Programare declarativă

Introducere în Semantica limbajelor de programare

Bazat pe cursul omonim de la Universitatea Cambridge

Traian Florin Șerbănuță

Departamentul de Informatică, FMI, UNIBUC
traian.serbanuta@fmi.unibuc.ro

7 octombrie 2014

Anunț

În atenția semigrupei 3441

Doar astăzi aveți șansa să nu faceți cu mine laboratorul de la ora 14 (sala 218).

Alternative

- azi de la ora 14 la sala 310 (cu grupa 3421) cu dna Adela Georgescu; sau
- joi, de la ora 10 la sala 218 (cu grupa 3442) cu mine

Prin ce e definit un limbaj de programare?

Sintaxa Simboluri de operație, cuvinte cheie, descriere (formală) a programelor/expresiilor bine formate

Practica Un limbaj e definit de modul cum poate fi folosit

- Manual de utilizare și exemple de bune practici
- Implementare (compilator/interpretor)
- Instrumente ajutătoare (analizor de sintaxă, depanator)

Semantica? Ce înseamnă / care e comportamentul unei instrucțiuni?

- De cele mai multe ori se dă din umeri și se spune că Practica e suficientă

La ce folosește semantica

- Să înțelegem un limbaj în profunzime
  - Ca programator: pe ce mă pot baza când programez în limbajul dat
  - Ca implementator al limbajului: ce garanții trebuie să ofer
- Ca instrument în proiectarea unui nou limbaj / a unei extensii
  - Înțelegerea componentelor și a relațiilor dintre ele
  - Exprimarea (și motivarea) deciziilor de proiectare
  - Demonstrarea unor proprietăți generice ale limbajului
    E.g., execuția nu se va bloca pentru programe care trec de analiza tipurilor
- Ca bază pentru demonstrarea corectitudinii programelor.

C

C

```
int main(void) {
  int x = 3;
  return x++ + x++ + x++ + x++;
}
```

C

```
int main(void) {
  int x = 0;
  return (x = 1) + (x = 2);
}
```

C

```
int main(void) {
  int x = 0;
  return (x = 1) + (x = 2);
}
```

Conform standardului C, comportamentul programului este nedefinit.

- GCC4, MSVC: valoarea întoarsă e 4
- GCC3, ICC, Clang: valoarea întoarsă e 3

C

```
int r;
int f(int x) {
  return (r = x);
}
int main() {
  return f(1) + f(2), r;
}
```

C

```
int r;
int f(int x) {
  return (r = x);
}
int main() {
  return f(1) + f(2), r;
}
```

Conform standardului C, comportamentul programului este subspecificat: poate întoarce atât valoarea 1 cât și 2.

C#

C#

```
delegate int IntThunk();
class M {
    public static void Main() {
        IntThunk[] funcs = new IntThunk[11];
        for (int i = 0; i <= 10; i++) {
            funcs[i] = delegate() { return i; };
        }
        foreach (IntThunk f in funcs) {
            System.Console.WriteLine(f());
        }
    }
}
```

JavaScript

JavaScript

```
function bar(x) {
  return function() { var x = 5; return x; };
}
var f = bar(200);
f()
```

JavaScript

JavaScript

```
function bar(x) {
  return function() { var x = x; return x; };
}
var f = bar(200);
f()
```

Java

Java

```
class Main {
    interface F<A, B> {           B a(A x);          }
    static <A, B, C> F<A, C> c(final F<A, B> f, final F<B, C> g) {
        return new F<A, C>() {
            public C a(A x) {  return g.a(f.a(x)); }
        };
    }
    public static void main(String[] args) {
        final Integer a = 2, b = 1;
        F<Integer, Integer> f = new F<Integer, Integer>() {
            public Integer a(Integer x) { return x + b; } };
        F<Integer, Integer> g = new F<Integer, Integer>() {
            public Integer a(Integer x) { return a * x; } };
        F<Integer, Integer> h = c( f, g );
        System.out.println(h(5));
    }
}
```

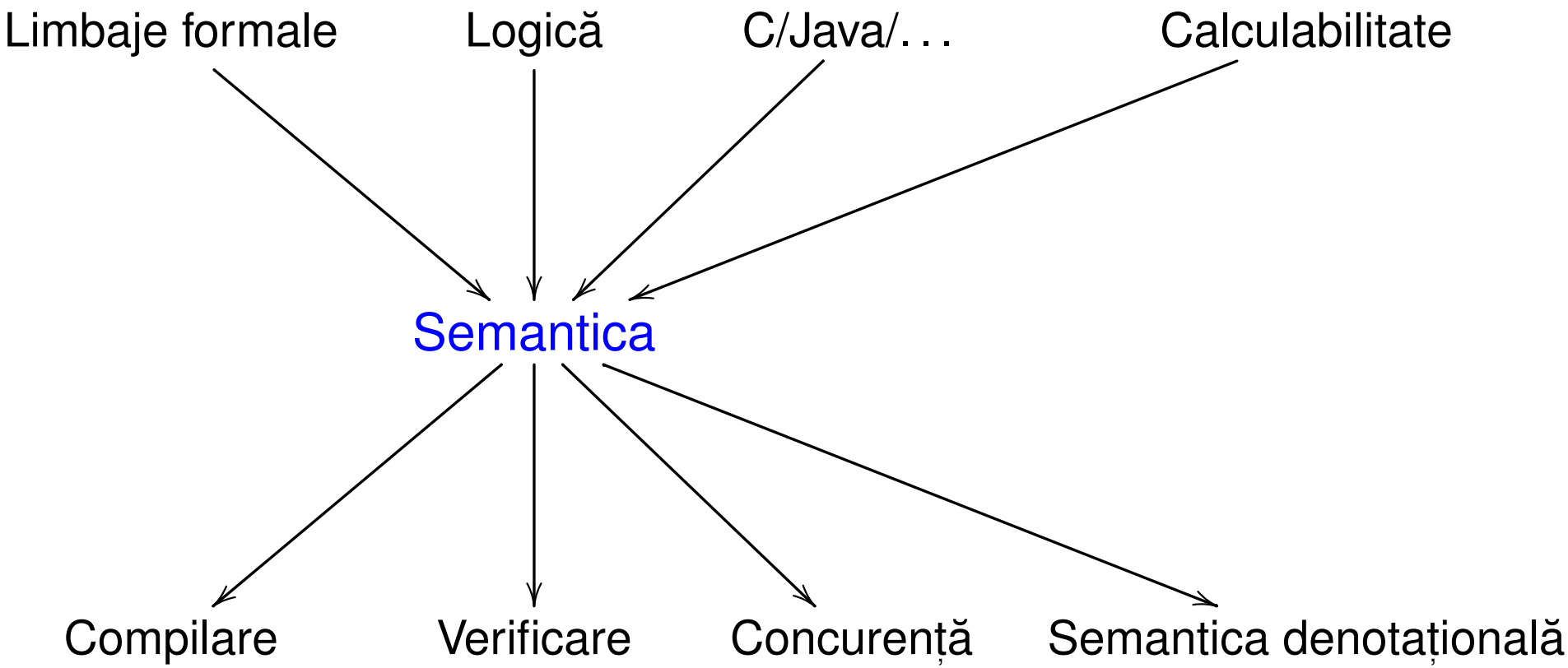
Descriere curs

- Definirea unui mini-limbaj de programare
  - Cu extensii, atât declarative, cât și imperative
  - Definirea sistemelor de tipuri
- Elemente de proiectarea limbajelor de programare
  - Tipuri, Funcții, Date structurate, Referințe
  - Subtipuri, Obiecte, Interacție și Concurență
- Proprietăți ale limbajelor de programare și ale programelor lor
  - Instrumente (simple) matematice pentru studiul programelor
  - Corectitudinea programelor în raport cu un sistem de tipuri
  - Echivalența semantică dintre programe
- La laborator
  - Elemente de programare funcțională folosind limbajul OCaml
  - Interpretoare și instrumente de analiză a programelor

Obiective

- Formalizarea și înțelegerea conceptelor de bază în proiectarea limbajelor de programare
  - Sintaxă, semantica execuției, semantica tipurilor
- Folosirea programării declarative pentru transpunerea definițiilor formale în practică
  - interpretoare, verificatoare de tipuri, . . .
- Deprinderea paradigmei de programare funcțională
  - declarativă, concisă, apropiată de limbajul matematic

Privire de ansamblu



Evaluare

- Examen scris: 70%
- Notă laborator: 30%
  - 2 teste (≈ săptămâna a 6-a și săptămâna a 12-a)
  - al doilea test de laborator poate fi înlocuit de un proiect
  - Nota = media artimetică a testelor
  - Minim 5 pentru a intra la examenul scris
- Bonus (max 1p) pentru activitate
  - La curs, la laborator, sau pe Moodle
  - Întrebări interesante, comentarii constructive, etc.
  - Doar daca aveti deja nota de promovare
- Activați-vă (dacă e cazul) contul Moodle
  - Puteți pune întrebări și răspunde la întrebări
  - Notele vor fi postate pe pagina Moodle a cursului

Examenul scris

- Va consta din probleme
- Scopul lui e să verifice fixarea cunoștințelor predate
- Cu acces la materiale tipărite/xeroxate (nu scrise de mână)

Precizare

Pentru promovare, nota de la examenul scris trebue sa fie minim 5.

Proiect—Opțional

Pentru cei care se plictisesc la laborator

- Stabilirea temei: înainte de primul test
- Grad de complexitate mai mare decât testul de laborator propriu-zis
- Prezentarea progresului la laborator (i.e., nu faceți proiectul în ultima săptămână)
- Termen de predare: înainte de ultimul curs

Resurse

- Pagina Moodle a cursului:
  http://moodle.fmi.unibuc.ro/course/view.php?id=449
  - Prezentările cursurilor, forumuri, resurse electronice
- Pagina cursului similar de la Cambridge
  http://www.cl.cam.ac.uk/teaching/1415/Semantics
  - Note de curs, programe în (Moscow) ML și Java
- Pagina OCaml http://ocaml.org

Programare declarativă vs. imperativă

Ce vs. cum

Programare imperativă (Cum)

Explic mașinii, pas cu pas, algoritmic, cum să facă ceva și ca rezultat, se întâmplă ce voiam să se întâmple.

Programare declarativă (Ce)

Îi spun mașinii ce vreau să se întâmple și o las pe ea să se prindă cum să realizeze acest lucru. :-)

Operații iterative pe colecții (JS)

http://latentflip.com/imperative-vs-declarative/

Imperativ

```
var numbers = [1,2,3,4,5]
var doubled = []
for(var i = 0; i < numbers.length; i++) {
  var newNumber = numbers[i] * 2
  doubled.push(newNumber)
}
console.log(doubled)
```

Operații iterative pe colecții (JS)

http://latentflip.com/imperative-vs-declarative/

Imperativ

```
var numbers = [1,2,3,4,5]
var doubled = []
for(var i = 0; i < numbers.length; i++) {
  var newNumber = numbers[i] * 2
  doubled.push(newNumber)
}
console.log(doubled)
```

Declarativ

```
var numbers = [1,2,3,4,5]
var doubled = numbers.map(function(n) {
  return n * 2
})
console.log(doubled)
```

Agregarea datelor dintr-o colecție (JS)

http://latentflip.com/imperative-vs-declarative/

Imperativ

```
var numbers = [1,2,3,4,5]
var total = 0

for(var i = 0; i < numbers.length; i++) {
  total += numbers[i]
}
console.log(total)
```

Agregarea datelor dintr-o colecție (JS)

http://latentflip.com/imperative-vs-declarative/

Imperativ

```
var numbers = [1,2,3,4,5]
var total = 0

for(var i = 0; i < numbers.length; i++) {
  total += numbers[i]
}
console.log(total)
```

Declarativ

```
var numbers = [1,2,3,4,5]

var total = numbers.reduce(function(sum, n) {
  return sum + n
});
console.log(total)
```

Extragerea informației din tabele asociate

http://latentflip.com/imperative-vs-declarative/

Imperativ

```
var dogsWithOwners = []
for(var di=0; di < dogs.length; di++) {
  dog = dogs[di]
  for(var oi=0; oi < owners.length; oi++) {
    owner = owners[oi]
    if (owner && dog.owner_id == owner.id) {
      dogsWithOwners.push({  dog: dog,   owner: owner      })
    }
  }}
}
```

Extragerea informației din tabele asociate

http://latentflip.com/imperative-vs-declarative/

Imperativ

```
var dogsWithOwners = []
for(var di=0; di < dogs.length; di++) {
  dog = dogs[di]
  for(var oi=0; oi < owners.length; oi++) {
    owner = owners[oi]
    if (owner && dog.owner_id == owner.id) {
      dogsWithOwners.push({  dog: dog,   owner: owner      })
    }
  }}
}
```

Declarativ

```
SELECT * from dogs
INNER JOIN owners
WHERE dogs.owner_id = owners.id
```

Programare imperativă vs. declarativă

Diferențe

- Modelul de computație: algoritm vs. relație
- Ce exprimă un program: cum vs. ce
- Variabile/parametrii: atribuire distructivă vs. non-distructivă
- Structuri de date: alterabile vs. explicite
- Ordinea de execuție: efecte laterale vs. neimportantă
- Expresii ca valori: nu vs. da
- Controlul execuției: responsabilitatea programatorului vs a mașinii