Chapter 2

The Ammunition Control System

Paul Mukherjee and John Fitzgerald

Summary

Proving properties of a specification can deepen our knowledge of the specification, leading to clearer specifications, and more elegant and efficient designs. In this chapter we use an existing specification (Mukherjee and Stavridou's model of UN regulations for safe storage of explosives) to illustrate this idea. In particular we demonstrate how to discharge a satisfiability proof obligation, and how to prove the correctness of a specification modification. We see that both proofs further our understanding of the specification and the system itself.

2.1 Introduction

In this chapter we describe the use of proof on the specification of the Ammunition Control System [8] (ACS). The ACS is a system used throughout the UK for controlling the safe storage of explosives. This specification has previously been analysed using a number of different techniques such as animation in the algebraic language OBJ3 [5] and syntax and type analysis using the IFAD VDM-SL Toolbox [3]. It has also had a validation conjecture discharged. However satisfiability of the specification has never been proved.

Here we prove two quite different properties of the ACS specification: we prove the satisfiability of a particular operation, and we prove the equivalence of the specification with a modified version of it (i.e. we prove the correctness of the modification), using the IFAD modular scheme. Our objectives are to demonstrate the use of proof, and to demonstrate typical domains in which proof might be used.

This chapter is organized as follows: in Section 2.2 we give a description of the ACS specification, sufficient for the purposes of the following sections. Then in Section 2.3

we describe the proof of satisfiability of an operation from the ACS specification. This is followed by a modification to the specification, and a proof of the correctness of this modification in Section 2.4. Finally in Section 2.5 we review the proofs, and discuss the outcomes of the exercise.

2.2 The Specification

The Ammunition Control System (ACS) described in [8] is a computer-based system used by the UK Ministry of Defence (MOD) for monitoring and controlling the safe storage of explosives at storage sites. The system is a transaction processing system; when a site takes delivery of a particular explosive object, the ACS system chooses the most appropriate building in the site for the object's storage. The choice is based on the danger of the object, and its compatibility with other explosive objects. The precise rules used by the MOD are not publically available, but are known to be consistent with the UN regulations for the Transport of Dangerous Goods [2].

The specification presented describes a model for a storage site, and the safety requirements which must be observed when a new explosive object is stored at a given storage site. A full description of the study may be found in [8]. Here we restrict ourselves to those portions of the specification relevant to the proofs we will perform.

2.2.1 Explosives Regulations

Explosives are classified in two complementary ways: the *hazard division*, which indicates how dangerous an explosive is; and the compatibility group, which describes what other goods this explosive may be mixed with in storage. There are four hazard divisions, 1 – 4, where 1 is the most dangerous, 4 the least dangerous. Compatibility groups are assigned letters, A–L, excluding I, and S. In VDM-SL we have:

$$Cg = \mathrm{A} \mid \mathrm{B} \mid \mathrm{C} \mid \mathrm{D} \mid \mathrm{E} \mid \mathrm{F} \mid \mathrm{G} \mid \mathrm{H} \mid \mathrm{J} \mid \mathrm{K} \mid \mathrm{L} \mid \mathrm{S}$$

$$Hzd = \mathbb{N}$$

$$\mathsf{inv}\ h \triangleq h \geq 1 \land h \leq 4$$

2.2.2 The Model

The model for a storage site is a hierarchical one. We describe it in a bottom-up fashion.

The fundamental building block is an *Object*. This consists of a description of the explosive item itself, together with an (x, y) co-ordinate representing the position of the object within the magazine in which it is stored. We first consider the description of the explosive item, denoted by the type *Object-desc*. Nett explosive quantity is measured in Kilograms:

$$Kg = \mathbb{R}$$

$$\textsf{inv}\ k \triangleq k \geq 0$$

Thus we can now define what an object-description is:

$$\begin{aligned} \mathit{Object\text{-}desc} :: \ & \mathit{neq} : \mathit{Kg} \\ & \mathit{hzd} : \mathit{Hzd} \\ & \mathit{cg} : \mathit{Cg} \\ & \mathit{xlen} : \mathit{Metre} \\ & \mathit{ylen} : \mathit{Metre} \\ & \mathit{zlen} : \mathit{Metre} \end{aligned}$$

Here, *neq* stands for the nett explosive quantity of an object, *hzd* stands for the hazard division of an object, *cg* the compatibility group of an object and *xlen, ylen* and *zlen* the x, y and z lengths of the object respectively, measured in *Metre*s (positive reals).

Now that we have defined the physical properties of objects, we are free to define objects themselves:

$$\begin{aligned} \mathit{Object} :: \ & \mathit{desc} : \mathit{Object\text{-}desc} \\ & \mathit{pos} : \mathit{Point} \end{aligned}$$

$$\begin{aligned} \mathit{Point} :: \ & x : \mathit{Metre} \\ & y : \mathit{Metre} \end{aligned}$$

Note that there is no z component for objects as they are not stacked, and therefore are always stored at ground level.

A magazine stores a collection of objects. Objects may only be stored in a given magazine provided all the objects are mutually compatible. A magazine is only allowed to store a certain amount of explosive, this amount being known as the *Maximum nett explosive quantity* for the magazine.

For any particular magazine, we will need to know what type of magazine it is (classified according to physical properties), the maximum nett explosive quantity for that magazine, the most dangerous hazard division that the magazine is capable of storing, the physical dimensions of the magazine, and the collection of objects stored in the magazine. In addition, we require that each object in the collection of objects be uniquely labelled to avoid any ambiguity when adding or removing objects to or from the magazine. Finally, we wish to specify that each object in the magazine lies within the bounds of the magazine, and no two objects in the magazine overlap. Thus we get the following definition:

$$\begin{array}{ll} \mathit{Magazine} :: & \mathit{type} : \mathit{Pes\text{-}types} \\ & \mathit{max\text{-}neq} : \mathit{Kg} \mid \mathrm{INFINITY} \\ & \mathit{hzd} : \mathit{Hzd} \\ & \mathit{length} : \mathit{Metre} \\ & \mathit{breadth} : \mathit{Metre} \\ & \mathit{height} : \mathit{Metre} \\ & \mathit{objects} : \mathit{Object\text{-}label} \overset{m}{\longleftrightarrow} \mathit{Object} \end{array}$$

$$\begin{array}{l} \mathsf{inv}\ m \triangleq \\ \quad \forall\, o \in \mathsf{rng}\ m.\mathit{objects} \cdot \\ \qquad \mathit{within\text{-}bounds}(o, m.\mathit{length}, m.\mathit{breadth}, m.\mathit{height}) \wedge \\ \quad \forall\, o1, o2 \in \mathsf{rng}\ m.\mathit{objects} \cdot o1 \neq o2 \;\Rightarrow\; \neg\, \mathit{overlap}(o1, o2) \end{array}$$

Here *Object-label* is a synonym for the type of tokens. Note that for relatively inert hazard divisions, the only limitation on the quantity that may be stored in such a magazine is limited by the physical dimensions of the magazine. In such cases the maximum net explosive quantity for the magazine is said to be INFINITY. The functions *suff-space-at, within-bounds* and *overlap* will be defined in section 2.2.3.

Finally, we describe a *Store* to be a map from magazine labels to magazines, where the label uniquely defines the magazine. Again *Magazine-label* is a synonym for tokens.

$$\begin{array}{l} \mathsf{state}\ \mathit{Store}\ \mathsf{of} \\ \quad \mathit{mags} : \mathit{Magazine\text{-}label} \xrightarrow{m} \mathit{Magazine} \\ \mathsf{end} \end{array}$$

This is a slight simplification of the specification given in [8], but is sufficient for our purposes.

2.2.3 Storing Objects

Safe storage of explosive objects depends on their hazard division and compatibility group. The UN regulations give a list of rules describing which compatibility groups may safely be mixed in storage. Details of these rules, and the errors found in them during the formalization process, may be found in [8]. In addition, an explosive object may never be stored in a magazine whose hazard division is less hazardous than the explosive object's.

When we come to specifying the compatibility rules in VDM, we give a set of pairs of compatibility groups. A pair lies in this set only if the two groups are compatible under these rules. As the relation is symmetric, it is not necessary to include a pair (m, n) if the pair (n, m) already lies in the set. (For brevity we do not give the full definition here.)

$$\mathit{Compatible\text{-}pairs} : \mathit{Cg} \times \mathit{Cg}\text{-}\mathsf{set} = \ldots$$

Note that this relation is not reflexive; for instance, articles of compatibility group

L are specifically required to be stored separate to other articles of the same compatibility group.

If we wish to store a particular object at a particular place in a magazine, we must satisfy four conditions: there is sufficient space for the object at that point in the magazine; the item is not more hazardous than the hazard the magazine was designed to store; the object is compatible with all other objects within the magazine; and the addition of this object does not cause the maximum nett explosive quantity of the magazine to be exceeded. We specify four functions which govern when each of these conditions is met and then combine these to form an overall safety predicate.

Our first function decides when there is sufficient space at a given point in a magazine to house the given object. An object o, may be placed at position (x, y) in a magazine provided that (x, y) lies within the given magazine, and such a placement does not cause the object to overlap with an object already in the magazine.

```
$suff\text{-}space\text{-}at : Object\text{-}desc \times Magazine \times Point \to \mathbb{B}$

$suff\text{-}space\text{-}at\ (od, m, op) \triangleq$
  let $new\text{-}o = mk\text{-}Object(od, op)$ in
  $within\text{-}bounds(new\text{-}o, m.length, m.breadth, m.height) \wedge$
  $\forall\, o1, o2 \in \mathsf{rng}\ m.objects \cup \{new\text{-}o\} \cdot$
        $o1 \neq o2 \;\Rightarrow\; \neg\, overlap(o1, o2)$
```

```
$within\text{-}bounds : Object \times Metre \times Metre \times Metre \to \mathbb{B}$

$within\text{-}bounds\ (o, l, b, h) \triangleq$
  $0 < o.pos.x + o.desc.xlen \wedge o.pos.x + o.desc.xlen \leq l \wedge$
  $0 < o.pos.y + o.desc.ylen \wedge o.pos.y + o.desc.ylen \leq b \wedge$
  $0 < o.desc.zlen \wedge o.desc.zlen \leq h$
```

```
$overlap : Object \times Object \to \mathbb{B}$

$overlap\ (o1, o2) \triangleq$
  abs $(o1.pos.x - o2.pos.x) < o1.desc.xlen \wedge$
  abs $(o1.pos.y - o2.pos.y) < o1.desc.ylen$
```

The above function assumes that a candidate point (x, y) has been found so we need to specify a function which does just that; that is, if there exists a point in a magazine at which there is sufficient space to store an object, then the function returns the co-ordinates of that point. This function is called *find-point*.

Due to the design of our model, it is quite straightforward to specify when an object's hazard division allows it to be stored in a given magazine. An object may only be stored in a magazine provided that the hazard division of the object is no less than the hazard division of the magazine. Thus our second function is:

```
$\mathit{within\text{-}hazard} : \mathit{Object\text{-}desc} \times \mathit{Magazine} \to \mathbb{B}$
$\mathit{within\text{-}hazard}\,(o, m) \triangleq$
  $o.\mathit{hzd} \geq m.\mathit{hzd}$
```

We say that two objects are compatible if under the storage rules they are allowed to be mixed in storage. The compatibility function is defined as:

```
$\mathit{compatible} : \mathit{Cg} \times \mathit{Cg} \to \mathbb{B}$
$\mathit{compatible}\,(m, n) \triangleq$
  $\mathit{mk}\text{-}(m, n) \in \mathit{Compatible\text{-}pairs} \vee \mathit{mk}\text{-}(n, m) \in \mathit{Compatible\text{-}pairs}$
```

We can then formalize our compatibility requirement for the addition of the object to a magazine. An object may be stored in a magazine only if it is compatible with all other objects in the magazine under the rules governing storage.

```
$\mathit{all\text{-}compatible} : \mathit{Object\text{-}desc} \times \mathit{Magazine} \to \mathbb{B}$
$\mathit{all\text{-}compatible}\,(o, m) \triangleq$
  $\forall\, \mathit{object} \in \mathsf{rng}\ m.\mathit{objects} \cdot \mathit{compatible}(o.\mathit{cg}, \mathit{object}.\mathit{desc}.\mathit{cg})$
```

As mentioned before, each magazine has a maximum nett explosive quantity; that is, a capacity of explosive which must not be exceeded. Thus we get the following safety requirement: An object may only be stored in a magazine provided that the addition of this object does not cause the aggregate nett explosive quantity of all the objects in the magazine to exceed the maximum nett explosive quantity of the magazine. So our final function is:

```
$\mathit{suff\text{-}capacity} : \mathit{Object\text{-}desc} \times \mathit{Magazine} \to \mathbb{B}$
$\mathit{suff\text{-}capacity}\,(o, m) \triangleq$
  if $m.\mathit{max\text{-}neq} \neq \mathrm{INFINITY}$
  then $\mathit{total\text{-}neq}(\{\mathit{object}.\mathit{desc} \mid \mathit{object} \in \mathsf{rng}\ m.\mathit{objects}\}) +$
           $o.\mathit{neq} \leq m.\mathit{max\text{-}neq}$
  else true
```

where the function *total-neq* computes the sum of the net explosive capacities of a collection of objects. Note that *suff-space-at* is included in the invariant for magazines, whereas the other three predicates relating magazines and objects are not. This is because magazines for which *suff-space-at* is not satisfied can not physically exist (junk elements). However magazines in which the other three predicates are not satisfied can exist, but would be unsafe.

We can now formally describe when the overall safety requirement for storing objects in magazines. An object may be stored at a point (x, y) in a magazine only if the point is currently unoccupied and the four predicates listed above are satisfied.

$$
\begin{array}{l}
\mathit{safe\text{-}addition} : \mathit{Object\text{-}desc} \times \mathit{Magazine} \times \mathit{Point} \rightarrow \mathbb{B} \\
\mathit{safe\text{-}addition}\,(o, m, p) \triangleq \\
\quad (\forall\, ob \in \mathsf{rng}\ m.\mathit{objects} \cdot ob.\mathit{pos} \notin p) \wedge \\
\quad \mathit{suff\text{-}space\text{-}at}(o, m, p) \wedge \mathit{within\text{-}hazard}(o, m) \wedge \\
\quad \mathit{all\text{-}compatible}(o, m) \wedge \mathit{suff\text{-}capacity}(o, m)
\end{array}
$$

The function *find-point* is used to locate a position within the magazine at which it is safe to store the given explosive object; it implicitly requires that *safe-addition* is satisfied at the chosen point:

$$
\begin{array}{l}
\mathit{find\text{-}point}\,(o : \mathit{Object\text{-}desc}, m : \mathit{Magazine})\ pt : \mathit{Point} \\
\mathsf{pre}\ \ \exists\, pt : \mathit{Point} \cdot \mathit{safe\text{-}addition}(o, m, pt) \\
\mathsf{post}\ \mathit{safe\text{-}addition}(o, m, pt)
\end{array}
$$

Finally we look at the operation for the addition of objects to magazines. Using the above predicate we have:

$$
\begin{array}{l}
\mathit{ADD\text{-}OBJECT}\,(o : \mathit{Object\text{-}desc}, \mathit{obj} : \mathit{Object\text{-}label}, \mathit{ml} : \mathit{Magazine\text{-}label}) \\
\mathsf{ext\ wr}\ \mathit{mags} : \mathit{Magazine\text{-}label} \xrightarrow{m} \mathit{Magazine} \\
\mathsf{pre}\ \ \mathit{ml} \in \mathsf{dom}\ \mathit{mags} \wedge \\
\quad\ \ \mathit{obj} \notin \mathsf{dom}\ \mathit{mags}(\mathit{ml}).\mathit{objects} \wedge \\
\quad\ \ \exists\, pt : \mathit{Point} \cdot \mathit{safe\text{-}addition}(o, \mathit{mags}(\mathit{ml}), pt) \\
\mathsf{post}\ \mathsf{let}\ p = \mathit{find\text{-}point}(o, \mathit{mags}(\mathit{ml}))\ \mathsf{in} \\
\qquad \mathsf{let}\ \mathit{new\text{-}objs} = \\
\qquad\qquad\qquad \mathit{mags}(\mathit{ml}).\mathit{objects} \dagger \{\mathit{obj} \mapsto \mathit{mk\text{-}Object}(o, p)\}\ \mathsf{in} \\
\qquad \mathsf{let}\ \mathit{new\text{-}mag} = \mu\,(\mathit{mags}(\mathit{ml}), \mathit{objects} \mapsto \mathit{new\text{-}objs})\ \mathsf{in} \\
\qquad \mathit{mags} = \overleftarrow{\mathit{mags}} \dagger \{\mathit{ml} \mapsto \mathit{new\text{-}mag}\}
\end{array}
$$

In the above precondition, the three conjuncts respectively formalize the following requirements: that the magazine is known in the store; that the label which we wish to give to the object is not already being used in the given magazine and that it is possible to add an object while satisfying the safety requirements of the previous section. In the postcondition we merely update the state to reflect the addition of the new object.

2.3 Satisfiability of *ADD-OBJECT*

An operation specification is said to be *satisfiable* if, for every input and "before" state satisfying the precondition, there exists some result and "after" state satisfying the postcondition. Showing satisfiability is a common proof task in the context of VDM. In this section, we show how the satisfiability of the *ADD-OBJECT* operation can be tackled using the framework presented in the "practitioner's guide" [1]. We will use the proof rules and notation defined in the guide, except for some small deviations.

The proof of *ADD-OBJECT* is typical of many satisfiability proofs, so the lessons from this example can be applied in many other contexts. This section illustrates a systematic approach to the construction of the satisfiability proof. We begin by attempting the main satisfiability proof. This suggests a splitting of the proof task into subtasks based on showing separate lemmas. We construct proofs of the lemmas, breaking these tasks down further as necessary, until we reach a stage where the lemmas relate to properties of the basic types and operators of VDM-SL rather than to the types and functions in the ACS specification.

It is worth reviewing the relevant parts of the specification before launching into the main proof itself. Sometimes it is very tempting to modify the specification to allow the proof task to proceed smoothly. In this case, the operation *ADD-OBJECT* appears normal, but two points are worth special note: the use of the record modification (μ) operator in the third line of the postcondition and the use of nested let expressions in the postcondition. Although μ is used as though it is a general operator on records, there is in fact a different operator for every possible modification of every possible record type ([1], pages 262-263). We make this explicit by defining an auxiliary function *f-new-mag* which achieves the same effect as the μ expression, but for which the proof theory is more straightforward. The treatment of the nested let expressions is discussed at the final stage in the completion of the satisfiability proof in Section 2.3.3 below.

This modification yields the following version of the operation:

$$
\begin{array}{l}
\textit{ADD-OBJECT}\,(o : \textit{Object-desc}, obj : \textit{Object-label}, ml : \textit{Magazine-label})\\
\mathsf{ext\ wr}\ mags : \textit{Magazine-label} \xrightarrow{m} \textit{Magazine}\\
\mathsf{pre}\ \ ml \in \mathsf{dom}\ mags \wedge\\
\quad obj \notin \mathsf{dom}\ mags(ml).objects \wedge\\
\quad \exists\, pt : \textit{Point} \cdot \textit{safe-addition}(o, mags(ml), pt)\\
\mathsf{post}\ \mathsf{let}\ p = \textit{find-point}(o, mags(ml))\ \mathsf{in}\\
\qquad \mathsf{let}\ \textit{new-objs} = mags(ml).objects \dagger \{obj \mapsto \textit{mk-Object}(o, p)\}\ \mathsf{in}\\
\qquad \mathsf{let}\ \textit{new-mag} = \textit{f-new-mag}(\overleftarrow{mags}(ml), \textit{new-objs})\ \mathsf{in}\\
\qquad mags = \overleftarrow{mags} \dagger \{ml \mapsto \textit{new-mag}\}
\end{array}
$$

The auxiliary function is defined as follows:

$$
\begin{array}{l}
\textit{f-new-mag} : \textit{Magazine} \times (\textit{Object-label} \xleftrightarrow{m} \textit{Object}) \to \textit{Magazine}\\
\textit{f-new-mag}\,(m, om) \triangleq\\
\quad \textit{mk-Magazine}\,(m.type, m.max\text{-}neq, m.hzd,\\
\qquad\qquad\qquad m.length, m.breadth, m.height,\\
\qquad\qquad\qquad om)
\end{array}
$$

In this study, which concentrates on the production of proofs, rather than on the proof theory, we depart slightly from the notation used in [1] for inference rules. Instead of the "horizontal line" notation separating hypotheses from the conclusion, we will list the hypotheses in a "from" line and give the conclusion in an "infer" line.

The satisfiability proof obligation, called '*ADD-OBJECT*-sat', is stated as follows:

from $o : \mathit{Object\text{-}desc}; \mathit{obj} : \mathit{Object\text{-}label}; \mathit{ml} : \mathit{Magazine\text{-}label};$
$\overleftarrow{\mathit{mags}} : \mathit{Magazine\text{-}label} \xrightarrow{m} \mathit{Magazine};$
$\mathit{pre\text{-}ADD\text{-}OBJECT}(o, \mathit{obj}, \mathit{ml}, \overleftarrow{\mathit{mags}})$
infer $\exists\, \mathit{mags} : \mathit{Magazine\text{-}label} \xrightarrow{m} \mathit{Magazine} \cdot$
$\mathit{post\text{-}ADD\text{-}OBJECT}(o, \mathit{obj}, \mathit{ml}, \overleftarrow{\mathit{mags}}, \mathit{mags})$

Notice that this is a slightly simplified version of the obligation given in [1]. The bound variable in the conclusion is not a whole *Store*, but a single *mags* component. As the state consists only of this one component, and has no invariant on it, this simplification is valid.

The remainder of this section describes the proof of '*ADD-OBJECT*-sat'. Rather than provide fully formal proofs in minute detail, the general structure of the proof is presented. More detailed discussion of the proofs can be found in [4].

2.3.1 Main Satisfiability Proof

The proof of '*ADD-OBJECT*-sat' follows the usual pattern for satisfiability proofs. Since the conclusion is an existential quantification, the proof will normally proceed by '∃-I' in which a witness value ([1], page 42) is proposed for the new *mags*. Typically, there are two parts to a proof applying the '∃-I' rule to show satisfiability: showing that the witness value is of the correct type and showing that it satisfies the postcondition. In the case of *ADD-OBJECT*, this yields the following proof structure, where the witness value is shown as '$\mathcal{W}$':

from $o : \mathit{Object\text{-}desc}; \mathit{obj} : \mathit{Object\text{-}label}; \mathit{ml} : \mathit{Magazine\text{-}label};$
$\overleftarrow{\mathit{mags}} : \mathit{Magazine\text{-}label} \xrightarrow{m} \mathit{Magazine};$
$\mathit{pre\text{-}ADD\text{-}OBJECT}(o, \mathit{obj}, \mathit{ml}, \overleftarrow{\mathit{mags}})$
⋮
a $\mathcal{W} : \mathit{Magazine\text{-}label} \xrightarrow{m} \mathit{Magazine}$
⋮
b $\mathit{post\text{-}ADD\text{-}OBJECT}(o, \mathit{obj}, \mathit{ml}, \overleftarrow{\mathit{mags}}, \mathcal{W})$
⋮
infer $\exists\, \mathit{mags} : \mathit{Magazine\text{-}label} \xrightarrow{m} \mathit{Magazine} \cdot$
$\mathit{post\text{-}ADD\text{-}OBJECT}(o, \mathit{obj}, \mathit{ml}, \overleftarrow{\mathit{mags}}, \mathit{mags})$ ∃-I(**a**,**b**)

Normally, showing type correctness (justifying line **a** in the proof above) takes up most of the effort in a proof of correctness. This is because showing the correct type of the witness value entails showing that the witness respects its type's invariant as well as showing that it has the correct basic structure.

We begin by proposing a witness value to stand for '$\mathcal{W}$' in the proof. The postcondition of *ADD-OBJECT* is not very implicit.In fact, it practically gives a "recipe" for the construction of the witness value. Operation specifications with postconditions of the form "*result = expression*" are quite common in practice and, in such cases, the construction of a witness value for the satisfiability proof is straightforward. The postcondition of *ADD-OBJECT* follows this pattern, although it is slightly disguised by the nested let expressions.

The usual approach to the construction of a witness value is to explicitly define an auxiliary function which, given the operation's inputs and "before" state component, constructs the witness value. Here, the function is called *f-mags*:

$$\begin{aligned}
&\textit{f-mags} : \textit{Object-desc} \times \textit{Object-label} \times \textit{Magazine-label} \times \\
&\qquad (\textit{Magazine-label} \xrightarrow{m} \textit{Magazine}) \to \textit{Magazine-label} \xrightarrow{m} \textit{Magazine} \\
&\textit{f-mags}\,(o, ol, ml, old) \triangleq \\
&\quad old \dagger \{ml \mapsto \textit{f-new-mag}(old(ml), \textit{f-new-objs}(old(ml), o))\}
\end{aligned}$$

The body of the function is inspired by the result expression in the postcondition of *ADD-OBJECT*, with auxiliary functions corresponding to the values defined in the let expressions.

The auxiliary function *f-new-mag* has already been introduced and the function *f-new-objs* builds the new objects mapping for inclusion in the new magazine. The definition of *f-new-objs* is as follows:

$$\begin{aligned}
&\textit{f-new-objs} : \textit{Magazine} \times \textit{Object-label} \times \textit{Object-desc} \to \\
&\qquad\qquad (\textit{Object-label} \xleftrightarrow{m} \textit{Object}) \\
&\textit{f-new-objs}\,(om, ol, od) \triangleq \\
&\quad om.objects \dagger \{ol \mapsto \textit{mk-Object}(od, \textit{find-point}(od, om))\}
\end{aligned}$$

Using the auxiliary functions defined so far, the proof of satisfiability is as follows:

$$\begin{array}{lll}
\textsf{from} & o : \textit{Object-desc};\ obj : \textit{Object-label};\ ml : \textit{Magazine-label}; & \\
& \overleftarrow{mags} : \textit{Magazine-label} \xrightarrow{m} \textit{Magazine}; & \\
& \textit{pre-ADD-OBJECT}(o, obj, ml, \overleftarrow{mags}) & \\
& \vdots & \\
\textsf{a} & \textit{f-mags}(o, obj, ml, \overleftarrow{mags}) : \textit{Magazine-label} \xrightarrow{m} \textit{Magazine} & \\
& \vdots & \\
\textsf{b} & \textit{post-ADD-OBJECT}(o, obj, ml, \overleftarrow{mags}, \textit{f-mags}(o, obj, ml, \overleftarrow{mags})) & \\
& \vdots & \\
\textsf{infer} & \exists\, mags : \textit{Magazine-label} \xrightarrow{m} \textit{Magazine} \cdot & \\
& \quad \textit{post-ADD-OBJECT}(o, obj, ml, \overleftarrow{mags}, mags) & \exists\text{-I}(\mathbf{a},\mathbf{b})
\end{array}$$

Now that the witness value has been proposed, we can be more precise about the work required to complete the proof: justifying lines **a** and **b**. These are substantial tasks, so rather than attempting them *in situ*, we define them as two lemmas to be proved separately, allowing us to complete the satisfiability proof itself, so that, when the lemmas are proved, the whole satisfiability obligation will have been discharged.

The first lemma, called '*f-mags*-form-1', is constructed by taking the hypotheses available in the satisfiability proof and showing that line **a** follows from them:

from $o : Object\text{-}desc;\ obj : Object\text{-}label;\ ml : Magazine\text{-}label;$
$\quad\overleftarrow{mags} : Magazine\text{-}label \xrightarrow{m} Magazine;$
$\quad pre\text{-}ADD\text{-}OBJECT(o, obj, ml, \overleftarrow{mags})$
infer $f\text{-}mags(o, obj, ml, \overleftarrow{mags}) : Magazine\text{-}label \xrightarrow{m} Magazine$

This lemma asserts that *f-mags*, when used in the context of the satisfiability proof, will return a mapping from Magazine labels to magazines. The second lemma, called 'p-ADD-OBJECT' is derived from line **b** in the same way. It asserts that the witness value satisfies the postcondition:

from $o : Object\text{-}desc;\ obj : Object\text{-}label;\ ml : Magazine\text{-}label;$
$\quad\overleftarrow{mags} : Magazine\text{-}label \xrightarrow{m} Magazine;$
$\quad pre\text{-}ADD\text{-}OBJECT(o, obj, ml, \overleftarrow{mags})$
infer $post\text{-}ADD\text{-}OBJECT(o, obj, ml, \overleftarrow{mags}, f\text{-}mags(o, obj, ml, \overleftarrow{mags}))$

Using these lemmas, the main satisfiability proof is completed as follows. The proofs of the lemmas are addressed in Sections 2.3.2 and 2.3.3 respectively.

from $o : Object\text{-}desc;\ obj : Object\text{-}label;\ ml : Magazine\text{-}label;$
$\quad\overleftarrow{mags} : Magazine\text{-}label \xrightarrow{m} Magazine;$
$\quad pre\text{-}ADD\text{-}OBJECT(o, obj, ml, \overleftarrow{mags})$
1 $\quad f\text{-}mags(o, obj, ml, \overleftarrow{mags}) : Magazine\text{-}label \xrightarrow{m} Magazine$
f-mags-form-1 (h1,h2,h3,h4,h5)
2 $\quad post\text{-}ADD\text{-}OBJECT(o, obj, ml, \overleftarrow{mags}, f\text{-}mags(o, obj, ml, \overleftarrow{mags}))$
p-ADD-OBJECT (h1,h2,h3,h4,h5)
infer $\exists\, mags : Magazine\text{-}label \xrightarrow{m} Magazine \cdot$
$\quad post\text{-}ADD\text{-}OBJECT(o, obj, ml, \overleftarrow{mags}, mags)$ $\exists$-I (1,2)

2.3.2 Formation of the Witness Value

The first lemma on which the satisfiability of *ADD-OBJECT* rests asserts that the witness value is of the correct type. We can expect in advance that its proof will rely on properties of *f-new-mag* and *f-new-objs*, and so we should be ready to deal with these as separate lemmas if required.

The lemma '*f-mags*-form-1' is a *formation* property. Given the hypotheses, the function *f-mags* must return a well-formed mapping as a result. The first stage in constructing the proof of a formation property is typically to reason backwards from the conclusion, expanding the definition of the function under analysis and justifying the conclusion by folding:

from $o : \mathit{Object\text{-}desc}; \mathit{obj} : \mathit{Object\text{-}label}; \mathit{ml} : \mathit{Magazine\text{-}label};$
$\overleftarrow{\mathit{mags}} : \mathit{Magazine\text{-}label} \xrightarrow{m} \mathit{Magazine};$
$\mathit{pre\text{-}ADD\text{-}OBJECT}(o, \mathit{obj}, \mathit{ml}, \overleftarrow{\mathit{mags}})$

$\vdots$

a $\overleftarrow{\mathit{mags}} \dagger \{\mathit{ml} \mapsto \mathit{f\text{-}new\text{-}mag}(\overleftarrow{\mathit{mags}}(\mathit{ml}), \mathit{f\text{-}new\text{-}objs}(\overleftarrow{\mathit{mags}}(\mathit{ml}), \mathit{obj}, o))\}$
$:\mathit{Magazine\text{-}label} \xrightarrow{m} \mathit{Magazine}$

infer $\mathit{f\text{-}mags}(o, \mathit{obj}, \mathit{ml}, \overleftarrow{\mathit{mags}}) : \mathit{Magazine\text{-}label} \xrightarrow{m} \mathit{Magazine}$ folding (**a**)

Now it can be seen that the body of the function is a mapping overwrite expression. The mapping $\overleftarrow{\mathit{mags}}$ is of the correct type, and *ml* is a *Magazine-label*, so, provided

$$\mathit{f\text{-}new\text{-}mag}(\overleftarrow{\mathit{mags}}(\mathit{ml}), \mathit{f\text{-}new\text{-}objs}(\overleftarrow{\mathit{mags}}(\mathit{ml}), \mathit{obj}, o)) : \mathit{Magazine}$$

we should be able to complete the proof. This last proviso is treated as a lemma, '*f-new-mag*-OK', which is defined as follows:

from $o : \mathit{Object\text{-}desc}; \mathit{obj} : \mathit{Object\text{-}label}; \mathit{ml} : \mathit{Magazine\text{-}label};$
$\overleftarrow{\mathit{mags}} : \mathit{Magazine\text{-}label} \xrightarrow{m} \mathit{Magazine};$
$\mathit{pre\text{-}ADD\text{-}OBJECT}(o, \mathit{obj}, \mathit{ml}, \overleftarrow{\mathit{mags}})$

infer $\mathit{f\text{-}new\text{-}mag}(\overleftarrow{\mathit{mags}}(\mathit{ml}), \mathit{f\text{-}new\text{-}objs}(\overleftarrow{\mathit{mags}}(\mathit{ml}), \mathit{obj}, o)) : \mathit{Magazine}$

Given this lemma, the proof of '*f-mags*-form-1' is completed as follows:

from $o : \mathit{Object\text{-}desc};\ obj : \mathit{Object\text{-}label};\ ml : \mathit{Magazine\text{-}label};$
$\overleftarrow{mags} : \mathit{Magazine\text{-}label} \xrightarrow{m} \mathit{Magazine};$
$\mathit{pre\text{-}ADD\text{-}OBJECT}(o, obj, ml, \overleftarrow{mags})$
1 $\mathit{f\text{-}new\text{-}mag}(\overleftarrow{mags}(ml), \mathit{f\text{-}new\text{-}objs}(\overleftarrow{mags}(ml), obj, o)) : \mathit{Magazine}$
$\mathit{f\text{-}new\text{-}mag}$-OK (h1,h2,h3,h4,h5)
2 $\{ml \mapsto \mathit{f\text{-}new\text{-}mag}(\overleftarrow{mags}(ml), \mathit{f\text{-}new\text{-}objs}(\overleftarrow{mags}(ml), obj, o))\}$
$:\mathit{Magazine\text{-}label} \xrightarrow{m} \mathit{Magazine}$ $\{a \mapsto b\}$-form (h3,1)
3 $\overleftarrow{mags} \dagger \{ml \mapsto \mathit{f\text{-}new\text{-}mag}(\overleftarrow{mags}(ml), \mathit{f\text{-}new\text{-}objs}(\overleftarrow{mags}(ml), obj, o))\}$
$:\mathit{Magazine\text{-}label} \xrightarrow{m} \mathit{Magazine}$ $\dagger$-form (h4,2)
infer $\mathit{f\text{-}mags}(o, obj, ml, \overleftarrow{mags}) : \mathit{Magazine\text{-}label} \xrightarrow{m} \mathit{Magazine}$ folding (3)

Now it remains to prove the new lemma. Again, this is a formation rule, and so we begin by expanding the function definition, working backwards from the conclusion, yielding the proof shown in Figure 2.1. The completed proof uses and three lemmas (at lines 10-12) which relate to the use of the function *f-new-objs*. We need to show that the function returns a bijective mapping and that the returned mapping respects the two conjuncts in the *Magazine* invariant. The first lemma is '*f-new-objs*-form-1':

from $o : \mathit{Object\text{-}desc};\ obj : \mathit{Object\text{-}label};\ ml : \mathit{Magazine\text{-}label};$
$\overleftarrow{mags} : \mathit{Magazine\text{-}label} \xrightarrow{m} \mathit{Magazine};$
$\mathit{pre\text{-}ADD\text{-}OBJECT}(o, obj, ml, \overleftarrow{mags})$
infer $\mathit{f\text{-}new\text{-}objs}(\overleftarrow{mags}(ml), obj, o) : \mathit{Object\text{-}label} \xleftrightarrow{m} \mathit{Object}$

The second lemma is '*new-objs*-bounds'. It asserts that the new object in the witness value is within the physical bounds of the magazine:

from $o : \mathit{Object\text{-}desc};\ obj : \mathit{Object\text{-}label};\ ml : \mathit{Magazine\text{-}label};$
$\overleftarrow{mags} : \mathit{Magazine\text{-}label} \xrightarrow{m} \mathit{Magazine};$
$\mathit{pre\text{-}ADD\text{-}OBJECT}(o, obj, ml, \overleftarrow{mags})$
infer $\forall o' \in \mathsf{rng}\ \mathit{f\text{-}new\text{-}objs}(\overleftarrow{mags}(ml), obj, o) \cdot$
$\mathit{within\text{-}bounds}(o', \overleftarrow{mags}(ml).\mathit{length}, \overleftarrow{mags}(ml).\mathit{breadth}, \overleftarrow{mags}(ml).\mathit{height})$

The third lemma is '*new-objs*-olap', which asserts that the new object does not overlap with existing objects.

from $o : \mathit{Object\text{-}desc};\ obj : \mathit{Object\text{-}label};\ ml : \mathit{Magazine\text{-}label};$
$\quad \overleftarrow{mags} : \mathit{Magazine\text{-}label} \xrightarrow{m} \mathit{Magazine};$
$\quad \mathit{pre\text{-}ADD\text{-}OBJECT}(o, obj, ml, \overleftarrow{mags})$

1 $\quad ml \in \mathsf{dom}\ \overleftarrow{mags} \wedge$
$\quad obj \notin \mathsf{dom}\ \overleftarrow{mags}(ml).objects \wedge$
$\quad \exists\, pt : \mathit{Point} \cdot \mathit{safe\text{-}addition}(o, \overleftarrow{mags}(ml), pt)$ — unfold (h5)

2 $\quad ml \in \mathsf{dom}\ \overleftarrow{mags}$ — $\wedge$-E-right (1)

3 $\quad \overleftarrow{mags}(ml) : \mathit{Magazine}$ — at-form (h3,h4,2)

4 $\quad \overleftarrow{mags}(ml).type : \mathit{Pes\text{-}types}$ — *type*-form (3)

5 $\quad \overleftarrow{mags}(ml).max\text{-}neq : \mathit{Kg} \mid \mathit{INFINITY}$ — *max-neq*-form (3)

6 $\quad \overleftarrow{mags}(ml).hzd : \mathit{Hzd}$ — *hzd*-form (3)

7 $\quad \overleftarrow{mags}(ml).length : \mathit{Metre}$ — *length*-form (3)

8 $\quad \overleftarrow{mags}(ml).breadth : \mathit{Metre}$ — *breadth*-form (3)

9 $\quad \overleftarrow{mags}(ml).height : \mathit{Metre}$ — *height*-form (3)

10 $\quad f\text{-}new\text{-}objs(\overleftarrow{mags}(ml), obj, o) : \mathit{Object\text{-}label} \overset{m}{\longleftrightarrow} \mathit{Object}$
— *f-new-objs*-form-1 (h1,h2,h3,h4,h5)

11 $\quad \forall\, o' \in \mathsf{rng}\ f\text{-}new\text{-}objs(\overleftarrow{mags}(ml), obj, o) \cdot$
$\quad \mathit{within\text{-}bounds}(o', \overleftarrow{mags}(ml).length, \overleftarrow{mags}(ml).breadth, \overleftarrow{mags}(ml).height)$
— *new-objs*-bounds (h1,h2,h3,h4,h5)

12 $\quad \forall\, o1, o2 \in \mathsf{rng}\ f\text{-}new\text{-}objs(\overleftarrow{mags}(ml), obj, o) \cdot$
$\qquad o1 \neq o2 \;\Rightarrow\; \neg\, \mathit{overlap}(o1, o2)$ — *new-objs*-olap (h1,h2,h3,h4,h5)

13 $\quad \mathit{inv\text{-}Magazine}(\overleftarrow{mags}.type, \overleftarrow{mags}.max\text{-}neq, \overleftarrow{mags}.hzd,$
$\qquad \overleftarrow{mags}.length, \overleftarrow{mags}.breadth, \overleftarrow{mags}.height,$
$\qquad f\text{-}new\text{-}objs(\overleftarrow{mags}, obj, o))$ — folding ($\wedge$-I(11,12))

14 $\quad \mathit{mk\text{-}Magazine}(\overleftarrow{mags}.type, \overleftarrow{mags}.max\text{-}neq, \overleftarrow{mags}.hzd,$
$\qquad \overleftarrow{mags}.length, \overleftarrow{mags}.breadth, \overleftarrow{mags}.height,$
$\qquad f\text{-}new\text{-}objs(\overleftarrow{mags}, obj, o)) : \mathit{Magazine}$
— *mk-Magazine*-form (3,4,5,6,7,8,9,10,13)

infer $f\text{-}new\text{-}mag(\overleftarrow{mags}(ml), f\text{-}new\text{-}objs(\overleftarrow{mags}(ml), obj, o)) : \mathit{Magazine}$
— folding (14)

Figure 2.1: Proof of '*f-new-mag*-OK'

from $o : \mathit{Object\text{-}desc};\ obj : \mathit{Object\text{-}label};\ ml : \mathit{Magazine\text{-}label};$
$\overleftarrow{mags} : \mathit{Magazine\text{-}label} \xrightarrow{m} \mathit{Magazine};$
$\mathit{pre\text{-}ADD\text{-}OBJECT}(o, obj, ml, \overleftarrow{mags})$
infer $\forall\, o1, o2 \in \mathsf{rng}\ \mathit{f\text{-}new\text{-}objs}(\overleftarrow{mags}(ml), obj, o) \cdot$
$o1 \neq o2 \ \Rightarrow\ \neg\, \mathit{overlap}(o1, o2)$

We will deal with each of these lemmas in order. First, the formation lemma which requires us to show that the addition of the new object does not compromise the bijective nature of the objects mapping in the magazine. The additions to the mapping are the label *obj* and the object

$$\mathit{mk\text{-}Object}(\mathit{find\text{-}point}(o, \overleftarrow{mags}(ml).\mathit{objects}))$$

In order to ensure that the *objects* mapping remains bijective, we should check that the new object is not already in the range of the mapping. This gives the following outline proof:

from $o : \mathit{Object\text{-}desc};\ obj : \mathit{Object\text{-}label};\ ml : \mathit{Magazine\text{-}label};$
$\overleftarrow{mags} : \mathit{Magazine\text{-}label} \xrightarrow{m} \mathit{Magazine};$
$\mathit{pre\text{-}ADD\text{-}OBJECT}(o, obj, ml, \overleftarrow{mags})$
a $\overleftarrow{mags}(ml).\mathit{objects} : \mathit{Object\text{-}label} \overset{m}{\longleftrightarrow} \mathit{Object}$
b $\mathit{mk\text{-}Object}(o, \mathit{find\text{-}point}(o, \overleftarrow{mags}(ml))) : \mathit{Object}$
c $\mathit{mk\text{-}Object}(o, \mathit{find\text{-}point}(o, \overleftarrow{mags}(ml))) \notin \mathsf{rng}\ \overleftarrow{mags}(ml).\mathit{objects}$
d $\overleftarrow{mags}(ml).\mathit{objects} \dagger \{obj \mapsto \mathit{mk\text{-}Object}(o, \mathit{find\text{-}point}(o, \overleftarrow{mags}(ml)))\}$
bimap-pres(a,h2,b,c)
infer $\mathit{f\text{-}new\text{-}objs}(\overleftarrow{mags}(ml), obj, o) : \mathit{object\text{-}label} \overset{m}{\longleftrightarrow} \mathit{Object}$ folding (**d**)

The 'bimap-pres' lemma used in line **d** asserts that adding a maplet to a bijective mapping does not compromise the bijection, provided the range element of the maplet is not already present in the mapping. Formally:

from $m : A \overset{m}{\longleftrightarrow} B;\ a : A;\ b : B;\ b \notin \mathsf{rng}\ m$
infer $m \dagger \{a \mapsto b\} : A \overset{m}{\longleftrightarrow} B$

The lemma is not proved here. A more detailed discussion of the proofs of lemmas used in this section can be found in [4].

To complete the proof of '*f-new-objs*-form-1', consider each of the five labelled lines in turn. Line **a** follows directly from the structure of the *Magazine* composite type. Line **b** follows from *mk-Object* formation, for *o* is an object description and *find-point* must return a point (we treat this as a lemma on *find-point*). Line **c**

from $o : Object\text{-}desc;\ obj : Object\text{-}label;\ ml : Magazine\text{-}label;$
$\overleftarrow{mags} : Magazine\text{-}label \xrightarrow{m} Magazine;$
$pre\text{-}ADD\text{-}OBJECT(o, obj, ml, \overleftarrow{mags})$

1 $ml \in \mathsf{dom}\ \overleftarrow{mags}$ $\wedge$-E-right(unfold (h5))

2 $\overleftarrow{mags}(ml) : Magazine$ at-form (h3,h4,1)

3 $\overleftarrow{mags}(ml).objects : Object\text{-}label \xleftrightarrow{m} Object$ *Objects*-form (2)

4 $\exists\, pt : Point \cdot safe\text{-}addition(o, \overleftarrow{mags}(ml), pt)$ $\wedge$-E, unfolding (h5)

5 $pre\text{-}find\text{-}point(o, \overleftarrow{mags}(ml))$ folding (4)

6 $find\text{-}point(o, \overleftarrow{mags}(ml)) : point$ *find-point*-form (h1,2,5)

7 $mk\text{-}Object(o, find\text{-}point(o, \overleftarrow{mags}(ml))) : Object$ *mk-Object*-form (h1,6)

8 $post\text{-}find\text{-}point(o, \overleftarrow{mags}(ml), find\text{-}point(o, \overleftarrow{mags}(ml)))$
find-point-defn (h1,2,5)

9 $safe\text{-}addition(o, \overleftarrow{mags}(ml), find\text{-}point(o, \overleftarrow{mags}(ml)))$ unfolding (8)

10 $\forall\, o \in \mathsf{rng}\ \overleftarrow{mags}(ml).objects \cdot o.pos \neq find\text{-}point(o, \overleftarrow{mags}(ml))$
$\wedge$-E, unfolding (9)

11 $\mathsf{rng}\ \overleftarrow{mags}(ml) : Object\text{-}\mathsf{set}$ rng-form-bimap (3)

12 $mk\text{-}Object(o, find\text{-}point(o, \overleftarrow{mags}(ml))) \notin \mathsf{rng}\ \overleftarrow{mags}(ml).objects$
point-excl(6,h1,11,10)

13 $\overleftarrow{mags}(ml).objects \dagger \{obj \mapsto mk\text{-}Object(o, find\text{-}point(o, \overleftarrow{mags}(ml)))\}$
bimap-pres(3,h2,7,12)

infer $f\text{-}new\text{-}objs(\overleftarrow{mags}(ml), obj, o) : object\text{-}label \xleftrightarrow{m} Object$ folding (13)

Figure 2.2: Completed proof of '*f-new-objs*-form-1'

follows because *safe-addition* (part of *pre-ADD-OBJECT*) ensures that no objects already exist at the same point. This gives the completed version of the proof shown in Figure 2.2. The completed proof of '*f-new-objs*-form-1' uses one further lemma. Called 'point-excl', this states that the object constructed at a new point cannot be in the set of existing objects:

from $pt : Point;\ od : Object\text{-}desc;\ s : Object\text{-}\mathsf{set};$
$\forall\, o \in s \cdot o.pos \neq pt$
infer $mk\text{-}Object(od, pt) \notin s$

The proof of this lemma is not discussed here, but is covered in [4].

The proof of '*f-new-objs*-form-1', for brevity, uses the "working versions" of the formation rules for the implicitly defined function *find-point*, as defined in [1], pgs. 324-325. The use of these working versions is contingent on the satisfiability of *find-point* having been shown, and again we assume this has been done.

from $o : \mathit{Object\text{-}desc};\ obj : \mathit{Object\text{-}label};\ ml : \mathit{Magazine\text{-}label};$
$\overleftarrow{mags} : \mathit{Magazine\text{-}label} \xrightarrow{m} \mathit{Magazine};$
$\mathit{pre\text{-}ADD\text{-}OBJECT}(o, obj, ml, \overleftarrow{mags})$

1 $ml \in \mathsf{dom}\ \overleftarrow{mags}$ — $\wedge$-E-right (unfolding (h5))

2 $\overleftarrow{mags}(ml) : \mathit{Magazine}$ — at-form (h3,h4,1)

3 $\exists\, pt : \mathit{Point} \cdot \mathit{safe\text{-}addition}(o, \overleftarrow{mags}(ml), pt)$ — $\wedge$-E, unfolding (h5)

4 $\mathit{pre\text{-}find\text{-}point}(o, \overleftarrow{mags}(ml))$ — folding (3)

5 $\mathit{post\text{-}find\text{-}point}(o, \overleftarrow{mags}(ml), \mathit{find\text{-}point}(o, \overleftarrow{mags}(ml)))$
— *find-point*-defn (h1,2,4)

6 $\mathit{safe\text{-}addition}(o, \overleftarrow{mags}(ml), \mathit{find\text{-}point}(o, \overleftarrow{mags}(ml)))$ — unfolding (5)

7 $\mathit{suff\text{-}space\text{-}at}(o, \overleftarrow{mags}(ml), \mathit{find\text{-}point}(o, \overleftarrow{mags}(ml)))$ — unfolding (6)

8 $\mathit{within\text{-}bounds}(\mathit{mk\text{-}Object}(o, \mathit{find\text{-}point}(o, \overleftarrow{mags}(ml))),$
$\overleftarrow{mags}(ml).\mathit{length}, \overleftarrow{mags}(ml).\mathit{breadth}, \overleftarrow{mags}(ml).\mathit{height})$
— $\wedge$-E, unfolding (7)

9 $\mathit{inv\text{-}Magazine}(\overleftarrow{mags}(ml))$ — *inv-Magazine*-I (2)

10 $\forall\, o' \in \mathsf{rng}\ \overleftarrow{mags}()ml.\mathit{objects} \cdot$
$\mathit{within\text{-}bounds}(o', \overleftarrow{mags}(ml).\mathit{length}, \overleftarrow{mags}(ml).\mathit{breadth}, \overleftarrow{mags}(ml).\mathit{height})$
— $\wedge$-E, unfolding (9)

11 $\overleftarrow{mags}(ml).\mathit{objects} : \mathit{Object\text{-}label} \overset{m}{\longleftrightarrow} \mathit{Object}$ — *objects*-form (2)

12 $\mathit{find\text{-}point}(o, \overleftarrow{mags}(ml)) : \mathit{Point}$ — *find-point*-form (h1,12)

13 $\mathit{mk\text{-}Object}(o, \mathit{find\text{-}point}(o, \overleftarrow{mags}(ml))) : \mathit{Object}$ — *mk-Object*-form (h1, 12)

14 $\mathsf{rng}\ \overleftarrow{mags}(ml).\mathit{objects} \dagger \{obj \mapsto \mathit{mk\text{-}Object}(o, \mathit{find\text{-}point}(o, \overleftarrow{mags}(ml)))\}$
$= \mathsf{rng}\ \overleftarrow{mags}(ml).\mathit{objects} \cup \{\mathit{mk\text{-}Object}(o, \mathit{find\text{-}point}(o, \overleftarrow{mags}(ml)))\}$
— rng-$\dagger$-bimap(11,h2,13,$\wedge$-E(unfold h5))

15 $\mathsf{rng}\ \overleftarrow{mags}(ml).\mathit{objects} : \mathit{Object}\text{-}\mathsf{set}$ — rng-form-bimap (11)

infer $\forall\, o' \in \mathsf{rng}\ \mathit{f\text{-}new\text{-}objs}(\overleftarrow{mags}(ml), obj, o) \cdot$
$\mathit{within\text{-}bounds}(o', \overleftarrow{mags}(ml).\mathit{length}, \overleftarrow{mags}(ml).\mathit{breadth}, \overleftarrow{mags}(ml).\mathit{height})$
— $\forall$-$\cup$-inh (15,11,14,10,8)

Figure 2.3: Proof of '*new-objs*-bounds'

We have established the basic type-correctness of the new objects mapping and hence of the magazine added to the state by the *ADD-POINT* operation. It remains to show that the new mapping respects the invariant on *Magazine* by proving the lemmas '*new-objs*-bounds' and '*new-objs*-olap'. The proof of '*new-objs*-bounds' is shown in Figure 2.3. The proof utilises two lemmas. The first, 'rng-$\dagger$-bimap' is a property of bijective mappings:

$\mathsf{from}\ m : A \xleftrightarrow{m} B; a : A; b : B; a \notin \mathsf{dom}\ m$
$\mathsf{infer}\ \mathsf{rng}\ m \dagger \{a \mapsto b\} = \mathsf{rng}\ m \cup \{b\}$

This lemma is not further discussed here. The second lemma used,'$\forall$-$\cup$-inh', is a property of sets:

$\mathsf{from}\ s1 : A\text{-}\mathsf{set}; a : A; s = s1 \cup \{a\};$
$\quad \forall x \in s1 \cdot P(x); P(a)$
$\mathsf{infer}\ \forall x \in s \cdot P(x)$

This lemma's proof is also omitted here. For this proof, we again expect *find-point* to have been proved satisfiable, so that the working versions of the function's formation and definition rules can be used.

By this stage in the proof, the lemmas generated from proofs are now no longer lemmas about the particular formal specification under analysis, but relate almost exclusively to the underlying data types and operators in VDM-SL. They represent results which should be provable from the theories given in [1].

The one remaining lemma is the proof of '*new-objs*-olap. Its proof follows similar lines to that of its companion lemma '*new-objs*-bounds' and the proof is not presented here.

All the lemmas required for the proof of '*f-new-mag*-OK', and thus for '*f-mags*-form-1' have been proved. We have shown that the witness value is a well-formed *Magazine*. It remains to show that this magazine respects the postcondition on *ADD-OBJECT*.

2.3.3 Satisfaction of Postcondition

Recall that satisfaction of *post-ADD-OBJECT* is described in the lemma 'p-ADD-OBJECT' as follows:

$\mathsf{from}\ o : \mathit{Object\text{-}desc}; \mathit{obj} : \mathit{Object\text{-}label}; \mathit{ml} : \mathit{Magazine\text{-}label};$
$\quad \overleftarrow{\mathit{mags}} : \mathit{Magazine\text{-}label} \xrightarrow{m} \mathit{Magazine};$
$\quad \mathit{pre\text{-}ADD\text{-}OBJECT}(o, \mathit{obj}, \mathit{ml}, \overleftarrow{\mathit{mags}})$
$\mathsf{infer}\ \mathit{post\text{-}ADD\text{-}OBJECT}(o, \mathit{obj}, \mathit{ml}, \overleftarrow{\mathit{mags}}, \mathit{f\text{-}mags}(o, \mathit{obj}, \mathit{ml}, \overleftarrow{\mathit{mags}}))$

Such satisfaction proofs are usually straightforward in cases where the operation specification is quite explicit and the postcondition is of the form "*result = expression*". The witness value has usually been chosen so that it is identical to *expression*,

from $o : \mathit{Object\text{-}desc}; \mathit{obj} : \mathit{Object\text{-}label}; \mathit{ml} : \mathit{Magazine\text{-}label};$
$\quad \overleftarrow{\mathit{mags}} : \mathit{Magazine\text{-}label} \xrightarrow{m} \mathit{Magazine};$
$\quad \mathit{pre\text{-}ADD\text{-}OBJECT}(o, \mathit{obj}, \mathit{ml}, \overleftarrow{\mathit{mags}})$

1 $\quad \mathit{f\text{-}mags}(o, \mathit{obj}, \mathit{ml}, \overleftarrow{\mathit{mags}}) : \mathit{Magazine\text{-}label} \xrightarrow{m} \mathit{Magazine}$
$\qquad\qquad$ $\mathit{f\text{-}mags\text{-}form}$-1 (h1,h2,h3,h4,h5)

2 $\quad \mathit{f\text{-}mags}(o, \mathit{obj}, \mathit{ml}, \overleftarrow{\mathit{mags}}) = \mathit{f\text{-}mags}(o, \mathit{obj}, \mathit{ml}, \overleftarrow{\mathit{mags}})$
$\qquad\qquad$ =-self-I (1)

3 $\quad \mathit{f\text{-}mags}(o, \mathit{obj}, \mathit{ml}, \overleftarrow{\mathit{mags}}) =$
$\qquad \overleftarrow{\mathit{mags}} \dagger \{\mathit{ml} \mapsto \mathit{f\text{-}new\text{-}mag}(\overleftarrow{\mathit{mags}}(\mathit{ml}), \mathit{f\text{-}new\text{-}objs}(\overleftarrow{\mathit{mags}}(\mathit{ml}), \mathit{obj}, o))\}$
$\qquad\qquad$ unfold (2)

4 $\quad \mathit{f\text{-}mags}(o, \mathit{obj}, \mathit{ml}, \overleftarrow{\mathit{mags}}) =$
$\qquad \overleftarrow{\mathit{mags}} \dagger \{\mathit{ml} \mapsto \mathit{f\text{-}new\text{-}mag}(\overleftarrow{\mathit{mags}}(\mathit{ml}),$
$\qquad \overleftarrow{\mathit{mags}}(\mathit{ml}).\mathit{objects} \dagger \{\mathit{obj} \mapsto \mathit{mk\text{-}Object}(o, \mathit{find\text{-}point}(o, \overleftarrow{\mathit{mags}}(\mathit{ml})))\})\}$
$\qquad\qquad$ unfold (3)

5 $\quad$ let $p = \mathit{find\text{-}point}(o, \overleftarrow{\mathit{mags}}(\mathit{ml}))$ in
$\quad$ let $\mathit{new\text{-}objs} = \overleftarrow{\mathit{mags}}(\mathit{ml}).\mathit{objects} \dagger \{\mathit{obj} \mapsto \mathit{mk\text{-}Object}(o, p)\}$ in
$\quad$ let $\mathit{new\text{-}mag} = \mathit{f\text{-}new\text{-}mag}(\overleftarrow{\mathit{mags}}(\mathit{ml}), \mathit{new\text{-}objs})$ in
$\quad \mathit{f\text{-}mags}(o, \mathit{obj}, \mathit{ml}, \overleftarrow{\mathit{mags}}) = \overleftarrow{\mathit{mags}} \dagger \{\mathit{ml} \mapsto \mathit{new\text{-}mag}\}$ $\qquad$ let, 4

infer $\mathit{post\text{-}ADD\text{-}OBJECT}(o, \mathit{obj}, \mathit{ml}, \overleftarrow{\mathit{mags}}, \mathit{f\text{-}mags}(o, \mathit{obj}, \mathit{ml}, \overleftarrow{\mathit{mags}}))$
$\qquad\qquad$ folding (5)

Figure 2.4: Rigorous proof of satisfaction of *post-ADD-OBJECT*

so the satisfaction proof amounts to showing this identity. The witness value has already been shown to denote a value of the correct type, so we can easily conclude the expression

$$\mathit{witness\text{-}value} = \mathit{witness\text{-}value}$$

because equality is reflexive for defined values. We then proceed to expand one side of the equality until we get the expression in the postcondition, giving "$\mathit{witness} = \mathit{expression}$". The same approach is applied to the *ADD-OBJECT* operation.

The only complication here is the use of nested let expressions in the postcondition. In the rigorous proof presented here, we treat the let expressions in the obvious way, omitting the details of how this can be justified by appealing to proof rules. This is discussed in more detail in [4]. The resulting proof of satisfaction of the postcondition is given in Figure 2.4.

2.3.4 Summary

We have shown how a typical satisfiability proof can be tackled in VDM-SL by breaking the task down into subtasks by means of lemmas. It is worth taking stock of the proof process before considering a less typical proof in the context of the ACS.

The proof of satisfiability often highlights errors in the specification of the operation and related auxiliary functions. The construction of the satisfiability proof for *ADD-OBJECT* raised issues which had not been addressed in any of the previous validations performed on the specification. For example, it became apparent that the original version of *ADD-OBJECT* had too weak a precondition, allowing a magazine label outside the domain of *mags* to be used. In a more subtle example, the specification failed to exclude the possible sharing of positions by objects in certain cases. This is not an argument in favour of formal proof as a validation mechanism, but it is an argument in favour of allowing proof structures to guide the attention of reviewers. For example, it is conceivable that very good tool support could have generated test cases testing which would have made at least the first of these errors apparent. The development of such tool support is an area of ongoing research.

2.4 Modifying the Specification

Our specification of the ammunition control system is static in the sense that modifications to the UN regulations would require wholesale modifications to the specification. However, the specification encompasses both the current safety requirements of the UN regulations, and their *interpretation.* Thus it would be convenient if we were able to separate these two aspects of the specification, so that future modifications to the regulations which do not alter the their interpretation (such as new compatibility groups) could be incorporated with ease. This is a very real requirement, as clearly explosives technology rapidly changes, and therefore new substances with different chemical properties to existing substances will have to be dealt with.

Constructing such a separated specification poses no real problems as such; however we need to demonstrate that the separated specification is equivalent to the original one. In this case, we take equivalence of specifications to mean that each specification captures precisely the same set of models as the other.

In this section we give an example of such a separation, and discharge the proof obligations necessary to establish the equivalence of the original and separated specifications. The separation that we perform concerns the test of whether two objects are compatible. We separate our specification by constructing a parametrised module modules to deal with compatibility groups, then instantiate this in the main specification. We use the modular scheme used in the IFAD VDM-SL Toolbox [3] for this purpose.

2.4.1 Modification to the Specification

We wish to make the treatment of compatibility more flexible. We do this by constructing a new module that deals exclusively with compatibility groups. For each interpretation of the UN regulations, we will instantiate this module with the collection of compatibility groups relevant to the current application, and the rules governing safe combinations of compatibility groups. Thus our module takes a type parameter:modules

$\mathsf{module}\ \mathit{Compatibility}$

$\quad\mathsf{parameters}$

$\qquad\mathsf{types}\ CG$

We construct a model of a generic compatibility relation, which will be instantiated by each interpretation of the regulations. We say that a compatibility relation is a map from a compatibility group cg to a set of compatibility groups s, such that each compatibility group in s is compatible with cg. Thus we have

types

$$\mathit{CompatRel} = CG \xrightarrow{m} CG\text{-}\mathsf{set}$$

$$\mathsf{inv}\ cr \triangleq \forall\, a \in \mathsf{dom}\ cr \cdot \forall\, b \in cr(a) \cdot b \in \mathsf{dom}\ cr \wedge a \in cr(b)$$

In the invariant we specify the property that compatibility is symmetric.

We can think of the module modules as an abstract data type – users of compatibility groups need not know how the type is specified, they merely need access to operations that manipulate the data type. Thus we have constructor functions for *CompatRel*:

$$\mathit{emptyCR} : () \to \mathit{CompatRel}$$

$$\mathit{emptyCR}\,() \triangleq$$
$$\quad \{\}$$

$$\mathit{addpair} : \mathit{CompatRel} \times CG \times CG \to \mathit{CompatRel}$$

$$\mathit{addpair}\,(cr, cg1, cg2) \triangleq$$
$$\quad cr \dagger \{cg1 \mapsto cr(cg1) \cup \{cg2\}, cg2 \mapsto cr(cg2) \cup \{cg1\}\}$$
$$\mathsf{pre}\ \ \{cg1, cg2\} \subseteq \mathsf{dom}\ cr$$

Note that a compatibility group cg_2 can only be added to those groups compatible with cg_1 if both cg_1 and cg_2 already lie in the domain of the relation. Thus we need a way of adding compatibility groups to the domain of a relation. The function *addCG* performs this.

```
$addCG : CG \times CompatRel \rightarrow CompatRel$
$addCG\,(cg, cr) \triangleq$
  if $cg \in$ dom $cr$
  then $cr$
  else $cr \dagger \{cg \mapsto \{\}\}$
```

Then we say that two compatibility groups cg_1 and cg_2 are compatible with respect to compatibility relation cr if $cg_2 \in cr(cg_1)$.

```
$compatible : CG \times CG \times CompatRel \rightarrow \mathbb{B}$
$compatible\,(cg1, cg2, cr) \triangleq$
  $cg2 \in cr(cg1)$
pre  $cg1 \in$ dom $cr$
```

In our original specification we represented the compatibility relation as a set of pairs of compatibility groups, with groups cg_1 and cg_2 compatible iff $mk\text{-}(cg_1, cg_2)$ lies in the set, or $mk\text{-}(cg_2, cg_1)$ lies in the set. In order to prove that the new representation is equivalent to the original one, we need some way of relating the two representations. We do this using the function *build-rel*, which takes a set of pairs of compatibility groups, and returns that element of *CompatRel* corresponding to this relation. This gives:

```
$build\text{-}rel : CG \times CG\text{-set} \rightarrow CompatRel$
$build\text{-}rel\,(cgs) \triangleq$
  if $cgs = \{\}$
  then $\{\}$
  else let $mk\text{-}(cg1, cg2) \in cgs$ in
      let $cr = build\text{-}rel(cgs \setminus \{mk\text{-}(cg1, cg2)\})$ in
      let $new\text{-}cr = addCG(cg2, addCG(cg1, cr))$ in
      $new\text{-}cr \dagger \{cg1 \mapsto new\text{-}cr(cg1) \cup \{cg2\}, cg2 \mapsto new\text{-}cr(cg2) \cup \{cg1\}\}$
```

This completes the module. The only remaining changes we need to make are to the ACS module itself.modules We make two changes to the module. The first alteration is to instantiate the Compatibility module with the type Cg, which we enumerate as previously:

```
module $ACS$
  instantiation
    $CgRel$ as $Compatibility\,(CG \rightarrow Cg)$all
  types
    $Cg = \mathrm{A} \mid \mathrm{B} \mid \mathrm{C} \mid \mathrm{D} \mid \mathrm{E} \mid \mathrm{F} \mid \mathrm{G} \mid \mathrm{H} \mid \mathrm{J} \mid \mathrm{K} \mid \mathrm{L} \mid \mathrm{S}$
```

The second alteration is to the test of compatibility: we use the operation defined in the new Compatibility module.

$$
\begin{aligned}
&\mathit{compatible} : Cg \times Cg \to \mathbb{B} \\
&\mathit{compatible}\,(m, n) \triangleq \\
&\quad \mathit{CgRel`compatible}(m, n, \mathit{CgRel`build\text{-}rel}(\mathit{Compatible\text{-}pairs}))
\end{aligned}
$$

This completes our description of the new specification. A couple of points arise: firstly, it is clear to see that the new specification is much more flexible that the original one, without greatly altering it. Secondly, in this instance we can think of the module mechanism as allowing us to specify information hiding (in this case, we are hiding the details of the compatibility relation). modules

Note that there are many ways that we could have specified the Compatibility module. For instance, we could have had a second parameter, which took a set of pairs of compatibility groups, and then had a state based module using this set of pairs. However, we chose this approach described because it is simple and it provides the desired functionality. Since we are performing two tasks here (delegating some functionality to a new module, modules and using maps rather than sets), we could have explicitly constructed the new specification in two stages (leading to two tiers of proof). However the complexity of the task was not considered sufficient to justify such an approach.

2.4.2 Proving Equivalence

We now turn our attention to proving that the new specification is equivalent to the original one. Here, by "equivalent" we mean that every model of the original specification is also a model of the modified specification, and vice-versa. Alternatively (but equivalently) we can think of it as each specification refining the other.

As the only modification we have made to our specification concerns dealing with compatibility, our equivalence condition will revolve around this part of the specification. In particular, we wish to prove that if two compatibility groups x and y are compatible in the original specification, then they are compatible in the modified specification, and vice-versa. Translating this into a predicate, our overall proof goal is:

$$
\begin{aligned}
&\forall cp : (CG \times CG)\textsf{-set}, x, y : CG \cdot \\
&\qquad \mathit{CgRel`compatible}(x, y, \mathit{CgRel`build\text{-}rel}(cp)) \\
&\Leftrightarrow \quad mk\text{-}(x, y) \in cp \vee mk\text{-}(y, x) \in cp
\end{aligned}
$$

Replacing $\mathit{CgRel'compatible}$ with its definition, and relaxing the notation a little, we get the following equivalent proof goal:

$$
\begin{aligned}
&\forall s : (CG \times CG)\textsf{-set}, x, y : CG \cdot \\
&\quad y \in \textsf{dom}\ \mathit{build\text{-}rel}(s) \wedge x \in \mathit{build\text{-}rel}(s)(y) \;\Leftrightarrow\; (mk\text{-}(x, y) \in s \vee mk\text{-}(y, x) \in s)
\end{aligned}
$$

We prove this equivalence in two stages: in stage 1, we prove the sub-goal

$$\forall s : (CG \times CG)\textsf{-set}, x, y : CG \cdot$$
$$(mk\text{-}(x, y) \in s \vee mk\text{-}(y, x) \in s) \;\Rightarrow\; y \in \textsf{dom}\; build\text{-}rel(s) \wedge x \in build\text{-}rel(s)(y).$$

Then in stage 2, we prove the converse, namely:

$$\forall s : (CG \times CG)\textsf{-set}, x, y : CG \cdot$$
$$y \in \textsf{dom}\; build\text{-}rel(s) \wedge x \in build\text{-}rel(s)(y) \;\Rightarrow\; (mk\text{-}(x, y) \in s \vee mk\text{-}(y, x) \in s).$$

Proof of Stage 1

If we define p as the following predicate:

$$\begin{array}{l} p : CG \times CG\textsf{-set} \to \mathbb{B} \\ p\,(s) \triangleq \\ \quad \forall x, y : CG \cdot mk\text{-}(x, y) \in s \;\Rightarrow\; \{x, y\} \subseteq \textsf{dom}\; build\text{-}rel(s) \wedge \\ \qquad\qquad\qquad\qquad\qquad\qquad x \in build\text{-}rel(s)(y) \wedge \\ \qquad\qquad\qquad\qquad\qquad\qquad y \in build\text{-}rel(s)(x) \end{array}$$

Then we prove by induction over s that

$$\forall s : (CG \times CG)\textsf{-set} \cdot p(s)$$

Separating out the two cases for the induction, we prove the following results:

1. $p(\{\,\})$
2. $\forall a, b : CG; s' : (CG \times CG)\textsf{-set}; p(s'); mk\text{-}(a, b) \notin s' \vdash p(add(mk\text{-}(a, b), s'))$

The first of these is trivial to prove, so assuming the proof of the second case, we get theorem 2 below:

```
from $s : CG \times CG\textsf{-set}$
1      from $x : CG; y : CG$
1.1          $mk\text{-}(x, y) \notin \{\}$                                          { }-is-empty(h1)
1.2          $\neg\, mk\text{-}(x, y) \in \{\}$                                      ∉-defn(1.1)
1.3          $\neg\, mk\text{-}(x, y) \in \{\} \vee$
             $(\{x, y\} \subseteq \textsf{dom}\; build\text{-}rel(\{\}) \wedge x \in build\text{-}rel(\{\})(y) \wedge$
             $y \in build\text{-}rel(\{\})(x))$                                     ∨-I-right(1.2)
1.4          $mk\text{-}(x, y) \in \{\} \Rightarrow$
             $(\{x, y\} \subseteq \textsf{dom}\; build\text{-}rel(\{\}) \wedge x \in build\text{-}rel(\{\})(y) \wedge$
             $y \in build\text{-}rel(\{\})(x))$                                     ⇒ -defn(1.3)
       infer $p(\{\})$                                                           p-defn(1.4)
2      from $a : CG; b : CG; s' : CG \times CG\textsf{-set}; p(s'); mk\text{-}(a, b) \notin s'$
       infer $p(add(mk\text{-}(a, b), s'))$                                      Theorem 1(h2)
infer $p(s)$                                                                     set-indn(1,2,h)
```

To prove the inductive step (theorem 1, shown in figure 2.5), we observe that we need to prove an implication. By the implication formation rule ($\Rightarrow$ -I), this means that we need to prove that we can infer the conclusion from the hypothesis, and also that the hypothesis is well-defined. The latter part is a straightforward application of type reasoning (lines 7 – 10). To prove the former, we need to prove

$$\begin{aligned}\forall x, y, a, b : CG; s' : CG \times CG\text{-}\mathsf{set}; p(s'); \mathit{mk}\text{-}(a, b) \notin s' \vdash \\ \{x, y\} \subseteq \mathsf{dom}\ \mathit{build}\text{-}\mathit{rel}(\mathit{add}(\mathit{mk}\text{-}(a, b), s')) \wedge \\ x \in \mathit{build}\text{-}\mathit{rel}(\mathit{add}(\mathit{mk}\text{-}(a, b), s'))(y) \wedge \\ y \in \mathit{build}\text{-}\mathit{rel}(\mathit{add}(\mathit{mk}\text{-}(a, b), s'))(x)\end{aligned}$$

We then distinguish two cases: $\mathit{mk}\text{-}(x, y) = \mathit{mk}\text{-}(a, b)$ (line 4) and $\mathit{mk}\text{-}(x, y) \in s'$ (line 5). We prove the conclusion in each of these cases, then infer that the conclusion must hold when $\mathit{mk}\text{-}(x, y) \in \mathit{add}(\mathit{mk}\text{-}(a, b), s')$ (line 6).

One point worthy of note is the use of lemma 1 on line 5.7: this is an example of what we might think of as a commuting result: having proved $t(x, y)$ we can infer $t(y, x)$. This greatly eases reasoning, and also helps the readability of proofs. The proof of lemma 1 itself is essentially just a re-arrangement of the type invariant:

```
from $cr : CompatRel; x : CG; y : CG; x \in \mathsf{dom}\ cr; y \in cr(x)$
1     $\forall a \in \mathsf{dom}\ cr \cdot \forall b \in cr(a) \cdot b \in \mathsf{dom}\ cr \wedge a \in cr(b)$          $inv\text{-}CompatRel$
2     $\forall b \in cr(x) \cdot b \in \mathsf{dom}\ cr \wedge x \in cr(b)$                                  $\forall$-E(1,h)
infer $y \in \mathsf{dom}\ cr \wedge x \in cr(y)$                                                  $\forall$-E(2,h)
```

Proof of Stage 2

We wish to prove

$$\forall s : (CG \times CG)\text{-}\mathsf{set}, x, y : CG \cdot \cdot \\ (y \in \mathsf{dom}\ \mathit{build}\text{-}\mathit{rel}(s) \wedge x \in \mathit{build}\text{-}\mathit{rel}(s)(y)) \Rightarrow (\mathit{mk}\text{-}(x, y) \in s \vee \mathit{mk}\text{-}(y, x) \in s).$$

One more, we perform induction over the set s. The base case is theorem 3 (shown in figure 2.6); the induction is performed in theorem 4 (figure 2.7).

This theorem has two parts: line 1 and line 2. Line 1 is the base case (proved above) and line 2 is the inductive case. For the inductive case, we wish to prove an implication of the form $A \Rightarrow B$, so we first prove $A \vdash B$ (line 2.1) then prove $\delta(A)$ (lines 2.2 – 2.8). We then infer $A \Rightarrow B$ by the implication-introduction rule ($\Rightarrow$ -I).

To prove

$$\begin{array}{lcl}(y \in \mathsf{dom}\ \mathit{build}\text{-}\mathit{rel}(\mathit{add}(\mathit{mk}\text{-}(a, b), s')) \wedge & \vdash & \mathit{mk}\text{-}(x, y) \in \mathit{add}(\mathit{mk}\text{-}(a, b), s') \vee \\ x \in \mathit{build}\text{-}\mathit{rel}(\mathit{add}(\mathit{mk}\text{-}(a, b), s'))(y)) & & \mathit{mk}\text{-}(y, x) \in \mathit{add}(\mathit{mk}\text{-}(a, b), s')\end{array}$$

from $x : CG;\ y : CG;\ a : CG;\ b : CG;\ s' : CG \times CG\text{-set};\ p(s');\ mk\text{-}(a, b) \notin s'$

1 $\{a, b\} \subseteq \mathsf{dom}\ build\text{-}rel(add(mk\text{-}(a, b), s)) \wedge$
$a \in build\text{-}rel(add(mk\text{-}(a, b), s))(b) \wedge$
$b \in build\text{-}rel(add(mk\text{-}(a, b), s))(a)$ Lemma 2(h)

2 $\forall\, x, y : CG \cdot mk\text{-}(x, y) \in s' \ \Rightarrow$
$(\{x, y\} \in \mathsf{dom}\ build\text{-}rel(s') \wedge x \in build\text{-}rel(s')(y) \wedge$
$y \in build\text{-}rel(s')(x))$ p-defn(h)

3 $mk\text{-}(x, y) \in s' \ \Rightarrow \ (\{x, y\} \in \mathsf{dom}\ build\text{-}rel(s') \wedge$
$x \in build\text{-}rel(s')(y) \wedge y \in build\text{-}rel(s')(x))$ $\forall$-E(h,2)

4 from $mk\text{-}(a, b) = mk\text{-}(x, y)$

4.1 $a = x \wedge b = y$ tuple-defn(h4)

infer $\{x, y\} \subseteq \mathsf{dom}\ build\text{-}rel(add(mk\text{-}(a, b), s')) \wedge$
$x \in build\text{-}rel(add(mk\text{-}(a, b), s'))(y) \wedge$
$y \in build\text{-}rel(add(mk\text{-}(a, b), s'))(x)$ =-subs(1,4.1)

5 from $mk\text{-}(x, y) \in s'$

5.1 $\{x, y\} \in \mathsf{dom}\ build\text{-}rel(s') \wedge$
$x \in build\text{-}rel(s')(y) \wedge y \in build\text{-}rel(s')(x)$ $\Rightarrow$ -E(h5,3)

5.2 $\mathsf{dom}\ build\text{-}rel(add(mk\text{-}(a, b), s')) =$
$\{a, b\} \cup \mathsf{dom}\ build\text{-}rel(s')$ Lemma 3(h)

5.3 $\{x, y\} \subseteq \{a, b\} \cup \mathsf{dom}\ build\text{-}rel(s')$ $\cup$-I-left-$\subseteq$(5.1)

5.4 $\{x, y\} \subseteq \mathsf{dom}\ build\text{-}rel(add(mk\text{-}(a, b), s'))$ =-subs(5.2,5.3)

5.5 $build\text{-}rel(s')(y) \subseteq build\text{-}rel(add(mk\text{-}(a, b), s'))(y)$
Lemma 4(5.1)

5.6 $x \in build\text{-}rel(add(mk\text{-}(a, b), s'))(y)$ $\subseteq$-I(5.1,5.5)

5.7 $y \in build\text{-}rel(add(mk\text{-}(a, b), s'))(x)$ Lemma 1(5.6)

infer $\{x, y\} \subseteq \mathsf{dom}\ build\text{-}rel(add(mk\text{-}(a, b), s')) \wedge$
$x \in build\text{-}rel(add(mk\text{-}(a, b), s'))(y) \wedge$
$y \in build\text{-}rel(add(mk\text{-}(a, b), s'))(x)$ $\wedge$-I(5.4,5.6,5.7)

6 from $mk\text{-}(x, y) \in add(mk\text{-}(a, b), s')$

6.1 $mk\text{-}(x, y) = mk\text{-}(a, b) \vee mk\text{-}(x, y) \in s'$ $\in$-add-E(h6)

infer $\{x, y\} \subseteq \mathsf{dom}\ build\text{-}rel(add(mk\text{-}(a, b), s')) \wedge$
$x \in build\text{-}rel(add(mk\text{-}(a, b), s'))(y) \wedge$
$y \in build\text{-}rel(add(mk\text{-}(a, b), s'))(x)$ $\vee$-E(6.1,4,5)

7 $mk\text{-}(a, b) : CG \times CG$ tuple-form(h)

8 $add(mk\text{-}(a, b), s') : CG \times CG\text{-set}$ add-form(7,h)

9 $mk\text{-}(x, y) : CG \times CG$ tuple-form(h)

10 $\delta(mk\text{-}(x, y) \in add(mk\text{-}(a, b), s'))$ δ-in-set(9,8)

infer $mk\text{-}(x, y) \in add(mk\text{-}(a, b), s') \ \Rightarrow$
$(\{x, y\} \subseteq \mathsf{dom}\ build\text{-}rel(add(mk\text{-}(a, b), s')) \wedge$
$x \in build\text{-}rel(add(mk\text{-}(a, b), s'))(y) \wedge$
$y \in build\text{-}rel(add(mk\text{-}(a, b), s'))(x))$ $\Rightarrow$ -I(6,10)

Figure 2.5: Proof of Theorem 1

```
from $x : CG; y : CG$
1      $build\text{-}rel(\{\}) =$
           if $\{\} = \{\}$ then $\{\}$
           else let $cr = build\text{-}rel(\{\})$ in
               let $new\text{-}cr = addCG(b, addCG(a, cr))$ in
                   $new\text{-}cr \dagger \{a \mapsto new\text{-}cr(a) \cup \{b\},$
                                 $b \mapsto new\text{-}cr(b) \cup \{a\}\}$          $build\text{-}rel$-defn(h)
2      if $\{\} = \{\}$ then $\{\}$
       else let $cr = build\text{-}rel(\{\})$ in
           let $new\text{-}cr = addCG(b, addCG(a, cr))$ in
               $new\text{-}cr \dagger \{a \mapsto new\text{-}cr(a) \cup \{b\},$
                             $b \mapsto new\text{-}cr(b) \cup \{a\}\} = \{\}$          condition-true(h)
3      $build\text{-}rel(\{\}) = \{\}$                                                 =-trans(1,2)
4      $\text{dom}\ \{\} = \{\}$                                                      dom -defn-$\{\mapsto\}$
5      $\text{dom}\ build\text{-}rel(\{\}) = \{\}$                                        =-subs(3,4)
6      $\neg\, y \in \{\}$                                                        { }-is-empty(h)
7      $\neg\, y \in \text{dom}\ build\text{-}rel(\{\})$                                   =-subs(5,6)
8      $\neg\,(y \in \text{dom}\ build\text{-}rel(\{\}) \wedge x \in build\text{-}rel(\{\})(y))$       $\neg$-$\wedge$-I-right(7)
9      $\neg\,(y \in \text{dom}\ build\text{-}rel(\{\}) \wedge x \in build\text{-}rel(\{\})(y)) \vee$
       $(mk\text{-}(x, y) \in \{\} \vee mk\text{-}(y, x) \in \{\})$                           $\vee$-I-right(8)
infer $(y \in \text{dom}\ build\text{-}rel(\{\}) \wedge x \in build\text{-}rel(\{\})(y)) \Rightarrow$
       $(mk\text{-}(x, y) \in \{\} \vee mk\text{-}(y, x) \in \{\})$                           $\Rightarrow$ -defn(9)
```

Figure 2.6: Proof of Theorem 3

we distinguish three cases: the two trivial ones in which $\{x, y\} = \{a, b\}$, and the third in which $mk\text{-}(x, y) \neq mk\text{-}(a, b) \wedge mk\text{-}(y, x) \neq mk\text{-}(a, b)$ (line 2.1.3). In the latter case the desired conclusion is yielded by Lemma 5, which we discuss below.

Finally, to prove $\delta(A)$, we explicitly construct A and use our type hypotheses to verify definedness at each stage of the construction. Thus we prove the required inference.

As stated above, the main result used in this induction is Lemma 5, shown in figure 2.8. The proof for this lemma proceeds in two parts. In the former (lines 1–11), x, y, a and b are manipulated until we are able to partition inequalities in the manner shown on line 11. This partition is then used in a case-by-case analysis (lines 12–15) using Lemmas 6–9. These lemmas, listed in appendix 2.7, are all technical results, and all appeal to the definition of *build-rel* for their proof. Details may be found in [7].

from $x : CG; y : CG; s : CG \times CG\text{-set}$

1 $y \in \text{dom } \mathit{build\text{-}rel}(\{\}) \land x \in \mathit{build\text{-}rel}(\{\})(y) \;\Rightarrow$
$(\mathit{mk}\text{-}(x,y) \in \{\} \lor \mathit{mk}\text{-}(y,x) \in \{\})$ Theorem 3(h)

2 from $a : CG; b : CG; s' : CG \times CG\text{-set}; \mathit{mk}\text{-}(a,b) \notin s';$
$y \in \text{dom } \mathit{build\text{-}rel}(s') \land x \in \mathit{build\text{-}rel}(s')(y) \;\Rightarrow$
$\mathit{mk}\text{-}(x,y) \in s' \lor \mathit{mk}\text{-}(y,x) \in s'$

2.1 from $y \in \text{dom } \mathit{build\text{-}rel}(\mathit{add}(\mathit{mk}\text{-}(a,b), s')) \land$
$x \in \mathit{build\text{-}rel}(\mathit{add}(\mathit{mk}\text{-}(a,b), s'))(y)$

2.1.1 from $\mathit{mk}\text{-}(x,y) = \mathit{mk}\text{-}(a,b)$

2.1.1.1 $\mathit{mk}\text{-}(x,y) \in \mathit{add}(\mathit{mk}\text{-}(a,b), s')$ $\in$-add-I-elem(h2.1.1)

infer $\mathit{mk}\text{-}(x,y) \in \mathit{add}(\mathit{mk}\text{-}(a,b), s') \lor$
$\mathit{mk}\text{-}(y,x) \in \mathit{add}(\mathit{mk}\text{-}(a,b), s')$ $\lor$-I-right(2.1.1.1)

2.1.2 from $\mathit{mk}\text{-}(y,x) = \mathit{mk}\text{-}(a,b)$

2.1.2.1 $\mathit{mk}\text{-}(y,x) \in \mathit{add}(\mathit{mk}\text{-}(a,b), s')$ $\in$-add-I-elem(h2.1.2)

infer $\mathit{mk}\text{-}(x,y) \in \mathit{add}(\mathit{mk}\text{-}(a,b), s') \lor$
$\mathit{mk}\text{-}(y,x) \in \mathit{add}(\mathit{mk}\text{-}(a,b), s')$ $\lor$-I-right(2.1.2.1)

2.1.3 from $\mathit{mk}\text{-}(x,y) \neq \mathit{mk}\text{-}(a,b) \land \mathit{mk}\text{-}(y,x) \neq \mathit{mk}\text{-}(a,b)$

2.1.3.1 $y \in \text{dom } \mathit{build\text{-}rel}(s') \land x \in \mathit{build\text{-}rel}(s')(y)$
Lemma 5(h,h2.1,h2.1.3)

2.1.3.2 $\mathit{mk}\text{-}(x,y) \in s' \lor \mathit{mk}\text{-}(y,x) \in s'$ $\Rightarrow$ -E(h2,2.1.3.1)

infer $\mathit{mk}\text{-}(x,y) \in \mathit{add}(\mathit{mk}\text{-}(a,b), s') \lor$
$\mathit{mk}\text{-}(y,x) \in \mathit{add}(\mathit{mk}\text{-}(a,b), s')$ $\in$-add-I-set-$\lor$(2.1.3.2)

infer $\mathit{mk}\text{-}(x,y) \in \mathit{add}(\mathit{mk}\text{-}(a,b), s') \lor$
$\mathit{mk}\text{-}(y,x) \in \mathit{add}(\mathit{mk}\text{-}(a,b), s')$ =-cases(2.1.1,2.1.2,2.1.3)

2.2 $\mathit{add}(\mathit{mk}\text{-}(a,b), s') : CG \times CG\text{-set}$ add-form(h2)

2.3 $\mathit{build\text{-}rel}(\mathit{add}(\mathit{mk}\text{-}(a,b), s')) : \mathit{CompatRel}$ $\mathit{build\text{-}rel}$-defn(2.2)

2.4 $\text{dom } \mathit{build\text{-}rel}(\mathit{add}(\mathit{mk}\text{-}(a,b), s')) : CG$ dom -form(2.3)

2.5 $\delta(y \in \text{dom } \mathit{build\text{-}rel}(\mathit{add}(\mathit{mk}\text{-}(a,b), s')))$ δ-in(2.4,h)

2.6 $\mathit{build\text{-}rel}(\mathit{add}(\mathit{mk}\text{-}(a,b), s'))(y) : CG\text{-set}$ $\mathit{CompatRel}$-defn(2.3)

2.7 $\delta(x \in \mathit{build\text{-}rel}(\mathit{add}(\mathit{mk}\text{-}(a,b), s'))(y))$ δ-in(2.6,h)

2.8 $\delta(y \in \text{dom } \mathit{build\text{-}rel}(\mathit{add}(\mathit{mk}\text{-}(a,b), s')) \land$
$x \in \mathit{build\text{-}rel}(\mathit{add}(\mathit{mk}\text{-}(a,b), s'))(y))$ δ-$\land$-inherit(2.5,2.7)

infer $(y \in \text{dom } \mathit{build\text{-}rel}(\mathit{add}(\mathit{mk}\text{-}(a,b), s')) \land$
$x \in \mathit{build\text{-}rel}(\mathit{add}(\mathit{mk}\text{-}(a,b), s'))(y)) \;\Rightarrow$
$(\mathit{mk}\text{-}(x,y) \in \mathit{add}(\mathit{mk}\text{-}(a,b), s') \lor$
$\mathit{mk}\text{-}(y,x) \in \mathit{add}(\mathit{mk}\text{-}(a,b), s'))$ $\Rightarrow$ -I(2.1,2.8)

infer $(y \in \text{dom } \mathit{build\text{-}rel}(s) \land x \in \mathit{build\text{-}rel}(s)(y)) \;\Rightarrow$
$\mathit{mk}\text{-}(x,y) \in s \lor \mathit{mk}\text{-}(y,x) \in s$ set-indn(1,2)

Figure 2.7: Proof of Theorem 4

from	$x : CG;\ y : CG;\ a : CG;\ b : CG;\ s : CG \times CG\text{-set};$	
	$y \in \mathsf{dom}\ \mathit{build\text{-}rel}(\mathit{add}(\mathit{mk\text{-}}(a, b), s));$	
	$x \in \mathit{build\text{-}rel}(\mathit{add}(\mathit{mk\text{-}}(a, b), s))(y);$	
	$\mathit{mk\text{-}}(x, y) \neq \mathit{mk\text{-}}(a, b) \wedge \mathit{mk\text{-}}(y, x) \neq \mathit{mk\text{-}}(a, b)$	
1	from $x = a \wedge y = b$	
	infer $\mathit{mk\text{-}}(x, y) = \mathit{mk\text{-}}(a, b)$	mk-defn(h,h1)
2	$\neg((x = a \wedge y = b) \wedge \mathit{mk\text{-}}(x, y) \neq \mathit{mk\text{-}}(a, b))$	$\neg$-$\wedge$-i-sqt(1)
3	$\neg(x = a \wedge y = b)$	$\neg$-$\wedge$-E-right(h,2)
4	$x \neq a \vee y \neq b$	$\wedge$-defn(3)
5	from $x = b \wedge y = a$	
	infer $\mathit{mk\text{-}}(y, x) = \mathit{mk\text{-}}(a, b)$	mk-defn(h,h5)
6	$\neg((x = b \wedge y = a) \wedge \mathit{mk\text{-}}(y, x) \neq \mathit{mk\text{-}}(a, b))$	$\neg$-$\wedge$-i-sqt(5)
7	$\neg(x = b \wedge y = a)$	$\neg$-$\wedge$-E-right(h,6)
8	$x \neq b \vee y \neq a$	$\wedge$-defn(7)
9	$(x \neq a \vee y \neq b) \wedge (x \neq b \vee y \neq a)$	$\wedge$-I(4,8)
10	$((x \neq a \vee y \neq b \wedge x \neq b) \vee$	
	$(x \neq a \vee y \neq b \wedge y \neq a))$	$\wedge$-$\vee$-dist-expand(9)
11	$(x \neq a \wedge x \neq b) \vee (y \neq b \wedge x \neq b) \vee$	
	$(x \neq a \wedge y \neq a) \vee (y \neq b \wedge y \neq a)$	$\wedge$-$\vee$-dist-expand(10)
12	from $x \neq a \wedge x \neq b$	
	infer $y \in \mathsf{dom}\ \mathit{build\text{-}rel}(s) \wedge x \in \mathit{build\text{-}rel}(s)(y)$	Lemma 6(h,h12)
13	from $y \neq b \wedge x \neq b$	
	infer $y \in \mathsf{dom}\ \mathit{build\text{-}rel}(s) \wedge x \in \mathit{build\text{-}rel}(s)(y)$	Lemma 7(h,h13)
14	from $x \neq a \wedge y \neq a$	
	infer $y \in \mathsf{dom}\ \mathit{build\text{-}rel}(s) \wedge x \in \mathit{build\text{-}rel}(s)(y)$	Lemma 8(h,h14)
15	from $y \neq b \wedge y \neq a$	
	infer $y \in \mathsf{dom}\ \mathit{build\text{-}rel}(s) \wedge x \in \mathit{build\text{-}rel}(s)(y)$	Lemma 9(h,h15)
infer	$y \in \mathsf{dom}\ \mathit{build\text{-}rel}(s) \wedge x \in \mathit{build\text{-}rel}(s)(y)$	$\vee$-E(11,12,13,14,15)

Figure 2.8: Proof of Lemma 5

Summary

Having proved the equivalence of the separated specification and the original specification, it is useful to step back and consider what we have achieved. At the outset, we suggested that we wished to modify the specification, then "demonstrate that the [modified] specification is equivalent to the original one." At this stage the notion of equivalence is left relatively vague and is only really detailed in section 2.4.2. Thus while we were modifying the specification, we were relying on some intuitive notion of equivalence, which (as it turned out) was sufficient. However in other cases formalizing our equivalence requirement then attempting to prove equivalence might lead to deficiencies in the specification being highlighted. This is not unexpected, as we can not always expect our intuition to be sufficiently precise to match the degree of rigour demonstrated in the proofs above. Nor is this unwelcome, for as described in [6], undertaking proof, whether successful or otherwise, gives us deeper insight into the underlying domain, in this case our specification.

Turning to the proofs themselves, the complete proofs run to nearly thirty pages, which is surprising given the relatively modest goal, notwithstanding the degree of formality used. However much of this is relatively straightforward symbolic manipulation, which could easily be automated. Somewhat more surprising is the proportion of the proof which was simple case analysis, as exemplified by the proofs presented here. On reflection, this case analysis reflects the structure of the specification, as there we see a number of conditional expressions. In fact it is interesting to see the degree to which the structure of the proofs is dictated by the structure of the specification; conditional expressions give rise to proofs based around case analysis, and recursive functions lead to inductive proofs. This emphasizes the relationship between specifications and proofs, in that when proving a property of the specification, we use the structure of the specification to structure the proof; the success (or otherwise) of the proof can then feed back into the specification.

2.5 Discussion

Our objective in this chapter has been to demonstrate the use of a variety of proof techniques in two scenarios: as a tool for ensuring the well-formedness of a specification; and as a mechanism for performing design in a rigorous manner. Thus we have proved the satisfiability of an operation, and we have proved the correctness of a design decomposition. Although we have used the specific example of the Ammunition Control System, the approach and techniques are not specific to this example, nor even to VDM-SL itself. The underlying theme throughout has been that proof deepens our understanding of specifications. We can draw an analogy with [6], where it is argued that proof deepens our understanding of theorems, and eventually improves them. Here, though our objects of interest differ (specifications rather than theorems) the same conclusion can be drawn.

One question that arose in this (and previous) exercises, is the extent to which we should alter the specification to simplify proofs. That is, some expressions give rise to simpler proof obligations than other expressions, which in a given context are equivalent. In this situation it seems that we can safely replace the original expressions with new ones, simplifying our proof obligations whilst leaving the meaning of the specification unchanged. However it is important to bear in mind that the specification document is meant as a communication medium between the various parties involved in the construction of a system. Such modification whilst preserving the meaning of the specification, need not preserve the clarity nor even the style of the original document. Moreover, as the number of such modifications increases, it becomes more tempting to accept the equivalence of the new specification with the old, with less rigour. Thus we argue that such modifications should be kept to a minimum, and, when used, should always be justified formally.

A problem that arose concerned the size and intricacy of the proofs described in section 2.4, which, as stated there, occupy nearly 30 pages. As the use of lemmas exploiting similar structure in distinct portions of the proof increased, so the relationship between proofs and the overall result weakened. Eventually the issue that two lemmas might be mutually dependent was confronted, as it was not obvious that this was not the case. Verifying that such mutual dependencies do not exist is both non-trivial and prone to errors when performed by hand. Therefore a simple tool was constructed to analyse proofs: for a given collection of proofs, it parses each proof then forms a list of the results (theorems and lemmas) on which this proof depends. A directed graph is then constructed in which nodes are results and arcs denote dependence of the proof of a result. This graph can be tested for cycles using standard algorithms. In this case no such cycles were found, so we can be confident that there is no circular reasoning in our proofs. However in the first instance a number of results were discovered upon which no other results depended! Thus this tool undoubtedly improved the quality of the proofs themselves, and therefore increased our confidence in the proofs.

Is there any place for hand proofs today, given the availability of powerful theorem provers such as PVS and Isabelle? (See Chapters 6 and 7. In an ideal world, no! As hinted above, a large proportion of the tasks associated with proof are better performed mechanically. However in practice this is not always possible, for a number of reasons. For instance, we might be using domain-specific extensions to the formal notation that are not supported by the proof tool. In such situations knowledge of how to perform hand proof is vital. Moreover at present, theorem provers are only able to prove non-trivial results when directed by a human user, so even using such tools, knowledge and understanding of the craft of proof is invaluable.

2.6 Bibliography

[1] J.C. Bicarregui, J.S. Fitzgerald, P.A. Lindsay, R. Moore, and B. Ritchie. *Proof in VDM: A Practitioner's Guide.* Springer-Verlag, 1994.

[2] Committee of Experts on the Transport of Dangerous Goods, New York. *Recommendations on the Transport of Dangerous Goods*, 5th revised edition,

1988.

[3] R. Elmstrøm, P.G. Larsen, and P.B. Lassen. The IFAD VDM-SL Toolbox: A Practical Approach to Formal Specifications. *ACM Sigplan Notices*, 29(9), 1994.

[4] J. S. Fitzgerald. A proof of Satisfiability in Mukherjee and Stavridou's Ammunition Control System. Technical Report 616, Dept. of Computing Science, University of Newcastle upon Tyne, Newcastle upon Tyne, NE1 7RU, UK, 1997.

[5] J. Goguen and T. Winkler. Introducing OBJ3. Technical Report SRI-CSL-88-9, SRI International, August 1988.

[6] I. Lakatos. *Proofs and Refutations*. Cambridge University Press, 1976.

[7] P. Mukherjee. Proof of Equivalence in the ACS specification. Technical Report SCS 97.34, University of Leeds, 1997.

[8] P. Mukherjee and V. Stavridou. The Formal Specification of Safety Requirements for Storing Explosives. *Formal Aspects of Computing*, 5(4):299–336, 1993.

2.7 Auxiliary Results

This section gives the statements of the results used in the equivalence proof in Section 2.4. The proofs of these results may be found in [7].

Lemma 2

from $a : CG; b : CG; s : CG \times CG\text{-set}$
infer $\{a, b\} \subseteq \mathsf{dom}\ build\text{-}rel(add(mk\text{-}(a, b), s)) \wedge$
$a \in build\text{-}rel(add(mk\text{-}(a, b), s))(b) \wedge$
$b \in build\text{-}rel(add(mk\text{-}(a, b), s))(a)$

Lemma 3

from $a : CG; b : CG; s : CG \times CG\text{-set}$
infer $\mathsf{dom}\ build\text{-}rel(add(mk\text{-}(a, b), s)) = \{a, b\} \cup \mathsf{dom}\ build\text{-}rel(s)$

Lemma 4

from $a : CG; b : CG; y : CG; s : CG \times CG\text{-set}; y \in \mathsf{dom}\ \mathit{build\text{-}rel}(s)$
infer $\mathit{build\text{-}rel}(s)(y) \subseteq \mathit{build\text{-}rel}(\mathit{add}(\mathit{mk\text{-}}(a, b), s))(y)$

Lemma 5

from $a : CG; b : CG; c : CG; d : CG; s : CG \times CG\text{-set};$
$b \in \mathsf{dom}\ \mathit{build\text{-}rel}(\mathit{add}(\mathit{mk\text{-}}(c, d), s));$
$a \in \mathit{build\text{-}rel}(\mathit{add}(\mathit{mk\text{-}}(c, d), s))(b);$
$\mathit{mk\text{-}}(a, b) \neq \mathit{mk\text{-}}(c, d) \wedge \mathit{mk\text{-}}(b, a) \neq \mathit{mk\text{-}}(c, d)$
infer $b \in \mathsf{dom}\ \mathit{build\text{-}rel}(s) \wedge a \in \mathit{build\text{-}rel}(s)(b)$

Lemma 6

from $a : CG; b : CG; c : CG; d : CG; s : CG \times CG\text{-set};$
$b \in \mathsf{dom}\ \mathit{build\text{-}rel}(\mathit{add}(\mathit{mk\text{-}}(c, d), s));$
$a \in \mathit{build\text{-}rel}(\mathit{add}(\mathit{mk\text{-}}(c, d), s))(b); a \neq c; a \neq d$
infer $b \in \mathsf{dom}\ \mathit{build\text{-}rel}(s) \wedge a \in \mathit{build\text{-}rel}(s)(b)$

Lemma 7

from $a : CG; b : CG; c : CG; d : CG; s : CG \times CG\text{-set};$
$b \in \mathsf{dom}\ \mathit{build\text{-}rel}(\mathit{add}(\mathit{mk\text{-}}(c, d), s));$
$a \in \mathit{build\text{-}rel}(\mathit{add}(\mathit{mk\text{-}}(c, d), s))(b); b \neq d; a \neq d$
infer $b \in \mathsf{dom}\ \mathit{build\text{-}rel}(s) \wedge a \in \mathit{build\text{-}rel}(s)(b)$

Lemma 8

from $a : CG; b : CG; c : CG; d : CG; s : CG \times CG\text{-set};$
$b \in \mathsf{dom}\ \mathit{build\text{-}rel}(\mathit{add}(\mathit{mk\text{-}}(c, d), s));$
$a \in \mathit{build\text{-}rel}(\mathit{add}(\mathit{mk\text{-}}(c, d), s))(b); a \neq c; b \neq c$
infer $b \in \mathsf{dom}\ \mathit{build\text{-}rel}(s) \wedge a \in \mathit{build\text{-}rel}(s)(b)$

Lemma 9

from $a : CG; b : CG; c : CG; d : CG; s : CG \times CG\text{-set};$
$\quad b \in \mathsf{dom}\ build\text{-}rel(add(mk\text{-}(c, d), s));$
$\quad a \in build\text{-}rel(add(mk\text{-}(c, d), s))(b); b \neq d; b \neq c$
infer $b \in \mathsf{dom}\ build\text{-}rel(s) \wedge a \in build\text{-}rel(s)(b)$

Index