# Programare Logică

2015-2016

FMI

# Cuprins

# Organizare

# Instructori

- **Denisa Diaconescu**
  - curs și laboratoare 2341, 2342
  - denisa.diaconescu@gmail.com; ddiaconescu@fmi.unibuc.ro

# Instructori

- **Denisa Diaconescu**
  - curs și laboratoare 2341, 2342
  - denisa.diaconescu@gmail.com; ddiaconescu@fmi.unibuc.ro
- **Ana Țurlea**
  - laboratoare 2331, 2332, 2431, 2432, 2441, 2442

# Instructori

- **Denisa Diaconescu**
  - curs și laboratoare 2341, 2342
  - denisa.diaconescu@gmail.com; ddiaconescu@fmi.unibuc.ro
- **Ana Țurlea**
  - laboratoare 2331, 2332, 2431, 2432, 2441, 2442
- **Andrei Sipoș**
  - laboratoare 2351

# Instructori

- **Denisa Diaconescu**
  - curs și laboratoare 2341, 2342
  - denisa.diaconescu@gmail.com; ddiaconescu@fmi.unibuc.ro
- **Ana Țurlea**
  - laboratoare 2331, 2332, 2431, 2432, 2441, 2442
- **Andrei Sipoș**
  - laboratoare 2351
- **Alexandra Otiman**
  - laboratoare 2352

# Instructori

- **Denisa Diaconescu**
  - curs și laboratoare 2341, 2342
  - denisa.diaconescu@gmail.com; ddiaconescu@fmi.unibuc.ro
- **Ana Țurlea**
  - laboratoare 2331, 2332, 2431, 2432, 2441, 2442
- **Andrei Sipoș**
  - laboratoare 2351
- **Alexandra Otiman**
  - laboratoare 2352
- **Claudia Mureșan**
  - laboratoare 2311,2312,2321,2322
  - laboratoare 2411,2412,2421,2422 (Carmen Chirița din aprilie)

# Instructori

- **Denisa Diaconescu**
  - curs și laboratoare 2341, 2342
  - denisa.diaconescu@gmail.com; ddiaconescu@fmi.unibuc.ro
- **Ana Țurlea**
  - laboratoare 2331, 2332, 2431, 2432, 2441, 2442
- **Andrei Sipoș**
  - laboratoare 2351
- **Alexandra Otiman**
  - laboratoare 2352
- **Claudia Mureșan**
  - laboratoare 2311,2312,2321,2322
  - laboratoare 2411,2412,2421,2422 (Carmen Chirița din aprilie)
- **Radu Ștefan Mincu**
  - laboratoare 311

# Suport curs

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

# 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: 30 puncte**
- **Examen: 70 puncte**

# Notare

- **Laborator: 30 puncte**
- **Examen: 70 puncte**

- Condiție minină pentru promovare: cel puțin 50% din fiecare probă
  - laborator: min. 15 puncte și
  - examen: min. 35 puncte

# Laborator: 30 puncte

**Lucrare:**

- Are loc în Săptămâna 7 (4 – 8 aprilie)
- Prezența la lucrare este obligatorie!
- Nu se poate reface
- Timp de lucru: o oră și jumătate

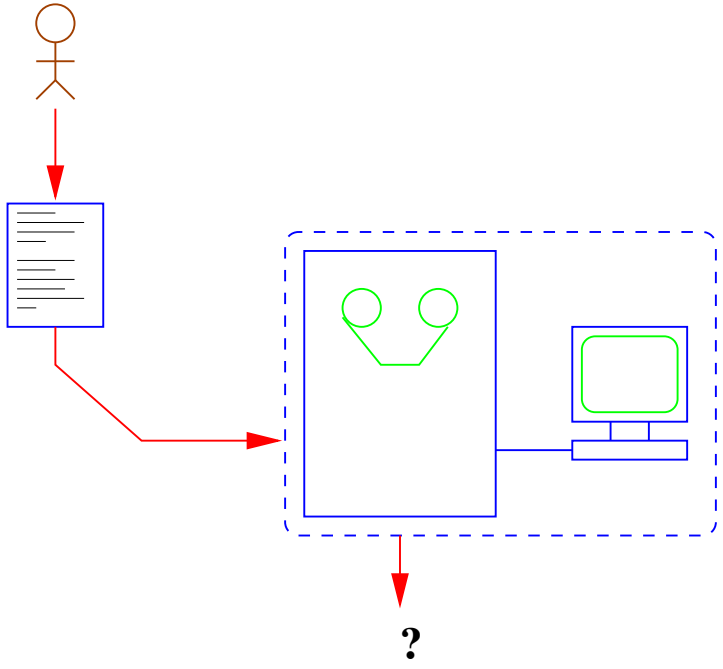
# Examen: 70 puncte

- 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 9-14).
- La examen, trebuie să obțineți min. 35 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 <iostream>
using namespace std;
main()
{
  int square;
  for(int i = 1; i <= 5; ++i)
  {
    square = i * i;
    cout <<  square << endl;
  }
}
```

# Un program imperativ simplu

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

- Este corect?

# Un program imperativ simplu

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

- Este corect? În raport cu ce?

# Un program imperativ simplu

```
#include <iostream>
using namespace std;
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:
  - să permită descrierea problemelor (specificații), și
  - să raționeze despre implementarea lor (corectitudinea programelor).

# Logica

- Un mijloc de a clarifica/modela procesul de a "raționa".

# 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.

# 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:
  - $a_1 : iubeste(Aristotel, prajituri)$
  - $a_2 : (\forall X)\ iubeste(X, prajituri) \rightarrow prieten(Platon, X)$
  - $a_3 : prieten(Platon, Aristotel)$
  - $a_1, a_2 \vdash a_3$

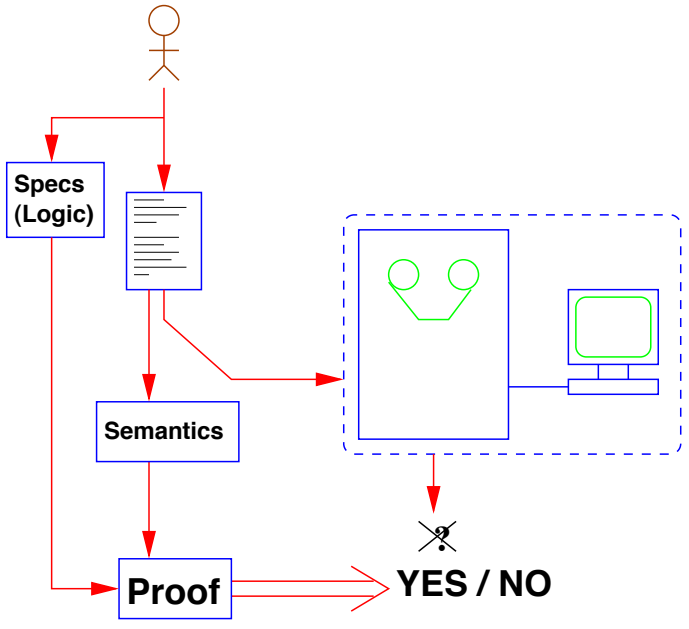
# Care logică?

- propozițională
- de ordinul I
- de ordin înalt
- logici modale
- $\lambda$-calcul
- logici cu mai multe valori
- ...

# Ce metodă de deducție?

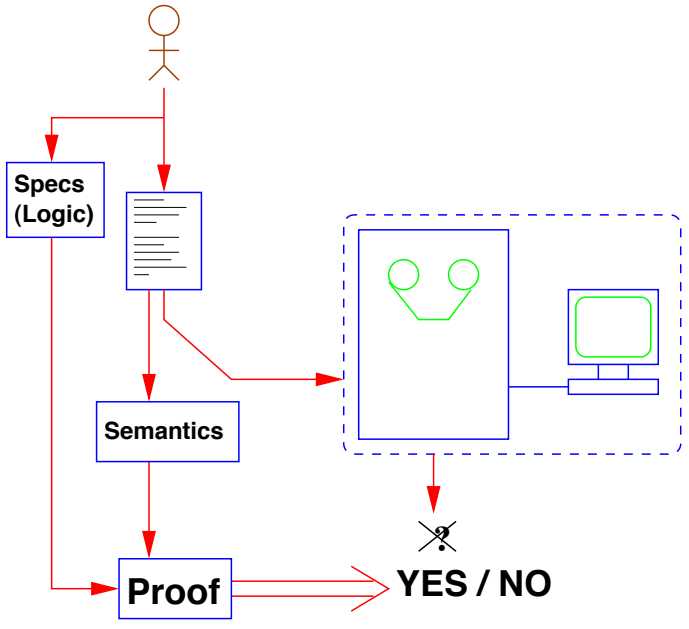
- deducție naturală
- rezoluție
- rescriere
- narrowing
- ...

# Folosind logica




Logica ne permite să reprezentăm/modelăm 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, ...)

# Semantica unui program

- Semantica dă un "înțeles" (obiect matematic) unui program.

# Semantica unui program

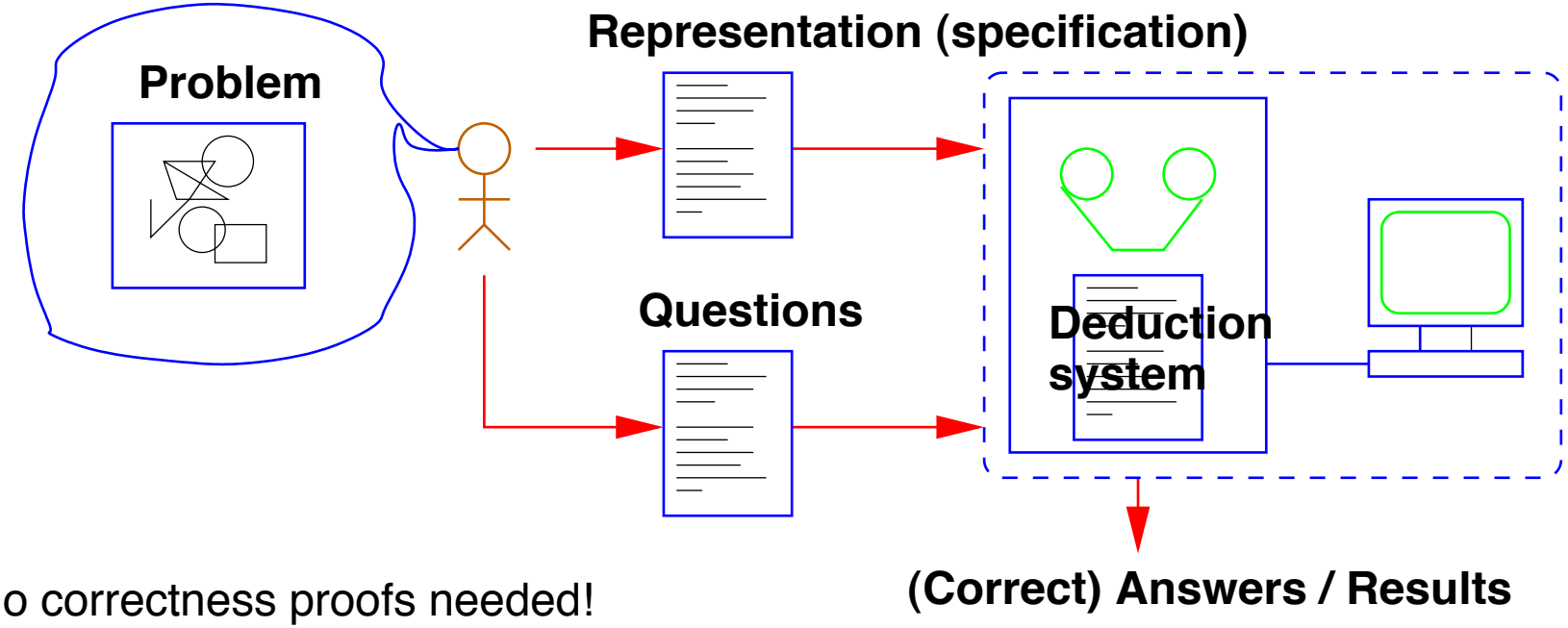
- Semantica dă un "înțeles" (obiect matematic) unui program.
- Semantica trebuie:
  - să poată verifica că un program satisface condițiile cerute.
  - să poată demonstra că două programe au aceeași semantica.
  - ...

# Tipuri de Semantică

- Denotațională:
  - Înțelesul programului este definit abstract ca element dintr-o structură matematică adecvată.
- 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.

# 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!

# Programare Logică

- Programarea logică este
  - o paradigmă de programare bazată pe logică
  - o instanța a programării declarative

# Programare Logică

- Programarea logică este
  - o paradigmă de programare bazată pe logică
  - o instanța a programării declarative

- Unul din sloganurile programării logice:

  **Program = Logică + Control** *(R. Kowalski)*

# Programare Logică

- Programarea logică este
  - o paradigmă de programare bazată pe logică
  - o instanța a programării declarative
- Unul din sloganurile programării logice:

  **Program = Logică + Control** *(R. Kowalski)*
- Programarea logică poate fi privită ca o deducție controlată.

# Programare Logică

- Programarea logică este
  - o paradigmă de programare bazată pe logică
  - o instanța a programării declarative
- Unul din sloganurile programării logice:

  **Program = Logică + Control** *(R. Kowalski)*
- Programarea logică poate fi privită ca o deducție controlată.
- Exemple de limbaje de programare logică:
  - Prolog
  - Answer set programming (ASP)
  - Datalog

# Programare Logică

- Un program scris într-un limbaj de programare logică este

  o listă de formule într-o logică

  ce exprimă fapte și reguli despre o problemă.

# Programare Logică

- Un program scris într-un limbaj de programare logică este

  o listă de formule într-o logică

  ce exprimă fapte și reguli despre o problemă.

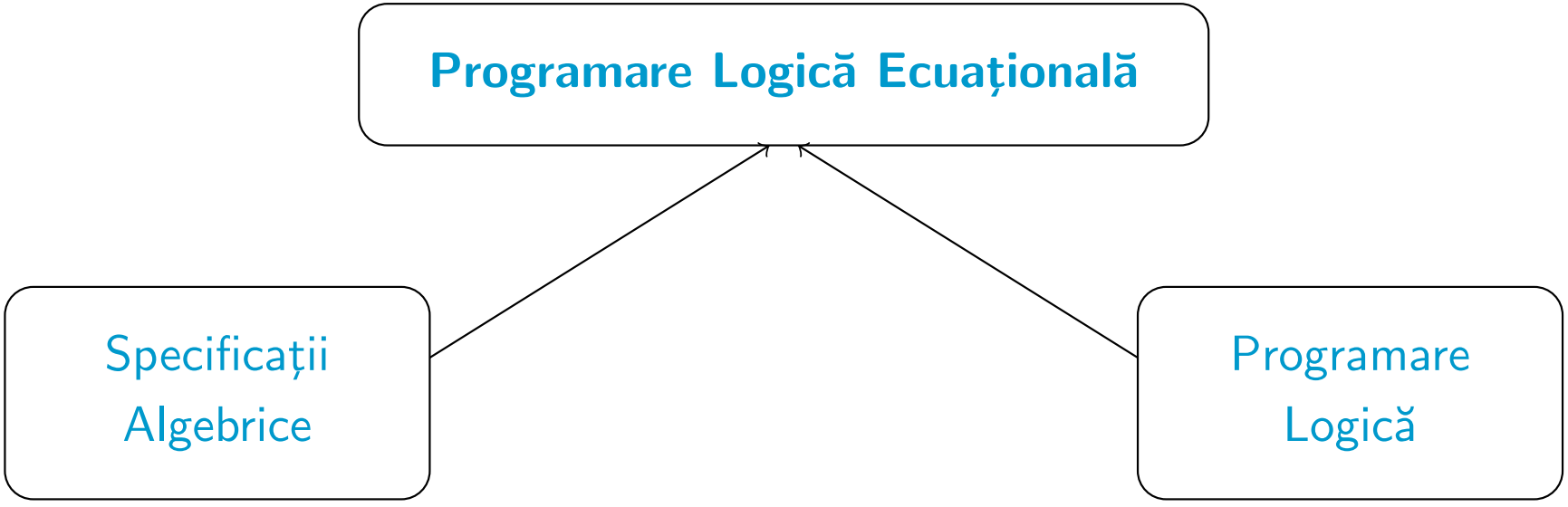
- Regulile sunt scrise sub formă de clauze

  $$\mathtt{H} \ :- \ \mathtt{B_1}, \ldots, \mathtt{B_n} \ .$$

  și sunt gândite ca implicații logice:

  $$H \text{ daca } B_1 \text{ si } \ldots \text{ si } B_n$$

# Programare Logică Ecuațională



# Programare Logică Ecuațională

- Programarea logică ecuațională unește
  - programarea logică bazată pe clauze Horn *(Prolog)*
  - programarea ecuațională bazată pe logica ecuațională

# Programare Logică Ecuațională

- Programarea logică ecuațională unește
  - programarea logică bazată pe clauze Horn *(Prolog)*
  - programarea ecuațională bazată pe logica ecuațională
- Câștig: Putem folosi egalitatea!

# Specificații algebrice

- Tipurile abstracte de date (ADTs) sunt des întalnite în dezvoltarea de aplicații/limbaje
- ADT sunt un obiect matematic pentru tipuri de date
  - un tip de date este definit prin comportamentul său (semantica) din punctul de vedere al utilizatorului
  - comportamentul este specificat în funcție de valorile posibile, operațiile posibile pe tipul de date etc.
- Tipuri abstracte de date vs. structuri de date
  - structurile de date sunt o reprezentare concretă a datelor și sunt punctul de vedere al unui limbaj, nu al utilizatorului.
- Specificațiile algebrice oferă un cadru matematic pentru a defini ADTs

# Ce veți vedea la curs

1. Specificații Algebrice
   - semantica denotațională
   - specificarea algebrică a tipurilor de date abstracte

# Ce veți vedea la curs

1. Specificații Algebrice
   - semantica denotațională
   - specificarea algebrică a tipurilor de date abstracte
2. Logica ecuațională
   - deducția ecuațională
   - asigurarea corectitudinii specificațiilor

# Ce veți vedea la curs

1. Specificații Algebrice
   - semantica denotațională
   - specificarea algebrică a tipurilor de date abstracte
2. Logica ecuațională
   - deducția ecuațională
   - asigurarea corectitudinii specificațiilor
3. Sisteme de Rescriere
   - semantica operațională
   - metodă de demonstrare (deducție) automată

# Ce veți vedea la curs

1. Specificații Algebrice
   - semantica denotațională
   - specificarea algebrică a tipurilor de date abstracte
2. Logica ecuațională
   - deducția ecuațională
   - asigurarea corectitudinii specificațiilor
3. Sisteme de Rescriere
   - semantica operațională
   - metodă de demonstrare (deducție) automată
4. Ideile programării logice ecuaționale
   - narrowing
   - rezoluție

# În ce logică vom lucra?

# În ce logică vom lucra?

Logica de ordinul I (**FOL**)

# În ce logică vom lucra?

Logica de ordinul I (**FOL**)

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

# În ce logică vom lucra?

Logica de ordinul I (**FOL**)

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

- Termen: $x \in \mathit{Var}$, $f(t_1, \ldots, t_n)$

# În ce logică vom lucra?

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: $P(t_1, \ldots, t_n)$, $t_1 \dot{=} t_2$

# În ce logică vom lucra?

## Logica de ordinul I (**FOL**)

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

- Termen: $x \in \mathit{Var}$, $f(t_1, \ldots, t_n)$
- Formulă atomică: $P(t_1, \ldots, t_n)$, $t_1 \dot{=} t_2$
- Formulă: formulă atomică, $\neg\varphi$, $\varphi \rightarrow \psi$, $\varphi \vee \psi$, $\varphi \wedge \psi$, $(\forall x)\varphi$, $(\exists x)\varphi$

# În ce logică vom lucra?

## 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 $\{Q_1, \ldots, Q_n\}$

# Subsisteme ale **FOL**

# Subsisteme ale **FOL**

- **HCL**:
  - formulele sunt clauzele Horn
  - fundamentul teoretic al limbajului Prolog

# Subsisteme ale **FOL**

- **HCL**:
  - formulele sunt clauzele Horn
  - fundamentul teoretic al limbajului Prolog
- **EQL**:
  - formulele sunt ecuații cuantificate universal
  - $\mathcal{P} = \emptyset$

# Subsisteme ale **FOL**

- **HCL**:
  - formulele sunt clauzele Horn
  - fundamentul teoretic al limbajului Prolog
- **EQL**:
  - formulele sunt ecuații cuantificate universal
  - $\mathcal{P} = \emptyset$
- **CEQL**:
  - **HCL** pentru $\mathcal{P} = \emptyset$

# Subsisteme ale **FOL**

- **HCL**:
  - formulele sunt clauzele Horn
  - fundamentul teoretic al limbajului Prolog
- **EQL**:
  - formulele sunt ecuații cuantificate universal
  - $\mathcal{P} = \emptyset$
- **CEQL**:
  - **HCL** pentru $\mathcal{P} = \emptyset$

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/`
- Maude REPL: `http://maude.cvlad.info/`

În plus, veți face exerciții suport pentru curs.

# Planificare laboratoare

- Săptămânile 1 - 6: Limbajul Maude
- Săptămâna 7: Lucrare
- Săptămâna 8: Finalizare note laborator
- Săptămânile 9 - 14: Seminarii - exerciții suport pentru curs


Pe săptămâna viitoare!