Incremental system modelling in Event-B*

Stefan Hallerstede

University of Düsseldorf
Germany
halstefa@cs.uni-duesseldorf.de

Abstract. A reasonable approach to formal modelling is to start with a specification that captures the requirements of a system and then use formal refinement to implement it.
The problem with this approach is that for complex systems the specification itself is complex. It becomes a challenge to say whether the specification is the right one for the given requirements. Sometimes requirements also concern features of a system closely related to its implementation. This would make an abstract specification necessarily incomplete.
We believe that it is better not to follow the rigid approach to modelling described above. Instead, we argue that the specification itself should be elaborated by refinement. Ultimately, the distinction between specification and implementation is no longer made in the strict sense above. There is only one model of the system that is connected by successive refinements. Using Event-B, we demonstrate how this can be applied to cope with the complexity of specifications. On the one hand we benefit from the reduced number of detail to consider at different times. On the other hand we are encouraged to reason about the formal model since the beginning and to rethink it occasionally to capture better its intended behaviour and match the requirements.

1 Introduction

When we create a complex model, usually, our understanding of it is incomplete at first; and a modelling method should help to improve our understanding of the model. During initial phases in the modelling process, we use refinement to manage the many details of a complex model. Refinement is seen as a technique to introduce detail gradually at a rate that eases understanding. We do not assume that we have one most abstract model, the specification, that could serve as point of reference for all further refinements. Instead, the model is completed by refinement until we are satisfied that the model captures all important requirements and assumptions. In this article we concern ourselves only with what is involved in coming up with an abstract model of some system. Refinement can also be used to produce implementations of abstract models, for instance, in terms of a sequential program [1,16]. But this is not discussed in this article.

Event-B [2] is a formal modelling method for discrete systems based on refinement [4,5,6]. Event-B and its predecessor, the B Method, have been used in large scale

* This research was carried out as part of the EU research project DEPLOY (Industrial deployment of system engineering methods providing high dependability and productivity) http://www.deploy-project.eu/.

industrial projects [7,8,19]. In Event-B formal modelling serves primarily for reasoning: reasoning is an essential part of modelling because it is the key to understanding complex models. Reasoning about complex models should not happen accidentally but needs systematic support within the modelling method. This thinking lies at the heart of the Event-B method. It gives a prominent rôle to proof obligations. Proof obligations serve to reason about a model and to provide meaning [11].

We briefly contrast the incremental refinement-based modelling approach to two well-known approaches, TLA+ [14] and ASM [9]. In TLA+ modelling begins with a *specification*, in ASM with a *ground model*. In both methods there are guidelines on how to begin. This is a difficult and serious problem as an inadequate model may not provide the kinds of insight we seek or may be plainly wrong (as a starting point for an implementation). Among the hints on how to begin we find: (1) choose the appropriate abstraction level; (2) be as abstract as possible yet complete; (3) be simple and concise. This article addresses (1) and (2) by effectively avoiding the precise choice of an initial abstraction level. We begin with some simple abstract model and introduce detail gradually. Guideline (3) remains, being the key to comprehensibility.

We understand incremental modelling in two ways. The first way is by formal refinement. An existing model is proved to be refined by another: all properties of the existing model are preserved in the refined model. The second way is by alteration of an existing model: properties of the existing model may be broken. When a model is shown to be not consistent, it needs to be modified in order to make it consistent. This reflects a learning process supported by various forms of reasoning about a model, for instance, proof, animation, or model-checking. This way of thinking about a model is common in mathematical methodology [13,17,18]. The first way is commonly used in formal methods, whereas the second is at least not acknowledged. We believe both ways are crucial for formal modelling.

The incremental approach is only feasible in the presence of software tools that make reasoning easy and modifications to a model painless. We have relied on the Rodin modelling tool [3] for Event-B for proof obligation generation and proof support and on the ProB tool [15] for animation and model-checking. Both tools are integrated in the Rodin platform and can be used seamlessly. In later sections we do not further specify the tools used, though, as this should be clear from the context. Also note that we present proof in an equational style [10,20] whereas the Rodin tool uses sequents as in [2].

Overview. In Section 2 we introduce Event-B. The following sections are devoted to solving a concrete problem in Event-B. In Section 3 the problem is stated. A first model is produced and discussed in Section 4. In Sections 5 and 7 we elaborate the model by refinement. Section 6 contains a small theory of transitive closures that is needed in the refinement. In Section 8 some further improvements of the model are made and limitations of formal modelling discussed.

2 Event-B

Event-B models are described in terms of the two basic constructs: *contexts* and *machines*. Contexts contain the static part of a model whereas machines contain the dy-

namic part. Contexts may contain *carrier sets*, *constants*, *axioms*, where carrier sets are similar to types [4]. In this article, we simply assume that there is some context and do not mention it explicitly. Machines are presented in Section 2.1, and proof obligations in Section 2.2 and Section 2.3. All proof obligations in this article are presented in the form of sequents: "premises" $\vdash$ "conclusion".

For the purpose of this article, we have reduced the Event-B notation used so that only a little notation suffices and formulas are easier to comprehend, in particular, concerning the relationship between formal model and proof obligations. We have also reduced the amount of proof obligations associated with a model. We have done this for two reasons: firstly, it is easier to keep track of what is to be proved; secondly, it permits us to make a point about a limitation of formal methods later on.

2.1 Machines

Machines provide behavioural properties of Event-B models. Machines may contain *variables*, *invariants*, *theorems*, *events*, and *variants*. Variables $v = v_1, \ldots, v_m$ define the state of a machine. They are constrained by invariants $I(v)$. Theorems are predicates that are implied by the invariants. Possible state changes are described by means of events $E(v)$. Each event is composed of a *guard* $G(t, v)$ and an *action* $x := S(t, v)$, where $t = t_1, \ldots, t_r$ are *parameters* the event may contain and $x = x_1, \ldots, x_p$ are the (distinct) variables it may change[1]. The guard states the necessary condition under which an event may occur, and the action describes how the state variables evolve when the event occurs. We denote an event $E(v)$ by

$$
\begin{array}{lcl}
E(v) & \widehat{=} & \mathsf{any}\ \ t\ \ \mathsf{when} \\
 & & \quad G(t, v) \\
 & & \mathsf{then} \\
 & & \quad x := S(t, v) \\
 & & \mathsf{end}
\end{array}
\qquad \text{or} \qquad
\begin{array}{lcl}
E(v) & \widehat{=} & \mathsf{begin} \\
 & & \quad x := S(v) \\
 & & \mathsf{end}
\end{array}
$$

The short form on the right hand side is used if the event does not have parameters and the guard is true. A dedicated event of the latter form is used for *initialisation*. All assignments of an action $x := S(t, v)$ occur simultaneously; variables y that do not appear on the left-hand side of an assignment of an action are not changed by the action, yielding one simultaneous assignment

$$x_1, \ldots, x_p, y_1, \ldots, y_q := S_1(t, v), \ldots, S_p(t, v), y_1, \ldots, y_q \quad , \tag{1}$$

where $x_1, \ldots, x_p, y_1, \ldots, y_q$ are the variables v of the machine. The action $x := S(t, v)$ of event $E(v)$ denotes the formula (1), whereas in the proper model we only specify those variables x_ℓ that may change.

[1] Note that, as x is a list of variables, $S(t, v) = S_1(t, v), \ldots, S_p(t, v)$ is a corresponding list of expressions.

2.2 Machine Consistency

Invariants are supposed to hold whenever variable values change. Obviously, this does not hold a priori for any combination of events and invariants $I(v) = I_1(v) \land \ldots \land I_i(v)$ and, thus, needs to be proved. The corresponding proof obligations are called *invariant preservation* ($\ell \in 1 \mathinner{..} i$):

$$\begin{array}{l} I(v) \\ G(t,v) \\ \vdash \\ I_\ell\left(S(t,v)\right) \quad , \end{array} \tag{2}$$

for every event $E(v)$. Similar proof obligations are associated with the initialisation event of a machine. The only difference is that neither an invariant nor a guard appears in the premises of proof obligation (2), that is, the only premises are axioms and theorems of the context. We say that a machine is consistent if all events preserve all invariants.

2.3 Machine Refinement

Machine refinement provides a means to introduce more details about the dynamic properties of a model [4]. The refinement theory of Event-B originates in the Action System formalism [6]. We present some important proof obligations for machine refinement that are used in this article.

A machine N can refine at most one other machine M. We call M the *abstract* machine and N a *concrete* machine. The state of the abstract machine is related to the state of the concrete machine by a *gluing invariant* $J(v,w) = J_1(v,w) \land \ldots \land J_j(v,w)$, where $v = v_1, \ldots, v_m$ are the variables of the abstract machine and $w = w_1, \ldots, w_n$ the variables of the concrete machine.

Each event $E(v)$ of the abstract machine is *refined* by a concrete event $F(w)$. Let abstract event $E(v)$ with parameters $t = t_1, \ldots, t_r$ and concrete event $F(w)$ with parameters $u = u_1, \ldots, u_s$ be

$$\begin{array}{lcl} E(v) & \widehat{=} & \mathsf{any}\ \ t\ \ \mathsf{when} \\ & & \quad G(t,v) \\ & & \mathsf{then} \\ & & \quad v := S(t,v) \\ & & \mathsf{end} \end{array} \qquad \text{and} \qquad \begin{array}{lcl} F(w) & \widehat{=} & \mathsf{any}\ \ u\ \ \mathsf{when} \\ & & \quad H(u,w) \\ & & \mathsf{with} \\ & & \quad t = W(u) \\ & & \mathsf{then} \\ & & \quad w := T(u,w) \\ & & \mathsf{end} \end{array}$$

Informally, concrete event $F(w)$ refines abstract event $E(v)$ if the guard of $F(w)$ is stronger than the guard of $E(v)$, and the gluing invariant $J(v,w)$ establishes a simulation of the action of $F(w)$ by the action of $E(v)$. The corresponding proof obligations are called *guard strengthening* ($\ell \in 1 \mathinner{..} g$):

$$\begin{array}{l} I(v) \\ J(v,w) \\ H(u,w) \\ \vdash \\ G_\ell\left(W(u),v\right) \quad , \end{array} \tag{3}$$

with the abstract guard $G(t,v) \;=\; G_1(t,v) \wedge \ldots \wedge G_g(t,v)$, and (again) *invariant preservation* ($\ell \in 1\,..\,j$):

$$\begin{array}{l} \quad I(v) \\ \quad J(v,w) \\ \quad H(u,w) \\ \vdash \\ \quad J_\ell\,(S(W(u),v),T(u,w)) \quad . \end{array} \tag{4}$$

The term $W(u)$ denotes *witnesses* for the abstract parameters t, specified by the equation $t \;=\; W(u)$ in event $F(w)$, linking abstract parameters to concrete parameters. It describes how $F(w)$ refines $E(v)$ just as the gluing invariant describes how concrete machine N refines abstract machine M[2].

(The variable lists v and w do not need to be disjoint. If a variable name gets reused in a refined machine, equality between the abstract and the concrete variable is postulated implicitly, corresponding to a gluing invariant "$v_{abs} \;=\; v_{con}$". Similarly, equality for common parameters of abstract and concrete events is postulated with a witness "$t_{abs} \;=\; t_{con}$".)

3 Problem Statement

In the following sections we develop a simple model of a secure building equipped with access control. The problem statement is inspired by a similar problem used by Abrial [2]. Instead of presenting a fully developed model, we illustrate the process of how we arrive at the model. We can not follow the exact path that we took when working on the model: we made changes to the model as a whole several times. So we would soon run out of space. We comment on some of the changes without going too much into detail in the hope to convey some of the dynamic character of the modelling process.

The model to be developed is to satisfy the following properties:

P1 : The system consists of persons and one building.
P2 : The building consists of rooms and doors.
P3 : Each person can be at most in one room.
P4 : Each person is authorised to be in certain rooms (but not others).
P5 : Each person is authorised to use certain doors (but not others).
P6 : Each person can only be in a room where the person is authorised to be.
P7 : Each person must be able to leave the building from any room where the person is authorised to be.
P8 : Each person can pass from one room to another if there is a door connecting the two rooms and the person has the proper authorisation.
P9 : Authorisations can be granted and revoked.

Properties P1, P2, P8, and P9 describe environment assumptions whereas properties P3, P4, P5, P6, and P7 describe genuine requirements. It is natural to mix them in the description of the system. Once we start modelling, the distinction becomes important. We have to prove that our model satisfies P3, P4, P5, P6, and P7 assuming we have P1, P2, P8, and P9.

[2] In full Event-B, instead of an equation $t \;=\; W(u)$ a witness can be any predicate. It can also have more free variables than just the abstract parameters u.

4 Getting started with a fresh model

Our aim is to produce a faithful formal model of the system described by the properties P1 to P9 of Section 3. The first decision we need to make concerns the use of refinement. We have decided to introduce the properties of the system in two steps. In the first step we deal only with persons and rooms, in the second also with doors. This approach appears reasonable. At first we let persons move directly between rooms. Later we state how they do it, that is, by passing through doors. In order to specify doors we need to know about rooms they connect. It is a good idea, though, to reconsider the strategy chosen for refinement when it turns out to be difficult to tackle the elements of the model in the planned order. For now, we intend to produce a model with one refinement:

(i) the *abstract machine* (this section) models room authorisations;
(ii) the *concrete machine* (sections 5 and 7) models room and door authorisations.

In Event-B we usually begin modelling by stating invariants that a machine should preserve. (For an alternative approach see, for instance, [12].) When events are introduced subsequently, we think more about how they preserve invariants than about what they would do. The focus is on the properties that have to be satisfied. We declare two carrier sets for persons and rooms, $Person$ and $Room$, and a constant $\boldsymbol{O}$, where $\boldsymbol{O} \in Room$, modelling the outside. We choose to describe the state by two variables for authorised rooms and locations of persons, arm and loc, with invariants

$inv1 : arm \in Person \leftrightarrow Room$ [3]	Property P4
$inv2 : Person \times \{\boldsymbol{O}\} \subseteq arm$	
$inv3 : loc \in Person \to Room$	Property P3
$inv4 : loc \subseteq arm$	Property P6

Invariant $inv2$, that *each person is authorised to be outside*, is necessary because we decided to model location by a total function making the outside a special room. In a first attempt, we made loc a partial function from $Person$ to $Room$ expressing that a person not in the domain of loc is outside. However, this turned out to complicate the gluing invariant when introducing doors into the model later on. (Because of property P7 we need an explicit representation of the outside in the model.) As a consequence of our decision we had to introduce invariant $inv2$. It corresponds to a new requirement that is missing from the list in Section 3 but that we have uncovered while reasoning formally about the system. In the following we focus on how formal reasoning is used to improve the model of the system.

In order to satisfy $inv2$, $inv3$ and $inv4$ we let

initialisation
  begin
    $act1 : arm := Person \times \{\boldsymbol{O}\}$
    $act2 : loc := Person \times \{\boldsymbol{O}\}$
  end

[3] The term $A \leftrightarrow B$ denotes the set of relations from A to B.

We model passage from one room to another by event $pass$,

$pass$
  any p, r when
    $grd1 : p \mapsto r \in arm$ — p is authorised to be in r
    $grd2 : p \mapsto r \notin loc$ — but not already in r
  then
    $act1 : loc := loc \lhd\!\!+ \{p \mapsto r\}$
  end

(For relations a and b relational overwriting $\lhd\!\!+$ is defined by $a \lhd\!\!+ b = \mathsf{dom}(b) \lhd a \cup b$, and, for a set s, domain subtraction $\lhd$ by $x \mapsto y \in s \lhd a \Leftrightarrow x \notin s \wedge x \mapsto y \in a$.) Event $pass$ preserves the invariants. For instance, it preserves $inv4$:

$loc \subseteq arm$	*Invariant* $inv4$
$p \mapsto r \in arm$	*Guard* $grd1$
$p \mapsto r \notin loc$	*Guard* $grd2$
$\vdash$	
$loc \lhd\!\!+ \{p \mapsto r\} \subseteq arm$	*Modified invariant* $inv4$

We prove,

	$loc \lhd\!\!+ \{p \mapsto r\}$	{ def. of $\lhd\!\!+$ }
$=$	$\{p\} \lhd loc \cup \{p \mapsto r\}$	{ $\{p\} \lhd loc \subseteq loc$ }
$\subseteq$	$loc \cup \{p \mapsto r\}$	{ $inv4$ and $grd1$ }
$\subseteq$	arm .	

Granting and revoking authorisations for rooms is modelled by the two events

$grant$
  any p, r when
    $grd1 : p \in Person$
    $grd2 : r \in Room$
  then
    $act1 : arm := arm \cup \{p \mapsto r\}$
  end

$revoke$
  any p, r when
    $grd1 : p \in Person$
    $grd2 : p \mapsto r \notin loc$
  then
    $act1 : arm := arm \setminus \{p \mapsto r\}$
  end

The two events do not yet model all of P9 which refers to authorisations in general, including authorisations for doors. Events $grant$ and $revoke$ appear easy enough to get them right. But it is as easy to make a mistake. This is why we have specified invariants: to safeguard us against mistakes. If the proof of an invariant fails, we have the opportunity to learn something about the model and improve it. The two events preserve all invariants except for $revoke$ which violates invariant $inv2$,

$Person \times \{\boldsymbol{O}\} \subseteq arm$	*Invariant* $inv2$
$p \in Person$	*Guard* $grd1$
$p \mapsto r \notin loc$	*Guard* $grd2$
$\vdash$	
$Person \times \{\boldsymbol{O}\} \subseteq arm \setminus \{p \mapsto r\}$	*Modified invariant* $inv2$

In an instance of the model with two different rooms $\boldsymbol{I}$ and $\boldsymbol{O}$ and one person P we find a counter example:

$$arm = \{P \mapsto \boldsymbol{I}, P \mapsto \boldsymbol{O}\}, \quad loc = \{P \mapsto \boldsymbol{I}\}, \quad p = P, \quad r = \boldsymbol{O} \quad .$$

In fact, we must not remove $\boldsymbol{O}$ from the set of authorised rooms of any person. To achieve this, we add a third guard to event $revoke$:

$$grd3 \ : \ r \neq \boldsymbol{O} \quad .$$

A counter example provides valuable information, pointing to a condition that it does not satisfy. It may not always be as simple to generalise but at least one can obtain an indication where to look closer.

The model we have obtained thus far is easy to understand. Ignoring the doors in the building, it is quite simple but already incorporates properties P3, P4, and P6. Its simplicity permits us to judge more readily whether the model is reasonable. We can inspect it or animate it and can expect to get a fairly complete picture of its behaviour. Way may ask: Is it possible to achieve a state where some person can move around in the building? We have only partially modelled the assumptions P1, P2, P8, and P9. We could split them into smaller statements that would be fully modelled but have decided not to do so. Instead, we are going to document how they are incorporated in the refinement that is to follow.

5 Elaboration of more details

We are satisfied with the abstract model of the secure building for now and turn to the refinement where doors are introduced into the model. In the refined model we employ two variables adr for authorised doors and loc for the locations of persons in the building (as before). The intention is to keep the information contained in the abstract variable arm implicitly in the concrete variable adr. That is, in the refined model variable arm would be redundant. We specify

$inv5 \ : \ adr \in Person \rightarrow (Room \leftrightarrow Room)$	Property P5
$inv6 \ : \ \forall q \cdot \mathsf{ran}(adr(q)) \subseteq arm[\{q\}]$ [4]	Property P4

5.1 Moving between rooms

Let us first look at event $pass$. Only a few changes are necessary to model property P8,

$pass$
  any p, r when
    $grd1 \ : \ loc(p) \mapsto r \in adr(p)$
  then
    $act1 \ : \ loc := loc \lhd\!\!+ \{p \mapsto r\}$
  end

[4] The term $R[A]$ denotes the relational image of the set A under the realtion R, that is, $R[A] = \{y \mid \exists x \cdot x \in A \wedge x \mapsto y \in R\}$.

We only have to show guard strengthening, because loc does not occur in $inv5$ and $inv6$. For the abstract guard $grd1$ we have to show:

$$\begin{array}{ll} & \forall\, q \cdot \mathsf{ran}(adr(q)) \subseteq arm[\{q\}] & \textit{Invariant } inv6 \\ & loc(p) \mapsto r \in adr(p) & \textit{Concrete guard } grd1 \\ \vdash & & \\ & p \mapsto r \in arm & \textit{Abstract guard } grd1 \end{array}$$

which holds because $r \in \mathsf{ran}(adr(p))$. The second guard strengthening proof obligation of event $pass$ is:

$$\begin{array}{ll} & loc \in Person \rightarrow Room & \textit{Invariant } inv3 \\ & loc(p) \mapsto r \in adr(p) & \textit{Concrete guard } grd1 \\ \vdash & & \\ & p \mapsto r \notin loc & \textit{Abstract guard } grd2 \end{array}$$

Using $inv3$ we can rephrase the goal,

$$\begin{array}{lll} & p \mapsto r \notin loc & \{\ inv3\ \} \\ \Leftrightarrow & loc(p) \neq r & \end{array}$$

Neither concrete guard $grd1$ nor the invariants $inv1$ to $inv6$ imply this. The invariant is too weak. We do not specify that doors connect *different* rooms. In fact, our model of the building is rather weak. We decide to model the building by the doors that connect the rooms in it. They are modelled by a constant $Door$. We make the following three assumptions about doors:

$$\begin{array}{lll} axm1 & : Door \in Room \leftrightarrow Room & \textit{Each door connects two rooms.} \\ axm2 & : Door \cap \mathsf{id}_{Room} = \varnothing & \textit{No door connects a room to itself.} \\ axm3 & : Door \subseteq Door^{-1} & \textit{Each door can be used in both directions.} \end{array}$$

These assumptions are based on our domain knowledge about properties of typical doors. They were omitted from the problem description because they seemed obvious. However, the validity of our model will depend on them. As such they ought to be included. We began to think about properties of doors because we did not succeed proving a guard strengthening proof obligation. If axiom $axm2$ would hold for all relations $adr(p)$, for $p \in Person$, we should succeed. Hence, we add a new invariant $inv7$. We realise that it captures much better property P5 than invariant $inv5$,

$$inv7 \ : \forall\, q \cdot adr(q) \subseteq Door \quad . \qquad \text{Property P5}$$

We prove,

$$\begin{array}{lll} & x \mapsto y \in adr(p) & \{\ inv7 \text{ with } \text{“}q := p\text{”}\ \} \\ \Rightarrow & x \mapsto y \in Door & \{\ axm2\ \} \\ \Rightarrow & x \mapsto y \notin \mathsf{id}_{Room} & \{\ \text{def. of } \mathsf{id}_{Room}\ \} \\ \Leftrightarrow & x \neq y \quad , & \end{array}$$

thus, $\forall\, x,\, y \cdot x \mapsto y \in adr(p) \Rightarrow x \neq y$, and with "$x,\, y := loc(p),\, r$" we are able to show:

$$
\begin{array}{ll}
& Door \cap \mathsf{id}_{Room} = \varnothing & \textit{Axiom } axm2 \\
& loc \in Person \rightarrow Room & \textit{Invariant } inv3 \\
& \forall\, q \cdot adr(q) \subseteq Door & \textit{Invariant } inv7 \\
& loc(p) \mapsto r \in adr(p) & \textit{Concrete guard } grd1 \\
\vdash & & \\
& p \mapsto r \notin loc & \textit{Abstract guard } grd2
\end{array}
$$

6 Intermezzo on transitive closures

Property P7 is more involved. It may be necessary to pass though various rooms in order to leave the building. We need to specify a property about the transitive relationship of the doors. We can rely on the well-known mathematical theory of the transitive closure of a relation.

A relation x is called *transitive* if $x\,;x \subseteq x$. In other words, any composition of elements of x is in x. The transitive closure of a relation x is the least relation that contains x and is transitive. We define the *transitive closure* x^+ of a relation x by

$$\forall\, x \cdot x \subseteq x^+ \tag{5}$$
$$\forall\, x \cdot x^+\,;x \subseteq x^+ \tag{6}$$
$$\forall\, x,\, z \cdot x \subseteq z \wedge z\,;x \subseteq z \Rightarrow x^+ \subseteq z \quad . \tag{7}$$

That is, x^+ is the least relation z satisfying $x \cup z\,;x \subseteq z$. Furthermore, the order in which the transitive closure is formed does not matter,

$$\forall\, x \cdot x \cup x^+\,;x = x^+ \tag{8}$$
$$\forall\, x \cdot x \cup x\,;x^+ = x^+ \quad . \tag{9}$$

The transitive closure is monotonic and maps identity and empty relation to themselves,

$$\forall\, x,\, y \cdot x \subseteq y \Rightarrow x^+ \subseteq y^+ \tag{10}$$
$$\forall\, w \cdot \mathsf{id}_w^+ = \mathsf{id}_w \tag{11}$$
$$\varnothing^+ = \varnothing \quad . \tag{12}$$

A relation x is called *symmetric* if $x \subseteq x^{-1}$. For a symmetric relation we can prove more laws about its transitive closure: it is symmetric too and the identity is contained in it,

$$\forall\, x \cdot x \subseteq x^{-1} \Rightarrow (x^+)^{-1} \subseteq x^+ \tag{13}$$
$$\forall\, x \cdot x \subseteq x^{-1} \Rightarrow \mathsf{id}_{\mathsf{dom}(x)} \subseteq x^+ \quad . \tag{14}$$

7 Towards a full model of the building

Using the transitive closure of authorised rooms we can express that every person can at least reach the authorised rooms from the outside,

$$inv8 \ : \forall q \cdot arm[\{q\}] \subseteq adr(q)^+[\{\boldsymbol{O}\}] \quad .$$

This invariant is weaker than property P7. However, given the discussion about properties of doors in Section 5 we should be able to prove that all invariants jointly imply property P7 which we formalise as a theorem,

$$thm1 \ : \forall q \cdot (arm[\{q\}] \setminus \{\boldsymbol{O}\}) \times \{\boldsymbol{O}\} \subseteq adr(q)^+ \quad . \qquad \text{Property P7}$$

We proceed like this because we expect that proving $inv8$ to be preserved would be much easier than doing the same with $thm1$. Let us continue working with $inv8$ for now and return to $thm1$ later.

7.1 Initialisation

In the abstract model all persons can only be outside initially. This corresponds to them not being authorised to use any doors,

initialisation
  begin
    $act1 \ : \ adr := Person \times \{\varnothing\}$
    $act2 \ : \ loc := Person \times \{\boldsymbol{O}\}$
  end

The invariant preservation proof obligations for $inv5$ and $inv6$ hold, as can easily be seen letting "$arm, \ adr := Person \times \{\boldsymbol{O}\}, \ Person \times \{\varnothing\}$" in $inv5$, $inv6$, and $inv7$,

$$\vdash \quad Person \times \{\varnothing\} \in Person \rightarrow (Room \leftrightarrow Room)$$
$$\vdash \quad \forall q \cdot \mathsf{ran}((Person \times \{\varnothing\})(q)) \subseteq (Person \times \{\boldsymbol{O}\})[\{q\}]$$
$$\vdash \quad \forall q \cdot (Person \times \{\varnothing\})(q) \subseteq Door$$

For invariant $inv8$ there is more work to do. We have to show:

$$\vdash \quad \forall q \cdot (Person \times \{\boldsymbol{O}\})[\{q\}] \subseteq (Person \times \{\varnothing\})(q)^+[\{\boldsymbol{O}\}]$$

We prove,

$$\begin{array}{lll} & (Person \times \{\varnothing\})(q)^+[\{\boldsymbol{O}\}] & \{ \text{ set theory } \} \\ = & \varnothing^+[\{\boldsymbol{O}\}] & \{ \text{ law (12) } \} \\ = & \varnothing[\{\boldsymbol{O}\}] & \{ \text{ set theory } \} \\ = & \varnothing & \\ \not\supseteq & \{\boldsymbol{O}\} & \{ \text{ set theory } \} \end{array}$$

$$= \quad (Person \times \{\boldsymbol{O}\})[\{q\}] \quad .$$

Invariant $inv8$ is too strong! Because of invariant $inv7$ we cannot initialise adr to $Person \times \{\{\boldsymbol{O} \mapsto \boldsymbol{O}\}\}$ and because of $inv6$ we cannot use any other door. Thus, we must weaken invariant $inv8$. We replace it by:

$$inv8' \ : \forall q \cdot arm[\{q\}] \subseteq adr(q)^+[\{\boldsymbol{O}\}] \cup \{\boldsymbol{O}\}$$

After analysing initialisation and event $pass$ of the refined machine, the gluing invariants of the refined machine have become

$$\begin{array}{ll} inv5 & : adr \in Person \rightarrow (Room \leftrightarrow Room) \\ inv6 & : \forall q \cdot \mathsf{ran}(adr(q)) \subseteq arm[\{q\}] \\ inv7 & : \forall q \cdot adr(q) \subseteq Door \\ inv8' & : \forall q \cdot arm[\{q\}] \subseteq adr(q)^+[\{\boldsymbol{O}\}] \cup \{\boldsymbol{O}\} \quad . \end{array}$$

7.2 Granting door authorisations

A new door authorisation can be granted to a person if (a) it has not been granted yet and (b) authorisation for one of the connected rooms has been granted to the person. We introduce constraint (a) to focus on the interesting case and constraint (b) to satisfy invariant $inv8'$. Thus,

$grant$
  any p, s, r when
    $grd1 : s \mapsto r \notin adr(p)$
    $grd2 : s \in \mathsf{dom}(adr(p))$
  then
    $act1 : adr := adr \lhd\!\!+ \{p \mapsto adr(p) \cup \{s \mapsto r, r \mapsto s\}\}$ [5]
  end

Invariant $inv5$ is preserved by event $grant$ by definition of relational overwriting $\lhd\!\!+$. For invariant $inv6$ we have to prove:

$$\begin{array}{lll} & \forall q \cdot \mathsf{ran}(adr(q)) \subseteq arm[\{q\}] & \textit{Invariant } inv6 \\ & s \mapsto r \notin adr(p) & \textit{Concrete guard } grd1 \\ & s \in \mathsf{dom}(adr(p)) & \textit{Concrete guard } grd2 \\ \vdash & \mathsf{ran}((adr \lhd\!\!+ \{p \mapsto adr(p) \cup \{s \mapsto r, r \mapsto s\}\})(q)) & \\ & \quad \subseteq (arm \cup \{p \mapsto r\})[\{q\}] & \textit{Modified invariant } inv6 \end{array}$$

for all q. For $q \neq p$ the proof is easy. For the other case $q = p$ we prove, letting $D = \{s \mapsto r, r \mapsto s\}$,

$$\mathsf{ran}(adr(p) \cup D) \subseteq (arm \cup \{p \mapsto r\})[\{p\}] \qquad \{ \text{ set theory, def. of } D \ \}$$

[5] Event-B has the shorter (and more legible) notation $adr(p) := adr(p) \cup \{s \mapsto r, r \mapsto s\}$ for this. We do not use it because we can use the formula above directly in proof obligations. We also try as much as possible to avoid introducing more notation than necessary.

$$
\begin{array}{lll}
\Leftrightarrow & \mathsf{ran}(adr(p)) \cup \{r,\, s\} \subseteq arm[\{p\}] \cup \{r\} & \{\ inv6 \text{ with ``}q := p\text{''}\ \} \\
\Leftrightarrow & \{r,\, s\} \subseteq arm[\{p\}] \cup \{r\} & \{\ \{r\} \subseteq \{r\}\ \} \\
\Leftrightarrow & \{s\} \subseteq arm[\{p\}] \cup \{r\} & \{\ \text{set theory}\ \} \\
\Leftrightarrow & s \in arm[\{p\}] \ \vee\ s = r & \{\ inv6 \text{ with ``}q := p\text{''}\ \} \\
\Leftarrow & s \in \mathsf{ran}(adr(p)) &
\end{array}
$$

We would expect $s \in \mathsf{ran}(adr(p))$ to hold because doors are symmetric and because of concrete guard $grd2$, that is, $s \in \mathsf{dom}(adr(p))$. We specified symmetry in axiom $axm3$ but this property is not covered by invariant $inv7$. We have to specify it explicitly,

$$
inv9 \ : \ \forall q \cdot adr(q) \subseteq adr(q)^{-1} \quad . \qquad \text{(see axiom } axm3\text{)}
$$

We can continue the proof where we left off

$$
\begin{array}{lll}
 & s \in \mathsf{ran}(adr(p)) & \{\ inv9 \text{ with ``}q := p\text{''}\ \} \\
\Leftarrow & s \in \mathsf{dom}(adr(p)) &
\end{array}
$$

It is easy to show that invariant $inv9$ itself is preserved by event $grant$:

$$
\begin{array}{ll}
 & \forall q \cdot adr(q) \subseteq adr(q)^{-1} \qquad \textit{Invariant } inv9 \\
 & s \mapsto r \notin adr(p) \qquad \textit{Concrete guard } grd1 \\
 & s \in \mathsf{dom}(adr(p)) \qquad \textit{Concrete guard } grd2 \\
\vdash & (adr \lhd\!\!+ \{p \mapsto adr(p) \cup \{s \mapsto r,\, r \mapsto s\}\})(q) \qquad \textit{Modified invariant } inv9 \\
 & \quad \subseteq (adr \lhd\!\!+ \{p \mapsto adr(p) \cup \{s \mapsto r,\, r \mapsto s\}\})(q)^{-1}
\end{array}
$$

for all q. Let $D = \{s \mapsto r,\, r \mapsto s\}$. The interesting case is $q = p$ as above,

$$
\begin{array}{lll}
 & adr(p) \cup D & \{\ inv9 \text{ with ``}q := p\text{''}\ \} \\
\subseteq & adr(p)^{-1} \cup D & \{\ D^{-1} = D\ \} \\
= & adr(p)^{-1} \cup D^{-1} & \{\ \text{set theory}\ \} \\
= & (adr(p) \cup D)^{-1} \quad . &
\end{array}
$$

Invariants $inv8'$ and $inv7$ remain to be analysed. We begin with the proof obligation for the preservation of invariant $inv7$:

$$
\begin{array}{ll}
 & \forall q \cdot adr(q) \subseteq Door \qquad \textit{Invariant } inv7 \\
 & s \mapsto r \notin adr(p) \qquad \textit{Concrete guard } grd1 \\
 & s \in \mathsf{dom}(adr(p)) \qquad \textit{Concrete guard } grd2 \\
\vdash & (adr \lhd\!\!+ \{p \mapsto adr(p) \cup \{s \mapsto r,\, r \mapsto s\}\})(q) \\
 & \quad \subseteq Door \qquad \textit{Modified invariant } inv7
\end{array}
$$

for all q. For $q = p$,

$$
adr(p) \cup \{s \mapsto r,\, r \mapsto s\} \subseteq Door \qquad \{\ inv7 \text{ and set theory}\ \}
$$

$$
\begin{array}{lll}
\Leftarrow & \{s \mapsto r,\, r \mapsto s\} \subseteq Door & \{\ inv9 \text{ and set theory }\} \\
\Leftarrow & s \mapsto r \in Door \quad . & \qquad (15)
\end{array}
$$

The guard of event $grant$ needs to be strengthened; we replace $grd1$ by $grd1'$,

$$grd1' \ : \ s \mapsto r \in Door \setminus adr(p) \quad ,$$

which implies $s \mapsto r \in Door$, that is, (15). With $grd1'$ in place of $grd1$ the proof succeeds. For invariant $inv8'$ we have some more work to do:

$$
\begin{array}{lll}
 & \forall\, q \cdot arm[\{q\}] \subseteq adr(q)^+[\{\boldsymbol{O}\}] \cup \{\boldsymbol{O}\} & \textit{Invariant } inv8' \\
 & s \mapsto r \in Door \setminus adr(p) & \textit{Guard } grd1' \\
 & s \in \mathsf{dom}(adr(p)) & \textit{Guard } grd2 \\
\vdash & & \\
 & (arm \cup \{p \mapsto r\})[\{q\}] & \textit{Modified invariant } inv8' \\
 & \quad \subseteq (adr \lhd\!\!+ \{p \mapsto adr(p) \cup \{s \mapsto r,\, r \mapsto s\}\})(q)^+[\{\boldsymbol{O}\}] \cup \{\boldsymbol{O}\} &
\end{array}
$$

for all q. Let $D = \{s \mapsto r,\, r \mapsto s\}$. For $q = p$ we have to prove:

$$arm[\{p\}] \cup \{r\} \subseteq (adr(p) \cup D)^+[\{\boldsymbol{O}\}] \cup \{\boldsymbol{O}\}\ ,$$

given that $(arm \cup \{p \mapsto r\})[\{p\}] = arm[\{p\}] \cup \{r\}$. We begin with the case "$arm[\{p\}] \subseteq (adr(p) \cup D)^+[\{\boldsymbol{O}\}] \cup \{\boldsymbol{O}\}$":

$$
\begin{array}{lll}
 & arm[\{p\}] & \{\ inv8' \text{ with "}q := p\text{"}\ \} \\
\subseteq & adr(p)^+[\{\boldsymbol{O}\}] \cup \{\boldsymbol{O}\} & \{\ \text{law (10) with "}x,\, y := adr(p),\, adr(p) \cup D\text{"}\ \} \\
\subseteq & (adr(p) \cup D)^+[\{\boldsymbol{O}\}] \cup \{\boldsymbol{O}\} \quad . &
\end{array}
$$

Before proving the second case "$\{r\} \subseteq (adr(p) \cup D)^+[\{\boldsymbol{O}\}] \cup \{\boldsymbol{O}\}$", we have a closer look at guard $grd2$,

$$
\begin{array}{lll}
 & s \in \mathsf{dom}(adr(p)) & \{\ inv9 \text{ with "}q := p\text{"}\ \} \\
\Rightarrow & s \in \mathsf{ran}(adr(p)) & \{\ inv6 \text{ with "}q := p\text{"}\ \} \\
\Rightarrow & s \in arm[\{p\}] & \{\ inv8' \text{ with "}q := p\text{"}\ \} \\
\Rightarrow & s \in adr(p)^+[\{\boldsymbol{O}\}] \cup \{\boldsymbol{O}\} & \{\ \text{set theory }\} \\
\Leftrightarrow & s \in adr(p)^+[\{\boldsymbol{O}\}] \vee s = \boldsymbol{O} \quad . & \qquad (16)
\end{array}
$$

Now we can conclude the proof, letting $AD = adr(p) \cup D$:

$$
\begin{array}{lll}
 & AD^+[\{\boldsymbol{O}\}] \cup \{\boldsymbol{O}\} & \{\ \text{set theory }\} \\
\supseteq & AD^+[\{\boldsymbol{O}\}] & \{\ \text{law (8) with "}x := AD\text{", set theory }\} \\
= & AD[\{\boldsymbol{O}\}] \cup (AD^+\,;AD)[\{\boldsymbol{O}\}] & \{\ \text{set theory }\} \\
\supseteq & D[\{\boldsymbol{O}\}] \cup (AD^+\,;D)[\{\boldsymbol{O}\}] & \{\ \text{law (10) with "}x,\, y := adr(p), AD\text{"}\ \} \\
\supseteq & D[\{\boldsymbol{O}\}] \cup (adr(p)^+\,;D)[\{\boldsymbol{O}\}] & \{\ \text{set theory }\} \\
= & D[\{\boldsymbol{O}\}] \cup D[adr(p)^+[\{\boldsymbol{O}\}]] & \{\ (16) \text{ and def. of } D\ \}
\end{array}
$$

$\supseteq \quad \{r\} \quad .$

Having specified invariant $inv9$ we would now succeed proving theorem $thm1$ postulated in the beginning of this section. This shows that our model satisfies property P7. We do not carry out the proof but turn to the last event not yet refined.

7.3 Revoking door authorisations

We model revoking of door authorisations symmetrically to granting door authorisations. A door authorisation can be revoked if (a) there is an authorisation for the door, (b) the corresponding person is not in the room that could be removed, and (c) the room is not the outside. Condition (a) is just chosen symmetrically to $grd1$ of refined event $revoke$ (for the same reason). The other two conditions (b) and (c) are already present in the abstraction. The two refined events $grant$ and $revoke$ together model property P9.

$revoke$
  any p, s, r when
    $grd1 : s \mapsto r \in adr(p)$
    $grd2 : p \mapsto r \notin loc$
    $grd3 : r \neq \boldsymbol{O}$
  then
    $act1 : adr := adr \lhd\!\!\!\!- \{p \mapsto adr(p) \setminus \{s \mapsto r, r \mapsto s\}\}$
  end

We expect that the guard of event $revoke$ will be to weak to preserve invariant $inv8'$. We are going to search for it in the corresponding proof. But we can get started without it, in particular, proving guard strengthening of the abstract guards $grd1$ to $grd3$ and preservation of $inv5$, $inv6$, $inv7$, and $inv9$. For instance, preservation of $inv6$:

$\forall q \cdot \mathsf{ran}(adr(q)) \subseteq arm[\{q\}]$	*Invariant* $inv6$
$s \mapsto r \in adr(p)$	*Concrete guard* $grd1$
$p \mapsto r \notin loc$	*Concrete guard* $grd2$
$r \neq \boldsymbol{O}$	*Concrete guard* $grd3$
$\vdash$	
$\mathsf{ran}((adr \lhd\!\!\!\!- \{p \mapsto adr(p) \setminus \{s \mapsto r, r \mapsto s\}\})(q))$	
$\quad \subseteq (arm \setminus \{p \mapsto r\})[\{q\}]$	*Modified invariant* $inv6$

for all q. For $q = p$ we have to prove

$$\mathsf{ran}(adr(p) \setminus \{s \mapsto r, r \mapsto s\}) \subseteq arm[\{p\}] \setminus \{r\} \quad ,$$

thus,

$$r \notin \mathsf{ran}(adr(p) \setminus \{s \mapsto r, r \mapsto s\}) \quad .$$

This does not look right. Indeed, we find a counter example with one person P and three different rooms $\boldsymbol{H}, \boldsymbol{I}, \boldsymbol{O}$:

$$adr = \{P \mapsto \{\boldsymbol{O} \mapsto \boldsymbol{H}, \boldsymbol{H} \mapsto \boldsymbol{O}, \boldsymbol{O} \mapsto \boldsymbol{I}, \boldsymbol{I} \mapsto \boldsymbol{O}, \boldsymbol{I} \mapsto \boldsymbol{H}, \boldsymbol{H} \mapsto \boldsymbol{I}\}\}$$
$$arm = \{P \mapsto \boldsymbol{H}, P \mapsto \boldsymbol{I}, P \mapsto \boldsymbol{O}\}$$
$$loc = \{P \mapsto \boldsymbol{O}\} \quad p = P \quad s = \boldsymbol{I} \quad r = \boldsymbol{H}$$

In order to resolve this problem we could remove all doors connecting to r. But this seems not acceptable: we grant door authorisations one by one and we should revoke them one by one. We could also strengthen the guard of the concrete event requiring, say, $adr(p)[\{r\}] = \{s\}$. But then we would not be able to revoke authorisations once there are two or more doors for the same room. The problem is in the abstraction! The abstract event $revoke$ should not always remove r. We weaken the guard of the abstract event,

$revoke$
  any p, R when
    $grd1 : p \in Person$
    $grd2 : loc(p) \notin R$
    $grd3 : R \in \mathbb{S}(Room \setminus \{\boldsymbol{O}\})$
  then
    $act1 : arm := arm \setminus (\{p\} \times R)$
  end

where for a set X by $\mathbb{S}(X)$ we denote all subsets of X with at most one element:

$$Y \in \mathbb{S}(X) \quad \widehat{=} \quad Y \subseteq X \wedge (\forall x, y \cdot x \in Y \wedge y \in Y \Rightarrow x = y) \quad .$$

With this the proof obligation for invariant preservation of $inv6$ becomes:

$$\begin{array}{ll}
\forall q \cdot \mathsf{ran}(adr(q)) \subseteq arm[\{q\}] & \textit{Invariant } inv6 \\
s \mapsto r \in adr(p) & \textit{Concrete guard } grd1 \\
p \mapsto r \notin loc & \textit{Concrete guard } grd2 \\
r \neq \boldsymbol{O} & \textit{Concrete guard } grd3 \\
\vdash & \\
\quad \mathsf{ran}((adr \lhd\!\!\!\cdot\, \{p \mapsto adr(p) \setminus \{s \mapsto r, r \mapsto s\}\})(q)) & \\
\quad\quad \subseteq (arm \setminus (\{p\} \times R))[\{q\}] & \textit{Modified invariant } inv6
\end{array}$$

for all q. For $q = p$ we have to prove,

$$\mathsf{ran}(adr(p) \setminus \{s \mapsto r, r \mapsto s\}) \subseteq arm[\{p\}] \setminus R \quad . \tag{17}$$

Before we can continue we need to make a connection between r and R. We need a witness for R. After some reflection we decide for

$$R = \{r\} \setminus \mathsf{ran}(adr(p) \setminus \{s \mapsto r, r \mapsto s\}) \quad . \tag{18}$$

Witness (18) explains how the concrete and the abstract event are related. If there is only one door s connecting to room r, then $R = \{r\}$ and the authorisation for room

r is revoked. Otherwise, $R = \varnothing$ and the authorisation for room r is kept. Now we are ready to prove (17). In case $r \in \mathsf{ran}(adr(p) \setminus \{s \mapsto r, r \mapsto s\})$, that is $R = \varnothing$ by (18),

$$\begin{array}{lll} & \mathsf{ran}(adr(p) \setminus \{s \mapsto r, r \mapsto s\}) & \{ \text{ set theory } \} \\ \subseteq & \mathsf{ran}(adr(p)) & \{ \mathit{inv}6 \text{ with } \text{“}q := p\text{”} \} \\ \subseteq & arm[\{p\}] & \{ R = \varnothing \} \\ = & arm[\{p\}] \setminus R \quad , & \end{array}$$

otherwise, that is in case $r \notin \mathsf{ran}(adr(p) \setminus \{s \mapsto r, r \mapsto s\})$,

$$\begin{array}{lll} & \mathsf{ran}(adr(p) \setminus \{s \mapsto r, r \mapsto s\}) & \{ r \notin \mathsf{ran}(\ldots) \} \\ \subseteq & \mathsf{ran}(adr(p)) \setminus \{r\} & \{ R = \{r\} \text{ by (18) } \} \\ = & \mathsf{ran}(adr(p)) \setminus R & \{ \mathit{inv}6 \text{ with } \text{“}q := p\text{”} \} \\ \subseteq & arm[\{p\}] \setminus R \quad . & \end{array}$$

We note without showing the proofs that guard strengthening of the abstract guards $grd1$ to $grd3$ and preservation of $inv5$, $inv7$, and $inv9$ all hold. Only preservation of invariant $inv8'$ remains:

$$\begin{array}{lll} & \forall q \cdot arm[\{q\}] \subseteq adr(q)^+[\{\boldsymbol{O}\}] \cup \{\boldsymbol{O}\} & \textit{Invariant } inv8' \\ & s \mapsto r \in \mathit{Door} \setminus adr(p) & \textit{Guard } grd1' \\ & s \in \mathsf{dom}(adr(p)) & \textit{Guard } grd2 \\ \vdash & (arm \setminus (\{p\} \times R))[\{q\}] & \textit{Modified invariant } inv8' \\ & \quad \subseteq (adr \lhd \{p \mapsto adr(p) \setminus \{s \mapsto r, r \mapsto s\}\})(q)^+[\{\boldsymbol{O}\}] \cup \{\boldsymbol{O}\} & \end{array}$$

for all q. Let $D = \{s \mapsto r, r \mapsto s\}$. For $q = p$ we have to show

$$(arm \setminus (\{p\} \times R))[\{p\}] \subseteq (adr(p) \setminus D)^+[\{\boldsymbol{O}\}] \cup \{\boldsymbol{O}\} \quad . \tag{19}$$

We have seen above that the term on term on the left hand side is either $arm[\{p\}]$ or $arm[\{p\}] \setminus \{r\}$. So we won't succeed proving (19) unless we add a guard to event $revoke$. We cannot use $arm[\{p\}]$ in the guard because the refined machine does not contain variable arm. If $inv6$ was an equality, we could use $\mathsf{ran}(adr(p))$ instead of $arm[\{p\}]$, obtaining the guard

$$grd4 \ : \ \mathsf{ran}(adr(p)) \setminus \{r\} \subseteq (adr(p) \setminus D)^+[\{\boldsymbol{O}\}] \cup \{\boldsymbol{O}\} \quad .$$

It says that all rooms except for r must still be reachable from the outside after revoking the authorisation for door D leading to room r. This sounds reasonable. We find that it is not possible to turn the set inclusion into an equality in invariant $inv6$. However, we can still prove the weaker theorem

$$thm2 \ : \ \forall q \cdot \mathsf{ran}(adr(q)) \cup \{\boldsymbol{O}\} = arm[\{q\}] \quad ,$$

using $inv2$, $inv6$, $inv8'$, and property (8) of the transitive closure. The authorised rooms are maintained precisely by means of the authorised doors. As a matter of fact, initially

we used $thm2$ as invariant instead of $inv6$ but then weakened the invariant to $inv6$ and proved $thm2$ as a theorem. This is a useful strategy for reducing the amount of proof necessary while keeping powerful properties such as $thm2$. Similarly, we get the theorem

$$thm3 \;:\; \forall q \cdot arm[\{q\}] = adr(q)^+[\{\boldsymbol{O}\}] \cup \{\boldsymbol{O}\} \quad .$$

We prove (19) by case distinction similarly to (17). In case $r \in \mathsf{ran}(adr(p) \setminus D)$, letting $AD = adr(p) \setminus D$,

$$\begin{array}{lll}
 & (arm \setminus (\{p\} \times R))[\{p\}] \subseteq AD^+[\{\boldsymbol{O}\}] \cup \{\boldsymbol{O}\} & \{\ R = \varnothing\ \} \\
\Leftrightarrow & arm[\{p\}] \subseteq AD^+[\{\boldsymbol{O}\}] \cup \{\boldsymbol{O}\} & \{\ thm2, \text{“}q := p\text{”}\ \} \\
\Leftrightarrow & \mathsf{ran}(adr(q)) \subseteq AD^+[\{\boldsymbol{O}\}] \cup \{\boldsymbol{O}\} & \{\ grd4\ \} \\
\Leftarrow & AD^+[\{\boldsymbol{O}\}] \cup \{\boldsymbol{O}\} \cup \{r\} \subseteq AD^+[\{\boldsymbol{O}\}] \cup \{\boldsymbol{O}\} & \{\ \text{set theory}\ \} \\
\Leftarrow & r \in AD^+[\{\boldsymbol{O}\}] & \{\ (8), \text{“}x := AD\text{”}\ \} \\
\Leftrightarrow & r \in (AD \cup AD^+ ; AD)[\{\boldsymbol{O}\}] & \{\ \text{set theory}\ \} \\
\Leftarrow & r \in AD[\{\boldsymbol{O}\}] \cup AD[AD^+[\{\boldsymbol{O}\}]] & \{\ AD^{-1}[\{r\}] \neq \varnothing\ \} \\
\Leftarrow & AD^{-1}[\{r\}] \subseteq \{\boldsymbol{O}\} \cup AD^+[\{\boldsymbol{O}\}] & \{\ inv9, \text{def. of } D\ \} \\
\Leftrightarrow & AD[\{r\}] \subseteq \{\boldsymbol{O}\} \cup AD^+[\{\boldsymbol{O}\}] & \{\ \text{set theory}\ \} \\
\Leftarrow & adr(p)[\{r\}] \subseteq \{\boldsymbol{O}\} \cup AD^+[\{\boldsymbol{O}\}] & \{\ r \notin adr(p)[\{r\}]\ \} \\
\Leftarrow & \mathsf{ran}(adr(p)) \setminus \{r\} \subseteq AD^+[\{\boldsymbol{O}\}] \cup \{\boldsymbol{O}\} \quad , &
\end{array}$$

and in case $r \notin \mathsf{ran}(adr(p) \setminus D)$,

$$\begin{array}{lll}
 & (arm \setminus (\{p\} \times R))[\{p\}] & \{\ R = \{r\}\ \} \\
= & arm[\{p\}] \setminus \{r\} & \{\ thm2 \text{ with “}q := p\text{”}\ \} \\
= & (\mathsf{ran}(adr(p)) \cup \{\boldsymbol{O}\}) \setminus \{r\} & \{\ r \neq \boldsymbol{O}\ \} \\
= & (\mathsf{ran}(adr(p)) \setminus \{r\}) \cup \{\boldsymbol{O}\} & \{\ grd4\ \} \\
\subseteq & (adr(p) \setminus D)^+[\{\boldsymbol{O}\}] \cup \{\boldsymbol{O}\} \quad . &
\end{array}$$

Now we have taken into account all important properties P1 to P9 and we have proved that the abstract and the concrete model are consistent. We have proved that all invariants $inv1$ to $inv9$ are preserved by the initialisation and the events $pass$, $grant$ and $revoke$.

8 Towards a better model

Assuming we have one person P and three different rooms $\boldsymbol{H}$, $\boldsymbol{I}$, and $\boldsymbol{O}$ we can inspect how the modelled system would behave.

Initially variables adr and loc have the values

$$adr = Person \times \{\varnothing\}$$

$$loc = Person \times \{\boldsymbol{O}\} \quad .$$

Event $pass$ is disabled as expected; $grd1$, that is, $loc(p) \mapsto r \in adr(p)$ cannot be satisfied for any p and r. Similarly, event $revoke$ is disabled, but also event $grant$: guard $grd2$, $s \in \mathsf{dom}(adr(p))$, cannot be satisfied for any s, leading to a deadlock. We have not proved all properties we would expect from our model. This property seems to be implicitly contained in properties P8 and P9, but we have missed it. We have to weaken $grd2$,

$grant$
  any p, s, r when
    $grd1' : s \mapsto r \in Door \setminus adr(p)$
    $grd2' : s \in \mathsf{dom}(adr(p)) \cup \{\boldsymbol{O}\}$
  then
    $act1 : adr := adr \Leftarrow \{p \mapsto adr(p) \cup \{s \mapsto r, r \mapsto s\}\}$
  end

As a consequence, we have to check again that concrete event $grant$ preserves all invariants. Fortunately, our proof of preservation of $inv8'$ can is easily adaptable because we have first inferred (16) from $grd2$; it is still implied by $grd2'$.

The proof obligations shown in Section 2 have been restricted not to take into account deadlock-freedom to emphasise the problem that we only verify properties where we expect difficulties but not more. This is a problem of formal modelling in general. But it is more visible in the incremental approach.

9 Conclusion

We have demonstrated how a model in Event-B is created incrementally by refinement and alteration. Refinement permits to structure a complex model thus to cope better with complexity. While reasoning formally about the model developed in this article as a whole we found some problems. These led us to alter the model, both the abstraction and the refinement. Although this is only mentioned in the introduction it should be clear how much this depends on good tool support [3,15]. Modifying a model is encouraged by these tools that that have been developed expressly to facilitate changes. Without such tools the approach would fail in practice. In this article we have focused more on methodological benefits than on how to use the respective tools because this is where the principal gain of using them is to be found. The techniques we have used are not meant to be comprehensive. For instance, we have not made use of temporal logic, behaviour specification, or testing.

We have not solved the problem of how to come up with a perfect specification. That is not our aim. We are content with achieving a model of good quality that captures required behaviour reasonably well and reasonably complete. By serious reasoning about the model we have gone some way towards a meticulous validation of the intended behaviour of the model. Some required properties are usually linked to the implementation. They would be difficult to incorporate into a more abstract model. Our solution would be not to incorporate them but to deal with them at the appropriate level.

Acknowledgment I am grateful to Michael Leuschel and the STUPS group at the University of Düsseldorf for their suggestions and stimulating discussions.

References

1. J.-R. Abrial. *The B-Book: Assigning Programs to Meanings*. CUP, 1996.
2. J.-R. Abrial. *Modeling in Event-B: System and Software Engineering*. Cambridge University Press, 2008. To appear.
3. J.-R. Abrial, M. Butler, S. Hallerstede, and L. Voisin. An open extensible tool environment for Event-B. In Z. Liu and J. He, editors, *ICFEM 2006*, volume 4260, pages 588–605. Springer, 2006.
4. J.-R. Abrial and S. Hallerstede. Refinement, Decomposition and Instantiation of Discrete Models: Application to Event-B. *Fundamentae Informatica*, 77(1-2), 2007.
5. J.-R. Abrial and L. Mussat. Introducing dynamic constraints in B. In Didier Bert, editor, *B'98*, volume 1393 of *LNCS*, pages 83–128. Springer, 1998.
6. R.-J. Back. Refinement Calculus II: Parallel and Reactive Programs. In J. W. deBakker, W. P. deRoever, and G. Rozenberg, editors, *Stepwise Refinement of Distributed Systems*, volume 430 of *Lecture Notes in Computer Science*, pages 67–93. Springer, May 1989.
7. Frédéric Badeau and Arnaud Amelot. Using B as a high level programming language in an industrial project: Roissy VAL. In Helen Treharne, Steve King, Martin Henson, and Steve Schneider, editors, *ZB 2005*, volume 3455 of *LNCS*, pages 334–354, 2005.
8. P. Behm, P. Desforges, and J.-M. Meynadier. MéTéOR: An industrial success in formal development. In D. Bert, editor, *B'98*, volume 1393 of *LNCS*, pages 26–26. Springer, 1998.
9. E. Börger and R. Stärk. *Abstract State Machines: A Method for High-Level System Design and Analysis*. Springer-Verlag, 2003.
10. D. Gries and F. B. Schneider. *A Logical Approach to Discrete Math*. Springer, 1994.
11. S. Hallerstede. On the purpose of event-B proof obligations. In E. Börger, M. J. Butler, J. P. Bowen, and P. Boca, editors, *ABZ*, volume 5238 of *LNCS*, pages 125–138. Springer, 2008.
12. T. S. Hoang, H. Kuruma, D. A. Basin, and J.-R. Abrial. Developing topology discovery in event-B. In M. Leuschel and H. Wehrheim, editors, *IFM*, volume 5423 of *LNCS*, pages 1–19. Springer, 2009.
13. I. Lakatos. *Proofs and Refutations*. Cambridge University Press, 1976.
14. L. Lamport. *Specifying Systems, The TLA+ Language and Tools for Hardware and Software Engineers*. Addison-Wesley, 2002.
15. M. Leuschel and M. Butler. ProB : an automated analysis toolset for the B method. *International Journal on Software Tools for Technology Transfer*, 10(2):185–203, 2008.
16. C. C. Morgan. *Programming from Specifications: Second Edition*. Prentice Hall, 1994.
17. G. Pólya. *Mathematics and Plausible Reasoning. Volume 1: Induction and Analogy in Mathematics*. Princeton University Press, Princeton/NJ, 1954.
18. G. Pólya. *How to Solve It: A New Aspect of Mathematical Method*. Princeton Science Library. Princeton University Press, second edition, 1957.
19. Guilhem Pouzancre. How to diagnose a modern car with a formal B model? In Didier Bert, Jonathan P. Bowen, Steve King, and Marina A. Waldén, editors, *ZB*, volume 2651 of *Lecture Notes in Computer Science*, pages 98–100. Springer, 2003.
20. A. J. M. van Gasteren. *On the Shape of Mathematical Arguments*, volume 445 of *LNCS*. Springer, 1990.