# H PROVING TAMED EFFECTS FOR THE SHOP/ACCOUNT EXAMPLE

In Section 2 we introduced a `Shop` that allows clients to make purchases through the `buy` method. The body if this method includes a method call to an unknown external object (`buyer.pay(...)`).

In this section we use our Hoare logic from Section 8 to outline the proof that the `buy` method does not expose the `Shop`'s `Account`, its `Key`, or allow the `Account`'s balance to be illicitly modified.

We outline the proof that $M_{good} \vdash S_2$, and that $M_{fine} \vdash S_2$. We also show why $M_{bad} \nvdash S_2$.

We first extend the semantics and the logic to deal with scalars (§H.1). We then extend the Hoare Logic with rules for conditionals, case analysis, and a contradiction rule (§H.2). We then rewrite the code of $M_{good}$ and so $M_{fine}$ so that it adheres to the syntax as defined in Fig. 9 (§H.3). We extend the specification $S_2$, so that is also makes a specification for the private method `set` (§H.4). After that, we outline the proofs (§H.6) – these proofs have been mechanized in Coq, and the source code will be submitted as an artefact. Finally, we discuss why $M_{bad} \nvdash S_2$ (§??).

## H.1 Extend the semantics and Hoare logic to accommodate scalars and conditionals

We extend the notion of protection to also allow it to apply to scalars.

**Definition H.1** (Satisfaction of Assertions – Protected From)**.** extending the definition of Def 5.4. We use $\alpha$ to range over addresses, $\beta$ to range over scalars, and $\gamma$ to range over addresses or scalars. We define $M, \sigma \models \langle\gamma\rangle \not\leftrightarrow \gamma_o$ as:

(1) $M, \sigma \models \langle\alpha\rangle \not\leftrightarrow \alpha_o \quad \triangleq$
- $\alpha \neq \alpha_0$, and
- $\forall n \in \mathbb{N}. \forall f_1, ... f_n..[\ \lfloor \alpha_o.f_1...f_n \rfloor_\sigma = \alpha \quad \Longrightarrow \quad M, \sigma \models \lfloor \alpha_o.f_1...f_{n-1} \rfloor_\sigma : C \ \wedge \ C \in M\ ]$

(2) $M, \sigma \models \langle\gamma\rangle \not\leftrightarrow \beta_o \quad \triangleq \quad true$

(3) $M, \sigma \models \langle\beta\rangle \not\leftrightarrow \alpha_o \quad \triangleq \quad false$

(4) $M, \sigma \models \langle e\rangle \not\leftrightarrow e_o \quad \triangleq$
$\exists \gamma, \gamma_o.[\ M, \sigma, e \hookrightarrow \gamma\ \wedge M, \sigma, e_0 \hookrightarrow \gamma_0\ \wedge\ M, \sigma \models \langle\gamma\rangle \not\leftrightarrow \gamma_o\ ]$

The definition from above gives rise to further cases of protection; we supplement the triples from Fig. 6 with some further inference rules, given in Fig. ??.

$$M \vdash x : \texttt{int} \rightarrow \langle y\rangle \not\leftrightarrow x \quad [\textsc{Prot-Int}] \qquad\qquad M \vdash x : \texttt{bool} \rightarrow \langle y\rangle \not\leftrightarrow x \quad [\textsc{Prot-Bool}]$$

$$M \vdash x : \texttt{str} \rightarrow \langle y\rangle \not\leftrightarrow x \quad [\textsc{Prot-Str1}] \qquad\qquad M \vdash \langle e\rangle \not\leftrightarrow e' \rightarrow e \neq e' \quad [\textsc{Prot-Neq}]$$

Fig. 15. Protection for Scalar Types

## H.2 More Hoare logic rules

We now extend the Hoare Logic with rules for conditionals, case analysis, and a contradiction rule. These are in Fig. 16, where we expect the obvious syntax and semantics for *Cond*.

## H.3 Expressing the `Shop` example in the syntax from Fig. 9

We now express our example in the syntax of Fig. 9. For this, we add a return type to each of the methods; We turn all local variables to parameter; We add an explicit assignment to the variable `res`: and We add a temporary variable `tmp` to which we assign the result of our `void` methods. For simplicity, we allow the shorthands += and −=. And we also allow definition of local variables, *e.g.* `int price := ..`

$$\frac{M \vdash \{\ A \wedge Cond\ \}\ stmt_1\ \{\ A'\ \}\ \parallel\ \{\ A''\ \} \qquad M \vdash \{\ A \wedge \neg Cond\ \}\ stmt_2\ \{\ A'\ \}\ \parallel\ \{\ A''\ \}}{M \vdash \{\ A\ \}\ \texttt{if}\ Cond\ \texttt{then}\ stmt_1\ \texttt{else}\ stmt_2\ \{\ A'\ \}\ \parallel\ \{\ A''\ \}}\ [\textsc{If\_Rule}]$$

$$\frac{}{M \vdash \{\ false\ \}\ stmt\ \{\ A'\ \}\ \parallel\ \{\ A''\ \}}\ [\textsc{Absurd}]$$

$$\frac{M \vdash \{\ A \wedge A_1\ \}\ stmt\ \{\ A'\ \}\ \parallel\ \{\ A''\ \} \qquad M \vdash \{\ A \wedge A_2\ \}\ stmt\ \{\ A'\ \}\ \parallel\ \{\ A''\ \}}{M \vdash \{\ A \wedge (A_1 \vee A_2)\ \}\ stmt\ \{\ A'\ \}\ \parallel\ \{\ A''\ \}}\ [\textsc{Cases}]$$

Fig. 16. Hoare Quadruple for conditionals, and more Substructural Hoare Quadruples

```
module M_good
  ...
  class Shop
    field accnt : Account,
    field invntry : Inventory,
    field clients: ..

    public method buy(buyer:external, anItem:Item, price: int,
            myAccnt: Account, oldBalance: int, newBalance: int, tmp:int) : int
      price := anItem.price;
      myAccnt := this.accnt;
      oldBalance := myAccnt.blnce;
      tmp := buyer.pay(myAccnt, price)     // external call!
      newBalance := myAccnt.blnce;
      if (newBalance == oldBalance+price) then
          tmp := this.send(buyer,anItem)
      else
         tmp := buyer.tell("you have not paid me") ;
      res := 0

      private method send(buyer:external, anItem:Item) : int
       ...
  class Account
    field blnce : int
    field key : Key

    public method transfer(dest:Account, key':Key, amt:nat) :int
      if (this.key==key') then
        this.blnce-=amt;
        dest.blnce+=amt
      else
        res := 0
      res := 0

     public method set(key':Key) : int
      if (this.key==null)  then
            this.key:=key'
      else
        res := 0
      res := 0
```

Remember that $M_{fine}$ is identical to $M_{good}$, except for the method `set`. We describe the module below.

```
module M_fine
  ...
  class Shop
     ...  as in M_good
  class Account
    field blnce : int
    field key : Key

    public method transfer(dest:Account, key':Key, amt:nat) :int
      ...  as in M_good

     public method set(key':Key, k'':Key) : int
      if (this.key==key')  then
            this.key:=key''
      else
        res := 0
      res := 0
```

### H.4 Proving that $M_{good}$ and $M_{fine}$ satisfy $S_2$

We redefine $S_2$ so that it also describes the behaviour of method `send`. and have:

$$
\begin{aligned}
S_{2a} \;&\triangleq\; \{\, \texttt{a} : \texttt{Account} \wedge \texttt{e} : \texttt{external} \wedge \langle\!\langle \texttt{a.key} \rangle\!\rangle \not\leftrightarrow \texttt{e} \,\} \\
&\qquad \texttt{private Shop} :: \texttt{send}(\texttt{buyer} : \texttt{external}, \texttt{anItem} : \texttt{Item}) \\
&\quad \{\, \langle\!\langle \texttt{a.key} \rangle\!\rangle \not\leftrightarrow e \,\} \;\|\; \{ \langle\!\langle \texttt{a.key} \rangle\!\rangle \not\leftrightarrow e \} \\
S_{2b} \;&\triangleq\; \{\, \texttt{a} : \texttt{Account} \wedge \texttt{a.blnce} = \texttt{b} \,\} \\
&\qquad \texttt{private Shop} :: \texttt{send}(\texttt{buyer} : \texttt{external}, \texttt{anItem} : \texttt{Item}) \\
&\quad \{\, \texttt{a.blnce} = \texttt{b} \,\} \;\|\; \{ \texttt{a.blnce} = \texttt{b} \} \\
S_{2,strong} \;&\triangleq\; S_2 \;\wedge\; S_{2a} \;\wedge\; S_{2b}
\end{aligned}
$$

For brevity we only show that `buy` satisfies our scoped invariants, as the all other methods of the $\texttt{M}_{good}$ interface are relatively simple, and do not make any external calls.

To write our proofs more succinctly, we will use `ClassId::methId.body` as a shorthand for the method body of `methId` defined in `ClassId`.

**Lemma H.2** ($M_{good}$ satisfies $S_{2,strong}$). $M_{good} \vdash S_{2,strong}$

Proof Outline In order to prove that

$$M_{good} \vdash \forall \texttt{a} : \texttt{Account}.\{\langle\!\langle \texttt{a.key} \rangle\!\rangle\}$$

we have to apply Invariant from Fig. 8. That is, for each class $C$ defined in $M_{good}$, and for each public method $m$ in $C$, with parameters $\overline{y : D}$, we have to prove that

$$
\begin{aligned}
M_{good} \;\vdash\; &\{\, \texttt{this} : \texttt{C}, \overline{y : D}, \texttt{a} : \texttt{Account} \;\wedge\; \langle\!\langle \texttt{a.key} \rangle\!\rangle \;\wedge\; \langle\!\langle \texttt{a.key} \rangle\!\rangle \not\leftrightarrow (\texttt{this}, \overline{y}) \,\} \\
&\qquad\qquad \texttt{C} :: \texttt{m.body} \\
&\quad \{\, \langle\!\langle \texttt{a.key} \rangle\!\rangle \;\wedge\; \langle\!\langle \texttt{a.key} \rangle\!\rangle \not\leftrightarrow \texttt{res} \,\} \;||\; \{\, \langle\!\langle \texttt{a.key} \rangle\!\rangle \,\}
\end{aligned}
$$

Thus, we need to prove three Hoare quadruples: one for `Shop::buy`, one for `Account::transfer`, and one for `Account::set`. That is, we have to prove that

$$
\begin{array}{ll}
(1?) & M_{good} \vdash \{\, \mathtt{A}_{buy},\ \mathtt{a}:\mathtt{Account} \wedge \langle\!\langle \mathtt{a.key} \rangle\!\rangle \wedge \langle\!\langle \mathtt{a.key} \rangle\!\rangle \not\leftrightarrow \mathtt{Ids}_{buy} \,\} \\
& \qquad\qquad \mathtt{Shop :: buy.body} \\
& \qquad \{\langle\!\langle \mathtt{a.key} \rangle\!\rangle \wedge \langle\!\langle \mathtt{a.key} \rangle\!\rangle \mathbin{-\!\nabla} \mathtt{res}\} \ \ || \ \ \{\langle\!\langle \mathtt{a.key} \rangle\!\rangle\} \\
(2?) & M_{good} \vdash \{\, \mathtt{A}_{trns},\ \mathtt{a}:\mathtt{Account} \wedge \langle\!\langle \mathtt{a.key} \rangle\!\rangle \wedge \langle\!\langle \mathtt{a.key} \rangle\!\rangle \not\leftrightarrow \mathtt{Ids}_{trns} \,\} \\
& \qquad\qquad \mathtt{Account :: transfer.body} \\
& \qquad \{\langle\!\langle \mathtt{a.key} \rangle\!\rangle \wedge \langle\!\langle \mathtt{a.key} \rangle\!\rangle \mathbin{-\!\nabla} \mathtt{res}\} \ \ || \ \ \{\langle\!\langle \mathtt{a.key} \rangle\!\rangle\} \\
(3?) & M_{good} \vdash \{\, \mathtt{A}_{set},\ \mathtt{a}:\mathtt{Account} \wedge \langle\!\langle \mathtt{a.key} \rangle\!\rangle \wedge \langle\!\langle \mathtt{a.key} \rangle\!\rangle \not\leftrightarrow \mathtt{Ids}_{set} \,\} \\
& \qquad\qquad \mathtt{Account :: set.body} \\
& \qquad \{\langle\!\langle \mathtt{a.key} \rangle\!\rangle \wedge \langle\!\langle \mathtt{a.key} \rangle\!\rangle \mathbin{-\!\nabla} \mathtt{res}\} \ \ || \ \ \{\langle\!\langle \mathtt{a.key} \rangle\!\rangle\}
\end{array}
$$

where we are using ? to indicate that this needs to be proven, and where we are using the shorthands

$$
\begin{array}{lcl}
\mathtt{A}_{buy} & \triangleq & \mathtt{this:Shop, buyer:external, anItem:Item, price:int,} \\
& & \mathtt{myAccnt:Account, oldBalance:int, newBalance:int, tmp:int.} \\
\mathtt{Ids}_{buy} & \triangleq & \mathtt{this, buyer, anItem, price, myAccnt, oldBalance, newBalance, tmp.} \\
\mathtt{A}_{trns} & \triangleq & \mathtt{this:Account, dest:Account, key':Key, amt:nat} \\
\mathtt{Ids}_{trns} & \triangleq & \mathtt{this, dest, key', amt} \\
\mathtt{A}_{set} & \triangleq & \mathtt{this:Account, key':Key, key'':Key.} \\
\mathtt{Ids}_{set} & \triangleq & \mathtt{this, key', key''.}
\end{array}
$$

We will also need to prove that `Send` satisfies specifications $S_{2a}$ and $S_{2b}$.

We outline the proof of (1?) in Lemma H.4,and the proof of (2) in Lemma H.5. We do not prove (3), but will prove that `set` from $M_{fine}$ satisfies $S_2$; shown in Lemma H.6 – ie for module $M_{fine}$. ☐

We also want to prove that $M_{fine}$ satisfies the specification $S_{2,strong}$.

**Lemma H.3** ($M_{fine}$ satisfies $S_{2,strong}$). $M_{fine} \vdash S_{2,strong}$

Proof Outline The proof of

$$M_{fine} \vdash \forall \mathtt{a}:\mathtt{Account}.\{\langle\!\langle \mathtt{a.key} \rangle\!\rangle\}$$

goes along similar lines to the proof of lemma H.2. Thus, we need to prove the following three Hoare quadruples:

$$
\begin{array}{ll}
(4?) & M_{fine} \vdash \{\, \mathtt{A}_{buy},\ \mathtt{a}:\mathtt{Account} \wedge \langle\!\langle \mathtt{a.key} \rangle\!\rangle \wedge \langle\!\langle \mathtt{a.key} \rangle\!\rangle \not\leftrightarrow \mathtt{Ids}_{buy} \,\} \\
& \qquad\qquad \mathtt{Shop :: buy.body} \\
& \qquad \{\langle\!\langle \mathtt{a.key} \rangle\!\rangle \wedge \langle\!\langle \mathtt{a.key} \rangle\!\rangle \mathbin{-\!\nabla} \mathtt{res}\} \ \ || \ \ \{\langle\!\langle \mathtt{a.key} \rangle\!\rangle\} \\
(5?) & M_{fine} \vdash \{\, \mathtt{A}_{trns},\ \mathtt{a}:\mathtt{Account} \wedge \langle\!\langle \mathtt{a.key} \rangle\!\rangle \wedge \langle\!\langle \mathtt{a.key} \rangle\!\rangle \not\leftrightarrow \mathtt{Ids}_{trns} \,\} \\
& \qquad\qquad \mathtt{Account :: transfer.body} \\
& \qquad \{\langle\!\langle \mathtt{a.key} \rangle\!\rangle \wedge \langle\!\langle \mathtt{a.key} \rangle\!\rangle \mathbin{-\!\nabla} \mathtt{res}\} \ \ || \ \ \{\langle\!\langle \mathtt{a.key} \rangle\!\rangle\} \\
(6?) & M_{fine} \vdash \{\, \mathtt{A}_{set},\ \mathtt{a}:\mathtt{Account} \wedge \langle\!\langle \mathtt{a.key} \rangle\!\rangle \wedge \langle\!\langle \mathtt{a.key} \rangle\!\rangle \not\leftrightarrow \mathtt{Ids}_{set} \,\} \\
& \qquad\qquad \mathtt{Account :: set.body} \\
& \qquad \{\langle\!\langle \mathtt{a.key} \rangle\!\rangle \wedge \langle\!\langle \mathtt{a.key} \rangle\!\rangle \mathbin{-\!\nabla} \mathtt{res}\} \ \ || \ \ \{\langle\!\langle \mathtt{a.key} \rangle\!\rangle\}
\end{array}
$$

The proof of (4?) is identical to that of (1?); the proof of (5?) is identical to that of (2?). We outline the proof (6?) in Lemma H.6 in §H.4. ☐

**Lemma H.4** ($\texttt{Shop::buy}$ satisfies $S_2$).

$$(1) \quad M_{good} \vdash \{\ \mathtt{A}_{buy}\ \mathtt{a}:\mathtt{Account}\ \wedge\ \langle\!\langle \mathtt{a.key} \rangle\!\rangle\ \wedge\ \langle\!\langle \mathtt{a.key} \rangle\!\rangle \not\leftrightarrow \mathtt{Ids}_{buy}\ \}$$
$$\mathtt{Shop::buy.body}$$
$$\{\langle\!\langle \mathtt{a.key} \rangle\!\rangle\ \wedge\ \langle\!\langle \mathtt{a.key} \rangle\!\rangle \not\!\triangledown\, \mathtt{res}\}\ \ ||\ \ \{\langle\!\langle \mathtt{a.key} \rangle\!\rangle\}$$

Proof Outline We will use the shorthand $\mathtt{A}_1 \triangleq \mathtt{A}_{buy},\ \mathtt{a}:\mathtt{Account}$. We will split the proof into 1) proving that statements 10, 11, 12 preserve the protection of `a.key` from the `buyer`, 2) proving that the external call

**1st Step: proving statements 10, 11, 12**

We apply the underlying Hoare logic and prove that the statements on lines 10, 11, 12 do not affect the value of `a.key`, ie that for a $z \notin \{\mathtt{price}, \mathtt{myAccnt}, \mathtt{oldBalance}\}$, we have

$$(10) \quad M_{good} \vdash_{ul} \{\ \mathtt{A}_1\ \wedge\ z = \mathtt{a.key}\}$$
$$\mathtt{price:=anItem.price;}$$
$$\mathtt{myAccnt:=this.accnt;}$$
$$\mathtt{oldBalance\ :=\ myAccnt.blnce;}$$
$$\{z = \mathtt{a.key}\}$$

We then apply Embed_UL, Prot-1 and Prot-2 and Combine and and Types-2 on (10) and use the shorthand $\mathtt{stmts}_{10,11,12}$ for the statements on lines 10, 11 and 12, and obtain:

$$(11) \quad M_{good} \vdash \{\ \mathtt{A}_1\ \wedge\ \langle\!\langle \mathtt{a.key} \rangle\!\rangle\ \wedge\ \langle\!\langle \mathtt{buyer} \rangle\!\rangle \not\leftrightarrow \mathtt{a.key}\}$$
$$\mathtt{stmts}_{10,11,12}$$
$$\{\ \langle\!\langle \mathtt{a.key} \rangle\!\rangle\ \wedge\ \langle\!\langle \mathtt{buyer} \rangle\!\rangle \not\leftrightarrow \mathtt{a.key}\}$$

We apply Mid on (11) and obtain

$$(12) \quad M_{good} \vdash \{\ \mathtt{A}_1\ \wedge\ \langle\!\langle \mathtt{a.key} \rangle\!\rangle \not\leftrightarrow \mathtt{buyer}\ \}$$
$$\mathtt{stmts}_{10,11,12}$$
$$\{\ \mathtt{A}_1\ \wedge\ \langle\!\langle \mathtt{a.key} \rangle\!\rangle\ \wedge\ \langle\!\langle \mathtt{buyer} \rangle\!\rangle \not\leftrightarrow \mathtt{a.key}\ \}\ ||$$
$$\{\ \langle\!\langle \mathtt{a.key} \rangle\!\rangle\ \}$$

**2nd Step: Proving the External Call**

We now need to prove that the external method call `buyer.pay(this.accnt, price)` protects the `key`. i.e.

$$(13?) \quad M_{good} \vdash \{\ \mathtt{A}_1\ \wedge\ \langle\!\langle \mathtt{a.key} \rangle\!\rangle,\ \wedge\ \langle\!\langle \mathtt{a.key} \rangle\!\rangle \not\leftrightarrow \mathtt{buyer}\}$$
$$\mathtt{tmp\ :=\ buyer.pay(myAccnt,\ price)}$$
$$\{\ \ \mathtt{A}_1\ \wedge\ \langle\!\langle \mathtt{a.key} \rangle\!\rangle\ \wedge\ \langle\!\langle \mathtt{buyer} \rangle\!\rangle \not\leftrightarrow \mathtt{a.key}\}\ \ ||$$
$$\{\ \langle\!\langle \mathtt{a.key} \rangle\!\rangle\ \}$$

We use that $M \vdash \forall \texttt{a} : \texttt{Account}.\{\langle\!\langle \texttt{a.key} \rangle\!\rangle\}$ and obtain

$$
\begin{array}{ll}
(14) & M_{good} \vdash \{\ \texttt{buyer} : \texttt{external},\ \langle\!\langle \texttt{a.key} \rangle\!\rangle \wedge \langle\!\langle \texttt{a.key} \rangle\!\rangle \not\leftrightarrow (\texttt{buyer}, \texttt{myAccnt}, \texttt{price})\ \} \\
& \qquad\qquad\qquad \texttt{tmp := buyer.pay(myAccnt, price)} \\
& \qquad\quad \{\ \langle\!\langle \texttt{a.key} \rangle\!\rangle \wedge \langle\!\langle \texttt{a.key} \rangle\!\rangle \not\leftrightarrow (\texttt{buyer}, \texttt{myAccnt}, \texttt{price})\ \} \ \ || \\
& \qquad\quad \{\ \langle\!\langle \texttt{a.key} \rangle\!\rangle\ \}
\end{array}
$$

In order to obtain (13?) out of (14), we apply Prot-Intl and Prot-Int$_1$, which gives us

(15) $\quad M_{good} \vdash \texttt{A}_1 \wedge \langle\!\langle \texttt{a.key} \rangle\!\rangle \longrightarrow \langle\!\langle \texttt{a.key} \rangle\!\rangle \not\leftrightarrow \texttt{myAccnt}$

(16) $\quad M_{good} \vdash \texttt{A}_1 \wedge \langle\!\langle \texttt{a.key} \rangle\!\rangle \longrightarrow \langle\!\langle \texttt{a.key} \rangle\!\rangle \not\leftrightarrow \texttt{price}$

We apply Consequ on (15), (16) and (14) and obtain (13)!

□

**Lemma H.5** ( `transfer` satisfies $S_2$).

$$
\begin{array}{ll}
(2) & M_{good} \vdash \{\ \texttt{A}_{trns},\ \texttt{a} : \texttt{Account} \wedge \langle\!\langle \texttt{a.key} \rangle\!\rangle \wedge \langle\!\langle \texttt{a.key} \rangle\!\rangle \not\leftrightarrow \texttt{Ids}_{trns}\ \} \\
& \qquad\qquad \texttt{Account} :: \texttt{transfer.body} \\
& \qquad \{\langle\!\langle \texttt{a.key} \rangle\!\rangle \wedge \langle\!\langle \texttt{a.key} \rangle\!\rangle -\!\!\triangledown \texttt{res}\} \ \ || \ \ \{\langle\!\langle \texttt{a.key} \rangle\!\rangle\}
\end{array}
$$

Proof Outline

To prove (2), we will need to prove that

$$
\begin{array}{ll}
(21?) & M_{good} \vdash \{\ \texttt{A}_{trns},\ \texttt{a} : \texttt{Account} \wedge \langle\!\langle \texttt{a.key} \rangle\!\rangle \wedge \langle\!\langle \texttt{a.key} \rangle\!\rangle \not\leftrightarrow \texttt{Ids}_{trns}\ \}
\end{array}
$$

```
if (this.key==key') then
 this.blnce:=this.blnce-amt
 dest.blnce:=dest.blnce+amt
else
 res:=0
res:=0
```

$$\{\langle\!\langle \texttt{a.key} \rangle\!\rangle \wedge \langle\!\langle \texttt{a.key} \rangle\!\rangle -\!\!\triangledown \texttt{res}\} \ \ || \ \ \{\langle\!\langle \texttt{a.key} \rangle\!\rangle\}$$

Using the underlying Hoare logic we can prove that no account's `key` gets modified, namely

$$(22) \quad M_{good} \vdash_{ul} \{\ \texttt{A}_{trns},\ \texttt{a} : \texttt{Account} \wedge \langle\!\langle \texttt{a.key} \rangle\!\rangle$$

```
if (this.key==key') then
 this.blnce:=this.blnce-amt
 dest.blnce:=dest.blnce+amt
else
 res:=0
res:=0
```

$$\{\langle\!\langle \texttt{a.key} \rangle\!\rangle\}$$

Using (22) and [Prot-1], we obtain

$$
\begin{array}{ll}
(23) & M_{good} \vdash \{\, \mathtt{A}_{trns},\ \mathtt{a}:\mathtt{Account} \wedge z = \mathtt{a.key} \} \\
& \quad \mathtt{if\ (this.key==key')\ then} \\
& \quad\ \ \mathtt{this.blnce:=this.blnce-amt} \\
& \quad\ \ \mathtt{dest.blnce:=dest.blnce+amt} \\
& \quad \mathtt{else} \\
& \quad\ \ \mathtt{res:=0} \\
& \quad \mathtt{res:=0} \\
& \{ z = \mathtt{a.key} \}
\end{array}
$$

Using (23) and [Embed-UL], we obtain

$$
\begin{array}{ll}
(24) & M_{good} \vdash \{\, \mathtt{A}_{trns},\ \mathtt{a}:\mathtt{Account} \wedge z = \mathtt{a.key} \} \\
& \quad \mathtt{if\ (this.key==key')\ then} \\
& \quad\ \ \mathtt{this.blnce:=this.blnce-amt} \\
& \quad\ \ \mathtt{dest.blnce:=dest.blnce+amt} \\
& \quad \mathtt{else} \\
& \quad\ \ \mathtt{res:=0} \\
& \quad \mathtt{res:=0} \\
& \{ z = \mathtt{a.key} \} \ \ || \ \ \{ z = \mathtt{a.key} \}
\end{array}
$$

[Prot_Int] and the fact that $z$ is an `int` gives us that $\langle\!\langle \mathtt{a.key} \rangle\!\rangle \!\!-\!\!\!\nabla \mathtt{res}$. Using [Types], and [Prot_Int] and [Consequ] on (24) we obtain (21?).

∎

We want to prove that this public method satisfies the specification $S_{2,strong}$, namely

**Lemma H.6** (`set` satisfies $S_2$).

$$
\begin{array}{ll}
(6) & M_{fine} \vdash \{\, \mathtt{A}_{set} \ \wedge\ \langle\!\langle \mathtt{a.key} \rangle\!\rangle \ \wedge\ \langle\!\langle \mathtt{a.key} \rangle\!\rangle \not\leftrightarrow \mathtt{Ids}_{set} \,\} \\
& \quad \mathtt{if\ (this.key==key')\ then} \\
& \quad\ \ \mathtt{this.key:=key''} \\
& \quad \mathtt{else} \\
& \quad\ \ \mathtt{res:=0} \\
& \quad \mathtt{res:=0} \\
& \{ \langle\!\langle \mathtt{a.key} \rangle\!\rangle \ \wedge\ \langle\!\langle \mathtt{a.key} \rangle\!\rangle \!-\!\!\!\nabla \mathtt{res} \} \ \ || \ \ \{ \langle\!\langle \mathtt{a.key} \rangle\!\rangle \}
\end{array}
$$

Proof Outline We will be using the shorthand $\mathtt{A}_2 \triangleq \mathtt{a}:\mathtt{Account},\ \mathtt{A}_{set}$.

To prove (6), we will use the SEQUENCE rule, and we want to prove

$$(61?) \quad M_{fine} \vdash \{\, \mathtt{A}_2 \wedge \langle\mathtt{a.key}\rangle \wedge \langle\mathtt{a.key}\rangle \not\leftrightarrow \mathtt{Ids}_{set} \,\} \\ \mathtt{if\ (this.key{=}{=}key')\ then} \\ \mathtt{this.key{:=}key''} \\ \mathtt{else} \\ \mathtt{res{:=}0} \\ \{\, \mathtt{A}_2 \wedge \langle\mathtt{a.key}\rangle \,\} \ \| \ \{\langle\mathtt{a.key}\rangle\}$$

and that

$$(62?) \quad M_{fine} \vdash \{\, \mathtt{A}_2 \wedge \langle\mathtt{a.key}\rangle \,\} \\ \mathtt{res{:=}0} \\ \{\langle\mathtt{a.key}\rangle \wedge \langle\mathtt{a.key}\rangle \not\nabla \mathtt{res}\} \ \| \ \{\langle\mathtt{a.key}\rangle\}$$

(62?) follows from the types, and PROT-INT$_1$, the fact that `a.key` did not change, and PROT-1.

We now want to prove (61?). For this, will apply the IF-RULE. That is, we need to prove that

$$(63?) \quad M_{fine} \vdash \{\, \mathtt{A}_2 \wedge \langle\mathtt{a.key}\rangle \wedge \langle\mathtt{a.key}\rangle \not\leftrightarrow \mathtt{Ids}_{set} \wedge \mathtt{this.key} = \mathtt{key'} \,\} \\ \mathtt{this.key{:=}key''} \\ \{\langle\mathtt{a.key}\rangle\} \ \| \ \{\langle\mathtt{a.key}\rangle\}$$

and that

$$(64?) \quad M_{fine} \vdash \{\, \mathtt{A}_2 \wedge \langle\mathtt{a.key}\rangle \wedge \langle\mathtt{a.key}\rangle \not\leftrightarrow \mathtt{Ids}_{set} \wedge \mathtt{this.key} \neq \mathtt{key'} \,\} \\ \mathtt{res{:=}0} \\ \{\langle\mathtt{a.key}\rangle\,\} \ \| \ \{\langle\mathtt{a.key}\rangle\}$$

(64?) follows easily from the fact that `a.key` did not change, and PROT-1.

We look at the proof of (63?). We will apply the CASES rule, and distinguish on whether `a.key=this.key`. That is, we want to prove that

$$(65?) \quad M_{fine} \vdash \{\, \mathtt{A}_2 \wedge \langle\mathtt{a.key}\rangle \wedge \langle\mathtt{a.key}\rangle \not\leftrightarrow \mathtt{Ids}_{set} \wedge \mathtt{this.key} = \mathtt{key'} \wedge \mathtt{this.key} = \mathtt{a.key}\} \\ \mathtt{this.key{:=}key''} \\ \{\langle\mathtt{a.key}\rangle\,\} \ \| \ \{\langle\mathtt{a.key}\rangle\}$$

and that

$$(66?) \quad M_{fine} \vdash \{\, \mathtt{A}_2 \wedge \langle\mathtt{a.key}\rangle \wedge \langle\mathtt{a.key}\rangle \not\leftrightarrow \mathtt{Ids}_{set} \wedge \mathtt{this.key} = \mathtt{key'} \wedge \mathtt{this.key} \neq \mathtt{a.key'} \,\} \\ \mathtt{this.key{:=}key''} \\ \{\langle\mathtt{a.key}\rangle\} \ \| \ \{\langle\mathtt{a.key}\rangle\}$$

We can prove (65?) through application of ABSURD, PROTNEQ, and CONSEQU, as follows

$$\text{(61c)}\quad M_{fine} \vdash \{\, false \,\} \quad \texttt{this.key:=key''} \quad \{\langle\texttt{a.key}\rangle\,\} \;\;||\;\; \{\langle\texttt{a.key}\rangle\}$$

By ProtNeq, we have $M_{fine} \vdash \langle\texttt{a.key}\rangle\not\leftrightarrow \texttt{key'} \longrightarrow \texttt{a.key} \neq \texttt{key'}$, and therefore obtain

$$\text{(61d)}\quad M_{fine} \vdash \ldots \wedge \langle\texttt{a.key}\rangle\not\leftrightarrow \texttt{Ids}_{set} \wedge \texttt{this.key} = \texttt{a.key} \wedge \texttt{this.key} = \texttt{key'} \longrightarrow false$$

We apply Consequ on (61c) and (61d) and obtain (61aa?).

We can prove (66?) by proving that $\texttt{this.key}\neq\texttt{a.key}$ implies that $\texttt{this} \neq \texttt{a}$ (by the underlying Hoare logic), which again implies that the assignment `this.key := ...` leaves the value of `a.key` unmodified. We apply Prot-1, and are done.

□

### H.5 Showing that $M_{bad}$ does not satisfy $S_2$ nor $S_3$

*H.5.1 $M_{bad}$ does not satisfy $S_2$.* $M_{bad}$ does not satisfy $S_2$. We can argue this semantically (as in §H.5.2), and also in terms of the proof system (as in H.5.3).

*H.5.2 $M_{bad} \nvDash S_2$.* The reason is that $M_{bad}$ exports the public method `set`, which updates the key without any checks. So, it could start in a state where the key of the account was protected, and then update it to something not protected.

In more detail: Take any state $\sigma$, where $M_{bad}, \sigma \models a_0 : \texttt{Account}, k_0 : \texttt{Key} \wedge \langle a_0.\texttt{key}\rangle$. Assume also that $M_{bad}, \sigma \models \texttt{extl}$. Finally, assume that the continuation in $\sigma$ consists of $a_0.\texttt{set}(k_0)$. Then we obtain that $M_{bad}, \sigma \rightsquigarrow^* \sigma'$, where $\sigma' = \sigma[a_0.\texttt{key} \mapsto k_0]$. We also have that $M_{bad}, \sigma' \models \texttt{extl}$, and because $k_0$ is a local variable, we also have that $M_{bad}, \sigma' \nvDash \langle k_0\rangle$. Moreover, $M_{bad}, \sigma' \models a_0.\texttt{krey} = k_0$. Therefore, $M_{bad}, \sigma' \nvDash \langle a_0.\texttt{key}\rangle$.

*H.5.3 $M_{bad} \nvdash S_2$.* In order to prove that $M_{bad} \vdash S_2$, we would have needed to prove, among other things, that `set` satisfied $S_2$, which means proving that

$$\text{(ERR\_1?)}\quad M_{bad} \vdash \{\, \texttt{this}:\texttt{Account}, \texttt{k'}:\texttt{Key}, a:\texttt{Account} \wedge \langle a.\texttt{key}\rangle \wedge \langle\texttt{a.key}\rangle\not\leftrightarrow \{\texttt{this},\texttt{k'}\}\,\} \quad \texttt{this.key:=k';} \quad \texttt{res} := 0 \quad \{\, \langle a.\texttt{key}\rangle \wedge \langle a.\texttt{key}\rangle\not\leftrightarrow \texttt{res}\,\} \;\;||\;\; \{\ldots\}$$

However, we cannot establish (ERR_1?). Namely, when we take the case where $\texttt{this} = a$, we would need to establish, that

$$\text{(ERR\_2?)}\quad M_{bad} \vdash \{\, \texttt{this}:\texttt{Account}, \texttt{k'}:\texttt{Key} \wedge \langle\texttt{this.key}\rangle \wedge \langle\texttt{this.key}\rangle\not\leftrightarrow \{\texttt{this},\texttt{k'}\}\,\} \quad \texttt{this.key:=k'} \quad \{\, \langle\texttt{this.key}\rangle\,\} \;\;||\;\; \{\ldots\}$$

And there is no way to prove (ERR_2?). In fact, (ERR_2?) is not sound, for the reasons outlined in §H.5.2.

*H.5.4 $M_{bad}$ does not satisfy $S_3$.* We have already argued in §?? that $M_{bad}$ does not satisfy $S_3$, by giving a semantic argument – ie we are in state where $\langle a_0.\texttt{key}\rangle$, and execute $\texttt{a}_0\texttt{.set(k1);a}_0\texttt{.transfer(...k1)}$.

Moroiever, if we attempted to prove that `set` satisfies $S_3$, we would have to show that

$$
\begin{array}{lll}
\text{(ERR\_3?)} & M_{bad} \vdash \{\, \texttt{this}:\texttt{Account},\ \texttt{k}':\texttt{Key}, a:\texttt{Account},\ b:\texttt{int} \wedge \\
& \quad \langle a.\texttt{key}\rangle \wedge \langle \texttt{a.key}\rangle \not\leftrightarrow \{\texttt{this},\texttt{k}'\} \wedge a.\texttt{blnce} \geq b \,\} \\
& \qquad \texttt{this.key:=k}'; \\
& \qquad \texttt{res} := 0 \\
& \{\, \langle a.\texttt{key}\rangle \wedge \langle a.\texttt{key}\rangle \not\leftrightarrow \texttt{res} \wedge a.\texttt{blnce} \geq b \,\} \ \ || \ \ \{...\}
\end{array}
$$

which, in the case of $a = \texttt{this}$ would imply that

$$
\begin{array}{lll}
\text{(ERR\_4?)} & M_{bad} \vdash \{\, \texttt{this}:\texttt{Account},\ \texttt{k}':\texttt{Key},\ b:\texttt{int} \wedge \\
& \quad \langle \texttt{this.key}\rangle \wedge \langle \texttt{this.key}\rangle \not\leftrightarrow \{\texttt{this},\texttt{k}'\} \wedge \texttt{this.blnce} \geq b \,\} \\
& \qquad \texttt{this.key:=k}' \\
& \{\, \langle \texttt{this.key}\rangle \,\} \ \ || \ \ \{...\}
\end{array}
$$

And (ERR_4?) cannot be proven and does not hold.

## H.6 Demonstrating that $M_{good} \vdash S_3$, and that $M_{fine} \vdash S_3$

## H.7 Extending the specification $S_3$

As in §H.4, we redefine $S_3$ so that it also describes the behaviour of method `send`. and have:

$$S_{3,strong} \ \triangleq\ S_3 \ \wedge\ S_{2a} \ \wedge\ S_{2b}$$

**Lemma H.7** (module $M_{good}$ satisfies $S_{3,strong}$)**.** $M_{good} \vdash S_{3,strong}$

Proof Outline In order to prove that

$$M_{good} \vdash \forall \texttt{a}:\texttt{Account}, b:\texttt{int}.\{\, \langle \texttt{a.key}\rangle \wedge \texttt{a.blnce} \geq b \,\}$$

we have to apply Invariant from Fig. 8. That is, for each class $C$ defined in $M_{good}$, and for each public method $m$ in $C$, with parameters $\overline{y:D}$, we have to prove that they satisfy the corresponding quadruples.

Thus, we need to prove three Hoare quadruples: one for `Shop::buy`, one for `Account::transfer`, and one for `Account::set`. That is, we have to prove that

$$
\begin{array}{lll}
(31?) & M_{good} \vdash \{\, \mathtt{A}_{buy},\ \texttt{a}:\texttt{Account}, b:\texttt{int} \wedge \langle \texttt{a.key}\rangle \wedge \langle \texttt{a.key}\rangle \not\leftrightarrow \texttt{Ids}_{buy} \wedge \texttt{a.blnce} \geq b \,\} \\
& \qquad \texttt{Shop :: buy.body} \\
& \{\langle \texttt{a.key}\rangle \wedge \langle \texttt{a.key}\rangle \nrightarrow \texttt{res} \wedge \texttt{a.blnce} \geq b\} \ \ || \ \ \{\langle \texttt{a.key}\rangle \wedge \texttt{a.blnce} \geq b\} \\
(32?) & M_{good} \vdash \{\, \mathtt{A}_{trns},\ \texttt{a}:\texttt{Account}, b:\texttt{int} \wedge \langle \texttt{a.key}\rangle \wedge \langle \texttt{a.key}\rangle \not\leftrightarrow \texttt{Ids}_{trns} \wedge \texttt{a.blnce} \geq b \,\} \\
& \qquad \texttt{Account :: transfer.body} \\
& \{\langle \texttt{a.key}\rangle \wedge \langle \texttt{a.key}\rangle \nrightarrow \texttt{res} \wedge \texttt{a.blnce} \geq b \,\} \ \ || \ \ \{\langle \texttt{a.key}\rangle \wedge \texttt{a.blnce} \geq b \,\} \\
(33?) & M_{good} \vdash \{\, \mathtt{A}_{set},\ \texttt{a}:\texttt{Account}, b:\texttt{int} \wedge \langle \texttt{a.key}\rangle \wedge \langle \texttt{a.key}\rangle \not\leftrightarrow \texttt{Ids}_{set} \wedge \texttt{a.blnce} \geq b \,\} \\
& \qquad \texttt{Account :: set.body} \\
& \{\langle \texttt{a.key}\rangle \wedge \langle \texttt{a.key}\rangle \nrightarrow \texttt{res} \wedge \texttt{a.blnce} \geq b \,\} \ \ || \ \ \{\langle \texttt{a.key}\rangle \wedge \texttt{a.blnce} \geq b \,\}
\end{array}
$$

where we are using ? to indicate that this needs to be proven, and where we are using the shorthands $\mathtt{A}_{buy}$, $\texttt{Ids}_{buy}$, $\mathtt{A}_{trns}$, $\texttt{Ids}_{trns}$, $\mathtt{A}_{set}$ as defined earlier.

□

The proofs for $M_{fine}$ are similar.

We outline the proof of (31?) in Lemma H.8. We outline the proof of (32?) in Lemma **??**.

### H.7.1 *Proving that* `Shop::buy` *from* $M_{good}$ *satisfies* $S_{3,strong}$ *and also* $S_4$.

**Lemma H.8** (function $M_{good}$ :: `Shop` :: `buy` satisfies $S_{3,strong}$ and also $S_4$).

$$(31)\quad M_{good} \vdash \{\, \texttt{A}_{buy},\ \texttt{a}:\texttt{Account}, b:\texttt{int}, \wedge \langle\texttt{a.key}\rangle \wedge \langle\texttt{a.key}\rangle \not\leftrightarrow \texttt{Ids}_{buy} \wedge \texttt{a.blnce} \geq b \,\}$$
$$\texttt{Shop} :: \texttt{buy.body}$$
$$\{\langle\texttt{a.key}\rangle \wedge \langle\texttt{a.key}\rangle \not\nabla \texttt{res} \wedge \texttt{a.blnce} \geq b\} \ \ || \ \ \{\langle\texttt{a.key}\rangle \wedge \texttt{a.blnce} \geq b\}$$

Proof Outline Note that (31) is a proof that $M_{good}$ :: `Shop` :: `buy` satisfies $S_{3,strong}$ and also hat $M_{good}$ :: `Shop` :: `buy` satisfies $S_4$. This is so, because application of [Method] on $S_4$ gives us exactly the proof obligation from (31).

This proof is similar to the proof of lemma H.4, with the extra requirement here that we need to argue about balances not decreasing. To do this, we will leverage the assertion about balances given in $S_3$.

We will use the shorthand $\texttt{A}_1 \triangleq \texttt{A}_{buy},\ \texttt{a}:\texttt{Account}, b:\texttt{int}$. We will split the proof into 1) proving that statements 10, 11, 12 preserve the protection of `a.key` from the `buyer`, 2) proving that the external call

**1st Step: proving statements 10, 11, 12**

We apply the underlying Hoare logic and prove that the statements on lines 10, 11, 12 do not affect the value of `a.key` nor that of `a.blnce`. Therefore, for a $z, z' \notin \{\texttt{price}, \texttt{myAccnt}, \texttt{oldBalance}\}$, we have

$$(40)\quad M_{good} \vdash_{ul} \{\, \texttt{A}_1 \ \wedge\ z = \texttt{a.key} \wedge\ z' = \texttt{a.blnce}\}$$
$$\texttt{price:=anItem.price;}$$
$$\texttt{myAccnt:=this.accnt;}$$
$$\texttt{oldBalance := myAccnt.blnce;}$$
$$\{z = \texttt{a.key} \wedge\ z' = \texttt{a.blnce}\}$$

We then apply Embed_UL, Prot-1 and Prot-2 and Combine and and Types-2 on (10) and use the shorthand $\texttt{stmts}_{10,11,12}$ for the statements on lines 10, 11 and 12, and obtain:

$$(41)\quad M_{good} \vdash \{\, \texttt{A}_1 \ \wedge\ \langle\texttt{a.key}\rangle \ \wedge\ \langle\texttt{buyer}\rangle \not\leftrightarrow \texttt{a.key} \wedge\ z' = \texttt{a.blnce}\}$$
$$\texttt{stmts}_{10,11,12}$$
$$\{\, \langle\texttt{a.key}\rangle \ \wedge\ \langle\texttt{buyer}\rangle \not\leftrightarrow \texttt{a.key} \wedge\ z' = \texttt{a.blnce}\}$$

We apply Mid on (11) and obtain

$$(42)\quad M_{good} \vdash \{\, \texttt{A}_1 \ \wedge\ \langle\texttt{a.key}\rangle \not\leftrightarrow \texttt{buyer} \ \wedge\ z' = \texttt{a.blnce}\}$$
$$\texttt{stmts}_{10,11,12}$$
$$\{\, \texttt{A}_1 \ \wedge\ \langle\texttt{a.key}\rangle \ \wedge\ \langle\texttt{buyer}\rangle \not\leftrightarrow \texttt{a.key} \ \wedge\ z' = \texttt{a.blnce}\} \ ||$$
$$\{\, \langle\texttt{a.key}\rangle \ \wedge\ z' = \texttt{a.blnce}\}$$

**2nd Step: Proving the External Call**

We now need to prove that the external method call `buyer.pay(this.accnt, price)` protects the `key`, and does nit decrease the balance, i.e.

$$(43?)\quad M_{good} \vdash \{\, \texttt{A}_1 \ \wedge\ \langle\texttt{a.key}\rangle \ \wedge\ \langle\texttt{a.key}\rangle \not\leftrightarrow \texttt{buyer} \wedge\ z' = \texttt{a.blnce}\}$$
$$\texttt{tmp := buyer.pay(myAccnt, price)}$$
$$\{\ \ \texttt{A}_1 \ \wedge\ \langle\texttt{a.key}\rangle \ \wedge\ \langle\texttt{buyer}\rangle \not\leftrightarrow \texttt{a.key} \wedge\ \texttt{a.blnce} \geq z' \ \} \ \ ||$$
$$\{\, \langle\texttt{a.key}\rangle \ \wedge\ \texttt{a.blnce} \geq z'\}$$

We use that $M \vdash \forall$ `a : Account, b : int,`.$\{\langle$`a.key`$\rangle \wedge$ `a.blnce` $\geq z'\}$ and obtain

(44) $M_{good} \vdash \{$ `buyer : external,` $\langle$`a.key`$\rangle \wedge \langle$`a.key`$\rangle \not\leftrightarrow$ `(buyer, myAccnt, price)` $\wedge\ z' \geq$ `a.blnce`$\}$

```
tmp := buyer.pay(myAccnt, price)
```

$\{\langle$`a.key`$\rangle \wedge \langle$`a.key`$\rangle \not\leftrightarrow$ `(buyer, myAccnt, price)` $\wedge\ z' \geq$ `a.blnce`$\}$ $||$

$\{\langle$`a.key`$\rangle \wedge\ z' \geq$ `a.blnce`$\}$

In order to obtain (43?) out of (44), we apply Prot-Intl and Prot-Int$_1$, which gives us

(45) $M_{good} \vdash$ $A_1 \wedge \langle$`a.key`$\rangle \longrightarrow \langle$`a.key`$\rangle \not\leftrightarrow$ `myAccnt`

(46) $M_{good} \vdash$ $A_1 \wedge \langle$`a.key`$\rangle \longrightarrow \langle$`a.key`$\rangle \not\leftrightarrow$ `price`

(47) $M_{good} \vdash$ $A_1 \wedge z' =$ `a.blnce` $\longrightarrow z' \geq$ `a.blnce`

We apply Consequ on (44), (45), (46) and (47) and obtain (43)!

**3nd Step: Proving the Remainder of the Body**

We now need to prove that lines 15-19 of the method preserve the protection of `a.key`, and do not decrease `a.balance`. We outline the remaining proof in less detail.

We prove the internal call on line 16, using the method specification for `send`, using $S_{2a}$ and $S_{2b}$, and applying rule [Call_Int], and obtain

(48) $M_{good} \vdash \{$ `buyer : external, item : Intem` $\wedge \langle$`a.key`$\rangle \wedge \langle$`a.key`$\rangle \not\leftrightarrow$ `(buyer` $\wedge\ z' =$ `a.blnce`$\}$

```
tmp := this.send(buyer, Item)
```

$\{\langle$`a.key`$\rangle \wedge \langle$`a.key`$\rangle \not\leftrightarrow$ `buyer` $\wedge\ z' =$ `a.blnce`$\}$ $||$

$\{\langle$`a.key`$\rangle \wedge\ z' =$ `a.blnce`$\}$

We now need to prove that the external method call `buyer.tell("You have not paid me")` also protects the `key`, and does nit decrease the balance. We can do this by applying the rule about protection from strings [Pror_Str], the fact that $M_{good} \vdash S_3$, and rule [Call_Extl_Adapt] and obtain:

(49) $M_{good} \vdash \{$ `buyer : external, item : Intem` $\wedge \langle$`a.key`$\rangle \wedge \langle$`a.key`$\rangle \not\leftrightarrow$ `(buyer` $\wedge\ z'/geq$`a.blnce`$\}$

```
tmp:=buyer.tell("You have not paid me")
```

$\{\langle$`a.key`$\rangle \wedge \langle$`a.key`$\rangle \not\leftrightarrow$ `buyer` $\wedge\ z' \geq$ `a.blnce`$\}$ $||$

$\{\langle$`a.key`$\rangle \wedge\ z' \geq$ `a.blnce`$\}$

We can now apply [If_Rule, and [Conseq on (49) and (50), and obtain

(50) $M_{good} \vdash \{$ `buyer : external, item : Intem` $\wedge \langle$`a.key`$\rangle \wedge \langle$`a.key`$\rangle \not\leftrightarrow$ `(buyer` $\wedge\ z' \geq$ `a.blnce`$\}$

```
if...then
 tmp:=this.send(buyer,anItem)
else
 tmp:=buyer.tell("You have not paid me")
```

$\{\langle$`a.key`$\rangle \wedge \langle$`a.key`$\rangle \not\leftrightarrow$ `buyer` $\wedge\ z' \geq$ `a.blnce`$\}$ $||$

$\{\langle$`a.key`$\rangle \wedge\ z' \geq$ `a.blnce`$\}$

The rest follows through application of [Prot_Int, and [Seq]. $\square$

**Lemma H.9** (function $M_{good}$ :: $\texttt{Account}$ :: $\texttt{transfer}$ satisfies $S_3$).

$$(32)\quad M_{good} \vdash \{\, \mathbb{A}_{trns},\ \texttt{a}:\texttt{Account}\,,b:\texttt{int}\ \wedge\ \langle\!\langle\texttt{a.key}\rangle\!\rangle\ \wedge\ \langle\!\langle\texttt{a.key}\rangle\!\rangle\not\leftrightarrow \texttt{Ids}_{trns}\ \wedge \texttt{a.blnce} \geq b \,\}$$
$$\texttt{Account} :: \texttt{transfer.body}$$
$$\{\langle\!\langle\texttt{a.key}\rangle\!\rangle\ \wedge\ \langle\!\langle\texttt{a.key}\rangle\!\rangle\text{-}\!\!\nabla\texttt{res} \wedge \texttt{a.blnce} \geq b\,\}\ \ ||\ \ \{\langle\!\langle\texttt{a.key}\rangle\!\rangle \wedge \texttt{a.blnce} \geq b\,\}$$

Proof Outline We will use the shorthand $stmts_{28-33}$ for the statements in the body of $\texttt{transfer}$. We will prove the preservation of protection, separately from the balance not decreasing when the key is protcted.

For the former, applying the steps in the proof of Lemma H.5, we obtain

$$(21)\quad M_{good} \vdash \{\, \mathbb{A}_{trns},\ \texttt{a}:\texttt{Account}\ \wedge\ \langle\!\langle\texttt{a.key}\rangle\!\rangle\ \wedge\ \langle\!\langle\texttt{a.key}\rangle\!\rangle\not\leftrightarrow \texttt{Ids}_{trns}\,\}$$
$$stmts_{28-33}$$
$$\{\langle\!\langle\texttt{a.key}\rangle\!\rangle\ \wedge\ \langle\!\langle\texttt{a.key}\rangle\!\rangle\text{-}\!\!\nabla\texttt{res}\}\ \ ||\ \ \{\langle\!\langle\texttt{a.key}\rangle\!\rangle\}$$

For the latter, we rely on the underlying Hoare logic to ensure that no balance decreases, except perhaps that of the receiver, in which case its key was not protected. Namely, we have that

$$(71)\quad M_{good} \vdash_{u} l\ \{\, \mathbb{A}_{trns},\ \texttt{a}:\texttt{Account}\ \wedge\ \texttt{a.blnce} = b\ \wedge\ (\texttt{this} \neq \texttt{a} \vee prgthis.\texttt{key} \neq \texttt{key}')\,\}$$
$$stmts_{28-33}$$
$$\{\texttt{a.blnce} \geq b\}$$

We apply rules Embed_UL and Mid on (71), and obtain

$$(72)\quad M_{good} \vdash \{\, \mathbb{A}_{trns},\ \texttt{a}:\texttt{Account}\ \wedge\ \texttt{a.blnce} = b\ \wedge\ (\texttt{this} \neq \texttt{a} \vee prgthis.\texttt{key} \neq \texttt{key}')\,\}$$
$$stmts_{28-33}$$
$$\{\texttt{a.blnce} \geq b\}\ \ ||\ \ \{\texttt{a.blnce} \geq b\}$$

Moreover, we have

$$(73)\quad M_{good} \quad \vdash \quad \langle\!\langle\texttt{a.key}\rangle\!\rangle\not\leftrightarrow \texttt{Ids}_{trns} \ \longrightarrow\ \langle\!\langle\texttt{a.key}\rangle\!\rangle\not\leftrightarrow \texttt{key}'$$
$$(74)\quad M_{good} \quad \vdash \quad \langle\!\langle\texttt{a.key}\rangle\!\rangle\not\leftrightarrow \texttt{key}' \ \longrightarrow\ \texttt{a.key} \neq \texttt{key}'$$
$$(75)\quad M_{good} \quad \vdash \quad \texttt{a.key} \neq \texttt{key}' \ \longrightarrow\ \texttt{a} \neq \texttt{this} \vee \texttt{this.key} \neq \texttt{key}'$$

normalsize

Applying (73), (74), (75) and Conseq on (72) we obtain:

$$(76)\quad M_{good} \vdash \{\, \mathbb{A}_{trns},\ \texttt{a}:\texttt{Account}\ \wedge\ \texttt{a.blnce} = b\ \wedge\ \langle\!\langle\texttt{a.key}\rangle\!\rangle\not\leftrightarrow \texttt{Ids}_{trns}\ \}$$
$$stmts_{28-33}$$
$$\{\texttt{a.blnce} \geq b\}\ \ ||\ \ \{\texttt{a.blnce} \geq b\}$$

We combine (72) and (76) through Combine and obtain (32).

□

## H.8 Dealing with polymorphic function calls

The case split rules together with the rule of consequence allow our Hoare logic to formally reason about polymorphic calls, where the receiver may be internal or external.

We demonstrate this through an example where we may have an external receiver, or a receiver from a class $C$. Assume we had a module $M$ with a scoped invariant (as in A), and an internal method specification as in (B).

$$(A)\quad M \quad \vdash \quad \forall y_1 : D.\{A\}$$
$$(B)\quad M \quad \vdash \quad \{\,A_1\,\}\,\texttt{private}\ C{::}m(y_1 : D)\,\{\,A_2\,\}||\,\{A_3\}$$

Assume also implications as in (C)-(H)

$$
\begin{array}{llll}
(C) & M & \vdash & A_0 \;\rightarrow\; A\text{-}\!\nabla(y_0, y_1) \\
(D) & M & \vdash & A\text{-}\!\nabla(y_0, y_1) \rightarrow A_4 \\
(E) & M & \vdash & A \rightarrow A_5 \\
(F) & M & \vdash & A_0 \rightarrow A_1[y_0/\texttt{this}] \\
(G) & M & \vdash & A_2[y_0, u/\texttt{this}, res] \rightarrow A_4 \\
(H) & M & \vdash & A_3 \rightarrow A_5
\end{array}
$$

Then, by application of CALL_EXT_ADAPT on (A) we obtain (I)

$$(I)\;\; M \vdash \{\; y_0 : external, y_1 : D \wedge A\text{-}\!\nabla(y_0, y_1) \;\}\; u := y_0.m(y_1) \;\{\; A\text{-}\!\nabla(y_0, y_1) \;\} \;\;\|\;\; \{\; A \;\}$$

By application of the rule of consequence on (I) and (C), (D), and (E), we obtain

$$(J)\;\; M \vdash \{\; y_0 : external, y_1 : D \wedge A_0 \;\}\; u := y_0.m(y_1) \;\{\; A_4 \;\} \;\;\|\;\; \{\; A_5 \;\}$$

Then, by application of [CALL_INTL] on (B) we obtain (K)

$$(K)\;\; M \vdash \{\; y_0 : C, y_1 : D \wedge A_1[y_0/\texttt{this}] \;\}\; u := y_0.m(y_1) \;\{\; A_2[y_0, u/\texttt{this}, res] \;\} \;\;\|\;\; \{\; A_3 \;\}$$

By application of the rule of consequence on (K) and (F), (G), and (H), we obtain

$$(L)\;\; M \vdash \{\; y_0 : C, y_1 : D \wedge A_0 \;\}\; u := y_0.m(y_1) \;\{\; A_4 \;\} \;\;\|\;\; \{\; A_5 \;\}$$

By case split, [CASES], on (J) and (L), we obtain

$$(polymoprhic)\;\; M \vdash \{\; (y_0 : external \vee y_0 : C), y_1 : D \wedge A_0 \;\}\; u := y_0.m(y_1) \;\{\; A_4 \;\} \;\;\|\;\; \{\; A_5 \;\}$$