

Rešitve izpita iz Teorije programskih jezikov

\$Id: tpj-resitve.lyx,v 1.2 2003/06/24 16:10:04 roman Exp \$

14. junij 2001

1. Kako so v logiki prvega reda definirani naslednji pojmi:

- (a) klavzalna oblika formul
- (b) Hornov stavek
- (c) definitni stavek

Odgovor:

- (a) Klavzalna oblika formul je oblika, v kateri kvantifikatorji niso eksplicitno zapisani.
 - (b) Hornov stavek je stavek z največ enim pozitivnim literalom.
 - (c) Definitni stavek je stavek z natanko enim pozitivnim literalom.
2. Opiši v grobem, kako poteka resolucijski dokaz formule oblike $A \implies B$, kjer sta A in B logični formuli prvega reda. Kakšno funkcijo ima v tem postopku resolucijski korak? Zapiši splošno obliko resolucijskega koraka.

Odgovor: Resolucijski dokaz formule $A \implies B$ poteka nekako takole:

- (a) Dokazujemo s protislovjem, zato formulo negiramo, da dobimo $A \wedge \neg B$.
- (b) Dobljeno pretvorimo v konjuktivno normalno obliko.
- (c) Z večkrat uporabljenim resolucijskim korakom pridemo do protislovja, ki je dokaz, da velja prvotna formula, tj. $A \implies B$.

Resolucijski korak nam poenostavi formulo s tem, da iz formule odstrani neuporabni literal. V splošnem se glasi:

$$\frac{P_1 \vee Q \quad \neg P_2 \vee R}{Q/\Theta \vee R/\Theta},$$

če lahko unificiramo z naborom spremenljivk Θ , torej tako, da velja

$$P_1/\Theta \equiv P_2/\Theta.$$

3. Definiraj v logiki pomen naslednjih stavkov:

- (a) Every student of Newton did an experiment that involved gravity.
- (b) Every dog chases a cat that hates the dog.
- (c) Every dog chases a cat that hates a dog.

Odgovor: ??? Ne vem sicer, če je res mišljena logika (ne kakšen čuden prolog), ampak, če je, gre nekako takole:

- (a) $\forall s : \text{StudentOfNewton}(s) \implies \text{DidGravityExperiment}(s)$.
- (b) $\forall d \forall c : (\text{Dog}(d) \wedge \text{Cat}(c) \wedge \text{Hates}(c, d)) \implies \text{Chases}(d, c)$.
- (c) $\forall d \forall c : (\text{Dog}(d) \wedge \text{Cat}(c) \wedge (\exists d' : \text{Dog}(d') \wedge \text{Hates}(c, d'))) \implies \text{Chases}(d, c)$

Če bi bilo treba napisati v prologu, če sem prav razumel, Matej in Peter predlagata:

(a) Definiramo:

```
all(S, studentOf(S, newton) ==>
    exists(E, experiment(E) and did(S,E) and involves(E, gravity))).
```

(b) Definiramo:

```
all(D, dog(D) ==> all(C, cat(C) and hates(C,D) ==>
    chases(D,C))).
```

(c) Definiramo:

```
all(D, dog(D) ==> all(C, cat(C) and exists(D2, dog(D2)
    and hates(C,D2)) ==> chases(D,C)).
```

Je tako prav?

4. Naj bodo dana naslednja gramatična pravila za angleščino:

```
sentence --> noun_phrase, verb_phrase.
noun_phrase --> determiner, noun.
determiner --> [every].
```

Tole pa je ponesrečen poskus definiranja pomena določnika „every“:

```
determiner( exists( X, Property and Assertion)) --> [every].
```

- (a) To pravilo je logično nepravilno, poleg tega pa je tudi nerodno glede na integracijo tega pravila z drugimi pravili v gramatiki. Popravi pravilo!
- (b) Katera stavčna fraza določi delni pomen `Property`, in katera fraza določi delni pomen `Assertion`.

Odgovor:

(a) Damir pravi, da takole:

```
determiner(X, Property, Assertion,
    all(X, Property ==> Assertion)) --> [every].
```

(b) Delni pomen `Property` v priloženi gramatiki določa fraza `noun`, saj se tam `Property` opredeli, v `noun_phrase` pa se že integrira v pomen člena. Delni pomen `Assertion` opredeli `verb_phrase`.

5. Vprašanja iz zadoščanja omejitev:

- (a) V grafu omejitev, čemu ustrezajo vozlišča, čemu povezave?
- (b) Kako je v problemih zadoščanja omejitev definirana konsistentnost povezave (X, Y) med vozliščema X in Y v grafu omejitev, če med spremenljivkama X in Y velja omejitev $p(X, Y)$?
- (c) Na katere tri različne načine se lahko konča konsistentnostni algoritem, ki deluje na povezavah grafa omejitev?
- (d) V primeru, da se konsistentnostni algoritem konča v stanju, ko imamo večvrednostne domene spremenljivke, kako pridemo do rešitve (oz. več rešitev) problema omejitev?

Odgovor:

- (a) Vozišča ustrezajo možnim vrednostim spremenljivk (npr. $D_X \subset \mathbb{Z}$ za spremenljivko X pri $\text{CLP}(\mathbb{Z})$), povezave pa relacijam med njimi.
- (b) Povezava (X, Y) je konsistentna, če $\forall X \in D_X \exists Y \in D_Y : p(X, Y)$.
- (c) Konsistentnosti algoritem se konča, ko:
 - i. nekatere domene postanejo \emptyset ali
 - ii. vsaka domena vsebuje natanko eno vrednost ali
 - iii. so vse domene neprazne in obstaja vsaj ena domena z vsaj dvema vrednostma.
- (d) V primeru večvrednostnih domen pridemo do rešitve tako, da kombinatorno iščemo po reducirani domeni; fiksiramo eno vrednost in pogledamo, če je to rešitev.

6.

- (a) Kaj pomeni v angleščini kratica EBG? Kako ta pojem prevedemo v slovenščino?
- (b) Kaj je pri postopku EBG podano in kaj je rezultat?
- (c) Postopek EBG je precej podoben transformaciji programov z „razvijanjem“ (angl. unfolding), kjer cilje nadomeščamo s cilji v telesih ustreznih stavkov, dokler ni izpolnjen določen pogoj. V čem je razlika med EBG in transformacijo z razvijanjem?

Odgovor:

- (a) EBG pomeni Explanation Based Generalization; posplošitev z razlago.
- (b) Podano:
 - i. teorija o domeni (pravila igre),
 - ii. kriteriji za operativnost,
 - iii. učni primer.Rezultat je posplošen dokaz učnega primera.
- (c) Pri transformaciji z razvijanjem, za razliko od EBG, ni učnega primera.

7. Najbolj pogosto uporabljena shema metainterpreterja v prologu je:

```
prove( true ).
prove( (G1,G2) ) :- % Resi konjunkcijo
...
prove( G ) :- % Resi cilj z uporabo stavka
...
```

- (a) Ustrezno dopolni gornji nastavek programa na mestih „...“.
- (b) Dopolni ta metainterpreter v `prove(Goal, Depth)`, ki vrne `Depth` kot največjo globino klica predikata v dokazu za `Goal` (če je `Goal` dejstvo, potem je `Depth = 0`).

Odgovor:

- (a) Tole smo napisali na predavanjih:

```
prove( true ) :- !.

prove( (G1, G2) ) :- !,
    prove( G1 ),
    prove( G2 ).
```

```

prove(Goal) :-
    clause(Goal, Body),
    prove(Body).

```

(b) Po zgledu prvega primera:

```

prove(true, 0) :- !.

prove((G1, G2), Depth) :- !,
    prove(G1, D1),
    prove(G2, D2),
    max(D1, D2, Depth).

prove(Goal, Depth) :-
    clause(Goal, Body),
    prove(Body, BodyDepth),
    Depth is BodyDepth + 1.

```

Tu je max/3 definiran takole:

```

max(X, Y, X) :- X >= Y, !.
max(X, Y, Y) :- Y >= X.

```

8. Dana je DCG gramatika:

```

move( P0, P) --> step( P0, P), { safe(P) }.
move( P0, P) --> step( P0, P1), { safe(P1) }, move( P1, P).
step( p(X,Z), p(X+1, Z)) --> [right].
step( p(X,Z), p(X-1, Z)) --> [left].
step( p(X,Z), p(X, Z+1)) --> [up].
step( p(X,Z), p(X, Z-1)) --> [down].
safe( p(X,Z)) :-
    -5 =< X, X =< 5,
    -5 =< Z, Z =< 5.

```

Kaj odgovori prolog na spodnja vprašanja:

- (a) ?- move(p(2,2), P, [up], []).
- (b) ?- move(p(2,2), P, [left,up,right,right], []).
- (c) ?- move(p(4,4), P, [up,right,right], []).
- (d) ?- move(p(5,5), P, L, []).

Podaj prve tri prologove odgovore.

- (e) Naj gornja gramatika opisuje gibanje robota. Kaj v tej gramatiki pomenita P0 in P? Kakšne robotove gibe dopušča ta gramatika?

Odgovor:

(a) Prolog vrne:

```

| ?- move(p(2,2), P, [up], []).
P = p(2,2+1) ? ;
no

```

(b) Prolog vrne:

```
| ?- move(p(2,2), P, [left,up,right,right], []).
```

```
P = p(2-1+1+1,2+1) ? ;
```

```
no
```

(c) Prolog vrne:

```
| ?- move(p(4,4), P, [up,right,right], []).
```

```
no
```

(d) Prolog vrne:

```
| ?- move(p(5,5), P, L, []).
```

```
L = [left]
```

```
P = p(5-1,5) ? ;
```

```
L = [down]
```

```
P = p(5,5-1) ? ;
```

```
L = [left,right]
```

```
P = p(5-1+1,5) ?
```

```
yes
```

(e) Spremenljivka P_0 označuje koordinate začetnega polja, P pa koordinate končnega, po premiku. Gramatika dopušča vodoravne in navpične premike za 1, pri čemer nikoli ne smemo zapustiti kvadrata $[-5, 5] \times [-5, 5]$.