Programozási nyelvek és paradigmák

Variancia

Kozsik Tamás (2020)

Sok nyelv megengedi a kovariáns visszatérési értéket!

```
deferred class FOOD end
class MILK inherit FOOD ...

class ANIMAL
feature
    prefers: detachable FOOD do end
...

class CAT
inherit ANIMAL redefine prefers end
feature
    prefers: attached MILK do create Result end
...
```

Kovariáns visszatérési érték: formálisan

- Legyen $A <: B$, és legyen T-ben egy f függvény B visszatérési típussal. Az $S <: T$ akkor is fennáll, ha S újradeklarálja f-et A visszatérési típussal.

Kovariáns visszatérési érték: formálisan

- Legyen $A <: B$, és legyen T-ben egy f függvény B visszatérési típussal. Az $S <: T$ akkor is fennáll, ha S újradeklarálja f-et A visszatérési típussal.
- Együtt változás

```
class T                    class S inherit T redefine f end
feature f: B               feature f: A
            ...                        ...
```

Kovariáns visszatérési érték: formálisan

- Legyen $A <: B$, és legyen T-ben egy f függvény B visszatérési típussal. Az $S <: T$ akkor is fennáll, ha S újradeklarálja f-et A visszatérési típussal.
- Együtt változás

```
class T                  class S inherit T redefine f end
feature f: B             feature f: A
          ...                      ...
```

- LSP: `object_T.f(params)` helyett `object_S.f(params)`

Kovariáns visszatérési érték: formálisan

- Legyen $A <: B$, és legyen T-ben egy f függvény B visszatérési típussal. Az $S <: T$ akkor is fennáll, ha S újradeklarálja f-et A visszatérési típussal.
- Együtt változás

```
class T                  class S inherit T redefine f end
feature f: B             feature f: A
          ...                      ...
```

- LSP: `object_T.f(params)` helyett `object_S.f(params)`
- Újradeklarál: pl. újradefiniál vagy effektívvé tesz

Kovariáns visszatérési érték: formálisan

- Legyen $A <: B$, és legyen T-ben egy f függvény B visszatérési típussal. Az $S <: T$ akkor is fennáll, ha S újradeklarálja f-et A visszatérési típussal.
- Együtt változás

```
class T                  class S inherit T redefine f end
feature f: B             feature f: A
          ...                      ...
```

- LSP: `object_T.f(params)` helyett `object_S.f(params)`
- Újradeklarál: pl. újradefiniál vagy effektívvé tesz
- Speciális eset: $A \equiv S$ és $B \equiv T$

Java: Object.clone

```
package java.lang;
public class Object {
    ...
    protected Object clone()
                        throws CloneNotSupportedException
    {
        if( this instanceof Cloneable )
            return [magic shallow copy];
        else
            throw new CloneNotSupportedException();
    }
}
```

Java: Object.clone szabályszerű felüldefiniálása

```
package java.lang;
public class Object {
    protected Object clone()
                        throws CloneNotSupportedException
    ...
}

public class Point implements Cloneable {
    ...
    @Override public Point clone() {
        try { return (Point) super.clone(); }
        catch( ClassCastException
             | CloneNotSupportedException e){
            assert false;  // cannot happen
            return null;
        }
    }
}
```

Java: `Object.clone` szabályszerű felüldefiniálása

```
package java.lang;
public class Object {
    protected Object clone()
                        throws CloneNotSupportedException
    ...
}

public class Point implements Cloneable {
    ...
    @Override public Point clone() { ... }
}
```

- Kovariancia normális és abnormális visszatérési értékre
- Láthatóság bővülhet?
- LSP: `object_T.f(params)` helyett `object_S.f(params)`

Klónozás Eiffelben?

```
class ANY
feature
    clone: attached ANY do [magic shallow copy] end
...


class POINT
inherit ANY redefine clone end
feature
    clone: attached POINT
        do
            check attached {POINT} Precursor as p then
                Result := p
            end
        end
...
```

Robusztus megoldás: kapcsolt típussal (anchored type)

```
class ANY
...
feature
    frozen twin: attached like Current
        do
            [magic shallow copy]
        end
...
end -- class ANY
```

- Nem kell újradeklarálni
- Adaptív a típus

Ismétlés: kovariáns visszatérési érték

```
deferred class FOOD end
class MILK inherit FOOD ...

class ANIMAL
feature
    prefers: detachable FOOD do end
...

class CAT
inherit ANIMAL redefine prefers end
feature
    prefers: attached MILK do create Result end
...
```

Ismétlés: covariant return – aktuális típus

```
class COMPLEX
feature times alias "*"( other: COMPLEX ): COMPLEX ...
...

class POLAR_COMPLEX inherit COMPLEX
feature times alias "*" ( other: COMPLEX ): POLAR_COMPLEX
...

class CART_COMPLEX inherit COMPLEX
feature times alias "*"( other: COMPLEX ): CART_COMPLEX
...
```

```
cc: CART_COMPLEX                c  := cc * cp  -- ok
pc: POLAR_COMPLEX               c  := c  * c   -- ok
c : COMPLEX                     cc := cc * cp  -- ok
```

Kapcsolt típus

```
class HOLDER[T]
create
    set
feature
    item: attached T assign set

    set( value: like item )
        do
            item := value
        ensure
            item = value
        end
end
```

Ismétlés: automatikusan kovariáns visszatérési érték

```
class ANY
...
feature
    frozen twin: attached like Current
        do
            [magic shallow copy]
        end
...
end -- class ANY
```

- Nem kell újradeklarálni
- Adaptív a típus

```
p, q: attached POINT
...
p := q.twin
```

Újradeklarálásnál kapcsolt típus is lehet kovariáns (1)

```
deferred class COMPLEX
feature
    times alias "*"( other: COMPLEX ): like Current
        deferred
        end
...

class POLAR_COMPLEX inherit COMPLEX
feature
    times alias "*" ( other: COMPLEX ): like Current
        do
            ...
        end
...
```

Újradeklarálásnál kapcsolt típus is lehet kovariáns (2)

```
class COMPLEX
feature
    times alias "*"( other: COMPLEX ): like Current
        do ...  end
...

class POLAR_COMPLEX
inherit
    COMPLEX redefine times end
feature
    times alias "*" ( other: COMPLEX ): like Current
        do ...  end
...
```

Altípusosság függvénytípusokon

LSP: ha $\mathsf{Int} <: \mathsf{Real}$, akkor egy $\mathsf{Int} \to \mathsf{Real}$ függvény helyett használhatunk egy $\mathsf{Real} \to \mathsf{Int}$ függvényt.

Altípusosság függvénytípusokon

LSP: ha Int $<:$ Real, akkor egy Int $\to$ Real függvény helyett használhatunk egy Real $\to$ Int függvényt.

$$\frac{\mathrm{Int} <: \mathrm{Real}}{\mathrm{Real} \to \mathrm{Int} \ <: \ \mathrm{Int} \to \mathrm{Real}}$$

Altípusosság függvénytípusokon

LSP: ha Int $<:$ Real, akkor egy Int $\to$ Real függvény helyett használhatunk egy Real $\to$ Int függvényt.

$$\frac{\mathrm{Int} <: \mathrm{Real}}{\mathrm{Real} \to \mathrm{Int} \;\; <: \;\; \mathrm{Int} \to \mathrm{Real}}$$

$$\frac{A <: A', \quad B' <: B}{A' \to B' \;\; <: \;\; A \to B}$$

Altípusosság függvénytípusokon

LSP: ha Int $<:$ Real, akkor egy Int $\to$ Real függvény helyett használhatunk egy Real $\to$ Int függvényt.

$$\frac{\mathrm{Int} <: \mathrm{Real}}{\mathrm{Real} \to \mathrm{Int} \;\; <: \;\; \mathrm{Int} \to \mathrm{Real}}$$

$$\frac{A <: A', \quad B' <: B}{A' \to B' \;\; <: \;\; A \to B}$$

- Kontravariáns paramétertípus
- Kovariáns visszatérési típus

Altípusosság függvénytípusokon

LSP: ha Int $<:$ Real, akkor egy Int $\to$ Real függvény helyett használhatunk egy Real $\to$ Int függvényt.

$$\frac{\text{Int} <: \text{Real}}{\text{Real} \to \text{Int} \;\; <: \;\; \text{Int} \to \text{Real}}$$

$$\frac{A <: A', \quad B' <: B}{A' \to B' \;\; <: \;\; A \to B}$$

- Kontravariáns paramétertípus
- Kovariáns visszatérési típus

Definiálja felül r' a megörökölt r rutint. Ekkor:

$$\text{PRE}(r) \;\Rightarrow\; \text{PRE}(r) \vee \texttt{require\_else}_{r'} \;=:\; \text{PRE}(r')$$

$$\text{POST}(r') \;:=\; \text{POST}(r) \wedge \texttt{ensure\_then}_{r'} \;\Rightarrow\; \text{POST}(r)$$

Kontravariáns paraméter?

- Legyen $A <: B$, és legyen T-ben egy r rutin A típusú paraméterrel. Az $S <: T$ akkor is fennáll, ha S újradeklarálja r-et B paramétertípussal.
- Ellentétes irányú változás

```
class T                  class S inherit T redefine r end
feature r(p:A)           feature r(p:B)
            ...                      ...
```

- LSP: `object_T.r(param_A)` helyett `object_S.r(param_B)`

Kontravariáns paraméter?

- Legyen $A <: B$, és legyen T-ben egy r rutin A típusú paraméterrel. Az $S <: T$ akkor is fennáll, ha S újradeklarálja r-et B paramétertípussal.
- Ellentétes irányú változás

```
class T                    class S inherit T redefine r end
feature r(p:A)             feature r(p:B)
            ...                        ...
```

- LSP: `object_T.r(param_A)` helyett `object_S.r(param_B)`
- Ilyet nem szoktak támogatni a programozási nyelvek
  - C++, Java...: nonvariancia a paraméter típusában (változás hatása: elfedés, túlterhelés)
  - Eiffel: típus versus szerződés?

Eiffel: kontravariáns előfeltétel (lényegében *paraméter*)

```
class BANK_CARD
feature
    withdraw( amount: INTEGER )
        require no_overdraw: amount <= balance
        ensure balance = old balance - amount
...
end


class CREDIT_CARD
inherit
    BANK_CARD redefine withdraw end
feature
    withdraw( amount: INTEGER )
        require else True   -- `balance` can go negative
...
end
```

Az Eiffel megengedi a paramétertípusok *kovarianciáját*

- Matematikai nonszensz
- LSP-nek ellentmond
- De a gyakorlatban hasznos lehet!

```
deferred class ANIMAL
feature feed( f: attached FOOD ) deferred end
...

class CAT
inherit ANIMAL
feature feed( m: attached MILK ) do ... end
...
```

Miért jó?

```
deferred class FOOD end
class MILK inherit FOOD end
class GRASS inherit FOOD end

deferred class ANIMAL
feature feed( f: attached FOOD ) deferred end
...

class CAT
inherit ANIMAL
feature feed( m: attached MILK ) do ... end
...

(create {CAT}).feed( (create {GRASS}) ) -- fordítási hiba!
```

Mi a baj?

```
local
    a_cat: CAT
    some_grass: GRASS
    polymorphic: ANIMAL
do
    create a_cat
    create some_grass
    a_cat.feed( some_grass ) -- fordítási hiba!
```

Mi a baj?

```
local
    a_cat: CAT
    some_grass: GRASS
    polymorphic: ANIMAL
do
    create a_cat
    create some_grass
    a_cat.feed( some_grass ) -- fordítási hiba!

    polymorphic := cat
    polymorphic.feed( some_grass )
```

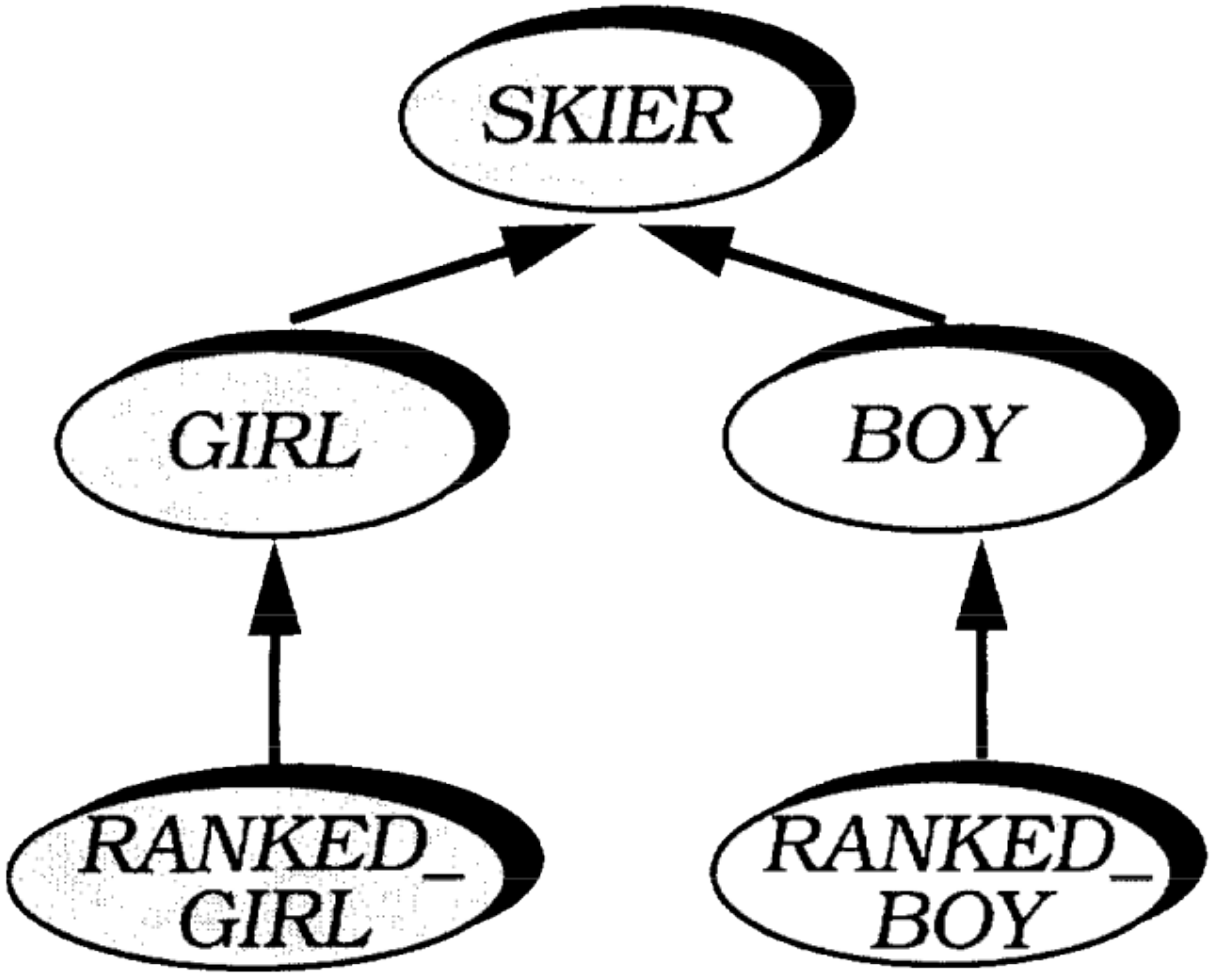
Mi a baj?

```
local
    a_cat: CAT
    some_grass: GRASS
    polymorphic: ANIMAL
do
    create a_cat
    create some_grass
    a_cat.feed( some_grass ) -- fordítási hiba!

    polymorphic := cat
    polymorphic.feed( some_grass )
```

Polymorphic CAT-call

B Meyer: Static Typing (OOPSLA 1995)



B Meyer: Static Typing (OOPSLA 1995) – modernizálva

```
class SKIER
feature
    roommate: detachable SKIER assign share
    share( s: detachable SKIER ) do roommate := s end
    ...
end
```

B Meyer: Static Typing (OOPSLA 1995) – modernizálva

```
class SKIER
feature
    roommate: detachable SKIER assign share
    share( s: detachable SKIER ) do roommate := s end
    ...
end


class GIRL
inherit SKIER redefine roommate, share end
feature
    roommate: detachable GIRL assign share
    share( g: detachable GIRL ) do roommate := g end
    ...
end
```

kovariáns paraméter, kovariáns mező

Adaptív paramétertípus

```
class SKIER
feature
    roommate: detachable SKIER assign share
    share( s: like roommate ) do roommate := s end
    ...
end

class GIRL
inherit SKIER redefine roommate end
feature
    roommate: detachable GIRL assign share
    ...
end
```

Adaptív mezőtípus

```
class SKIER
feature
    roommate: detachable like Current assign share
    share( s: like roommate ) do roommate := s end
    ...
end

class GIRL
inherit SKIER
end
```

Adaptív mezőtípus

```
class SKIER
feature
    roommate: detachable like Current assign share
    share( s: like roommate ) do roommate := s end
    ...
end


class GIRL
inherit SKIER
end


g: GIRL
b: BOY
...
g.share(b)  -- fordítási hiba
```

B Meyer: Static Typing (OOPSLA 1995)

"Strangely enough, some workers in the field have been advocating a contravariant policy."

```
class GIRL
inherit SKIER
feature
    share( g: GIRL ) ...
end

class RANKED_GIRL
inherit GIRL redefine share end
feature
    share( g: SKIER ) ...
end
```

"Here is what, under various mathematical excuses, some professors have been promoting. No wonder teenage pregnancies are on the rise."

Az Eiffel egy „kovariáns nyelv''

- Megengedi a paramétertípusok kovarianciáját
  - Rutinok paramétereire
  - Generikus típusok paramétereire
- A szerződésekre viszont kontravariáns előfeltétel (kontravariáns megkötés a paraméterre)

A statikus típusrendszer nem elég erős

```
class SKIER
feature
    roommate: detachable like Current assign share
    share( s: like roommate ) do roommate := s end
end

class GIRL inherit SKIER end
```

A statikus típusrendszer nem elég erős

```
class SKIER
feature
    roommate: detachable like Current assign share
    share( s: like roommate ) do roommate := s end
end

class GIRL inherit SKIER end

g: SKIER
b: BOY
...
create {GIRL} g
g.share(b)  -- nincs fordítási hiba
```

Ha be van kapcsolva a “catcall”-ellenőrzés

```
class SKIER
feature
    roommate: detachable like Current assign share
    share( s: like roommate ) do roommate := s end
end


class GIRL inherit SKIER end


g: SKIER
b: BOY
...
create {GIRL} g
g.share(b)  -- nincs fordítási hiba
```

```
Catcall detected in {SKIER}.share for arg#1:
                    expected GIRL but got BOY
```

CAT: Changed Availability or Type

Ha leszármaztatásnál

- örökölt feature-nek csökken a láthatósága, vagy
- mezőnek vagy rutin paraméterének szűkül a típusa.

```
class SKIER
feature
    roommate: detachable like Current assign share
    share( s: like roommate ) do roommate := s end
end

class GIRL
inherit
    SKIER
        export {NONE} roommate
    end
end
```

CAT-call detection

- Futás közben hibát/jelzést kaphatunk
- Ha a statikus típusrendszert kicseleztük

Hibásan működő program (1)

```
deferred class ANIMAL
feature
    feed( f: FOOD ) deferred end
end

deferred class FOOD end

class GRASS inherit FOOD

class MILK inherit FOOD
feature
    volume: INTEGER
    add( amount: INTEGER )
        do volume := volume + amount end
end
```

Hibásan működő program (2)

```
class CAT
inherit ANIMAL
feature
    stomach: attached MILK attribute create Result end
    feed( some_milk: attached MILK )
        do
            stomach.add(some_milk.volume)
        end
end
```

```
c: ANIMAL
...
create {CAT} c
c.feed( create {GRASS} )
```

Hibásan működő program (3)

```
Catcall detected in {CAT}.feed for arg#1: expected MILK but got GRASS

main: system execution failed.
Following is the set of recorded exceptions:

******************************** Thread exception ********************************
In thread           Root thread            0x0 (thread id)
**********************************************************************************
--------------------------------------------------------------------------------
Class / Object      Routine                Nature of exception           Effect
--------------------------------------------------------------------------------
CAT                 feed @1                Segmentation fault:
<00007F70566C9598>                         Operating system signal.      Fail
--------------------------------------------------------------------------------
CAT                 feed @1
<00007F70566C9598>                         Routine failure.              Fail
--------------------------------------------------------------------------------
MAIN                make @2
<00007F70566C9558>                         Routine failure.              Fail
--------------------------------------------------------------------------------
MAIN                root's creation
<00007F70566C9558>                         Routine failure.              Exit
--------------------------------------------------------------------------------
```

Polimorf változó

Ha a (referencia típusú) változó mutathat alosztályba tartozó objektumra is.

Feltételek:

- A deklarált típusnak van alosztálya.
- Fordítási időben nem zárható ki, hogy a változó alosztályba tartozó értéket kap.

Ban polymorphic CAT-calls

- Statikus típusbiztonság
- A fordító ellenőrzi, hogy polimorf változón nem hivatkozunk CAT-tulajdonságú feature-t
- Ne legyen túl szigorú: teljes programra kiterjedő elemzés (whole program analysis)
  - Nagyon időigényes lehet
  - Szerkesztéskor végezhető el

Binary method

- Kétparaméteres művelet az osztályon
- Szimmetria versus altípusosság

```
class FRACTION
...
feature
    numerator, denominator: INTEGER
    ...
    plus alias "+" ( other: like Current ): like Current
        ...
end
```

Tiltsuk ki a reprezentációk keverését

```
class COMPLEX
feature
    times alias "*"( other: like Current ): like Current
...

class POLAR_COMPLEX inherit COMPLEX
feature
    times alias "*" ( other: like Current ): like Current
...

class CART_COMPLEX inherit COMPLEX
feature
    times alias "*" ( other: like Current ): like Current
...
```

Reprezentációk közötti konverzió explicit!

```
class POLAR_COMPLEX inherit COMPLEX
feature
    times alias "*" ( other: like Current ): like Current
...

pc: attached POLAR_COMPLEX
cc: attached CART_COMPLEX
...
pc := pc * cc          -- fordítási hiba
pc := pc * cc.polar    -- ok
cc := pc.cart * cc     -- ok
```