Seppo Horsmanheimo | Maryam Kamali | Mikko Kolehmainen | Mats Neovius | Luigia Petre | Mauno Rönkkö | Petter Sandvik

On Proving Recoverability of Smart Electrical Grids

TURKU CENTRE *for* COMPUTER SCIENCE

TUCS Technical Report
No 1096, November 2013

On Proving Recoverability of Smart Electrical Grids

Seppo Horsmanheimo
VTT Technical Research Centre of Finland
seppo.horsmanheimo@vtt.fi
Maryam Kamali
Åbo Akademi University, Department of Computer Science
Joukahaisenkatu 3-5A, 20520 Turku, Finland
mkamali@abo.fi
Mikko Kolehmainen
University of Eastern Finland, Department of Environmental Sciences
70211 Kupio, Finland
mkolehmainen@uef.fi
Mats Neovius
Åbo Akademi University, Department of Computer Science
mneovius@abo.fi
Luigia Petre
Åbo Akademi University, Department of Computer Science
lpetre@abo.fi
Mauno Rönkkö
University of Eastern Finland, Department of Environmental Sciences
mronkko@uef.fi
Petter Sandvik
Åbo Akademi University, Department of Computer Science,
Turku Centre for Computer Science
psandvik@abo.fi

TUCS Technical Report
No 1096, November 2013

Abstract

Smart electrical grids refer to networked systems for distributing and transporting electricity from producers to consumers, by dynamically configuring the network through remotely controlled (dis)connectors. The consumers of the grid have typically distinct priorities, e.g., a hospital and an airport have the highest priority and the street lighting has a lower priority. This means that when electricity supply is compromised, e.g., during a storm, then the highest priority consumers should either not be affected or should be the first for whom electricity provision is recovered. In this paper, we propose a general formal model to study the provability of such a property. We have chosen Event-B as our formal framework due to its abstraction and refinement capabilities that support correct-by-construction stepwise development of models; also, Event-B is tool supported. Being able to prove various properties for such critical systems is fundamental nowadays, as our society is increasingly powered by dynamic digital solutions to traditional problems.

Keywords: Formal methods, Smart electrical grids, Event-B, Correct-by-Construction

TUCS Laboratory
Distributed Systems Laboratory

1 Introduction

Our society and lifestyles are rapidly changing to being powered by digital technologies. One prominent example is provided by the electrical grids that are increasingly digital. In grids, as well as in other control systems, action is determined by sensing, monitoring, and measuring. The control part of the paradigm is nowadays implemented in software, which forms a critical infrastructure for decision making in these *smart grids*. Hence, the high-quality of the controlling software is of utmost importance.

This modus operandi of smart grids leads to a high degree of flexibility for the grid configuration and functions. Smart grids are networked systems for connecting electricity generators to consumers, by dynamically configuring the network through remotely controlled (dis)connectors. The consumers of the grid have typically distinct priorities, e.g., a hospital and an airport have the highest priority and the street lighting has a lower priority. This means that when electricity supply is compromised, e.g., due to a storm or peak consumption, then the order of (re-establishing) electricity provision is with respect to the priorities. Another example of functional flexibility of smart grids is the possibility of (regularly) changing priorities, e.g., in the evenings, factories and office buildings have a lower lighting priority than living areas. In this context, the *problem* that we address here is how can we trust that consumers with the highest priority have almost always a path in the grid from an electricity generator; and, if such a connection is lost, is finding an alternative one guaranteed?

The *solution* that we propose in this paper is based on formal methods [36]. Formal methods refer to the application of mathematical techniques to the design of computer hardware and software, for delivering *provably correct* systems, i.e., systems of high-quality. Formal methods are based on the capture of system requirements in a specific, precise model. Importantly, such a model can be analysed for various properties and, if the formal method permits, in some cases also stepwise developed until an implementation is formed. By following such a formal development, we are *sure* that the final result correctly implements the requirements of the system.

Examples of the industrial undertaking of formal methods are increasing. The famous line 14 of the driverless Parisian metro [11], developed in 1998 using the B method [1], is the first notable example of a formal method-based development, reviewed in [25]. The method used by Siemens for developing the software controlling the line 14 train ensured its correctness in a mathematical manner that effectively eliminated the unit testing from the software lifecycle. No human resources are now needed to operate the trains and in addition, the trains are faster, hence fewer are needed in total. More recent examples of formal method usage in industry can be seen for instance with Space Systems [15] and SAP [8].

The fundamental design techniques of formal methods that we employ here are *abstraction* and *refinement*. We start from a simple, abstract model of the smart grid, that is striped away of many details (including connections among grid nodes), so that it is easy to prove some desired properties for it. Among others, we prove that high-priority consumers can recover from failure. Then, we add details in a stepwise manner to our model, until we reach a level of abstraction that agrees to our purposes. In this paper, we prove that, when connections to high-priority consumers fail, the smart grid can find alternative connections for these consumers, whenever there are connections available. We also *prove* that the more detailed models correctly refine the less detailed models.

In our work we use the Event-B formal method [2] for system modelling and analysis, due to its abstraction and refinement capabilities. Event-B comes with an associated toolset, the Rodin Platform, a theorem prover-based environment where proofs about the models are generated automatically and discharged either automatically or interactively.

Our contribution is thus twofold:

1. We propose smart grid models at different levels of abstraction and prove that the more concrete models refine the more abstract ones.
2. We demonstrate the recoverability property for smart grids as an invariant for our models.

Importantly, as our proving is based on assumptions, we are able to reason on when we can prove the recoverability property for smart grids and discuss why. Also, our modelling is developed for reusability: depending on various criteria, our recovery methodology can produce different reconnected paths, thus our most detailed model can be reused for various purposes.

We proceed as follows. In Section 2 we describe the smart grids and our model restrictions, in Section 3 we outline Event-B, and in Section 4 we describe our Event-B model to the extent needed in this paper. In Section 5 we discuss some interesting proving aspects of our approach. Related work is reviewed in Section 6 and some conclusions are presented in Section 7.

2 Smart Grids

The term *grid* indicates a structured framework of interconnected elements. Within the domain of the electricity, the electrical grid refers to the interconnected network delivering electricity from power plants to consumers. The grid comprises electricity generators labelled G_i, substations labelled S_j, and consumers labelled C_k, where i,j,k denote natural numbers. This is illustrated in Figure 1a) where edges between G_i and S_j or in between S_j denote the (high-voltage) transmission network and the edges between S_j and C_k denote the distribution network.

The term *Smart Grid* (SG) comprises an intelligent grid of this kind, typically considered an enhancement of the traditional 20th century grid. Some significant

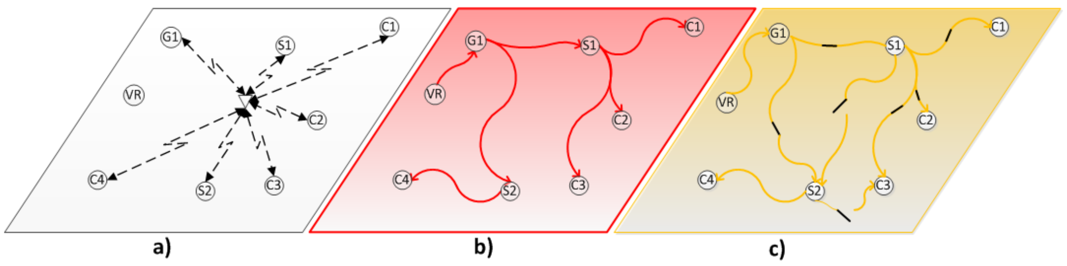

Figure 1: Smart Grid's Layers of Operation: a) Information Flow; b) Power Flow Tree; c) Smart Grid (SG) graph

differences are presented in Table 1 [14]. Gharavi and Ghafurian [17] define SG as a system that uses information and provides two-way, cyber-secure communication. This enables computational intelligence for a clean, safe, secure, reliable, resilient, efficient and sustainable system. On the differences between the 20th century grid and SG, we note that the bidirectional communication is fundamental, implying several of the others. This connects SG with the information revolution era, where "smart" refers to using information regarding the grid for enabling pervasive self* properties. A categorisation on the physical realisation of the bidirectional communication [14] and a survey on SG [13] may be found elsewhere.

Table 1: Comparison between the 20th century grid and the smart grid

	20th century grid [14]	Smart grid [14]	Smart grid [17]
1	Electromechanical	Digital	
2	One-way communication	Two-way communication	Interactive
3	Centralised generation	Distributed generation	
4	Hierarchical	Network	
5	Few sensors	Sensors throughout	
6	Blind	Self-monitoring	
7	Manual restoration	Self-healing	Self-healing
8	Failures and blackouts	Adaptive and islanding	Flexible
9	Manual Check/test	Remote Check/test	
10	Limited control	Pervasive control	
11	Few customer choices	Many customer choices	

Research on SG has several points of focus. First, smart infrastructure systems are studied, in particular the communication infrastructure (the information flow layer in Fig. 1a)). This addresses the communication technology and protocols, the information subsystem concerned with information interoperability and the energy subsystem concerned with, among others, small scale energy production such as via solar panels. Second, the management system is of interest, considering energy efficiency, operation costs reduction, demand and supply balance, emission control and utility maximization. Third, the smart protection system is very relevant, being concerned with user errors, equipment failures, natural disasters and deliberate cyber attacks [14]. Of these, in this paper we are concerned with the smart protection system and its feature of failure recovery, implemented in software. This is of outmost importance to increase the SG reliability and is strongly motivated by the annual costs of outage, e.g., in 2002 these were estimated in the US to 79 billion dollars [24].

For this failure recovery software, we assume reliable outage detection; reports on this by phasor measurement may be found elsewhere [33, 34, 37]. With respect to failure recovery, we formally outline the requirements of a fully and a partly connected grid (network) topology. We do recovery by switching (dis)connectors on the edges of this network, i.e. by operating the dis(connectors) depicted on the edges in Fig. 1c). Hence, we assume each element of the network to be remotely controllable. We assume the momentary topology to be of a tree structure with an added virtual root node, labelled VR in Fig. 1b). This tree connects all consumers (leafs) with the generators (first level nodes) by substations (intermediate nodes). The intermediate nodes may be connected (an open edge between S1 and S2), much alike power substations in reality for the sake of redundancy. In addition, we consider the consumers to have a priority of criticality determined by the cost (monetary, moral, etc) of a shortage, where a hospital is more critical than residential houses. Contrary to the 20th century grid, except the momentary hierarchical tree structure, the network structure covering for all possible connections is not hierarchical; we assume it to be a graph.

A blackout partitions the grid tree to a tree connected to the virtual root and a disconnected compromised subtree. Recovery effectively means reconnecting this compromised subtree's leafs to the virtual root node by finding an alternative route in the tree. As the recovery strategy implemented may vary depending on the setting, we implement a general strategy of circumventing the compromised subtree's root node; this general strategy can be adapted to many settings later. When recovery is not possible (there are no alternative paths to choose from), an operator could dispatch human resources for repairing the SG point of failure. The priorities of consumers set the recovery order when needed. An optimal condition with respect to our problem is thereby reached when the tree is fully connected, i.e. all leafs, independently of priority, have a path to the root.

3 Event-B

The earlier B-Method [1] is a formal approach for specifying and developing highly dependable software, and it has been successfully used in the development of several complex real-life applications [11, 12]. The Event-B [2] formalism is derived from the B-Method and the Action Systems [4, 6, 35] framework, and was created for modelling and reasoning about parallel, distributed and reactive systems. The associated Rodin Platform [3, 12] tool provides automated support for modelling and verification by theorem proving in Event-B. In the following, we shortly describe the Event-B language to the extent needed in this paper.

3.1 The Event-B Language

In Event-B, a model of a system consists of a dynamic part, *machine*, and optionally also a static part, *context*. An Event-B context can specify constants, carrier sets, and axioms about these. A machine in Event-B optionally *sees* a context, and describes the model state by *variables*, updated by *events*. Events are atomic sets of (simultaneous) variable updates, and each event may contain *guards*, which are associated predicates, that must evaluate to true for the event to be enabled, i.e., be able to execute. Events can be declared as *convergent*, i.e., there is a natural number expression called *variant* in the machine that decreases when this event is executed. It is the proof method for expressing liveness properties and ensuring that our model will eventually terminate. If more than one event is enabled simultaneously, the choice between the events is non-deterministic. An Event-B machine should also include *invariants*, i.e., properties that must hold for any reachable state of the model. Thus, these properties must be established by a special initialisation event and hold before and after every occurrence of any other event. The proof manager in the Rodin Platform [2, 12] tool automatically generates what needs to be proved in order for an invariant to hold.

Event-B provides a stepwise refinement-based approach to system development, preserving correctness by gradually detailing a model. Starting from an abstract model, refinement can be either *horizontal* or *vertical*. Horizontal refinement, or superposition refinement [7, 23], refers to adding new variables, events and constants in addition to existing ones. Old events can also be modified, typically either updating the newly introduced variables or introducing more deterministic assignments on the old variables, while also strengthening the event guards. Vertical refinement, or data refinement [5], corresponds to replacing some abstract variables with their concrete counterparts and accordingly changing the events. When we only update the newly variables or strengthening the guards, an abstract event *extends* to its refined model. Otherwise, an abstract event is *refined* to the corresponding refined event. A refined event might contain the *with* clause that declares a witness for each disappearing parameter of the corresponding abstract event. The witness allows us to prove the correctness of refinement. In order to guarantee the correctness of refinement, we need to show that the execution of a refined event is not contradictory to its abstract version. It is shown by discharging the simulation proof obligation (SIM) which is automatically generated by Roding platform tool.

4 Three Smart Grid Models: M_0, M_1, M_2

In this section we describe the high-level models M_0, M_1 and M_2 for SG. Our models are at three increasing levels of detail so that each model is a refinement of the previous one: $M_0 \sqsubseteq M_1 \sqsubseteq M_2$. In the initial model, we specify the set of consumers with different priorities, introduce random electricity outage and specify the behaviour of a recovery mechanism. We define the correctness properties of the recovery mechanism based on the priority of the consumers. We fold (hide) further detail of SG and magically recover from the outage problem. In the second model, we unfold (add) new data, refine events to model the tree structures in SG as well as link failures. We design the recovery mechanism in a way that guarantees that, after recovery, the structure of the momentary SG remains a tree and all high-priority consumers are connected to a generator. In the third model, we detail our recovery mechanism by refining (splitting) events to distinguish between successful and unsuccessful recovery; in the latter case, we provide information on the failed subtree, for human-directed reconfiguration, that we assume to take place.

4.1 The Initial Model M_0

The initial model M_0 that we construct is very abstract: we do not consider the underlying SG connections but only the SG consumers; the SG connections are introduced in the subsequent refinement. M_0 thus allows us to specify our recovery goal very abstractly. The model M_0 is formed of the static part and the dynamic part, as follows.

4.1.1 The Static Part

The static part of our model is described below and contains the sets $NODE$, $MODE$, $STATE$ and the constants $consumer$, $generator$, $pr1$, $pr2$, $normal$, $recovery$, $optimal$, on and off. The SG nodes are elements of the finite and non-empty $NODE$ set (**axm1** and **axm2**). Nodes can be either consumers or generators, so the $NODE$ set is partitioned into $consumer$ and $generator$ (**axm3**). Some consumers have higher priority, so we partition $consumer$ into non-empty set of $pr1$ and $pr2$ (**axm4-5**). SG can be in three different modes (**axm6**) and the state of consumers can be either on or off (**axm7**).

CONSTANTS $consumer \quad generator \quad pr1 \quad pr2 \quad normal \quad recovery \quad optimal \quad on \quad off$
SETS $NODE \quad MODE \quad STATE$
AXIOMS
- **@axm1** $finite(NODE)$
- **@axm2** $NODE \neq \varnothing$
- **@axm3** $partition(NODE, consumer, generator)$
- **@axm4** $partition(consumer, pr1, pr2)$
- **@axm5** $pr1 \neq \varnothing \wedge pr2 \neq \varnothing \wedge generator \neq \varnothing$
- **@axm6** $partition(MODE, \{optimal\}, \{normal\}, \{recovery\})$
- **@axm7** $partition(STATE, \{on\}, \{off\})$

4.1.2 The Dynamic Part

In the dynamic part of the model M_0, we define the state of consumers and the SG mode. We specify that whenever there is some power outage in higher priority consumers, there would be a mechanism to recover from the outage. We model a magical recovery mechanism which eventually recovers high-priority consumers from the outage.

The outline of the dynamic part is shown below. The $cons\_st$ variable models the state of each consumer (**inv1**) and the $mode$ variable models the current mode of SG (**inv2**). The $pr1$ consumers have higher priority than the $pr2$ consumers with respect to recovery, modelled by invariant **inv3**: when SG is in $normal$ mode, all high-priority consumers are in on state, however, some lower priority consumers are in off state (**inv4**). If SG mode is $optimal$, all consumers (higher and lower priority) are in on state (**inv5**). When SG is in $recovery$ mode, there is power outage of at least one higher priority consumer (**inv6**).

```
VARIABLES
  $cons\_st\ mode$
INVARIANTS
  @inv1 $cons\_st \in consumer \rightarrow STATE$
  @inv2 $mode \in MODE$
  @inv3 $mode = normal \Rightarrow (\forall c \cdot c \in pr1 \Rightarrow cons\_st(c) = on)$
  @inv4 $mode = normal \Rightarrow (\exists r \cdot r \in pr2 \land cons\_st(r) = off)$
  @inv5 $mode = optimal \Rightarrow (\forall c \cdot c \in consumer \Rightarrow cons\_st(c) = on)$
  @inv6 $mode = recovery \Rightarrow (\exists c \cdot c \in pr1 \land cons\_st(c) = off)$
```

Our assumption is that initially all consumers are on and as a consequence, to satisfy the invariants, SG is in the $optimal$ state, as shown in the **INITIALISATION** event above. The **failure** event models a random power outage that happens to a subset i of consumers. SG mode changes from $normal$ or $optimal$ to either $recovery$ or $normal$ depending on the priority of consumers. The state of the consumer subset i with power outage updates from on to off.

```
INITIALISATION
BEGIN
  @act1 $mode := optimal$
  @act2 $cons\_st := consumer \times \{on\}$
END
```

```
failure
ANY i m WHERE
  @grd1 $mode \in \{normal, optimal\}$
  @grd2 $i \subset consumer \land i \neq \varnothing$
  @grd3 $((i \cap pr1 \neq \varnothing) \land m = recovery) \lor$
        $((i \cap pr1 = \varnothing) \land m = normal)$
THEN
  @act1 $mode := m$
  @act2 $cons\_st := cons\_st \mathbin{<\!\!\!+} (i \times \{off\})$
END
```

Two events **failure_recovery** and **recovery_complete** magically solve the outage, gradually reconnecting the compromised high- priority consumers, by switching their state to on. When the state of the last subset of consumers updates to on in **recovery_ complete**, the SG mode switches to either $normal$ or $optimal$ depending on the state of lower priority consumer. When SG is in $normal$ mode, there are still some lower priority consumers with power outage.

```
failure_recovery
ANY i WHERE
  @grd1 $i \subset pr1$
  @grd2 $cons\_st[i] = \{off\} \wedge i \neq \varnothing$
  @grd3 $mode = recovery$
  @grd4 $\exists j \cdot j \notin i \wedge j \in pr1 \wedge$
          $cons\_st(j) = off$
THEN
  @act1 $cons\_st := cons\_st \mathbin{<\!\!+} (i \times \{on\})$
END
```

```
recovery_complete
ANY i m WHERE
  @grd1 $i \subset pr1$
  @grd2 $cons\_st[i] = \{off\} \wedge i \neq \varnothing$
  @grd3 $mode = recovery$
  @grd4 $cons\_st[(pr1 \setminus i)] = \{on\}$
  @grd5 $(cons\_st[(consumer \setminus i)] = \{on\} \wedge m = optimal) \vee$
        $(cons\_st[(consumer \setminus i)] \neq \{on\} \wedge m = normal)$
THEN
  @act1 $cons\_st := cons\_st \mathbin{<\!\!+} (i \times \{on\})$
  @act2 $mode := m$
END
```

To be sure that the recovery mechanism, modelled by **failure_ recovery** and **recovery_ complete** events, eventually recovers the higher priority consumers from power outage, we define the $cons\_st^{-1}[\{off\}]$ variant (not shown here). We have proved that, with each execution of either **failure_ recovery** or **recovery_ complete** events, the number of high-priority consumers with state off decreases.

The **optimising** and **optimising_complete** events deal with this situation. The SG mode switches to $optimal$ when all consumers are in the on state.

```
optimising
ANY i WHERE
  @grd1 $mode = normal$
  @grd2 $i \subset pr2 \wedge i \neq \varnothing$
  @grd3 $cons\_st[i] = \{off\}$
  @grd4 $(\exists j \cdot j \notin i \wedge j \in pr2 \wedge cons\_st(j) = off)$
THEN
  @act1 $cons\_st := cons\_st \mathbin{<\!\!+} (i \times \{on\})$
END
```

```
optimising_complete
ANY i WHERE
  @grd1 $mode = normal$
  @grd2 $i \subseteq pr2 \wedge i \neq \varnothing$
  @grd3 $cons\_st[i] = \{off\}$
  @grd4 $cons\_st[(consumer \setminus i)] = \{on\}$
THEN
  @act1 $mode := optimal$
  @act2 $cons\_st := cons\_st \mathbin{<\!\!+} (i \times \{on\})$
END
```

In addition, we define a theorem to guarantee that the model is deadlock-free as follows: the theorem is a disjunction of the guard of all events, meaning that at any moment at least one event is enabled.

theorem

$$\begin{aligned}
&(mode = normal \wedge (\exists\, i \cdot i \subseteq consumer \wedge i \neq \varnothing \wedge \\
&\quad (\exists m \cdot ((i \cap pr1 \neq \varnothing) \wedge m = recovery) \vee ((i \cap pr1 = \varnothing) \wedge m = normal)))) \vee \\
&(mode = optimal \wedge (\exists i \cdot i \subseteq consumer \wedge i \neq \varnothing \wedge \\
&\quad (\exists m \cdot ((i \cap pr1 \neq \varnothing) \wedge m = recovery) \vee ((i \cap pr1 = \varnothing) \wedge m = normal))) \vee \\
&(mode = recovery \wedge (\exists i \cdot i \subseteq pr1 \wedge cons\_st[i] = \{off\} \wedge i \neq \varnothing \wedge (\exists j \cdot j \notin i \wedge j \in pr1 \wedge cons\_st(j) = off))) \vee \\
&(mode = recovery \wedge (\exists i \cdot i \subseteq pr1 \wedge cons\_st[i] = \{off\} \wedge i \neq \varnothing \wedge cons\_st[(pr1 \setminus i)] = \{on\} \wedge \\
&\quad (\exists m \cdot (cons\_st[(consumer \setminus i)] = \{on\} \wedge m = optimal) \vee \\
&\quad (cons\_st[(consumer \setminus i)] \neq \{on\} \wedge m = normal)))) \vee \\
&(mode = normal \wedge (\exists i \cdot i \subseteq pr2 \wedge cons\_st[i] = \{off\} \wedge i \neq \varnothing \wedge (\exists j \cdot j \notin i \wedge j \in pr2 \wedge cons\_st(j) = off))) \vee \\
&(mode = normal \wedge cons\_st[consumer] = \{on\} \wedge (\exists i \cdot i \subseteq pr2 \wedge i \neq \varnothing \wedge cons\_st[(consumer \setminus i)] = \{on\}))
\end{aligned}$$

4.2 The Second Model M_1

Recovery from the power outage that affects high-priority consumers is achieved in the model M_0 simply by switching their state from off to on. In this section, we refine the initial model M_0 to also specify links between nodes in SG. For this, we keep two structures. The SG *graph* denotes the entire grid, with all the available links and nodes. The momentary SG *tree* denotes the currently used links and nodes. This is illustrated in Figure 2. The SG tree is, in fact, extracted from the SG graph. When the momentary SG tree suffers a failure, it needs to find an alternative path, in the SG graph, to make all the higher priority consumers connected to SG again.

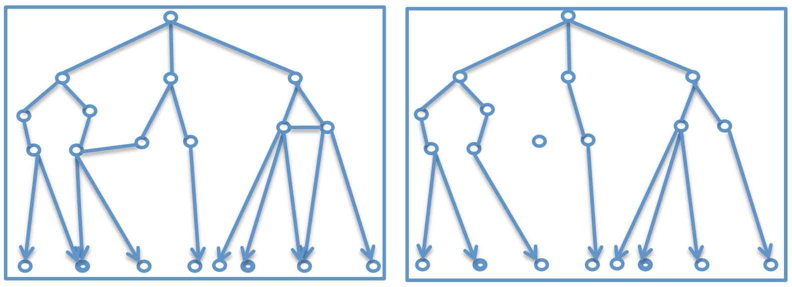

Figure 2: An SG graph and an instance of one of its trees

4.2.1 The Static Part

In order to express and reason about the recovery property, we need to define a $closure$ property for relations. In the context shown below, the constant cl is defined as a total function from $netrel$ ($NODE \leftrightarrow NODE$, **axm7**) to $netrel$ (**axm8**). The characteristic properties of $cl(r)$ are: (i) The relation r is included in $cl(r)$. (ii) The forward composition of $cl(r)$ with r is included in $cl(r)$. (iii) The relation $cl(r)$ is the smallest relation dealing with (i) and (ii). These properties are modelled respectively by **axm9**, **axm10**, and **axm11**; the tree-adapted closure theorems are introduced in **thm1-3**.

We also model a tree structure of the network, in the constant $tree$, with the following properties: (i) A tree is a total surjection function from $NODE \setminus \{root\}$ to $NODE \setminus consumer$ (**axm12**). (ii) The $consumer$ set are leaves of tree and $root$ is the top point in a tree (**axm12**). (iii) The tree is acyclic (**axm13**). The theorems (**thm4-9**) further describe tree properties.

Moreover, we define the net and $initial\_net\_setting$ constants. The net constant is an asymmetric, irreflexive and total surjective function modelling the SG graph (**axm14**, **axm15**). The $initial\_net\_setting$ constant is a tree (**axm16-18**), extracted from net (**axm19**).

The adapted theorems for closure allow us to prove the adapted theorems of the tree structure which will be used in the dynamic part of the second model to guarantee the correctness of the SG tree evolution during the recovery process.

constants
$netrel \quad cl \quad tree \quad root \quad net \quad initial\_net\_setting$
axioms
@axm7 $netrel = NODE \leftrightarrow NODE$
@axm8 $cl \in netrel \rightarrow netrel$
@axm9 $\forall r \cdot r \subseteq cl(r)$
@axm10 $\forall r \cdot r; cl(r) \subseteq cl(r)$
@axm11 $\forall r, t \cdot (r \subseteq t \wedge r; t \subseteq t \Rightarrow cl(r) \subseteq t)$
theorem @thm1 $\forall r \cdot r \in netrel \Rightarrow cl(r) = r \cup (r; cl(r))$
theorem @thm2 $\forall t \cdot (\forall s \cdot s \subseteq t^{-1}[s] \Rightarrow s = \varnothing) \Rightarrow cl(t) \cap (NODE \lhd id) = \varnothing$
theorem @thm3 $cl(\varnothing) = \varnothing$
@axm12 $tree \in NODE \setminus \{root\} \twoheadrightarrow NODE \setminus consumer$
@axm13 $\forall s \cdot s \subseteq tree^{-1}[s] \Rightarrow s = \varnothing$
theorem @thm4 $\forall T \cdot root \in T \wedge tree^{-1}[T] \subseteq T \Rightarrow NODE \subseteq T$
theorem @thm5 $cl(tree^{-1})[\{root\}] \cup \{root\} = NODE$
theorem @thm6 $cl(tree)[consumer] \cup consumer = NODE$
theorem @thm7 $\forall T \cdot consumer \subseteq T \wedge tree[T] \subseteq T \Rightarrow NODE \subseteq T$
theorem @thm8 $\forall S \cdot S \subseteq tree[S] \Rightarrow S = \varnothing$
theorem @thm9 $cl(tree) \cap (NODE \lhd id) = \varnothing$
@axm14 $net \in NODE \setminus \{root\} \leftrightarrow\!\!\!\!\twoheadrightarrow generator$
@axm15 $net \cap net^{-1} = \varnothing \wedge NODE \lhd id \cap net = \varnothing$
@axm16 $initial\_net\_setting \in NODE \setminus \{root\} \twoheadrightarrow NODE \setminus consumer$
@axm17 $cl(initial\_net\_setting^{-1})[root] \cup \{root\} = NODE$
@axm18 $\forall S \cdot S \subseteq initial\_net\_setting[S] \Rightarrow S = \varnothing$
@axm19 $initial\_net\_setting \subseteq net$
END

4.2.2 The Dynamic Part

In M_1, we show how the recovery mechanism solves the power outage problem by changing the SG momentary tree. There are five variables in M_1: $node$, dyn_net, $failed\_link$, $failed\_path$ and $failed\_flag$. The $node$ variable is a subset of $NODE$ and models the nodes of the SG momentary tree (**inv6**). In our model, $root$ always belongs to the SG tree, hence $root \in node$ (**inv7**). In **inv8** we link $node$ to the abstract variable $cons\_st$ from M_0: we model that there is no consumer with state off in set $node$, but all consumers that are in state on are members of $node$. The dyn_net models the SG tree with nodes and links between them: a surjective function from $node \setminus \{root\}$ to $node \setminus consumer$ (**inv9**) that is loop-free (**inv10**).

In the initial model, power outage hits a subset of consumers randomly. In M_1, it happens due to some link failures in the SG tree. The partial function $failed\_link$ denotes the set of link failures in SG (**inv12**). We note that the root cannot fail, because the root is virtual and actually not belonging to SG. We use it for the sake of modelling. Each link failure leads to disconnectivity in a part of tree. We store the disconnected subtree for each link failure in the $failed\_path$ function (**inv13**). In order to guarantee that a disconnected subtree of each link failure is a tree we introduce **inv14** and **inv15**. Invariant **inv14** denotes that each subtree is a partial function from $NODE \setminus \{root\}$ to $NODE \setminus \{root\}$. Invariant **inv15** denotes that all the nodes in a subtree of a failed link, say $i \mapsto j$, can be reached from both nodes i and j. In other words, any node in the subtree is a child of node i and j. In addition, we ensure that the built subtree of each $failed\_link$ satisfies **inv9**: consumers are leaves of the subtree (**inv16**).

VARIABLES

$node \quad dyn\_net \quad failed\_link \quad failed\_path \quad failed\_flag$

INVARIANTS

@inv6 $node \subseteq NODE$

@inv7 $root \in node$

@inv8 $cons\_st^{-1}[\{off\}] \cap node = \varnothing \wedge cons\_st^{-1}[\{on\}] \subseteq node$

@inv9 $dyn\_net \in node \setminus \{root\} \twoheadrightarrow node \setminus consumer$

@inv10 $\forall s \cdot (s \subseteq dyn\_net^{-1}[s] \Rightarrow s = \varnothing)$

@inv12 $failed\_link \in NODE \setminus \{root\} \nrightarrow NODE \setminus \{root\}$

@inv13 $failed\_path \in failed\_link \nrightarrow (NODE \leftrightarrow NODE)$

@inv14 $\forall f \cdot f \in dom(failed\_path) \Rightarrow failed\_path(f) \in NODE \setminus \{root\} \nrightarrow NODE \setminus \{root\}$

@inv15 $\forall i, j \cdot i \mapsto j \in dom(failed\_path) \Rightarrow (\forall k \cdot k \in ((dom(failed\_path(i \mapsto j)) \cup$
$ran(failed\_path(i \mapsto j))) \setminus \{j\}) \wedge k \in (cl(failed\_path(i \mapsto j)))^{-1}[\{j\}])$

@inv16 $\forall i, j \cdot i \mapsto j \in dom(failed\_path) \Rightarrow consumer \cap ran(failed\_path(i \mapsto j)) = \varnothing$

@inv17 $failed\_flag \in BOOL$

@inv18 $mode = recovery \Rightarrow dom(failed\_path) = failed\_link$

@inv19 $failed\_link \neq \varnothing \wedge dyn\_net \cap failed\_link = \varnothing \Rightarrow$
$(\exists r \cdot r \in consumer \wedge r \notin (cl(dyn\_net))^{-1}[\{root\}])$

@inv20 $failed\_flag = FALSE \wedge mode \neq recovery \Rightarrow failed\_link = \varnothing \wedge failed\_path = \varnothing$

@inv21 $\forall r \cdot r \in consumer \wedge r \notin (cl(dyn\_net))^{-1}[\{root\}] \wedge failed\_link \neq \varnothing \wedge$
$dyn\_net \cap failed\_link = \varnothing \Rightarrow cons\_st(r) = off$

theory @mthm1 $\forall S \cdot (root \in S \wedge dyn\_net^{-1}[S] \subseteq S \Rightarrow node \subseteq S)$

theory @mthm2 $node \setminus \{root\} \subseteq (cl(dyn\_net))^{-1}[\{root\}]$

theory @mthm3 $cl(dyn\_net) \cap (NODE \lhd id) = \varnothing$

END

To update the value of the $failed\_path$ variable after a link failure, we need to define a flag to allow us to do preprocessing before recovery. This is just for modelling purposes. The flag $failed\_flag$ is modelled by **inv17**. When $failed\_flag = TRUE$ it models that changes to $recovery$ mode. In other words, when the network is in $recovery$ mode, it means that $failed\_path$ contains all subtrees of all failures (**inv18**). The consequence of any failure is having at least one consumer which suffers from power outage due to the tree structure of SG. Invariant **inv19** guarantees that the property is satisfied. Finally, invariant **inv20** denotes that when the network is not in recovery mode, $failed\_link$ is empty. It is formulated as shown in **inv21**.

In order to ensure that every node in the dyn_net is reachable from the root node, we model the theorems **mthm1-3**, derived from existing invariants. Theorem **mthm2** denotes that all nodes in $node$ except $root$ are reachable from $root$. Theorem **mthm1** is used in the proof of **mthm2** and **mthm3** is introduced to satisfy the no-loop property.

To address the newly added structures we add three new events in M_1 as well as refine the abstract events. The newly introduced events define failure of links (**Link_fail** event), preprocessing to update function $failed\_path$ (**Failed_path_ update** event) and distributing the knowledge about link failure to the grid (**Failed_tree_ update** event). We explain these in more detail below.

Link_fail This event models the failure of a non-empty (**grd3**) subset of links in SG, except virtual links to root (**grd1**) when SG is in normal or optimal mode (**grd4**). The $failed\_flag$ variable which is initially $FALSE$ (**grd2**) is set to $TRUE$ (**act1**) in order to enable the preprocessing event. The $failed\_link$ set updates to the list of failed links(**act2**) and $failed\_path$ initiates to $\varnothing$ (**act3**). The only enabled event after the execution of the **Link_fail** event is **Failed_path_update** event, to ensure that preprocessing is performed before any further actions.

Failed_path_update This event is enabled when the $failed\_flag$ is $TRUE$ (**grd1**). The event is enabled until all failed links (**grd2**) become members of $dom(failed\_path)$ (**grd3**). This event is aimed at storing a set of subtrees of the grid that are unreachable due to link failures. In order to compute the subtree for each individual failed link, we need to add two computational parameters des and $subtree$. Parameter des is the set of all descendants of nodes of a failed link (**grd4**). Parameter $subtree$ is the set of all links which form the subtree (**grd5**). Function $failed\_path$ is updated so that $failed\_link$ becomes a symbolic 'root' for $subtree$ (**act1**).

```
Link_fail
ANY l WHERE
  @grd1 $l \subseteq dyn\_net \rhd \{root\}$
  @grd2 $failed\_flag = FALSE$
  @grd3 $l \neq \varnothing$
  @grd4 $mode \in \{optimal, normal\}$
THEN
  @act1 $failed\_flag := TRUE$
  @act2 $failed\_link := l$
  @act3 $failed\_path := \varnothing$
END
```

```
Failed_path_update
ANY f des subtree l i WHERE
  @grd1 $failed\_flag = TRUE$
  @grd2 $l \in failed\_link \land l = i \mapsto f$
  @grd3 $l \notin dom(failed\_path)$
  @grd4 $des = (cl(dyn\_net))^{-1}[\{f\}]$
  @grd5 $subtree = des \lhd dyn\_net$
THEN
  @act1 $failed\_path := failed\_path \cup$
                    $(\{l\} \times \{subtree\})$
END
```

Failed_tree_update This event is enabled when preprocessing is complete: the subtree of all failed links are assigned (**grd2**). In order to update SG based on link failures, we remove a failed link, say l, with its subtree, stored in $failed\_path(l)$ from dyn_net. The $node$ set, the set of reachable nodes from root, and the function $cons\_st$ update accordingly. Function $cons\_st$ updates in this event in order to guarantee that if a consumer is unreachable in the grid, the consumer is in off state (**inv8** and **inv21**).

Failure event: The **Failure** event in the initial model is refined to two separate events: **pr1_Failure** and **pr2_Failure**. The **pr1_Failure** event denotes failure of higher priority with/without lower priority consumers (**grd2**). The **pr2_Failure** event denotes failure of lower priority consumers (**grd2**). These events are enabled when the preprocessing (**grd3**) and the SG updating are completed (**grd4-6**). They simply reset $failed\_flag$ (**act2**) and update the SG mode (**act1**). Since parameter i in the abstract model is removed from the concrete events, we need to discharge the similarity (SIM) proof obligations. Thus, we introduce $cons\_st^{-1}[\{off\}]$ as a witness for the removed parameter i. SIM proof obligation is proved by showing that there is at least one instance of i in $cons\_st^{-1}[\{off\}]$ which is concluded from (**inv21**)

```
Failed_tree_update
ANY f  des  subtree  l  i  r  WHERE
 @grd1 $failed\_flag = TRUE$
 @grd2 $dom(failed\_path) = failed\_link$
 @grd3 $l \in failed\_link$
 @grd4 $l \in dyn\_net$
 @grd5 $l = i \mapsto f$
 @grd6 $l \notin dom(failed\_path)$
 @grd7 $des = (cl(dyn\_net))^{-1}[\{f\}]$
 @grd8 $subtree = des \lhd dyn\_net$
 @grd9 $\forall k \cdot k \in consumer \land k \in des \Rightarrow k \in r$
THEN
 @act1 $dyn\_net := dyn\_net \setminus$
         $(\{l\} \cup subtree)$
 @act2 $node := node \setminus des$
 @act3 $cons\_st := cons\_st \Lleftarrow (r \times \{off\})$
END
```

```
pr1_Failure refines failure
WHERE
 @grd1 $mode \in \{optimal, normal\}$
 @grd2 $rcv\_st^{-1}[\{off\}] \cap pr1 \neq \varnothing$
 @grd3 $failed\_flag = TRUE$
 @grd4 $failed\_link \neq \varnothing$
 @grd5 $\forall f \cdot f \in failed\_link \Rightarrow f \in dom(failed\_path)$
 @grd6 $\forall f \cdot f \in failed\_link \Rightarrow f \notin dyn\_net$
WITH
  i   $i = cons\_st^{-1}[\{off\}]$
  m   $m = recovery$
THEN
 @act1 $mode := recovery$
 @act2 $failed\_flag := FALSE$
END
```

```
pr2_Failure refines failure
WHERE
 @grd1 $mode \in \{optimal, normal\}$
 @grd2 $rcv\_st^{-1}[\{off\}] \cap pr1 = \varnothing$
 @grd3 $failed\_flag = TRUE$
 @grd4 $failed\_link \neq \varnothing$
 @grd5 $\forall f \cdot f \in failed\_link \Rightarrow f \in dom(failed\_path)$
 @grd6 $\forall f \cdot f \in failed\_link \Rightarrow f \notin dyn\_net$
WITH
  i   $i = cons\_st^{-1}[\{off\}]$
  m   $m = normal$
THEN
 @act1 $mode := normal$
 @act2 $failed\_flag := FALSE$
 @act3 $failed\_link := \varnothing$
 @act4 $failed\_path := \varnothing$
END
```

Fail_rec event: There are two approaches for power outage recovery: (i) to find an alternative path from the original graph, modeled as net in the static part of the second model and (ii) to repair failed links (human intervention). The **Fail_rec** event combines these two approaches which can be taken to solve the power outage problem for high-priority consumers. The abstract event is extended by adding three computational parameters res, rec_set and $last$. Parameter res is a subset of all subtrees, extracted from $failed\_path$ function (**grd5-6**).

```
Fail_rec extends failure_recovery
ANY i    rec_set    last res    WHERE
  @grd5 $\forall r \cdot r \in failed\_link \Rightarrow failed\_path(r) \subseteq res$
  @grd6 $\forall k \cdot k \in res \Rightarrow (\exists i \cdot i \in failed\_link \land failed\_path(i) \subseteq res)$
  @grd7 $rec\_set \subseteq res \cup (net \setminus failed\_link)$
  @grd8 $dyn\_net \cup rec\_set \in NODE \setminus \{root\} \rightarrowtail NODE$
  @grd7 $dyn\_net \cup rec\_set \in$
        $(node \cup dom(rec\_set) \cup ran(rec\_set)) \setminus \{root\} \rightarrow (node \cup dom(rec\_set) \cup ran(rec\_set))$
  @grd9 $\forall s \cdot (s \subseteq (dyn\_net \cup recovery\_set)^{-1}[s] \Rightarrow s = \varnothing)$
  @grd10 $\forall p \cdot p \in pr1 \setminus last \Rightarrow p \in (cl(dyn\_net \cup recovery\_set))^{-1}[\{root\}]$
  @grd11 $last \subseteq pr1$
  @grd12 $cons\_st[last] = \{off\} \land (last \cap cl(dyn\_net \cup rec\_set)^{-1}[\{root\}] = \varnothing)$
  @grd13 $(cons\_st^{-1}[\{off\}] \cap pr1) \setminus last \subseteq dom(rec\_set \cup dyn\_net)$
  @grd14 $(node \cup dom(rec\_set) \cup ran(rec\_set)) \subseteq \{root\} \cup (cl(dyn\_net \cup rec\_set))^{-1}[\{root\}]$
  @grd15 $i \subseteq dom(rec\_set)$
THEN
  @act2 $dyn\_net := dyn\_net \cup rec\_set$
  @act3 $node := node \cup dom(rec\_set) \cup ran(rec\_set)$
END
```

Parameter rec_set is a subset of parameter res and all existing links in the original graph (**grd7**) except those failed ones with the following conditions: (i) The union of rec_set and dyn_net does not introduce a loop and preserve the tree structure of the grid (**grd8-9**) (ii) Consumers are either leaves or unreachable (**grd10**), and (iii) The set of consumers that belongs to rec_set are reachable (from $root$) (**grd14**). Parameter $last$ is a subset of high-priority consumers (**grd11**) that are neither reachable in dyn_net nor reachable in $dyn\_net \cup dom(rec\_set)$ (**grd12-13**). Parameter $last$ is an evidence that the recovery is not completed yet. Parameter rec_set is a solution for the power outage problem of a subset of high-priority consumers, represented as i in the event (**grd15**). The set of recovery links are added to the current grid (**act2**) and correspondingly the $node$ is updated by (**act3**).

Recovery_complete event: This event is similar to **Fail_ rec** event, except it is the final stage of recovery and the SG mode changes to $normal$ or $optimal$ depending on the state of lower priority consumer after the event execution. In order to show that it is the last recovery step, we need to add **grd9** that denotes all high-priority consumers are reachable in $dyn\_net \cup recovery\_set$: there is no power outage problem that hits any high-priority consumers. Therefore, the grid mode can switch to $normal$ or $optimal$ and $failed\_link$ and $failed\_path$ update to $emptyset$ (**act2-3**).

```
Recovery_complete extends Recovery_complete
ANY rec_set res WHERE
  @grd5 $\forall r \cdot r \in failed\_link \Rightarrow failed\_path(r) \subseteq res$
  @grd6 $\forall k \cdot k \in res \Rightarrow (\exists i \cdot i \in failed\_link \land failed\_path(i) \subseteq res)$
  @grd7 $rec\_set \subseteq res \cup (net \setminus failed\_link)$
  @grd8 $dyn\_net \cup rec\_set \in$
      $(node \cup dom(rec\_set) \cup ran(rec\_set)) \setminus \{root\} \to (node \cup dom(rec\_set) \cup ran(rec\_set))$
  @grd9 $\forall s \cdot (s \subseteq (dyn\_net \cup rec\_set)^{-1}[s] \Rightarrow s = \varnothing)$
  @grd10 $\forall p \cdot p \in pr1 \Rightarrow p \in (cl(dyn\_net \cup recovery\_set))^{-1}[\{root\}]$
  @grd11 $(node \cup dom(rec\_set) \cup ran(rec\_set)) \subseteq \{root\} \cup (cl(dyn\_net \cup rec\_set))^{-1}[\{root\}]$
THEN
  @act2 $failed\_link := \varnothing$
  @act3 $failed\_path := \varnothing$
  @act4 $node := node \cup dom(rec\_set) \cup ran(re\_set)$
  @act5 $dyn\_net := dyn\_net \cup rec\_set$
END
```

Optimising and **Optimising_complete** events are refined to add links to the grid to connect low priority consumers with the power outage problem provided the newly added links preserve the tree structure of the grid. As the focus of our modeling is on the recovery of high-priority consumers and lower priority consumers, we keep these events consistent to the model and we do not prove that **Optimising** and **Optimising_complete** eventually solve the outage problem of low priority consumers. However, it is observable that the same strategy can be taken to prove properties for SG when it is in the $optimal$ mode.

4.3 The Third Model M_2

In M_1, we have defined the tree structure of the grid and non-deterministically constructed a solution for power outage of high-priority consumers, regardless of its cost. For instance, a solution could be repairing some failed links while there would be other alternatives in the original network graph. In the second model M_1, we have not included any policy for selecting a solution from a set of possible solutions because we intended to construct a reusable correct model. The generic model supports further refinements following particular policies. One basic policy for selecting a recovery solution can be first searching the original network graph for existing links. If there are links in the network graph, those alternatives are used first and then the rest of failures are recovered by repairing the failed links. In the refined model M_2, we specify this policy by restricting the non-determinism of the abstract model M_1.

4.3.1 The Dynamic Part

We add two new variables, which are the partial functions $reconfigure\_link$ and $reestablish\_link$, denoting respectively the links that exist in the network graph and those that need to be repaired (**inv22-23**). During the recovery process the set of links in $reconfigure\_link$ and $reestablish\_link$ are gradually added to dyn_net tree (**inv24-25**). These variables illustrate which links have been taken from the network graph and which ones have been repaired. We split the **Recovery_complete** and **Fail_rec** events in the second model each to two events **Fail_reconfigure** and **Fail_reestablish** which gradually construct the solution regarding the given policy.

VARIABLES
  $reconfigure\_link \quad reestablish\_link$
INVARIANTS
  @inv22 $reconfigure\_link \in NODE \setminus \{root\} \nrightarrow NODE$
  @inv23 $reestablish\_link \in NODE \setminus \{root\} \nrightarrow NODE$
  @inv24 $mode = recovery \Rightarrow reconfigure\_link \subseteq dyn\_net$
  @inv25 $mode = recovery \Rightarrow reestablish\_link \subseteq dyn\_net$
END

Fail_reconfigure event: This event is enabled if there is at least two high-priority consumers with power outage and reachable in the network graph, excluding failed links (**grd1-4**). If there is only one high-priority consumer with power outage and reachable in the network graph, **Fail_reconfigure_complete** event will be enabled. If there is such consumers, one of the possible paths (routes) is computed in the guard of the event (

5 Verification of Models

Deciding from what kind of abstraction to start modelling and what details to add at each refinement step is problem-specific. Often, we take decisions that address modelling/proving complexity and enable reusability of models.

In order to prove that the models satisfy their correctness properties we have to check that they respect their invariants, in our case, the tree properties for SG and the gradual reconstruction of the tree due to node failures. To prove this, we have generated the proof obligations for all the models using the Rodin platform tool. The proof statistics for our models are shown in Table 2. These figures express the number of proof obligations generated by the Rodin platform as well as the number of obligations automatically discharged by the platform and those interactively proved. A high number of interactive proofs were due to reasoning about set comprehension and unions, not currently supported automatically in Rodin. In addition, the interactive proving often involved manually suggesting values to discharging various properties containing logical disjunctions or existential quantifiers. Extra proving was also due to the fact that currently, we cannot create proof scripts and reuse them whenever needed in Rodin. Thus, in some cases we had to manually repeat very similar or almost identical proofs.

Table 2: Proof Statistics

Model	Number of Proof Obligations	Automatically Discharged	Interactively Discharged
Context	19	11	8
M_0 Model	41	30	11
M_1 Model	112	83	29
M_2 Model	76	54	22
Total	248	178	70

One of the most essential requirements for proving the correctness of the recovery mechanism is to show that each step of the recovery reduces the number of disconnected high priority consumers. To prove this, we define the recovery events as $convergent$ [2] and introduce a numeric $variant$. The Rodin platform generates a variant proof obligation ensuring that each convergent event decreases the proposed numeric variant. The other essential requirements are to construct a correct tree: each step of the recovery preserves the tree structure of dyn_net. For instance, to prove the preservation of the invariant **inv11**, stating that all consumers which are in set $node$ are leaves, for the event **Fail_rec** of model M_1, we have distributed union and intersection throughout the generated proof obligation. The distribution allowed us to split the proof obligation into three simple ones:

$$(1)\ consumer\ \cap\ node\ \cap (ran(dyn\_net)\ \cup ran(rec\_set)) = \varnothing$$
$$(2)\ consumer\ \cap\ dom(rec\_set)\ \cap (ran(dyn\_net)\ \cup ran(rec\_set)) = \varnothing$$
$$(3)\ consumer\ \cap\ dom(rec\_set)\ \cap (ran(dyn\_net)\ \cup ran(rec\_set)) = \varnothing$$

To prove these proof obligations, we applied case distinction for the parameter rec_set. When links in rec_set were corresponding links in the range of $failed\_path$, we could not prove the proof obligation. This was due to a missing invariant: denoting links in range of $failed\_path$ also follows the requirement that consumers are leaves. We added the invariant **inv16** and those discharged branches of proof obligation were proved.

Hence, our most important result in this paper can be formulated as follows. If an SG is constructed according to our modelling, then we can recover from link failures by providing alternative paths from generator to high-priority consumers, when there are paths available. There are two important issues to note here. First, if there are no available paths to choose from, then, obviously, we cannot provide any. However, in this case, we save (in $failed\_path$) the subtrees corresponding to the failed links and at least human resources could be dispatched to repair them. Second, our modelling is fundamentally based on the momentary SG having a tree structure. We can recover from failure because we have the consumers as leaves in this tree, i.e., they are not distributing electricity further. They might have this capability in smart grids, and the available SG graph may have them as substations. But in the momentary SG tree, they are leaves. This illustrates the strength of abstraction (and proof) in modelling.

6 Related work

The area of failure identification, diagnosis and recovery in smart grids has been extensively studied. However, until now the energy distribution and communication related problems have been analysed separately. There has been less research on the interdependency between smart grids and mobile communication networks in large-scale fault scenarios. Clark and Pavlovski have presented their experiences as a case study in deploying a cellular wireless solution to support smart grid solutions [10]. Gao et al. have conducted a systematic review of communication technologies in smart grids [16]. In [19] and [20], a system-level simulation model including interdependencies between electricity distribution and mobile communication networks has been presented and evaluated with field trials. The method proposed here can have significant added value to system level simulations to enable better adaptation and proactive fault management in smart grids and future telecommunication systems. In addition, this could offer new crisis recovery planning tools for public authorities, distribution system operators (DSOs), and mobile network operators. The method proposed here is of specific interest, for instance, to manufacturers of automated switch systems [31].

The formal aspects of electrical grids have also been studied before, for instance by Calderaro et al: they captured the details of modelling a protection system for the grid, using Petri Nets [9]. Probabilistic graphical models for modelling spatially correlated data from phasor measurement units, as well as using statistical hypothesis testing for fault diagnosis, was proposed by He and Zhang [18]. Apart from electrical grids, other types of distributed networks have also previously been formally modelled, for instance sensor-actor networks [22], peer-to-peer networks [28, 32], and network recovery [21].

The general formal model proposed in this article addresses the safety and recoverability analysis of many of the advanced properties of smart grids as discussed for instance by Moslehi et al [27]. It can be extended to cover the main aspects of smart grids, as discussed by Fang et al [13], and it should be noted that the proposed model could be refined further to take into account various load profiles [30] and even personalised electricity use information [29], to provide further adaptation capabilities against failures and power outages. The proposed method is of high interest also to smart grid (communication) capacity planning, as discussed for instance by Luan et al [26]. The plan must take into account various modes of operation. Luan et al discussed two modes, or scenarios, which they called “Blue Sky Day” and “Storm Day”. Such scenario based planning can be verified and proven correct with the proposed method.

7 Conclusions

In this paper we have illustrated the use of formal methods, notably the abstraction and refinement techniques, in modelling recovery in smart electrical grids. This is useful from several points of view. First, there is no report of *proving* this kind of properties in related literature, hence our methodology proposes a novel view on addressing these problems. Second, our modelling is intended for reusability and can be extended in several directions. For instance, consumers could be divided into more than the two classes of priority that we have modelled. However, by abstracting away details we have captured the basic problem of addressing first the high-priority consumers and the rest of the consumers could now be modelled by partitioning $pr2$ consumers into other sets. Also, our link failures are non-deterministic; however, we might need to simulate particular failures to see what happens, if alternative paths are found. For instance, this is useful when a storm is forecasted and the SG operators need to ensure that their high-priority consumers are safe. To model this, we only need to refine the non-deterministic choice with a particular choice and our models continue to work. Moreover, when we have several recovery paths to choose from, some may be preferred with respect to various criteria; in that case, the nondeterministic choice in M_2 can be refined.

We have employed Event-B as our formal method due to its integrated support of abstraction and refinement; also, the tool support from the Rodin platform is very useful and can help for an easier acceptance of this methodology in industry. An interesting aspect of Event-B is that the context modelling the constants, sets and axioms on them needs, in principle, no proofs. However, for proving recovery as an invariant of our modelling, we needed to ensure the tree structure of the momentary SG, and so in M_1 we have part of the axioms proved, to ensure consistency of our context. This makes for a stronger model and it was largely facilitated due to the

Event-B tool support.

In our model we have only considered a binary situation, i.e., there is a failure or not. In reality, this type of failure is called blackout; there is at least another type of failure, called brownout, referring to a situation where the power is not completely disconnected but the voltage drops below acceptable levels. Furthermore, brownouts can be intentional, whereby dropping the voltage overall can prevent a blackout in some part of the grid. As we have not modelled this situation in our work, it remains a future research topic.

Acknowledgement

This research is based on the cooperation between the Academy of Finland FResCo project (grant numbers 264060 and 263925, “FResCo: High-quality Measurement Infrastructure for Future Resilient Control Systems”) and the Tekes/SHOK Smart Grids and Energy Markets (SGEM) programme.

References

[1] Abrial, J.R.: The B-Book: Assigning Programs to Meanings. Cambridge University Press (1996)

[2] Abrial, J.R.: Modeling in Event-B: System and Software Engineering. Cambridge University Press (2010)

[3] Abrial, J.R., Butler, M., Hallerstede, S., Hoang, T.S., Mehta, F., Voisin, L.: Rodin: An Open Toolset for Modelling and Reasoning in Event-B. International Journal on Software Tools for Technology Transfer (STTT) 6, 447–466 (2010)

[4] Back, R., Kurki-Suonio, R.: Decentralization of process nets with centralized control. In: Proceedings of the 2nd ACM SIGACT-SIGOPS Symposium on Principles of Distributed Computing. pp. 131–142 (1983)

[5] Back, R., Sere, K.: Stepwise Refinement of Action Systems. Structured Programming 12(1), 17–30 (1991)

[6] Back, R., Sere, K.: From Modular Systems to Action Systems. Software - Concepts and Tools 13, 26–39 (1996)

[7] Back, R., Sere, K.: Superposition Refinement of Reactive Systems. Formal Asp. Comput. 8(3), 324–346 (1996)

[8] Bryans, J., Wei, W.: Formal Analysis of BPMN Models Using Event-B. In: Kowalewski, S., Roveri, M. (eds.) Proceedings of the 15th International

Workshop on Formal Methods for Industrial Critical Systems, FMICS 2010. Lecture Notes in Computer Science, vol. 6371, pp. 33–49. Springer-Verlag (2010)

[9] Calderaro, V., Hadjicostis, C.N., Piccolo, A., Siano, P.: Failure identification in smart grids based on Petri Net modeling. IEEE Transactions on Industrial Electronics 58(10), 4613–4623 (2011)

[10] Clark, A., Pavlovski, C.: Wireless Networks for the Smart Energy Grid: Application Aware Networks. In: Proceedings of the International Multi Conference of Engineers and Computer Scientists. vol. 2 (2010)

[11] Craigen, D., Gerhart, S., Ralson, T.: Case Study: Paris Metro Signaling System. In: Proceedings of IEEE Software. pp. 32–35. IEEE (1994)

[12] Event-B and the Rodin Platform. http://www.event-b.org/ (Accessed November 2013)

[13] Fang, X., Misra, S., Xue, G., Yang, D.: Smart grid - the new and improved power grid: A survey. IEEE Communications Surveys and Tutorials 14(4), 944–980 (2012)

[14] Farhangi, H.: The path of the smart grid. IEEE Power & Energy Mag. 8(1), 18–28 (2010)

[15] Fathabadi, A.S., Rezazadeh, A., Butler, M.: Applying Atomicity and Model Decomposition to a Space Craft System in Event-B. In: Proceedings of the THIRD NASA FORMAL METHODS SYMPOSIUM. Lecture Notes in Computer Science, vol. 6671, pp. 328–342 (2011)

[16] Gao, J., Xiao, Y., Liu, J., Liang, W., Chen, C.: A survey of communication/networking in Smart Grids. Future Generation Computer Systems 28(2), 391–404 (February 2012)

[17] Gharavi, H., Ghafurian, R.: Smart grid: The electric energy system of the future. Proc. IEEE 99(6), 917–921 (2011)

[18] He, M., Zhang, J.: Fault detection and localization in smart grid: A probabilistic dependence graph approach. In: IEEE SmartGridComm '10. pp. 43–48 (2010)

[19] Horsmanheimo, S., Maskey, N., Kokkoniemi-Tarkkanen, H., Savolainen, P., Tuomimäki, L.: A Tool for Assessing Interdependency of Mobile Communication and Electricity-distribution Network. In: IEEE International Conference on Smart Grid Communications, SmartGridComm (October 2013)

[20] Horsmanheimo, S., Maskey, N., Kokkoniemi-Tarkkanen, H., Savolainen, P., Tuomimäki, L.: Evaluation of Interdependencies between Mobile Communication and Electricity Distribution Networks in Fault Scenarios. In: IEEE PES Innovative Smart Grid Technologies Conference Asia (ISGT Asia 2013) (November 2013)

[21] Kamali, M., Laibinis, L., Petre, L., Sere, K.: A Distributed Design of a Network Recovery Algorithm. International Journal of Critical Computer-Based Systems 4(1), 45–68 (2013)

[22] Kamali, M., Laibinis, L., Petre, L., Sere, K.: Formal development of wireless sensor-actor networks. Science of Computer Programming 80, Part A(0), 25 – 49 (2014)

[23] Katz, S.: A superimposition control construct for distributed systems. ACM Transactions on Programming Languages and Systems 15(2), 337–356 (April 1993)

[24] LaCommare, K., Eto, J.: Cost of power interruptions to electricity consumers in the United States. Tech. rep., Ernest Orlando Lawrence Berkeley National Laboratory (February 2006), lBNL-58164

[25] Lecomte, T.: Applying a Formal Method in Industry: A 15-Year Trajectory. In: Alpuente, M., Cook, B., Joubert, C. (eds.) Proceedings of the Formal Methods for Industrial Critical Systems, FMICS 2009. Lecture Notes in Computer Science, vol. 5825, pp. 26–34. Springer-Verlag (2009)

[26] Luan, W., Sharp, D., Lancashire, S.: Smart grid communication network capacity planning for power utilities. In: Proceedings of Transmission and Distribution Conference and Exposition. pp. 1–4. IEEE (2010)

[27] Moslehi, K., Kumar, R.: A Reliability Perspective of the Smart Grid. IEEE Transaction on Smart Grid 1(1), 57–64 (June 2010)

[28] Petre, L., Sandvik, P., Sere, K.: Node Coordination in Peer-to-Peer Networks. In: Sirjani, M. (ed.) Proceedings of the 14th International Conference on Coordination Models and Languages (COORDINATION 2012). Lecture Notes in Computer Science, vol. 7274, pp. 196–211. Springer (2012)

[29] Räsänen, T., Ruuskanen, J., Kolehmainen, M.: Reducing energy consumption by using self-organizing maps to create more personalized electricity use information. APPLIED ENERGY 85(9), 830–840 (2008)

[30] Räsänen, T., Voukantsis, D., Niska, H., Karatzas, K., Kolehmainen, M.: Data-based method for creating electricity use load profiles using large amount of customer-specific hourly measured electricity use data. APPLIED ENERGY 87(11), 3538–3545 (2010)

[31] Reid, B.: Oncor Electric Delivery Smart Grid initiative. In: Proceedings of Protective Relay Engineers. pp. 8–15. IEEE (2009)

[32] Sandvik, P., Sere, K.: Formal Analysis and Verification of Peer-to-Peer Node Behaviour. In: Proceedings of AP2PS '11 (November 2011)

[33] Tate, J., Overbye, T.: Line Outage Detection Using Phasor Angle Measurements. Power Systems, IEEE Transactions on 23(4), 1644–1652 (November 2008)

[34] Tate, J., Overbye, T.: Double line outage detection using phasor angle measurements. In: Power & Energy Society General Meeting, 2009. PES '09. pp. 1–5. IEEE (July 2009)

[35] Waldén, M., Sere, K.: Reasoning About Action Systems Using the B-Method. Formal Methods in Systems Design 13, 5–35 (1998)

[36] Woodcock, J., Larsen, P.G., Bicarregui, J., Fitzgerald, J.: Formal Methods: Practice and Experience. ACM Computing Surveys 41(4), 1–36 (2009)

[37] Zhu, J., Abur, A.: Improvements in Network Parameter Error Identification via Synchronized Phasors. Power Systems, IEEE Transactions on 25(1), 44–50 (February 2010)

ISBN 978-952-12-2991-6
ISSN 1239-1891