The Coffee Club

Manuel Carro

IMDEA Software Institute &
Universidad Politécnica de Madrid

Requirements for *The Coffee Club*

1. The Coffee Club has members.
2. The Members of the Coffee Club contribute with money.
3. The Coffee Club buys coffee in bulk using the contributed funds.
4. The Coffee Club never incurs in debts.
5. Coffee is bought at the discretion of the President of the Club.
6. Members can order coffee cups in the Club according to the funds they contributed.

Approach

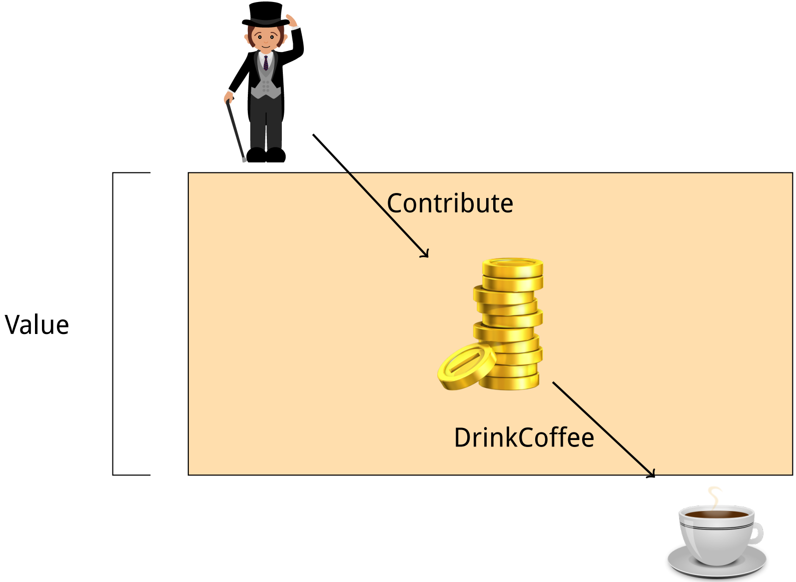
Simplifying Assumptions

- One cup of coffee costs one currency unit.
- Coffee is bought by cups.

Strategy: Three Models

1. Contribute to common funds, order from common funds.
   - No concept of members.
   - No separation cash / coffee.
2. Add the distinction cash / coffee.
3. Add the support for members and what every member contributed.

Initial Model



```
MACHINE CC_m0
VARIABLES
    value
INVARIANTS
    inv1:  value ∈ ℕ
EVENTS
Initialisation
    begin
        act1: value := 0
    end
```

```
Event Contribute ⟨ordinary⟩ ≙
    any
        am
    where
        grd1:  am > 0
    then
        act1: value := value + am
    end
Event DrinkCoffee ⟨ordinary⟩ ≙
    when
        grd1:  value > 0
    then
        act1: value := value − 1
    end
END
```

Types

- $\mathbb{N}, \mathbb{N}1, \mathbb{Z}$.
- Constants (arithmetic or not).
- Sets S (of perhaps unknown elements — example later).
- Relationships $R \times S$
  (and total, partial, injective, surjective, bijective functions as specific kinds of relationships — example later).

Reminder: Invariant Proof Obligation

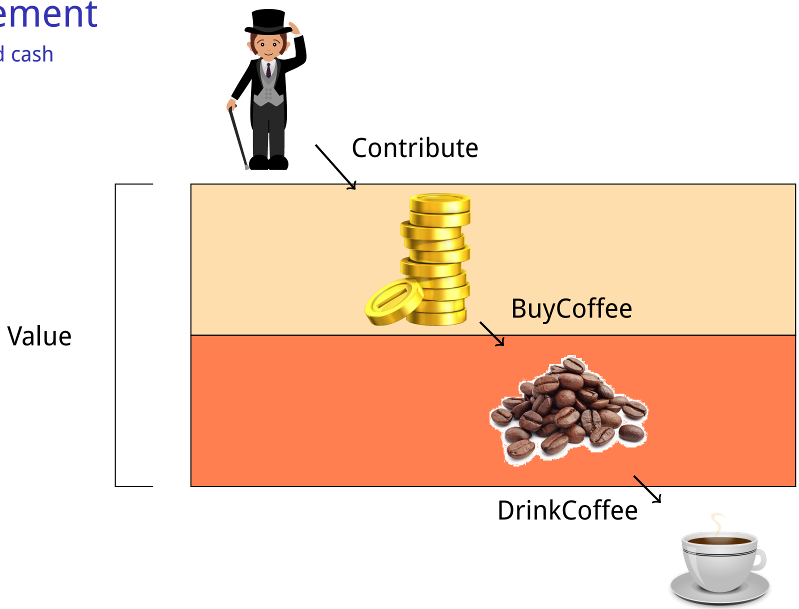
- Close to 40 types of proof obligations. For example, invariant preservation: for all event i, invariant j

$$A, G_i, I_{1\ldots n}(v) \vdash I_j(E_i(v))$$

  - A axioms
  - G_i guard of event i
  - $I_{1\ldots n}(v)$ all the invariants
  - $I_j(v)$ invariant j
  - $E_i(v)$ result of action i

First Refinement

Separate coffee and cash

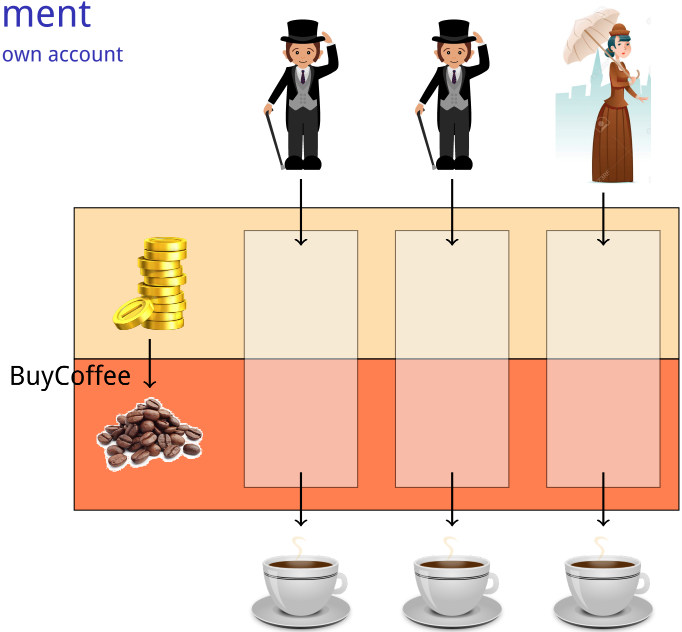


MACHINE CC_m1
REFINES CC_m0
VARIABLES
  cash
  coffee
INVARIANTS
  inv1: $cash \in \mathbb{N}$
  inv2: $coffee \in \mathbb{N}$
  inv3: $cash + coffee = value$
EVENTS
Initialisation
  begin
    act2: $cash := 0$
    act3: $coffee := 0$
  end
Event Contribute ⟨ordinary⟩ $\widehat{=}$
refines Contribute
  any
    am
  where
    grd1: $am > 0$
  then
    act2: $cash := cash + am$
  end

Event DrinkCoffee ⟨ordinary⟩ $\widehat{=}$
refines DrinkCoffee
  when
    grd2: $coffee > 0$
  then
    act2: $coffee := coffee - 1$
  end
Event BuyCofee ⟨ordinary⟩ $\widehat{=}$
  any
    am
  where
    grd1: $am > 0$
    grd2: $cash \geq am$
  then
    act1: $cash := cash - am$
    act2: $coffee := coffee + am$
  end
END

Second Refinement

Every member as his/her own account



CONTEXT CC_c2
SETS
MEMBER_IDS
AXIOMS
axm1: $finite(MEMBER\_IDS)$
END

VARIABLES
cash
coffee
account
INVARIANTS
inv1: $account \in MEMBER\_IDS \nrightarrow \mathbb{N}$

EVENTS
Initialisation ⟨extended⟩
begin
act2: $cash := 0$
act3: $coffee := 0$
act4: $account := \varnothing$
end
Event NewMember ⟨ordinary⟩ $\widehat{=}$
any
m
where
grd1: $m \in MEMBER\_IDS \setminus dom(account)$
then
act1: $account(m) := 0$
end

Event Contribute ⟨ordinary⟩ $\widehat{=}$
extends Contribute
any
am
m
where
grd1: $am > 0$
grd2: $m \in dom(account)$
then
act2: $cash := cash + am$
act3: $account(m) := account(m) + am$
end
Event DrinkCoffee ⟨ordinary⟩ $\widehat{=}$
extends DrinkCoffee
any
m
where
grd2: $coffee > 0$
grd3: $m \in dom(account)$
grd4: $account(m) > 0$
then
act2: $coffee := coffee - 1$
act3: $account(m) := account(m) - 1$
end

Can Every Member have its Cup of Coffee?

- What could happen if $m \notin dom(account)$ was not in the guard of $\mathrm{NewMember}$?
- Everyone who contributed should have coffee (or cash) available at the club.
- Strongest invariant:

$$coffee + cash = \sum_{m \in dom(acc)} f(m)$$

- However, $\sum_i f(i)$ not directly in Event B toolkit.
- Workaround necessary (e.g., plugin extended the basic Event B theory). See `bre.is/8YtsnRG5`
- Possible alternative: "there is coffee / cash for every single person":

$$\forall m \cdot (m \in dom(acc) \Rightarrow cash + coffee \geq acc(m))$$

- However:
  - Does it capture the requirement we want?
  - In any case, is it inductive?

Deadlock, Termination, Progress

Absence of Deadlocks

- Most of the time, we do not want deadlocks.
- Prove that a transition can always occur.
- Transitions enabled by guards $\Rightarrow$ prove that the disjunction of the guards is always true.
- Specialized version for absence of deadlock in refinements: "assuming that the previous machine does not deadlock...".

Termination

- Not always necessary (in a reactive system).
- `VARIANT EXP`: an expression that
  - Has a lower bound (e.g., $\mathtt{EXP} \in \mathbb{N}$, or `EXP` is a set).
  - Is decreased by every event.

Ensuring Progress

- Ensure that a set E of events does not *dominate* the execution and prevent others from firing.
- Use a `VARIANT` expression involving events in E.
- Does not imply termination: when events in E cannot proceed, other events not in E can fire and move the model to a state where events in E are enabled.