1.Podaj określenie wartościowania w KRZ i jego rozszerzenia na dowolne formuły. Podaj przykład wartościowania formuły w KRZ

2. Podaj określenia następujących pojęć w KRZ: spełnialność formuły i zbioru formuł przez wartościowanie, formuła spełnialna, zbiór spełnialny.

3. Omów wynikanie logiczne w KRZ. Podaj określenie logicznej reguły wnioskowania w KRZ i reguły rezolucji zdaniowej. Wykaż, że reguła rezolucji zdaniowej jest logiczną regułą wnioskowania.

4. Podaj określenie derywacji klauzuli ze zbioru klauzul oraz refutacji zbioru klauzul w KRZ. Podaj przykład derywacji i refutacji.

5. Uzasadnij algorytm sprawdzania tautologiczności formuł KRZ za pomocą rezolucji zdaniowej.

6. Podaj określenie języka pierwszego rzędu oraz termów, formuł atomowych, formuł, zdań, termu bazowego i atomu bazowego.

7. Podaj określenia następujących pojęć w LPR: zdanie, domknięcie formuły, prawdziwość formuły w interpretacji.

8. Podaj definicję i własności podstawienia. Na czym polega złożenie podstawień? Podaj przykład złożenia podstawień.

9. Podaj określenie interpretacji języka pierwszego rzędu. Przykład

10. Podaj określenie wyrażenia prostego, przemianowania zmiennych, wariantu wyrażenia prostego. Każde pojęcie zilustruj przykładem.

11. Podaj określenie formuły preneksowej postaci normalniej. Jaki jest związek między formułą A i PNF(A)?

12. Podaj określenie formuły w postaci Skolema. Jaki jest związek między formułą A i SKOL(A)?

13. Co rozumiemy przez konkretyzację formuły(klauzuli) w LPR? Czym jest gr(∑) dla zbioru klauzul ∑? Podaj twierdzenie dotyczące ∑ i gr(∑).

14. Uzasadnij algorytm dowodzenia praw LPR za pomocą rezolucji zdaniowej.

15. Podaj określenie zbioru niezgodności dla zbioru wyrażeń prostych i zilustruj przykładem.

16. Podaj algorytm wyznaczania najbardziej ogólnego unifikatora zbioru wyrażeń oraz twierdzenie o unifikacji.

17. Podaj określenie klauzuli definitywnej, programu definitywnego, klauzuli celu, klauzuli Horna.

18. Omów strategię Prologu

19. Podaj regułę rezolucji liniowej i derywacji liniowej.

20. Podaj określenie refutacji liniowej dla programu P i celu G oraz odpowiedzi obliczonej.

21. Podaj określenie odpowiedzi obliczonej i poprawnej dla $P \cup \{G\}$, gdzie P jest programem definitywnym, a G – celem. Sformułuj twierdzenie o poprawności rezolucji liniowej. Podaj silne twierdzenie o pełności rezolucji liniowej.

22. Podaj określenie SLD-drzewa dla $P \cup \{G\}$ zgodnego z regułą selekcji R.

23. Podaj określenie reguły selekcji i sformułuj twierdzenie o nieistotności reguł selekcji.

24. Podaj podstawowe cechy prologu jako implementacji SLD rezolucji. Objaśnij użyte pojęcia.

25. Termy rachunku lambda

26. Alfa-konwersja. Definicja, przykłady

27.Beta-redukcja. Definicja, przykłady

28. Postać normalna termu w rachunku lambda

29. Własność Churcha-Rossera

30. Reprezentacja liczb naturalnych w rachunku lambda.

1.Podaj określenie wartościowania w KRZ i jego rozszerzenia na dowolne formuły. Podaj przykład wartościowania formuły w KRZ

1. Wartości logiczne: 1 (prawda), 0 (fałsz).
2. Wartościowaniem nazywamy funkcję: w : V→{0,1}.
3. Funkcją logiczną nazywamy funkcję f : $\{0,1\}^n$→{0,1}.
Określamy funkcje:
$f_¬$:{0,1}->{0,1} następująco:

x	$f_¬(x)$
0	1
1	0

$f_{\wedge}$, $f_{\vee}$, $f_{->}$, $f_{<->}$: $\{0,1\}^2$->{0,1} następująco:

x	y	$f_{\wedge}(x,y)$	$F_v(x,y)$	$f_{->}(x,y)$	$F_{<->}(x,y)$
1	1	1	1	1	1
1	0	0	1	0	0
0	1	0	1	1	0
0	0	0	0	1	1

4. Każde wartościowanie w : V→{0,1} rozszerzamy do funkcji
ŵ:FOR->{0,1}, (ŵ (A) – wartość logiczna formuły A przy wartościowaniu w) następująco:
ŵ(*p*) = *w*(*p*) dla *p* ∈ *V*
ŵ(~*A*) = $f_¬$ (ŵ(*A*))
ŵ(*A*^*B*) = *f*^ (ŵ(*A*), ŵ(*B*))
ŵ(*AvB*) = *fv*(ŵ(*A*),ŵ(*B*))
ŵ(*A_B*) = $f_¬$ (ŵ(*A*),ŵ(*B*))
ŵ(*A*<=>*B*) = *f*<=>(ŵ(*A*),ŵ(*B*))

Przykład wartościowania:
A: *p->q* $_{\vee}$ *r*
w: w(p)=1, w(q)=0, w(r)=1,
ŵ(A)=ŵ(*p->q* $_v$ *r)= f*$_{->}$(ŵ(p),(ŵ(q_vr))=$f_{->}$(ŵ(q),f_v(ŵ(q),ŵ(r)))=
$f_{->}$(1,f_v(0,1)) = $f_{->}$(1,1)=1

2. Podaj określenia następujących pojęć w KRZ: spełnialność formuły i zbioru formuł przez wartościowanie, formuła spełnialna, zbiór spełnialny.

Spełnialność formuły i zbioru formuł przez wartościowanie:
1. Wartościowanie *w* spełnia formułę A (oznaczenie: *w* ⊨ A), jeżeli ŵ(A) = 1.

2. Wartościowanie *w* spełnia zbiór Γ c FOR (oznaczenie $w \models \Gamma$) jeżeli $\hat{w}(A) = 1$ dla każdej formuły $A \in \Gamma$

Formułę A nazywamy spełnialną, jeżeli istnieje wartościowanie takie, że $w \models A$.
Zbiór formuł Γ c FOR nazywamy spełnialnym, jeśli istnieje wartościowanie takie, że $w \models \Gamma$.

3. Omów wynikanie logiczne w KRZ. Podaj określenie logicznej reguły wnioskowania w KRZ i reguły rezolucji zdaniowej. Wykaż, że reguła rezolucji zdaniowej jest logiczną regułą wnioskowania.

Wynikanie logiczne:

a) Formuła A wynika ze zbioru formuł Γ w KRZ
b) Zbiór $\Gamma \cup (\neg A)$ nie jest spełnialny

(po staremu było: Formuła A wynika ze zbioru formuł Γ w KRZ, jeżeli dla każdego wartościowania w zachodzi warunek: Jeżeli $w|=\Gamma$, to $w|=A$.)

Logiczna reguła wnioskowania:
Jeżeli ze zbioru formuł {A1,…,An} które będziemy nazywać przesłankami, wynika formuła A, którą będziemy nazywać wnioskiem, to schemat:

$$\begin{array}{c} A1 \\ \vdots \\ An \\ \hline A \end{array}$$

nazywamy logiczną regułą wnioskowania.

Literałem pozytywnym nazywamy dowolną zmienną zdaniową $\mathbf{p \in V}$, a literałem negatywnym negację zmiennej zdaniowej $\neg \mathbf{p}, \mathbf{p \in V}$. Literały oznaczamy L1,L2,…
Wtedy **reguła rezolucji zdaniowej ma postać**:

$$\frac{\{L1,L2,\ldots,Lm,p\} \quad \{L1',L2',\ldots,Ln',\neg p\}}{\{L1,L2,\ldots,Lm,L1',L2',\ldots,Ln'\}}$$

dla $n,m \geq 0$

Dowód, że reguła rezolucji zdaniowej jest logiczną regułą wnioskowania:
Trzeba pokazać że dla każdego wartościowania w zachodzi warunek:
Jeżeli $w \models \{L1 ,L2 ,\ldots,Lm, p\}$ i $w \models \{L1',L2',\ldots,Ln',\neg p\}$ to $w \models \{L1 ,L2 ,\ldots,Lm, L1',L2',\ldots,Ln'\}$
Zakładamy że:

1) $w \models \{L1 ,L2 ,\ldots,Lm, p\}$
2) $w \models \{L1',L2',\ldots,Ln',\neg p\}$

Rozważamy osobno oba przypadki:
1) $w(p)=1$, wtedy $w(\neg p)=0$, czyli $w|\neq(\neg p)$, wtedy zał 1) $w \models Li'$ gdzie $i \in \{1,\ldots,n\}$
Stąd $w \models \{L1 ,L2 ,\ldots,Lm, L1',L2',\ldots,Ln'\}$

2) $w(p)=0$ wtedy $w(\neg p)=1$, czyli $w|\neq(p)$ wtedy zał 2) $w \models Li$ gdzie $i \in \{1,\ldots,m\}$
Stąd $w \models \{L1 ,L2 ,\ldots,Lm, L1',L2',\ldots,Ln'\}$

Stara wersja:
Zakładamy, że istnieje wartościowanie *w* spełniające obie przesłanki: $w|=\{L1,L2,\ldots,Ln,p\}$, $w|=\{L1',L2',\ldots,Ln',\neg p\}$. Rozważmy dwa przypadki:

1. *w*(p)=1, wtedy w(¬ p)=0, tzn. w|≠(¬ p), stąd *w*(Li')=1 dla pewnego 1 ≤ i ≤ n. Zatem *w* spełnia wniosek *w* |= {L1',L2',…,Ln',}.

2. w(p)=0, wtedy istnieje 1 ≤ j ≤ n takie, że *w*(Lj)=1.

Zatem *w* spełnia wniosek *w*|={L1,L2,…,Lm,}.

4. Podaj określenie derywacji klauzuli ze zbioru klauzul oraz refutacji zbioru klauzul w KRZ. Podaj przykład derywacji i refutacji.

Derywacją klauzuli A ze zbioru klauzul Σ nazywamy drzewo etykietowane (X, E, f) takie, że:

1. E jest zbiorem klauzul Σ.

2. dla każdego liścia x drzewa X, f(x) ϵ Σ.

3. f(**Λ**) = A.

4. dla każdego wierzchołka x drzewa nie będącego liściem f(x) jest rezolwentą etykiet bezpośrednich potomków wierzchołka x.

Refutacja zbioru klauzul:

Derywację klauzuli pustej □ ze zbioru klauzul Σ nazywamy refutacją zbioru Σ.

Derywacja:

$\Sigma = \{ \{ p, q \}, \{ \neg p , q \} \}, \Sigma \vdash_{RZ} \{ q \}$

$\{p,q\} \quad \{\neg p,q\}$

$\backslash \ /$

$\{q\}$

Refutacja:

$\Sigma = \{\{p, q\}, \{\neg p, q\}, \{p, \neg q\}, \{\neg p, \neg q\}\}, \quad \Sigma \vdash_{RZ} \{q\}$

$\{p, q\} \quad \{\neg p, q\} \qquad \{p, \neg q\} \quad \{\neg p, \neg q\}$

$\backslash \ /$ $\qquad\qquad$ $\backslash \ /$

$\{q\} \qquad\qquad \{\neg q\}$

$\backslash \qquad\qquad /$

□

5. Uzasadnij algorytm sprawdzania tautologiczności formuł KRZ za pomocą rezolucji zdaniowej.

Algorytm:

Wejście: formuła A

Wyjście: odpowiedź tak, jeżeli formuła A jest tautologią KRZ, nie, w przeciwnym wypadku.

(1) bierzemy formułę B identyczną z ¬A,

(2)sprowadzamy formułę B do koniunkcyjnej postaci normalnej

(3) KPN(B) przedstawiamy w postaci klauzulowej Σ(KPN(B)).

(4) Szukamy derywacji klauzuli pustej □ ze zbioru Σ(KPN(B)).

(5) Jeżeli klauzula pusta została otrzymana, to odpowiedź-**T**

(6) Jeżeli algorytm zatrzymał się ze względu na brak możliwości stosowania reguły rezolucji, mamy odp. -**NIE**.

Uzasadnienie:

Formuła A jest tautologią KRZ.

↕ *Formuła A jest tautologią KRZ ↔ Formuła KPN(¬A) nie jest spełnialna*

Formuła KPN(¬A) nie jest spełnialna.

↕ *w ⊨A ↔w ⊨KPN(A) ↔w ⊨ Σ(KPN(A))*

Formuła KPN(¬A) nie jest spełnialna.

↕ *w ⊨A ↔w ⊨KPN(A) ↔w ⊨ Σ(KPN(A))*

Zbiór Σ(KPN(¬A)) nie jest spełnialny.

↕ Zbiór Σ nie jest spełnialny ↔ $\Sigma \vdash_{RZ} \square \leftrightarrow \Sigma(KPN(\neg A)) \vdash_{RZ} \square$

$\Sigma(KPN(\neg A)) \vdash_{RZ} \square$

Jeżeli zbiór Σ jest skończony, to istnieje algorytm sprawdzania czy $\vdash_{RZ} \square$

Na podstawie powyższego faktu dla skończonego zbioru klauzul istnieje algorytm sprawdzania czy da się z tego zbioru wyprowadzić klauzule pustą przy pomocy rezolucji zdaniowej

6. Podaj określenie języka pierwszego rzędu oraz termów, formuł atomowych, formuł, zdań, termu bazowego i atomu bazowego.

Język pierwszego rzędu: Symbole:

a) logiczne

- zmienne przedmiotowe: x,y,z,...,x',y',z',x1,x2,...
  VAR - zbiór wszystkich zmiennych przedmiotowych
- stałe logiczne: $\neg, \wedge, \vee, \rightarrow, \leftrightarrow, \forall, \exists$.
- symbole pomocnicze: (;);,;[;]

b) pozalogiczne

- symbole relacyjne (predykaty): P,Q,R,...
- symbole funkcyjne: f,g,h,...
- stałe przedmiotowe: a,b,c,...

Językiem pierwszego rzędu nazywamy układ:

L=(R1,...,Rn, f1,...,fm, a1,...,ak, p)

gdzie:

R - relacje

f - funkcje

a - stałe

p - funkcja, która dla każdego symbolu relacyjnego i funkcyjnego określa jego arność tzn. liczbę argumentów przy czym: $p(Ri) > 0$ dla $i=1,...,n$ oraz $p(fi)>0$ dla $i=1,...,m$

Termami języka L nazywamy wyrażenia określone przez następujące warunki:

- każda zmienna i stała przedmiotowa jest termem;
- jeżeli f jest symbolem funkcyjnym, p(f) = n oraz t1,...,tn są termami, to f(t1,...,tn) jest termem.

Formułami atomowymi (atomami) języka L nazywamy wyrażenia postaci: R(t1,...,tn), gdzie R jest symbolem relacyjnym, p(R) = n, a t1,...,tn są termami.

Formułami języka L nazywamy wyrażenia określone przez następujące warunki:

- Każda formuła atomowa jest formułą.
- Jeżeli A i B są formułami,
  to $(\neg A)$, $(A \wedge B)$, $(A \vee B)$, $(A \rightarrow B)$, $(A \leftrightarrow B)$,są formułami.
- Jeżeli A jest formułą i jest x zmienną przedmiotową, to wyrażenia $(\forall x A)$, $(\exists x A)$ są formułami.

Zdaniem (formułą domkniętą) nazywamy formułę bez zmiennych wolnych.

Termem bazowym nazywamy term nie zawierający zmiennych.

Atomem bazowym nazywamy formułę atomową nie zawierającą zmiennych.

7. Podaj określenia następujących pojęć w LPR: zdanie, domknięcie formuły, prawdziwość formuły w interpretacji.

Zdaniem (formułą domkniętą) nazywamy formułę bez zmiennych wolnych.

Niech A będzie formułą o zbiorze zmiennych wolnych V(A)={x1,…,xn}. Wtedy:

Domknięcie uniwersalne formuły A: $\forall x1 \ldots \forall xn\, A$.

Domknięcie egzystencjalne formuły A: $\exists x_1 \ldots \exists x_n$ A.

Domknięcie uniwersalne i egzystencjalne formuły A jest zdaniem.

Formuła A jest prawdziwa w interpretacji M (M |= A), jeżeli jej domknięcie uniwersalne jest prawdziwe w M.

8. Podaj definicję i własności podstawienia. Na czym polega złożenie podstawień? Podaj przykład złożenia podstawień.

Podstawieniem nazywamy funkcję σ : VAR→TER_L. Gdzie VAR - zbiór wszystkich zmiennych przedmiotowych, a TER_L zbiór wszystkich termów języka L. Stosujemy notację postfiksową: zamiast σ(x) piszemy xσ.

Własności:

ε–podstawienie identycznościowe: xε=x dla wszystkich ***x ϵVAR***

x(ση) = (xσ)η

γ(ση) = (γσ)η

εσ = σε = σ

Złożenie podstawień:

Niech: η = {*y*1/*s*1 ,…, *y*m/*s*m }, σ = {*x*1/*t*1 ,…, *x*n/*t*n }. Podstawienie ησ otrzymujemy ze zbioru {*y*1/*s*1σ ,…, *y*m/*s*mσ , *x*1/*t*1 ,…, *x*n/*t*n }

przez usunięcie:

- elementów *y*i / *s*i σ takich, że *y*i = *s*i σ
- elementów *x*j / *t*j takich, że *x*j ∫ {*y*1 ,…, *y*m }

Przykład

a) *Podstawianie:*

A: P(x,y,f(a,y)), σ = { x|b, y|a }

Aσ = P(x,y,f(a,y)) σ^ = P(xσ, yσ, f(a,y)σ^) = P(xσ, yσ, f(aσ^,yσ)) = P(b,a,f(a,b))

b) *Składanie podstawień:*

η = { x/f(y), y/z, u/g(u) }, σ = { x/a, y/b, z/y, u/c }

ησ = { x/f(y), y/z, u/g(u); x/a, y/b, z/y, u/c } = { x/f(b), y/y, u/g(c); x/a, y/b, z/y, u/c } = { x/f(b), u/g(c); y/b, z/y }

c) $\eta\sigma \neq \sigma\eta$

σ = { x/f(y), y/z }, η= { x/a, z/b }

ση = { x/f(y), y/b; x/a, z/b } = { x/f(y), y/b, z/b }

ησ = { x/a, z/b; x/f(y), y/z } = { x/a, z/b, y/z }

9. Podaj określenie interpretacji języka pierwszego rzędu. Przykład

Interpretacją języka L nazywamy układ:

$M=(|M|,R_1^M,\ldots, R_n^M; f_1^M,\ldots, f_m^M; a_1^M,\ldots, a_k^M)$

Gdzie:

- |M| -niepusty zbiór zwany dziedziną lub uniwersum interpretacji,
- R_n^M -n – argumentowa relacja na zbiorze |M|, gdzie n = ρ (R_i) tzn. $R_i^M \subset |M|^n = \{(u_1,\ldots,u_n) : u_1,\ldots,u_n \in |M|\}$,
- f_n^M - n-argumentowe działanie na zbiorze |M|, n = $\rho(f_i)$, tzn. $f^M_i : |M|^n \rightarrow |M|$,
- a_n^M – element zbioru |M|,

Dla każdego termu bazowego języka L, t∈TB_L tzn. termu nie zawierającego zmiennych, określamy $t^M \in |M|$ następująco:

a) a_i^M jest dane przez interpretacje M,

b) $(f_i(t_1,\ldots,t_n))^M = f_i^M(t_1^M,\ldots,t_n^M)$,

Przykład:

10. Podaj określenie wyrażenia prostego, przemianowania zmiennych, wariantu wyrażenia prostego. Każde pojęcie zilustruj przykładem.

Wyrażeniem prostym nazywamy termy i atomy (formuły atomowe).

Wprowadzamy oznaczenia:

E – wyrażenie proste

V(E) – zbiór zmiennych występujących w E

σ|V(E) – ograniczenie podstawienia σ do zbioru V(E);

jest to podstawienie η takie, że

xη=xσ dla x$\in$ V(E)

xη=x dla x$\notin$ V(E)

Przemianowaniem zmiennych nazywamy podstawienie σtakie, że

σ|V(E):V(E) $\rightarrow$(1-1) VAR

xη=x dla x$\notin$ V(E)

Przykład przemianowania Wyrażenia:

E=P(f(x,y),g(z))

F=P(f(y,x),g(u))

są swoimi wariantami ponieważ

η={x/y,y/x,z/u}

η|V(E) jest 1-1 (różnowartościowe)

Eη=F

oraz dla

σ={y/x,x/y,u/z}

σ|V(E) jest 1-1 (różnowartościowe)

Fσ=E

Def. Wariant wyrażenia prostego

Jeżeli σ jest przemianowaniem zmiennych w wyrażeniu E, to wyrażenie Eσ nazywamy wariantem wyrażenia E.

Przykład stworzenia wariantu dla przemianowania

E=P(f(x),y,z)

V(E)={x,y,z}

Podstawienie σ={x/y,y/x} daje Eσ=P(f(y),x,z)

11. Podaj określenie formuły preneksowej postaci normalniej. Jaki jest związek między formułą A i PNF(A)?

Def. Preneksowa postać normalna.

Formuła LPR znajduje się w *preneksowej postaci normalnej* (PNF), jeżeli jest postaci

Q1x1...Qkxk A // A – matryca Q… - prefix

gdzie Q1,Q2,...,Qk $\in$ {$\forall$,$\exists$}, zaś formuła *A* nie zawiera kwantyfikatorów.

Tw. 2.1.

Dla każdej formuły *A* istnieje formuła **B** w preneksowej postaci normalnej, która jest logicznie równoważna formule A, tzn każda interpretacja M spełnia formułę A$\leftrightarrow$B.

12. Podaj określenie formuły w postaci Skolema. Jaki jest związek między formułą A i SKOL(A)?

Def. Postać Skolema formuły LPR

Formuła A jest w postaci Skolema, gdy jest w preneksowej postaci normalnej, a jej prefiks nie zawiera kwantyfikatorów egzystencjalnych. Skolemowy odpowiednik formuły A będziemy oznaczać przez SKOL(A). Formuła A jest słabo równoważna formule SKOL(A), tzn. formuła A jest spełnialna wtw, gdy spełnialny jest jej odpowiednik Skolema, SKOL(A).

13. Co rozumiemy przez konkretyzację formuły(klauzuli) w LPR? Czym jest gr(∑) dla zbioru klauzul ∑? Podaj twierdzenie dotyczące ∑ i gr(∑).

Każde podstawienie formuły (klauzuli) nazywamy przykładem lub konkretyzacją tej formuły (klauzuli). Jeżeli podstawienie to jest podstawieniem bazowym, to i konkretyzację nazywamy bazową lub ustaloną.
∑ - zbiór klauzul
gr(∑) – zbiór wszystkich ustal. konkretyzacji klauzul ze zbioru ∑.

Dla każdego zbioru klauzul ∑ 2 warunki są równoważne
a) Zbiór ∑ jest spełnialny.
b) Zbiór gr(∑) jest spełnialny.

14. Uzasadnij algorytm dowodzenia praw LPR za pomocą rezolucji zdaniowej.

Algorytm 2.3. Sprawdzanie tautologiczności formuł LPR z wykorzystaniem rezolucji zdaniowej
Dane: formuła *A*
Wynik: odpowiedź *tak*, jeżeli *A* jest tautologią oraz *nie* w przeciwnym przypadku
(1) niech *B* będzie formułą PNF(~*A*)
(2) znajdujemy formułę SKOL(*B*)
(3) w formule SKOL(*B*) opuszczamy kwantyfikatory i jeżeli nie jest w postaci KPN, to sprowadzamy ją do tej postaci otrzymując formułę *C*.
(4) formułę *C* przedstawiamy w postaci klauzulowej Σ(*C*)
(5) szukamy derywacji klauzuli pustej ze zbioru *gr*(Σ(*C*))
(6) jeżeli klauzula pusta została otrzymana: odpowiedź *tak*
(7) jeżeli algorytm zatrzymał się ze względu na brak możliwości stosowania reguły rezolucji, odpowiedź *nie.*
Uwaga!! Algorytm 2.3
- daje odpowiedź *tak* wtedy i tylko wtedy, gdy formuła *A* jest tautologią LPR
- gdy formuła *A* nie jest tautologią LPR daje odpowiedź *nie* lub się zapętla.

Fakt 2.1 – patrz odp. 10
Tw. 2.1 – patrz odp. 14
Tw. 2.2 – patrz odp. 15
Def .Spełnialności – patrz odp. 2
Tw. 2.3 – patrz odp. 16
Tw. 2.4
Dla każdego zbioru zdań ustalonych Γ następujące warunki są równoważne:
a) zbiór Γ jest spełnialny w sensie (zakresie) LPR (tzn. przez interpretację)
b) zbiór Γ jest spełnialny w sensie rachunku zdań (tzn. przez wartościowanie)

Uzasadnienie
Formuła *A* jest tautologią LPR
↕ Fakt 2.1 -> ~A jest nie spełnialna
↕ Tw. 2.1b -> PNF(~A) nie jest spełnialna
↕ Tw. 2.2 - > SKOL(PNF(~*A*)) nie jest spełnialna // oznacz. ją B
↕ def. Spełnialności -> C = matryca (*B*) nie jest spełnialna
↕ Zbiór klauzul Σ(*C*) nie jest spełnialny
↕ Zbiór klauzul *gr*(Σ(*C*)) nie jest spełnialny
↕ Tw. 2.4 -> Istnieje refutacja zbioru *gr*(Σ(*C*)), tzn. istnieje wyprowadzenie klauzuli pustej ze zbioru *gr*(Σ(*C*)), za pomocą rezolucji zdaniowej.

15. Podaj określenie zbioru niezgodności dla zbioru wyrażeń prostych i zilustruj przykładem.

Def. Zbiór niezgodności

Niech S będzie skończonym zbiorem wyrażeń zawierających więcej niż jedno wyrażenie:

$$S = \{E1, \ldots, En\}, n \geq 2$$

Zbiór niezgodności D dla zboru S wyznaczamy następująco . Znajdujemy pierwszą od lewej pozycję, na której przynajmniej 2 wyrażenia zbioru S mają różne symbole. Na tej pozycji znajduje się stała, zmienna, symbol funkcyjny lub symbol relacyjny. Z każdego wyrażenia do zbioru D włączamy to podwyrażenie, które zaczyna się od znalezionej pozycji.

a) S = { P(x,f(y)), P(x,g(z)) } D = { f(y), g(z) }

b) S = { P(f(x),h(y),e), P(f(x),z,e), P(f(x),h(y), b) } D = { h(y), z }

16. Podaj algorytm wyznaczania najbardziej ogólnego unifikatora zbioru wyrażeń oraz twierdzenie o unifikacji.

Algor. 3.1 Algorytm unifikacji – procedura poszukiwania MGU

Wejście: skończony zbiór wyrażeń prostych S

Wyjście: MGU dla S, jeśli S jest zunifikowany, NIE w przeciwnym wypadku.

1) $k_i = 0$, $\sigma_0 = \varepsilon$

2) Wyznaczamy zbiór Sσk

Jeżeli $S\sigma_k$ jest jednoelementowy, to STOP;

Wyjście: σk. W przeciwnym wypadku wyznaczamy zbiór niezgodności D_k dla $S\sigma_k$

3) Wybieramy w zbiorze D_k parę x, t, taką, że x ⅄ VAR, t ⅄ TER oraz x ⅄ V(t), przyjmijmy $\sigma k+1 = \sigma_k\{x/t\}$, wracamy do punku 2. Algorytm przy k=k+1, Jeżeli nie ma takiej pary, to STOP, wyjście nie.

Twierdzenie 3.1 O Unifikacji.

Jeżeli zbiór S jest unifikowany, to algorytm kończy pracę z wyjściem będącym MGU zbioru S. Jeżeli S nie jest unifikowany, to

algorytm kończy pracę z wyjściem nie.

17. Podaj określenie klauzuli definitywnej, programu definitywnego, klauzuli celu, klauzuli Horna.

Def. Klauzula definitywna

Klauzulą definitywną nazywamy klauzulę zawierającą dokładnie jeden literał pozytywny, co zapisujemy:

B1, ..., Bn→A, gdzie B1, ..., Bn = B1 ⊗ B2 ⊗ ... ⊗ Bn

w szczególności:

n = 0

→ A – klauzula jednostkowa – Fakt (1→A)

n > 0

B1, ..., Bn → A – klauzula nie jednostkowa: reguła; regułę w praktyce zapisujemy jako:

A ← B1, ..., Bn, gdzie A to głowa, B1, ..., Bn, - body – ciało reguły.

Def. Program definitywny

Programem definitywnym nazywamy dowolny skończony zbiór klauzul definitywnych, czyli zbiór formuł postaci:

$\forall(B_1 \wedge \ldots \wedge B_n \rightarrow A)$.

Def. Klauzula celu.

Celem nazywamy klauzulę nie zawierającą literałów pozytywnych (m=0)

$\leftarrow B_1,\ldots,B_n$

która jest równoważna formule

∀(<–(B1 ⊗ .... ⊗ Bn)) na podstawie tautologii KRZ: (p→0)→¬p

Def. Klauzula Horna.

Klauzulą Horna nazywamy klauzulę zawierającą co najwyżej jeden literał pozytywny, tzn klauzulę definitywną lub klauzulę celu.

18. Omów strategię Prologu

Fakt 2.2

2 następujące warunki są równoważne:

- formuła A wynika logicznie ze zbioru formuł Γ w LPR
- zbiór $\Gamma \cup (\neg A)$ nie jest spełnialny.

Strategia Prologu:

Niech P będzie programem definitywnym, a G-celem

Bierzemy pod uwagę zbiór P ∪ {G}

Jeżeli wykażemy, że zbiór P ∪ {G} nie jest spełnialny, to na podstawie FAKTU 2.2. otrzymamy, że formuła ¬G wynika logicznie ze zbioru formuł tworzących program P.

Cel G ma postać

$/\therefore(\sim(B1 \otimes .... \otimes Bn)) <=> \sim\therefore/(B1 \otimes ... \otimes Bn)$

zatem formuła

$\therefore/x1 ... \therefore/xk\ (B1 \otimes ... \otimes Bn)$

wynika logicznie z P, gdzie $x_1,\ldots,x_k$ są wszystkimi zmiennymi występującymi w G. Tym samym znajdziemy obiekty $x_1,\ldots,x_k$ spełniające cel G.

19. Podaj regułę rezolucji liniowej i derywacji liniowej.

Def. Reguła rezolucji liniowej

Reguła rezolucji liniowej ma postać

←A1,…,Am,…,An – cel G

B←B1,…,Bk – wariant klauzuli C program P

$\leftarrow (A_1, ..., A_{m-1}, B_1, ..., B_k, A_{m+1}, ..., A_n)\sigma$ - nowy cel.

gdzie σ jest MGU zbioru $\{A_m,B\}$, tzn: $A_{m\sigma} = B_\sigma$

□

Def. Derywacja liniowa

Niech P będzie programem definitywnym, a G celem.

Derywacja liniowa dla P ∪ {G} nazywamy skończony lub nieskończony ciąg którego wyrazami są:

(G_n,C_n,σ_n), $1<=n<=p$ lub $1<=n$ spełniający warunki:

a) Dla każdego n, C_n jest wariantem klauzuli programu P nie zawierającym zmiennych występujących w $G_0,G_1,\ldots,G_{n-1}$

b) Dla każdego n, G_n jest rezolwentą liniową celu G_{n-1} oraz klauzuli C_n otrzymamy za pomocą podstawienia σ_n

20. Podaj określenie refutacji liniowej dla programu P i celu G oraz odpowiedzi obliczonej.

Liniową refutacją dla $P \cup \{G\}$ nazywamy skończoną liniową derywację dla $P \cup \{G\}$, której ostatni cel jest klauzulą pustą.

Niech (G_n, C_n, σ_n), $1 \le n \le p$, będzie liniową refutacją dla $P \cup \{G\}$. Odpowiedzią obliczoną dla $P \cup \{G\}$ wyznaczoną przez tę refutację nazywamy podstawienie $\sigma_1 \ldots \sigma_n$ ograniczone do zmiennych występujących w G. tzn. $(\sigma_1 \ldots \sigma_n)|V(G)$.

21. Podaj określenie odpowiedzi obliczonej i poprawnej dla $P \cup \{G\}$, gdzie P jest programem definitywnym, a G – celem. Sformułuj twierdzenie o poprawności rezolucji liniowej. Podaj silne twierdzenie o pełności rezolucji liniowej.

Niech (G_1, C_1, σ_1), $1 \leq n \leq p$, będzie liniową refutacją dla $P \cup \{G\}$. Odpowiedzią obliczoną dla $P \cup \{G\}$ wyznaczoną przez tę refutację nazywamy podstawienie $\sigma_1 \ldots \sigma_n$ ograniczone do zmiennych występujących w G. tzn. $(\sigma_1 \ldots \sigma_n) | V(G)$.

Podstawienie σ nazywamy odpowiedzią poprawną dla $P \cup \{G\}$, gdzie G: ←B1,…,Bn, jeżeli formuła $(B1 \wedge \ldots \wedge Bn)\sigma$ wynika logicznie w LPR z programu P, co zapisujemy P $|=_{LPR}$ $(B1 \wedge \ldots \wedge Bn)\sigma$. Przyjmujemy dodatkowo, że dziedzina podstawienia σ jest zawarta w $V(B1 \wedge \ldots \wedge Bn)$.

Tw. o poprawności rezolucji liniowej.
Każda odpowiedź obliczona dla $P \cup \{G\}$ jest odpowiedzią poprawną dla $P \cup \{G\}$, tzn. jeżeli $\sigma = (\sigma_1 \ldots \sigma_k) | V(G)$ jest odpowiedzią obliczoną dla $P \cup \{G\}$, gdzie G: ←B1,…,Bn, to P $|=_{LPR}$ $(B1 \wedge \ldots \wedge Bn)\sigma$.

Silne twierdzenie o pełności rezolucji liniowej.
Dla każdej odpowiedzi poprawnej θ dla $P \cup \{G\}$ istnieje odpowiedź obliczona σ dla $P \cup \{G\}$ oraz podstawienie γ takie, że θ = (σγ) | V(G).

22. Podaj określenie SLD-drzewa dla $P \cup \{G\}$ zgodnego z regułą selekcji R.

SLD-drzewo dla $P \cup \{G\}$ zgodne z R jest drzewem skończonym lub nieskończonym, którego wierzchołki etykietowane są celami i spełniającym następujące warunki:
a) korzeń drzewa jest etykietowany celem G,
b) dla dowolnego wierzchołka etykietowanego celem G' następniki tego wierzchołka są etykietowane kolejnymi celami, które powstają z G' przez uzgodnienie wybranego zgodnie z R atomu z kolejnymi klauzulami programu,
a) gałęzie pełnego SLD-drzewa dla $P \cup \{G\}$ reprezentują wszystkie derywacje dla $P \cup \{G\}$ zgodne z R, przy czym: gałęzie nieskończone reprezentują nieskończoną derywację (procedura się pętli), gałęzie skończone to: gałęzie udane (sukcesu), gałęzie chybione (porażki).

23. Podaj określenie reguły selekcji i sformułuj twierdzenie o nieistotności reguł selekcji.

Regułą selekcji nazywamy funkcję R, która każdej skończonej derywacji liniowej (Gn,Cn,σn), $1 \leq n \leq p$ takiej, że Gp = □ przyporządkowuje liczbę naturalną $1 \leq m \leq k$, gdzie k jest długością celu Gk.

Tw. o nieistotności reguł selekcji.
Niezależnie od wyboru reguły selekcji otrzymamy te same odpowiedzi obliczone dla $P \cup \{G\}$ z dokładnością do wariantu, tzn. jeżeli R1 i R2 są regułami selekcji i σ1 jest odpowiedzią obliczoną dla $P \cup \{G\}$ zgodną z R1 i σ2 jest odpowiedzią obliczoną zgodną z R2, to Gσ1 i Gσ2 są swoimi wariantami.

24. Podaj podstawowe cechy prologu jako implementacji SLD rezolucji. Objaśnij użyte pojęcia.

W prologu stosujemy:
- SLD-rezolucję,
- regułę selekcji „left first",
- strategię przeszukiwania drzewa „depth first",
- niezmienny porządek klauzul w procedurach,

Prolog jako metoda dowodzenia twierdzeń nie jest pewny, tzn. nie spełnia twierdzenia o pełności.
Left first – reguła selekcji, w której R() = 1, tzn. w danym celu wybierany jest zawsze podlec pierwszy z lewej.

Depth first – strategia przeszukiwania drzewa – przeszukiwanie lewymi gałęziami; dla pewnych drzew strategia ta może nie doprowadzić do rozwiązania, ale jest łatwa w realizacji.

25. Termy rachunku lambda

Zbiór Λ – terów w rachunku lambda konstruuje się indukcyjnie jak zbiór słów nad alfabetem

Var u {(,), λ} gdzie Var to przeliczalny zbiór zmiennych. Do zbioru Λ termów należą tylko wyrażenia skonstruowane w poniższy sposób

- Jeżeli x ϵ Var, to x ϵ Λ,
- Jeżeli M ϵ Λ, x ϵVar, to (λ x M) ϵ Λ. Ten sposób tworzenia termu nazywamy abstrakcją.
- Jeżeli M ϵ Λ i N ϵ Λ, to (MN) ϵ Λ Tę metodę tworzenia termów nazywamy metodą aplikacji

Termy rachunku λ to procedury, których parametry sprecyzowane są przez λ abstrakcje

$$\lambda \underbrace{x_1 \ldots x_n}_{\text{parametry}} . \underbrace{M}_{\text{ciało procedury}}$$

λ x1…xn. M przy czym { x1…xn} to parametry a M to ciało procedury

Nawiasy są często pomijane według zasady łączności lewostronnej: na przykład $MNPQ \equiv (((MN)P)Q)$. Wieloargumentowe funkcje są reprezentowane przez funkcje jednoargumentowe, których argumentami są funkcje. Zatem to, co zwykle jest zapisywane w postaci $F(N, M)$, w rachunku lambda zapisuje się $((FN)M) \equiv FNM$. Powtórzenia symbolu λ pomijane są według zasady łączności prawostronnej, tzn. $\lambda xyz.M \equiv (\lambda x.(\lambda y.(\lambda z.M)))$.

Zbiór zmiennych wolnych termu M, oznaczany przez $FV(M)$, zdefiniowany jest, w zależności od budowy termu M, w następujący sposób:

- Jeżeli M jest pojedynczą zmienną, tzn. $M \equiv x$, to zmienna ta jest wolna, a zatem $FV(M) = \{x\}$.
- Jeżeli M jest postaci $\lambda x.M'$, to λx wiąże wszystkie wolne wystąpienia zmiennej x w termie M', a zatem $FV(M) = FV(M') - \{x\}$.
- Jeżeli M jest postaci PQ, to zmiennymi wolnymi termu M są wszystkie zmienne wolne występujące w termach P i Q, czyli $FV(M) = FV(P) \cup FV(Q)$.

Zmienna jest *zmienną wolną* termu M, jeżeli należy do zbioru $FV(M)$, a *zmienną związaną* w przeciwnym wypadku.

Term M jest *domknięty* wtedy i tylko wtedy, gdy $FV(M) = \emptyset$.

Przykład

- W termie $M = y(\lambda x.x)$ zmienna y jest zmienną wolną, a x zmienną związaną. W termie $\lambda y.M$ nie ma zmiennych wolnych, jest to więc term domknięty.

- W termie $M = x(\lambda x.x)$ tylko pierwsze wystąpienie zmiennej x jest wolne, chociaż $FV(M) = \{x\}$. Oczywiście term M nie jest termem domkniętym.

26. Alfa-konwersja. Definicja, przykłady

α konwersja – zamiana nazw zmiennych związanych

λx.M→ $^{\alpha}$λ.y.M[x|y], gdzie y ∉ FV(M),M[x|y]

wynik podstawienia zmiennej y za każde wolne wystąpienie zmiennej x w M

Można napisać że $T_1 =_{\alpha} T_2$ aby powiedzieć że termy T_1 i T_2 są równe modulo α konwersja tzn że mogą być zredukowane do tego samego termu poprzez zastosowanie skończonej liczby α konwersji.

PRZYKŁADY:

λx.x= $^{\alpha}$ λ y.y

λx.z= $^{\alpha}$ λy.yz

λx.xy≠ $^{\alpha}$ λx.xz

27.Beta-redukcja. Definicja, przykłady

β redukcja – obliczenia wykonane przez procedurę symulowane są w rachunku lambda przez proces zwany β redukcją. Aplikujemy procedurę λx.M→ $^{\beta}$M[xIN] w taki sposób aby każda zmienna wolna w N pozostała wolna po podstawieniu Ewentualnie w razie potrzeby stosujemy najpierw α konwersję.

Można napisać że $T_1 =_{\beta} T_2$ aby powiedzieć że termy T_1 i T_2 są równe modulo α konwersja tzn że mogą być zredukowane do tego samego termu poprzez zastosowanie skończonej liczby β redukcji.

Aplikację PQ nazywamy β-redeksem, jeżeli P jest w postaci λ abstrakcji. Term M jest wpostaci normalnej jeżeli nie zawiera β redeksów

PRZYKŁADY:

(λxy.xy)y= $_{\alpha}$ (λxz.xz)y= $_{\beta}$ λz.yz

(λxy.xy)y≠ $_{\beta}$ λy.yy

28. Postać normalna termu w rachunku lambda

Aplikację PQ nazywamy β-*redeksem*, jeżeli term P jest w postaci λ-abstrakcji. Term M jest w *postaci normalnej,* jeżeli nie zawiera β-redeksów, tzn. żaden podterm termu M nie jest β-redeksem.

Term T jest *normalizowalny*, jeżeli można go zredukować do postaci normalnej, a jest *silnie normalizowalny*, jeżeli każda droga redukcji prowadzi do postaci normalnej. Beztypowy rachunek lambda nie ma żadnej z tych własności. Wprowadźmy oznaczenie: $\Delta = \lambda x.xx$. Najprostszym przykładem termu nie posiadającego postaci normalnej jest $\Delta\Delta$:

$$\Delta\Delta = (\lambda x.xx)\Delta =_{\beta} \Delta\Delta.$$

Przykładem termu, w którym nie każda droga redukcji prowadzi do postaci normalnej, jest:

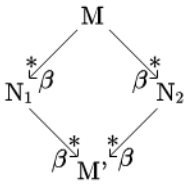
$$\overbrace{(\lambda x.y)(\underbrace{\Delta\Delta}_{(1)})}^{(2)} =_\beta (\lambda x.y)(\Delta\Delta) \quad =_\beta \quad \ldots \qquad \text{| redukując (1)}$$

$$=_\beta y \qquad \text{| redukując (2)}$$

Istnieją strategie redukcji, które zawsze doprowadzają do postaci normalnej, oczywiście gdy postać normalna istnieje. Taką strategią jest na przykład metoda redukowania zawsze najbardziej lewego redeksu.

29. Własność Churcha-Rossera

Jeżeli term M poprzez pewne ciągi β redukcji, redukuje się do termów N_1 i N_2 to istnieje term M tai że zarówno N_1 jak i N_2 można zredukować do M (poprzez ciąg β redukcji)



Z powyższej własności wynika że postać normalna (o ile istnieje jest unikalna z dokładnością do α konwersji

30. Reprezentacja liczb naturalnych w rachunku lambda.

Popatrzmy na liczbę naturalną $n$jak na procedurę, która n-krotnie stosuje następnik do zera. Oczywiste jest, że taka procedura w sposób jednoznaczny określa liczbę, i na odwrót. Mając dany przepis wiemy, jaką liczbę naturalną liczy, a mając liczbę naturalną doskonale wiemy, który przepis zastosować.

- Term $\lambda sx.\ x$reprezentuje liczbę naturalną 0.

  $$\lambda sx.\ \underbrace{s(s(\ldots(s\ x)\ldots))}_{\text{n razy}}$$

- Term reprezentuje liczbę naturalną n.

Term $\lambda sx.\ x$to po prostu algorytm: „weź następnik s, weź zero xi zwróć zero”; natomiast term $\lambda sx.\ \underbrace{s(s(\ldots(s\ x)\ldots))}_{\text{n razy}}$ to algorytm: „weź następnik s, weź zero x, zastosuj n-krotnie następnik do zera i zwróć wynik”. Zatem liczba naturalna w naszym rozumieniu to procedura, która ma dwa parametry: s (następnik) oraz x(zero).

Oczywiście powyższe termy podane są z dokładnością do α-konwersji. Np. term $\lambda ty.t(ty)$reprezentuje 2, gdyż $\lambda ty.t(ty) =_\alpha \lambda sx.s(sx)$.

Mówimy, że *term* E*rachunku lambda reprezentuje funkcję* na liczbach naturalnych $e : \mathbb{N}^k \to \mathbb{N}$wtedy i tylko wtedy, gdy $E\underline{n_1}\ldots\underline{n_k} =_{\alpha\beta} \underline{e(n_1,\ldots,n_k)}$, gdzie $\underline{n}$oznacza reprezentację liczby nw rachunku lambda.