1 Objectives of this Lecture

Please note: This is an article version of the ATM lecture. At the moment there is a problem producing slides from the \LaTeX{} version of the machines produced by the Rodin \LaTeX{} generator.

- to demonstrate that nondeterminism can be closer to reality than determinism.
- to illustrate the above using a simple ATM example.
2 ATM0: A Simplistic Model of an ATM

We want to produce a model of an ATM. The model will be kept reasonably simple, but also reasonably realistic.

Required ATM operations:

- an operation to insert the card and provide a password;
- an operation to withdraw money;

The initial attempt might be as shown in the ATM0 machine. This is likely to be the type of specification produced by someone familiar only with machine level development.

2.1 ATM0

ATM Context

CONTEXT ATM_ctx

SETS ACCOUNT The set of account IDs RESPONSES Set of responses

CONSTANTS OK REFUSED RESPONSE Possible responses

AXIOMS

axm1: finite(ACCOUNT)
axm4: RESPONSES = {OK, REFUSED}
axm5: OK ≠ REFUSED
axm6: RESPONSE = {{OK}, {REFUSED}, ∅}

END

Password context

CONTEXT Password

SETS PASSWORD

END

MACHINE ATM0

SEES ATM_ctx, Password

VARIABLES accounts password balance customer response

INVARIANTS
inv1: accounts ⊆ ACCOUNT
inv2: finite(accounts)
inv3: password ∈ accounts → PASSWORD
inv4: balance ∈ accounts → \mathbb{Z}
inv5: customer ⊆ accounts
inv6: card(customer) ≤ 1
inv7: response ∈ RESPONSE

EVENTS

Initialisation

\textbf{begin}

\textbf{act}1: accounts := ∅

\textbf{act}2: password := ∅

\textbf{act}3: balance := ∅

\textbf{act}4: customer := ∅

\textbf{act}5: response := ∅

\textbf{end}

\textbf{Event} \textit{InsertCard} ≜

\textbf{any} account pass

\textbf{when}

\textbf{grd}1: account ∈ ACCOUNT

\textbf{grd}2: pass ∈ PASSWORD

\textbf{grd}3: customer = ∅

\textbf{grd}4: response = ∅

\textbf{then}

\textbf{act}1: response, customer :

\textbf{|}

(account ∈ accounts ∧ pass = password(account) \\
⇒ response' = \{OK\} ∧ customer' = \{account\})

∧ ((account \notin accounts ∨ pass \neq password(account)) \\
⇒ response' = \{REFUSED\} ∧ customer' = ∅)

\textbf{end}

\textbf{Event} \textit{Withdraw} ≜

\textbf{any} amount account
when

grd1: \( \text{response} = \emptyset \)

grd2: \( \text{customer} \neq \emptyset \)

grd3: \( \text{amount} \in \mathbb{N} \)

grd4: \( \{\text{account}\} = \text{customer} \)

then

act1: \( \text{response} : | \)
\( (\text{balance}(\text{account}) \geq \text{amount} \)
\( \Rightarrow \text{response}' = \{\text{OK}\}) \)
\( \land (\text{balance}(\text{account}) < \text{amount} \)
\( \Rightarrow \text{response}' = \{\text{REFUSED}\}) \)

act2: \( \text{balance} : | \)
\( (\text{balance}(\text{account}) \geq \text{amount} \)
\( \Rightarrow \text{balance}' = \text{balance} \)
\( \Leftarrow \{\text{account} \mapsto \text{balance}(\text{account}) - \text{amount}\} \)
\( \land (\text{balance}(\text{account}) < \text{amount} \)
\( \Rightarrow \text{balance}' = \text{balance} \)

end

Event \( \text{ResetResponse} \triangleq \) Resets response

when

grd1: \( \text{response} \neq \emptyset \)

then

act1: \( \text{response} := \emptyset \)

end

END

2.2 Improving the Model

This \( \text{ATM0} \) model is seriously ill-conceived. It puts bank-like state inside the ATM. This is obviously wrong: ATMs have no banking knowledge, they are simply boxes in the wall that interact with a card user and communicate with a remote banking system.

We will attempt to build a more realistic model that separates the ATM and the remote banking system. First, we need to specify the context information that is common to both the ATM and the remote banking system. This is shown in \( \text{CardStatus} \) and \( \text{Password} \) contexts. It’s split into two machines because the account, service card and response modelling “belongs” to the banking system, but the modelling of passwords is global.
One way of thinking about how you should model a machine like an ATM is to imagine watching another person using an ATM, from a little distance. Your model should describe what you see, not what you might imagine is going on inside the machine.

ATM0 could never be the result of such an exercise.

We are emphasising what rather than how, perhaps more than in any previous exercise.

**ATM machine**

**MACHINE** ATM

**SEES** ATM_ctx, Password

**VARIABLES** response The variables of this machine model customer what we may think of as a User Interface. balance Each variable is a set that may be either empty money or contain a single value.

**INVARIANTS**

inv1: customer ⊆ ACCOUNT
inv2: card(customer) ≤ 1
inv3: response ∈ RESPONSE
inv4: balance ⊆ ℤ
inv5: finite(balance)
inv6: card(balance) ≤ 1
inv7: money ⊆ ℕ
inv8: finite(money)
inv9: card(money) ≤ 1

**EVENTS**

**Initialisation**

begin
act1: customer := ∅
act2: response := ∅
act3: balance := ∅
act4: money := ∅
end

Event InsertCard ≜ Insert service card into ATM
any scard pass
when

$\text{grd1: customer} = \emptyset$

$\text{grd2: response} = \emptyset$

$\text{grd3: scard} \in \text{SCARD}$

$\text{grd4: pass} \in \text{PASSWORD}$

then

$\text{act1: } \text{response, customer : | }$

$\text{response}' \in \{\{\text{OK}\}, \{\text{REFUSED}\}\} \\
\wedge \text{customer}' \in \mathbb{P}(\text{ACCOUNT}) \wedge (\text{response}' = \text{OK}) \\
\Rightarrow \text{customer}' = \{\text{GENSCARD}^{-1}(\text{scard})\} \\
\wedge (\text{response}' = \{\text{REFUSED}\} \Rightarrow \text{customer}' = \emptyset)$

end

Event $\text{Withdraw} \triangleq \text{Make withdrawal from ATM}$

any $\text{amount}$

when

$\text{grd1: customer} \neq \emptyset$

$\text{grd2: amount} \in \mathbb{N}$

then

$\text{act1: } \text{response, money, balance : | }$

$\text{response}' \in \{\{\text{OK}\}, \{\text{REFUSED}\}\} \\
\wedge \text{balance}' \subseteq \mathbb{Z} \wedge \text{finite(balance')} \wedge (\text{response}' = \text{OK}) \\
\Rightarrow \text{money}' = \{\text{amount}\} \wedge \text{balance}' \in \mathbb{P}(\mathbb{Z}) \\
\wedge \text{card(balance')} \leq 1 \\
\wedge (\text{response}' = \{\text{REFUSED}\} \Rightarrow \text{money}' = \emptyset \wedge \text{balance}' = \emptyset)$

end

Event $\text{RemoveCard} \triangleq \text{Customer terminates session}$

when

$\text{grd1: customer} \neq \emptyset$

then

$\text{act1: } \text{response} := \{\text{OK}\}$

$\text{act2: } \text{customer} := \emptyset$

end

Event $\text{ResetResponse} \triangleq \text{Reset response when no customer using ATM}$
when
grd1: \( \text{customer} = \emptyset \)
grd2: \( \text{response} \neq \emptyset \)
then
act1: \( \text{response} := \emptyset \)
end

Event \( \text{ResetUI} \equiv \) Reset User Interface

when
grd1: \( \text{customer} \neq \emptyset \Rightarrow \text{money} \neq \emptyset \)
grd2: \( \text{customer} \neq \emptyset \Rightarrow \text{balance} \neq \emptyset \)
grd3: \( \text{customer} \neq \emptyset \Rightarrow \text{response} \neq \emptyset \)
then
act1: \( \text{money} := \emptyset \)
act2: \( \text{balance} := \emptyset \)
act3: \( \text{response} := \emptyset \)
end
END

CardStatus context

CONTEXT CardStatus
EXTENDS ServiceCards
SETS CARDSTATUS
CONSTANTS validaccounts currentbalance withdrawlimit password CARDOK CARDNOK
AXIOMS
axm1: \( \text{validaccounts} \subseteq \text{ACCOUNT} \)
axm2: \( \text{currentbalance} \in \text{validaccounts} \rightarrow \mathbb{Z} \)
axm3: \( \text{withdrawlimit} \in \text{validaccounts} \rightarrow \mathbb{N} \)
axm4: \( \text{CARDSTATUS} = \{\text{CARDOK, CARDNOK}\} \)
axm5: \( \text{CARDOK} \neq \text{CARDNOK} \)
axm6: \( \text{password} \in \text{validaccounts} \rightarrow \text{PASSWORD} \)
END
ServicCard context

CONTEXT ServiceCards
EXTENDS ATM_ctx

SETS SCARD The set of service cards

CONSTANTS GENSCARD An injective function that maps service cards to accounts

AXIOMS

axm1: finite(SCARD)
axm2: GENSCARD ∈ ACCOUNT ↠ SCARD

END

We are now modelling a service card, distinct from the account. We assume that the service card can be represented by information that is generated from the account, and that the account can be extracted from the service card.

2.3 ATMR0

We show two stages in refinement of the ATM. The first attempt, ATMR0, is nearly what we are aiming for, but it contains modelling of the login management that is really nothing to do with the pure interface view of an ATM.

MACHINE ATMR0
REFINES ATM
SEES CardStatus

VARIABLES response The variables of this machine model customer what we may think of as a User Interface. balance Each variable is a set that may be either empty money or contain a single value.

INVARINTS

inv1: customer ∈ P(validaccounts)

EVENTS

Initialisation

begin

act1: customer := ∅
act2: response := ∅
act3: balance := ∅
act4: money := ∅
end

Event $\text{InsertCard}_{\text{ok}}$ $\triangleq$ Insert service card into ATM
refines $\text{InsertCard}$
any scard pass account
when
$\text{grd}1$: customer = $\emptyset$
$\text{grd}2$: response = $\emptyset$
$\text{grd}3$: scard $\in$ SCARD
$\text{grd}4$: account = GENSCARD$^{-1}$(scard)
$\text{grd}5$: account $\in$ validaccounts
$\text{grd}6$: pass = password(account)
then
act1: response := \{OK\}
act2: customer := \{account\}
end

Event $\text{InsertCard}_{\text{nok}}$ $\triangleq$
refines $\text{InsertCard}$
any scard pass account
when
$\text{grd}1$: customer = $\emptyset$
$\text{grd}2$: response = $\emptyset$
$\text{grd}3$: scard $\in$ SCARD
$\text{grd}4$: account = GENSCARD$^{-1}$(scard)
$\text{grd}5$: account $\in$ validaccounts $\Rightarrow$ pass $\neq$ password(account)
then
act1: response := \{REFUSED\}
end

Event $\text{Withdraw}_{\text{ok}}$ $\triangleq$ Make withdrawal from ATM
refines $\text{Withdraw}$
any amount account
when

grd1: \( \text{customer} \neq \emptyset \)
grd2: \( \text{amount} \in \mathbb{N} \)
grd3: \( \text{customer} = \{\text{account}\} \)
grd4: \( \text{amount} \leq \text{withdrawlimit}(\text{account}) \)

then

act1: \( \text{response} := \{\text{OK}\} \)
act2: \( \text{balance} \in \{\emptyset\} \cup \{n \cdot n \in \mathbb{Z}|\{n\}\} \)
act3: \( \text{money} := \{\text{amount}\} \)

end

Event \text{Withdraw}\_\text{nok} \equiv

refines \text{Withdraw}

\text{any amount account}

when

grd1: \( \text{customer} \neq \emptyset \)
grd2: \( \text{amount} \in \mathbb{N} \)
grd3: \( \text{customer} = \{\text{account}\} \)
grd4: \( \text{amount} > \text{withdrawlimit}(\text{account}) \)

then

act1: \( \text{response} := \{\text{REFUSED}\} \)
act2: \( \text{balance} := \emptyset \)
act3: \( \text{money} := \emptyset \)

end

Event \text{RemoveCard} \equiv \text{Customer terminates session}

refines \text{RemoveCard}

when

grd1: \( \text{customer} \neq \emptyset \)

then

act1: \( \text{response} := \{\text{OK}\} \)
act2: \( \text{customer} := \emptyset \)
Event $\text{ResetResponse} \triangleq$ Reset response when no customer using ATM
refines $\text{ResetResponse}$
when
\begin{align*}
\text{grd1:} & \quad \text{customer} = \emptyset \\
\text{grd2:} & \quad \text{response} \neq \emptyset
\end{align*}
then
\begin{align*}
\text{act1:} & \quad \text{response} := \emptyset
\end{align*}
end

Event $\text{ResetUI} \triangleq$ Reset User Interface
refines $\text{ResetUI}$
when
\begin{align*}
\text{grd1:} & \quad \text{customer} \neq \emptyset \Rightarrow \text{money} \neq \emptyset \\
\text{grd2:} & \quad \text{customer} \neq \emptyset \Rightarrow \text{balance} \neq \emptyset \\
\text{grd3:} & \quad \text{customer} \neq \emptyset \Rightarrow \text{response} \neq \emptyset
\end{align*}
then
\begin{align*}
\text{act1:} & \quad \text{money} := \emptyset \\
\text{act2:} & \quad \text{balance} := \emptyset \\
\text{act3:} & \quad \text{response} := \emptyset
\end{align*}
end
END

2.4 Password Encryption

In $\text{ATMR0}$ we model the mapping from account to password with a function 
\[
\text{accounts} \to \text{PASSWORD}.
\]
Looking ahead to implementation, we recognise that it would be unwise to implement a mapping from account to a plaintext password. It would be more secure to encrypt the password. To provide facilities for this we introduce a new machine $\text{Encryption}$.

We also specify the operation $\text{CheckPassword}$ as comparing encrypted passwords, rather than comparing plain passwords. Notice that we need to “think ahead” on this issue: if we specified the operation as comparing plain passwords, we could not later decide to implement the operation using comparison of encrypted passwords as this is weaker than comparing plain passwords and is hence not a refinement.
CONTEXT Encryption
EXTENDS Password
SETS CRYPT
CONSTANTS ENCRYPT

AXIOMS

axm1: \( ENCRYPT \in PASSWORD \rightarrow CRYPT \)

END

CONTEXT CardStatus1
EXTENDS CardStatus
CONSTANTS cryptpass We will store encrypted passwords, not plain passwords

AXIOMS

axm1: cryptpass \in validaccounts \rightarrow CRYPT
axm2: \( \forall acc \cdot acc \in validaccounts \Rightarrow cryptpass(acc) = ENCRYPT(password(acc)) \)

THEOREMS

thm1: \( \forall acc, pass \cdot acc \in validaccounts \Rightarrow (pass = password(acc) \Rightarrow ENCRYPT(pass) = cryptpass(acc)) \)

END

MACHINE ATMR1
REFINES ATMR0
SEES CardStatus1

VARIABLES response The variables of this machine model customer what we may think of as a User Interface. balance Each variable is a set that may be either empty money or contain a single value.

EVENTS

Initialisation

begin

act1: customer := ∅
act2: response := ∅
act3: balance := ∅
act4: money := ∅

end

Event InsertCard_ok ≜ Insert service card into ATM
refines InsertCard_ok

any scard pass account

when

grd1: customer = ∅
grd2: response = ∅
grd3: scard ∈ SCARD
grd4: account = GENSCARD⁻¹(scard)
grd5: account ∈ validaccounts

grd6: ENCRYPT(pass) = cryptpass(account)
then

act1: response := {OK}
act2: customer := {account}

end

Event InsertCard_nok ≜
refines InsertCard_nok

any scard pass account

when

grd1: customer = ∅
grd2: response = ∅
grd3: scard ∈ SCARD
grd4: account = GENSCARD⁻¹(scard)
grd5: account ∈ validaccounts ⇒ ENCRYPT(pass) ≠ cryptpass(account)
then

act1: response := {REFUSED}

end

Event Withdraw_ok ≜ Make withdrawal from ATM
refines Withdraw_ok
any amount account

when
grd1: customer \( \neq \emptyset \)
grd2: amount \( \in \mathbb{N} \)
grd3: customer = \{ account \}  
grd4: amount \leq withdrawlimit(account)  

then
act1: response := \{ OK \} 
act2: balance := \emptyset \cup \{ n \cdot n \in \mathbb{Z} | \{ n \} \}  
act3: money := \{ amount \}  

end

Event \( \text{Withdraw\_nok} \) \( \cong \)  
refines \( \text{Withdraw\_nok} \)  

any amount account  

when
grd1: customer \( \neq \emptyset \)  
grd2: amount \( \in \mathbb{N} \)  
grd3: customer = \{ account \}  
grd4: amount > withdrawlimit(account)  

then
act1: response := \{ REFUSED \}  
act2: balance := \emptyset  
act3: money := \emptyset  

end

Event \( \text{RemoveCard} \)  
refines \( \text{RemoveCard} \)  

Customer terminates session

when
grd1: customer \( \neq \emptyset \)  

then
act1: response := \{ OK \}  

act2 : \( \text{customer} := \emptyset \)
end

Event \( \text{ResetResponse} \triangleq \) Reset response when no customer using ATM
refines \( \text{ResetResponse} \)
when
grd1 : \( \text{customer} = \emptyset \)
grd2 : \( \text{response} \neq \emptyset \)
then
act1 : \( \text{response} := \emptyset \)
end

Event \( \text{ResetUI} \triangleq \) Reset User Interface
refines \( \text{ResetUI} \)
when
grd1 : \( \text{customer} \neq \emptyset \Rightarrow \text{money} \neq \emptyset \)
grd2 : \( \text{customer} \neq \emptyset \Rightarrow \text{balance} \neq \emptyset \)
grd3 : \( \text{customer} \neq \emptyset \Rightarrow \text{response} \neq \emptyset \)
then
act1 : \( \text{money} := \emptyset \)
act2 : \( \text{balance} := \emptyset \)
act3 : \( \text{response} := \emptyset \)
end

END