Programare Logică

2013-2014

FMI

Cuprins

Organizare

Instructori

Denisa Diaconescu - curs și laboratoare

- Studii:
  - Licența Informatică:
    "Logica matematică și aplicații în verificarea sistemelor"
    - 2007, FMI
    - prof. Gheorghe Ștefănescu
  - Master Informatică:
    "Teoria modelelor pentru logici cu mai multe valori"
    - 2009, Școala Normală Superioară București, IMAR
    - prof. Răzvan Diaconescu
  - Doctorat Matematică:
    "Logici multivalente cu conjuncții necomutative"
    - 2012, FMI
    - prof. George Georgescu
- Poziții:
  - 2013 - prezent: Lector Universitar, FMI
  - 2008 - 2013: Preparator Universitar, FMI

Denisa Diaconescu - curs și laboratoare

- Domenii de cercetare:
  - logici neclasice (logici multivalente, logici modale)
  - modelarea matematică a fenomenelor vagi și incerte
  - aplicații ale logicii în verificarea sistemelor
- Contact:
  - ddiaconescu@fmi.unibuc.ro
  - denisa.diaconescu@gmail.com

Andrei Sipoș - laboratoare

- Studii:
  - Licență Matematică: "Calcul Schubert"
    - 2012, FMI
    - prof. Mihai Sorin Stupariu
  - Licență Informatică: "Ultraproduse în teoria modelelor"
    - 2012, FMI
    - prof. George Georgescu
- Poziții:
  - 2012 - prezent: Student Master "Algebră", FMI
- Domenii de cercetare:
  - geometrie algebrică
  - logică categorială
- Contact:
  - andrei.sipos@my.fmi.unibuc.ro

Suport curs

Site-uri curs

- Moodle
- https://sites.google.com/site/ddiaconescupl/

Bibliografie

- J. Goguen, **Theorem Proving and Algebra**, manuscris.
- F. Baader, T. Nipkow, **Terms Rewriting and All That**, Cambridge University Press, 1998.
- F.L. Ţiplea, **Fundamentele algebrice ale informaticii**, (II40405, biblioteca FMI).
- V.E. Căzănescu, **Note de curs**.

Notare

Notare

- Laborator: 40 puncte
- Examen: 60 puncte

- Condiție minină pentru promovare: cel puțin 50% din fiecare probă
  - laborator: min. 20 puncte și
  - examen: min. 30 puncte

Laborator: 40 puncte

- Lucrare: 30 puncte
  - Are loc în Săptămâna 8 (7 - 11 aprilie)
  - Prezența la lucrare este obligatorie!
  - Nu se poate reface
  - Timp de lucru: o oră și jumătate
- Proiect: 10 puncte
  - Se distribuie în Săptămâna 9 (14 - 17 aprilie)
  - Se predă în Săptămâna 14 (26 - 30 mai)
- Din cele două probe de Laborator trebuie să adunați min. 20 puncte.

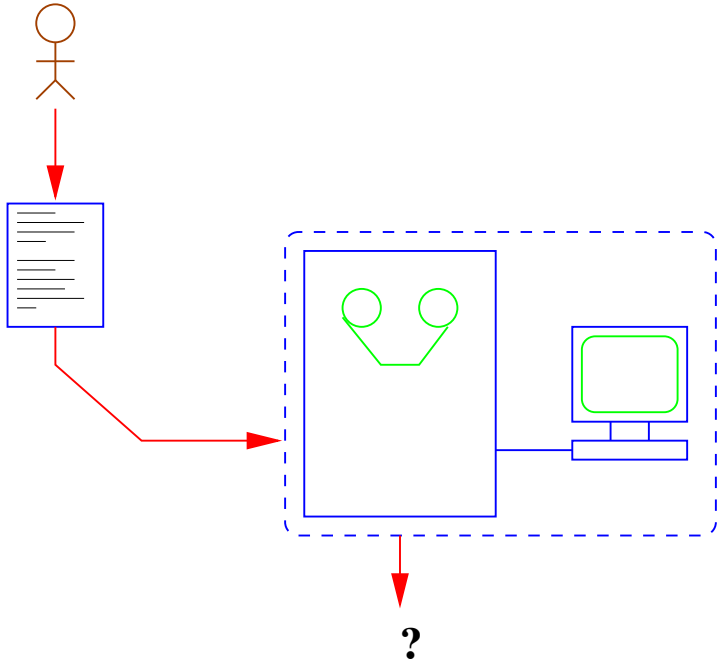
Examen: 60 puncte

- Subiecte de teorie și exerciții.
- Timp de lucru: 2 ore
- În Săptămâna 14 veți primi o foaie cu teorie cu care puteți veni la examen!
- Subiectele de teorie constau în demonstrarea unor rezultate din curs (demonstrate la curs sau lăsate ca temă).
- Subiectele de exerciții vor fi în stilul celor rezolvate la seminar (în Laboratoarele 10-13).
- La examen, trebuie să adunați min. 30 puncte.

Privire de ansamblu

Curs

Problema corectitudinii programelor



- Pentru metodele convenționale de programare (imperative), nu este ușor să vedem că un program este corect sau să înțelegem ce înseamnă că este corect (în raport cu ce?!).
- Devine o problemă din ce în ce mai importantă, nu doar pentru aplicații "safety-critical".
- Avem nevoie de metode ce asigură "calitate", capabile să ofere "garanții".

Un program imperativ simplu

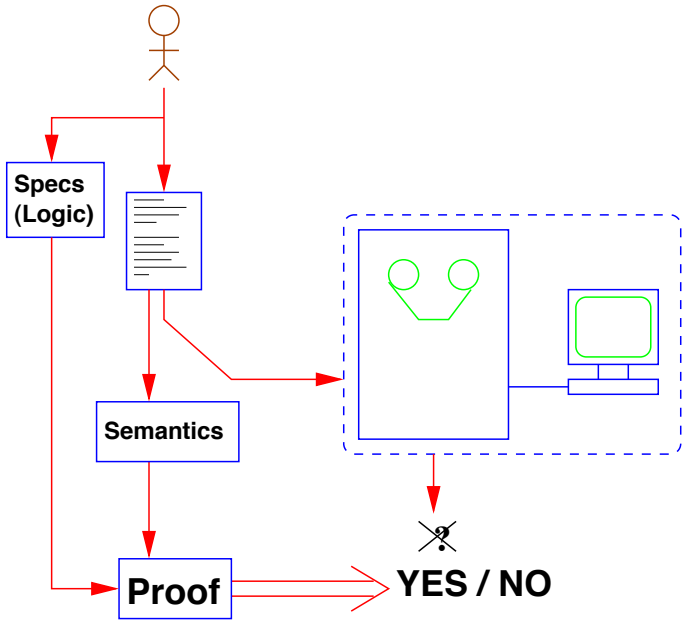
```
#include <stdio.h>
main() {
  int Number, Square;
  Number = 0;
    while(Number <= 5)
      { Square = Number * Number;
        printf("%d\n",Square);
        Number = Number + 1; } }
```

- Este corect? În raport cu ce?
- Un formalism adecvat trebuie:
  - să permită descrierea problemelor (specificații), și
  - să raționeze despre implementarea lor (corectitudinea programelor).

Logica

- Un mijloc de a clarifica/modela procesul de a "raționa".
- De exemplu, în logica clasica putem modela raționamentul:
  - Aristotel iubește prăjiturile, și
  - Platon este prieten cu oricine iubește prăjiturile, deci
  - Platon este prieten cu Aristotel.
- Simbolic:
  - $a_1 : iubeste(Aristotel, prajituri)$
  - $a_2 : (\forall X)\ iubeste(X, prajituri) \rightarrow prieten(Platon, X)$
  - $a_3 : prieten(Platon, Aristotel)$
  - $a_1, a_2 \vdash a_3$
- Cum poate fi folosită logica pentru:
  - a descrie probleme (specificații)?
  - a rezolva probleme?

Folosind logica



Logica ne permite să reprezentăm/modelăm probleme.

Pentru a scrie specificații și a raționa despre corectitudinea programelor:

- Limbaje de specificații (modelarea problemelor)
- Semantica programelor (operațională, denotațională, ...)
- Demonstrații (verificarea programelor, ...)

Pătratele numerelor naturale ≤ 5

Numerele naturale: reprezentarea lui Peano

$$0 \mapsto 0 \qquad 1 \mapsto s(0) \qquad 2 \mapsto s(s(0)) \qquad 3 \mapsto s(s(s(0))) \qquad \ldots$$

- Definirea numerelor naturale:
  $nat(0) \wedge nat(s(0)) \wedge nat(s(s(0))) \wedge \ldots$
- O soluție mai bună:
  $nat(0) \wedge (\forall X)(nat(X) \rightarrow nat(s(X)))$
- Ordinea pe numere naturale:
  $(\forall X) le(0, X) \wedge$
  $(\forall X, Y)(le(X, Y) \rightarrow le(s(X), s(Y)))$
- Adunarea numerelor naturale:
  $(\forall X)(nat(X) \rightarrow add(0, X, X)) \wedge$
  $(\forall X, Y, Z)(add(X, Y, Z) \rightarrow add(s(X), Y, s(Z)))$

Pătratele numerelor naturale ≤ 5

- Înmulțirea numerelor naturale:
  $(\forall X)(nat(X) \rightarrow mult(0, X, 0)) \wedge$
  $(\forall X, Y, Z, W)(mult(X, Y, W) \wedge add(W, Y, Z) \rightarrow$
  $mult(s(X), Y, Z))$

- Pătrate de numere naturale:
  $(\forall X, Y)(nat(X) \wedge nat(Y) \wedge mult(X, X, Y)) \rightarrow square(X, Y))$

Acum putem spune clar ce condiții vrem să satisfacă programul:

- Preconditie:
  niciuna

- Postconditie:
  $(\forall X)($
  $(\exists Y)nat(Y) \wedge le(Y, s(s(s(s(s(0)))))) \wedge square(Y, X)) \rightarrow$
  $output(X))$

Semantica

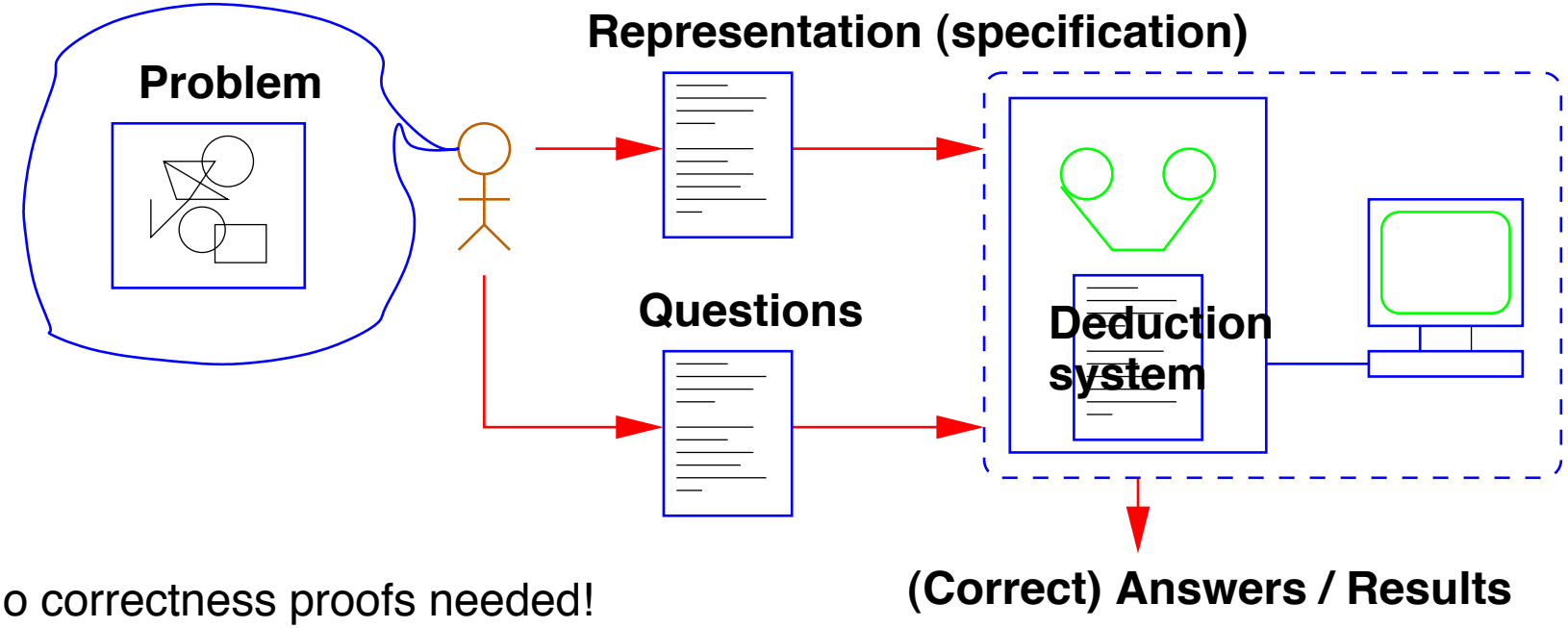
- Semantica dă un "înțeles" (obiect matematic) unui program.
- Semantica trebuie:
  - să poată verifica că un program satisface condițiile cerute.
  - să poată demonstra că două programe au aceeași semantica.
  - ...

Tipuri de Semantică

- Operațională:
  - Înțelesul programului este definit în funcție de pașii (transformări dintr-o stare în alta) care apar în timpul execuției.
- Axiomatică:
  - Înțelesul programului este definit indirect în funcție de axiomele și regulile unei logici.
- Denotațională:
  - Înțelesul programului este definit abstract ca element dintr-o structură matematică adecvată.
- Bazata pe modele:
  - Înțelesul programului este definit ca un model minimal al unei logici.

De la reprezentare/specificare la calcul

Presupunând existența unei metode (automate) de demonstrație (metodă de deducție), rezolvarea unor probleme se poate face astfel:



No correctness proofs needed!

Pătratele numerelor naturale ≤ 5

Query	Answer
$nat(s(0))$ **?**	(yes)
$\exists X\ add(s(0), s(s(0)), X)$ **?**	$X = s(s(s(0)))$
$\exists X\ add(s(0), X, s(s(s(0))))$ **?**	$X = s(s(0))$
$\exists X\ nat(X)$ **?**	$X = 0 \vee X = s(0) \vee X = s(s(0)) \vee \ldots$
$\exists X \exists Y\ add(X, Y, s(0))$ **?**	$(X = 0 \wedge Y = s(0)) \vee (X = s(0) \wedge Y = 0)$
$\exists X\ nat\_square(s(s(0)), X)$ **?**	$X = s(s(s(s(0))))$
$\exists X\ nat\_square(X, s(s(s(s(0))))))$ **?**	$X = s(s(0))$
$\exists X \exists Y\ nat\_square(X, Y)$ **?**	$(X = 0 \wedge Y = 0) \vee (X = s(0) \wedge Y = s(0)) \vee (X = s(s(0)) \wedge Y = s(s(s(s(0))))) \vee \ldots$
$\exists X output(X)$ **?**	$X = 0 \ \vee \ X = s(0) \ \vee \ X = s(s(s(s(0)))) \ \vee \ X = s^9(0) \ \vee \ X = s^{16}(0) \ \vee \ X = s^{25}(0)$

Care logică?

- propozițională
- de ordinul I
- de ordin înalt
- logici modale
- λ-calcul
- ...

Ce metodă de deducție?

- deducție naturală
- rezoluție
- rescriere
- narrowing
- ...

Ce veți vedea la curs

"Bucătăria" din spatele limbajului de specificații Maude și nu numai!

1. Algebre multisortate
   - semantica denotațională
   - specificarea algebrică a tipurilor de date abstracte
2. Logica ecuațională
   - deducția ecuațională
   - asigurarea corectitudinii specificațiilor
3. Rescrieri
   - semantica operațională
   - metodă de demonstrare (deducție) automată
4. Ideile programării logice
   - narrowing, rezoluție ...

În ce logică ne vom situa?

Logica de ordinul I (**FOL**)

- Var mulț. variabilelor,
- $\mathcal{F}$ mulț. simb. de funcții,
- $\mathcal{P}$ mulț. simbolurilor de relații,
- $\dot{=}, \neg, \rightarrow, \vee, \wedge, \forall, \exists$.

- Termen: $x \in Var$, $f(t_1, \ldots, t_n)$
- Formulă atomică: $P(t_1, \ldots, t_n)$, $t_1 \dot{=} t_2$
- Formulă: formulă atomică, $\neg\varphi$, $\varphi \rightarrow \psi$, $\varphi \vee \psi$, $\varphi \wedge \psi$, $(\forall x)\varphi$, $(\exists x)\varphi$
  - Clauză Horn: $(\forall x_1 \ldots x_k)((Q_1 \wedge \ldots \wedge Q_n) \rightarrow Q)$,
    - $Q_1, \ldots, Q_n, Q$ sunt formule atomice
    - Q if $\{Q_1, \ldots, Q_n\}$

În ce logică ne vom situa?

Subsisteme ale lui **FOL**

- **HCL**: formulele sunt clauzele Horn
  - fundamentul teoretic al limbajului Prolog
- **EQL**: formulele atomice sunt ecuații cuantificate universal
  - $\mathcal{P} = \emptyset$
- **CEQL**: **HCL** $\cap$ **EQL**
  - **HCL** pentru $\mathcal{P} = \emptyset$
  - $t_1 \doteq t_2$ if $\{u_1 \doteq v_1, \ldots, u_n \doteq v_n\}$
  - cuantificată universal cu toate variabilele care apar

La curs vom folosi **CEQL** (logica ecuațională condiționată) în varianta multisortată!

Laborator

Ce veți vedea la laborator

Pentru partea practică veți folosi limbajul Maude:

- un limbaj de specificații executabil,
- un fragment este bazat pe logica ecuațională,
- semantica operațională este bazată pe rescriere,
- `http://maude.cs.uiuc.edu/`

În plus, veți face exerciții suport pentru curs.

Planificare laboratoare

- Săptămânile 1 - 7: Limbajul Maude
- Săptămâna 8: Lucrare
- Săptămâna 9: Distribuiere proiecte
- Săptămânile 9 - 13: Seminarii - exerciții suport pentru curs
- Săptămâna 14: Predare proiecte

Pe săptămâna viitoare!