Curs 1

2021-2022

Fundamentele Limbajelor de Programare

Cuprins

Organizare

Instructori

Curs

- **Ioana Leutean (seriile 23, 25)**
- **Denisa Diaconescu (seria 24)**

Laborator

Seria 23
- **Ana Iova (urlea) (231)**
- **Horatiu Cheval (232)**
- **Bogdan Macovei (233,234)**

Seria 24
- **Natalia Ozunu (241)**
- **Bogdan Macovei (242,243,244)**

Seria 25
- **Ana Iova (urlea) (251)**
- **Bogdan Macovei (252)**

Resurse

- Seriile 23, 25
  - Moodle
  - Pagina externa
- Seria 24
  - Moodle
  - Pagina externa
- Suporturile de curs si laborator/seminar, resurse electronice
- Stiri legate de curs vor fi postate pe Moodle si pe paginile externe

Prezenta

Prezenta la curs sau la laboratoare/seminarii nu este obligatorie, dar extrem de incurajata.

Notare

Notare

- Nota finala: 1 punct (oficiu) + examen
- Conditie minima pentru promovare: nota finala > 4.99
- Puncte bonus
  - La sugestia profesorului coordonator al laboratorului/seminarului, se poate nota activitatea în plus fata de cerintele obisnuite
  - Maxim 1 punct

Examen

- In sesiunea
- Durata 2 ore
- Cu materialele ajutatoare
- Mai multe detalii vor fi oferite pana la jumatatea semestrului

Curs/laborator

- Curs

  Bazele programrii logice
  - Logica clauzelor Horn, Unificare, Rezolutie

  Semantica limbajelor de programare
  - Semantic operaional, static i axiomatic
  - Inferarea automat a tipurilor

  Bazele programrii funcionale
  - Lambda Calcul, Codificri Church, corespondenta Curry-Howard
  - Lambda Calcul cu tipuri de date

- Laborator/Seminar:

  Prolog Cel mai cunoscut limbaj de programare logica
  - Verificator pentru un mini-limbaj imperativ
  - Inferena tipurilor pentru un mini-limbaj funcional

  Haskell Limbaj pur de programare funcional
  - Interpretoare pentru mini-limbaje

  Exercitii suport pentru curs

Bibliografie

- B.C. Pierce, **Types and programming languages**. MIT Press.2002
- G. Winskel, **The formal semantics of programming languages**. MIT Press. 1993
- H. Barendregt, E. Barendsen, **Introduction to Lambda Calculus**, 2000.
- J. Lloyd. **Foundations of Logic Programming**, second edition. Springer, 1987.
- P. Blackburn, J. Bos, and K. Striegnitz, **Learn Prolog Now!** (Texts in Computing, Vol. 7),College Publications, 2006
- M. Huth, M. Ryan, **Logic in Computer Science (Modelling and Reasoning about Systems)**, Cambridge University Press, 2004.

Privire de ansamblu

Bazele programrii funcionale / logice

Principalele paradigme de programare

- Imperativ (cum calculm)
  - Procedural
  - Orientat pe obiecte
- Declarativ (ce calculm)
  - Logic
  - Functional

Fundamentele paradigmelor de programare

Imperativ	Execuia unei Maini Turing
Logic	Rezoluia în logica clauzelor Horn
Funcional	Beta-reducie în Lambda Calcul

Programare declarativ

- Programatorul spune ce vrea s calculeze, dar nu specific concret cum calculeaz.
- Este treaba interpretorului (compilator/implementare) s identifice cum s efectueze calculul respectiv.
- Tipuri de programare declarativ:
  - Programare functional (e.g., Haskell)
  - Programare logic (e.g., Prolog)
  - Limbaje de interogare (e.g., SQL)

Programare funcional

Esen: funcii care relaioneaz intrrile cu ieirile

Caracteristici:
- funcii de ordin înalt – funcii parametrizate de funcii
- grad înalt de abstractizare (e.g., functori, monade)
- grad înalt de reutilizarea codului — polimorfism

Fundamente:
- Teoria funciilor recursive
- Lambda-calcul ca model de computabilitate (echivalent cu maina Turing)

Inspiraie:
- Inferena tipurilor pentru templates/generics in POO
- Model pentru programarea distribuit/bazat pe evenimente (callbacks)

Programare logic

- Programarea logic este o paradigm de programare bazat pe logic formal.
- Unul din sloganurile programrii logice:

  Program = Logic + Control (R. Kowalski)

- Programarea logic poate fi privit ca o deductie controlat.
- Un program scris intr-un limbaj de programare logic este

  o list de formule intr-o logic

  ce exprim fapte i reguli despre o problem.
- Exemple de limbaje de programare logic:
  - Prolog
  - Answer set programming (ASP)
  - Datalog

Semantica Limbajelor de Programare

Ce definete un limbaj de programare?

Sintaxa Simboluri de operaie, cuvinte cheie, descriere (formal) a programelor/expresiilor bine formate

Practica Un limbaj e definit de modul cum poate fi folosit

- Manual de utilizare i exemple de bune practici
- Implementare (compilator/interpretor)
- Instrumente ajuttoare (analizor de sintax, verificator de tipuri, depanator)

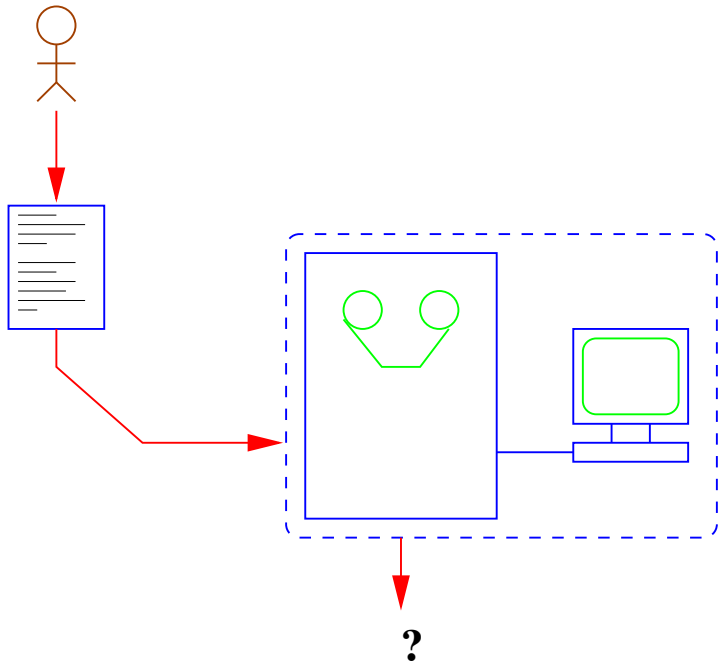
Semantica? Ce înseamn / care e comportamentul unei instruciuni?

- De cele mai multe ori se d din umeri i se spune c Practica e suficient
- Limbajele mai utilizate sunt standardizate

La ce folosete semantica

- S înelegem un limbaj în profunzime
  - Ca programator: pe ce m pot baza când programez în limbajul dat
  - Ca implementator al limbajului: ce garanii trebuie s ofer
- Ca instrument în proiectarea unui nou limbaj / a unei extensii
  - Înelegerea componentelor i a relaiilor dintre ele
  - Exprimarea (i motivarea) deciziilor de proiectare
  - Demonstrarea unor proprieti generice ale limbajului
    E.g., execuia nu se va bloca pentru programe care trec de analiza tipurilor
- Ca baz pentru demonstrarea corectitudinii programelor.

Problema corectitudinii programelor



- Pentru anumite metode de programare (e.g., imperativ, orientat pe obiecte), nu este uor s stabilim c un program este corect sau s înelegem ce înseamn c este corect (e.g, în raport cu ce?!).
- Corectitudinea programelor devine o problem din ce în ce mai important, nu doar pentru aplicaii "safety-critical".
- Avem nevoie de metode ce asigur "calitate", capabile s ofere "garanii".

C

```
#include <iostream>
using namespace std;
int main()
{
  int square;
  for(int i = 1; i <= 5; ++i)
  {
    square = i * i;
    cout <<  square << endl;
  }
}
```

- Este corect? În raport cu ce?
- Un formalism adecvat trebuie:
  - s permit descrierea problemelor (specificaii), i
  - s raioneze despre implementarea lor (corectitudinea programelor).

Care este comportamentul corect?

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

Conform standardului C, comportamentul programului este nedefinit.

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

Care este comportamentul corect?

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

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

Tipuri de semantic

Semantica d "îneles" unui program.

- Operaional:
  - Înelesul programului este definit în funcie de paii (transformri dintr-o stare în alta) care apar în timpul execuiei.
- Denotaional:
  - Înelesul programului este definit abstract ca element dintr-o structur matematic adecvat.
- Axiomatic:
  - Înelesul programului este definit indirect în funcie de axiomele i regulile pe care le verific.
- Static / a tipurilor
  - Reguli de bun-formare pentru programe
  - Ofer garanii privind execuia (e.g., nu se blocheaz)

Programare logic & Prolog

Programare logic - în mod idealist

- Un "program logic" este o colecie de proprieti presupuse (sub form de formule logice) despre lume (sau mai degrab despre lumea programului).
- Programatorul furnizeaz i o proprietate (o formula logic) care poate s fie sau nu adevrat în lumea respectiv (întrebare, query).
- Sistemul determin dac proprietatea aflat sub semnul întrebrii este o consecin a proprietilor presupuse în program.
- Programatorul nu specific metoda prin care sistemul verific dac întrebarea este sau nu consecin a programului.

Exemplu de program logic

$$
\begin{aligned}
\texttt{oslo} &\rightarrow \texttt{windy} \\
\texttt{oslo} &\rightarrow \texttt{norway} \\
\texttt{norway} &\rightarrow \texttt{cold} \\
\texttt{cold} \wedge \texttt{windy} &\rightarrow \texttt{winterIsComing} \\
&\quad\ \texttt{oslo}
\end{aligned}
$$

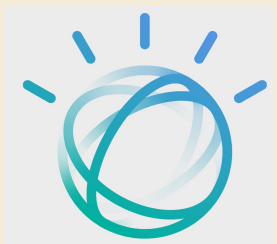
Exemplu de întrebare

Este adevrat `winterIsComing`?

Prolog

- bazat pe logica clauzelor Horn
- semantica operaional este bazat pe rezoluie
- este Turing complet

Limbajul Prolog este folosit pentru programarea sistemului IBM Watson!



Putei citi mai multe detalii aici.

Exemplul de mai sus în SWI-Prolog

Program:

```
windy :- oslo.
norway :- oslo.
cold :- norway.
winterIsComing :- windy, cold.
oslo.
```

Intrebare:

```
?- winterIsComing.
true
```

http://swish.swi-prolog.org/

Quiz time!

https://www.questionpro.com/t/AT4NiZrHFn

Sintax: constante, variabile, termeni compui

- Atomi: `brian, 'Brian Griffin', brian_griffin`
- Numere: `23, 23.03,-1`

  Atomii i numerele sunt constante.
- Variabile: `X, Griffin, _family`
- Termeni compui: `father(peter, stewie_griffin)`,
  `and(son(stewie,peter), daughter(meg,peter))`
  - forma general: atom(termen,. . ., termen)
  - atom-ul care denumete termenul se numete functor
  - numrul de argumente se numete aritate

Un mic exerciiu sintactic

Care din urmtoarele iruri de caractere sunt constante i care sunt variabile în Prolog?

- vINCENT – constant
- Footmassage – variabil
- variable23 – constant
- Variable2000 – variabil
- big_kahuna_burger – constant
- 'big kahuna burger' – constant
- big kahuna burger – nici una, nici alta
- 'Jules' – constant
- _Jules – variabil
- '_Jules' – constant

Program în Prolog = baz de cunotine

Exemplu

Un program în Prolog:

```
father(peter,meg).
father(peter,stewie).

mother(lois,meg).
mother(lois,stewie).

griffin(peter).
griffin(lois).

griffin(X) :- father(Y,X),  griffin(Y).
```

Un program în Prolog este o baz de cunotine (Knowledge Base).

Program în Prolog = mulime de predicate

Practic, gândim un program în Prolog ca o mulime de predicate cu ajutorul crora descriem *lumea* (*universul*) programului respectiv.

Exemplu

```
father(peter,meg).
father(peter,stewie).

mother(lois,meg).
mother(lois,stewie).

griffin(peter).
griffin(lois).

griffin(X) :- father(Y,X),  griffin(Y).
```

Predicate:
father/2
mother/2
griffin/1

Un program în Prolog

Program

- Un program în Prolog este format din reguli de forma

```
Head :- Body.
```

- Head este un predicat, iar Body este o secven de predicate separate prin virgul.
- Regulile fr `Body` se numesc fapte.

Exemplu

- Exemplu de regul: `griffin(X) :- father(Y,X), griffin(Y).`
- Exemplu de fapt: `father(peter,meg).`

Interpretarea din punctul de vedere al logicii

- operatorul `:-` este implicaia logic $\leftarrow$

Exemplu

```
comedy(X) :- griffin(X)
```

dac `griffin(X)` este adevrat, atunci `comedy(X)` este adevrat.

- virgula `,` este conjuncia $\wedge$

Exemplu

```
griffin(X) :- father(Y,X), griffin(Y).
```

dac `father(Y,X)` i `griffin(Y)` sunt adevrate,
atunci `griffin(X)` este adevrat.

Interpretarea din punctul de vedere al logicii

- mai multe reguli cu acelai `Head` definesc acelai predicat, între defiii fiind un sau logic.

Exemplu

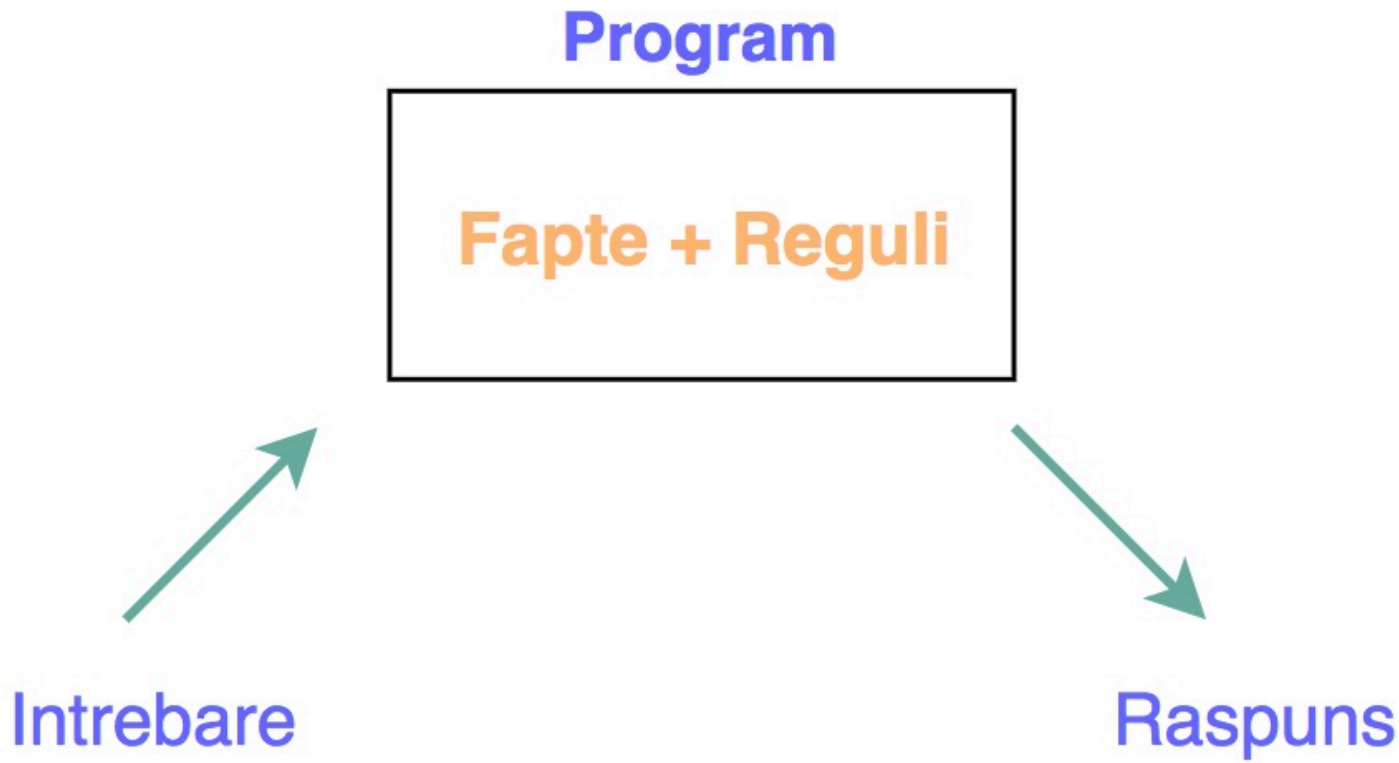
```
comedy(X) :- family_guy(X).
comedy(X) :- south_park(X).
comedy(X) :- disenchantment(X).
```

dac
`family_guy(X)` este adevrat sau `south_park(X)` este adevrat sau `disenchantment(X)` este adevrat,
atunci
`comedy(X)` este adevrat.

Un program în Prolog

Cum folosim un program în Prolog?

Întrebri în Prolog



Întrebri i inte în Prolog

- Prolog poate rspunde la întrebri legate de consecinele relaiilor descrise într-un program în Prolog.
- Întrebrile sunt de forma:

  $$\texttt{?- predicat}_1\texttt{(...),...,predicat}_n\texttt{(...).}$$

- Prolog verific dac întrebarea este o consecin a relaiilor definite în program.
- Dac este cazul, Prolog caut valori pentru variabilele care apar în întrebare astfel încât întrebarea s fie o consecin a relaiilor din program.
- Un predicat care este analizat pentru a se rspunde la o întrebare se numete int (goal).

Întrebri în Prolog

Prolog poate da 2 tipuri de rspunsuri:

- `false` – în cazul în care întrebarea nu este o consecin a programului.
- `true` sau valori pentru variabilele din întrebare în cazul în care întrebarea este o consecin a programului.

Exemplu

```
?- griffin(meg)
true
?- griffin(glenn)
false
```

```
?- griffin(X)
X = petr ;
X = lois ;
X = meg ;
X = stewie ;
false
```

Pe sptmâna viitoare!