Exam Review II

Wednesday December 11

Math Models

REL. SET FUN

r1, r2 : REL[I, S]

Command

imp.

union (other : REL)

API ∪

infix |\/|

r1. union(r2)

modify r1

queries

contract

unioned (other : REL) : REL

r3 := r1. unioned (r2)

not modified

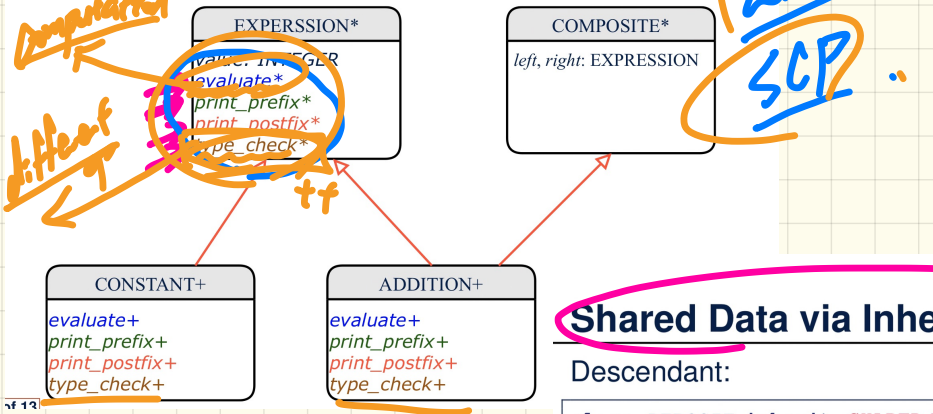
WP rules

$wp(x := e, R) = ?$

$wp(\text{if} \cdot\cdot \text{then} \cdot\cdot \text{else}, R) = ?$

$wp(S_1 ; S_2, R) = ?$

Extend the *composite pattern* to support *operations* such as `evaluate`, pretty printing (`print_prefix`, `print_postfix`), and `type_check`.



Shared Data via Inheritance

Descendant:

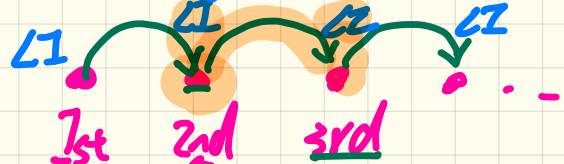
```
class DEPOSIT inherit SHARED_DATA
      -- 'maximum_balance' relevant
end

class WITHDRAW inherit SHARED_DATA
      -- 'minimum_balance' relevant
end

class INT_TRANSFER inherit SHARED_DATA
      -- 'exchange_rate' relevant
end

class ACCOUNT inherit SHARED_DATA
feature
      -- 'interest_rate' relevant
      deposits: DEPOSIT_LIST
      withdraws: WITHDRAW_LIST
end
```

Ancestor:

```
class
  SHARED_DATA
feature
  interest_rate: REAL
  exchange_rate: REAL
  minimum_balance: INTEGER
  maximum_balance: INTEGER
  ...
end
```

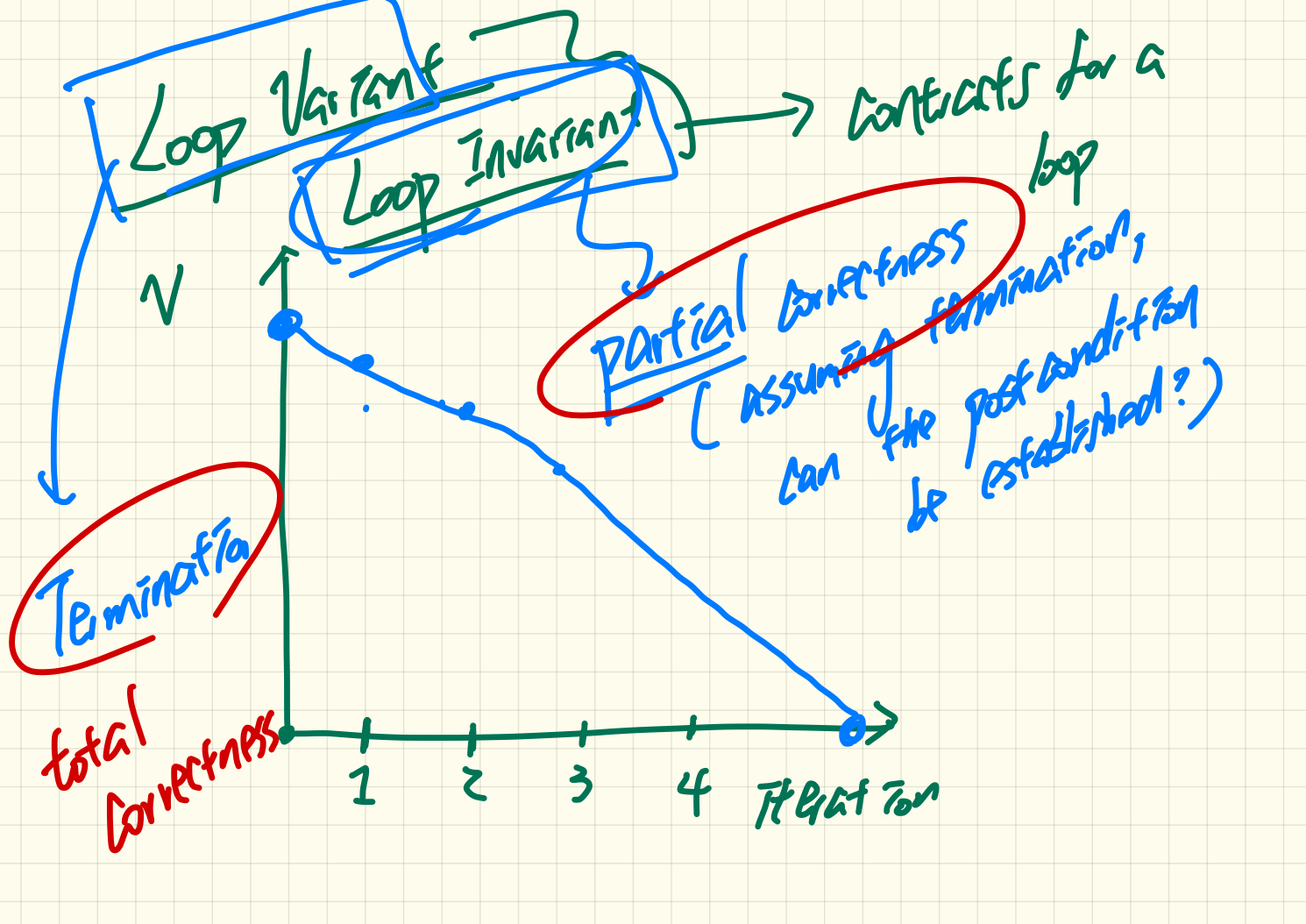
Problems?

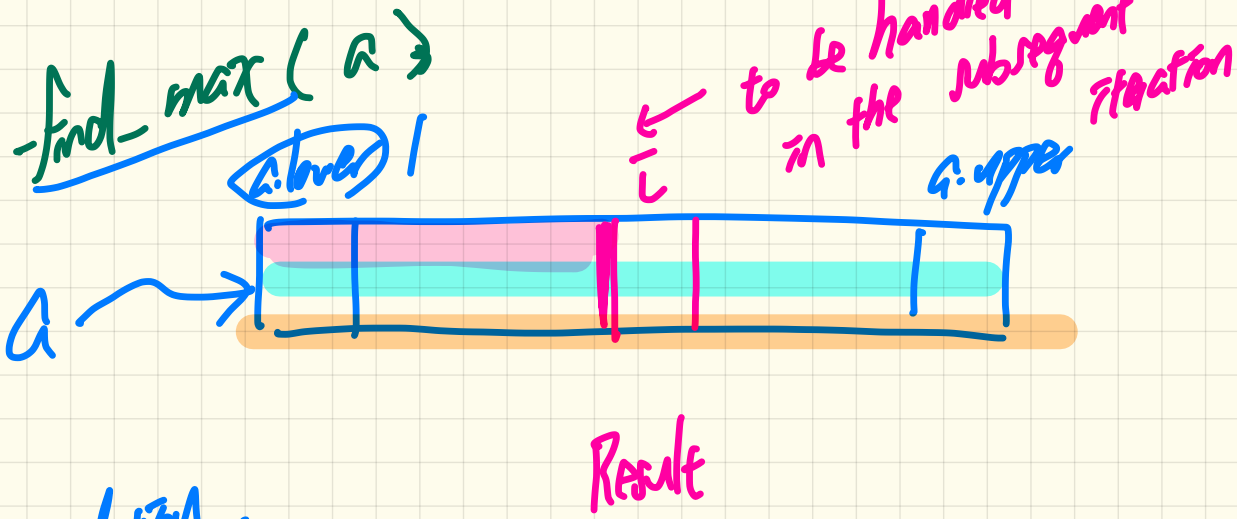
Correct Loops: Proof Obligations



{Q}
from
S_{init}
invariant
I
until
B
loop
S_{body}
variant
V
end {R}

- A loop is ***partially*** *correct* if:
  - Given precondition **Q**, the initialization step S_{init} establishes **LI** I.
    $$\{Q\}\ S_{init}\ \{I\}$$
  - At the end of S_{body}, if not yet to exit, **LI** I is maintained.
    $$\{I \wedge \neg B\}\ S_{body}\ \{I\}$$
  - If ready to exit and **LI** I maintained, postcondition **R** is established.
    $$I \wedge B \Rightarrow R$$
- A loop *terminates* if:
  - Given **LI** I, and not yet to exit, S_{body} maintains **LV** V as non-negative.
    $$\{I \wedge \neg B\}\ S_{body}\ \{V \geq 0\}$$
  - Given **LI** I, and not yet to exit, S_{body} decrements **LV** V.
    $$\{I \wedge \neg B \wedge V = v\}\ S_{body}\ \{V < v\}$$

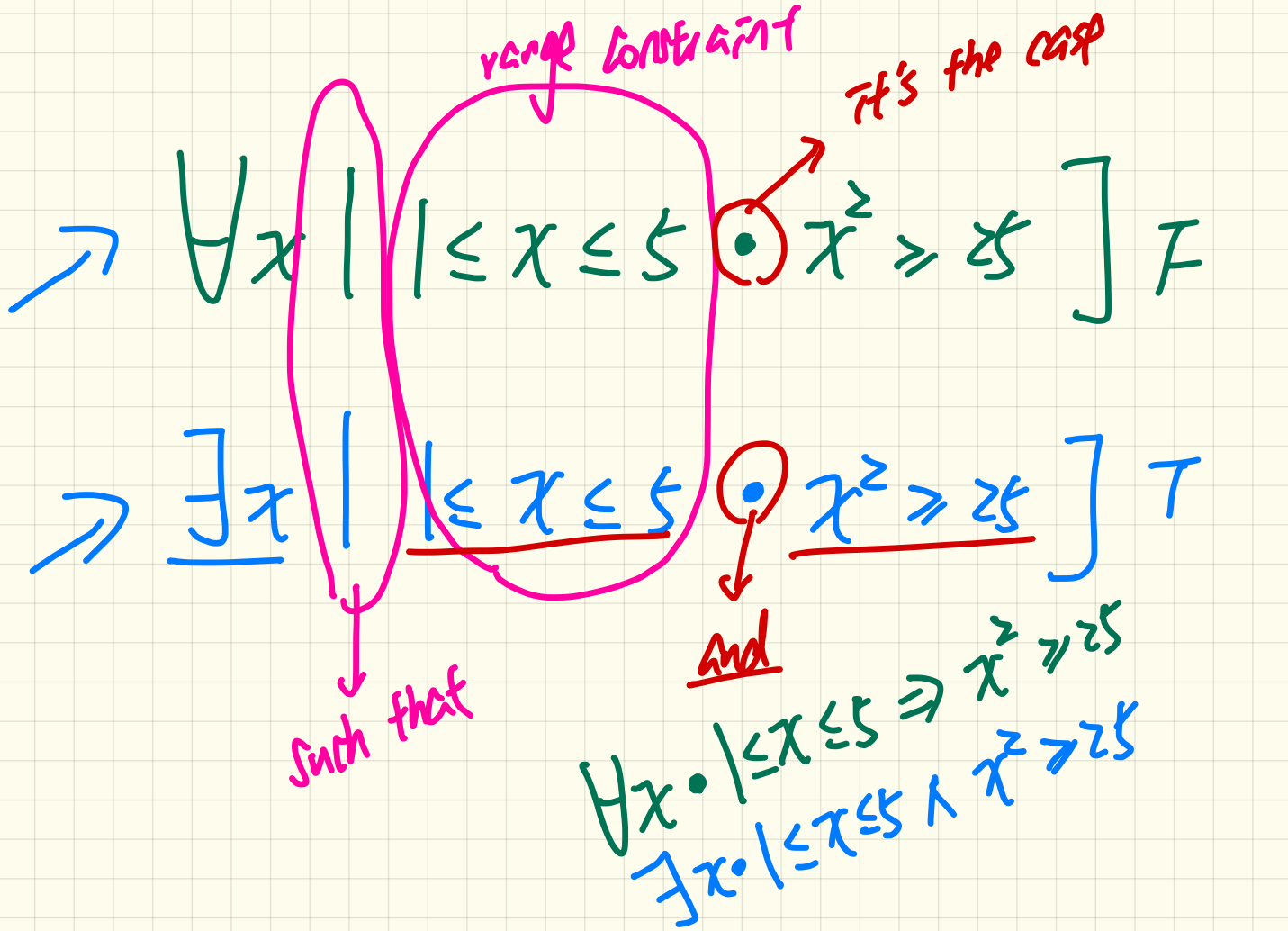

Loop Variant
Loop Invariant
→ contracts for a loop
V
Partial correctness (assuming termination, can the post condition be established?)
Termination
total correctness
1
2
3
4
Iteration



Postcondition

$$\forall i \mid 1 \le i \le a.Count \bullet Result \ge a[i]$$

$$\forall j \mid 1 \le j \le i-1 \bullet Result \ge a[j]$$


range constraint
it's the cap
$\forall x \mid 1 \le x \le 5 \bullet x^2 \ge 25 \rceil F$
$\exists x \mid 1 \le x \le 5 \bullet x^2 \ge 25 \rceil T$
such that
and
$\forall x \bullet 1 \le x \le 5 \Rightarrow x^2 \ge 25$
$\exists x \bullet 1 \le x \le 5 \wedge x^2 \ge 25$

Correct Loops: Proof Obligations

```
find_max (a: ARRAY [INTEGER]): INTEGER
  local i: INTEGER
  do
    from
      i := a.lower ; Result := a[i]
    invariant
      loop_invariant: ∀j | a.lower ≤ j < i • Result ≥ a[j]
    until
      i > a.upper
    loop
      if a [i] > Result then Result := a [i] end
      i := i + 1
    variant
      loop_variant: a.upper − i + 1
    end
  ensure
    correct_result: ∀j | a.lower ≤ j ≤ a.upper • Result ≥ a[j]
  end
end
```

Initialization:

Before Termination:

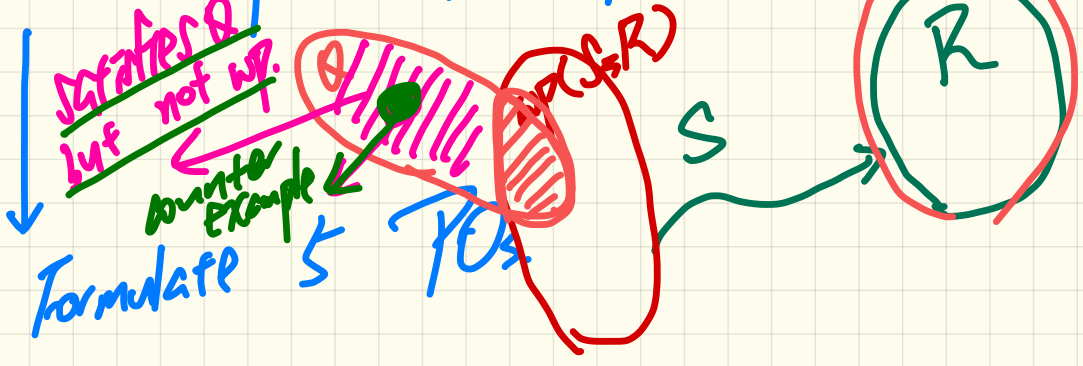
Upon Termination:

Non-Negative Variant:

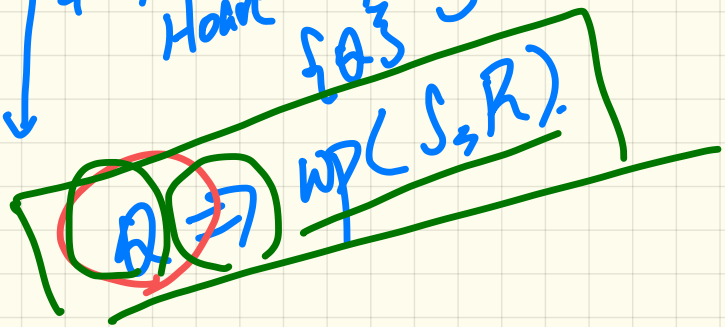
Decreasing Variant:

$\{ \forall j \mid a.lower \le j < i \cdot Result \ge a[j] \wedge \neg(a > a.upper) \}$

Given a loop (Eiffel syntax)



4 of them
Hoare Triples
$\{\theta\}\ S\ \{R\}$



```
max_of (x: INTEGER ; y: INTEGER)
  do
    Result := x
    if y > x then
      Result := y
    end
  ensure
    Result ≥ x ∧ Result ≥ y
  end
```

wp(if .. then ... else ... , R)

= [ ] ⇒ wp(..., ...)

∧

[ ] ⇒ wp(..., ...)

wp(R := x, P ∧ Q) = P ∧ Q[R := x]

0. Formulate:

{True} ——— { Q } ——— 2. True ⇒ (P ∧ Q)

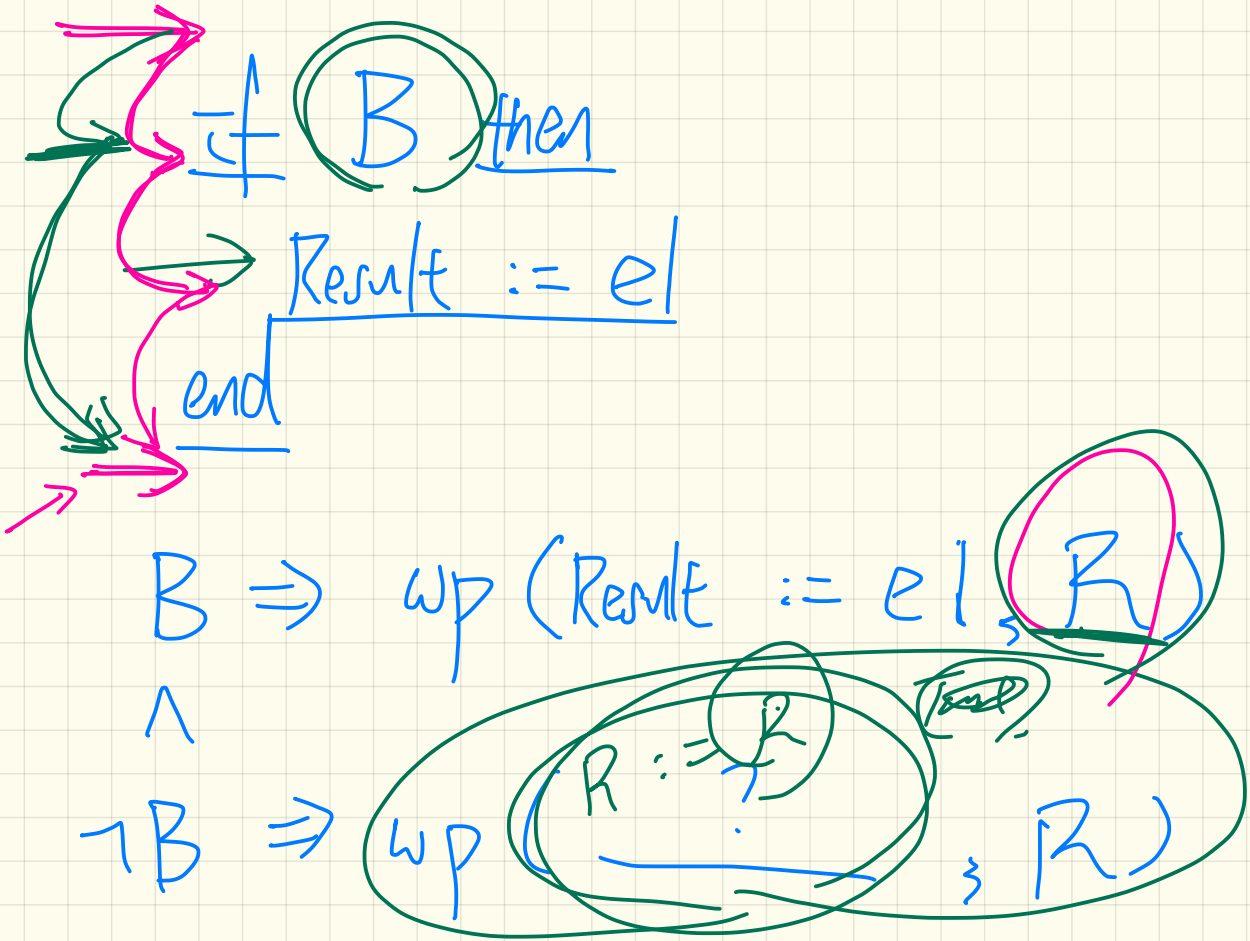
= P ∧ Q

✓ 1. Calculate wp(Result := x ; if y > x then Result := y else Result := Result , R ≥ x ∧ R ≥ y)

¬(y > x) ⇒ { rule of ; }

wp(R := R) = wp(Result := x, wp(——— , ———))

= R ≥ x ∧ R ≥ y


if B then
Result := e1
end
B ⇒ wp (Result := e1, R)
∧
¬B ⇒ wp (
R := R
end
, R)

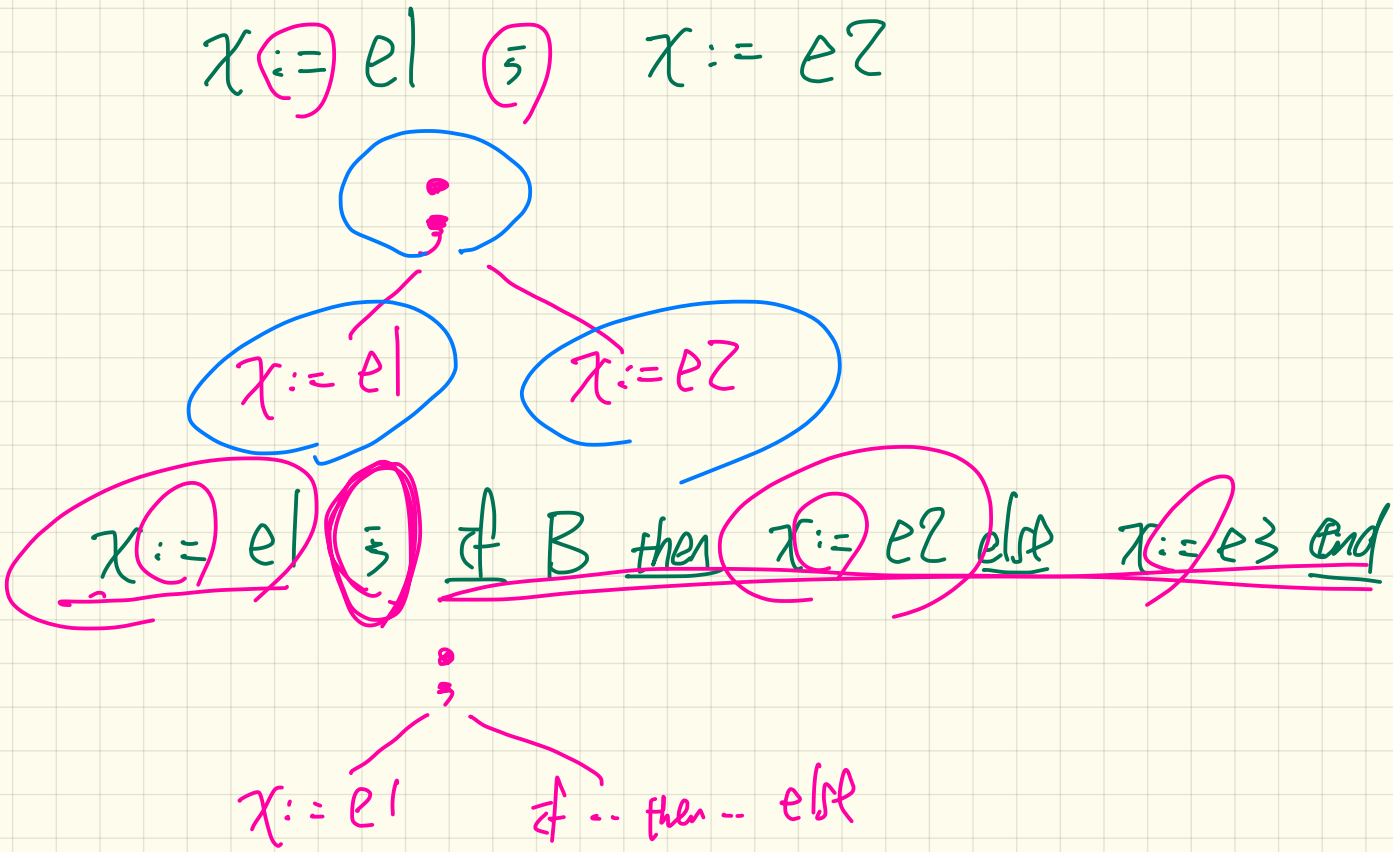
T

$P \Rightarrow (q \wedge r)$

$\equiv (p \Rightarrow q) \wedge (p \Rightarrow r)$

$q \wedge r$ $(x>0 \Rightarrow q) \wedge$

$(x>0 \Rightarrow r)$


x := e1 ; x := e2
;
x := e1
x := e2
x := e1 ; if B then x := e2 else x := e3 end
;
x := e1
if .. then .. else

```
require Q
do
  ! S
ensure
  - R1
  - R2
end
```

$\{Q\}\ S\ \{R_1 \wedge R_2\}$

$\{Q\}\ S\ \{R_1\}$

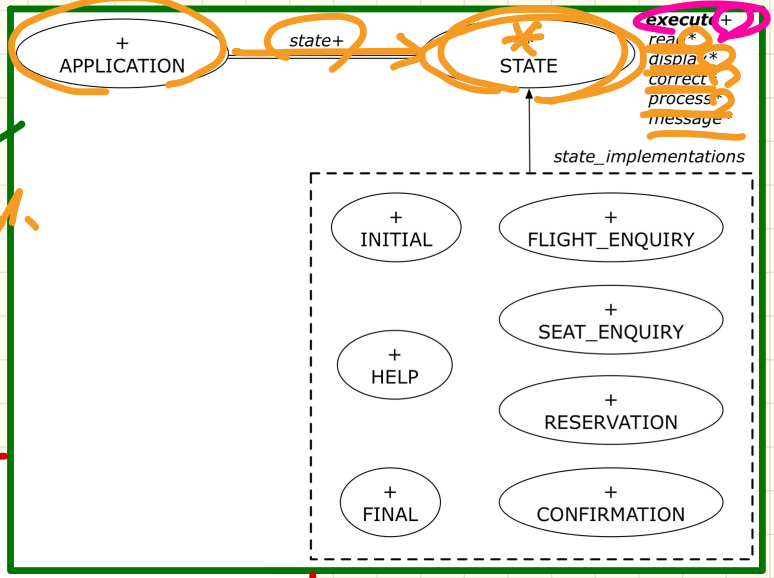
$\wedge$

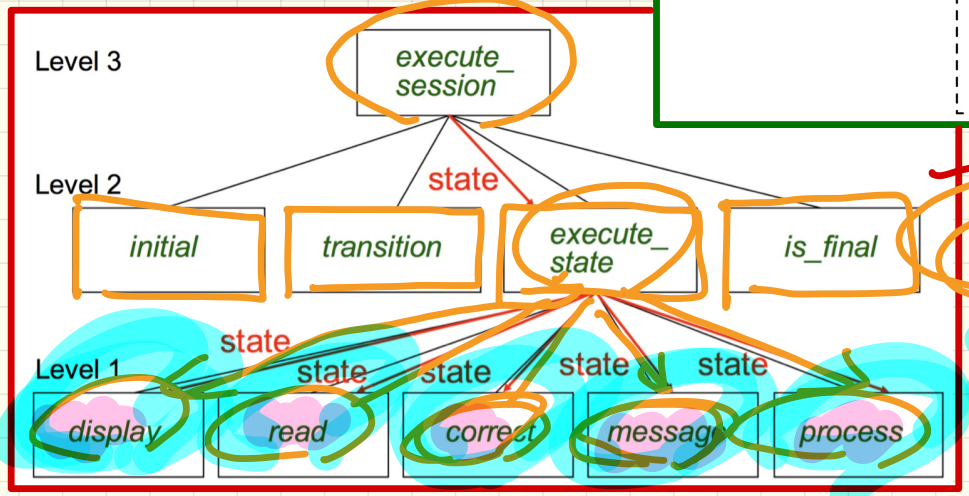
$\{Q\}\ S\ \{R_2\}$

Interactive System: Top-Down Design vs. OO Design

Object-Oriented

current_state: **STATE**
current_state.*execute_session*

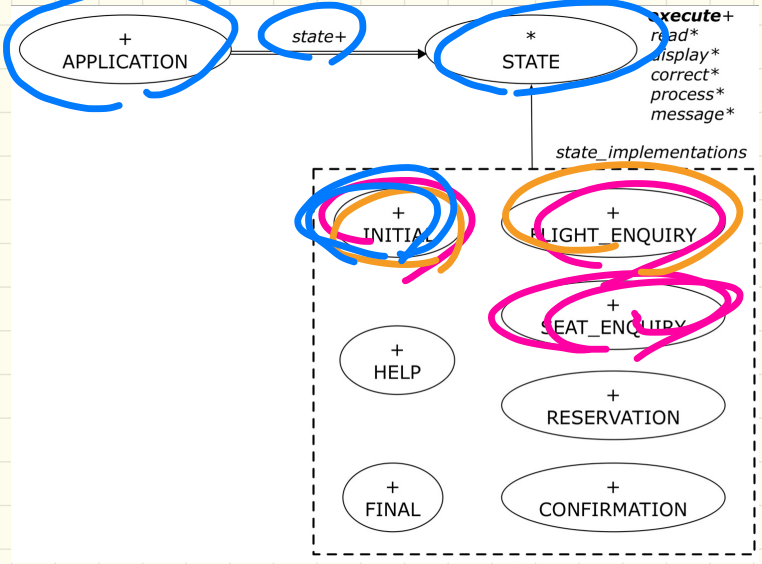




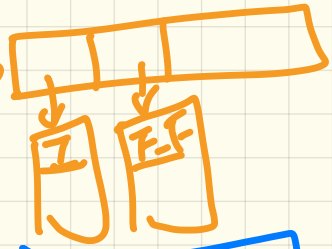
Top-Down

current_state: **INTEGER**
execute_session(current_stste)

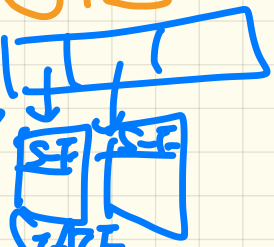
Code to the interface not to the imp.



states →



states2 →



class APP

state:

State

state2: SEAT_FULL

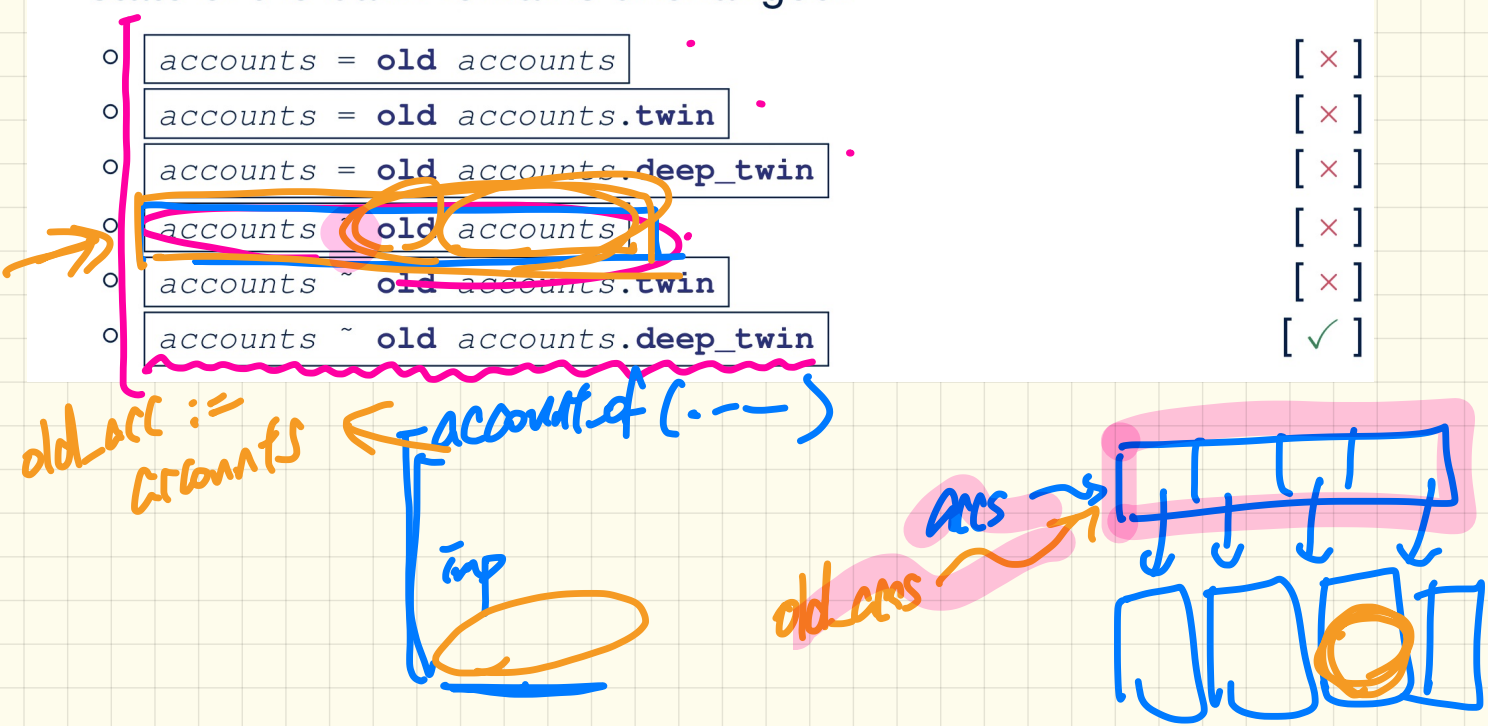
states: A[STATE]

states2: ALS_[ ]

: ACCOUNT

- Consider the query *account_of (n: STRING)* of *BANK*.
- How do we specify (part of) its postcondition to assert that the state of the bank remains unchanged:

`accounts = old accounts`	[×]
`accounts = old accounts.twin`	[×]
`accounts = old accounts.deep_twin`	[×]
`accounts ~ old accounts`	[×]
`accounts ~ old accounts.twin`	[×]
`accounts ~ old accounts.deep_twin`	[✓]



First Design Attempt

```
class Student {
   Course[] courses;
   int noc;
   int kind;
   double premiumRate;
   double discountRate;
   Student (int kind){
      this.kind = kind;
   }
   ...
}
```

not related to each other

Cohesion

```
double getTuition(){
   double tuition = 0;
   for(int i = 0; i < noc; i++){
      tuition += courses[i].fee;
   }
   if (this.kind == 1) {
      return tuition * premiumRate;
   }
   else if (this.kind == 2) {
      return tuition * discountRate;
   }
}
```

```
double register(Course c){
   int MAX;
   if (this.kind == 1) { MAX = 6; }
   else if (this.kind == 2) { MAX = 4; }
   if (noc == MAX) { /* Error */ }
   else {
      courses[noc] = c;
      noc++;
   }
}
```