dep_inst[{s}]) ≥ min_ins t(s) not theorem

WITH

new_inst: new_inst=dep_inst[{s}]

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} ⩤ 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
M07

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

IN guyants
  inv1: token_owner ∈ SERVICES → PEERS not theorem
  inv2: unav_peers ⊆ PEERS not theorem
  inv3: ∀ s ∈ SERVICES ⇒ token_owner(s) ∈ run_peers(s) \ unav_peers
  inv4: ∀ s ∈ SERVICES ∧ s ∈ dom(susp_peers) ⇒ token_owner(s) ∈ susp_peers(s)
  inv5: susp_inst ∈ PEERS ↔ (SERVICES×PEERS) not theorem
  inv6: ∀ ld, s ∈ PEERS ∧ s ∈ SERVICES ∧ s ∈ dom(susp_inst[ld])
  ⇒ ld = token_owner(s) not theorem
  inv7: ∀ ld, s ∈ PEERS ∧ s ∈ SERVICES ∧ s ∈ dom(susp_inst[ld]) ∧
  ld = token_owner(s) ⇒ ld ∈ susp_inst[ld][s] not theorem
  inv8: ∀ ld, s, stt ∈ PEERS ∧ s ∈ SERVICES ∧ s ∈ dom(susp_inst[ld]) ∧
  ld = token_owner(s) ⇒ susp_inst[ld][s] ⊂ run_peers(s) not theorem
  inv9: ∀ ld, s, stt ∈ 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

WHERE

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

new value for token owner per service if needed

\[ \text{grd4: } \forall \text{srv} \cdot \text{srv} \in \text{SERVICES} \land \text{token_owner}(\text{srv}) \in \text{prs} \Rightarrow E(\text{srv}) = \text{token_owner}(\text{srv}) \not\text{ theorem } \]
\[ \text{If the token owner of a service srv does not belong to prs, the token owner is not changed} \]
\[ \text{grd5: } \forall \text{srv} \cdot \text{srv} \in \text{SERVICES} \land \text{token_owner}(\text{srv}) \in \text{prs} \land \text{srv} \notin \text{dom}(\text{susp_peers}) \Rightarrow E(\text{srv}) \in \text{run_peers}(\text{srv}) \setminus (\text{unav_peers} \cup \text{prs} \cup \text{fail_peers}[[\text{srv}]]) \not\text{ theorem } \]
\[ \text{if the owner of the token for a service becomes unavailable and the service is not suspicious,} \]

then a new token owner among available peers is chosen

\[ \text{grd6: } \forall \text{srv} \cdot \text{srv} \in \text{SERVICES} \land \text{token_owner}(\text{srv}) \in \text{prs} \land \text{srv} \notin \text{dom}(\text{susp_peers}) \Rightarrow E(\text{srv}) \in \text{run_peers}(\text{srv}) \setminus (\text{unav_peers} \cup \text{prs} \cup \text{fail_peers} \setminus \text{run_peers}[[\text{srv}]]) \not\text{ theorem } \]
\[ \text{if the owner of the token for a service becomes unavailable, and the service possess suspicious instances, then a new token owner among available and not} \]

suspicious peers is chosen

THEN

\[ \text{act1: } \text{unav_peers} = \text{unav_peers} \cup \text{prs} \not\text{ theorem } \]
\[ \text{the peers in prs become unavailable} \]
\[ \text{act2: } \text{token_owner} = \text{token_owner} \setminus E \not\text{ theorem } \]
\[ \text{new value for token owner per service is given if needed} \]
\[ \text{act3: } \text{susp_inst} = \text{prs} \setminus \text{susp_inst} \not\text{ theorem } \]
\[ \text{the peers in prs can not suspect instances anymore} \]

END

SUSPECT_INST: not extended ordinary

ANY

\[ s \not\text{ a service s} \]
\[ \text{susp} \not\text{ suspicious instances} \]

WHERE

\[ \text{grd1: } s \in \text{SERVICES} \not\text{ theorem } \]
\[ \text{grd2: } \text{susp} \subseteq \text{PEERS} \not\text{ theorem } \]
\[ \text{grd3: } \text{susp} = \text{run_peers(s)} \land \text{unav_peers} \not\text{ theorem } \]
\[ \text{instances in susp are suspicious if the peers running them becomes unavailable} \]
\[ \text{grd4: } s \in \text{dom}(\text{susp_inst}[[\text{token_owner}(s)]]) \not\text{ theorem } \]
\[ \text{the member of susp have not yet been suspected for s by the token owner of s} \]
\[ \text{grd5: } s \mapsto \text{RUN_4} \in \text{serviceState_4} \not\text{ theorem } \]
\[ \text{the state of s is OK} \]

Page 2
THEN

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

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

END

FAIL: not extended ordinary

REFINES

FAIL

ANY

s

WHERE

grd1: s ∈ SERVICES not theorem

grd2: s ↦ RUN 4 ∈ serviceState_4 not theorem

grd3: susp_inst[{{token_owner(s)}}][{s}] ≠ ∅ not theorem

WITH

fp: fp = susp_inst[{{token_owner(s)}}][{s}]

THEN

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

act2: susp_peers(s) := susp_inst[{{token_owner(s)}}][{s}]

act3: susp_inst := susp_inst \ ({s} ↦ ran(susp_inst))

END

FAIL_DETECT: extended ordinary

REFINES

FAIL_DETECT

ANY

s

sf

WHERE

grd1: s ∈ SERVICES not theorem

grd2: s ↦ FAIL_4 ∈ serviceState_4 not theorem

grd5: susp_peers(s) ≠ ∅ not theorem

grd6: sf ⊆ PEERS not theorem

grd7: sf ⊆ susp_peers(s) not theorem

THEN

act1: serviceState_4 := (serviceState_4 \ {s ↦ FAIL_4}) ∪ {s ↦ FAIL_DETECT_4}

act2: susp_peers(s) := susp_peers(s) \ sf

END

IS_OK: extended ordinary

REFINES

IS_OK

ANY

s

WHERE

grd1: s ∈ SERVICES not theorem
M08

grd2: \( s \mapsto \text{FAIL\_DETECT\_4} \in \text{serviceState\_4} \text{ not theorem} \) 
grd5: \( \text{susp\_peers}(s) = \emptyset \text{ not theorem} \)

THEN

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

END

FAIL\_ACTIV: extended ordinary

REFINES

FAIL\_ACTIV

ANY

s

WHERE

grd1: \( s \in \text{SERVICES} \text{ not theorem} \) 
grd2: \( s \mapsto \text{FAIL\_DETECT\_4} \in \text{serviceState\_4} \text{ not theorem} \) 
grd5: \( \text{susp\_peers}(s) \neq \emptyset \text{ not theorem} \)

THEN

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

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: \( s \in \text{SERVICES} \text{ not theorem} \) 
grd2: \( s \mapsto \text{FAIL\_ACTIV\_4} \in \text{serviceState\_4} \text{ not theorem} \) 
grd3: \( \text{card}(\text{run\_peers}(s)) < \text{min\_inst}(s) \text{ not theorem} \)

THEN

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

END

FAIL\_IGNORE: extended ordinary

REFINES

FAIL\_IGNORE

ANY

s

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: \( serviceState_4 = (serviceState_4 \setminus \{s \mapsto FAIL_ACTIV_4\}) \cup \{s \mapsto FAIL_IGN_4\} \) 

\( \text{END} \)

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

\( \text{REFINES } \)

\( \text{IGNORE } \)

\( \text{ANY } \)

\( s \)

\( \text{WHERE } \)

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

\( \text{grd2: } s \mapsto FAIL_IGN_4 \in \text{serviceState}_4 \not\text{ theorem} \)

\( \text{THEN } \)

\( \text{act1: } serviceState_4 = (serviceState_4 \setminus \{s \mapsto FAIL_IGN_4\}) \cup \{s \mapsto RUN_4\} \) 

\( \text{END} \)

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

\( \text{REFINES } \)

\( \text{REDEPLOY_INST } \)

\( \text{ANY } \)

\( s \)

\( \text{dep } \)

\( \text{WHERE } \)

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

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

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

\( \text{grd4: } \text{dep} \cap \text{run_peers}(s) = \emptyset \not\text{ theorem} \)

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

\( \text{grd6: } \text{card} (\text{dep}) = \text{deplo_inst}(s) \not\text{ theorem} \)

\( \text{grd7: } \text{card} (\text{dep}_{\text{inst}}(s)) + \text{card} (\text{run_peers}(s)) < \text{min}_\text{inst}(s) \not\text{ theorem} \)

\( \text{grd8: } s \mapsto FAIL_\text{CONFIG}_4 \in serviceState_4 \not\text{ theorem} \)

\( \text{THEN } \)

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

\( \text{END} \)

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

\( \text{REFINES } \)

\( \text{REDEPLOY } \)

\( \text{ANY } \)

\( s \)

\( \text{WHERE } \)

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

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

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

\( \text{grd4: } \text{card} (\text{run_peers}(s)) + \text{card} (\text{dep}_{\text{inst}}([s])) \geq \text{min}_\text{inst}(s) \not\text{ theorem} \)
THEN
act1: \( \text{serviceState}_4 \equiv (\text{serviceState}_4 \setminus \{s \mapsto \text{FAIL\_CONFIG}_4\}) \cup \{s \mapsto \text{DPL}_4\} \)
act2: \( \text{run\_peers}(s) = \text{run\_peers}(s) \cup \text{dep\_inst}[\{s\}] \)
act3: \( \text{dep\_inst} = \{s\} \leftarrow \text{dep\_inst} \)
END

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

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

MAKE\_PEER\_AVAIL: not extended ordinary
ANY
p
WHERE
grd1: \( p \in \text{PEERS} \) not theorem
grd2: \( p \in \text{unav\_peers} \) not theorem
THEN
act1: \( \text{unav\_peers} = \text{unav\_peers} \setminus \{p\} \)
END
END
MACHINE M09
REFINES M08
SEES C08
VARIABLES
  serviceState_4
  run_peers
  susp_peers
  fail_peers
  dep_inst
  token_owner
  unav_peers
  susp_inst
  rec_inst
  rct_inst

INvariants
inv1: rec_inst ∈ PEERS ↔ (SERVICES×PEERS) not theorem
inv2: rct_inst ∈ PEERS ↔ (SERVICES×PEERS) not theorem
inv3: ∀ ld, s · ld ∈ PEERS ∧ s ∈ SERVICES ∧ rct_inst[ld][s] ≠ ∅ ⇒ rec_inst[ld][s] ≠ ∅ not theorem
inv4: ∀ ld, s · ld ∈ PEERS ∧ s ∈ SERVICES ∧ rct_inst[ld][s] ≠ ∅ ⇒ rct_inst[ld][s] ⊆ rec_inst[ld][s] not theorem
inv5: ∀ ld, s · ld ∈ PEERS ∧ s ∈ SERVICES ∧ s ∈ dom(rec_inst[ld]) ⇒ ld = token_owner(s) not theorem
inv6: ∀ ld, s · ld ∈ PEERS ∧ s ∈ SERVICES ∧ s ∈ dom(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 := ∅
act10: \[ \text{rec\_inst} = \emptyset \]
act11: \[ \text{rct\_inst} = \emptyset \]

MAKE_PEER_UNAVAIL: extended ordinary

REFINES
MAKE_PEER_UNAVAIL

ANY
prs >
E > new values for token owner per service if needed

WHERE
grd1: prs ⊆ \text{PEERS} \not\text{theorem} >
grd2: prs \not\subseteq \text{unav\_peers} \not\text{theorem} >
grd3: E \in \text{SERVICES} \rightarrow \text{PEERS} \not\text{theorem} > new value for token owner per service if needed
grd4: \forall \text{srv} \cdot \text{srv} \in \text{SERVICES} \land \text{token\_owner(srv)} \notin prs \Rightarrow E(srv) = \text{token\_owner(srv)} \not\text{theorem} > \text{If the token owner of a service srv does not belong to prs, the token owner is not changed}
grd5: \forall \text{srv} \cdot \text{srv} \in \text{SERVICES} \land \text{token\_owner(srv)} \in 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} > \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}
grd6: \forall \text{srv} \cdot \text{srv} \in \text{SERVICES} \land \text{token\_owner(srv)} \in 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} > \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}

THEN
act1: unav\_peers = unav\_peers \cup prs \rightarrow \text{the peers in prs become unavailable}
act2: token\_owner = token\_owner \leftrightarrow E \rightarrow \text{new value for token owner per service is given if needed}
act3: susp\_inst = prs \Leftarrow susp\_inst \rightarrow \text{the peers in prs can not suspect instances anymore}
act4: rec\_inst = prs \Leftarrow rec\_inst \rightarrow
act5: rct\_inst = prs \Leftarrow rct\_inst \rightarrow

END

SUSPECT_INST: extended ordinary

REFINES

Page 2
M09

SUSPECT_INST

ANY
s → a service s
susp → suspicious instances

WHERE

grd1: s ∈ SERVICES not theorem
grd2: susp ⊆ PEERS not theorem

in susp are suspicious if the peers running them becomes unavailable

grd3: susp = run_peers(s) n unav_peers not theorem

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

grd4: s ∈ dom(suspInst{{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

is OK

THEN

act1: suspInst = suspInst ∪ ({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: suspInst{{token_owner(s)}}[s] ≠ ø not theorem

the state of s is OK

THEN

act1: serviceState_4 = (serviceState_4∖{s ↦ RUN_4}) ∪ {s ↦ FAIL_4}

act2: suspPeers(s) = suspInst{{token_owner(s)}}[s]

act3: suspInst = suspInst ⩥ (ran(suspInst))

END

RECONTACT_INST_OK:

not extended ordinary

ANY
s → a service s
i → an instance i

WHERE

grd1: s ∈ SERVICES not theorem
grd2: i ∈ PEERS not theorem
grd3: s ↦ FAIL_4 ∈ serviceState_4 not theorem

the state of s is SUSPICIOUS

grd4: suspPeers(s) ≠ ø not theorem

the set of suspicious peers for s is not empty

grd5: i ∈ suspPeers(s)∖unav_peers not theorem

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

grd6: token_owner(s) ⇒ (s ↦ i) ∈ rec_inst not theorem

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

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

THEN

act1: \( \text{rec}_\text{inst} = \text{rec}_\text{inst} \cup \{\text{token}_\text{owner}(s) \leftrightarrow (s \leftrightarrow i)\} \) \> the token owner of s has tried to recontact i

act2: \( \text{rec}_\text{inst} = \text{rct}_\text{inst} \cup \{\text{token}_\text{owner}(s) \leftrightarrow (s \leftrightarrow i)\} \) \> i is recontacted by the token owner of s successfully

END

RECONTACT_INST_KO: not extended ordinary

ANY

\( s \) \> a service s
\( i \) \> an instance i

WHERE

\[ \text{grd1: } s \in \text{SERVICES} \] not theorem
\[ \text{grd2: } i \in \text{PEERS} \] not theorem
\[ \text{grd3: } s \rightarrow \text{FAIL}_4 \in \text{serviceState}_4 \] not theorem 

the state of s is SUSPICIOUS

\[ \text{grd4: } \text{susp}_\text{peers}(s) \neq \emptyset \] not theorem 

the set of suspicious peers for s is not empty

\[ \text{grd5: } i \in \text{susp}_\text{peers}(s) \cap \text{unav}_\text{peers} \] not theorem 

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

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

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

THEN

act1: \( \text{rec}_\text{inst} = \text{rec}_\text{inst} \cup \{\text{token}_\text{owner}(s) \leftrightarrow (s \leftrightarrow i)\} \) \> the token owner of s has tried to recontact i

END

FAIL_DETECT: not extended ordinary

REFINES

FAIL_DETECT

ANY

\( s \)

WHERE

\[ \text{grd1: } s \in \text{SERVICES} \] not theorem
\[ \text{grd2: } s \rightarrow \text{FAIL}_4 \in \text{serviceState}_4 \] not theorem
\[ \text{grd5: } \text{susp}_\text{peers}(s) \neq \emptyset \] not theorem
\[ \text{grd8: } \text{rec}_\text{inst}([\text{token}_\text{owner}(s)])[[s]] = \text{susp}_\text{peers}(s) \] not theorem

WITH

\( sf \): \( sf=\text{rct}_\text{inst}([\text{token}_\text{owner}(s)])[[s]] \) 

THEN
**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
  grd3: card(run_peers(s)) < min_inst(s) not theorem

THEN

act1: serviceState_4 ≔ (serviceState_4 \ {s ↦ FAIL_DETECT_4}) u {s ↦ FAIL_ACTIV_4}

act2: run_peers(s) = run_peers(s) \ susp_peers(s)

act3: susp_peers(s) = ∅

act4: fail_peers = fail_peers u (s) x 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
  grd3: card(run_peers(s)) < min_inst(s) not theorem

END
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: extended ordinary ›
REFINES REDEPLOY_INST
ANY s ›
WHERE
  grd1: s ∈ SERVICES not theorem ›
  grd2: dep ⊆ PEERS not theorem ›
  grd3: finite(dep) not theorem ›
  grd4: dep ∩ run_peers(s) = ∅ not theorem ›
  grd5: dep ∩ fail_peers[{s}] = ∅ not theorem ›
  grd6: card(dep) = deplo_inst(s) not theorem ›
  grd7: card(dep_inst[{s}]) + card(run_peers(s)) < min_inst(s) not theorem ›
  grd8: s ↦ FAIL_CONFIG_4 ∈ serviceState_4 not theorem ›
THEN  
act1: \[ \text{dep}\_\text{inst} = \text{dep}\_\text{inst} \cup \{s\times\text{dep}\} \]  
END

REDEPLOY: extended ordinary

REFINES
REDEPLOY
ANY
s >
WHERE
grd1: \[ s \in \text{SERVICES} \text{ not theorem} \]  
grd2: \[ s \mapsto \text{FAIL}\_\text{CONFIG}\_4 \in \text{serviceState}\_4 \text{ not theorem} \]  
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: \[ \text{serviceState}\_4 = (\text{serviceState}\_4 \setminus \{s \mapsto \text{FAIL}\_\text{CONFIG}\_4\}) \]  
act2: \[ \text{run}\_\text{peers}(s) = \text{run}\_\text{peers}(s) \cup \text{dep}\_\text{inst}[\{s\}] \]  
act3: \[ \text{dep}\_\text{inst} = \{s\} \ltimes \text{dep}\_\text{inst} \]  
END

HEAL: extended ordinary

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

UNFAIL_PEER: extended ordinary

REFINES
UNFAIL_PEER
ANY
s >  
p >
WHERE
grd1: \[ s \in \text{SERVICES} \text{ not theorem} \]  
grd2: \[ p \in \text{PEERS} \text{ not theorem} \]  
grd3: \[ s \mapsto p \in \text{fail}\_\text{peers} \text{ not theorem} \]  
THEN
act1: \[ \text{fail}\_\text{peers} = \text{fail}\_\text{peers}\setminus\{s \mapsto p\} \]  
END
MAKE_PEEER_AVAIL: extended ordinary
REFINES
  MAKE_PEEER_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
  rct_inst
  actv_inst

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

EVENTS

INITIALISATION: extended ordinary

THEN
  act1: serviceState_4 ≔ InitState_4
  act2: run_peers ≔ InitSrvcPeers
  act3: susp_peers = ∅
  act4: fail_peers = ∅
  act5: dep_inst = ∅
  act6: token_owner = init_tok
  act7: unav_peers = ∅
  act8: susp_inst = ∅
  act10: rec_inst = ∅
  act11: rct_inst = ∅
  act12: actv_inst = ∅
MAKE_PEER_UNAVAIL:  extended ordinary →
REFINES
MAKE_PEER_UNAVAIL
ANY
prs
E → new values for token owner per service if needed
WHERE
grd1: prs ⊆ PEERS not theorem →
grd2: prs ∉ unav_peers not theorem →
grd3: E ∈ SERVICES → PEERS not theorem → new value for token
owner per service if needed
grd4: ∀ srv · srv ∈ SERVICES ∧ token_owner(srv) ∉ prs ⇒ E (srv) = token_owner(srv) not theorem → If the token owner of a service srv does
not belong to prs, the token owner is not changed
grd5: ∀ srv · srv ∈ SERVICES ∧ token_owner(srv) ∈ prs ∧ srv ∉
dom(susp_peers) ⇒ E(srv) ∈ run_peers(srv)\(\text{unav_peers ∪ prs ∪ fail_peers}[[\{srv\}]]) not theorem → if the owner of the token for a service
becomes unavailable and the service is not suspicious,

then a new token owner among available peers is chosen
grd6: ∀ srv · srv ∈ SERVICES ∧ token_owner(srv) ∈ prs ∧ srv ∈
dom(susp_peers) ⇒ E(srv) ∈ run_peers(srv)\(\text{unav_peers ∪ prs ∪ susp_peers(srv) ∪ fail_peers}[[\{srv\}]]) not theorem → if the owner of the token for a service becomes
unavailable, and the service
possess suspicious instances, then a new token owner among available and not
suspicous peers is chosen
THEN
act1: unav_peers = unav_peers ∪ prs → the peers in prs become
unavailable
act2: token_owner = token_owner ∪ E → new value for token owner
per service is given if needed
act3: susp_inst = prs ∖ susp_inst → the peers in prs can not
suspect instances anymore
act4: rec_inst = prs ∖ rec_inst
act5: rct_inst = prs ∖ rct_inst
act6: actv_inst = prs ∖ actv_inst →
END

SUSPECT_INST:  extended ordinary →
REFINES
SUSPECT_INST
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 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} ◁ 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:  \( \text{token\_owner}(s) \rightarrow (s \rightarrow i) \notin \text{rec\_inst} \) not theorem \( \Rightarrow \) the token owner of \( s \) has not yet tried to recontact \( i \)

grd7:  \( \text{rec\_inst}[[\text{token\_owner}(s)]][[s]] \subset \text{susp\_peers}(s) \) not theorem \( \Rightarrow \) the token owner of \( s \) has not yet tried to recontact all the suspicious instances of \( s \)

THEN

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

act2:  \( \text{rct\_inst} = \text{rct\_inst} \cup \{\text{token\_owner}(s) \rightarrow (s \rightarrow i)\} \) \( \Rightarrow \) \( i \) is recontacted by the token owner of \( s \) successfully

END

RECONTACT\_INST\_KO:    extended ordinary \( \Rightarrow \)
REFINES
RECONTACT\_INST\_KO
ANY
s \( \rightarrow \) 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:  \( s \rightarrow \text{FAIL\_4} \in \text{serviceState\_4} \) not theorem \( \Rightarrow \) the state of \( s \) is SUSPICIOUS

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

grd5:  \( i \in \text{susp\_peers}(s) \cap \text{unav\_peers} \) not theorem \( \Rightarrow \) \( i \) is a suspicious instance of \( s \) and is unavailable (can not be contacted)

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

grd7:  \( \text{rec\_inst}[[\text{token\_owner}(s)]][[s]] \subset \text{susp\_peers}(s) \) not theorem \( \Rightarrow \) the token owner of \( s \) has not yet tried to recontact all the suspicious instances of \( s \)

THEN

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

END

FAIL\_DETECT:    extended ordinary \( \Rightarrow \)
REFINES
FAIL\_DETECT
ANY
s \( \rightarrow \) a service \( s \)
WHERE

grd1:  \( s \in \text{SERVICES} \) not theorem \( \Rightarrow \)
grd2:  \( s \rightarrow \text{FAIL\_4} \in \text{serviceState\_4} \) 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) = \ø 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) ≠ \ø not theorem

THEN

act1: serviceState_4 ≔ (serviceState_4 \ {s ↦ FAIL_DETECT_4})

u \{s ↦ FAIL_ACTIV_4\}

act2: run_peers(s) ≔ run_peers(s) \ susp_peers(s)

act3: susp_peers(s) ≔ \ø

act4: fail_peers ≔ fail_peers \{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

Page 5
M10

\[ \text{grd3: card(run_peers(s)) < min_inst(s) not theorem} \]
\[ \text{THEN} \]
\[ \text{act1: serviceState_4 = (serviceState_4 \{ s \mapsto FAIL_ACTIV_4 \}) \cup \{ s \mapsto FAIL_CONFIG_4 \} \} \]
\[ \text{END} \]

FAIL_IGNORE: extended ordinary

REFINES
FAIL_IGNORE

ANY \[ s \]

WHERE
\[ \text{grd1: s \in SERVICES not theorem} \]
\[ \text{grd2: s \mapsto FAIL_ACTIV_4 \in serviceState_4 not theorem} \]
\[ \text{grd3: card(run_peers(s)) \geq min_inst(s) not theorem} \]

THEN
\[ \text{act1: serviceState_4 = (serviceState_4 \{ s \mapsto FAIL_ACTIV_4 \}) \cup \{ s \mapsto FAIL_IGN_4 \} \} \]
\[ \text{END} \]

IGNORE: extended ordinary

REFINES
IGNORE

ANY \[ s \]

WHERE
\[ \text{grd1: s \in SERVICES not theorem} \]
\[ \text{grd2: s \mapsto FAIL_IGN_4 \in serviceState_4 not theorem} \]

THEN
\[ \text{act1: serviceState_4 = (serviceState_4 \{ s \mapsto FAIL_IGN_4 \}) \cup \{ s \mapsto RUN_4 \} \} \]
\[ \text{END} \]

REDEPLOY_INSTC: not extended ordinary

ANY
\[ s \mapsto \text{a service s} \]
\[ i \mapsto \text{an instance i} \]

WHERE
\[ \text{grd1: s \in SERVICES not theorem} \]
\[ \text{grd2: i \in PEERS not theorem} \]
\[ \text{grd3: i \notin run_peers(s) \cup fail_peers\{s\} \cup unav_peers \cup dep_inst\{s\} not theorem} \]
\[ \text{i does not run s, is not failed for s, is not unavailable and is not already activated for s} \]
\[ \text{grd4: token_owner(s) \mapsto (s \mapsto i) \notin actv_inst not theorem} \]
\[ \text{grd5: s \mapsto FAIL_CONFIG_4 \in serviceState_4 not theorem} \]
\[ \text{grd6: card(actv_inst\{token_owner(s)\}\{s\}) < deplo_inst(s) not theorem} \]
\[ \text{grd7: card(dep_inst\{s\}) + card(run_peers(s)) < min_inst(s)} \]
not theorem THEN
act1: \( \text{actv\_inst} = \text{actv\_inst} \cup \{\text{token\_owner}(s) \mapsto (s \mapsto i)\} \)
END

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

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

\text{HEAL: \ extended ordinary} \>
\text{REFINES}
\text{HEAL}
\text{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: 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
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 owners
  i_state

INVARIANTS
  inv1: i_state ∈ (PEERS × SERVICES) ⇸ STATES_4 not theorem
  inv2: ∀ s · s ∈ SERVICES ⇒ token_owner(s) ↦ s ∈ dom(i_state)
  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
  inv3: ∀ p, s · p ∈ PEERS ∧ s ∈ SERVICES ∧ (p ↦ s) ∈ dom(i_state) ⇒ p = token_owner(s) not theorem

EVENTS

INITIALISATION: not extended ordinary
  THEN
    act2: run_peers = InitSrvcPeers
    act3: susp_peers = ∅
    act4: fail_peers = ∅
    act5: dep_inst = ∅
    act6: token_owner = init_tok
    act7: unav_peers = ∅
    act8: susp_inst = ∅
    act10: rec_inst = ∅
    act11: rct_inst = ∅
    act12: actv_inst = ∅
    act13: i_state = InitStatus

MAKE_PEER_UNAVAIL: not extended ordinary
  REFINES MAKE_PEER_UNAVAIL
  ANY
prs } new values for token owner per service if needed
i_s }

WHERE

grd1: \text{prs} \subseteq \text{PEERS} \not\text{ theorem } \\
grd2: \text{prs} \not\subseteq \text{unav_peers} \not\text{ theorem } \\
grd3: \text{E} \in \text{SERVICES} \rightarrow \text{PEERS} \not\text{ theorem } \rightarrow \text{new value for token owner per service if needed}

grd4: \text{i_s} \in (\text{PEERS} \times \text{SERVICES}) \leftrightarrow \text{STATES}_4 \not\text{ theorem } \\
grd5: \forall \text{srv} \cdot \text{srv} \in \text{SERVICES} \land \text{token_owner}(\text{srv}) \notin \text{prs} \Rightarrow \text{E}(\text{srv}) = \text{token_owner}(\text{srv}) \not\text{ theorem } \rightarrow \text{If the token owner of a service srv does not belong to prs, the token owner is not changed}

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

then a new token owner among available peers is chosen

grd7: \forall \text{srv} \cdot \text{srv} \in \text{SERVICES} \land \text{token_owner}(\text{srv}) \notin \text{prs} \land \text{srv} \notin \text{dom(suspeers)} \Rightarrow \text{E}(\text{srv}) \in \text{run_peers}(\text{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,}

possess suspicious instances, then a new token owner among available and not suspicious 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 = \text{E}(s) \not\text{ theorem } \\
grd9: \forall \text{srv} \cdot \text{srv} \in \text{SERVICES} \Rightarrow (\text{E}(\text{srv}) \ni \text{srv}) \ni \text{i_state} \not\text{ theorem } \\

THEN

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

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
      instances in susp are suspicious if the peers running them becomes unavailable
      grd4:  s ∉ dom(suspect_inst[{token_owner(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
      state of s is OK
      THEN
      act1:  suspect_inst = suspect_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:  suspect_inst[{token_owner(s)}][{s}] ≠ ∅ not theorem
      the state of s is SUSPICIOUS
      grd4:  susp_peers(s) ≠ ∅ not theorem
      the set of suspicious peers for s is not empty
      THEN
      act1:  i_state(token_owner(s) ↦ s) = FAIL_4
      act2:  susp_peers(s) = suspect_inst[{token_owner(s)}][{s}]
      act3:  suspect_inst = suspect_inst ⩥ ({s} ◁ ran(suspect_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
      the state of s is SUSPICIOUS
      grd4:  susp_peers(s) ≠ ∅ not theorem
      the set of suspicious peers for s is not empty

Page 3
M11

\begin{align*}
\text{grd5: } & \quad i \in \text{susp}_\text{peers}(s) \setminus \text{unav}_\text{peers} \quad \text{not theorem} \quad \Rightarrow \quad \text{i is a suspicious instance of s and is available (can be contacted)} \\
\text{grd6: } & \quad \text{token}_\text{owner}(s) \Rightarrow (s \Rightarrow i) \notin \text{rec}_\text{inst} \quad \text{not theorem} \quad \Rightarrow \quad \text{the token owner of s has not yet tried to recontact i} \\
\text{grd7: } & \quad \text{rec}_\text{inst}[(\text{token}_\text{owner}(s))][](s) \subset \text{susp}_\text{peers}(s) \quad \text{not theorem} \quad \Rightarrow \quad \text{the token owner of s has not yet tried to recontact all the suspicious instances of s} \\
\text{THEN} \\
\text{act1: } & \quad \text{rec}_\text{inst} = \text{rec}_\text{inst} \cup \{\text{token}_\text{owner}(s) \Rightarrow (s \Rightarrow i)\} \quad \Rightarrow \quad \text{the token owner of s has tried to recontact i} \\
\text{act2: } & \quad \text{rct}_\text{inst} = \text{rct}_\text{inst} \cup \{\text{token}_\text{owner}(s) \Rightarrow (s \Rightarrow i)\} \quad \Rightarrow \quad \text{i is recontacted by the token owner of s successfully} \\
\text{END} \\
\end{align*}

\text{RECONTACT INST KO: not extended ordinary} \\
\text{REFINES} \\
\text{RECONTACT INST KO} \\
\text{ANY} \\
\quad s \rightarrow \text{a service s} \\
\quad i \rightarrow \text{an instance i} \\
\text{WHERE} \\
\quad \text{grd1: } s \in \text{SERVICES} \quad \text{not theorem} \quad \Rightarrow \quad \text{the state of s is SUSPICIOUS} \\
\quad \text{grd2: } i \in \text{PEERS} \quad \text{not theorem} \quad \Rightarrow \quad \text{peers for s is not empty} \\
\quad \text{grd3: } \text{i}_\text{state}(\text{token}_\text{owner}(s) \Rightarrow s) = \text{FAIL}_4 \quad \text{not theorem} \quad \Rightarrow \quad \text{the set of suspicious peers for s is not empty} \\
\quad \text{grd4: } \text{susp}_\text{peers}(s) \neq \emptyset \quad \text{not theorem} \quad \Rightarrow \quad \text{suspicious instance of s and is unavailable (can not be contacted)} \\
\quad \text{grd6: } \text{token}_\text{owner}(s) \Rightarrow (s \Rightarrow i) \notin \text{rec}_\text{inst} \quad \text{not theorem} \quad \Rightarrow \quad \text{the token owner of s has not yet tried to recontact i} \\
\quad \text{grd7: } \text{rec}_\text{inst}[(\text{token}_\text{owner}(s))][](s) \subset \text{susp}_\text{peers}(s) \quad \text{not theorem} \quad \Rightarrow \quad \text{the token owner of s has not yet tried to recontact all the suspicious instances of s} \\
\text{THEN} \\
\text{act1: } & \quad \text{rec}_\text{inst} = \text{rec}_\text{inst} \cup \{\text{token}_\text{owner}(s) \Rightarrow (s \Rightarrow i)\} \quad \Rightarrow \quad \text{the token owner of s has tried to recontact i} \\
\text{END} \\
\text{FAIL DETECT: not extended ordinary} \\
\text{REFINES} \\
\text{FAIL DETECT} \\
\text{ANY} \\
\quad s \rightarrow \text{a service s} \\
\text{WHERE} \\
\quad \text{grd1: } s \in \text{SERVICES} \quad \text{not theorem} \quad \Rightarrow \quad \text{the state of s is SUSPICIOUS} \\
\quad \text{grd2: } \text{i}_\text{state}(\text{token}_\text{owner}(s) \Rightarrow s) = \text{FAIL}_4 \quad \text{not theorem} \quad \Rightarrow \quad \text{the token owner of s has tried to recontact i} \\
\quad \text{grd5: } \text{susp}_\text{peers}(s) \neq \emptyset \quad \text{not theorem} \quad \Rightarrow \quad \text{suspicious instance of s and is unavailable (can not be contacted)}
M11

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

THEN

act1: \text{i\_state}(\text{token\_owner}(s) \mapsto s) = \text{FAIL\_DETECT\_4} \not\text{ theorem}
act2: \text{susp\_peers}(s) = \text{susp\_peers}(s) \setminus \text{rct}\_\text{inst}[\{\text{token\_owner}(s)\}][\{s\}] \not\text{ theorem}

act3: \text{rec}\_\text{inst} = \text{rec}\_\text{inst} \setminus \{s\} < \text{ran}(\text{rec}\_\text{inst}) \not\text{ theorem}
act4: \text{rct}\_\text{inst} = \text{rct}\_\text{inst} \setminus \{s\} < \text{ran}(\text{rct}\_\text{inst}) \not\text{ theorem}

\text{END}

\text{IS\_OK: } \not\text{ extended ordinary}
\text{REFINES}
\text{IS\_OK}
\text{ANY}
\text{s}
\text{WHERE}
\text{grd1: } s \in \text{SERVICES} \not\text{ theorem}
\text{grd2: } \text{i\_state}(\text{token\_owner}(s) \mapsto s) = \text{FAIL\_DETECT\_4} \not\text{ theorem}

\text{grd5: } \text{susp\_peers}(s) = \emptyset \not\text{ theorem}
\text{THEN}
\text{act1: } \text{i\_state}(\text{token\_owner}(s) \mapsto s) = \text{RUN\_4} \not\text{ theorem}
\text{END}

\text{FAIL\_ACTIV: } \not\text{ extended ordinary}
\text{REFINES}
\text{FAIL\_ACTIV}
\text{ANY}
\text{s}
\text{WHERE}
\text{grd1: } s \in \text{SERVICES} \not\text{ theorem}
\text{grd2: } \text{i\_state}(\text{token\_owner}(s) \mapsto s) = \text{FAIL\_DETECT\_4} \not\text{ theorem}

\text{grd5: } \text{susp\_peers}(s) \neq \emptyset \not\text{ theorem}
\text{THEN}
\text{act1: } \text{i\_state}(\text{token\_owner}(s) \mapsto s) = \text{FAIL\_ACTIV\_4} \not\text{ theorem}
\text{act2: } \text{run\_peers}(s) = \text{run\_peers}(s) \setminus \text{susp\_peers}(s) \not\text{ theorem}
\text{act3: } \text{susp\_peers}(s) = \emptyset \not\text{ theorem}
\text{act4: } \text{fail\_peers} = \text{fail\_peers} \cup \{(s)\times\text{susp\_peers}(s)\} \not\text{ theorem}
\text{END}

\text{FAIL\_CONFIGURE: } \not\text{ extended ordinary}
\text{REFINES}
\text{FAIL\_CONFIGURE}
\text{ANY}
\text{s}
\text{WHERE}
\text{grd1: } s \in \text{SERVICES} \not\text{ theorem}
FAIL_IGNORE: not extended ordinary

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_CONFIG_4

END

FAIL_IGNORE: not extended ordinary

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: not extended ordinary

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: not extended ordinary

REDEPLOY_INSTC

ANY

s
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

THEN

act1: token_owner(s) ↦ (s ↦ i) ∉ actv_inst not theorem

END

Page 6
M11

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

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

\[\text{END}\]

\[\text{REDEPLOY}\_\text{INSTS: } \text{not extended ordinary }\]

\[\text{REFINES } \text{REDEPLOY}\_\text{INSTS}\]

\[\text{ANY } s \]

\[\text{WHERE}\]
\[\text{grd1: } s \in \text{SERVICES } \text{not theorem }\]
\[\text{not theorem }\]
\[\text{grd6: card}(\text{actv}\_\text{inst}[\{\text{token}\_\text{owner}(s)\}][\{s\}]) = \text{deplo}\_\text{inst}(s)\]

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

\[\text{grd8: i}\_\text{state}(\text{token}\_\text{owner}(s) \Rightarrow s) = \text{FAIL}\_\text{CONFIG}\_4 \text{not theorem}\]

\[\text{THEN}\]
\[
\text{act1: } \text{dep}\_\text{inst} = \text{dep}\_\text{inst} \cup \{\{s\} \times \text{actv}\_\text{inst}[\{\text{token}\_\text{owner}(s)\}][\{s\}]\}
\]
\[
\text{act2: } \text{actv}\_\text{inst} = \text{actv}\_\text{inst} \Rightarrow \{\{s\} \Rightarrow \text{ran}(\text{actv}\_\text{inst})\}
\]

\[\text{END}\]

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

\[\text{REFINES } \text{REDEPLOY}\]

\[\text{ANY } s \]

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

\[\text{grd7: } \text{actv}\_\text{inst}[\{\text{token}\_\text{owner}(s)\}][\{s\}]=\emptyset \text{not theorem }\]
\[\text{not theorem }\]
\[\text{grd6: } \text{dep}\_\text{inst}[\{s\}] \neq \emptyset \text{not theorem }\]
\[\text{not theorem }\]
\[\text{grd4: card}(\text{run}\_\text{peers}(s))+\text{card}(\text{dep}\_\text{inst}[\{s\}]) \geq \text{min}\_\text{inst}(s) \text{not theorem}\]

\[\text{THEN}\]
\[
\text{act1: } i\_\text{state}(\text{token}\_\text{owner}(s) \Rightarrow s) = \text{DPL}\_4
\]
\[
\text{act2: } \text{run}\_\text{peers}(s) = \text{run}\_\text{peers}(s) \cup \text{dep}\_\text{inst}[\{s\}]
\]
\[
\text{act3: } \text{dep}\_\text{inst} = \{s\} \Rightarrow \text{dep}\_\text{inst}
\]

\[\text{END}\]

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

\[\text{REFINES } \text{HEAL}\]

\[\text{ANY}\]
M11

\[
\begin{align*}
\text{s} & \\
\text{WHERE} & \\
\text{grd1: } & s \in \text{SERVICES} \not\text{ theorem} \\
\text{grd2: } & i_\text{state}(\text{token_owner}(s) \rightarrow s) = \text{DPL}_4 \not\text{ theorem} \\
\text{THEN} & \\
\text{act1: } & i_\text{state}(\text{token_owner}(s) \rightarrow s) = \text{RUN}_4 \\
\text{END} & \\
\end{align*}
\]

\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\text{ theorem} \\
\text{grd2: } & p \in \text{PEERS} \not\text{ theorem} \\
\text{grd3: } & s \rightarrow p \in \text{fail\_peers} \not\text{ theorem} \\
\text{THEN} & \\
\text{act1: } & \text{fail\_peers} = \text{fail\_peers}\setminus\{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\text{ theorem} \\
\text{grd2: } & p \in \text{unav\_peers} \not\text{ theorem} \\
\text{THEN} & \\
\text{act1: } & \text{unav\_peers} = \text{unav\_peers}\setminus\{p\} \\
\text{END} & \\
\end{align*}
\]
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
  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)
  inv3: ∀ p, s · p ∈ PEERS ∧ s ∈ SERVICES ∧ p = token_owner(s) ⇒ (p ↦ s) ∈ dom(suspc_peers)
  gluing_tok_ow1: ∀ s · s ∈ SERVICES ∧ s ∈ dom(suspc_peers) ⇒ suspc_peers(s) = suspc_peers(token_owner(s) ↦ s)

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

MAKE_PEER_UNAVAIL: not extended ordinary
  REFINES MAKPEER_UNAVAIL
  ANY
    prs
    E new values for token owner per service if needed
M12

WHERE

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

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

grd3: \( E \in \text{SERVICES} \rightarrow \text{PEERS} \) not theorem \( \rightarrow \) 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: \( \forall \text{srv} \cdot \text{srv} \in \text{SERVICES} \land \text{token_owner(srv)} \in \text{prs} \Rightarrow E(\text{srv}) = \text{token_owner(srv)} \) not theorem \( \rightarrow \) if the token owner of a service srv does not belong to prs, the token owner is not changed

grd6: \( \forall \text{srv} \cdot \text{srv} \in \text{SERVICES} \land \text{token_owner(srv)} \in \text{prs} \land \text{token_owner(srv)} \rightarrow \text{srv} \notin \text{dom(suspc_peers)} \Rightarrow E(\text{srv}) \in \text{run_peers(srv)} \setminus (\text{unav_peers} \cup \text{prs} \cup \text{fail_peers}\{\text{srv}\}) \) 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

grd7: \( \forall \text{srv} \cdot \text{srv} \in \text{SERVICES} \land \text{token_owner(srv)} \in \text{prs} \land \text{token_owner(srv)} \rightarrow \text{srv} \in \text{dom(suspc_peers)} \Rightarrow E(\text{srv}) \in \text{run_peers(srv)} \setminus (\text{unav_peers} \cup \text{prs} \cup \text{fail_peers}\{\text{srv}\}) \cup \text{suspc_peers(token_owner(srv) \rightarrow \text{srv})} \) 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

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

grd9: \( \forall \text{srv} \cdot \text{srv} \in \text{SERVICES} \Rightarrow (E(\text{srv}) \rightarrow \text{srv}) \rightarrow i\_\text{state} \)

(token_owner(srv) \rightarrow \text{srv}) \in i_s not theorem

grd10: \( p_s \in (\text{PEERS} \times \text{SERVICES}) \leftrightarrow \mathbb{P} (\text{PEERS}) \) not theorem

grd11: \( \forall p, s \cdot p \in \text{PEERS} \land s \in \text{SERVICES} \land p \not\in s \in \text{dom(p_s)} \Rightarrow p = E(s) \) not theorem

(token_owner(srv) \rightarrow \text{srv}) \in p_s not theorem \( \rightarrow \) THEN

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

act2: token_owner \( = \) token_owner \( \leftarrow E \) \( \rightarrow \) 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 \( \rightarrow \) the peers in prs can not try to recontact instances anymore

act5: rct_inst \( = \) prs \( \leftarrow \) rct_inst \( \rightarrow \) the peers in prs can not recontact instances anymore
M12

act6: \( \text{actv\_inst} := \text{prs} \leftarrow \text{actv\_inst} \)

act7: \( \text{i\_state} := \text{i\_s} \)

act8: \( \text{susp\_peers} := \text{p\_s} \)

END

SUSPECT\_INST: extended ordinary

REFINES

SUSPECT\_INST

ANY

s \( \triangleright \) a service s

susp \( \triangleright \) suspicious instances

WHERE

grd1: \( s \in \text{SERVICES} \)

grd2: \( \text{susp} \subseteq \text{PEERS} \)

grd3: \( \text{susp} = \text{run\_peers}(s) \cap \text{unav\_peers} \)

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

grd4: \( s \notin \text{dom}(\text{susp\_inst}[\{\text{token\_owner}(s)\}][\{s\}]) \)

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

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

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

THEN

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

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

END

FAIL: not extended ordinary

REFINES

FAIL

ANY

s \( \triangleright \)

WHERE

grd1: \( s \in \text{SERVICES} \)

grd2: \( \text{i\_state}(\text{token\_owner}(s) \mapsto s) = \text{RUN\_4} \)

grd3: \( \text{susp\_inst}[\{\text{token\_owner}(s)\}][\{s\}] \neq \emptyset \)

THEN

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

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

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

END

RECONTACT\_INST\_OK: not extended ordinary

REFINES

RECONTACT\_INST\_OK

ANY

s \( \triangleright \) a service s

i \( \triangleright \) an instance i

WHERE
M12

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) ≠ ∅ 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 i is available (can be contacted)

grd6: token_owner(s) ↦ (s ↦ i) ∉ rec_inst not theorem

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

grd7: rec_inst[{{token_owner(s)}][{s}]} ⊂ suspc_peers

(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: rec_inst = rec_inst u {token_owner(s) ↦ (s ↦ i)} not theorem

the token owner of s has tried to recontact i

act2: rct_inst = rct_inst u {token_owner(s) ↦ (s ↦ 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 → a service s

i → an instance i

WHERE

grd1: s ∈ SERVICES not theorem

grd2: i ∈ PEERS not theorem

grd3: i_state(token_owner(s) ↦ s) = FAIL_4 not theorem

state of s is SUSPICIOUS

grd4: suspc_peers(token_owner(s) ↦ s) ≠ ∅ 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 i is unavailable (can not be contacted)

grd6: token_owner(s) ↦ (s ↦ i) ∉ rec_inst not theorem

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

grd7: rec_inst[{{token_owner(s)}][{s}]} ⊂ suspc_peers

(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: rec_inst = rec_inst u {token_owner(s) ↦ (s ↦ 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} ◁ ran(rec_inst))
act4:  rct_inst ≔ rct_inst ⩥ ({s} ◁ ran(rct_inst))

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 → a service s
i → an instance i
WHERE
  grd1: s ∈ SERVICES not theorem →
grd2: \( i \in \text{PEERS} \) not theorem

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

grd4: \( \text{token\_owner}(s) \mapsto (s \mapsto i) \notin \text{actv\_inst} \) not theorem

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

\( \text{card}(\text{actv\_inst}\{\{\text{token\_owner}(s)\}\}[[s]]) < \text{deplo\_inst}(s) \) not theorem

\( \text{card}(\text{dep\_inst}[\{s\}]) + \text{card}(\text{run\_peers}(s)) < \text{min\_inst}(s) \) not theorem

THEN

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

END

REDEPLOY\_INSTS: extended ordinary

REFINES

REDEPLOY\_INSTS

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

\( \text{card}(\text{dep\_inst}[\{s\}]) \neq \emptyset \) not theorem

\( \text{card}(\text{run\_peers}(s))+\text{card}(\text{dep\_inst}[\{s\}]) \geq \text{min\_inst}(s) \) not theorem

THEN

act2: \( \text{actv\_inst} = \text{actv\_inst} \mapsto (\{s\} \leftarrow \text{ran}(\text{actv\_inst})) \)

END

REDEPLOY: extended ordinary

REFINES

REDEPLOY

ANY \( s \)

WHERE

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

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

\( \text{actv\_inst}\{\{\text{token\_owner}(s)\}\}[[s]] = \emptyset \) not theorem

\( \text{dep\_inst}[\{s\}] \neq \emptyset \) not theorem

\( \text{card}(\text{run\_peers}(s))+\text{card}(\text{dep\_inst}[\{s\}]) \geq \text{min\_inst}(s) \) not theorem

THEN
M12

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

END

HEAL: extended ordinary

REFINES

HEAL

ANY

s >

WHERE

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

THEN

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

END

UNFAIL_PEER: extended ordinary

REFINES

UNFAIL_PEER

ANY

s >
p >

WHERE

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

THEN

act1: fail\_peers = fail\_peers\{s \mapsto p\}

END

MAKE_PEER_AVAIL: extended ordinary

REFINES

MAKE_PEER_AVAIL

ANY

p >

WHERE

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

THEN

act1: unav\_peers = unav\_peers \setminus \{p\}

END

END
MACHINE M13
REFINES M12
SEES C08
VARIABLES
  run_peers
  suspc_peers
  fail_peers
  dep_inst
  token_owner
  unav_peers
  suspc_inst
  rec_inst
  rct_inst
  actv_inst
  instances that are tried to be recontacted
  instances effectively recontacted after a try
  instances activated by token owners
  i_state
INVARINTS
  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 = Ø
  act5: dep_inst = Ø
  act6: token_owner = init_tok
  act7: unav_peers = Ø
  act8: suspc_inst = InitSuspPeers
  act10: rec_inst = Ø
  act11: rct_inst = Ø
  act12: actv_inst = Ø
  act13: i_state = InitStatus
END
MAKE_PEER_UNAVAIL: not extended ordinary
REFINES MAKE_PEER_UNAVAIL
ANY
  prs
  E new values for token owner per service if needed
M13

\(i_s \ >\)
\(p_s \ >\)
\(s_i \ >\)

WHERE

\(\text{grd1}: \ prs \in \text{PEERS} \ \text{not theorem} \ >\)
\(\text{grd2}: \ prs \not\in \text{unav_peers} \ \text{not theorem} \ >\)
\(\text{grd3}: \ E \in \text{SERVICES} \rightarrow \text{PEERS} \ \text{not theorem} \rightarrow \text{new value for token}\)

owner per service if needed

\(\text{grd4}: \ i_s \in (\text{PEERS} \times \text{SERVICES}) \leftrightarrow \text{STATES}_4 \ \text{not theorem} \ >\)
\(\text{grd5}: \ \forall \ srv \cdot \ srv \in \text{SERVICES} \land \text{token_owner}(\text{srv}) \in \text{prs} \Rightarrow E(\text{srv}) = \text{token_owner}(\text{srv}) \ \text{not theorem} \rightarrow \text{If the token owner of a service srv does not belong to prs, the token owner is not changed}\)

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

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

\(\text{grd8}: \ \forall \ p, s \cdot \ p \in \text{PEERS} \land s \in \text{SERVICES} \land p \mapsto s \in \text{dom}(i_s) \Rightarrow \\
p = E(s) \ \text{not theorem} \ >\)
\(\text{grd9}: \ \forall \ \text{srv} \cdot \ \text{srv} \in \text{SERVICES} \Rightarrow (E(\text{srv}) \mapsto \text{srv}) \mapsto i_s \)

\(\text{grd10}: \ p_s \in (\text{PEERS} \times \text{SERVICES}) \leftrightarrow \text{P(PEERS)} \ \text{not theorem} \ >\)
\(\text{grd11}: \ \forall \ p, s \cdot \ p \in \text{PEERS} \land s \in \text{SERVICES} \land p \mapsto s \in \text{dom}(p_s) \Rightarrow \\
p = E(s) \ \text{not theorem} \ >\)
\(\text{grd12}: \ \forall \ \text{srv} \cdot \ \text{srv} \in \text{SERVICES} \Rightarrow (E(\text{srv}) \mapsto \text{srv}) \mapsto \text{suspc_peers} \)

\(\text{grd13}: \ s_i \in (\text{PEERS} \times \text{SERVICES}) \leftrightarrow \text{P(PEERS)} \ \text{not theorem} \ >\)
\(\text{grd14}: \ \forall \ p, s \cdot \ p \in \text{PEERS} \land s \in \text{SERVICES} \land p \mapsto s \in \text{dom}(s_i) \Rightarrow \\
p = E(s) \ \text{not theorem} \ >\)
\(\text{grd15}: \ \forall \ \text{srv} \cdot \ \text{srv} \in \text{SERVICES} \land \text{token_owner}(\text{srv}) \in \text{prs} \Rightarrow (E(\text{srv}) \mapsto \text{srv}) \mapsto \text{suspc_inst}(E(\text{srv}) \mapsto \text{srv}) \in s_i \not\text{not theorem} \ >\)
\(\text{grd16}: \ \forall \ \text{srv} \cdot \ \text{srv} \in \text{SERVICES} \land \text{token_owner}(\text{srv}) \in \text{prs} \Rightarrow (E(\text{srv}) \mapsto \text{srv}) \mapsto \emptyset \in s_i \not\text{not theorem} \ >\)

THEN

\(\text{act1}: \ \text{unav_peers} := \text{unav_peers} \cup \text{prs} \rightarrow \text{the peers in prs become unavailable}\)
act2: token_owner := token_owner ⊕ E \rightarrow \text{new value for token owner per service is given if needed}

act3: rec_inst := prs ⩤ rec_inst \rightarrow \text{the peers in prs can not try to recontact instances anymore}

act4: rct_inst := prs ⩤ rct_inst \rightarrow \text{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 \rightarrow

REFINES
SUSPECT_INST

ANY

s \rightarrow \text{a service s}

susp \rightarrow \text{suspicious instances}

WHERE

grd1: s ∈ SERVICES not theorem \rightarrow

grd2: susp ⊆ PEERS not theorem \rightarrow

grd3: susp = run_peers(s) ∩ unav_peers not theorem \rightarrow \text{instances in susp are suspicious if the peers running them becomes unavailable}

grd4: suspc_inst(token_owner(s) ↦ s) = ∅ not theorem \rightarrow \text{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 \rightarrow \text{the state of s is OK}

grd6: susp ≠ ∅ not theorem \rightarrow

THEN

act1: suspc_inst(token_owner(s) ↦ s) = susp \rightarrow \text{the members of susp become suspected instances for s by the token owner of s}

END

FAIL: not extended ordinary \rightarrow

REFINES
FAIL

ANY

s \rightarrow

WHERE

grd1: s ∈ SERVICES not theorem \rightarrow

grd2: i_state(token_owner(s) ↦ s) = RUN_4 not theorem \rightarrow

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

THEN

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

act2: suspc_peers(token_owner(s) ↦ s) ≠ suspc_inst(token_owner(s) ↦ s) \rightarrow

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

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:  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:  token_owner(s) ↦ (s ↦ i) ∉ rec_inst not theorem

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

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

the token owner of s has tried to recontact i
act2:  rct_inst = rct_inst u \{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:  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 unavailable (can not be contacted)
grd6:  token_owner(s) ↦ (s ↦ i) ∉ rec_inst not theorem

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

(token_owner(s) ↦ s) not theorem

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

THEN

\[
\text{act1: } \text{rec\_inst} = \text{rec\_inst} \cup \{\text{token\_owner}(s) \mapsto (s \mapsto i)\} \quad \text{the token owner of } s \text{ has tried to recontact } i
\]

END

**FAIL\_DETECT:** extended ordinary \>

*REFINES*

FAIL\_DETECT

Any

\(s\) >

Where

- \(\text{grd1: } s \in \text{SERVICES} \not \text{ theorem} >\)
- \(\text{grd2: } \text{i\_state}(\text{token\_owner}(s) \mapsto s) = \text{FAIL\_4} \not \text{ theorem} >\)
- \(\text{grd5: } \text{susp\_peers}(\text{token\_owner}(s) \mapsto s) \neq \emptyset \not \text{ theorem} >\)
- \(\text{grd8: } \text{rec\_inst}[\{\text{token\_owner}(s)\}][\{s\}] = \text{susp\_peers} >\)

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

THEN

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

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

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

\[
\text{act3: } \text{rec\_inst} = \text{rec\_inst} \cup \{\{s\} \leftarrow \text{ran}(\text{rec\_inst})\} \quad >
\]

\[
\text{act4: } \text{rct\_inst} = \text{rct\_inst} \cup \{\{s\} \leftarrow \text{ran}(\text{rct\_inst})\} \quad >
\]

END

**IS\_OK:** extended ordinary \>

*REFINES*

IS\_OK

Any

\(s\) >

Where

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

THEN

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

END

**FAIL\_ACTIV:** extended ordinary \>

*REFINES*

FAIL\_ACTIV

Any

\(s\) >

Where

- \(\text{grd1: } s \in \text{SERVICES} \not \text{ theorem} >\)
- \(\text{grd2: } \text{i\_state}(\text{token\_owner}(s) \mapsto s) = \text{FAIL\_DETECT\_4} \not \text{ theorem} >\)
M13

\[ \text{grd5: } \text{susp}_\text{c}_\text{p}_\text{e}_\text{ers}(\text{token}_\text{owner}(s) \mapsto s) \neq \emptyset \text{ not theorem } \]

THEN

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

\text{FAIL\_CONFIGURE: extended ordinary }
\text{REFINES FAIL\_CONFIGURE}
\text{ANY s }
\text{WHERE}
\[ \text{grd1: } s \in \text{SERVICES not theorem } \]
\[ \text{grd2: } i\_\text{state}(\text{token}_\text{owner}(s) \mapsto s) = \text{FAIL\_ACTIV\_4 not theorem } \]
\[ \text{grd3: } \text{card}(\text{run}\_\text{peers}(s)) < \text{min}\_\text{inst}(s) \text{ not theorem } \]
THEN
\[ \text{act1: } i\_\text{state}(\text{token}_\text{owner}(s) \mapsto s) = \text{FAIL\_CONFIG\_4 } \]
END

\text{FAIL\_IGNORE: extended ordinary }
\text{REFINES FAIL\_IGNORE}
\text{ANY s }
\text{WHERE}
\[ \text{grd1: } s \in \text{SERVICES not theorem } \]
\[ \text{grd2: } i\_\text{state}(\text{token}_\text{owner}(s) \mapsto s) = \text{FAIL\_ACTIV\_4 not theorem } \]
\[ \text{grd3: } \text{card}(\text{run}\_\text{peers}(s)) \geq \text{min}\_\text{inst}(s) \text{ not theorem } \]
THEN
\[ \text{act1: } i\_\text{state}(\text{token}_\text{owner}(s) \mapsto s) = \text{FAIL\_IGN\_4 } \]
END

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

ANY
s \rightarrow a service s
i \rightarrow an instance i

WHERE
grd1: \quad s \in SERVICES \quad \text{not theorem}
\quad \text{not theorem}

grd2: \quad i \in \text{PEERS} \quad \text{not theorem}
\quad \text{not theorem}

grd3: \quad i \notin \text{run_peers}(s) \cup \text{fail_peers}[][s] \cup \text{unav_peers} \cup \text{dep_inst}[][s] \quad \text{not theorem}
\quad \text{i does not run s, is not failed for s, is not unavailable and is not already activated for s}

grd4: \quad \text{token_owner}(s) \mapsto (s \mapsto i) \notin \text{actv_inst} \quad \text{not theorem}

grd5: \quad \text{i_state}(\text{token_owner}(s) \mapsto s) = \text{FAIL_CONFIG_4} \quad \text{not theorem}

grd6: \quad \text{card}(\text{actv_inst}[][\text{token_owner}(s)][s]) < \text{deplo_inst}(s)
\quad \text{not theorem}

grd7: \quad \text{card}(\text{dep_inst}[][s]) + \text{card}(\text{run_peers}(s)) < \text{min_inst}(s)
\quad \text{not theorem}

\text{THEN}
act1: \quad \text{actv_inst} = \text{actv_inst} \cup \{\text{token_owner}(s) \mapsto (s \mapsto i)\}

END

REDEPLOY_INSTS: extended ordinary
REFINES
REDEPLOY_INSTS

ANY
s \rightarrow

WHERE
grd1: \quad s \in SERVICES \quad \text{not theorem}
\quad \text{not theorem}

grd6: \quad \text{card}(\text{actv_inst}[][\text{token_owner}(s)][s]) = \text{deplo_inst}(s)
\quad \text{not theorem}

grd7: \quad \text{card}(\text{dep_inst}[][s]) + \text{card}(\text{run_peers}(s)) < \text{min_inst}(s)
\quad \text{not theorem}

grd8: \quad \text{i_state}(\text{token_owner}(s) \mapsto s) = \text{FAIL_CONFIG_4} \quad \text{not theorem}

\text{THEN}
act1: \quad \text{dep_inst} = \text{dep_inst} \cup \{s \times \text{actv_inst}[][\text{token_owner}(s)][s]\}\n
act2: \quad \text{actv_inst} = \text{actv_inst} \mapsto \{s \mapsto \text{ran}(\text{actv_inst})\}

END

REDEPLOY: extended ordinary
REFINES
REDEPLOY

ANY
s \rightarrow

WHERE
M13

\[ s \in \text{SERVICES} \not\text{ theorem} \]
\[ \text{i\_state(token\_owner(s) \mapsto s) = FAIL\_CONFIG\_4} \not\text{ theorem} \]
\[ \text{act\_inst[\{token\_owner(s)\}][s]} = \emptyset \not\text{ theorem} \]
\[ \text{dep\_inst[s]} \neq \emptyset \not\text{ theorem} \]
\[ \text{card(run\_peers(s)) + card(dep\_inst[s])} \geq \text{min\_inst(s)} \not\text{ theorem} \]

THEN

act1: \text{i\_state(token\_owner(s) \mapsto s) = DPL\_4} \not\text{ theorem} \]
act2: \text{run\_peers(s) = run\_peers(s) \cup dep\_inst[s]} \not\text{ theorem} 
act3: \text{dep\_inst = \{s\} \triangleleft dep\_inst} \not\text{ theorem} \]

END

HEAL: extended ordinary

\[ \text{REFINES HEAL} \]
\[ \text{ANY s} \]
WHERE

\[ s \in \text{SERVICES} \not\text{ theorem} \]
\[ \text{i\_state(token\_owner(s) \mapsto s) = DPL\_4} \not\text{ theorem} \]

THEN

act1: \text{i\_state(token\_owner(s) \mapsto s) = RUN\_4} \not\text{ theorem} \]

END

UNFAIL_PEER: extended ordinary

\[ \text{REFINES UNFAIL\_PEER} \]
\[ \text{ANY s p} \]
WHERE

\[ s \in \text{SERVICES} \not\text{ theorem} \]
\[ p \in \text{PEERS} \not\text{ theorem} \]
\[ s \mapsto p \in \text{fail\_peers} \not\text{ theorem} \]

THEN

act1: \text{fail\_peers = fail\_peers\{s \mapsto p\}} \not\text{ theorem} \]

END

MAKE_PEER_AVAIL: extended ordinary

\[ \text{REFINES MAKE\_PEER\_AVAIL} \]
\[ \text{ANY p} \]
WHERE

\[ p \in \text{PEERS} \not\text{ theorem} \]
\[ p \in \text{unav\_peers} \not\text{ theorem} \]

THEN
M.13

\[ \text{act1}: \quad \text{unav}_\text{peers} = \text{unav}_\text{peers} \setminus \{ p \} \]

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
inv2:  ∀ p, s · p ∈ PEERS ∧ s ∈ SERVICES ∧ (p ↦ s) ∈ dom(rect_inst) ⇒ p = token_owner(s) not theorem
inv3:  ∀ p, s · p ∈ PEERS ∧ s ∈ SERVICES ∧ p = token_owner(s) ⇒ (p ↦ s) ∈ dom(rect_inst) not theorem
  gluing_tok_own rec1:  ∀ p, s · p ∈ PEERS ∧ s ∈ SERVICES ∧ (p ↦ s) ∈ dom(rect_inst) ⇒ rec_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) not theorem
inv6:  ∀ p, s · p ∈ PEERS ∧ s ∈ SERVICES ∧ 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
  gluing_fail_1:  ∀ s · s ∈ SERVICES ⇒ failr_peers[{s}] = failr_peers(s) not theorem
  gluing_act_1:  ∀ s · s ∈ SERVICES ⇒ dep_instc[{s}] = 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
 Possess suspicious instances, then a new token owner among available and not
suspicous 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(srv) ↦ srv)) not theorem

(srv) → srv ∈ rt_s ∧ ((E(srv) → srv) → rect_inst(E(srv) → srv) ∈ rc_s) not theorem

grd14: ∀ srv · srv ∈ SERVICES ∧ token_owner(srv) ∈ prs ⇒ ((E(srv) → srv) → ø ∈ s_i) ∧ ((E(srv) → srv) → ø ∈ rt_s) ∧ ((E(srv) → srv) → ø ∈ rc_s) not theorem

THEN

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

token_owner = token_owner ≔ E → new value for token owner

per service is given if needed

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

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

act5: actv_inst ≔ prs ⩤ actv_inst → the peers in prs become suspected instances for s by the token owner of s

act6: i_state = i_s → the state of s is OK

act7: suspc_peers = p_s → the members of susp become suspected instances for s by the token owner of s

act8: suspc_inst = s_i → extended ordinary

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) ∩ unav_peers not theorem → instances in susp are suspicious if the peers running them becomes unavailable

grd4: susp = token_owner(s) → susp = ø 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

THEN

act1: susp = susp ≔ susp = s_i → extended ordinary

FAIL: extended ordinary

REFINES

FAIL

ANY

s →

WHERE

grd1: s ∈ SERVICES not theorem
M14

\[ \text{grd2: } \text{i\_state(token\_owner(s) \mapsto s) = RUN\_4} \not\text{ theorem} \] \\
\[ \text{grd3: } \text{susp\_inst(token\_owner(s) \mapsto s) \neq \emptyset} \not\text{ theorem} \] \\
\text{THEN} \\
\[ \text{act1: } \text{i\_state(token\_owner(s) \mapsto s) = FAIL\_4} \] \\
\[ \text{act2: } \text{susp\_peers(token\_owner(s) \mapsto s) = susp\_inst(token\_owner(s) \mapsto s)} \] \\
\[ \text{act3: } \text{susp\_inst(token\_owner(s) \mapsto s) = \emptyset} \] \\
\text{END} \\

\text{RECONTACT\_INST\_OK: not extended ordinary} \\
\text{REFINES} \\
\text{RECONTACT\_INST\_OK} \\
\text{ANY} \\
\quad s \mapsto \text{a service s} \\
\quad i \mapsto \text{an instance i} \\
\text{WHERE} \\
\quad \text{grd1: } s \in \text{SERVICES} \not\text{ theorem} \\
\quad \text{grd2: } i \in \text{PEERS} \not\text{ theorem} \\
\quad \text{grd3: } \text{i\_state(token\_owner(s) \mapsto s) = FAIL\_4} \not\text{ theorem} \] \\
\text{the state of s is SUSPICIOUS} \\
\quad \text{grd4: } \text{susp\_peers(token\_owner(s) \mapsto s) \neq \emptyset} \not\text{ theorem} \] \\
\text{the set of suspicious peers for s is not empty} \\
\quad \text{grd5: } i \in \text{susp\_peers(token\_owner(s) \mapsto s) \setminus \text{unav\_peers}} \not\text{ theorem} \] \\
\text{i is a suspicious instance of s and is available (can be contacted)} \\
\quad \text{grd6: } i \not\in \text{rect\_inst(token\_owner(s) \mapsto s)} \not\text{ theorem} \] \\
\text{the token owner of s has not yet tried to recontact i} \\
\quad \text{grd7: } \text{rect\_inst(token\_owner(s) \mapsto s) \subset \text{susp\_peers(token\_owner(s) \mapsto s)}} \not\text{ theorem} \] \\
\text{the token owner of s has not yet tried to recontact all the suspicious instances of s} \\
\text{THEN} \\
\quad \text{act1: } \text{rect\_inst(token\_owner(s) \mapsto s) = rect\_inst(token\_owner(s) \mapsto s) \cup \{i\}} \not\text{ theorem} \] \\
\text{the token owner of s has tried to recontact i} \\
\quad \text{act2: } \text{rctt\_inst(token\_owner(s) \mapsto s) = rctt\_inst(token\_owner(s) \mapsto s) \cup \{i\}} \not\text{ theorem} \] \\
\text{i is recontacted by the token owner of s successfully} \\
\text{END} \\

\text{RECONTACT\_INST\_KO: not extended ordinary} \\
\text{REFINES} \\
\text{RECONTACT\_INST\_KO} \\
\text{ANY} \\
\quad s \mapsto \text{a service s} \\
\quad i \mapsto \text{an instance i} \\
\text{WHERE} \\
\quad \text{grd1: } s \in \text{SERVICES} \not\text{ theorem} \\
\quad \text{grd2: } i \in \text{PEERS} \not\text{ theorem} \\
\quad \text{grd3: } \text{i\_state(token\_owner(s) \mapsto s) = FAIL\_4} \not\text{ theorem} \] \\
\text{the state of s is SUSPICIOUS} \\
\quad \text{grd4: } \text{susp\_peers(token\_owner(s) \mapsto s) \neq \emptyset} \not\text{ theorem} \] \\
\text{the set of suspicious peers for s is not empty} \\
\quad \text{grd5: } i \in \text{susp\_peers(token\_owner(s) \mapsto s) \setminus \text{unav\_peers}} \not\text{ theorem} \] \\
\text{i is a suspicious instance of s and is available (can be contacted)} \\
\quad \text{grd6: } i \not\in \text{rect\_inst(token\_owner(s) \mapsto s)} \not\text{ theorem} \] \\
\text{the token owner of s has not yet tried to recontact i} \\
\quad \text{grd7: } \text{rect\_inst(token\_owner(s) \mapsto s) \subset \text{susp\_peers(token\_owner(s) \mapsto s)}} \not\text{ theorem} \] \\
\text{the token owner of s has not yet tried to recontact all the suspicious instances of s} \\
\text{THEN} \\
\quad \text{act1: } \text{rect\_inst(token\_owner(s) \mapsto s) = rect\_inst(token\_owner(s) \mapsto s) \cup \{i\}} \not\text{ theorem} \] \\
\text{the token owner of s has tried to recontact i} \\
\quad \text{act2: } \text{rctt\_inst(token\_owner(s) \mapsto s) = rctt\_inst(token\_owner(s) \mapsto s) \cup \{i\}} \not\text{ theorem} \] \\
\text{i is recontacted by the token owner of s successfully} \\
\text{END}
of suspicious peers for $s$ is not empty

\[ \text{grd5: } i \in \text{suspc_peers}(\text{token_owner}(s) \rightarrow s) \cap \text{unav_peers} \]

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

\[ \text{grd6: } i \notin \text{rect_inst}(\text{token_owner}(s) \rightarrow s) \]

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

\[ \text{grd7: } \text{rect_inst}(\text{token_owner}(s) \rightarrow s) \subset \text{suspc_peers}(\text{token_owner}(s) \rightarrow s) \]

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

THEN

act1: \( \text{rect_inst}(\text{token_owner}(s) \rightarrow s) = \text{rect_inst}(\text{token_owner}(s) \rightarrow s) \cup \{i\} \)

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

END

FAIL_DETECT: \( \text{not extended ordinary} \Rightarrow \)

REFINES

FAIL_DETECT

ANY

s

WHERE

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

\[ \text{grd2: } \text{i_state}(\text{token_owner}(s) \rightarrow s) = \text{FAIL_4} \]

\[ \text{grd5: } \text{suspc_peers}(\text{token_owner}(s) \rightarrow s) \neq \emptyset \]

\[ \text{grd8: } \text{rect_inst}(\text{token_owner}(s) \rightarrow s) = \text{suspc_peers}(\text{token_owner}(s) \rightarrow s) \]

not theorem \( \Rightarrow \)

THEN

act1: \( \text{i_state}(\text{token_owner}(s) \rightarrow s) = \text{FAIL_DETECT_4} \)

act2: \( \text{suspc_peers}(\text{token_owner}(s) \rightarrow s) = \text{suspc_peers}(\text{token_owner}(s) \rightarrow s) \) \( \setminus \) \( \text{rectt_inst}(\text{token_owner}(s) \rightarrow s) \)

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

act4: \( \text{rectt_inst}(\text{token_owner}(s) \rightarrow s) = \emptyset \)

END

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

REFINES

IS_OK

ANY

s

WHERE

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

\[ \text{grd2: } \text{i_state}(\text{token_owner}(s) \rightarrow s) = \text{FAIL_DETECT_4} \]

\[ \text{grd5: } \text{suspc_peers}(\text{token_owner}(s) \rightarrow s) = \emptyset \]

THEN

act1: \( \text{i_state}(\text{token_owner}(s) \rightarrow s) = \text{RUN_4} \)

END

FAIL_ACTIV: \( \text{not extended ordinary} \Rightarrow \)
REFINES
FAIL_ACTIV
ANY
s
WHERE
  grd1: s ∈ SERVICES not theorem
  grd2: i_state(token_owner(s) ↦ s) = FAIL_DETECT_4 not theorem
  grd5: suspc_peers(token_owner(s) ↦ s) ≠ ∅ not theorem
THEN
  act1: i_state(token_owner(s) ↦ s) ≔ FAIL_ACTIV_4
  act2: run_peers(s) ≔ run_peers(s) \ suspc_peers(token_owner(s) ↦ s)
  act3: failr_peers(s) ≔ failr_peers(s) \ suspc_peers(token_owner(s) ↦ s)
  act4: suspc_peers(token_owner(s) ↦ s) ≔ ∅
END

FAIL_CONFIGURE: extended ordinary
REFINES
FAIL_CONFIGURE
ANY
s
WHERE
  grd1: s ∈ SERVICES not theorem
  grd2: i_state(token_owner(s) ↦ s) = FAIL_Activ_4 not theorem
  grd3: card(run_peers(s)) < min_inst(s) not theorem
THEN
  act1: i_state(token_owner(s) ↦ s) ≔ FAIL_CONFIG_4
END

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

IGNORE: extended ordinary
REFINES
IGNORE
ANY
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: not extended ordinary →
REFINES
  REDEPLOY_INSTC
ANY
  s → a service s
  i → an instance i
WHERE
  grd1: s ∈ SERVICES not theorem →
  grd2: i ∈ PEERS not theorem →
  grd3: i ∉ run_peers(s) ∪ failr_peers(s) ∪ unav_peers ∪ dep_instc(s) not theorem → i does not run s, is not failed for s, is not unavailable and is not already activated for s
  grd4: token_owner(s) ↦ (s ↦ i) ∉ actv_inst not theorem →
  grd5: i_state(token_owner(s) ↦ s) = FAIL_CONFIG_4 not theorem →
  grd6: card(actv_inst[{token_owner(s)}][{s}]) < deplo_ins(s) → card(actv_inst[{token_owner(s)}][{s}]) < deplo_ins(s)
  grd7: card(dep_instc(s)) + card(run_peers(s)) < min_inst(s) → card(dep_instc(s)) + card(run_peers(s)) < min_inst(s)
THEN
  act1: actv_inst = actv_inst ∪ {token_owner(s) ↦ (s ↦ i)} →
END

REDEPLOY_INSTS: not extended ordinary →
REFINES
  REDEPLOY_INSTS
ANY
  s →
WHERE
  grd1: s ∈ SERVICES not theorem →
  grd6: card(actv_inst[{token_owner(s)}][{s}]) = deplo_ins(s) →
  grd7: card(dep_instc(s)) + card(run_peers(s)) < min_inst(s) →
  grd8: i_state(token_owner(s) ↦ s) = FAIL_CONFIG_4 not theorem →
  grd9: actv_inst = actv_inst[{token_owner(s)}][{s}] →
THEN
  act1: dep_instc(s) = dep_instc(s) ∪ actv_inst[{token_owner(s)}][{s}] →
  act2: actv_inst = actv_inst ∪ ({s} ↼ ran(actv_inst)) →

REDEPLOY: not extended ordinary

REDEPLOY

\begin{align*}
\text{REFINES} & \quad \text{REDEPLOY} \\
\text{ANY} & \quad s \\
\text{WHERE} & \\
\text{grd1:} & \quad s \in \text{SERVICES} \not \text{ theorem} \\
\text{grd2:} & \quad \text{i_state(token_owner(s) ↦ s)} = \text{FAIL\_CONFIG\_4} \not \text{ theorem} \\
\text{grd7:} & \quad \text{actv\_inst[{\{token\_owner(s)\}}][{s}]=\emptyset} \not \text{ theorem} \\
\text{grd6:} & \quad \text{dep\_instc(s)} \neq \emptyset \not \text{ theorem} \\
\text{grd4:} & \quad \text{card(run_peers(s)) + card(dep\_instc(s))} \geq \text{min\_inst(s)} \not \text{ theorem} \\
\end{align*}

\begin{align*}
\text{THEN} & \\
\text{act1:} & \quad \text{i_state(token_owner(s) ↦ s)} = \text{DPL\_4} \\
\text{act2:} & \quad \text{run_peers(s)} = \text{run_peers(s)} \cup \text{dep\_instc(s)} \\
\text{act3:} & \quad \text{dep\_instc(s)} = \emptyset \\
\end{align*}

END

HEAL: extended ordinary

HEAL

\begin{align*}
\text{REFINES} & \quad \text{HEAL} \\
\text{ANY} & \quad s \\
\text{WHERE} & \\
\text{grd1:} & \quad s \in \text{SERVICES} \not \text{ theorem} \\
\text{grd2:} & \quad \text{i_state(token_owner(s) ↦ s)} = \text{DPL\_4} \not \text{ theorem} \\
\text{THEN} & \\
\text{act1:} & \quad \text{i_state(token_owner(s) ↦ s)} = \text{RUN\_4} \\
\end{align*}

END

UNFAIL\_PEER: not extended ordinary

UNFAIL\_PEER

\begin{align*}
\text{REFINES} & \quad \text{UNFAIL\_PEER} \\
\text{ANY} & \quad s \\
\text{p} & \quad \text{WHERE} \\
\text{grd1:} & \quad s \in \text{SERVICES} \not \text{ theorem} \\
\text{grd2:} & \quad p \in \text{PEERS} \not \text{ theorem} \\
\text{grd3:} & \quad p \in \text{failr\_peers(s)} \not \text{ theorem} \\
\text{THEN} & \\
\text{act1:} & \quad \text{failr\_peers(s)} = \text{failr\_peers(s)\\{p\}} \\
\end{align*}

END

MAKE\_PEER\_AVAIL: extended ordinary

Page 8
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
M15

MACHINE
M15
REFINES
M14
SEES
C08
VARIABLES
run_peers
suspc_peers
failr_peers
dep_instc
token_owner
unav_peers
suspc_inst
rect_inst
rctt_inst
actv_instc
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

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
WHERE

grd1: \( \text{prs} \subseteq \text{PEERS} \) not theorem >
grd2: \( \text{prs} \not\subseteq \text{unav_peers} \) not theorem >
grd3: \( E \in \text{SERVICES} \rightarrow \text{PEERS} \) not theorem > new value for token owner per service if needed

g grd4: \( i_s \in (\text{PEERS} \times \text{SERVICES}) \leftrightarrow \text{STATES}_4 \) not theorem >
grd5: \( p_s \in (\text{PEERS} \times \text{SERVICES}) \leftrightarrow \mathcal{P}(\text{PEERS}) \) not theorem >
grd6: \( s_i \in (\text{PEERS} \times \text{SERVICES}) \leftrightarrow \mathcal{P}(\text{PEERS}) \) not theorem >
grd7: \( r_t_s \in (\text{PEERS} \times \text{SERVICES}) \leftrightarrow \mathcal{P}(\text{PEERS}) \) not theorem >
grd8: \( r_c_s \in (\text{PEERS} \times \text{SERVICES}) \leftrightarrow \mathcal{P}(\text{PEERS}) \) not theorem >
grd9: \( a_c_i \in (\text{PEERS} \times \text{SERVICES}) \leftrightarrow \mathcal{P}(\text{PEERS}) \) not theorem >
grd10: \( \text{dom}(i_s) = E \wedge \text{dom}(p_s) = \text{dom}(i_s) \wedge \text{dom}(s_i) = \text{dom}(r_t_s) = \text{dom}(r_c_s) = \text{dom}(ac_i) = \text{dom}(i_s) \) not theorem >

grd11: \( \forall \text{srv} \cdot \text{srv} \in \text{SERVICES} \wedge \text{token_owner}(\text{srv}) \notin \text{prs} \Rightarrow\)
\( E(\text{srv}) = \text{token_owner}(\text{srv}) \wedge \)
\( s_i(E(\text{srv}) \rightarrow \text{srv}) = \text{suspc_inst}(E(\text{srv}) \rightarrow \text{srv}) \wedge \)
\( r_t_s(E(\text{srv}) \rightarrow \text{srv}) = \text{rctt_inst}(E(\text{srv}) \rightarrow \text{srv}) \wedge \)
\( r_c_s(E(\text{srv}) \rightarrow \text{srv}) = \text{rect_inst}(E(\text{srv}) \rightarrow \text{srv}) \wedge \)
\( a_c_i(E(\text{srv}) \rightarrow \text{srv}) = \text{actv_instc}(E(\text{srv}) \rightarrow \text{srv}) \) not theorem >

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

grd12: \( \forall \text{srv} \cdot \text{srv} \in \text{SERVICES} \wedge \text{token_owner}(\text{srv}) \in \text{prs} \Rightarrow\)
\( E(\text{srv}) \in \text{run_peers}(\text{srv}) \setminus (\text{unav_peers} \cup \text{prs} \cup \text{failr_peers}(\text{srv}) \cup \text{suspc_peers}(\text{token_owner}(\text{srv}) \rightarrow \text{srv})) \wedge \)
\( s_i(E(\text{srv}) \rightarrow \text{srv}) = \emptyset \wedge \)
\( r_t_s(E(\text{srv}) \rightarrow \text{srv}) = \emptyset \wedge \)
\( r_c_s(E(\text{srv}) \rightarrow \text{srv}) = \emptyset \wedge \)
\( a_c_i(E(\text{srv}) \rightarrow \text{srv}) = \emptyset \) not theorem > if the owner of the token for a service becomes unavailable, and the service possess suspicious instances, then a new token owner among available and not suspicious peers is chosen:

grd13: \( \forall \text{srv} \cdot \text{srv} \in \text{SERVICES} \Rightarrow i_s(E(\text{srv}) \rightarrow \text{srv}) = \text{i_state}(\text{token_owner}(\text{srv}) \rightarrow \text{srv}) \wedge p_s(E(\text{srv}) \rightarrow \text{srv}) = \text{suspc_peers}(\text{token_owner}(\text{srv}) \rightarrow \text{srv}) \) not theorem >

THEN

act1: \( \text{unav_peers} = \text{unav_peers} \cup \text{prs} \) the peers in prs become unavailable
M15

\[\text{act2: } \text{token\_owner} \leftarrow \text{token\_owner} \uparrow \text{E} \rightarrow \text{new value for token owner}\]

per service is given if needed

\[\text{act3: } \text{rect\_inst} \leftarrow \text{rc\_s} \rightarrow \text{the peers in prs can not try to}\]

recontact instances anymore

\[\text{act4: } \text{rctt\_inst} \leftarrow \text{rt\_s} \rightarrow \text{the peers in prs can not recontact}\]

instances anymore

\[\text{act5: } \text{actv\_instc} \leftarrow \text{ac\_i} \rightarrow \]

\[\text{act6: } \text{i\_state} \leftarrow \text{i\_s} \rightarrow \]

\[\text{act7: } \text{susp\_peers} \leftarrow \text{p\_s} \rightarrow \]

\[\text{act8: } \text{susp\_inst} \leftarrow \text{s\_i} \rightarrow \]

END

\textit{SUSPECT\_INST: } extended ordinary

\textit{REFINES}

\textit{SUSPECT\_INST}

\textit{ANY}

\textit{s} \rightarrow \text{a service s}

\textit{susp} \rightarrow \text{suspicious instances}

\textit{WHERE}

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

\text{grd2: } \text{susp} \subseteq \text{PEERS} \not\text{ theorem} \rightarrow

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

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

\text{grd4: } \text{susp\_inst(token\_owner(s) \uparrow s)} = \emptyset \not\text{ theorem} \rightarrow

\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) \uparrow s)} = \text{RUN\_4} \not\text{ theorem} \rightarrow

\text{the state of s is OK}

\text{grd6: } \text{susp} \neq \emptyset \not\text{ theorem} \rightarrow

\text{THEN}

\text{act1: } \text{susp\_inst(token\_owner(s) \uparrow s)} = \text{susp} \rightarrow

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

END

\textit{FAIL: } extended ordinary

\textit{REFINES}

\textit{FAIL}

\textit{ANY}

\textit{s} \rightarrow

\textit{WHERE}

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

\text{grd2: } \text{i\_state(token\_owner(s) \uparrow s)} = \text{RUN\_4} \not\text{ theorem} \rightarrow

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

\text{THEN}

\text{act1: } \text{i\_state(token\_owner(s) \uparrow s)} = \text{FAIL\_4} \rightarrow

\text{act2: } \text{susp\_peers(token\_owner(s) \uparrow s)} = \text{susp\_inst(token\_owner(s) \uparrow s)} \rightarrow

\text{act3: } \text{susp\_inst(token\_owner(s) \uparrow s)} = \emptyset \rightarrow

\text{END}
RECONTACT_INST_OK: extended ordinary

REFINES
RECONTACT_INST_OK

ANY
s \rightarrow a service s
i \rightarrow an instance i

WHERE

grd1: s \in SERVICES not theorem
grd2: i \in PEERS not theorem
grd3: \text{io}_{state}(token\textunderscore owner(s) \mapsto s) = FAIL_4 not theorem \rightarrow the state of s is SUSPICIOUS
grd4: \text{susp}_\text{pc}_\text{peers}(token\textunderscore owner(s) \mapsto s) \neq \varnothing not theorem \rightarrow the set of suspicious peers for s is not empty
grd5: i \in \text{susp}_\text{pc}_\text{peers}(token\textunderscore owner(s) \mapsto s) \setminus \text{unav}_\text{peers} not theorem \rightarrow i is a suspicious instance of s and is available (can be contacted)
grd6: i \notin \text{rect}_\text{inst}(token\textunderscore owner(s) \mapsto s) not theorem \rightarrow the token owner of s has not yet tried to recontact i
grd7: \text{rect}_\text{inst}(token\textunderscore owner(s) \mapsto s) \subset \text{susp}_\text{pc}_\text{peers}(token\textunderscore 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}_\text{inst}(token\textunderscore owner(s) \mapsto s) = \text{rect}_\text{inst}(token\textunderscore owner(s) \mapsto s) \cup \{i\} \rightarrow the token owner of s has tried to recontact i
act2: \text{rctt}_\text{inst}(token\textunderscore owner(s) \mapsto s) = \text{rctt}_\text{inst}(token\textunderscore owner(s) \mapsto s) \cup \{i\} \rightarrow i is recontacted by the token owner of s successfully

END

RECONTACT_INST_KO: extended ordinary

REFINES
RECONTACT_INST_KO

ANY
s \rightarrow a service s
i \rightarrow an instance i

WHERE

grd1: s \in SERVICES not theorem
grd2: i \in PEERS not theorem
grd3: \text{io}_{state}(token\textunderscore owner(s) \mapsto s) = FAIL_4 not theorem \rightarrow the state of s is SUSPICIOUS
grd4: \text{susp}_\text{pc}_\text{peers}(token\textunderscore owner(s) \mapsto s) \neq \varnothing not theorem \rightarrow the set of suspicious peers for s is not empty
grd5: i \in \text{susp}_\text{pc}_\text{peers}(token\textunderscore owner(s) \mapsto s) \setminus \text{unav}_\text{peers} not theorem \rightarrow i is a suspicious instance of s and is unavailable (can not be contacted)
grd6: i \notin \text{rect}_\text{inst}(token\textunderscore owner(s) \mapsto s) not theorem \rightarrow the token owner of s has not yet tried to recontact i
grd7: \text{rect}_\text{inst}(token\textunderscore owner(s) \mapsto s) \subset \text{susp}_\text{pc}_\text{peers}(token\textunderscore 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: 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: 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) ≠ ø not theorem >
      grd8: rect_inst(token_owner(s) ↦ s) = suspc_peers(token_owner(s) ↦ s) not theorem >
    THEN
      act1: i_state(token_owner(s) ↦ s) = FAIL_DETECT_4 >
      act2: suspc_peers(token_owner(s) ↦ s) = suspc_peers(token_owner(s) ↦ s) \ rctt_inst(token_owner(s) ↦ s) >
      act3: rect_inst(token_owner(s) ↦ s) = ø >
      act4: rctt_inst(token_owner(s) ↦ s) = ø >
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) = ø 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 >
    THEN

M15

\[
\text{grd5: } \text{suspce_peers}(\text{token_owner}(s) \mapsto s) \neq \emptyset \text{ not theorem }
\]

THEN

\[
\text{act1: } \text{i_state}(\text{token_owner}(s) \mapsto s) = \text{FAIL_ACTIV}_4 \\
\text{act2: } \text{run_peers}(s) = \text{run_peers}(s) \setminus \text{suspce_peers}(\text{token_owner}(s) \mapsto s) \\
\text{act3: } \text{failr_peers}(s) = \text{failr_peers}(s) \cup \text{suspce_peers}(\text{token_owner}(s) \mapsto s) \\
\text{act4: } \text{suspce_peers}(\text{token_owner}(s) \mapsto s) = \emptyset
\]

END

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

ANY \( s \)

WHERE

\[
\text{grd1: } s \in \text{SERVICES not theorem } \\
\text{grd2: } \text{i_state}(\text{token_owner}(s) \mapsto s) = \text{FAIL_ACTIV}_4 \text{ not theorem } \\
\text{grd3: } \text{card}(\text{run_peers}(s)) < \text{min_inst}(s) \text{ not theorem }
\]

THEN

\[
\text{act1: } \text{i_state}(\text{token_owner}(s) \mapsto s) = \text{FAIL_CONFIG}_4
\]

END

\[
\text{FAIL_IGNORE: extended ordinary } \\
\text{REFINES } \\
\text{FAIL_IGNORE}
\]

ANY \( s \)

WHERE

\[
\text{grd1: } s \in \text{SERVICES not theorem } \\
\text{grd2: } \text{i_state}(\text{token_owner}(s) \mapsto s) = \text{FAIL_ACTIV}_4 \text{ not theorem } \\
\text{grd3: } \text{card}(\text{run_peers}(s)) \geq \text{min_inst}(s) \text{ not theorem }
\]

THEN

\[
\text{act1: } \text{i_state}(\text{token_owner}(s) \mapsto s) = \text{FAIL_IGN}_4
\]

END

\[
\text{IGNORE: extended ordinary } \\
\text{REFINES } \\
\text{IGNORE}
\]

ANY \( s \)

WHERE

\[
\text{grd1: } s \in \text{SERVICES not theorem } \\
\text{grd2: } \text{i_state}(\text{token_owner}(s) \mapsto s) = \text{FAIL_IGN}_4 \text{ not theorem }
\]

THEN

\[
\text{act1: } \text{i_state}(\text{token_owner}(s) \mapsto s) = \text{RUN}_4
\]

END
M15

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) u failr_peers(s) u unav_peers u dep_instc(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

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)

act2: actv_instc(token_owner(s) ↦ s) = ø

END

REDEPLOY: not extended ordinary
REFINES
REDEPLOY

ANY
s →
WHERE
  grd1: \( s \in \text{SERVICES} \) not theorem >
  grd2: \( \text{i\_state}(\text{token\_owner}(s) \mapsto s) = \text{FAIL\_CONFIG\_4} \) not theorem >
  grd7: \( \text{actv\_instc}(\text{token\_owner}(s) \mapsto s) = \emptyset \) not theorem >
  grd6: \( \text{dep\_instc}(s) \neq \emptyset \) not theorem >
  grd4: \( \text{card}(\text{run\_peers}(s)) + \text{card}(\text{dep\_instc}(s)) \geq \text{min\_inst}(s) \) not theorem >

THEN
  act1: \( \text{i\_state}(\text{token\_owner}(s) \mapsto s) = \text{DPL\_4} \) >
  act2: \( \text{run\_peers}(s) = \text{run\_peers}(s) \cup \text{dep\_instc}(s) \) >
  act3: \( \text{dep\_instc}(s) = \emptyset \) >

END

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

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

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

    act1: unav_peers = unav_peers \ {p} >

END

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

INVARIANTS
  inv1: inst_state ∈ (PEERS×SERVICES) ⇸ STATES_4 not theorem
  inv2: ∀ s · s ∈ SERVICES ⇒ token_owner(s) ↦ s ∈ dom(inst_state) not theorem
  gluing_state_1: ∀ s · s ∈ SERVICES ⇒ i_state(token_owner(s) ↦ s) = inst_state(token_owner(s) ↦ s) not theorem
  inv3: ∀ s · s ∈ SERVICES ⇒ rctt_inst(token_owner(s) ↦ s) ⊆ run_peers(s) not theorem
  inv4: ∀ s · s ∈ SERVICES ⇒ suspc_peers(token_owner(s) ↦ s) ⊆ run_peers(s) not theorem
  inv5: ∀ s · s ∈ SERVICES ⇒ suspc_inst(token_owner(s) ↦ s) ⊆ run_peers(s) not theorem
  inv6: ∀ s · s ∈ SERVICES ⇒ rect_inst(token_owner(s) ↦ s) ⊆ run_peers(s) not theorem
  inv7: ∀ s · s ∈ SERVICES ⇒ token_owner(s) ∉ suspc_inst(token_owner(s) ↦ s) not theorem
  inv8: ∀ s · s ∈ SERVICES ⇒ token_owner(s) ∉ suspc_peers(token_owner(s) ↦ s) not theorem
  inv9: ∀ s · s ∈ SERVICES ⇒ token_owner(s) ∉ rctt_inst(token_owner(s) ↦ s) not theorem
  inv10: ∀ s · s ∈ SERVICES ⇒ token_owner(s) ∉ rect_inst(token_owner(s) ↦ s) not theorem
  inv11: ∀ s · s ∈ SERVICES ⇒ suspc_inst(token_owner(s) ↦ s) ∩ suspc_peers(token_owner(s) ↦ s) = ∅ not theorem
  inv12: ∀ s · s ∈ SERVICES ∧ inst_state(token_owner(s) ↦ s) ∉ {FAIL_4,FAIL_DETECT_4} ⇒ suspc_peers(token_owner(s) ↦ s) = ∅ not theorem
  inv13: ∀ s · s ∈ SERVICES ∧ inst_state(token_owner(s) ↦ s) ≠ FAIL_4 ⇒ rctt_inst(token_owner(s) ↦ s) = ∅ not theorem
  inv14: ∀ s · s ∈ SERVICES ∧ inst_state(token_owner(s) ↦ s) ≠ FAIL_4 ⇒ rect_inst(token_owner(s) ↦ s) = ∅ not theorem

Page 1
EVENTS
INITIALISATION: not extended ordinary

THEN
act1: run_peers = InitSrvcPeers
act2: suspc_peers = InitSuspPeers
act3: failr_peers = InitFail
act4: dep_instc = InitFail
act5: token_owner = init_tok
act6: unav_peers = ∅
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
new values for token owner per service if needed
p_s
s_i
rc_s
rt_s
ac_i

WHERE
grd1: prs ⊆ PEERS not theorem
grd2: prs ⊈ unav_peers not theorem
grd3: ∀ srv · srv ∈ SERVICES ⇒ dom(dom(inst_state) ▷ {sr v}) \prs ≠ ∅ not theorem
grd4: E ∈ SERVICES → PEERS not theorem new value for token owner per service if needed
grd5: p_s ∈ (PEERS×SERVICES) ↦ ℙ(PEERS) not theorem
grd6: s_i ∈ (PEERS×SERVICES) ↦ ℙ(PEERS) not theorem
grd7: rt_s ∈ (PEERS×SERVICES) ↦ ℙ(PEERS) not theorem
grd8: rc_s ∈ (PEERS×SERVICES) ↦ ℙ(PEERS) not theorem
grd9: ac_i ∈ (PEERS×SERVICES) ↦ ℙ(PEERS) not theorem
grd10: dom(p_s) = E~ ∧ dom(s_i) = E~ ∧ dom(rc_s) = E~ ∧ dom(rt_s) = E~ ∧ dom(ac_i) = E~ not theorem
grd11: ∀ srv · srv ∈ SERVICES ∧ token_owner(srv) ∉ prs
⇒
E(srv) = token_owner(srv) ∧
s_i(E(srv) ↦ srv) = suspc_inst(E(srv) ↦ srv) ∧
rt_s(E(srv) ↦ srv) = rctt_inst(E(srv) ↦ srv) ∧
rc_s(E(srv) ↦ srv) = rect_inst(E(srv) ↦ srv) ∧
ac_i(E(srv) ↦ srv) = actv_instc(E(srv) ↦ srv) not
Theorem If the token owner of a service \( \text{srv} \) does not belong to \( \text{prs} \), the token owner is not changed

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

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

\[
\text{grd13: } \forall \text{srv} \cdot \text{srv} \in \text{SERVICES} \Rightarrow \text{p\_s(E(srv) \Rightarrow srv)} = \\
\text{susp\_peers(token\_owner(\text{srv}))} \land \\
\text{WITH} \\
\text{i\_s: } i\_s = E\_\left\langle \text{inst\_state} \right\rangle \\
\text{THEN} \\
\text{act1: } \text{unav\_peers} = \text{unav\_peers} \cup \text{prs} \Rightarrow \text{the peers in prs become unavailable} \\
\text{act2: } \text{token\_owner} = \text{token\_owner} \Rightarrow E \Rightarrow \text{new value for token owner per service is given if needed} \\
\text{act3: } \text{rect\_inst} = \text{rc\_s} \Rightarrow \text{the peers in prs can not try to recontact instances anymore} \\
\text{act4: } \text{rctt\_inst} = \text{rt\_s} \Rightarrow \text{the peers in prs can not recontact instances anymore} \\
\text{act5: } \text{activ\_instc} = \text{ac\_i} \\
\text{act6: } \text{susp\_peers} = \text{p\_s} \\
\text{act7: } \text{susp\_inst} = \text{s\_i} \\
\text{act8: } \text{inst\_state} = (\text{prs} \times \text{SERVICES}) \Rightarrow \text{inst\_state}
\]

\text{END}

\text{SUSPECT\_INST: not 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: } s \in \text{SERVICES} \text{ not theorem} \\
\text{grd2: } \text{susp} \notin \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}
M16

grd4: \( \text{suspc\_inst}(\text{token\_owner}(s) \mapsto s) = \emptyset \) not theorem \( \rightarrow \) the member of susp have not yet been suspected for \( s \) by the token owner of \( s \)
grd5: \( \text{inst\_state}(\text{token\_owner}(s) \mapsto s) = \text{RUN\_4} \) not theorem \( \rightarrow \) the state of \( s \) is OK
grd6: \( \text{suspc} \neq \emptyset \) not theorem \( \rightarrow \)

THEN

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

END

FAIL: not extended ordinary \( \rightarrow \)

REFINES

FAIL

ANY

s \( \rightarrow \) a service \( s \)
prop \( \rightarrow \) prop

WHERE

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

THEN

act1: \( \text{inst\_state} = \text{inst\_state} \setminus (\text{prop} \times \{s\}) \times \{\text{FAIL\_4}\} \) \( \rightarrow \)
act2: \( \text{suspc\_peers}(\text{token\_owner}(s) \mapsto s) = \text{suspc\_inst}(\text{token\_owner}(s) \mapsto s) \) \( \rightarrow \)

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

END

RECONTACT\_INST\_OK: not extended ordinary \( \rightarrow \)

REFINES

RECONTACT\_INST\_OK

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: \( \text{inst\_state}(\text{token\_owner}(s) \mapsto s) = \text{FAIL\_4} \) not theorem \( \rightarrow \) the state of \( s \) is SUSPICIOUS
grd4: \( \text{suspc\_peers}(\text{token\_owner}(s) \mapsto s) \neq \emptyset \) not theorem \( \rightarrow \) the set of suspicious peers for \( s \) is not empty
grd5: \( i \in \text{suspc\_peers}(\text{token\_owner}(s) \mapsto s) \setminus \text{unav\_peers} \) not theorem \( \rightarrow \) \( i \) is a suspicious instance of \( s \) and \( i \) is available (can be contacted)
grd6: \( i \not\in \text{rect\_inst}(\text{token\_owner}(s) \mapsto s) \) not theorem \( \rightarrow \) the token owner of \( s \) has not yet tried to recontact \( i \)
grd7: \( \text{rect\_inst}(\text{token\_owner}(s) \mapsto s) \subseteq \text{suspc\_peers}(\text{token\_owner}(s) \mapsto s) \) not theorem \( \rightarrow \)
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 SERVICES not theorem \rightarrow the state of s is SUSPICIOUS
grd2: i \in PEERS not theorem \rightarrow the set of suspicious peers for s is not empty
grd3: \text{inst_state}(\text{token_owner}(s) \rightarrow s) = \text{FAIL}_4 not theorem \rightarrow i is a suspicious instance of s and is unavailable (can not be contacted)
grd4: \text{suspc_peers}(\text{token_owner}(s) \rightarrow s) \neq \emptyset not theorem \rightarrow i is a suspicious instance of s and is unavailable (can not be contacted)
grd5: \text{rect_inst}(\text{token_owner}(s) \rightarrow s) = \emptyset not theorem \rightarrow i is a suspicious instance of s and is unavailable (can not be contacted)
grd6: \text{rect_inst}(\text{token_owner}(s) \rightarrow s) \neq \emptyset not theorem \rightarrow the token owner of s has not yet tried to recontact i
grd7: \text{rect_inst}(\text{token_owner}(s) \rightarrow s) \subseteq \text{suspc_peers}(\text{token_owner}(s) \rightarrow s) \neq \emptyset 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) \rightarrow s) = \text{rect_inst}(\text{token_owner}(s) \rightarrow 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 a service s
prop \rightarrow
WHERE
grd1: s \in SERVICES not theorem \rightarrow the state of s is SUSPICIOUS
grd2: prop \in PEERS not theorem \rightarrow the set of suspicious peers for s is not empty
grd3: \text{inst_state}(\text{token_owner}(s) \rightarrow s) = \text{FAIL}_4 not theorem \rightarrow the token owner of s has not yet tried to recontact all the suspicious instances of s
grd4: \text{suspc_peers}(\text{token_owner}(s) \rightarrow s) \neq \emptyset not theorem \rightarrow the token owner of s has not yet tried to recontact all the suspicious instances of s
grd5: \text{rect_inst}(\text{token_owner}(s) \rightarrow s) = \text{suspc_peers}(\text{token_owner}(s) \rightarrow s) \neq \emptyset not theorem \rightarrow the token owner of s has not yet tried to recontact all the suspicious instances of s
grd6: \text{rect_inst}(\text{token_owner}(s) \rightarrow s) \neq \emptyset not theorem \rightarrow the token owner of s has not yet tried to recontact i
grd7: \text{rect_inst}(\text{token_owner}(s) \rightarrow s) \subseteq \text{suspc_peers}(\text{token_owner}(s) \rightarrow s) \neq \emptyset 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) \rightarrow s) = \text{rect_inst}(\text{token_owner}(s) \rightarrow s) \cup \{i\} \rightarrow the token owner of s has tried to recontact i
END
M16

grd6:  prop = ((run_peers(s) \ suspc_peers(token_owner(s) \ s)) u rctt_inst(token_owner(s)\ s))\unav_peers not theorem

THEN
act1:  inst_state = inst_state \ ((prop\{s}\{\{FAIL_DETECT_4\}\)

act2:  suspc_peers(token_owner(s) ↦ s) = suspc_peers
(token_owner(s) ↦ s) \ rctt_inst(token_owner(s) ↦ s)
act3:  rectt_inst(token_owner(s) ↦ s) = ø
act4:  rctt_inst(token_owner(s) ↦ s) = ø

END

IS_OK:  not extended ordinary
REFINES
IS_OK

ANY
s

prop

WHERE
grd1:  s ∈ SERVICES not theorem
grd2:  prop ⊆ PEERS not theorem
grd3:  inst_state(token_owner(s) ↦ s) = FAIL_DETECT_4 not

THEN
grd4:  suspc_peers(token_owner(s) ↦ s) = ø not theorem
grd5:  prop = run_peers(s)\unav_peers not theorem

THEN
act1:  inst_state = inst_state \ ((prop\{s}\{\{RUN_4\}\)

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

THEN
grd4:  suspc_peers(token_owner(s) ↦ s) ≠ ø not theorem
grd5:  prop = run_peers(s) \ (unav_peers u suspc_peers
(token_owner(s) ↦ s)) not theorem

THEN
act1:  inst_state = inst_state \ ((prop\{s}\{\{FAIL_ACTIV_4\}\)
act2:  run_peers(s) = run_peers(s) \ suspc_peers(token_owner(s)

failr_peers(s) = failr_peers(s) u suspc_peers
(token_owner(s) ↦ s)
M16

act4: suspc_peers(token_owner(s) ↦ 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_ACTIV_4 not theorem >

grd4: card(run_peers(s)) < min_inst(s) not theorem >

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

THEN

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

END

FAIL_IGNORE: not 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_peers(s)) ≥ min_inst(s) not theorem >

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

THEN

act1: inst_state = inst_state ≔ ((prop×{s})×{FAIL_IGN_4}) >

END

IGNORE: not 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 >


REDEPLOY INSTC: not extended ordinary

REDEPLOY INSTS: not extended ordinary

REDEPLOY INSTC

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

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

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

MAKE_PEER_AVAIL: extended ordinary 
REFINES MAKE_PEER_AVAIL
ANY
p
WHERE
grd1: p ∈ 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
suspc_peers
failr_peers
dep_instc
token_owner
unav_peers
suspc_inst
rect_inst
rctt_inst
actv_instc
instances that are tried to be recontacted
instances effectively recontacted after a try
instances activated by token owners
instances activated by token owners
INVARIANTS
inv1: suspct_peers ∈ (PEERS×SERVICES) ↔ ℙ(PEERS) not theorem
inv2: ∀ s · s ∈ SERVICES ⇒ token_owner(s) ↦ s ∈ dom(suspc_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
M17

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

grd3: \(\forall \text{srv} \cdot \text{srv} \in \text{SERVICES} \implies \text{dom}(\text{dom}(\text{inst_state}) \setminus \{\text{srv}\}) \neq \emptyset\)

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

grd6: \(\forall \text{srv} \cdot \text{srv} \in \text{SERVICES} \land \text{token_owner}(\text{srv}) \in \text{prs} \implies E(\text{srv}) \in \text{run_peers}(\text{srv}) \setminus \text{unav_peers} \cup \text{prs} \cup \text{failr_peers}(\text{srv}) \cup \text{suspct_peers}(\text{token_owner}(\text{srv}) \mapsto \text{srv}) \land\)

\(\text{suspct_peers} \land\)

\(\text{inst_state}(E(\text{srv}) \mapsto \text{srv}) = \text{inst_state}(\text{token_owner}(\text{srv}) \mapsto \text{srv})\) 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

\(p_s\): \(p_s = \ E \sim \text{\ suspct_peers}\)

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

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

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

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

THEN

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

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

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

Page 2
anymore (1)

act4: \( rctt\_inst = ((prs\times SERVICES) \triangleleft rctt\_inst) \triangleleft ((({E}\setminus token\_owner)\setminus)\times\emptyset) \)  
> the peers in prs can not try to recontact instances anymore (2)

act5: \( actv\_instc = ((prs\times SERVICES) \triangleleft actv\_instc) \triangleleft ((({E}\setminus token\_owner)\setminus)\times\emptyset) \)  
> the peers in prs can not activate instances anymore

act6: \( suspct\_peers = (prs\times SERVICES) \triangleleft suspct\_peers \)  
> the peers in prs can not suspect instances anymore (1)

act7: \( suspc\_inst = ((prs\times SERVICES) \triangleleft suspc\_inst) \triangleleft ((({E}\setminus token\_owner)\setminus)\times\emptyset) \)  
> the peers in prs can not suspect instances anymore (2)

act8: \( inst\_state = (prs\times SERVICES) \triangleleft inst\_state \)  
> the peers in prs can not monitor the state of the services provided anymore

END

SUSPECT\_INST: extended ordinary >
REFINES
SUSPECT\_INST

ANY
s > a service s
susp > suspicious instances

WHERE

grd1: \( s \in SERVICES \) not theorem >
grd2: \( susp \subseteq PEERS \) not theorem >
grd3: \( susp = run\_peers(s) \cap unav\_peers \) not theorem >

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

grd4: \( suspc\_inst(token\_owner(s) \mapsto s) = \emptyset \) not theorem >

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

grd5: \( inst\_state(token\_owner(s) \mapsto s) = RUN\_4 \) not theorem >

the state of s is OK

grd6: \( susp \neq \emptyset \) not theorem >

THEN

act1: \( suspc\_inst(token\_owner(s) \mapsto s) = susp \) >

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

END

FAIL: not extended ordinary >
REFINES
FAIL

ANY
s >
prop >

WHERE

grd1: \( s \in SERVICES \) not theorem >
grd2: \( prop \subseteq PEERS \) not theorem >
grd3: \( inst\_state(token\_owner(s) \mapsto s) = RUN\_4 \) not theorem >
grd4: \( suspc\_inst(token\_owner(s) \mapsto s) \neq \emptyset \) not theorem >
grd5: \( prop = run\_peers(s)\setminus(suspc\_inst(token\_owner(s) \mapsto s) \cup unav\_peers) \) not theorem >
THEN
    \begin{align*}
    & \text{act1: } \text{inst\_state} = \text{inst\_state} \triangleleft ((\text{prop} \times \{s\}) \times \{\text{FAIL\_4}\}) \\
    & \text{act2: } \text{suspct\_peers} = \text{suspct\_peers} \triangleleft ((\text{prop} \times \{s\}) \times \{\text{suspc\_inst}(\text{token\_owner}(s) \mapsto s)\}) \\
    & \text{act3: } \text{suspc\_inst}(\text{token\_owner}(s) \mapsto s) = \emptyset
    \end{align*}
END

\text{RECONTACT\_INST\_OK: } \text{not extended ordinary}
\begin{align*}
& \text{REFINES } \text{RECONTACT\_INST\_OK} \\
& \text{ANY}\quad s \, \rightarrow \text{a service } s \\
& \quad i \, \rightarrow \text{an instance } i \\
& \text{WHERE}\quad \\
& \quad \text{grd1: } s \in \text{SERVICES} \, \text{not theorem} \\
& \quad \text{grd2: } i \in \text{PEERS} \, \text{not theorem} \\
& \quad \text{grd3: } \text{inst\_state}(\text{token\_owner}(s) \mapsto s) = \text{FAIL\_4} \, \text{not theorem} \quad \text{the state of } s \text{ is SUSPICIOUS} \\
& \quad \text{grd4: } \text{suspct\_peers}(\text{token\_owner}(s) \mapsto s) \neq \emptyset \, \text{not theorem} \quad \text{the set of suspicious peers for } s \text{ is not empty} \\
& \quad \text{grd5: } i \in \text{suspct\_peers}(\text{token\_owner}(s) \mapsto s) \setminus \text{unav\_peers} \, \text{not theorem} \quad \text{i is a suspicious instance of } s \text{ and is available (can be contacted)} \\
& \quad \text{grd6: } i \notin \text{rect\_inst}(\text{token\_owner}(s) \mapsto s) \, \text{not theorem} \quad \text{the token owner of } s \text{ has not yet tried to recontact } i \\
& \quad \text{grd7: } \text{rect\_inst}(\text{token\_owner}(s) \mapsto s) \subseteq \text{suspct\_peers}(\text{token\_owner}(s) \mapsto s) \, \text{not theorem} \quad \text{the token owner of } s \text{ has not yet tried to recontact all the suspicious instances of } s
\end{align*}
THEN
    \begin{align*}
    & \text{act1: } \text{rect\_inst}(\text{token\_owner}(s) \mapsto s) = \text{rect\_inst}(\text{token\_owner}(s) \mapsto s) \cup \{i\} \quad \text{the token owner of } s \text{ has tried to recontact } i \\
    & \text{act2: } \text{rctt\_inst}(\text{token\_owner}(s) \mapsto s) = \text{rctt\_inst}(\text{token\_owner}(s) \mapsto s) \cup \{i\} \quad \text{i is recontacted by the token owner of } s \text{ successfully}
    \end{align*}
END

\text{RECONTACT\_INST\_KO: } \text{not extended ordinary}
\begin{align*}
& \text{REFINES } \text{RECONTACT\_INST\_KO} \\
& \text{ANY}\quad s \, \rightarrow \text{a service } s \\
& \quad i \, \rightarrow \text{an instance } i \\
& \text{WHERE}\quad \\
& \quad \text{grd1: } s \in \text{SERVICES} \, \text{not theorem} \\
& \quad \text{grd2: } i \in \text{PEERS} \, \text{not theorem} \\
& \quad \text{grd3: } \text{inst\_state}(\text{token\_owner}(s) \mapsto s) = \text{FAIL\_4} \, \text{not theorem} \quad \text{the state of } s \text{ is SUSPICIOUS} \\
& \quad \text{grd4: } \text{suspct\_peers}(\text{token\_owner}(s) \mapsto s) \neq \emptyset \, \text{not theorem} \quad \text{the set of suspicious peers for } s \text{ is not empty} \\
& \quad \text{grd5: } i \in \text{suspct\_peers}(\text{token\_owner}(s) \mapsto s) \setminus \text{unav\_peers} \, \text{not theorem} \quad \text{i is a suspicious instance of } s \text{ and is unavailable (can not be contacted)} \\
& \quad \text{grd6: } i \notin \text{rect\_inst}(\text{token\_owner}(s) \mapsto s) \, \text{not theorem} \quad \text{the token owner of } s \text{ has not yet tried to recontact } i \\
& \quad \text{grd7: } \text{rect\_inst}(\text{token\_owner}(s) \mapsto s) \subseteq \text{suspct\_peers}(\text{token\_owner}(s) \mapsto s) \, \text{not theorem} \quad \text{the token owner of } s \text{ has not yet tried to recontact all the suspicious instances of } s
\end{align*}
M17

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

\begin{align*}
\text{grd6: } & i \notin \text{rect\_inst(token\_owner(s) ↦ s)} \quad \text{not theorem} \quad \triangleright \text{the token owner of } s \text{ has not yet tried to recontact } i \\
\text{grd7: } & \text{rect\_inst(token\_owner(s) ↦ s)} \subseteq \text{suspect\_peers(token\_owner(s) ↦ s)} \quad \text{not theorem} \quad \triangleright \text{the token owner of } s \text{ has not yet tried to recontact all the suspicious instances of } s \\
\end{align*}

\begin{align*}
\text{THEN} \\
\text{act1: } & \text{rect\_inst(token\_owner(s) ↦ s)} = \text{rect\_inst(token\_owner(s) ↦ s)} \cup \{i\} \\
\end{align*}

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

\text{ANY}

\begin{align*}
& s \\
& \text{prop} \\
& \text{suspect} \\
\text{WHERE}
\end{align*}

\begin{align*}
\text{grd1: } & s \in \text{SERVICES} \quad \text{not theorem} \quad \triangleright \\
\text{grd2: } & \text{prop} \subseteq \text{PEERS} \quad \text{not theorem} \quad \triangleright \\
\text{grd7: } & \text{suspect} \subseteq \text{PEERS} \quad \text{not theorem} \quad \triangleright \\
\text{grd3: } & \text{inst\_state(token\_owner(s) ↦ s)} = \text{FAIL\_4} \quad \text{not theorem} \quad \triangleright \\
\text{grd4: } & \text{suspect\_peers(token\_owner(s) ↦ s)} \neq \emptyset \quad \text{not theorem} \quad \triangleright \\
\text{grd5: } & \text{rect\_inst(token\_owner(s) ↦ s)} = \text{suspect\_peers(token\_owner(s) ↦ s)} \quad \text{not theorem} \quad \triangleright \\
\text{grd6: } & \text{prop} = \left((\text{run\_peers(s)} \setminus \text{suspect\_peers(token\_owner(s) ↦ s)}) \cup \text{rctt\_inst(token\_owner(s) ↦ s)}) \setminus \text{unav\_peers} \right) \quad \text{not theorem} \quad \triangleright \\
\text{grd8: } & \text{suspect} = \text{suspect\_peers(token\_owner(s) ↦ s)} \setminus \text{rctt\_inst(token\_owner(s) ↦ s)} \quad \text{not theorem} \quad \triangleright \\
\text{THEN} \\
\text{act1: } & \text{inst\_state} = \text{inst\_state} \setminus \left((\text{prop} \times \{s\}) \times \{\text{FAIL\_DETECT\_4}\}\right) \\
\text{act2: } & \text{suspect\_peers} = \text{suspect\_peers} \setminus \left((\text{prop} \times \{s\}) \times \{\text{suspect}\}\right) \\
\text{act3: } & \text{rect\_inst(token\_owner(s) ↦ s)} = \emptyset \\
\text{act4: } & \text{rctt\_inst(token\_owner(s) ↦ s)} = \emptyset \\
\text{END}
\end{align*}

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

\text{ANY}

\begin{align*}
& s \\
& \text{prop} \\
\text{WHERE}
\end{align*}

\begin{align*}
\text{grd1: } & s \in \text{SERVICES} \quad \text{not theorem} \quad \triangleright \\
\text{grd2: } & \text{prop} \subseteq \text{PEERS} \quad \text{not theorem} \quad \triangleright
\end{align*}
M17

\begin{align*}
&\text{grd3: } \text{inst\_state}(\text{token\_owner}(s) \mapsto s) = \text{FAIL\_DETECT\_4} \not\text{ theorem} \\
&\text{grd4: } \text{suspct\_peers}(\text{token\_owner}(s) \mapsto s) = \emptyset \not\text{ theorem} \quad \text{grd5: } \text{prop} = \text{run\_peers}(s) \setminus \text{unav\_peers} \not\text{ theorem} \\
&\text{THEN} \\
&\quad \text{act1: } \text{inst\_state} = \text{inst\_state} \setminus ((\text{prop} \times \{s\}) \times \{\text{RUN\_4}\}) \quad \text{act2: } \text{run\_peers}(s) = \text{run\_peers}(s) \setminus \text{suspct\_peers}(\text{token\_owner}(s) \mapsto s) \\
&\quad \quad \text{act3: } \text{failr\_peers}(s) = \text{failr\_peers}(s) \cup \text{suspct\_peers}(\text{token\_owner}(s) \mapsto s) \\
&\quad \quad \text{act4: } \text{suspct\_peers} = \text{suspct\_peers} \setminus ((\text{prop} \times \{s\}) \times \{\emptyset\}) \\
&\text{END}
\end{align*}

FAIL\_ACTIV:\not\text{ extended ordinary} \\
\text{REFINES} \quad \text{FAIL\_ACTIV} \\
\text{ANY} \\
\quad s \quad \text{prop} \\
\text{WHERE} \\
\quad \text{grd1: } s \in \text{SERVICES} \not\text{ theorem} \\
\quad \text{grd2: } \text{prop} \subseteq \text{PEERS} \not\text{ theorem} \\
\quad \text{grd3: } \text{inst\_state}(\text{token\_owner}(s) \mapsto s) = \text{FAIL\_DETECT\_4} \not\text{ theorem} \\
\text{THEN} \\
\quad \text{grd4: } \text{suspct\_peers}(\text{token\_owner}(s) \mapsto s) \neq \emptyset \not\text{ theorem} \\
\quad \text{grd5: } \text{prop} = \text{run\_peers}(s) \setminus (\text{unav\_peers} \cup \text{suspct\_peers}(\text{token\_owner}(s) \mapsto s)) \not\text{ theorem} \\
\text{END}

\begin{align*}
&\text{FAIL\_CONFIGURE: } \text{extended ordinary} \\
&\text{REFINES} \quad \text{FAIL\_CONFIGURE} \\
&\text{ANY} \\
&\quad s \quad \text{prop} \\
&\text{WHERE} \\
&\quad \text{grd1: } s \in \text{SERVICES} \not\text{ theorem} \\
&\quad \text{grd2: } \text{prop} \subseteq \text{PEERS} \not\text{ theorem} \\
&\quad \text{grd3: } \text{inst\_state}(\text{token\_owner}(s) \mapsto s) = \text{FAIL\_ACTIV\_4} \not\text{ theorem} \\
\text{THEN} \\
&\quad \text{grd4: } \text{card}(\text{run\_peers}(s)) < \text{min\_inst}(s) \not\text{ theorem} \\
&\quad \text{grd5: } \text{prop} = \text{run\_peers}(s) \setminus \text{unav\_peers} \not\text{ theorem} \\
&\text{THEN} \\
&\quad \text{act1: } \text{inst\_state} = \text{inst\_state} \setminus ((\text{prop} \times \{s\}) \times \{\text{FAIL\_CONFIG\_4}\}) \\
&\text{END}
\end{align*}
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_peers(s)) ≥ min_inst(s) not theorem
  grd5: prop = run_peers(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_peers(s) \ unav_peers not theorem
THEN
  act1: inst_state = inst_state ≔ ((prop×{s})×{RUN_4})
END

REDEPLOY_INSTC: extended ordinary
REFINES REDEPLOY_INSTC
ANY
s → a service s
i → an instance i
WHERE
  grd1: s ∈ SERVICES not theorem
  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
M17

\[ \text{grd6: } \text{card}(\text{actv\_instc}(\text{token\_owner}(s) \rightarrow s)) < \text{deplo\_inst}(s) \text{ not theorem} \]

\[ \text{grd7: } \text{card}(\text{dep\_instc}(s)) + \text{card}(\text{run\_peers}(s)) < \text{min\_inst}(s) \text{ not theorem} \]

\[ \text{THEN} \]

\[ \text{act1: } \text{actv\_instc}(\text{token\_owner}(s) \rightarrow s) = \text{actv\_instc}(\text{token\_owner}(s) \rightarrow s) \cup \{i\} \]

\[ \text{END} \]

\[ \text{REDEPLOY\_INSTS: extended ordinary} \]

\[ \text{REFINES} \]

\[ \text{REDEPLOY\_INSTS} \]

\[ \text{ANY} s \]

\[ \text{WHERE} \]

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

\[ \text{grd2: } \text{card}(\text{actv\_instc}(\text{token\_owner}(s) \rightarrow s)) = \text{deplo\_inst}(s) \text{ not theorem} \]

\[ \text{grd3: } \text{card}(\text{dep\_instc}(s)) + \text{card}(\text{run\_peers}(s)) < \text{min\_inst}(s) \text{ not theorem} \]

\[ \text{grd4: } \text{inst\_state}(\text{token\_owner}(s) \rightarrow s) = \text{FAIL\_CONFIG\_4 not theorem} \]

\[ \text{THEN} \]

\[ \text{act1: } \text{dep\_instc}(s) = \text{dep\_instc}(s) \cup \text{actv\_instc}(\text{token\_owner}(s) \rightarrow s) \]

\[ \text{act2: } \text{actv\_instc}(\text{token\_owner}(s) \rightarrow s) = \emptyset \]

\[ \text{END} \]

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

\[ \text{REFINES} \]

\[ \text{REDEPLOY} \]

\[ \text{ANY} s \]

\[ \text{prop} \]

\[ \text{WHERE} \]

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

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

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

\[ \text{grd4: } \text{actv\_instc}(\text{token\_owner}(s) \rightarrow s)=\emptyset \text{ not theorem} \]

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

\[ \text{grd6: } \text{card}(\text{run\_peers}(s)) + \text{card}(\text{dep\_instc}(s)) \geq \text{min\_inst}(s) \text{ not theorem} \]

\[ \text{grd7: } \text{prop} = \text{run\_peers}(s) \setminus \text{unav\_peers not theorem} \]

\[ \text{THEN} \]

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

\[ \text{act2: } \text{run\_peers}(s) = \text{run\_peers}(s) \cup \text{dep\_instc}(s) \]

\[ \text{act3: } \text{dep\_instc}(s) = \emptyset \]
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 >
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 · s ∈ SERVICES ⇒ token_owner(s) ↦ s ∈ dom(run_inst) not theorem
   gluing_run_1: ∀ s · s ∈ SERVICES ⇒ run_inst(token_owner(s) ↦ s) = run_peers(s) not theorem
EVENTS
INITIALISATION: not extended ordinary
THEN
   act1: run_inst ≔ InitRunPeers
   act2: suspct_peers ≔ InitSuspPrs
   act3: failr_peers ≔ InitFail
   act4: dep_instc ≔ InitFail
   act5: token_owner ≔ init_tok
   act6: unav_peers ≔ ∅
   act7: suspc_inst ≔ InitSuspPeers
   act8: rect_inst ≔ InitSuspPeers
   act9: rctt_inst ≔ InitSuspPeers
   act10: actv_instc ≔ InitSuspPeers
   act11: inst_state ≔ InitStateSrv
END
MAKE_PEER_UNAVAIL: not extended ordinary
REFINES
MAKE_PEER_UNAVAIL
ANY
prs > Peers that will become unavailable
E > Values for token owner per service
WHERE
   grd1: prs ⊆ PEERS not theorem

Page 1
grd2:  \( \text{prs} \not\in \text{unav peers} \) not theorem \(\rightarrow\) the peers in prs are not yet unavailable

grd3:  \( \forall \text{srv} \cdot \text{srv} \in \text{SERVICES} \Rightarrow \text{dom(\text{dom(\text{inst_state})} \setminus \{\text{srv}\})} \)
\(\not= \emptyset\) 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:  \( \forall \text{srv} \cdot \text{srv} \in \text{SERVICES} \wedge \text{token_owner(srv)} \in \text{prs} \Rightarrow \)
\( \text{E(srv)} \in \text{run_inst(token_owner(srv) ↦ srv) \setminus (unav_peers u prs u failr_peers(srv) u suspct_peers(token_owner(srv) ↦ srv))} \)
\(\wedge \)
\( \text{E(srv)} \wedge \text{srv} \in \text{dom(\text{inst_state}) \setminus (\text{dom(suspct_peers)} \cap \text{dom(run_inst)})} \wedge \)
\( \text{run_inst(E(srv) ↦ srv)} = \text{run_inst(token_owner(srv) ↦ srv)} \wedge \)
\( \text{inst_state(E(srv) ↦ srv)} = \text{inst_state(token_owner(srv) ↦ srv)} \wedge \)
\( \text{suspct_peers(E(srv) ↦ srv)} = \text{suspct_peers(token_owner(srv) ↦ srv)} \) 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

THEN

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

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

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

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

act5:  \( \text{actv_instc} = (((\text{prs} \times \text{SERVICES}) \bowtie \text{actv_instc}) \bowtie (((\text{\text{\text{E}}(\text{token_owner})\sim}) \times \{\emptyset\})) \) \(\rightarrow\) the peers in prs can not activate instances anymore (1)

act6:  \( \text{suspc_inst} = (((\text{prs} \times \text{SERVICES}) \bowtie \text{suspc_inst}) \bowtie (((\text{\text{\text{E}}(\text{token_owner})\sim}) \times \{\emptyset\})) \) \(\rightarrow\) the peers in prs can not suspect instances anymore (1)
M18

(((E\token_owner～)×{∅}) \rightarrow the peers in prs can not suspect instances anymore (2)

act7: suspc_inst ≔ ((prs×SERVICES) ⩤ suspc_inst) \rightarrow the peers in prs can not monitor the state of the services provided anymore

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

act9: run_inst = (prs×SERVICES) ⩤ run_inst \rightarrow

END

SUSPECT_INST: not extended ordinary \rightarrow

REFINES SUSPECT_INST

ANY

s \rightarrow a service s

susp \rightarrow suspicious instances

WHERE

grd1: s ∈ SERVICES not theorem \rightarrow

grd2: susp ⊆ PEERS not theorem \rightarrow

grd3: susp = run_inst(token_owner(s) ↦ s) ∩ unav_peers not theorem \rightarrow instances in susp are suspicious if the peers running them becomes unavailable

grd4: suspc_inst(token_owner(s) ↦ s) = ∅ not theorem \rightarrow 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 \rightarrow the state of s is OK

grd6: susp ≠ ∅ not theorem \rightarrow

THEN

act1: suspc_inst(token_owner(s) ↦ s) = susp \rightarrow the members of susp become suspected instances for s by the token owner of s

END

FAIL: not extended ordinary \rightarrow

REFINES FAIL

ANY

s \rightarrow

prop \rightarrow

WHERE

grd1: s ∈ SERVICES not theorem \rightarrow

grd2: prop ⊆ PEERS not theorem \rightarrow

grd3: inst_state(token_owner(s) ↦ s) = RUN_4 not theorem \rightarrow

grd4: suspc_inst(token_owner(s) ↦ s) ≠ ∅ not theorem \rightarrow

grd5: prop = run_inst(token_owner(s) ↦ s) \setminus (suspc_inst (token_owner(s) ↦ s) ∪ unav_peers) not theorem \rightarrow

THEN

act1: inst_state = inst_state \setminus ((prop×{s})×{FAIL_4}) \rightarrow

act2: suspct_peers = suspct_peers \setminus ((prop×{s})×{suspc_inst (token_owner(s) ↦ s)}) \rightarrow

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

END
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\implies \text{theorem} \implies \text{the state of s is SUSPICIOUS}
  grd2: i \in \text{PEERS} \not\implies \text{theorem} \implies \text{an instance i}
  grd3: \text{inst_state}(\text{token_owner(s) ↦ s}) = \text{FAIL}_4 \not\implies \text{theorem} \implies \text{the set of suspicious peers for s is not empty}
  grd4: \text{suspct_peers}(\text{token_owner(s) ↦ s}) \neq \emptyset \not\implies \text{theorem} \implies \text{the set of suspicious peers for s is not empty}
  grd5: i \in \text{suspct_peers}(\text{token_owner(s) ↦ s}) \not\implies \text{unav_peers not theorem} \implies \text{i is a suspicious instance of s and is available (can be contacted)}
  grd6: i \not\in \text{rect_inst}(\text{token_owner(s) ↦ s}) \not\implies \text{theorem} \implies \text{the token owner of s has not yet tried to recontact i}
  grd7: \text{rect_inst}(\text{token_owner(s) ↦ s}) \subset \text{suspct_peers}(\text{token_owner(s) ↦ s}) \not\implies \text{theorem} \implies \text{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) ↦ s}) \triangledown \text{rect_inst}(\text{token_owner(s) ↦ s}) \implies \text{the token owner of s has tried to recontact i}
act2: \text{rctt_inst}(\text{token_owner(s) ↦ s}) \triangledown \text{rctt_inst}(\text{token_owner(s) ↦ s}) \implies \text{i is recontacted by the token owner of s 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\implies \text{theorem} \implies \text{the state of s is SUSPICIOUS}
  grd2: i \in \text{PEERS} \not\implies \text{theorem} \implies \text{an instance i}
  grd3: \text{inst_state}(\text{token_owner(s) ↦ s}) = \text{FAIL}_4 \not\implies \text{theorem} \implies \text{the set of suspicious peers for s is not empty}
  grd4: \text{suspct_peers}(\text{token_owner(s) ↦ s}) \neq \emptyset \not\implies \text{theorem} \implies \text{the set of suspicious peers for s is not empty}
  grd5: i \in \text{suspct_peers}(\text{token_owner(s) ↦ s}) \cap \text{unav_peers not theorem} \implies \text{i is a suspicious instance of s and is unavailable (can not be contacted)}
  grd6: i \not\in \text{rect_inst}(\text{token_owner(s) ↦ s}) \not\implies \text{theorem} \implies \text{the token owner of s has not yet tried to recontact i}
  grd7: \text{rect_inst}(\text{token_owner(s) ↦ s}) \subset \text{suspct_peers}(\text{token_owner(s) ↦ s}) \not\implies \text{theorem} \implies \text{the token owner of s has not yet tried to recontact all the suspicious instances of s}
THEN
act1: rect_inst(token_owner(s) ↦ s) ≔ rect_inst(token_owner(s) ↦ s) ∪ {i}  # the token owner of s has tried to recontact i
END

FAIL_DETECT: not extended ordinary
REFINES
FAIL_DETECT

ANY
s
prop
 susp
WHERE
grd1: s ∈ SERVICES not theorem
grd2: prop ⊆ PEERS not theorem
grd7: susp ⊆ PEERS not theorem
grd3: inst_state(token_owner(s) ↦ s) = FAIL_DETECT_4 not theorem
grd4: suspct_peers(token_owner(s) ↦ s) ≠ ∅ not theorem
grd5: rect_inst(token_owner(s) ↦ s) = suspct_peers(token_owner(s) ↦ s) not theorem
grd8: (token_owner(s) ↦ s) not theorem
THEN
act1: inst_state = inst_state  ((prop×{s})×{FAIL_DETECT_4})
act2: suspct_peers = suspct_peers  ((prop×{s})×{suspc})
act3: rect_inst(token_owner(s) ↦ s) = ∅
act4: rctt_inst(token_owner(s) ↦ s) = ∅
END

IS_OK: not extended ordinary
REFINES
IS_OK

ANY
s
prop
WHERE
grd1: s ∈ SERVICES not theorem
grd2: prop ⊆ PEERS not theorem
grd3: inst_state(token_owner(s) ↦ s) = FAIL_DETECT_4 not theorem
grd4: suspct_peers(token_owner(s) ↦ s) = ∅ not theorem
grd5: prop = run_inst(token_owner(s) ↦ s)\unav_peers not theorem
THEN
act1: inst_state = inst_state  ((prop×{s})×{RUN_4})
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)\suspect_peers(token_owner(s) ↦ s)})
act3: failr_peers(s) = failr_peers(s) u suspect_peers
(token_owner(s) ↦ s)
act4: suspect_peers = suspect_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_ACTIV_4 not theorem

THEN

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

END

FAIL_IGNORE: not extended ordinary

REFINES
FAIL_IGNORE

ANY

Page 6
\( s \) \( \triangleright \)
\( \text{prop} \) \( \triangleright \)
WHERE
\[ \text{grd1}: \ s \in \text{SERVICES} \not\Rightarrow \text{theorem} \]
\[ \text{grd2}: \ \text{prop} \subseteq \text{PEERS} \not\Rightarrow \text{theorem} \]
\[ \text{grd3}: \ \text{inst}_{\text{state}}(\text{token}_{\text{owner}}(s) \mapsto s) = \text{FAIL}\_\text{ACTIV}\_4 \not\Rightarrow \text{theorem} \]
\[ \text{grd4}: \ \text{card}(\text{run}_{\text{inst}}(\text{token}_{\text{owner}}(s) \mapsto s)) \geq \text{min}_{\text{inst}}(s) \not\Rightarrow \text{theorem} \]
\[ \text{grd5}: \ \text{prop} = \text{run}_{\text{inst}}(\text{token}_{\text{owner}}(s) \mapsto s) \setminus \text{unav}_{\text{peers}} \not\Rightarrow \text{theorem} \]

THEN
\[ \text{act1}: \ \text{inst}_{\text{state}} = \text{inst}_{\text{state}} \triangleright ((\text{prop} \times \{s\}) \times \{\text{FAIL}\_\text{IGN}\_4\}) \]

END

\text{IGNORE}: \not\Rightarrow \text{extended}\_\text{ordinary} \]
\text{REFINES}
\text{IGNORE}
\text{ANY}
\[ s \] \( \triangleright \)
\[ \text{prop} \] \( \triangleright \)
WHERE
\[ \text{grd1}: \ s \in \text{SERVICES} \not\Rightarrow \text{theorem} \]
\[ \text{grd2}: \ \text{prop} \subseteq \text{PEERS} \not\Rightarrow \text{theorem} \]
\[ \text{grd3}: \ \text{inst}_{\text{state}}(\text{token}_{\text{owner}}(s) \mapsto s) = \text{FAIL}\_\text{IGN}\_4 \not\Rightarrow \text{theorem} \]
\[ \text{grd4}: \ \text{prop} = \text{run}_{\text{inst}}(\text{token}_{\text{owner}}(s) \mapsto s) \setminus \text{unav}_{\text{peers}} \not\Rightarrow \text{theorem} \]

THEN
\[ \text{act1}: \ \text{inst}_{\text{state}} = \text{inst}_{\text{state}} \triangleright ((\text{prop} \times \{s\}) \times \{\text{RUN}\_4\}) \]

END

\text{REDEPLOY}_{\text{INSTC}}: \not\Rightarrow \text{extended}\_\text{ordinary} \]
\text{REFINES}
\text{REDEPLOY}_{\text{INSTC}}
\text{ANY}
\[ s \] \( \triangleright \)
a service \( s \)
\[ i \] \( \triangleright \)
an instance \( i \)
WHERE
\[ \text{grd1}: \ s \in \text{SERVICES} \not\Rightarrow \text{theorem} \]
\[ \text{grd2}: \ i \in \text{PEERS} \not\Rightarrow \text{theorem} \]
\[ \text{grd3}: \ i \notin \text{run}_{\text{inst}}(\text{token}_{\text{owner}}(s) \mapsto s) \cup \text{failr}_{\text{peers}}(s) \cup \text{unav}_{\text{peers}} \cup \text{deplo}_{\text{instc}}(s) \not\Rightarrow \text{theorem} \]
i does not run \( s \), is not failed for \( s \), is not unavailable and is not already activated for \( s \)
\[ \text{grd4}: \ i \notin \text{actv}_{\text{instc}}(\text{token}_{\text{owner}}(s) \mapsto s) \not\Rightarrow \text{theorem} \]
\[ \text{grd5}: \ \text{inst}_{\text{state}}(\text{token}_{\text{owner}}(s) \mapsto s) = \text{FAIL}\_\text{CONFIG}\_4 \not\Rightarrow \text{theorem} \]
\[ \text{grd6}: \ \text{card}(\text{actv}_{\text{instc}}(\text{token}_{\text{owner}}(s) \mapsto s)) < \text{deplo}_{\text{inst}}(s) \not\Rightarrow \text{theorem} \]

Page 7
M18

THEOREM

grd7: \[ \text{card}(\text{dep_instc}(s)) + \text{card}(\text{run_inst}(\text{token_owner}(s) \mapsto s)) < \text{min_inst}(s) \not\text{ theorem} \]

THEN

act1: \[ \text{actv_instc}(\text{token_owner}(s) \mapsto s) = \text{actv_instc}(\text{token_owner}(s) \mapsto s) \cup \{i\} \]

END

REDEPLOY_INSTS: not extended ordinary

REFINES

REDEPLOY_INSTS

ANY

s

WHERE

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

grd2: \[ \text{card}(\text{actv_instc}(\text{token_owner}(s) \mapsto s)) = \text{deplo_inst}(s) \not\text{ theorem} \]

grd3: \[ \text{card}(\text{dep_instc}(s)) + \text{card}(\text{run_inst}(\text{token_owner}(s) \mapsto s)) < \text{min_inst}(s) \not\text{ theorem} \]

grd4: \[ \text{inst_state}(\text{token_owner}(s) \mapsto s) = \text{FAIL_CONFIG}_4 \not\text{ theorem} \]

THEN

act1: \[ \text{dep_instc}(s) = \text{dep_instc}(s) \cup \text{actv_instc}(\text{token_owner}(s) \mapsto s) \]

act2: \[ \text{actv_instc}(\text{token_owner}(s) \mapsto s) = \emptyset \]

END

REDEPLOY: not extended ordinary

REFINES

REDEPLOY

ANY

s

prop

WHERE

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

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

grd3: \[ \text{inst_state}(\text{token_owner}(s) \mapsto s) = \text{FAIL_CONFIG}_4 \not\text{ theorem} \]

grd4: \[ \text{actv_instc}(\text{token_owner}(s) \mapsto s) = \emptyset \not\text{ theorem} \]

grd5: \[ \text{dep_instc}(s) \not= \emptyset \not\text{ theorem} \]

grd6: \[ \text{card}(\text{run_inst}(\text{token_owner}(s) \mapsto s)) + \text{card}(\text{dep_instc}(s)) \geq \text{min_inst}(s) \not\text{ theorem} \]

grd7: \[ \text{prop} = \text{run_inst}(\text{token_owner}(s) \mapsto s) \setminus \text{unav_peers} \not\text{ theorem} \]

THEN

act1: \[ \text{inst_state} = \text{inst_state} \setminus ((\text{prop} \times s) \times \{\text{DPL}_4\}) \]

act2: \[ \text{run_inst} = \text{run_inst} \setminus ((\text{prop} \times s) \times \{\text{run_inst}(\text{token_owner}(s) \mapsto s) \cup \text{dep_instc}(s)\}) \]

Page 8
M18

act3: \( \text{dep\_instc}(s) = \emptyset \)  
END

HEAL: not extended ordinary  
REFINES  
HEAL
ANY  
s  
prop  
WHERE  
grd1: \( s \in \text{SERVICES} \) not theorem  
grd2: \( \text{prop} \subseteq \text{PEERS} \) not theorem  
grd3: \( \text{inst\_state}(\text{token\_owner}(s) \mapsto s) = \text{DPL\_4} \) not theorem  
grd4: \( \text{prop} = \text{run\_inst}(\text{token\_owner}(s) \mapsto s) \setminus \text{unav\_peers} \) not theorem  
THEN  
act1: \( \text{inst\_state} = \text{inst\_state} \circ (\text{prop} \times \{s\} \times \{\text{RUN\_4}\}) \)  
END

UNFAIL\_PEER: extended ordinary  
REFINES  
UNFAIL\_PEER
ANY  
s  
p  
WHERE  
grd1: \( s \in \text{SERVICES} \) not theorem  
grd2: \( p \in \text{PEERS} \) not theorem  
grd3: \( p \in \text{failr\_peers}(s) \) not theorem  
THEN  
act1: \( \text{failr\_peers}(s) = \text{failr\_peers}(s) \setminus \{p\} \)  
END

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

END

Page 9
M19

MACHINE M19
REFINES M18
SEES C09

VARIABLES
  run_inst
  suspct_peers
  failr_inst
  dep_instc
  token_owner
  unav_peers
  suspc_inst
  rect_inst
  rctt_inst
  actv_instc
  actv_instc

IN INVARIANTS
  inv1: failr_inst ∈ (PEERS×SERVICES) ⇸ ℙ(PEERS) not theorem
  inv2: ∀ s : s ∈ SERVICES ⇒ token_owner(s) ↦ s ∈ dom(failr_inst) not theorem
  gluing_fail_1: ∀ s : s ∈ SERVICES ⇒ failr_inst(token_owner(s) ↦ s) = failr_peers(s) not theorem

EVENTS
  INITIALISATION: not extended ordinary
  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

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

grd3: \( \forall \text{srv} \cdot \text{srv} \in \text{SERVICES} \Rightarrow \text{dom}(\text{dom}(\text{inst\_state}) \supset \{\text{srv}\}) \)
\( \text{\textbackslash prs} \neq \emptyset \) 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: \( \forall \text{srv} \cdot \text{srv} \in \text{SERVICES} \wedge \text{token\_owner}(\text{srv}) \notin \text{prs} \Rightarrow \text{E}(\text{srv}) = \text{token\_owner}(\text{srv}) \) not theorem \( \rightarrow \) If the token owner of a service srv does not belong to prs, the token owner is not changed

grd6: \( \forall \text{srv} \cdot \text{srv} \in \text{SERVICES} \wedge \text{token\_owner}(\text{srv}) \in \text{prs} \Rightarrow \)
\( \text{E}(\text{srv}) \in \text{run\_inst}(\text{token\_owner}(\text{srv}) \mapsto \text{srv}) \setminus (\text{unav\_peers} \cup \text{prs} \cup \text{failr\_inst}(\text{token\_owner}(\text{srv}) \mapsto \text{srv}) \cup \text{suspct\_peers}(\text{token\_owner}(\text{srv}) \mapsto \text{srv})) \)
\( \wedge \)
\( \text{E}(\text{srv}) \mapsto \text{srv} \in \text{dom}(\text{inst\_state}) \cap \text{dom}(\text{suspct\_peers}) \cap \text{dom}(\text{run\_inst}) \cap \text{dom}(\text{failr\_inst}) \)
\( \Rightarrow \)
\( \text{run\_inst}(\text{E}(\text{srv}) \mapsto \text{srv}) = \text{run\_inst}(\text{token\_owner}(\text{srv}) \mapsto \text{srv}) \)
\( \wedge \)
\( \text{inst\_state}(\text{E}(\text{srv}) \mapsto \text{srv}) = \text{inst\_state}(\text{token\_owner}(\text{srv}) \mapsto \text{srv}) \)
\( \wedge \)
\( \text{suspct\_peers}(\text{E}(\text{srv}) \mapsto \text{srv}) = \text{suspct\_peers}(\text{token\_owner}(\text{srv}) \mapsto \text{srv}) \)
\( \text{E}(\text{srv}) \mapsto \text{srv} \in \text{dom}(\text{inst\_state}) \cap \text{dom}(\text{suspct\_peers}) \cap \text{dom}(\text{run\_inst}) \cap \text{dom}(\text{failr\_inst}) \)

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

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

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

act5: \( \text{actv\_instc} = ((\text{prs} \times \text{SERVICES}) \circ \text{actv\_instc}) \)
\( (((\text{E}\setminus \text{token\_owner})\setminus\emptyset)) \) \( \rightarrow \) the peers in prs can not activate instances anymore

act6: \( \text{suspct\_peers} = (\text{prs} \times \text{SERVICES}) \circ \text{suspct\_peers} \) \( \rightarrow \) the peers
M19

in prs can not suspect instances anymore (1)
act7: suspct_inst = ((prs×SERVICES) ≔ suspct_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 ›
END

SUSPECT_INST: extended ordinary ›
REFINES
SUSPECT_INST
ANY
s  ›a service s
susp  ›suspicous instances
WHERE
grd1: s ∈ SERVICES not theorem ›
grd2: susp ⊆ PEERS not theorem ›
grd3: susp = run_inst(token_owner(s) ↦ s) n unav_peers not theorem › instances in susp are suspicious if the peers running them becomes unavailable
grd4: suspct_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: suspct_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: suspct_inst(token_owner(s) ↦ s) ≠ ∅ not theorem ›
grd5: prop = run_inst(token_owner(s) ↦ s)\(suspc_inst (token_owner(s) ↦ s) v unav_peers) not theorem ›
THEN
act1: inst_state = inst_state « ((prop×{s})×{FAIL_4}) ›
act2: suspct_peers = suspct_peers « ((prop×{s})×{suspc_inst (token_owner(s) ↦ s)})

Page 3
M19

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

END

RECONTACT\_INST\_OK: extended ordinary →

REFINES

RECONTACT\_INST\_OK

ANY

s → a service s
i → an instance i

WHERE

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

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

grd3: \( \text{inst\_state}(\text{token\_owner}(s) \mapsto s) = \text{FAIL\_4} \) not theorem → the state of s is SUSPICIOUS

grd4: \( \text{suspct\_peers}(\text{token\_owner}(s) \mapsto s) \neq \emptyset \) not theorem → the set of suspicious peers for s is not empty

grd5: \( i \in \text{suspct\_peers}(\text{token\_owner}(s) \mapsto s) \setminus \text{unav\_peers} \) not theorem → i is a suspicious instance of s and is available (can be contacted)

grd6: \( i \notin \text{rect\_inst}(\text{token\_owner}(s) \mapsto s) \) not theorem → the token owner of s has not yet tried to recontact i

acting on s

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

act2: \( \text{rctt\_inst}(\text{token\_owner}(s) \mapsto s) = \text{rctt\_inst}(\text{token\_owner}(s) \mapsto s) \cup \{i\} \) → 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: \( s \in \text{SERVICES} \) not theorem →

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

grd3: \( \text{inst\_state}(\text{token\_owner}(s) \mapsto s) = \text{FAIL\_4} \) not theorem → the state of s is SUSPICIOUS

grd4: \( \text{suspct\_peers}(\text{token\_owner}(s) \mapsto s) \neq \emptyset \) not theorem → the set of suspicious peers for s is not empty

grd5: \( i \in \text{suspct\_peers}(\text{token\_owner}(s) \mapsto s) \cap \text{unav\_peers} \) not theorem → i is a suspicious instance of s and is unavailable (can not be contacted)

grd6: \( i \notin \text{rect\_inst}(\text{token\_owner}(s) \mapsto s) \) not theorem → the token owner of s has not yet tried to recontact i
M19

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

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

\[\text{END}\n\]

\[
\text{FAIL\_DETECT: extended ordinary} \quad \text{REFINES} \quad \text{FAIL\_DETECT}
\]

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

\[\text{WHERE}\]

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

\[
grd2: \quad \text{prop} \in\text{PEERS} \quad \not\text{ theorem} \quad \not\text{ theorem}\n\]

\[
grd7: \quad \text{susp} \in\text{PEERS} \quad \not\text{ theorem} \quad \not\text{ theorem}\n\]

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

\[
grd4: \quad \text{susp\_peers(token\_owner(s) \mapsto s) \neq \emptyset} \quad \not\text{ theorem} \quad \not\text{ theorem}\n\]

\[
grd5: \quad \text{rect\_inst(token\_owner(s) \mapsto s) = susp\_peers(token\_owner(s) \mapsto s)} \quad \not\text{ theorem} \quad \not\text{ theorem}\n\]

\[
\text{THEN} \quad \text{act1: inst\_state = inst\_state \setminus ((prop \times \{s\}) \times \{\text{FAIL\_DETECT\_4}\})}
\]

\[
\quad \text{act2: susp\_peers = susp\_peers \setminus ((prop \times \{s\}) \times \{\text{susp}\})}
\]

\[
\quad \text{act3: rect\_inst(token\_owner(s) \mapsto s) = \emptyset}
\]

\[
\quad \text{act4: rctt\_inst(token\_owner(s) \mapsto s) = \emptyset}
\]

\[\text{END}\n\]

\[
\text{IS\_OK: extended ordinary} \quad \text{REFINES} \quad \text{IS\_OK}
\]

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

\[\text{WHERE}\]

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

\[
grd2: \quad \text{prop} \in\text{PEERS} \quad \not\text{ theorem} \quad \not\text{ theorem}\n\]

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

\[
grd4: \quad \text{susp\_peers(token\_owner(s) \mapsto s) = \emptyset} \quad \not\text{ theorem} \quad \not\text{ theorem}\n\]

\[
grd5: \quad \text{prop = run\_inst(token\_owner(s) \mapsto s) \setminus unav\_peers} \quad \not\text{ theorem} \quad \not\text{ theorem}\n\]
M19

THEOREM

THEN

act1: \( \text{inst\_state} = \text{inst\_state} \oplus ((\text{prop} \times \{s\}) \times \{\text{RUN}_4\}) \)

END

FAIL\_ACTIV: not extended ordinary

REFINES

FAIL\_ACTIV

ANY

s >

prop >

WHERE

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

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

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

THEOREM

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

grd5: \( \text{prop} = \text{run\_inst}(\text{token\_owner}(s) \mapsto s) \setminus (\text{unav\_peers} \cup \text{suspct\_peers}(\text{token\_owner}(s) \mapsto s)) \) not theorem

THEN

act1: \( \text{inst\_state} = \text{inst\_state} \oplus ((\text{prop} \times \{s\}) \times \{\text{FAIL\_ACTIV}_4\}) \)

act2: \( \text{run\_inst} = \text{run\_inst} \oplus ((\text{prop} \times \{s\}) \times \{\text{run\_inst}(\text{token\_owner}(s) \mapsto s) \setminus \text{suspct\_peers}(\text{token\_owner}(s) \mapsto s)\}) \)

act3: \( \text{failr\_inst} = \text{failr\_inst} \oplus ((\text{prop} \times \{s\}) \times \{\text{failr\_inst}(\text{token\_owner}(s) \mapsto s) \cup \text{suspct\_peers}(\text{token\_owner}(s) \mapsto s)\}) \)

act4: \( \text{suspct\_peers} = \text{suspct\_peers} \oplus ((\text{prop} \times \{s\}) \times \{\emptyset\}) \)

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

THEOREM

grd4: \( \text{card}(\text{run\_inst}(\text{token\_owner}(s) \mapsto s)) < \text{min\_inst}(s) \) not theorem

grd5: \( \text{prop} = \text{run\_inst}(\text{token\_owner}(s) \mapsto s) \setminus \text{unav\_peers} \) not theorem

THEN

act1: \( \text{inst\_state} = \text{inst\_state} \oplus ((\text{prop} \times \{s\}) \times \{\text{FAIL\_CONF}\}) \)

END

FAIL\_IGNORE: extended ordinary
REFINES  
    FAIL_IGNORE
ANY
    s  
    prop  
WHERE
    grd1:  s ∈ SERVICES not theorem 
    grd2:  prop ⊆ PEERS not theorem 
    grd3:  inst_state(token_owner(s) ↦ s) = FAIL_ACTIV_4 not theorem 
    grd4:  card(run_inst(token_owner(s) ↦ s)) ≥ min_inst(s) not theorem 
    grd5:  prop = run_inst(token_owner(s) ↦ s) \ unav_peers not theorem 
THEN
    act1:  inst_state ≔ inst_state  \ ((prop×{s})×{FAIL_IGN_4}) 
END

IGNORE: extended ordinary

REFINES  
    IGNORE
ANY
    s  
    prop  
WHERE
    grd1:  s ∈ SERVICES not theorem 
    grd2:  prop ⊆ PEERS not theorem 
    grd3:  inst_state(token_owner(s) ↦ s) = FAIL_IGN_4 not theorem 
    grd4:  prop = run_inst(token_owner(s) ↦ s) \ unav_peers not theorem 
THEN
    act1:  inst_state ≔ inst_state  \ ((prop×{s})×{RUN_4}) 
END

REDEPLOY_INSTC: not extended ordinary

REFINES  
    REDEPLOY_INSTC
ANY
    s  ↦ a service s 
    i  ↦ an instance i
WHERE
    grd1:  s ∈ SERVICES not theorem 
    grd2:  i ∈ PEERS not theorem 
    grd3:  i ∈ run_inst(token_owner(s) ↦ s) ∪ failr_inst (token_owner(s) ↦ s) ∪ unav_peers ∪ dep_instc(s) not theorem 
i does not run s, is not failed for s, is not unavailable and is not already activated for s 
    grd4:  i ∉ actv_instc(token_owner(s) ↦ s) not theorem 

M19

\[ grd5: \quad \text{inst\_state}(\text{token\_owner}(s) \rightarrow s) = \text{FAIL\_CONFIG\_4} \quad \text{not theorem} \]

\[ grd6: \quad \text{card}(\text{actv\_instc}(\text{token\_owner}(s) \rightarrow s)) < \text{deplo\_inst}(s) \quad \text{not theorem} \]

\[ grd7: \quad \text{card}(\text{dep\_instc}(s)) + \text{card}(\text{run\_inst}(\text{token\_owner}(s) \rightarrow s)) < \text{min\_inst}(s) \quad \text{not theorem} \]

THEN

\[ \begin{align*}
   \text{act1:} & \quad \text{actv\_instc}(\text{token\_owner}(s) \rightarrow s) = \text{actv\_instc}(\text{token\_owner}(s) \rightarrow s) \cup \{i\} \\
   \text{END} & \\
\end{align*} \]

\text{REDEPLOY\_INSTS:} \quad \text{extended ordinary} \quad \text{REFINES} \quad \text{REDEPLOY\_INSTS} \\
\text{ANY} \quad s \quad \text{WHERE} \\
\quad \begin{align*}
   \text{grd1:} & \quad s \in \text{SERVICES} \quad \text{not theorem} \\
   \text{grd2:} & \quad \text{card}(\text{actv\_instc}(\text{token\_owner}(s) \rightarrow s)) = \text{deplo\_inst}(s) \quad \text{not theorem} \\
   \text{grd3:} & \quad \text{card}(\text{dep\_instc}(s)) + \text{card}(\text{run\_inst}(\text{token\_owner}(s) \rightarrow s)) < \text{min\_inst}(s) \quad \text{not theorem} \\
   \text{grd4:} & \quad \text{inst\_state}(\text{token\_owner}(s) \rightarrow s) = \text{FAIL\_CONFIG\_4} \quad \text{not theorem} \\
\end{align*} \]

THEN

\[ \begin{align*}
   \text{act1:} & \quad \text{dep\_instc}(s) = \text{dep\_instc}(s) \cup \text{actv\_instc}(\text{token\_owner}(s) \rightarrow s) \\
   \text{act2:} & \quad \text{actv\_instc}(\text{token\_owner}(s) \rightarrow s) = \emptyset \\
   \text{END} & \\
\end{align*} \]

\text{REDEPLOY:} \quad \text{extended ordinary} \quad \text{REFINES} \quad \text{REDEPLOY} \\
\text{ANY} \quad s \quad \text{WHERE} \\
\quad \begin{align*}
   \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\_CONFIG\_4} \quad \text{not theorem} \\
   \text{grd4:} & \quad \text{actv\_instc}(\text{token\_owner}(s) \rightarrow s) = \emptyset \quad \text{not theorem} \\
   \text{grd5:} & \quad \text{dep\_instc}(s) \neq \emptyset \quad \text{not theorem} \\
   \text{grd6:} & \quad \text{card}(\text{run\_inst}(\text{token\_owner}(s) \rightarrow s)) + \text{card}(\text{dep\_instc}(s)) \geq \text{min\_inst}(s) \quad \text{not theorem} \\
   \text{grd7:} & \quad \text{prop} = \text{run\_inst}(\text{token\_owner}(s) \rightarrow s) \setminus \text{unav\_peers} \quad \text{not theorem} \\
\end{align*} \]

THEN
act1: \begin{align*} \text{inst} = \text{inst} \triangleleft \{(\text{prop} \times \text{s}) \times \text{DPL}_4\} \end{align*} \\
act2: \begin{align*} \text{run} = \text{run} \triangleleft \{(\text{prop} \times \text{s}) \times \{\text{run}(\text{token} \rightarrow \text{s})\} \end{align*} \\
act3: \begin{align*} \text{dep} = \text{dep} = \emptyset \end{align*} 

**END**

**HEAL**: extended ordinary 

**REFINES**

**HEAL**

**ANY**

\begin{align*} & \text{s} \triangleright \\
& \text{prop} \triangleright \\
\end{align*}

**WHERE**

\begin{align*} & \text{grd1: } \text{s} \in \text{SERVICES} \not \text{theorem} \triangleright \\
& \text{grd2: } \text{prop} \subseteq \text{PEERS} \not \text{theorem} \triangleright \\
& \text{grd3: } \text{inst}(\text{token} \rightarrow \text{s}) = \text{DPL}_4 \not \text{theorem} \triangleright \\
& \text{grd4: } \text{prop} = \text{run}(\text{token} \rightarrow \text{s}) \setminus \text{unav} \not \text{theorem} \triangleright \\
\end{align*}

**theorem**

**THEN**

\begin{align*} & \text{act1: } \text{inst} = \text{inst} \triangleleft \{(\text{prop} \times \text{s}) \times \text{RUN}_4\} \end{align*}

**END**

**UNFAIL_PEER**: not extended ordinary 

**REFINES**

**UNFAIL_PEER**

**ANY**

\begin{align*} & \text{s} \triangleright \\
& \text{p} \triangleright \\
& \text{prop} \triangleright \\
\end{align*}

**WHERE**

\begin{align*} & \text{grd1: } \text{s} \in \text{SERVICES} \not \text{theorem} \triangleright \\
& \text{grd2: } \text{prop} \subseteq \text{PEERS} \not \text{theorem} \triangleright \\
& \text{grd3: } \text{p} \in \text{PEERS} \not \text{theorem} \triangleright \\
& \text{grd4: } \text{p} \in \text{failr}(\text{token} \rightarrow \text{s}) \not \text{theorem} \triangleright \\
& \text{grd5: } \text{prop} = \text{run}(\text{token} \rightarrow \text{s}) \setminus \text{unav} \not \text{theorem} \triangleright \\
\end{align*}

**THEOREM**

**THEN**

\begin{align*} & \text{act1: } \text{failr} = \text{failr} \triangleleft \{(\text{prop} \times \text{s}) \times \text{failr} \setminus \{\text{p}\}\} \end{align*}

**END**

**MAKE_PEER_AVAIL**: extended ordinary 

**REFINES**

**MAKE_PEER_AVAIL**

**ANY**

\begin{align*} & \text{p} \triangleright \\
\end{align*}

**WHERE**

\begin{align*} & \text{grd1: } \text{p} \in \text{PEERS} \not \text{theorem} \triangleright \\
\end{align*}
M19

grd2:  p ∈ unav_peers not theorem
THEN
act1:  unav_peers = unav_peers \ {p}
END

END
M20

MACHINE M20
REFINES M19
SEES C09

VARIABLES
  run_inst
  suspct_peers
  failr_inst
  dep_instcs
  token_owner
  unav_peers
  suspc_inst
  rctt_inst
  inst_state
  actv_instc

INVARIANTS
  inv1: dep_instcs ∈ (PEERS×SERVICES) ⇸ ℙ(PEERS) not theorem
  inv2: ∀ s · s ∈ SERVICES ⇒ token_owner(s) ↦ s ∈ dom(dep_instcs) not theorem
  gluing_act_1: ∀ s · s ∈ SERVICES ⇒ dep_instcs(token_owner(s) ↦ s) = dep_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

Page 1
\[ \text{grd2: } \text{prs} \not\in \text{unav_peers} \not\text{ theorem } \rightarrow \text{the peers in prs are not yet unavailaible} \]

\[ \text{grd3: } \forall \text{srv} \cdot \text{srv} \in \text{SERVICES} \Rightarrow \text{dom(dom(inst_state)} \supseteq \{\text{srv}\} \]
\[ \not\text{ theorem } \rightarrow \text{for each service srv, there must always be at least 1 peer available} \]

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

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

\[ \text{grd6: } \forall \text{srv} \cdot \text{srv} \in \text{SERVICES} \land \text{token_owner(srv)} \in \text{prs} \Rightarrow \]
\[ E(srv) \in \text{run_inst(token_owner(srv)} \rightarrow \text{srv}) \cup \{\text{unav_peers} \cup \text{prs} \cup \text{failr_inst(token_owner(srv)} \rightarrow \text{srv}) \cup \text{suspct_peers(token_owner(srv)} \rightarrow \text{srv})\} \]
\[ \rightarrow E(srv) \rightarrow \text{srv} \in \text{dom(inst_state)} \cap \text{dom(failr_inst)} \cap \text{dom(suspct_peers)} \cap \text{dom(dep_instcs)} \]
\[ \rightarrow \text{run_inst}(E(srv) \rightarrow \text{srv}) = \text{run_inst(token_owner(srv)} \rightarrow \text{srv}) \]
\[ \rightarrow \text{inst_state}(E(srv) \rightarrow \text{srv}) = \text{inst_state(token_owner(srv)} \rightarrow \text{srv}) \]
\[ \rightarrow \text{suspct_peers}(E(srv) \rightarrow \text{srv}) = \text{suspct_peers(token_owner(srv)} \rightarrow \text{srv}) \]
\[ \rightarrow \text{failr_inst}(E(srv) \rightarrow \text{srv}) = \text{failr_inst(token_owner(srv)} \rightarrow \text{srv}) \]
\[ \rightarrow \text{dep_instcs}(E(srv) \rightarrow \text{srv}) = \text{dep_instcs(token_owner(srv)} \rightarrow \text{srv}) \]

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

\[ \text{act1: } \text{unav_peers} \equiv \text{unav_peers} \cup \text{prs} \rightarrow \text{the peers in prs become unavailable} \]
\[ \text{act2: } \text{token_owner} \equiv \text{token_owner} \equiv E \rightarrow \text{new values for token owner per service} \]
\[ \text{act3: } \text{rect_inst} = (((\text{prs} \times \text{SERVICES}) \leftarrow \text{rect_inst}) \leftarrow \]
\[ (((\text{E} \setminus \text{token_owner}) \times \{\emptyset\}) \rightarrow \text{the peers in prs can not try to recontact instances anymore (1)} \]
\[ \text{act4: } \text{rctt_inst} = (((\text{prs} \times \text{SERVICES}) \leftarrow \text{rctt_inst}) \leftarrow \]
\[ (((\text{E} \setminus \text{token_owner}) \times \{\emptyset\}) \rightarrow \text{the peers in prs can not try to recontact instances anymore (2)} \]
\[ \text{act5: } \text{actv_instc} = (((\text{prs} \times \text{SERVICES}) \leftarrow \text{actv_instc}) \leftarrow \]

Page 2
act5: \[ \text{actv}_{\text{instc}} = ((\text{prs} \times \text{SERVICES}) \downarrow \text{actv}_{\text{instc}}) \not\in ((E \backslash \text{token\_owner}) \backsim) \times \{\emptyset\} \]

\[ \text{the peers in prs can not activate instances anymore (1)} \]

act6: \[ \text{suspc}_{\text{peers}} = (\text{prs} \times \text{SERVICES}) \downarrow \text{suspc}_{\text{peers}} \]

\[ \text{the peers in prs can not suspect instances anymore (1)} \]

act7: \[ \text{suspc}_{\text{inst}} = ((\text{prs} \times \text{SERVICES}) \downarrow \text{suspc}_{\text{inst}}) \not\in ((E \backslash \text{token\_owner}) \backsim) \times \{\emptyset\} \]

\[ \text{the peers in prs can not suspect instances anymore (2)} \]

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

prs can not monitor the state of the services provided anymore

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

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

act11: \[ \text{dep\_instcs} = (\text{prs} \times \text{SERVICES}) \downarrow \text{dep\_instcs} \]

END

\[ \text{SUSPECT\_INST: extended ordinary} \]

\[ \text{REFINES} \]

\[ \text{SUSPECT\_INST} \]

\[ \text{ANY} \]

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

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

\[ \text{WHERE} \]

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

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

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

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

\[ \text{grd4: } \text{suspc}_{\text{inst}}(\text{token\_owner}(s) \mapsto s) = \emptyset \text{ not theorem} \]

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

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

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

\[ \text{grd6: } \text{susp} \neq \emptyset \text{ not theorem} \]

\[ \text{THEN} \]

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

END

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

\[ \text{REFINES} \]

\[ \text{FAIL} \]

\[ \text{ANY} \]

\[ s \to \]

\[ \text{prop} \to \]

\[ \text{WHERE} \]

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

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

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

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

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

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

\textit{RECONTACT\_INST\_OK}: \quad \text{extended ordinary} \triangleright \\
\textit{REFINES} \quad \text{RECONTACT\_INST\_OK} \quad \text{ANY} \\
\quad s \quad \rightarrow \text{a service s} \\
\quad i \quad \rightarrow \text{an instance i} \\
\text{WHERE} \\
\quad \text{ grd1}: \quad s \in \text{SERVICES} \not\equiv \text{theorem} \triangleright \text{the state of s is SUSPICIOUS} \\
\quad \text{ grd4}: \quad \text{suspct\_peers}(\text{token\_owner}(s) \mapsto s) \neq \emptyset \not\equiv \text{theorem} \triangleright \text{the set of suspicious peers for s is not empty} \\
\quad \text{ grd5}: \quad i \in \text{suspct\_peers}(\text{token\_owner}(s) \mapsto s) \setminus \text{unav\_peers} \not\equiv \text{theorem} \triangleright \text{i is a suspicious instance of s and is available (can be contacted)} \\
\text{ THEN} \\
\quad \text{ act1}: \quad \text{rect\_inst}(\text{token\_owner}(s) \mapsto s) \equiv \text{rect\_inst}(\text{token\_owner}(s) \mapsto s) \cup \{i\} \triangleright \text{the token owner of s has tried to recontact i} \\
\quad \text{ act2}: \quad \text{rctt\_inst}(\text{token\_owner}(s) \mapsto s) \equiv \text{rctt\_inst}(\text{token\_owner}(s) \mapsto s) \cup \{i\} \triangleright \text{i is recontacted by the token owner of s successfully} \\
\text{END}

\textit{RECONTACT\_INST\_K0}: \quad \text{extended ordinary} \triangleright \\
\textit{REFINES} \quad \text{RECONTACT\_INST\_K0} \quad \text{ANY} \\
\quad s \quad \rightarrow \text{a service s} \\
\quad i \quad \rightarrow \text{an instance i} \\
\text{WHERE} \\
\quad \text{ grd1}: \quad s \in \text{SERVICES} \not\equiv \text{theorem} \triangleright \text{the state of s is SUSPICIOUS} \\
\quad \text{ grd4}: \quad \text{suspct\_peers}(\text{token\_owner}(s) \mapsto s) \neq \emptyset \not\equiv \text{theorem} \triangleright \text{the set of suspicious peers for s is not empty} \\
\quad \text{ grd5}: \quad i \in \text{suspct\_peers}(\text{token\_owner}(s) \mapsto s) \cap \text{unav\_peers} \not\equiv \text{theorem} \triangleright \text{i is a suspicious instance of s and is unavailable (can not be contacted)}
contacted)

\[\text{grd6: } i \notin \text{rect_inst}(\text{token_owner}(s) \mapsto s) \quad \text{not theorem} \Rightarrow \text{the token owner of s has not yet tried to recontact i}\]

\[\text{grd7: } \text{rect_inst}(\text{token_owner}(s) \mapsto s) \subseteq \text{suspct_peers}(\text{token_owner}(s) \mapsto s) \quad \text{not theorem} \Rightarrow \text{the token owner of s has not yet tried to recontact all the suspicious instances of s}\]

**THEN**

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

**FAIL_DETECT:** extended ordinary

**REFINES**

*FAIL_DETECT*

**ANY**

\[s \]  
prop  
 susp

**WHERE**

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

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

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

\[\text{grd3: } \text{inst_state}(\text{token_owner}(s) \mapsto s) = \text{FAIL_4} \quad \text{not theorem}\]

\[\text{grd4: } \text{suspct_peers}(\text{token_owner}(s) \mapsto s) \neq \emptyset \quad \text{not theorem}\]

\[\text{grd5: } \text{rect_inst}(\text{token_owner}(s) \mapsto s) = \text{suspct_peers}(\text{token_owner}(s) \mapsto s) \quad \text{not theorem}\]

\[\text{grd6: } \text{prop} = ((\text{run_inst}(\text{token_owner}(s) \mapsto s) \setminus \text{suspct_peers}(\text{token_owner}(s) \mapsto s)) \cup \text{rctt_inst}(\text{token_owner}(s) \mapsto s) \setminus \text{unav_peers} \quad \text{not theorem}\]

**THEN**

\[\text{act1: } \text{inst_state} = \text{inst_state} \circ ((\text{prop} \times \{s\}) \times \{\text{FAIL_DETECT}_4\})\]

\[\text{act2: } \text{suspct_peers} = \text{suspct_peers} \circ ((\text{prop} \times \{s\}) \times \{\text{susp}\})\]

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

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

**END**

**IS_OK:** extended ordinary

**REFINES**

*IS_OK*

**ANY**

\[s \]  
 prop

**WHERE**

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

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

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

Page 5
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})×{RUN_4}) >
act2: run_inst = run_inst « ((prop×{s})×{run_inst(token_owner(s) ↦ s) ↦ s)\suspct_peers(token_owner(s) ↦ s)\suspect_peers(token_owner(s) ↦ s)}) »
act3: failr_inst = failr_inst « ((prop×{s})×{failr_inst(token_owner(s) ↦ s) u suspct_peers(token_owner(s) ↦ s)}) »
act4: suspct_peers = suspct_peers « ((prop×{s})×{∅}) »
END

FAIL_CONFIGURE: extended ordinary >
REFINES
    FAIL_CONFIGURE

ANY
    s >
    prop >
WHERE
    grd1: s ∈ SERVICES not theorem >
    grd2: prop ⊆ PEERS not theorem >
    grd3: inst_state(token_owner(s) ↦ s) = FAIL_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) ↦ s)\suspct_peers(token_owner(s) ↦ s)\suspct_peers(token_owner(s) ↦ s)}) »
act3: failr_inst = failr_inst « ((prop×{s})×{failr_inst(token_owner(s) ↦ s) u suspct_peers(token_owner(s) ↦ s)}) »
act4: suspct_peers = suspct_peers « ((prop×{s})×{∅}) »
END
FAIL_IGNORE: extended ordinary

REFINES
FAIL_IGNORE

ANY
s
prop

WHERE
grd1: s ∈ SERVICES not theorem
grd2: prop ⊆ PEERS not theorem
grd3: \(\text{inst}_\text{state}(\text{token}_{\text{owner}}(s) \mapsto s) = \text{FAIL}_\text{ACTIV}_4\) not theorem
grd4: card(\(\text{run}_\text{inst}(\text{token}_{\text{owner}}(s) \mapsto s)\)) ≥ min\_inst(s) not theorem
grd5: \(\text{prop} = \text{run}_\text{inst}(\text{token}_{\text{owner}}(s) \mapsto s)\setminus \text{unav}\_\text{peers}\) not theorem

THEN
act1: \(\text{inst}_\text{state} = \text{inst}_\text{state} \odot ((\text{prop} \times \{s\}) \times \{\text{FAIL}_\text{IGN}_4\})\)

END

IGNORE: extended ordinary

REFINES
IGNORE

ANY
s
prop

WHERE
grd1: s ∈ SERVICES not theorem
grd2: prop ⊆ PEERS not theorem
grd3: \(\text{inst}_\text{state}(\text{token}_{\text{owner}}(s) \mapsto s) = \text{FAIL}_\text{IGN}_4\) not theorem
grd4: \(\text{prop} = \text{run}_\text{inst}(\text{token}_{\text{owner}}(s) \mapsto s)\setminus \text{unav}\_\text{peers}\) not theorem

THEN
act1: \(\text{inst}_\text{state} = \text{inst}_\text{state} \odot ((\text{prop} \times \{s\}) \times \{\text{RUN}_\text{4}\})\)

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
\((\text{token\_owner}(s) \Rightarrow s) \cup \text{unav\_peers} \cup \text{dep\_instcs}(\text{token\_owner}(s) \Rightarrow s)\) is not theorem

> i does not run s, is not failed for s, is not unavailable and is not already activated for s

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

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

\begin{align*}
\text{THEN} & \quad \text{act1:} \quad \text{actv\_instc}(\text{token\_owner}(s) \Rightarrow s) = \text{actv\_instc}(\text{token\_owner}(s) \Rightarrow s) \cup \{i\}
\end{align*}

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

\text{ANY} \\
\text{s} \quad \text{prop}

\text{WHERE}

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

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

\begin{align*}
\text{THEN} & \quad \text{act1:} \quad \text{dep\_instcs} = \text{dep\_instcs} \cup ((\text{prop}\times\{i\}) \times \{\text{dep\_instcs} (\text{token\_owner}(s)\Rightarrow s) \cup \text{actv\_instc}(\text{token\_owner}(s)\Rightarrow s)\}) \\
\text{act2:} & \quad \text{actv\_instc}(\text{token\_owner}(s) \Rightarrow s) = \emptyset
\end{align*}

\text{END}

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

\text{ANY} \\
\text{s} \quad \text{prop}

\text{WHERE}

\begin{align*}
\text{grd1:} & \quad s \in \text{SERVICES} \text{ not theorem} \\
\text{grd2:} & \quad \text{prop} \subseteq \text{PEERS} \text{ not theorem} \\
\text{grd3:} & \quad \text{inst\_state}(\text{token\_owner}(s) \Rightarrow s) = \text{FAIL\_CONFIG\_4} \text{ not theorem}
\end{align*}
M20

\[ \text{theorem} \]

\begin{align*}
\text{grd4: } & \quad \text{actv\_instc(token\_owner(s) \mapsto s)} = \emptyset \quad \text{not theorem} \\
\text{grd5: } & \quad \text{dep\_instcs(token\_owner(s) \mapsto s)} \neq \emptyset \quad \text{not theorem} \\
\text{grd6: } & \quad \text{card(run\_inst(token\_owner(s) \mapsto s))} + \text{card(dep\_instcs (token\_owner(s) \mapsto s))} \\
& \quad \geq \min\_inst(s) \quad \text{not theorem} \\
\text{grd7: } & \quad \text{prop} = \text{run\_inst(token\_owner(s) \mapsto s)} \setminus \text{unav\_peers} \quad \text{not theorem}
\end{align*}

\[ \text{THEN} \]

\begin{align*}
\text{act1: } & \quad \text{inst\_state} \coloneqq \text{inst\_state} \upharpoonright ((\text{prop} \times \{s\}) \times \{\text{DPL}\_4\}) \\
\text{act2: } & \quad \text{run\_inst} = \text{run\_inst} \upharpoonright ((\text{prop} \times \{s\}) \times \{\text{run\_inst(token\_owner} \\\n& \quad (s \mapsto s) \cup \text{dep\_instcs(token\_owner(s) \mapsto s))}\}) \\
\text{act3: } & \quad \text{dep\_instcs} = \text{dep\_instcs} \upharpoonright ((\text{prop} \times \{s\}) \times \{\emptyset\})
\end{align*}

\[ \text{END} \]

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

\[ \text{REFINES} \]

\[ \text{HEAL} \]

\[ \text{ANY} \]

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

\[ \text{WHERE} \]

\begin{align*}
\text{grd1: } & \quad s \in \text{SERVICES} \quad \text{not theorem} \\
\text{grd2: } & \quad \text{prop} \in \text{PEERS} \quad \text{not theorem} \\
\text{grd3: } & \quad \text{inst\_state(token\_owner(s) \mapsto s)} = \text{DPL}\_4 \quad \text{not theorem} \\
\text{grd4: } & \quad \text{prop} = \text{run\_inst(token\_owner(s) \mapsto s)} \setminus \text{unav\_peers} \quad \text{not theorem}
\end{align*}

\[ \text{theorem} \]

\[ \text{THEN} \]

\begin{align*}
\text{act1: } & \quad \text{inst\_state} = \text{inst\_state} \upharpoonright ((\text{prop} \times \{s\}) \times \{\text{RUN}\_4\})
\end{align*}

\[ \text{END} \]

\[ \text{UNFAIL\_PEER: extended ordinary} \]

\[ \text{REFINES} \]

\[ \text{UNFAIL\_PEER} \]

\[ \text{ANY} \]

\[ s \quad p \quad \text{prop} \]

\[ \text{WHERE} \]

\begin{align*}
\text{grd1: } & \quad s \in \text{SERVICES} \quad \text{not theorem} \\
\text{grd2: } & \quad \text{prop} \in \text{PEERS} \quad \text{not theorem} \\
\text{grd3: } & \quad p \in \text{PEERS} \quad \text{not theorem} \\
\text{grd4: } & \quad p \in \text{failr\_inst(token\_owner(s) \mapsto s)} \quad \text{not theorem} \\
\text{grd5: } & \quad \text{prop} = \text{run\_inst(token\_owner(s) \mapsto s)} \setminus \text{unav\_peers} \quad \text{not theorem}
\end{align*}

\[ \text{theorem} \]

\[ \text{THEN} \]

\begin{align*}
\text{act1: } & \quad \text{failr\_inst} = \text{failr\_inst} \upharpoonright ((\text{prop} \times \{s\}) \times \{\text{failr\_inst (token\_owner(s) \mapsto s) \setminus \{p\}}\})
\end{align*}

\[ \text{END} \]
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
M21

MACHINE M21 →
REFINES M20
SEES C09
VARIABLES
  run_inst
  suspct_peers
  failr_inst
  dep_instcs
  token_owner
  unav_peers
  suspc_inst
  rct_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: 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 = ø →
    act7: suspc_inst = InitSuspPeers →
    act8: rct_inst = InitSuspPeers →
    act9: rctt_inst = InitSuspPeers →
    act10: actv_instc = InitSuspPeers →
    act11: inst_state = InitStateSrv →

END

MAKE_PEER_UNAVAIL: not extended ordinary →
REFINES MAKE_PEER_UNAVAIL
ANY
  prs → Peers that will become unavailable
  E → Values for token owner per service
WHERE
  grd1: prs ∈ PEERS not theorem →
  grd2: prs ∉ unav_peers not theorem → the peers in prs are not yet unavailable
  grd3: ∀ srv · srv ∈ SERVICES ⇒ dom(dom(inst_state) ∈ {sr})
  \prs ≠ ø not theorem → for each service srv, there must always be at least 1
peer available

\[ \forall \text{srv} \cdot \text{srv} \in \text{SERVICES} \Rightarrow \text{dom} (\text{inst}_{\text{state}}) \cap \{\text{srv}\} \neq \emptyset \]

for each service srv, there must always be at least 1 peer available

\[ E \in \text{SERVICES} \rightarrow \text{PEERS} \]

not theorem

\[ \text{Value for token owner per service} \]

\[ \forall \text{srv} \cdot \text{srv} \in \text{SERVICES} \land \text{token}_{\text{owner}}(\text{srv}) \notin \text{prs} \Rightarrow E(\text{srv}) = \text{token}_{\text{owner}}(\text{srv}) \]

not theorem

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

\[ \forall \text{srv} \cdot \text{srv} \in \text{SERVICES} \land \text{token}_{\text{owner}}(\text{srv}) \in \text{prs} \]

not theorem

\[ E(\text{srv}) = \text{run}_{\text{inst}}(\text{token}_{\text{owner}}(\text{srv}) \Rightarrow \text{srv}) \cap \left( \text{unav}_{\text{peers}} \cup \text{failr}_{\text{inst}}(\text{token}_{\text{owner}}(\text{srv}) \Rightarrow \text{srv}) \cup \text{suspc}_{\text{peers}}(\text{token}_{\text{owner}}(\text{srv}) \Rightarrow \text{srv}) \right) \]

not theorem

\[ \text{if the owner of the token for a service becomes unavailable,} \]

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

\[ \text{THEN} \]

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

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

\[ \text{act3: rect}_{\text{inst}} = ((\text{prs} \times \text{SERVICES}) \setminus \text{rect}_{\text{inst}}) \Rightarrow (\{(\text{E} \setminus \text{token}_{\text{owner}}) \times \emptyset\}) \Rightarrow \text{the peers in prs can not try to recontact instances anymore (1)} \]

\[ \text{act4: rctt}_{\text{inst}} = ((\text{prs} \times \text{SERVICES}) \setminus \text{rctt}_{\text{inst}}) \Rightarrow (\{(\text{E} \setminus \text{token}_{\text{owner}}) \times \emptyset\}) \Rightarrow \text{the peers in prs can not try to recontact instances anymore (2)} \]

\[ \text{act5: actv}_{\text{instc}} = ((\text{prs} \times \text{SERVICES}) \setminus \text{actv}_{\text{instc}}) \Rightarrow (\{(\text{E} \setminus \text{token}_{\text{owner}}) \times \emptyset\}) \Rightarrow \text{the peers in prs can not activate instances anymore (1)} \]

\[ \text{act6: suspc}_{\text{peers}} = (\text{prs} \times \text{SERVICES}) \setminus \text{suspc}_{\text{peers}} \Rightarrow \text{the peers in prs can not suspect instances anymore (2)} \]

\[ \text{act7: suspc}_{\text{inst}} = ((\text{prs} \times \text{SERVICES}) \setminus \text{suspc}_{\text{inst}}) \Rightarrow \]
(((E\token_owner-)×{∅})) → the peers in prs can not suspect instances anymore (2)

act8: \text{inst\_state} = (prs×SERVICES) \triangleleft \text{inst\_state} → the peers in prs can not monitor the state of the services provided anymore

act9: \text{run\_inst} = (prs×SERVICES) \triangleleft \text{run\_inst} →

act10: \text{failr\_inst} = (prs×SERVICES) \triangleleft \text{failr\_inst} →

act11: \text{dep\_instcs} = (prs×SERVICES) \triangleleft \text{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) \cap 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: \text{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: \text{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) \setminus (suspc\_inst (token\_owner(s) ↦ s) \cup unav\_peers) not theorem →

THEN

act1: \text{inst\_state} = \text{inst\_state} \triangleleft ((prop×{s})×{FAIL\_4}) →

act2: suspct\_peers = suspct\_peers \triangleleft ((prop×{s})×{suspc\_inst (token\_owner(s) ↦ s)}) →

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

Page 3
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
grd7: rect_inst(token_owner(s) ↦ s) ⊂ suspct_peers(token_owner
(s) ⇒ s) not theorem ⇒ the token owner of s has not yet tried to recontact all the suspicious instances of s

THEN

act1: rect_inst(token_owner(s) ⇒ s) = rect_inst(token_owner(s) ⇒ s) ∪ \{i\} ⇒ the token owner of s has tried to recontact i

END

FAIL_DETECT: extended ordinary

REFINES

FAIL_DETECT

ANY

s

prop

susp

WHERE

grd1: s ∈ SERVICES not theorem

grd2: prop ⊆ PEERS not theorem

grd7: susp ⊆ PEERS not theorem

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

grd4: suspct_peers(token_owner(s) ⇒ s) ≠ ∅ not theorem

grd5: rect_inst(token_owner(s) ⇒ s) = suspct_peers(token_owner(s) ⇒ s) not theorem

(s) ⇒ s) not theorem

grd6: prop = ((run_inst(token_owner(s) ⇒ s) \ suspct_peers(token_owner(s) ⇒ s)) ∪ rctt_inst(token_owner(s) ⇒ s)) \ unav_peers not theorem

grd8: susp = suspct_peers(token_owner(s) ⇒ s) \ rctt_inst(token_owner(s) ⇒ s) not theorem

THEN

act1: inst_state = inst_state \ ((prop × \{s\}) × {FAIL_DETECT_4})

act2: suspct_peers = suspct_peers \ ((prop × \{s\}) × {susp})

act3: rect_inst(token_owner(s) ⇒ s) = ∅

act4: rctt_inst(token_owner(s) ⇒ s) = ∅

END

IS_OK: extended ordinary

REFINES

IS_OK

ANY

s

prop

WHERE

grd1: s ∈ SERVICES not theorem

grd2: prop ⊆ PEERS not theorem

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

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

grd5: prop = run_inst(token_owner(s) ⇒ s) \ unav_peers not theorem

Page 5
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>
    grd4:  suspct_peers(token_owner(s) ↦ s) ≠ ∅ not theorem>
    grd5:  prop = run_inst(token_owner(s) ↦ s) \ (unav_peers ∪ suspct_peers(token_owner(s) ↦ s)) not theorem>
  THEN
    act1:  inst_state = inst_state ⌁ ((prop×{s})×{FAIL_ACTIV_4})
    act2:  run_inst = run_inst ⌁ ((prop×{s})×{run_inst(token_owner(s) ↦ s) \ suspct_peers(token_owner(s) ↦ s)})
    act3:  failr_inst = failr_inst ⌁ ((prop×{s})×{failr_inst(token_owner(s) ↦ s) ∪ suspct_peers(token_owner(s) ↦ s)})
    act4:  suspct_peers = suspct_peers ⌁ ((prop×{s})×{∅})
END

FAIL_CONFIGURE:  extended ordinary
  REFINES  FAIL_CONFIGURE
  ANY
    s  >
    prop  >
  WHERE
    grd1:  s ∈ SERVICES not theorem>
    grd2:  prop ⊆ PEERS not theorem>
    grd3:  inst_state(token_owner(s) ↦ s) = FAIL_ACTIV_4 not theorem>
    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})
END

FAIL_IGNORE:  extended ordinary
  REFINES  FAIL_IGNORE
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: extended ordinary
REFINES
REDEPLOY_INSTC

ANY
s → a service s
i → an instance i
WHERE
grd1: s ∈ SERVICES not theorem
grd2: i ∈ PEERS not theorem
grd3: i ∈ run_inst(token_owner(s) ↦ s) ∪ failr_inst(token_owner(s) ↦ s) ∪ unav_peers ∪ dep_instc(token_owner(s) ↦ s) not theorem
grd4: i ∈ actv_instc(token_owner(s) ↦ s) not theorem
M21

\[ \text{grd5: } \text{inst\_state}(\text{token\_owner}(s) \Rightarrow s) = \text{FAIL\_CONFIG\_4} \not= \text{FAIL\_CONFIG\_4} \]

\[ \text{grd6: } \text{card}(\text{actv\_instc}(\text{token\_owner}(s) \Rightarrow s)) < \text{deplo\_inst}(s) \]

\[ \text{grd7: } \text{card}(\text{dep\_instcs}(\text{token\_owner}(s) \Rightarrow s)) + \text{card}(\text{run\_inst}
\text{(token\_owner}(s) \Rightarrow s)) < \text{min\_inst}(s) \]

THEN

\[ \text{act1: } \text{actv\_instc}(\text{token\_owner}(s) \Rightarrow s) = \text{actv\_instc}(\text{token\_owner}
(s) \Rightarrow s) \cup \{i\} \]

END

REDEPLOY\_INSTS: extended ordinary

REFINES

REDEPLOY\_INSTS

ANY

\[ \text{s } \]

prop

WHERE

\[ \text{grd1: } s \in \text{SERVICES} \not= \text{FAIL\_CONFIG\_4} \]
\[ \text{grd2: } \text{prop} \subseteq \text{PEERS} \not= \text{FAIL\_CONFIG\_4} \]
\[ \text{grd3: } \text{card}(\text{actv\_instc}(\text{token\_owner}(s) \Rightarrow s)) = \text{deplo\_inst}(s) \]

THEN

\[ \text{grd4: } \text{card}(\text{dep\_instcs}(\text{token\_owner}(s) \Rightarrow s)) + \text{card}(\text{run\_inst}
\text{(token\_owner}(s) \Rightarrow s)) < \text{min\_inst}(s) \]

\[ \text{grd5: } \text{inst\_state}(\text{token\_owner}(s) \Rightarrow s) = \text{FAIL\_CONFIG\_4} \]

\[ \text{grd6: } \text{prop} = \text{run\_inst}(\text{token\_owner}(s) \Rightarrow s) \setminus \text{unav\_peers} \]

THEN

\[ \text{act1: } \text{dep\_instcs} = \text{dep\_instcs} \cup ((\text{prop} \times \{s\}) \times \{\text{dep\_instcs}
\text{(token\_owner}(s) \Rightarrow s) \cup \text{actv\_instc}(\text{token\_owner}(s) \Rightarrow s)\}) \]

\[ \text{act2: } \text{actv\_instc}(\text{token\_owner}(s) \Rightarrow s) = \emptyset \]

END

REDEPLOY: extended ordinary

REFINES

REDEPLOY

ANY

\[ \text{s } \]

prop

WHERE

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

THEN

\[ \text{grd4: } \text{actv\_instc}(\text{token\_owner}(s) \Rightarrow s) = \emptyset \not= \text{FAIL\_CONFIG\_4} \]
\[ \text{grd5: } \text{dep\_instcs}(\text{token\_owner}(s) \Rightarrow s) \not= \emptyset \not= \text{FAIL\_CONFIG\_4} \]
\[ \text{grd6: } \text{card}(\text{run\_inst}(\text{token\_owner}(s) \Rightarrow s)) + \text{card}(\text{dep\_instcs}
(token_owner(s) ⇒ s) ≥ min_inst(s) not theorem > 
    grd7: prop = run_inst(token_owner(s) ⇒ s)\unav_peers not theorem > 

THEN 
    act1: inst_state = inst_state ((prop×{s})×{DPL_4}) > 
    act2: run_inst = run_inst ((prop×{s})× {run_inst(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 theorem > 
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 theorem > 
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 

Page 9
M21

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

END