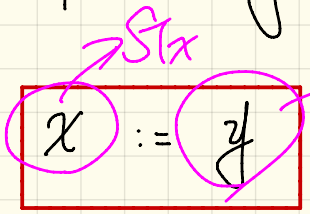
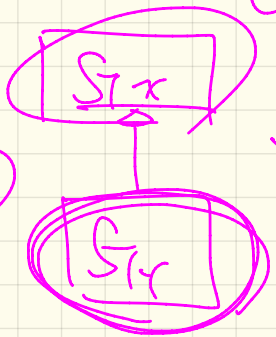
Wednesday February 27

Lecture 13

Type Checking Rules (1)

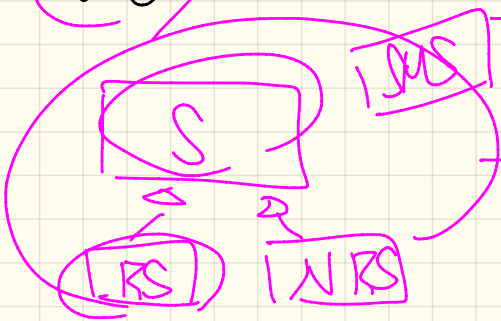


STy is a descendant of STx



s1 : STUDENT
s2 : RS
s3 : NRS

S
s1 := s2 RS
s1 := s3
s2 := s1
RS
s3 := s1 S
s2 := s3
s3 := S → not declared!



Type Checking Rules (2)

```
check attached {C} y then
  ...
end
```

```
class SMS
  get_s(i: INTEGER): STUDENT
    do
      ...
    end
end
```

C is either ancestor of ST of y (upward) or descendant of ST of y (downward)

```
sms: SMS
s1: STUDENT
s2: RS
s3: IRS
```

```
check attached {RS} s1 then ... end
check attached {STUDENT} s2 then ... end
check attached {SMS} s1 then ... end
check attached {RS} s3 then ... end
check attached {RS} sms.get(1) then ... end
```

Type Checking Rules (3)

```
class SMS
    get_S(i: INTEGER): STUDENT
    do
        ...
    end
end
```

```
check attached {C} y as temp then
    x := temp
end
```

C an. de. of ST

ST temp = C

ST x

C

sms: SMS

s1: STUDENT

s2: RS

s3: JRS

→

```
check attached {RS} sms.get_s(1) as temp then
    s1 := temp
    s3 := temp
end
```

Feature Call Arguments : Client

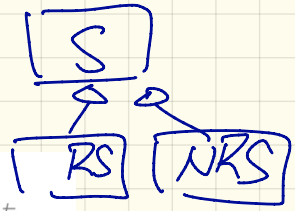
```
class STUDENT_MANAGEMENT_SYSTEM {
  ss : ARRAY [STUDENT]  -- ss[i] has static type Student
  add_s (s: STUDENT) do ss[0] := s end
  add_rs (rs: RESIDENT_STUDENT) do ss[0] := rs end
  add_nrs (nrs: NON_RESIDENT_STUDENT) do ss[0] := nrs end
```

```
test_polymorphism_feature_arguments
  local
    s1, s2, s3: STUDENT
    rs: RESIDENT_STUDENT ; nrs: NON_RESIDENT_STUDENT
    sms: STUDENT_MANAGEMENT_SYSTEM
  do
    create sms.make
    create {STUDENT} s1.make ("s1")
    create {RESIDENT_STUDENT} s2.make ("s2")
    create {NON_RESIDENT_STUDENT} s3.make ("s3")
    create {RESIDENT_STUDENT} rs.make ("rs")
    create {NON_RESIDENT_STUDENT} nrs.make ("nrs")
```

sms.add_s (rs)

S := rs

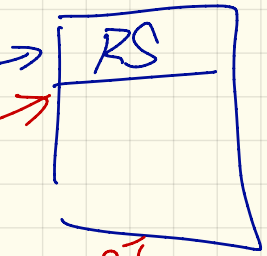
sms.add_rs (s1)



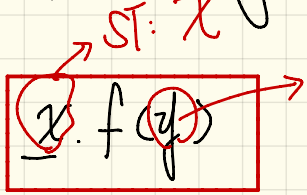
ST: STUDENT

ss[1]

ss[ss.count]



Type Checking Rules (4)

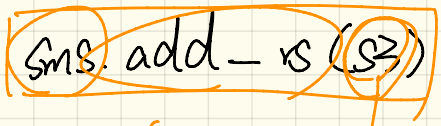


① feature f is declared in X

② ST of y is a des. of formal par. type of f.

sms : SMS
s1 : STUDENT
s2 : RS
s3 : NRS

s1.add_rs(s2)
sms.add_rs(s1)
sms.add_rs(s2)
sms.add_rs(s3)



X

ST: NRS

S := S3

Is NRS descendat of RS ?

NO

```
class SMS
  add_rs (s: RS)
  do
  end ...
end
```

Type Checking Rules (5)

```
check attached {C} y as temp then
    x.f(temp)
end
```

temp : C

```
class SMS
    get_s(i: INTEGER): STUDENT
    do
        ...
    end
    add_rs(s: RS)
    do
        ...
    end
```

sms : SMS

s1 : STUDENT

s2 : RS

s3 : NRS

```
check attached {RS} sms.get_s(1) as temp then
    sms.add_rs(temp)
end
```

```
check attached {NRS} sms.get_s(1) as temp then
    sms.add_rs(temp)
end
```

Polymorphic Collection


```
class STUDENT_MANAGEMENT_SYSETM
  students: LINKED_LIST[STUDENT]
  add_student(s: STUDENT)
    do
      students.extend (s)
    end
  registerAll (c: COURSE)
    do
      across
        students as s
      loop
        s.item.register (c)
      end
    end
end
```

```
test_sms_polymorphism: BOOLEAN
  local
    rs: RESIDENT_STUDENT
    nrs: NON_RESIDENT_STUDENT
    c: COURSE
    sms: STUDENT_MANAGEMENT_SYSTEM
  do
    create rs.make ("Jim")
    rs.set_pr (1.5)
    create nrs.make ("Jeremy")
    nrs.set_dr (0.5)
    create sms.make
    sms.add_s (rs)
    sms.add_s (nrs)
    create c.make ("EECS3311", 500)
    sms.register_all (c)
    Result := sms.ss[1].tuition = 750 and sms.ss[2].tuition = 250
  end
```

Feature Call Return Value

Supplier

```
class STUDENT_MANAGEMENT_SYSTEM {
  ss: LINKED_LIST[STUDENT]
  add_s (s: STUDENT)
    do
      ss.extend (s)
    end
  get_student(i: INTEGER): STUDENT
    require 1 <= i and i <= ss.count
    do
      Result := ss[i]
    end
end
```

ST: STUDENT

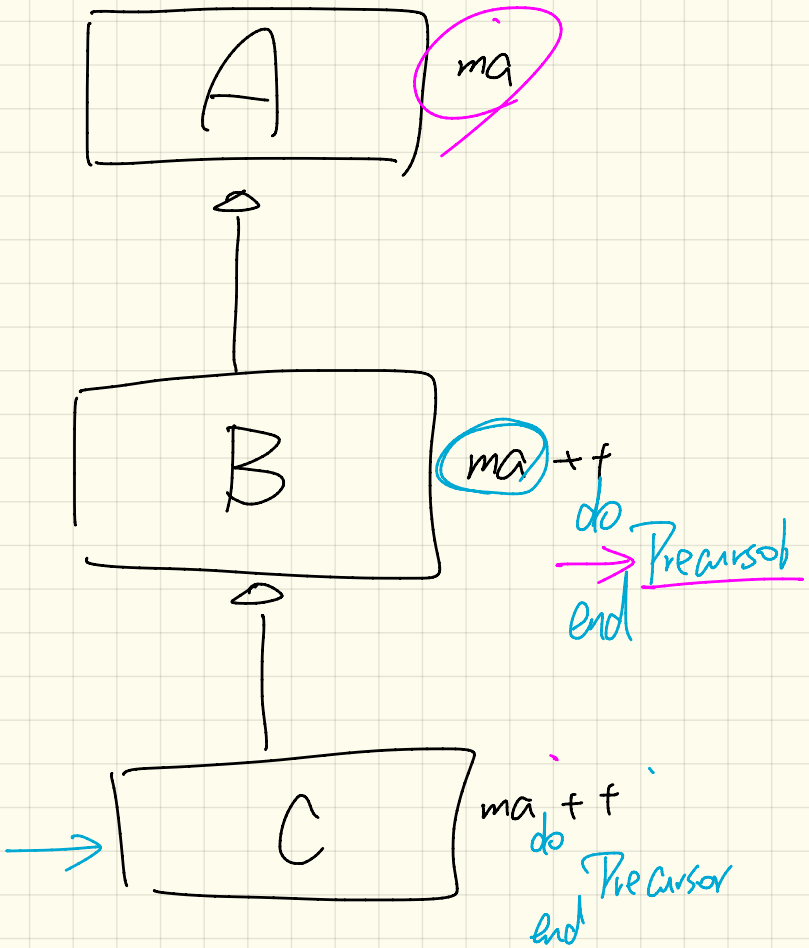
ST: S

Client

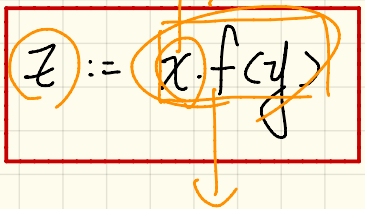
```
test_sms_polymorphism: BOOLEAN
local
  rs: RESIDENT_STUDENT ; nrs: NON_RESIDENT_STUDENT
  c: COURSE ; sms: STUDENT_MANAGEMENT_SYSTEM
do
  create rs.make ("Jim") ; rs.set_pr (1.5)
  create nrs.make ("Jeremy") ; nrs.set_dr (0.5)
  create sms.make ; sms.add_s (rs) ; sms.add_s (nrs)
  create c.make ("EECS3311", 500) ; sms.register_all (c)
  Result :=
      get_student (1).tuition = 750
    and get_student (2).tuition = 250
end
```

ST	DT
S	RS
S	NRS

Q: Possible DTs of Result?


A
ma
B
ma + f
do
Precursor
end
C
ma + f
do
Precursor
end

Type Checking Rules (6)




```
class SMS
    get_S(i: INTEGER): STUDENT
    do
      ...
    end
end
```

sms: SMS
s1: STUDENT
s2: RS
s3: NRS

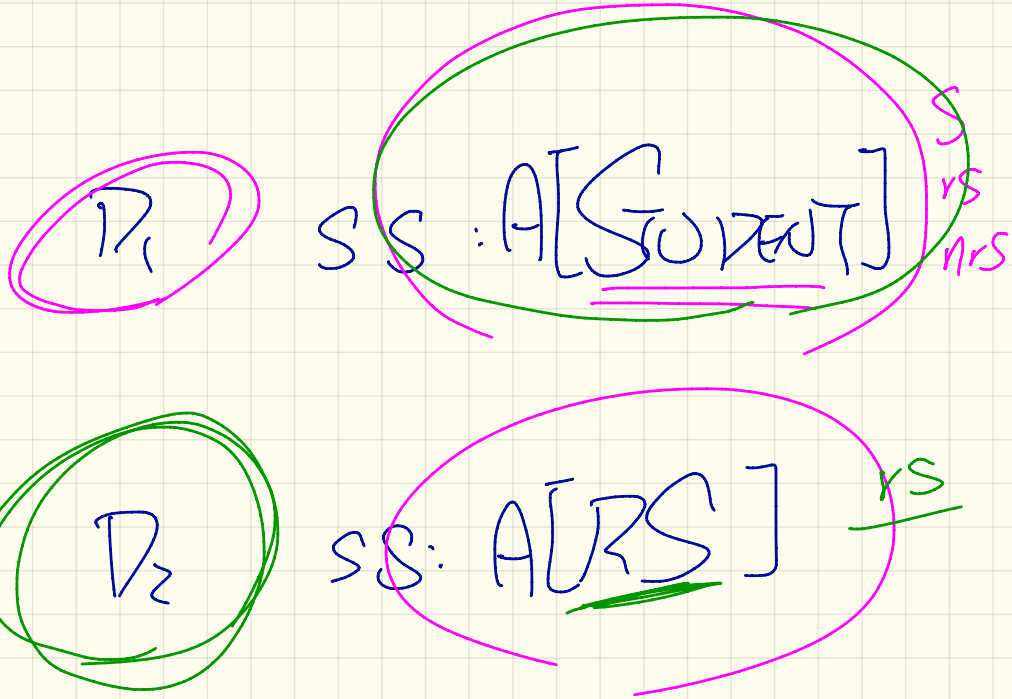
→ s1 := s2.get_S(1)

s1 := sms.get_S(1)

s2 := sms.get_S(1) → s1: STUDENT

s3 := sms.get_S(1)

→ sms.get_S("2")


D1
SS : A[STUDENT]
S
rs
nrs
SS[i] . regulation pr x
D2
SS: A[RS]
rs
SS[i] . reg enr- pr self-pr
D3, SS: A[NRS]

General Book

Supplier

```
class BOOK
  names: ARRAY[STRING]
  records: ARRAY[ANY]
  -- Create an empty book
  make do ... end
  -- Add a name-record pair to the book
  add (name: STRING; record: ANY) do ... end
  -- Return the record associated with a given name
  get (name: STRING): ANY do ... end
end
```

Client

```
1  birthday: DATE; phone_number: STRING
2  b: BOOK; is_wednesday: BOOLEAN
3  create {BOOK} b.make
4  phone_number := "416-677-1010"
5  b.add ("SuYeon", phone_number)
6  create {DATE} birthday.make(1975, 4, 10)
7  b.add ("Yuna", birthday)
8  is_wednesday := b.get("Yuna").get_day_of_week = 4
```

ST: ANY