LOGIČKO PROGRAMIRANJE

Logičko programiranje

- Vrsta deklarativnog programiranja
  - Definiše šta program treba da reši, ne i kako

- Oblasti primene logičkog programiranja
  - Dokazivanje teorema
  - Obrada prirodnih jezika
  - Baze podataka
  - Mašinko učenje (rule-based machine learning)

Funkcionalno i logičko programiranje

- Logičko programiranje
  - Bazirano na matematičkoj definiciji funkcije
- Logičko programiranje:
  - Bazirano na matematičkoj logici
    - Konkretnije: na **logici predikata prvog reda**

Iskazna i predikatska logika

- Iskaz je svaka rečenica koja može da ima istinitosnu vrednost: tačno ili netačno
- Složeni iskazi se grade korišćenjem operatora:
  - $\neg$, $\wedge$, $\vee$, $\Rightarrow$, $\Leftrightarrow$, ...
- Iskazna logika se bavi proverom da li je vrednost nekog iskaza tačna ili netačna.

- Predikat je iskaz čija istinitosna vrednost zavisi od vrednosti nekog parametra
- Primer:
  - „3 je paran broj“ – iskaz
  - „x je paran“ - predikat

Kvantifikatori ∀ i ∃

- U predikatskoj logici se osim logičkih operatora u građenju formule koriste i kvantifikatori:
  - Univerzalni kvantifikator: ∀
  - Egzistencijalni kvantifikator: ∃

- Uvođenjem kvantifikatora istinitosna vrednost predikatske formule prestaje da zavisi od parametara.
- Predikatske formule sa kvantifikatorima:

  $(\forall x)\,(P(x))$

  $(\exists x)\,(P(x))$
- Predikatske formule sa kvantifikatorima:

  $(\forall x)$ („x je paran broj“) - $\bot$

  $(\exists\, x)$ („x je paran broj“) - $\top$

Formalna definicija predikatske logike prvog reda

- Predikatska logika prvog reda sadrži:
  - Termove
  - Atiomične forumule
  - Formule

- Termovi – objekti nekog domena:
  - Konstante
  - Promenljive
  - Funkcije
- Atomične formule i formule mogu imati istinitosnu vrednost

Atomične formule i formule

- Definicija atomičnih formula
  - Logičke konstante T i ⊥ su atomične formule.
  - Ako je ρ n-arni relacijski simbol i $t_1, t_2, \ldots, t_n$ termovi , onda je $\rho(t_1, t_2, \ldots, t_n)$ atomična formula.

- Definicija formule
  - Atomične formule su formule.
  - Ako je A formula onda je $\neg(A)$ formula.
  - Ako su A i B formule, onda su i $A \vee B$, $A \wedge B$, $A \Rightarrow B$ i $A \Leftrightarrow B$ formule.
  - Ako je A formula, a x promenljiva onda su i $(\forall x)(A)$ i $(\exists x)\ (A)$ formule.

Logički programski jezici

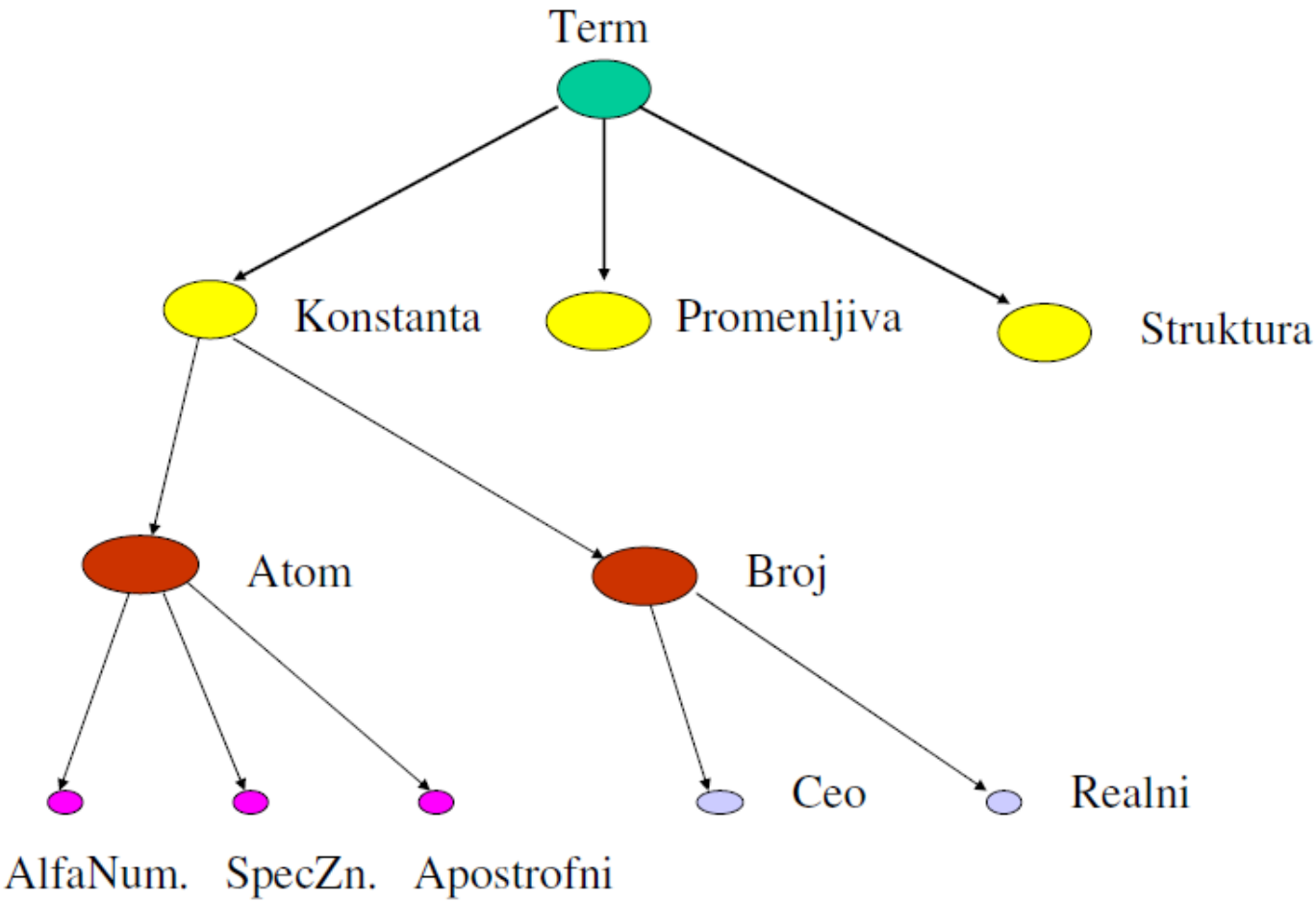
- Logički programski jezici treba da obezbede:
  - Način za predstavljanje činjenica – činjenica odgovara formuli logike predikata prvog reda
  - Način za predstavljanje relacija među činjenicama
  - Način za izvođenje zaključaka (novih činjenica) na osnovu postojećih

- Predstavnici logičkih programskih jezika:
  - PROLOG
  - DataLOG
  - ASP (Answer Set Programming), NE ~~Active Server Page~~

ELEMENTI LOGIČKIH JEZIKA KROZ PROGRAMSKI JEZIK PROLOG

Term u programskom jeziku PROLOG

- Termovi su strukture podataka u programskom jeziku Prolog



Konstante u programskom jeziku PROLOG

- Numeričke
  - Celi brojevi – označeni ili neoznačeni niz dekadnih difara
  - Realni brojevi - označeni ili neoznačeni niz dekadnih difara koji sadrži decimalnu tačku
- Atomične
  - Alfanumeričke (simbličke) – niz slova cifara i _ u kojem je prvi znak malo slovo
    - `petar, pavle, radna_nedelja`
  - Specijalnoznačne
    - ?-, :-, ::=, ...
  - Apostrofne (stringovi) – niz karaktera između apostrofa
    - `‘Ovo je strig’`

Operacija unifikacije (izjednačavanja)

Unifikacija objekata T i S se vrši na sledeći način:

- Ako su T i S konstante, unifikacija je moguća samo ako se radi o identičnom objektu
- Ako je T promenljiva, a S objekat: promenljivoj T se dodeljuje vrednost objekta S
- Ako je S promenljiva, a T objekat: promenljivoj S se dodeljuje vrednost objekta T
- Ako su S i T strukture, unifikacija se može izvršiti ako:
  - Imaju isto ime i n-arnost
  - Sve odgovarajuće kompnente se mogu unifikovati

Promenljive i strukture u programskom jeziku Prolog

- Promenljiva – niz slova, cifara i _ u kojem je prvi znak Veliko slovo:
  - Nije: simboličko ime memorijske lokacije gde se čuva neka vrednost
  - Jeste: matematičko tumačenje promenljive – moždeda se zameni bilo kojom vrednošću

- Struktura – ako su t_1, t_2, ... , t_n, termovi tada je struktura:

  <ime>(t_1, t_2, ... , t_n)

  - Primeri struktura:

```
datum(22, maj, 2019)
knjiga(Naslov, Autor, Izdavac, Godina)
trougao(tacka(2,3), tacka(5,6), tacka(2,5))
```

Naredbe probramskog jezika PROLOG

- Postoje 3 osnovne vrste naredbi:
  - Činjenice – predikatske formule čija je intinitosna vrednost uvek T
  - Pravila – definišu način za izvođenje novih činjenica (zaključaka)
  - Upiti – definišu problem koji program treba da reši

- Činjenice i pravila se čuvaju na disku i predstavljaju bazu znanja.
- Upiti – naredbe programa – program korišćenjem baza znanja kreira odgovor na upit

Činjenice

- Činjenica sadrži neke informacije o objektima i relacijama između njih.
- Činjenica je naredba napisana u jednom redu koja počinje predikatom i završava se tačkom.
- Primeri činjenica:

```
zena(ana).
roditelj(ana, petar).
roditelj(pavle, ana).
```

Pravila

- Pravilo sadži 2 dela:
  - Zaključak (posledica) – leva strana pravila
  - Uslov (telo) – desna strana pravila

- Format pravila:
  - **<zaključak> :- <uslov>.**

- Primer pravila:

```
predak(A,B) :- roditelj(A,B).
predak(A,B) :- roditelj(A,C), predak(C,B).
```

Upiti

- Format upita:

  ?- <predikat>.

- Postoje 2 osnovna tipa upita u programskom jeziku Prolog (zavisno od toga kakav odgovor traži):
  - *Yes/No query* – upit u čijem predikatu učestvuju samo konstante.
    - Odgovor na ovakav upit je može biti samo T ili $\perp$ (YES/NO).
  - *Unification* /No query – upit u čijem predikatu učestvuju i promenljive.
    - Odgovor na ovakav upit su k-torke objekata koje zadovoljavaju upit (gde k predstavlja broj promenljivih u predikatu) ili $\perp$ (ne postoji katorka objekata koji zadovoljavaju upit)

Upiti - primeri

- Upit:

```
?- predak(pavle, petar)
```

- Odgovor:

```
yes
```

- Upit:

```
?- predak(X, petar)
```

- Odgovor:

```
X = ana
X = pavle
```

Kreiranje odgovora u Prologu

- Ukoliko se upit sastoji od više podupita, treba tražiti odgovore za svaki podupit
  - Ukoliko se radi o „Yes/No“ upitu, cilj je zadovoljen, ako su svi podciljevi zadovoljeni. Npr. upit

    ```
    ?- zena(ana), predak(ana, petar)
    ```

    ima vrednost T, jer svi njegovi podupiti imaju vrednost T.
  - Ukoliko se radi o „Unification/No“ upitu, rešenje je presek skupova unifikacija koje zadovoljavaju podupite. Npr. rešenje upita

    ```
    ?- zena(X), predak(X, petar)
    ```

    je X=ana, jer su rešenja podupita

    ```
    ?- zena(X)    X=ana
    ?- predak(X, petar)    X=ana, X=pavle
    ```

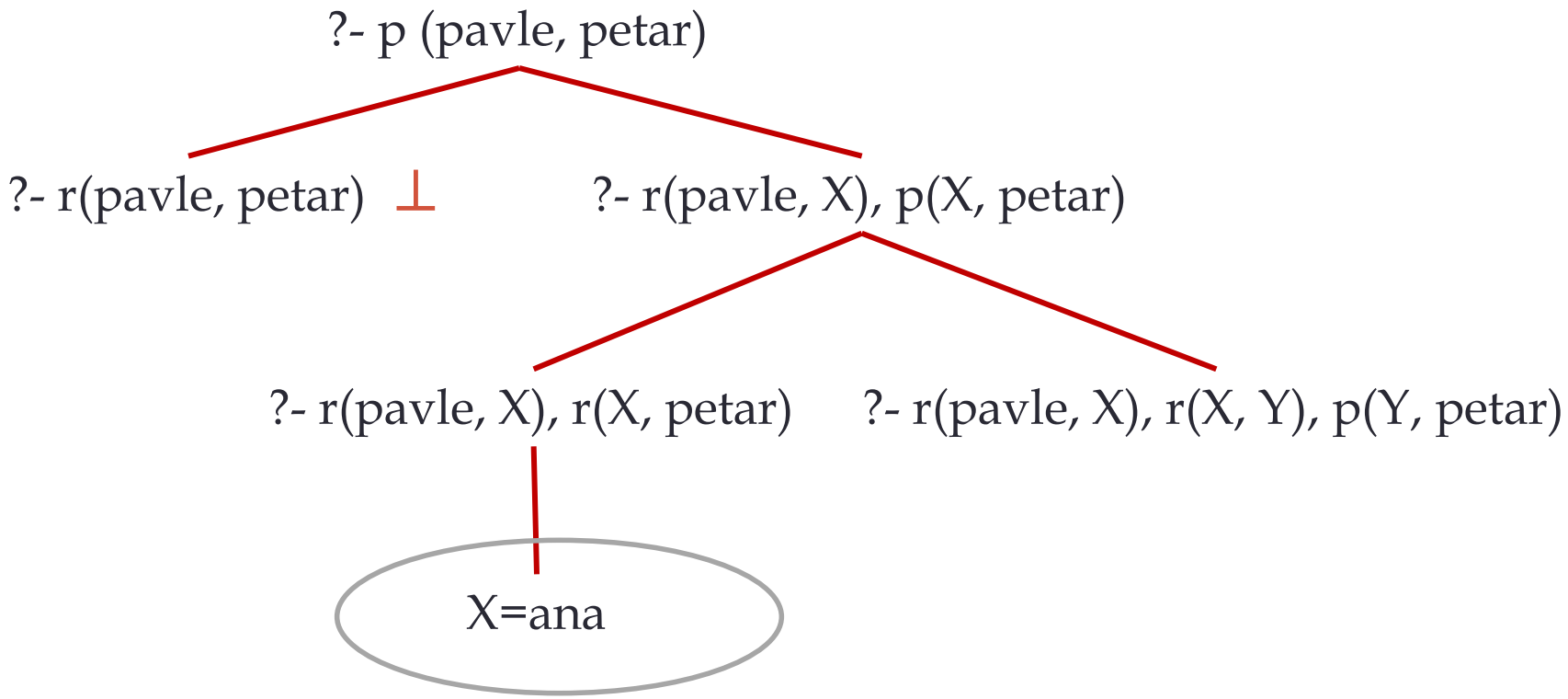
Kreiranje odgovora

- Za svaki podupit se kreira stablo traženja rešenja na sledeći način:
  - Koren stabla je podupit čije se rešenje traži
  - Terminalni čvorovi:
    - Čvor uspeha:
      - Činjenica (podupit je uvek tačan)
      - Skup unifikacija za koje je podupit tačan (tj. skup unifikacija koje upit preslikavaju u činjenicu)
    - Čvor neuspeha
      - Podupit koji nema rešenje (ne postoji pravilo kojim bi se se podupit razvijao niti unifikacija kojom bi se preslikao u činjenicu)
  - Čvorovi potomci se kreiraju primenom pravila za generisanje „zaključka“ koji je sadržan u roditeljskom čvoru (podupitu)

Kreiranje odgovora

- Stablo traženja se može kreirati na dva načina:
  - Top – down metodom – polazeći od upita, primenom konačnog broja pravila pokušavamo da dođemo do činjenica
  - Bottom-up metodom – polazeći od činjenica, primenom konačnog broja pravila pokušavamo da dođemo do istinitosne vrednosti upita
- Prolog interpretator koristi top-down metodu

Kreiranje odgovora – primer 1



Kreiranje odgovora – primer 2