Wednesday February 13

Lecture 11

Testing REL in MATHMODELS

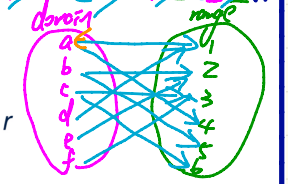
$r.\mathbf{overridden}(\{(a,3),(c,4)\})$

$= \underbrace{\{(a,3),(c,4)\}}_{t} \cup \underbrace{\{(b,2),(b,5),(d,1),(e,2),(f,3)\}}_{r.\mathbf{domain\_subtracted}(\underbrace{t.\mathbf{domain}}_{\{a,c\}})}$

$= \{(a,3),(c,4),(b,2),(b,5),(d,1),(e,2),(f,3)\}$

```
test_rel: BOOLEAN
  local
    r, t: REL[STRING, INTEGER]
    ds: SET[STRING]
  do
    create r.make_from_tuple_array (
      <<["a", 3], ["b", 2], ["c", 3],
        ["a", 4], ["b", 5], ["c", 6],
        ["d", 1], ["e", 2], ["f", 3]>>)
    create ds.make_from_array (<<"a">>)
    -- r is not changed by the query 'domain_subtracted'
    t := r.domain_subtracted (ds)
    Result :=
      t /~ r and not t.domain.has ("a") and r.domain.has ("a")
    check Result end
    -- r is changed by the command 'domain_subtract'
    r.domain_subtract (ds)
    Result :=
      t ~ r and not t.domain.has ("a") and not r.domain.has ("a")
  end
```

Say $r = \{(a,1),(b,2),(c,3),(a,4),(b,5),(c,6),(d,1),(e,2),(f,3)\}$



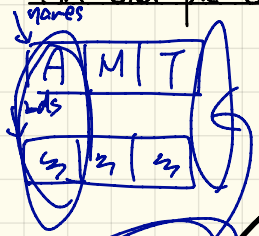
- r.*domain*: set of first-elements from *r*
  - r.**domain** = $\{ d \mid (d,r) \in r \}$
  - e.g., r.**domain** = $\{a,b,c,d,e,f\}$
- r.*range*: set of second-elements from *r*
  - r.**range** = $\{ r \mid (d,r) \in r \}$
  - e.g., r.**range** = $\{1,2,3,4,5,6\}$
- r.*inverse*: a relation like *r* except elements are in reverse order
  - r.**inverse** = $\{ (r,d) \mid (d,r) \in r \}$
  - e.g., r.**inverse** = $\{(1,a),(2,b),(3,c),(4,a),(5,b),(6,c),(1,d),(2,e),(3,f)\}$
- r.*domain_restricted*(ds): sub-relation of *r* with domain *ds*.
  - r.**domain_restricted**(ds) = $\{ (d,r) \mid (d,r) \in r \wedge d \in ds \}$
  - e.g., r.**domain_restricted**({a, b}) = {(**a**,1),(**b**,2),(**a**,4),(**b**,5)}
- r.*domain_subtracted*(ds): sub-relation of *r* with domain not *ds*.
  - r.**domain_subtracted**(ds) = $\{ (d,r) \mid (d,r) \in r \wedge d \notin ds \}$
  - e.g., r.**domain_subtracted**({a, b}) = {(**c**,6),(**d**,1),(**e**,2),(**f**,3)}
- r.*range_restricted*(rs): sub-relation of *r* with range *rs*.
  - r.**range_restricted**(rs) = $\{ (d,r) \mid (d,r) \in r \wedge r \in rs \}$
  - e.g., r.**range_restricted**({1, 2}) = $\{(a,1),(b,2),(d,1),(e,2)\}$
- r.*range_subtracted*(ds): sub-relation of *r* with range not *ds*.
  - r.**range_subtracted**(rs) = $\{ (d,r) \mid (d,r) \in r \wedge r \notin rs \}$
  - e.g., r.**range_subtracted**({1, 2}) = $\{(c,3),(a,4),(b,5),(c,6)\}$

r relation

ds set of pairs

$$\left(\begin{array}{l} r.\ \text{domain\_restricted}\ (ds) \\ \mid\backslash\mid \\ r.\ \text{domain\_subtracted}\ (ds) \end{array} \right) \sim r$$

An Example Birthday Book



hash_table

→ b.put("Jim", Nov-29)

→ b.put("Alan", Aug-23)

overridden by

domain

range

b.ran_res(oct-15) domain

"Jim"

b.ran_MB(oct-15)

Nov-29

"Alan"

Aug-23

August-11

"Mark"

"Tom"

October-15

b.remind(oct-15) << Mark, Tom >

b.remind(aug-23) << alan >>

b.remind(feb-13) << >>

Birthday Book : Design

m: 2
d: 30

BIRTHDAY_BOOK

model: FUN[NAME, BIRTHDAY]
-- abstraction function

count: INTEGER
-- number of entries

put(n: NAME; d: BIRTHDAY)
ensure
model_operation: model ~ (**old** model.deep_twin).overriden_by ([n,d])
-- infix symbol for override operator: @<+

remind(d: BIRTHDAY): ARRAY[NAME]
ensure
nothing_changed: model ~ (**old** model.deep_twin)
same_counts: **Result**.count = (model.range_restricted_by(d)).count
same_contents: ∀ name ∈ (model.range_restricted_by(d)).domain: name ∈ **Result**
-- infix symbol for range restriction: model @> (d)

invariant:
consistent_book_and_model_counts: count = model.count

model ~ (old model.deep_twin).overriden_by ([n,d])

@<+ old model.dt not necessary

model: FUN[NAME, ...]

(m = 1 ∨ m = 3 ...) ⇒ d = 31

(m = 4 ∨ m = 6 ∨ ...) ⇒ d = 30

remind: ARRAY[NAME]

BIRTHDAY

day: INTEGER
month: INTEGER
year: INTEGER
invariant
1 ≤ month ≤ 12
1 ≤ day ≤ 31

is_leap_year ∧ m = 2 ⇒ d = 29

NAME

item: STRING
invariant
item[1] ∈ A..Z

across Result as (n) all model[n.item] ~ d end

name: STRING

name: NAME
add (s: STRING ; ...)
require

name → "@#()"

name → [NAME | item] → "@#()"

inv. violation

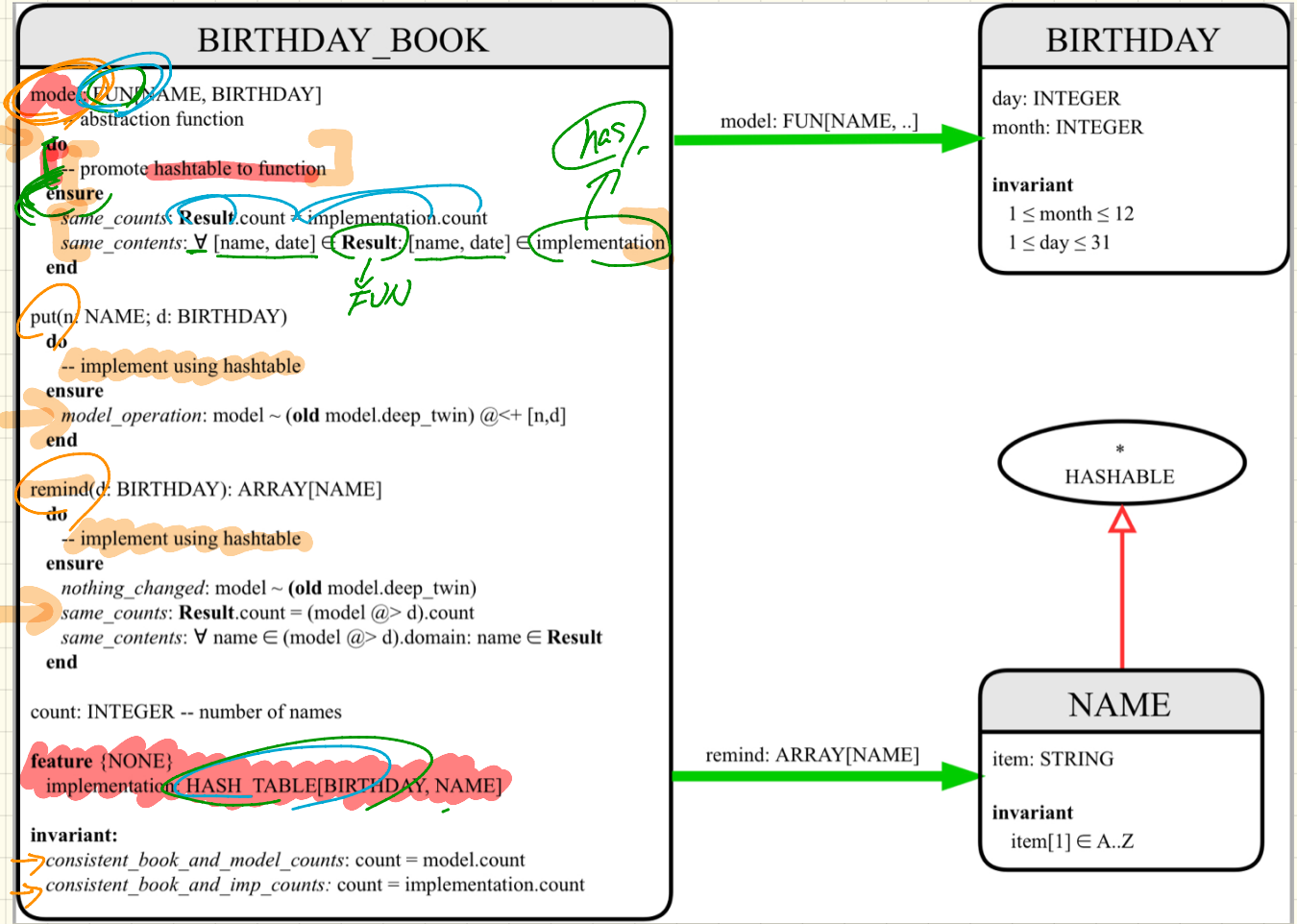
(V2) ✓

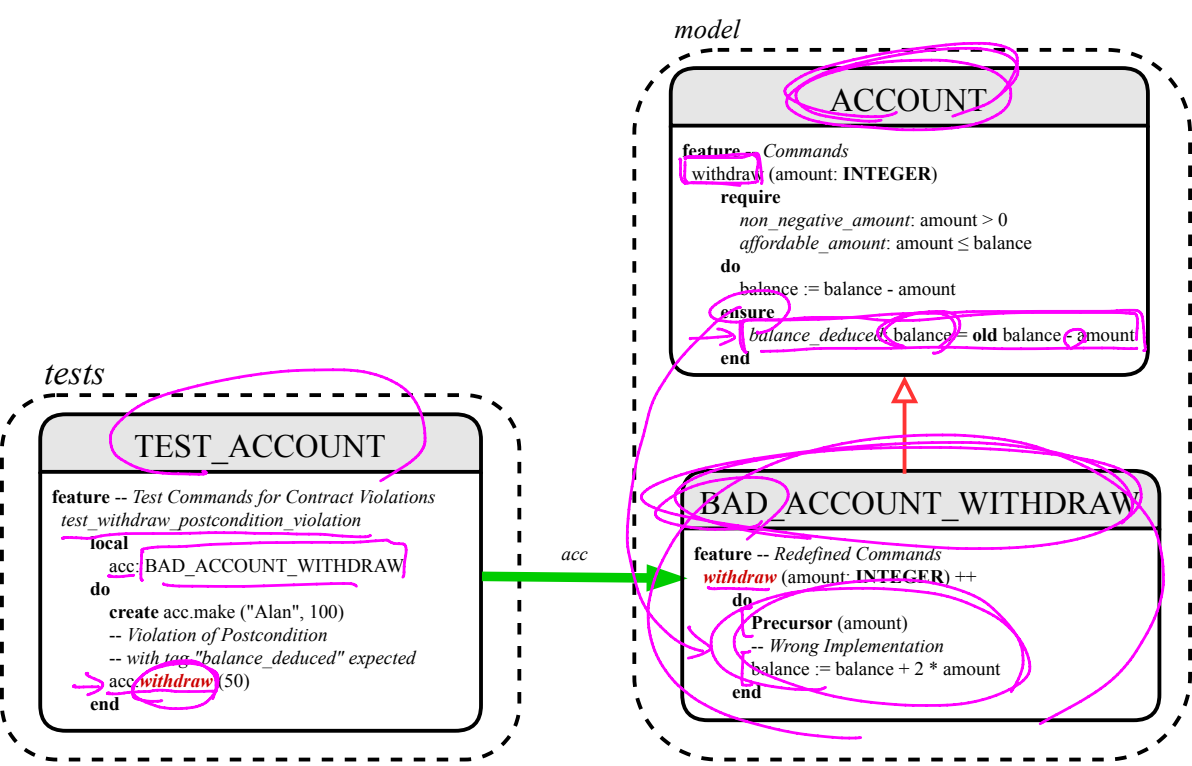
across model.ran_res_by(d).domain as n

Result.has(n.item)

end

Birthday Book: Implementation




model
ACCOUNT
feature -- Commands
withdraw (amount: INTEGER)
require
non_negative_amount: amount > 0
affordable_amount: amount ≤ balance
do
balance := balance - amount
ensure
balance_deduced: balance = old balance - amount
end
tests
TEST_ACCOUNT
feature -- Test Commands for Contract Violations
test_withdraw_postcondition_violation
local
acc: BAD_ACCOUNT_WITHDRAW
do
create acc.make ("Alan", 100)
-- Violation of Postcondition
-- with tag "balance_deduced" expected
acc.withdraw (50)
end
acc
BAD_ACCOUNT_WITHDRAW
feature -- Redefined Commands
withdraw (amount: INTEGER) ++
do
Precursor (amount)
-- Wrong Implementation
balance := balance + 2 * amount
end

Adding Postcondition Tests

```
class TEST_ACCOUNT
inherit ES_TEST
create make
feature -- Constructor for adding tests
  make
    do
      add_violation_case_with_tag ("balance_deducted",
        agent test_withdraw_postcondition_violation)
    end
feature -- Test commands (test to fail)
  test_withdraw_postcondition_violation
    local
      acc: BAD_ACCOUNT_WITHDRAW
    do
      comment ("test: expected postcondition violation of withdraw")
      create acc.make ("Alan", 100)
      -- Postcondition Violation with tag "balance_deduced" to occur.
      acc.withdraw (50)
    end
end
```