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

Def. Wartościowanie.
1. Wartości logiczne : 1 (prawda), 0 (fałsz)
2. *Wartościowaniem* nazywamy funkcję *w*: *V*->{0, 1};
gdzie *V* = {p, q, r,...,p1,...,p',...} - nieskończony, ale przeliczalny zbiór *zmiennych zdaniowych*
3. *Funkcją logiczną* nazywamy funkcję *f*: {0, 1}^n -> {0, 1}.
Określamy funkcje: *f*⌐:{0,1}->{0,1} i *f*∧, *f*v , *f*->, *f*<-> :{0,1}^2->{0,1}
następująco:
x, y ,*f*⌐(*x*), *f*∧(*x, y*) ,*f*v(*x, y*) ,*f*->(*x, y*) *f*<->(*x, y*)
1,1,0,0
1,0,1,0
0,0,1,1
1,0,0,0
1,1,1,0
1,0,1,1
1,0,0,1
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(*A*v*B*) = *f*v(*ŵ*(*A*),*ŵ*(*B*))
ŵ(*A*->*B*) = *f*->(*ŵ*(*A*),*ŵ*(*B*))
ŵ(*A*<->*B*) = *f*<->(*ŵ*(*A*),*ŵ*(*B*))
Przykład 1.2. Wartościowanie formuł.
Formuła *A*: *p* -> *q* v *r*. Wartościowanie: *w*(*p*)=1, *w*(*q*)=0, *w*(*r*)=1.
ŵ(*A*) = *ŵ*(*p*-> *q* v *r*) = *f*-> (*ŵ*(*p*), *ŵ*(*q* v *r*)) = *f*->(*ŵ*(*p*), *f* v (*ŵ* (*q*), *ŵ* (*r*))) = *f*->(*w*(*p*),*f* v (*w*(*q*), *w*(*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. Wykaż, że następujące warunki są równoważne:

Def. Spełnianie formuły i zbioru formuł przez wartościowanie.
1. Wartościowanie *w spełnia* formułę *A* (oznaczenie: *w* |= *A*), jeżeli *ŵ* (*A*) = 1.
2. Wartościowanie *w spełnia* zbiór formuł Γ ⊂ *FOR* (ozn. *w* |= Γ), jeżeli *ŵ* (*A*) = 1 dla każdej formuły *A* ∈ Γ.
Def. Spełnialność formuły i zbioru formuł.
1. Formułę *A* nazywamy *spełnialną*, jeżeli istnieje wartościowanie *w* takie, że *w* |= *A*.
2. Zbiór formuł Γ ⊂ *FOR* nazywamy *spełnialnym*, jeżeli istnieje wartościowanie *w* takie, że *w* |= Γ.

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.

Def. Wynikanie logiczne.
Formuła *A wynika* ze zbioru formuł Γ w KRZ, jeżeli dla każdego wartościowania *w* zachodzi warunek: jeżeli w |= Γ, to w |= A.
Def. Logiczna reguła wnioskowania.
Jeżeli ze zbioru formuł {*A*1 , ..., *A*n}, które będziemy nazywać *przesłankami* wynika formuła A, którą będziemy nazywać *wnioskiem*, to schemat
A1
.
An
___
A nazywamy *logiczną regułą wnioskowania.*
Reguła rezolucji zdaniowej
{*L1, L2, ..., Lm, p*}
{*L1', L2',…,Ln',*⌐ *p*}
____
{*L1, L2, ..., Lm,L1',L2', ..., Ln'*} n,m >= 0
Dowód
Zał., że istnieje wartościowanie *w* spełniające obie przesłanki: *w* |= {*L*1, *L*2,..., *L*m, *p*} *w* |= {*L*1', *L*2',..., *L*n', ⌐*p*}
Rozważmy dwa przypadki:
1. *w*(*p*)=1, wtedy *w*(⌐*p*)=0, tzn *w*| nie=(⌐*p*), stąd w(*L*'i)=1 dla pewnego 1=< i =<n. Zatem *w* spełnia wniosek *w* |= {*L*1', *L*2',..., *L*n' }
2. w(p)=0, wtedy istnieje 1=< j =<m takie, że w(Lj)=1. Zatem w spełnia wniosek *w* |= {*L*1, *L*2,..., *L*m }

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

Def. Derywacja klauzuli ze zbioru klauzul.
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*)∈Σ;
3) *f* (Λ)= *A* ;
4) dla każdego wierzchołka *x* drzewa *X* nie będącego liściem *f* (*x*) jest rezolwentą etykiet bezpośrednich potomków wierzchołka *x*.
Przykład
1) Σ={{*p*, *q*},{¬*p*, *q*}}
Σ |–RZ {*q*}: // derywacja {q} z Σ; poniżej mamy drzewo etykietowane, gdzie korzeń drzewa to {q};
Przykład
Niech Σ = { {*p*, *q*, *r*}, {*p*, *q*, ¬*r* }, {¬*q*, *r*}, {*q*, *r*} }. Przykład derywacji (wyprowadzania) Σ |–RZ (*p*, *r*): narysujdrzewo
Def. Refutacja zbioru klauzul.
Derywację klauzuli pustej ze zbioru klauzul Σ nazywamy ***refutacją*** zbioru Σ.
Przykład
Σ={{*p*, *q*},{¬*p*, *q*},{*p*, ¬*q*},{¬*p*, ¬*q*}}
narysujdrzewo

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

Algorytm 1.2. Sprawdzanie tautologiczności formuł KRZ za pomocą rezolucji zdaniowej.
Dane : formuła *A*
Wynik : odpowiedź *tak*, jeżeli formuła *A* jest tautologia KRZ, odpowiedź *nie* w przeciwnym przypadku
(1) bierzemy formułę *B* identyczną z formułą (¬*A*).
(2) sprowadzamy formułę *B* do koniunkcyjnej postaci normalnej KPN(*B*)
(3) formułę 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, odpowiedź *tak*
(6) jeżeli algorytm zatrzymał się ze względu na brak możliwości stosowania reguły rezolucji, odpowiedź *nie*.
Fakt 1.1 – patrzy odp. 2
Fakt 1.8
Następujące warunki są równoważne:
(i) w |= A
(ii) w |= KPN(A)
(iii) w |= ∑(KPN(A))
Tw. 1.3 O zgodności i pełności rezolucji zdaniowej
Następujące warunki są równoważne:
1) zbiór ∑ nie jest spełnialny
2) ∑ |-RZ
Uzasadnienie
A jest tautologią KRZ
□ fakt 1.1
¬A jest nie spełnialna
□ fakt 1.8
KPN(¬A) nie jest spełnialna
□ fakt 1.8
Σ(KPN(¬*A*)) nie jest spełnialna
□ tw 1.3 o zgodności i pełności rezolucji zdaniowej
Σ(KPN(¬*A*)) |–RZ □

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

Def. Język pierwszego rzędu.
Symbole
a) logiczne
– zmienne przedmiotowe: *x*, *y*, *z*,..., *x*',..., *x*1,...
VAR – zbiór wszystkich zmiennych przedmiotowych
– stałe logiczne: ¬, ∧, ∨, ->, ↔, ∀, ∃
– 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; ρ)
gdzie: R – relacje, f – funkcje, a – stałe, ρ jest funkcją , która dla każdego symbolu relacyjnego i funkcyjnego określa jego arność, tzn. liczbę argumentów, przy czym ρ(Ri)> 0, i=1,2,...,n oraz ρ(fi)> 0, i=1,2,...,m.
Def. Termy.
Termami języka *L* nazywamy wyrażenia określone przez następujące warunki:
– każda zmienna i stała przedmiotowa jest termem,
– jeżeli *f* jest symbolem funkcyjnym, ρ (*f*) = n oraz *t*1,..., *t*n są termami, to *f* (*t*1,..., *t*n) jest termem.
TERL– zbiór wszystkich termów języka *L*.
Def. Formuły atomowe.
Formułami atomowymi (*atomami*) języka *L* nazywamy wyrażenia postaci *R* (*t*1,..., *t*n), gdzie *R* jest symbolem relacyjnym, ρ (*R*) = n, a *t*1,..., *t*n są termami.
Def. Formuły
Formułami języka *L* nazywamy wyrażenia określone przez następujące warunki:
– każda formuła atomowa jest formułą,
– jeżeli *A* i *B* są formułami, to wyrażenia (¬*A*), (*A* ∧ *B*), (*A* ∨ *B*), (*A* □ *B*), (*A* ↔ *B*) są formułami,
– jeżeli *A* jest formułą i *x* jest zmienną przedmiotową, to wyrażenia (∀*x A*), (∃*x A*) są formułami.
*FOR*L – zbiór wszystkich formuł języka *L*.
Def. Zdania.
Zdaniem (*formułą domkniętą*) nazywamy formułę bez zmiennych wolnych.
SENL – zbiór wszystkich zdań języka L.
Niech
V(*A*)– zbiór wszystkich zmiennych wolnych w formule *A*.
V(*t*)– zbiór wszystkich zmiennych występujących w termie *t*.
Def. Termy bazowe.
Termem bazowym nazywamy term nie zawierający zmiennych.
TBL – zbiór wszystkich termów bazowych języka *L*.
Def. Atomy bazowe.
Atomem bazowym nazywamy formułę atomową nie zawierającą zmiennych.
ABL – zbiór wszystkich atomów bazowych języka *L*.

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

Def. Zdania.
Zdaniem (*formułą domkniętą*) nazywamy formułę bez zmiennych wolnych.
SENL– zbiór wszystkich zdań języka L.
Def. Domknięcie formuły.
Niech *A* będzie formułą o zbiorze zmiennych wolnych *V* (*A*) = {*x*1,..., *x*n }.
Domknięcie uniwersalne formuły *A*: ∀*x*1...∀*x*n *A*.
Domknięcie egzystencjalne formuły *A*: ∃*x*1...∃*x*n *A*.
Domknięcie uniwersalne i egzystencjalne formuły *A* jest zdaniem.
Def. *M* |= *A*. A – formuła.
Formuła *A* jest prawdziwa w interpretacji *M* lub inaczej: interpretacja *M* spełnia formułę *A* (*M* |= *A*), jeżeli jej domknięcie uniwersalne jest prawdziwe w *M*.
Formuła *A* jest spełnialna jeżeli istnieje interpretacja, która ją spełnia..

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

Def. Podstawienie.
Podstawieniem nazywamy funkcję σ: *VAR* □ *TERL*. Stosujemy notację postfiksową: zamiast σ (*x*) piszemy *x*σ.
Dziedzina podstawienia: *DOM* (σ) = {*x*∈*VAR*: *x*σ ≠ *x* }.
ε – podstawienie identycznościowe: *x*ε = *x* dla wszystkich *x*∈*VAR*, *DOM*(ε) = ∅.
Podstawienie naz. *skończonym*, jeśli jego dziedzina jest skończona, tzn. *DOM* (σ) = {*x*1,..., *x*n }. Jeżeli dla tego podstawienia *x*iσ = *t*i , i = 1, 2,..., n , to podstawienie to przedstawiamy jako zbiór przypisań: σ = {*x*1/t1,..., *x*n/*t*n }.
Podstawienie złożone:
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 xj / *t*j takich, że xj ∈ {*y*1 ,…, *y*m }
Własności podstawienia:
x (ση) = (xσ)η
γ(ση) = (γσ)η
εσ = σε = σ
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) ησ ≠ ση
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.

Niech L = (R1, ..., Rn; f1, ..., fm; a1, ..., ak; ρ) będzie językiem pierwszego rzędu.
Def. Interpretacja języka.
Interpretacją języka L nazywamy układ
M = (|M|; R^M1, ..., R^Mn;
F^M1, ..., f^Mm; a^M1, ..., a^Mk) gdzie
|M| – niepusty zbiór zwany dziedziną lub uniwersum interpretacji,
R^Mi – *n*-argumentowa relacja na zbiorze |M|, n = ρ(Ri), tzn.
R^Mi⊂ |M|n= { (u1, ..., un) : u1, ..., un ∈ |M|},
f^Mi – *n*-argumentowe działanie na zbiorze |M|, n = ρ(fi), tzn.
f^Mi : |M|n-> |M|,
a^Mi - element zbioru |M|.
Dla każdego termu bazowego języka *L*, *t*∈*TBl,* tzn. termu nie zawierającego zmiennych, określamy t^M ∈ |M| następująco:
a) *a^Mi* jest dane przez interpretację *M*,
b) *(fi(t1,...,tn))^M =*
f^Mi(t^M1,...,t^Mn).

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

Def. Wyrażenia proste
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 xÎV(E)
xη=x dla xÏV(E)
Def. Przemianowanie zmiennych
Przemianowaniem zmiennych nazywamy podstawienie σ takie, że
σ|V(E):V(E) →(1-1) VAR
xη=x dla xÏV(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 wynikania logicznego w LPR i sformułuj warunki konsekwencji, jakie spełnia wynikanie logiczne.

Def. Wynikanie logiczne.
Formuła *A wynika logicznie* ze zbioru formuł G , jeżeli formuła *A* jest prawdziwa we wszystkich modelach dla zbioru G .
Wprowadzamy oznaczenie Γ|=LPR A – formuła A wynika logicznie ze zbioru Γ na gruncie LPR.
Wynikanie logiczne na gruncie LPR spełnia warunki konsekwencji tzn.:
(C1) jeżeli AÎΓ, to Γ|=LPR A.
(C2) jeżeli ΓÌΔ oraz Γ|=LPR A to Δ |=LPR A
(C3) jeżeli Γ|=LPR Ai dla wszystkich 1<=i<=n oraz A1,…,An |=LPR A to Γ|=LPR A

12. Podaj określenie formuły w preneksowej postaci normalnej. 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 Î {",$}, 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* <->*B* .

13. 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* (SKOL(*A*)), gdy jest w preneksowej postaci normalnej, a jej prefiks nie zawiera kwantyfikatorów egzystencjalnych.
Formuła *A* jest słabo równoważna formule SKOL(*A*), tzn. formuła *A* jest spełnialna wtedy i tylko wtedy, gdy spełnialny jest jej odpowiednik Skolema, SKOL(*A*).

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

Def. Konkretyzacja formuły.
Każde podstawienie formuły (klauzuli) nazywamy *przykładem* lub ***konkretyzacją*** tej formuły (klauzuli).
Jeżeli podstawienie to jest podstawieniem bazowym, to i konkretyzację nazywamy bazową lub ustaloną.
Niech å– zbiór klauzul
gr(å) – zbiór wszystkich ustalonych konkretyzacji klauzul ze zbioru S
Tw. 2.3.
Dla każdego zbioru klauzul S następujące warunki są równoważne:
a) zbiór å jest spełnialny,
b) zbiór gr(å) jest spełnialny.
Dowód
(Þ)
Załóżmy, że S jest spełnialny, tzn. istnieje struktura *M* taka, że ∀A∈∑ M |= A
Elementami zbioru *gr(*S*)* są klauzule postaci A[x1/t1, ... , xn/tn] dla *A*∈Σ.
Na podstawie prawa LPR ∀x A>A[x/t] mamy: ∀x A -> A[x1/t1, ... , xn/tn] jest tautologią LPR.
Zatem ∀A∈∑ M |= A[x1/t1, ..., xn/tn] czyli *M* |= *gr(*S*)* jest spełnialny.
P(x, y) : P(a, a), P(a, b), P(b, a), P(b,b) – wszystkie konkretyzacje języka
P(f(a), a), P(f(f(a)), a), ... – bardzo dużo konkretyzacji gdy w języku jest dostępny symbol funkcyjny.
(⇐)
Zakładamy, że *gr*(S) jest spełnialny.
Istnieje struktura *M* taka, że *M* |= *gr*(S). Tworzymy strukturę *N* następująco:
za nośnik tej struktury przyjmujemy | *N* | - zbiór wszystkich ustalonych termów języka.
a^N = a
f^N (t1, ..., tn) = f(t1, ..., tn)
P^N (t1, ..., tn) = P^M (t1^M, ..., tn^M)
Stąd dla każdego zdania ustalonego *A* mamy: *M* |= *A* <-> *N*|=*A* , czyli *N* |= *gr*(S). Zatem S jest spełnialny.

15. Uzasadnij algorytm dowodzenia praw w 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. 2.1
PNF(¬A) nie jest spełnialna
□ Tw. 2.2
SKOL(PNF(¬*A*)) nie jest spełnialna // oznaczamy ją B
□ def. Spełnialności
C = matryca (*B*) nie jest spełnialna
□
Zbiór klauzul Σ(*C*) nie jest spełnialny
□ Tw. 2.3
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

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

Zbiór niezgodności.
Niech S będzie skończonym zbiorem wyrażeń zawierających więcej niż jedno wyrażenie: S = {E1, ... , En}, n ≥ 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 }

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

Algorytm 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) ki = 0, σ0 = ε
2) Wyznaczamy zbiór Sσk
Jeżeli Sσk jest jednoelementowy, to STOP;
Wyjście: σk. W przeciwnym wypadku wyznaczamy zbiór niezgodności Dk dla Sσk

3) Wybieramy w zbiorze Dk parę x, t, taką, że x ∈ VAR, t ∈ TER oraz x ∉ V(t), przyjmijmy σk+1 = σk{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.

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

Klauzulą definitywną nazywamy klauzulę zawierającą dokładnie jeden pozytywny literał, 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, co zapisujemy jako:
A <- B1, ..., Bn, gdzie A to głowa, B1, ..., Bn, - body – ciało reguły
Programem definitywnym nazywamy dowolny skończony zbiór klauzul definitywnych, tzn. zbiór o postaci: ∀(B1, ..., Bn -> A).
Klauzulą celu nazywamy klauzulę nie zawierającą literałów pozytywnych, co zapisujemy: <- B1, ..., Bn, które jest równoważne formule: ∀(¬(B1 ∧ ... ∧ Bn)), z KRZ (p -> 0) -> ¬p
Klauzula Horna to klauzula definitywna lub klauzula celu zawierająca co najwyżej jeden literał pozytywny.

19. Omów strategię 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 F.2.2 (patrz odp. 11) otrzymamy, że formuła ¬G wynika logicznie ze zbioru formuł tworzących program
P. Cel G ma postać ∀(¬(B1 ∧ .... ∧ Bn)) ⇔ ¬ ∃(B1 ∧ ... ∧ Bn) Zatem formuła ∃x1 ... ∃xk (B1 ∧ ... ∧ Bn) wynika logicznie z P, gdzie x1, ..., xk są wszystkimi zmiennymi występującymi w G. Tym samym znajdziemy wszystkie obiekty x1 ... xk spełniające cel G.

20. Podaj regułę rezolucji liniowej i derywacji liniowej.

Rezolucja liniowa:
<- A1, ..., Am, ..., An - cel G
B <- B1, …, Bk - wariant klauzuli C programu P

<-(A1, ..., Am-1, B1, ..., Bk, Am+1, ..., An)σ - nowy cel.
gdzie σ jest MGU zbioru {Am, B}, tzn. Amσ = Bσ
Derywacją liniową: dla P ∪ {G} nazywamy skończony lub nieskończony ciąg, którego wyrazami są (Gn, Cn, σn), 1 <= n <= p lub 1 <= n, spełniający warunki:
a) dla każdego n, Cn jest wariantem klauzuli programu P nie zawierającym zmiennych nie występujących w G0, G1, ..., Gn-1.
b) dla każdego n, Gn jest rezolucją liniową celu Gn-1 oraz klauzuli Cn otrzymaną za pomocą podstawienia σn.

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

Refutacją liniową dla P ∪ {G} nazywamy skończoną liniową derywację dla P ∪ {G}, której ostatni cel jest klauzulą pustą.
Odpowiedź obliczeniowa to podstawienie σ1 ... σn ograniczone do zmiennych w G(σ1 ... σn)|V(G) gdzie (Gn, Cn, σn) jest refutacją liniową dla P ∪ {G}

22. Podaj określenie odpowiedzi obliczonej i poprawnej dla P È {G}, gdzie P jest programem definitywnym, a G – celem. Sformułuj twierdzenie o poprawności rezolucji liniowej.

Def. Niech σ 1 . . . σ k będą wszystkimi podstawieniami pewnej refutacji na gruncie P ∪ {G}.
Odpowiedzią obliczoną dla P ∪ {G} przez tę refutację nazywamy podstawieniem.
(σ 1 , ... , σ k) | V(G).
Tw. (o zgodności)
Każda odpowiedź obliczona dla pewnych G jest odpowiedzią poprawną dla P∪{G}
W logice 1-rzędu zachodzą warunki:
A ╞LPR Aσ A - dowolna formuła, nie ma kolizji zmiennych przy podstawieniach
A(x1,...,xn) ╞LPR A(t1,...,tn) x1/t1, ... ,xn/ tn
Wynika z prawa logiki pierwszego rzędu
∀xA →A[x/t]
Jeżeli A jest klauzulą to tak jest dla każdego podstawienia σ.

23. Podaj określenie SLD-drzewa, dla P È {G} zgodnego z regułą selekcji R

SLD - Drzewo
Pojęcia:
Przyjmujemy: Program jest ciągiem klauzul.
P = (C1 ,..., Cp)
w Prologu - liczy się kolejność klauzul.
Def.
SLD - drzewo dla P ∪ {G} zgodne z R to takie drzewo które:
Opis drzewa:
SLD drzewo - to drzewo skończone lub nieskończone, którego wierzchołki są etykietowane celami i spełnione są warunki:
a) korzeń 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 atomu (zgodnie z R) z kolejnymi klauzulami programu (pod warunkiem, że istnieją takie klauzule programu).
Gałąź - ciąg wierzchołków, że każde dwa sąsiednie są połączone krawędziami.
Gałęzie tego drzewa SLD pełnego reprezentują wszystkie derywacje dla P ∪ {G} zgodnie z R.
Gałęzie mogą być skończone lub nieskończone.
Gałęzie nieskończone - reprezentują nieskończoną derywacje (procedura się pętli)
c) skończone gałęzie drzewa to:
refutacja

gałąź chybiona
Tylko 1-ne pełne drzewo (dokładnie jedno) rozwija tak długo jak się da.
W tym drzewie się zawrą wszystkie SLD (możliwe drogi).
Ma praktyczne znaczenie - cel znajdować te właściwe.

24. Podaj określenie reguły selekcji i sformułuj twierdzenie o nieistotności reguł selskcji.

Reguła selekcji - metoda , procedura dowodzenia twierdzeń, ale to jeszcze nie algorytm.
Dwa "stopnie swobody" w rezolucji liniowej
¬ A1,...,A i, ...,A k Ai – stopnie swobody
C = (BÎ B1,...,Bn) ÎP C – klauzulę, którą będziemy unifikować
mamy A1 s = B s s - MGU
Def. (nieformalna)
Reguła selekcji określa, który atom A i ma być wybrany na danym kroku (każdym kroku).
/ nieczuła na kontekst - łatwiejsza /
W praktyce najczęściej stosuje się regułę LEFT - FIRST (lewy najpierw) tzn. w celu ¬ A1,...,A n wybrany zawsze A1
Def.
SLD - derywacją zgodną z regułą selekcji R nazywamy liniową derywację, w której na każdym kroku wybieramy atom zgodnie z R. / nie musi kończyć się pustą klauzulą /
SLD - refutacja zgodna z R / musi kończyć się pustą klauzulą /
Tw. O nieistotności reguł selekcji
R1, R2 - reguły selekcji
Jeżeli s1 jest odpowiedzią obliczoną dla P È {G} zgodnie z R1 , to istnieje odpowiedź obliczona s2 dla P È {G} zgodnie z R2, taka że G s2 jest wariantem G s1.
Jeżeli weźmiemy 2 różne reguły selekcji, to zmienne odpowiedzi co najwyżej będą przemianowane:

25. Podaj podstawowe cechy Prologu jako implementacji SLD rezolucji. Objaśnij użyte pojęcia.

Poszukiwanie gałęzi udanych w SLD - drzewie (reguła przeszukiwania)
Reguła przeszukiwania, poszukiwania (search rule) - metoda przeszukiwania SLD - drzewa (generowanie).
W praktyce stosuję się regułę **"depth first" (najpierw w głąb)** (przeszukujemy lewymi gałęziami). To znaczy (wcześniejsza reguła selekcji) ustalamy, co unifikuje się z pierwszą możliwą klauzulą, którą można uzgodnić. Gdy doszliśmy do udanej idzie dalej, a gdy dojdzie do chybionej to cofa się do tej, gdzie możliwy był wybór i dokonuje następnego wyboru. Tą metodę się wybiera - bo ją łatwo jest realizować (pamięć programu stosu dostęp tylko do jednej na wierzchu). Porządek klauzul jest ważny w Prologu. Dla pewnych drzew ta reguła może nie znaleźć rozwiązania. Dlatego też stosuje się inną reguła poszukiwania: **"breadth-first" (najpierw wszerz).**

25. 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.

26. Termy Rachunku lambda

(lambda) –zb. lambda termów konstruuje się indukcyjnie nad alfabetem Var u { (,), h λ}, gdzie Var jest przeliczalnym zbiorem zmiennych.
Do zbioru (lambda) – termów należą tylko termy skonstruowane następująco :
- dowolna zmienna x należąca do Var jest termem,
- jeżeli M należy do (lambda) – termów i x należy do Var, to (małe lambda od x, M) należy do lambda – termów. Powyższy sposób tworzenia termów nazywamy abstrakcją.
- jeżeli M należy do lambda – termów i N należy do lambda – termów, to (MN) należy do lambda – termów. Tą metodę nazywamy metodą aplikacji.

27. Alfa-konwersja. Definicja, przykłady

α – konwersja – zamiana zmiennych związanych
(mała lambda)x.M -> α λy. M[x/y] gdzie y nie należy FV(M), a M[x/y] oznacza wynik
Podstawienia zmiennej y za wszystkie wolne wystąpienia zmiennej x w termie M.
Jeżeli ciało A działa na ciało B a ciało B działa na ciało A to ciało C korzysta inaczej: Gdzie dwóch się bije tam 3 korzysta.
Będziemy używać oznaczenia T1= αT2 , aby powiedzieć, że T1 i T2 są równe modulo - α – konwersja
tzn. że mogą być zredukowane do tego samego termu stosując tylko operacje α – konwersja.
Przykład: λx.x = α λy.y
λx. xz /= α λy.yz
λx. xy /= α λx.xz

28. Beta-redukcja. Definicja, przykłady

Def. β – redukcja – obliczenia wykonywane przez procedurę symulowane są w rachunku lambda poprzez proces β – redukcji,
aplikujemy procedurę λx.M do argumentu N:
(λx.M)N -> β M[x/N] w taki sposób, aby każda zmienna wolna termu N Pozostała wolna po wykonywaniu podstawienia. Jeżeli to jest konieczne stosujemy najpierw odpowiednią alfa – konwersję np.
(λxy.xy)y (przekreślona strzałka) λy.yy – nieprawidłowe
Prawidłowo jest w taki sposób: (λxy.xy)y = α (λxz.xz)y=β λz.yz
T1 =β T2 oznaczy że termy T1 i T2 są równe modulo – β – redukcja, jeżeli mogą być sprawdzone do tego samego termu dzięki zastosowaniu skończonej liczby β –redukcji.
Przykład:
1. (λx.x(xy))N = β N(Ny)
2. (λx.y)N = β y
3. (λx.xxy)(λx.xxy) = β (λx.xxy)(λx.xxy)y = β (λx.xxy)(λx.xxy)yy…
4. (λx.x(x(yz)x)(λu.uv) =β (λu.uv)(λu.uv(yz)(λu.uv)) =β (λu.uv)((yz)v)(λu.uv) = β (yz)vv(λu.uv)

29. postać normalna termu w rachunku lambda

Def. Aplikację PQ nazywamy B – redeksem jeżeli P jest w postaci λ – abstrakcji.
Term M jest w postaci normalnej, jeżeli nie zawiera żadnych B – redeksów
Postać normalna (o ile istnieje) jest unikalna z dokładnością co do α – konwersji.
Term T jest normalizowalny, jeśli można go zredukować do postaci normalnej, a T jest silnie normalizowalny, jeżeli każda droga β – redukcji prowadzi do otrzymywania postaci normalnej
Przykłady:
Δ = λx.xx Term ΔΔ – przykład termu, który nie jest normalizowalny
ΔΔ = (λx.xx)(λx.xx) =β (λx.xx)(λx.xx) = ΔΔ
(λx.y)(ΔΔ) – przykład termu, gdzie nie każda droga redukcji prowadzi do postaci normalnej (λx.y)(ΔΔ) = β (λx.y)(ΔΔ) = β …………….. Redukując(1)
(λx.y)(ΔΔ) = β y (λx.xy)(λu.vuu) =β (λu.vuu)y = β vyy - postać normalna
(λxy.yx)uv = β (λx(λy.yx))uv =β (hy.yu)v = β vu – postać normalna (λx.xxy)(λy.yz) =β (λy.yz)(λy.yz)y = β ((λy.yz)z)y =β zzy - postać normalna

30. Własność Churcha-Rossera Tw. Churcha – Rossera – własność karo.
Jeżeli term M poprzez pewne ciągi Beta – redukcji da się zredukować do termów N1 i N2, to istnieje term M’ taki do którego da się zredukować N1 i N2 poprzez pewne ciągi Beta-redukcji

31. Reprezentacja liczb naturalnych w rachunku lambda Term hsx. X – reprezentuje liczbę naturalną zero („weź następnik s, weź zero x i zwróć zero”) Term hsx. s(s(…s(x)…)) reprezentuje liczbę naturalną n. („weź następnik s, weź zero x i zwróc n-krotnie zastosowany następnik s do zero”)
Powyższe termy są podane z dokładnością do alfa –konwersji np. hty. t(t(y)) reprezentuje 2, ponieważ λsx. S(s(x)) = alfa λ ty. t(t(y)) Mówimy, że term E rachunku lambda reprezentuje funkcję na liczbach naturalnych e: N^k -> N wtw gdy En1…nk = e(n1,…,nk) , gdzie n oznacza reprezentację liczby n w rachunku lambda. Term A = λmusx. (ms)(nsx) reprezentuje dodawanie liczb naturalnych w rachunku lambda. (m-krotnie stosujemy następnik do liczby n i w wyniku otrzymujemy n+m) *Przykład* A 1 2 = { [λm.(λn.(λsx.(ms)(nsx)))] 1 } 2 = (w miejsce m podstawiamy 2) [λn.(λsx.(2s)(nsx))]1= (w miejsce n podstawiamy 1) λsx.(2s)(1sx) = λsx.(2s)(((λt.(λy.ty))s)x) = =(t:=s) λsx.(2s)[(λy.sy) x] = (y:=x) λsx.(2s)(sx) = λsx {[λt.(λy.t(ty))] s} (sx) = (t:=s) = = λsx. (λy.s(sy))(sx) = (y:=sx) λsx. S(s(sx)) = 3.