LOGICZNE PODSTAWY INFORMATYKI

Wojciech Buszkowski

Zakład Teorii Obliczeń

Wydział Matematyki i Informatyki UAM

1. Rezolucja zdaniowa

Formuły rachunku zdań: zbudowane ze zmiennych zdaniowych za pomocą spójników logicznych $\neg$, $\wedge$, $\vee$, $\rightarrow$, $\leftrightarrow$ i nawiasów

Wartości logiczne: 1 (prawda), 0 (fałsz)

V - przeliczalny zbiór *zmiennych zdaniowych*

Wartościowanie: dowolna funkcja $w : V \mapsto \{0, 1\}$

$w(A)$ - *wartość logiczna formuły A dla wartościowania w*

$w(\neg A) = 1 - w(A)$,

$w(A \wedge B) = \min(w(A), w(B))$, $w(A \vee B) = \max(w(A), w(B))$,

$w(A \rightarrow B) = 1 \Leftrightarrow w(A) \leq w(B)$, $w(A \leftrightarrow B) = 1 \Leftrightarrow w(A) = w(B)$

Dowolne zmienne zdaniowe oznaczamy literami $p.q, r, s$ (z indeksami). Litery A, B, C, D oznaczają dowolne formuły. Litera S oznacza dowolny zbiór formuł.

Def. 1. Wartościowanie w *spełnia* formułę A, jeżeli $w(A) = 1$; spełnia zbiór formuł, jeżeli spełnia każdą formułę tego zbioru. *Tautologia* jest to formuła spełniona przez każde wartościowanie.

Def. 2. Formułę nazywamy *spełnialną*, jeżeli istnieje wartościowanie, które ją spełnia. Zbiór formuł nazywamy *spełnialnym*, jeżeli istnieje wartościowanie, które spełnia ten zbiór.

Def. 3. Formuła A *logicznie wynika* ze zbioru formuł S, jeżeli każde wartościowanie spełniające zbiór S spełnia formułę A.

Fakt 1. Formuła A jest tautologią wtw, gdy formuła $\neg A$ nie jest spełnialna.

Fakt 2. Formuła A logicznie wynika ze zbioru S wtw, gdy zbiór $S \cup \{\neg A\}$ nie jest spełnialny.

Twierdzenie 1 (o zwartości w rachunku zdań). Dla dowolnego zbioru formuł S następujące warunki są równoważne:

(a) zbiór S jest spełnialny,

(b) każdy skończony podzbiór zbioru S jest spełnialny.

Dowód. Implikacja (a)$\Rightarrow$(b) jest oczywista. Dowodzimy (b)$\Rightarrow$(a).

Niech $V = \{p_n\}_{n\geq 1}$.

Każde wartościowanie $w : V \mapsto \{0, 1\}$ można reprezentować jako nieskończony ciąg binarny $(w_n)_{n\geq 1}$, przyjmując $w_n = w(p_n)$.

Zakładamy (b). Jeżeli zbiór S jest skończony, to (a) natychmiast wynika z (b). Niech zbiór S będzie nieskończony.

Ponieważ zbiór wszystkich formuł jest przeliczalny, więc S jest też przeliczalny, czyli można ustawić wszystkie formuły zbioru S w ciąg nieskończony $(A_n)_{n\geq 1}$. Określamy zbiory $S_n = \{A_1, \ldots, A_n\}$.

Określamy rekurencyjnie wartosciowanie $w = (w_n)_{n \geq 1}$.

Przyjmujemy $w_1 = 1$, jeżeli dla nieskończenie wielu zbiorów S_n istnieje wartościowanie w^n, spełniające S_n i takie, że $w^n_1 = 1$; w przeciwnym przypadku przyjmujemy $w_1 = 0$.

Z (b) wynika, że dla nieskończenie wielu zbiorów S_n istnieje wartościowanie w^n, spełniające S_n i takie, że $w^n_1 = w_1$. Ponieważ $S_m \subseteq S_n$ dla $m \leq n$, więc dla każdego zbioru S_n istnieje takie wartościowanie w^n.

Zakładamy, że określono już $w_1, \ldots, w_k$ spełniające warunek: (*) *dla każdego zbioru* S_n *istnieje wartościowanie* w^n, *spełniające* S_n *i takie, że* $w^n_i = w_i$ *dla wszystkich* $1 \leq i \leq k$. Określamy w_{k+1} tak, by warunek (*) był spełniony przy $k := k + 1$.

Niech $A_n \in S$. Oczywiście $A_n \in S_n$. Wszystkie zmienne występujące w A_n należą do $\{p_1, \ldots, p_k\}$ dla pewnego $k \geq 1$. Na mocy (*), istnieje wartościowanie w^n, spełniające S_n, a stąd A_n, które pokrywa się z w dla zmiennych $p_1, \ldots, p_k$. Zatem w spełnia A_n. Q.E.D.

Literał: dowolna formuła postaci p lub $\neg p$

Klauzula: alternatywa skończenie wielu literałów, np. $p \vee \neg q \vee r$

Fakt 3. Wartościowanie w spełnia klauzulę A wtw, gdy w spełnia przynajmniej jeden literał klauzuli A.

Klauzula pusta: 0 (przyjmujemy, że nie jest spełniona przez żadne wartościowanie)

Def. 4. Formuła A jest *logicznie równoważna* formule B, jeżeli $w(A) = w(B)$ dla każdego wartościowania w (tzn. $A \leftrightarrow B$ jest tautologią).

Przykład. $(p \rightarrow q) \leftrightarrow (\neg p \vee q)$ jest tautologią, więc $p \rightarrow q$ i $\neg p \vee q$ są logicznie równoważne.

Fakt 4. Każda formuła jest logicznie równoważna koniunkcji skończenie wielu (niepustych) klauzul (tzn. formule w koniunkcyjnej postaci nornmalnej, kpn).

Sprowadzanie formuły do kpn wymaga czasu wykładniczego.

Wielomianowa procedura przekształcania dowolnej formuły A w formułę B w kpn taką, że:

A jest spełnialna wtw, gdy B jest spełnialna.

Jeżeli A jest zmienną, to $B = A$. Zakładamy, że A nie jest zmienną.

Każdej podformule C formuły A przypisujemy zmienną p_C, przy czym $p_q = q$ dla zmiennych q występujących w A.

$C_1, \ldots, C_k$ - wszystkie złożone podformuły formuły A, przy czym $C_1 = A$.

$B = p_A \wedge F(C_1) \wedge \cdots \wedge F(C_k)$, gdzie:

$F(C_i)$ jest kpn formuły $p_{C_i} \leftrightarrow \neg p_D$, jeżeli $C_i = \neg D$, czyli $F(C_i) = (\neg p_{C_i} \vee \neg p_D) \wedge (p_D \vee p_{C_i})$,

$F(C_i)$ jest kpn formuły $p_{C_i} \leftrightarrow (p_D \wedge p_E)$, jeżeli $C_i = D \wedge E$;

podobnie dla $C_i = D \vee E$, $C_i = D \rightarrow E$, $C_i = D \leftrightarrow E$.

Fakt 5. Formuła w kpn jest spełnialna wtw, gdy zbiór klauzul tej formuły jest spełnialny.

Metoda dowodzenia tautologii:

(1) wyznaczamy formułę B, która jest spełnialna wtw, gdy formuła $\neg A$ jest spełnialna,

(2) wykazujemy, że zbiór klauzul formuły B jest niespełnialny.

Reguła rezolucji zdaniowej (A. Blake 1937, J Robinson 1965)

$$\text{(RRZ)}\ \frac{A \vee p\,;\ B \vee \neg p}{A \vee B}$$

Na przykład:

$$\frac{p \vee q \vee r\,;\ p \vee \neg s \vee \neg r}{p \vee q \vee \neg s}$$

Kolejność i powtórzenia literałów klauzuli są nieistotne.

Wniosek reguły (RRZ) nazywamy rezolwentą przesłanek.

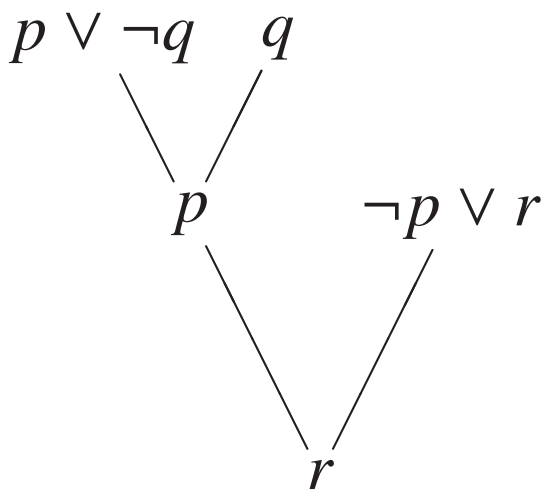
Fakt 6. Wniosek reguły (RRZ) logicznie wynika ze zbioru przesłanek.

Def. 5. *Dowodem rezolucyjnym* klauzuli A na podstawie zbioru klauzul S nazywamy skończony ciąg klauzul $(A_1, \ldots, A_n)$ taki, że $A_n = A$ i każda klauzula A_i jest elementem zbioru S lub rezolwentą pewnych klauzul A_j, A_k dla $j, k < i$.

Przykład. $S = \{p \vee \neg q,\ \neg p \vee r,\ q\}$.

1. $p \vee \neg q$ (z S)
2. q (z S)
3. p (z 1,2)
4. $\neg p \vee r$ (z S)
5. r (z 3,4)

Dowód rezolucyjny można przedstawić jako drzewo.



Def. 6. $S \vdash_{RZ} A$ (A jest *wyprowadzalne z S przez rezolucję zdaniową*) wtw, gdy istnieje dowód rezolucyjny klauzuli A na podstawie zbioru klauzul S.

Fakt 7. Jeżeli $S \vdash_{RZ} A$, to A logicznie wynika z S.

Wniosek 1. Jeżeli $S \vdash_{RZ} 0$, to zbiór S jest niespełnialny.

Twierdzenie 2 (o pełności rezolucji zdaniowej). Zbiór klauzul S jest niespełnialny wtw, gdy $S \vdash_{RZ} 0$.

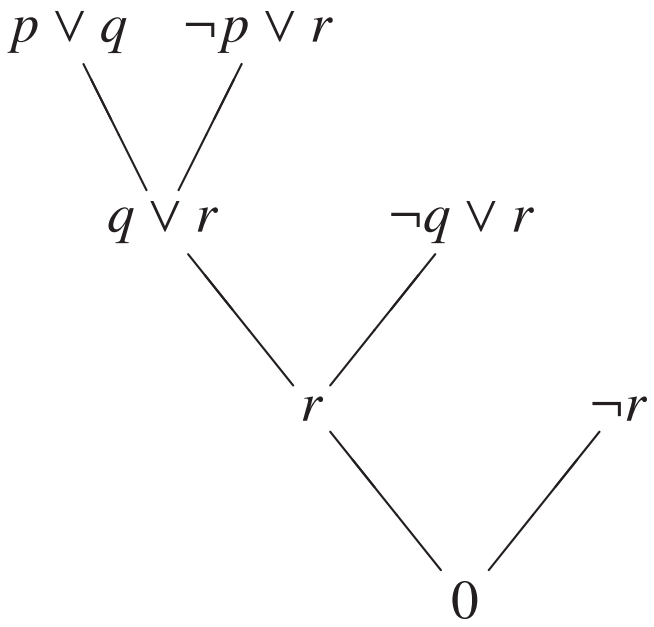
Implikacja ($\Leftarrow$) to wniosek 1. Dla dowodu ($\Rightarrow$) potrzebny jest lemat.

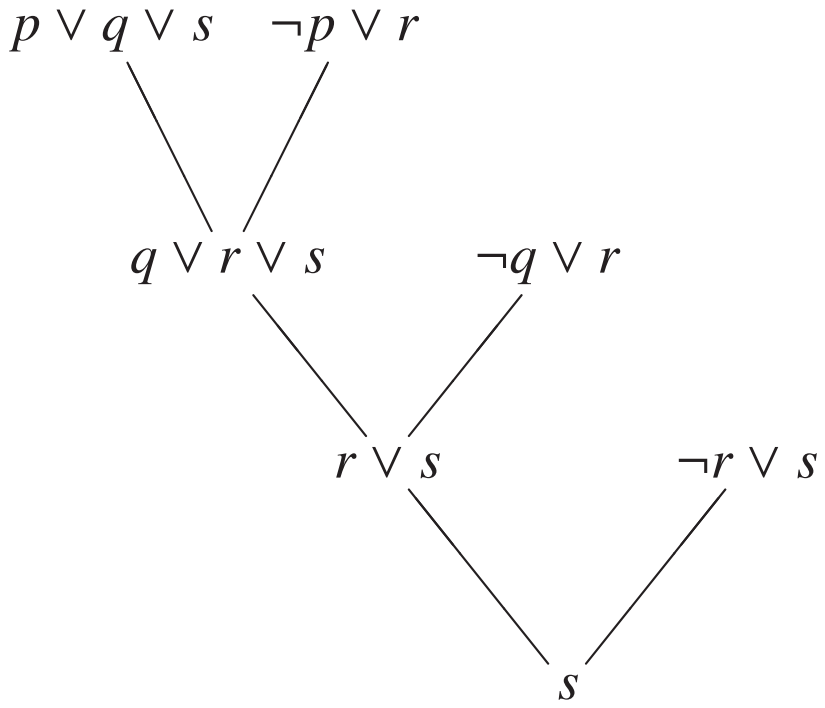
Lemat 1. Niech $S \vdash_{RZ} 0$, a literał l nie występuje w żadnej klauzuli zbioru S. Niech S' powstaje z S przez dołaczenie l do pewnych klauzul użytych w pewnym dowodzie rezolucyjnym klauzuli 0 na podstawie S. Wtedy $S' \vdash_{RZ} l$.

Ten prosty lemat wystarczy zilustrować przykładem.

$S = \{p \vee q,\ \neg p \vee r,\ \neg q \vee r,\ \neg r\}, l = s,$

$S' = \{p \vee q \vee s,\ \neg p \vee r,\ \neg q \vee r, \neg r \vee s\}$





Dowód twierdzenia 2. Implikację ($\Rightarrow$) udowodnimy najpierw dla skończonych zbiorów S przez indukcję po liczbie zmiennych występujących w klauzulach zbioru S; oznaczmy tę liczbę przez $v(S)$.

$v(S) = 0$. Zakładamy, że zbiór S jest niespełnialny. Ponieważ pusty zbiór formuł jest spełnialny, więc $S \neq \emptyset$. Zatem $S = \{0\}$, a stąd $S \vdash_{RZ} 0$.

$v(S) = n > 0$. Zakładamy, że ($\Rightarrow$) zachodzi dla wszelkich zbiorów S' takich, że $v(S') < n$. Niech $p_1, \ldots, p_n$ będą wszystkimi zmiennymi, występującymi w klauzulach zbioru S.

Określamy zbiory S_1 i S_2. S_1 powstaje z S przez usunięcie wszystkich klauzul zawierających p_n i usunięcie $\neg p_n$ z pozostałych klauzul. S_2 powstaje z S przez usunięcie wszystkich klauzul zawierających $\neg p_n$ i usunięcie p_n z pozostałych klauzul. Oczywiście zmienna p_n nie występuje w klauzulach zbioru S_1, ani S_2.

Zakładamy, że zbiór S jest niespełnialny. Wtedy zbiory S_1 i S_2 są niespełnialne. Gdyby jakieś skończone wartościowanie w, określone dla zmiennych $p_1, \ldots, p_{n-1}$, spełniało S_1, to przyjmując $w(p_n) = 1$, otrzymalibyśmy wartościowanie spełniające S. Podobnie rozumujemy dla S_2, przyjmując $w(p_n) = 0$.

Ponieważ $v(S_1) < n$ i $v(S_2) < n$, więc $S_1 \vdash_{RZ} 0$ i $S_2 \vdash_{RZ} 0$ na mocy założenia indukcyjnego.

Jeżeli istnieje dowód rezolucyjny klauzuli 0 z S_1 lub z S_2, w którym występują wyłącznie klauzule z S (tzn. takie klauzule, które nie zostały usunięte ani ograniczone przy tworzeniu S_1 i S_2), to $S \vdash_{RZ} 0$.

W przeciwnym przypadku, dowód klauzuli 0 z S_1 zawiera przynajmniej jedną klauzulę, powstającą z klauzuli z S przez usunięcie $\neg p_n$. Wtedy $S \vdash_{RZ} \neg p_n$ na mocy lematu 1. Podobnie $S \vdash_{RZ} p_n$, ponieważ dowód klauzuli 0 z S_2 zawiera istotnie przynajmniej jedną klauzulę powstającą z klauzuli z S przez usunięcie p_n. Zatem $S \vdash_{RZ} 0$ na mocy (RRZ).

Dla nieskończonych zbiorów S korzystamy z twierdzenia 1.

Zakładamy, że zbiór S jest niespełnialny. Wtedy istnieje skończony zbiór $S' \subseteq S$, który nie jest spełnialny. Mamy $S' \vdash_{RZ} 0$, a więc $S \vdash_{RZ} 0$. Q.E.D.

Dygresja. Twierdzenie o zwartości w rachunku zdań wynika ze zwartości przestrzeni topologicznej $\{0,1\}^N$ (nieskończonych ciągów binarnych, czyli wartościowań) z topologią produktową wyznaczoną przez topologię dyskretną na $\{0,1\}$. Dla każdego A zbiór $W(A) = \{w : w(A) = 1\}$ jest domknięty. Jeżeli każdy skończony podzbiór zbioru S jest spełnialny, to rodzina $\{W(A) : A \in S\}$ jest scentrowaną rodziną zbiorów domkniętych; stąd $\bigcap_{A\in S} W(A) \neq \emptyset$, a więc zbiór S jest spełnialny.

2. Klasyczny Rachunek Predykatów

KRP jest działem logiki formalnej, wzbogacającym KRZ o nowe stałe logiczne (kwantyfikatory, równość) i oferującym dokładniejszą analizę struktury logicznej zdań. Inne nazwy: klasyczny rachunek logiczny, logika pierwszego rzędu, logika elementarna.

W matematyce przez *strukturę relacyjną* rozumie się zbiór z wyróżnionymi relacjami, operacjami (działaniami) i elementami. Np. zbiór liczb całkowitych $\mathbb{Z}$ z relacjami równości $=$ i mniejszości $<$, działaniami dodawania $+$ i mnożenia $\cdot$ oraz elementami 0 i 1 jest strukturą relacyjną. Inną strukturą relacyjną jest zbiór liczb rzeczywistych z analogicznymi relacjami, operacjami i elementami wyróżnionymi.

Języki formalne KRP (tzw. *języki elementarne*), są dostosowane do opisu struktur relacyjnych: różne języki odpowiadają różnym typom struktur relacyjnych. Typ struktury relacyjnej określa liczbę relacji, operacji i elementów wyróżnionych oraz liczbę ich argumentów.

2.1. Symbole języka elementarnego

A. Symbole logiczne (wspólne dla wszystkich języków):

- stałe logiczne: $\neg, \wedge, \vee, \rightarrow, \leftrightarrow, \forall, \exists$; stała $\forall$ to *kwantyfikator ogólny* (generalny, uniwersalny, duży), stała $\exists$ to *kwantyfikator szczegółowy* (egzystencjalny, istnienia, mały);

- zmienne indywiduowe, dla których rezerwujemy litery x, y, z (także z indeksami); dany jest nieskończony ciąg tych zmiennych;

- symbole pomocnicze: nawiasy i przecinek.

B. Symbole pozalogiczne (różne w różnych językach):

- symbole relacyjne (predykatowe); notacja ogólna: P^i, Q^j, R^k; w konkretnych językach np. $=, <, \leq$;

- symbole funkcyjne; notacja ogólna: f^i, g^j, h^k; w konkretnych językach np. $+, \cdot$;

- stałe indywiduowe; notacja ogólna: a, b, c; w konkretnych językach np. $0, 5, \pi, \emptyset$.

Górny indeks symbolu relacyjnego lub funkcyjnego oznacza liczbę argumentów (arność) danego symbolu. Te indeksy podajemy zazwyczaj tylko w deklaracji języka.

Język elementarny jest jednoznacznie określony przez swoje symbole pozalogiczne. Możemy formalnie określić język elementarny jako układ $L = (R_L, F_L, C_L)$ taki, że R_L, F_L, C_L są zbiorami rozłącznymi, przy czym R_L jest zbiorem niepustym. R_L jest zbiorem symboli relacyjnych, F_L zbiorem symboli funkcyjnych, a C_L zbiorem stałych indywiduowych języka L.

Na przykład, $L = (\{=^2, <^2\}, \{+^2, \cdot^2\}, \{0, 1\})$ jest językiem elementarnym, stosownym do opisu struktur relacyjnych liczb całkowitych i liczb rzeczywistych, wspomnianych powyżej. W notacji ogólnej można zadeklarować taki sam język jako $L = (\{P^2, Q^2\}, \{f^2, g^2\}, \{a, b\})$.

Deklaracja języka elementarnego nie różni się istotnie od deklaracji typu struktury relacyjnej.

2.2. Termy i formuły

Wyrażeniem języka L nazywamy dowolny skończony ciąg symboli języka L. Zakładamy, że żaden symbol języka nie jest wyrażeniem tego języka. Wyrażenia przedstawiamy jako napisy; np. ab przedstawia ciąg dwóch symboli a i b (w notacji matematycznej zastosowano by zapis (a, b)).

Wyróżniamy dwa rodzaje wyrażeń sensownych: termy i formuły. Termy są wyrażeniami nazwowymi, formuły wyrażeniami zdaniowymi. Rolą termów jest oznaczanie elementów struktury relacyjnej. Formuły wyrażają własności elementów struktury relacyjnej lub całej struktury; formuły są prawdziwe lub fałszywe.

Definicja 1. *Termy* dzielą się na proste i złożone. Termy proste są to zmienne indywiduowe i stałe indywiduowe. Termy złożone są to wyrażenia $f(t_1, \ldots, t_n)$ takie, że f jest $n-$argumentowym symbolem funkcyjnym, a $t_1, \ldots, t_n$ są termami.

Przykład. Niech $F_L = \{f^1, g^2\}$, $C_L = \{a\}$. Termami prostymi są np. x, y, z, a. Termy złożone to np.
$f(a), g(a,a), f(g(a,a)), f(x), g(x,y), g(f(x),a)$ itp.

Definicja 1 jest definicją rekurencyjną. Najpierw określa termy proste (krok początkowy). Następnie podaje zasadę konstrukcji termów złożonych z termów o mniejszej złożoności; złożoność $f(t_1,\ldots,t_n)$ jest większa od złożoności każdego z termów $t_1,\ldots,t_n$. *Złożoność termu* określamy ściśle jako liczbę wszystkich wystąpień symboli funkcyjnych w tym termie. Termy proste mają złożoność 0, termy $f(x), g(a,y)$ złożoność 1, term $g(f(x),y)$ złożoność 2 itd.

Uwaga. W Definicji 1 i przykładzie termy zapisywano w notacji *prefiksowej*: symbol funkcyjny (operator) przed argumentami. Standardowe operatory dwuargumentowe zwykle piszemy między argumentami (notacja *infiksowa*), np. $x + y$ zamiast $+(x,y)$. W logice stosujemy notację prefiksową w rozważaniach ogólnych; w przykładach stosujemy notację infiksową dla znanych operatorów.

Definicja 2. *Formuły* dzielą się na atomowe i złożone. Formuły atomowe to wyrażenia $P(t_1, \ldots, t_n)$ takie, że P jest $n-$argumentowym symbolem relacyjnym, a $t_1, \ldots, t_n$ są termami. Formuły złożone są to wyrażenia $(\neg A)$, $(A \wedge B)$, $(A \vee B)$, $(A \rightarrow B)$, $(A \leftrightarrow B)$, $(\forall x A)$, $(\exists x A)$ takie, że A, B są formułami, a x jest zmienną indywiduową.

Powyższa definicja jest też definicją rekurencyjną. *Złożoność formuły* określamy jako liczbę wszystkich wystąpień stałych logicznych w tej formule. Krok początkowy określa formuły atomowe; mają one złożoność 0. Krok indukcyjny podaje zasady konstrukcji formuł złożonych z formuł o mniejszej złożoności.

Przykład. Niech $R_L = \{P^1, Q^2\}$, $F_L = \{f^1\}$, $C_L = \{a\}$. Wtedy np. $P(x)$, $Q(a, y)$, $P(f(x))$, $Q(f(x), f(y))$ są formułami atomowymi (atomami). Formułami złożonymi są np. $(\neg P(x))$, $(\forall x(\neg P(x)))$, $(\forall x(P(x) \rightarrow (\exists y Q(x, y))))$. W praktyce opuszczamy nawiasy: (1) zewnętrzne, (2) jak w KRZ, dodatkowo przyjmując, że $\forall x$ i $\exists x$ mają tę samą siłę, co $\neg$. Piszemy: $\neg P(x)$, $\forall x \neg P(x)$, $\forall x(P(x) \rightarrow \exists y Q(x, y))$.

Formułę $\forall xA$ czytamy: dla każdego x A.

Formułę $\exists xA$ czytamy: istnieje x takie, że A.

Znaczenie kwantyfikatorów jest intuicyjnie jasne, lecz nie można go zdefiniować za pomocą innych, bardziej podstawowych pojęć. Jeżeli opisujemy ustaloną, skończoną strukturę relacyjną, której wszystkie elementy są oznaczone stałymi indywiduowymi $a_1, \ldots, a_n$, to $\forall xA(x)$ jest równoważne koniunkcji $A(a_1) \wedge \cdots \wedge A(a_n)$, a $\exists xA(x)$ jest równoważne alternatywie $A(a_1) \vee \cdots \vee A(a_n)$. Tu $A(x)$ reprezentuje dowolną formułę, a $A(a_i)$ reprezentuje wynik podstawienia a_i za x w $A(x)$ (podstawianie w formułach KRP omawiamy dokładnie w podrozdziale 2.3).

W przypadku struktur nieskończonych otrzymalibyśmy koniunkcje i alternatywy nieskończenie wielu formuł, niedopuszczalne w KRP. Są dopuszczalne w *logikach infinitarnych*, znacznie bardziej skomplikowanych od KRP.

Kwantyfikatory wiążą zmienne. W formule $\forall x P(x, y)$ zmienna y jest wolna, a zmienna x jest związana; dokładniej, oba wystąpienia zmiennej x są związane.

Pojęcie podformuły danej formuły można określić rekurencyjnie: (1) jedyną podformułą formuły atomowej A jest A, (2) podformułami formuły $\neg A$ ($\forall x A$, $\exists x A$) są ta formuła i wszystkie podformuły formuły A, (3) podformułami formuły $A \wedge B$ ($A \vee B$, $A \rightarrow B$, $A \leftrightarrow B$) są ta formuła oraz wszystkie podformuły formuł A, B.

Definicja 3. Wystąpienie zmiennej x w formule A jest *związane*, jeżeli występuje w pewnej podformule B formuły A takiej, że B jest postaci $\forall x C$ lub $\exists x C$; pozostałe wystąpienia x w A są *wolne*. Zmienną nazywamy *wolną w formule* A, jeżeli w A występuje przynajmniej jedno wolne wystąpienie tej zmiennej.

Definicja 4. Formuły nie zawierające zmiennych wolnych nazywamy *zdaniami* (albo: formułami domkniętymi). Termy i atomy nie zawierające zmiennych nazywamy *ustalonymi*.

2.3. Podstawianie w termach i formułach

TER_L oznacza zbiór wszystkich termów języka L. Podstawieniem nazywamy dowolną funkcję $\sigma : V \mapsto \mathrm{TER}_L$ taką, że V jest skończonym zbiorem zmiennych indywiduowych. Podstawienie σ reprezentujemy jako listę przypisań:

$$\sigma = [x_1/t_1, \ldots, x_n/t_n],$$

gdzie $t_i = \sigma(x_i)$ dla $i = 1, \ldots, n$. Wtedy $V = \{x_1, \ldots, x_n\}$. Przyjmujemy, że te zmienne są różne.

Symbolem ε oznaczamy podstawienie identycznościowe [].

Dla każdego termu t i podstawienia σ powyższej postaci określamy term $t\sigma$, powstający z t w wyniku równoczesnego podstawienia termu t_i za każde wystąpienie zmiennej x_i dla $i = 1, \ldots n$.

Dla każdej formuły A i podstawienia σ powyższej postaci określamy formułę $A\sigma$, powstającą z A w wyniku równoczesnego podstawienia t_i za każde wolne wystąpienie zmiennej x_i dla $i = 1, \ldots, n$.

Przykład. $((x + y) \cdot x)[x/y,\ y/z + 1] \equiv (y + (z + 1)) \cdot y$.
$(\exists x x = y)[x/y,\ y/z + 1] \equiv \exists x x = z + 1$.

Rekurencyjna definicja $t\sigma$ i $A\sigma$ dla $\sigma = [x_1/t_1, \ldots, x_n/t_n]$.

$x_i\sigma \equiv t_i$,

$y\sigma \equiv y$ dla $y \notin \{x_1, \ldots, x_n\}$,

$a\sigma \equiv a$ dla $a \in C_L$,

$f(s_1, \ldots, s_m)\sigma \equiv f(s_1\sigma, \ldots, s_m\sigma)$,

$P(s_1, \ldots, s_m)\sigma \equiv P(s_1\sigma, \ldots, s_m\sigma)$,

$(\neg A)\sigma \equiv \neg A\sigma$ (przyjmujemy, że σ wiąże najsilniej),

$(A * B)\sigma \equiv A\sigma * B\sigma$,

$(\forall x A)\sigma \equiv \forall x A\sigma$, $(\exists x A)\sigma \equiv \exists x A\sigma$, jeżeli $x \notin \{x_1, \ldots, x_n\}$,

$(\forall x_i A)\sigma \equiv \forall x_i A\sigma'$, $(\exists x_i A)\sigma \equiv \exists x_i A\sigma'$, gdzie σ' powstaje z σ przez usunięcie przypisania x_i/t_i.

Dla dowolnych podstawień σ, η określamy *złożenie* $\sigma\eta$ jako podstawienie, polegające na kolejnym wykonaniu podstawień σ i η.

Niech $\sigma = [x_1/t_1, \ldots, x_n/t_n]$, $\eta = [y_1/s_1, \ldots, y_m/s_m]$, przy czym nie wykluczamy $x_i \equiv y_j$ dla pewnych i, j. Wtedy:

$$\sigma\eta = [x_1/t_1\eta, \ldots, x_n/t_n\eta, y_{i_1}/s_{i_1}, \ldots, y_{i_k}/s_{i_k}],$$

gdzie $y_{i_1}, \ldots, y_{i_k}$ są tymi wszystkimi zmiennymi ze zbioru $\{y_1, \ldots, y_m\}$, które nie należą do zbioru $\{x_1, \ldots, x_n\}$.

Przyjmujemy, że $\sigma = \eta$, jeżeli $x\sigma = x\eta$ dla każdej zmiennej x. Wtedy mamy $t\sigma = t\eta$ oraz $A\sigma = A\eta$ dla wszystkich termów t i formuł A. Zachodzą następujące równości:

$t(\sigma\eta) = (t\sigma)\eta$, $A(\sigma\eta) = (A\sigma)\eta$ (dla otwartych A), $t\varepsilon = t$, $A\varepsilon = A$

$(\sigma\eta)\theta = \sigma(\eta\theta)$ (prawo łączności złożenia podstawień),

$\varepsilon\sigma = \sigma = \sigma\varepsilon$ (ε jest elementem neutralnym dla złożenia podstawień).

Definicja 5. Mówimy, że term t jest podstawialny za zmienną x w formule A, jeżeli żadne wolne wystąpienie x w A nie występuje w podformule postaci $\forall y B$, ani $\exists y B$ takiej, że y występuje w t.

Przykład. Niech $A \equiv \exists y\, x < y$. Wtedy t jest podstawialne za x w A wtw, gdy y nie występuje w t. Na przykład, termy z, $z + x$, 0 są podstawialne za x w A, lecz termy y, $y + 1$ nie są podstawialne za x w A. Zauważmy, że formuła A jest prawdziwa w strukturze liczb całkowitych. Jeżeli t jest podstawialne za x w A, to formuła $A[x/t]$ jest też prawdziwa w tej strukturze, np. $\exists y\, z < y$, $\exists y\, z + x < y$, $\exists y\, 0 < y$. Jeżeli t nie jest podstawialne za x w A, to $A[x/t]$ może nie być formułą prawdziwą w tej strukturze, np. $\exists y\, y < y$, $\exists y\, y + 1 < y$. Jeżeli t nie jest podstawialne za x w A, to mówimy, że nastąpiła *kolizja zmiennych* przy podstawianiu $A[x/t]$.

Uwaga. Każdy term ustalony jest podstawialny za dowolną zmienną w dowolnej formule. Każdy term jest podstawialny za dowolną zmienną w dowolnej formule, nie zawierającej kwantyfikatorów.

3. Interpretacje i twierdzenie Herbranda

3.1. Interpretacje, modele, prawa

Definicja 1. *Interpretacją* języka L nazywamy parę $M = (U_M, (-)^M)$ taką, że U_M jest niepustym zbiorem (uniwersum interpretacji M), a $(-)^M$ jest funkcją określoną na zbiorze symboli pozalogicznych języka L, spełniającą następujące warunki:

jeżeli $P^n \in R_L$, to $(P^n)^M$ jest $n-$argumentową relacją na zbiorze U_M,

jeżeli $f^n \in F_L$, to $(f^n)^M$ jest $n-$argumentową operacją na zbiorze U_M,

jeżeli $a \in C_L$, to a^M jest elementem zbioru U_M.

Oznaczenie: TER_L^u - zbiór termów ustalonych języka L

Definicja 2. Niech M będzie interpretacją języka L. Dla każdego termu $t \in \mathrm{TER}_L^u$ określamy element $t^M \in U_M$, zwany *wartością* termu t w interpretacji M:

a^M jest określone przez M dla $a \in C_L$,

$(f(t_1, \ldots, t_n))^M = f^M(t_1^M, \ldots t_n^M)$.

Definicja 3. Niech M będzie interpretacją języka L. Język L wzbogacamy do języka L_M, dodając do L nowe stałe indywiduowe $\overline{d}$ dla wszystkich elementów $d \in U_M$ takich, że $d \neq t^M$ dla każdego $t \in \mathrm{TER}_L^u$.

Interpretację M wzbogacamy do interpretacji dla języka L_M, przyjmując $(\overline{d})^M = d$ dla każdej nowej stałej indywiduowej d.

Każdy element zbioru U_M jest wartością pewnego ustalonego termu języka L_M.

Definicja 4. Niech M będzie interpretacją języka L. Określamy pojęcie *zdania prawdziwego* w M dla zdań języka L_M.

$M \models A$ czytamy: zdanie A jest prawdziwe w M.

$M \models P(t_1, \ldots, t_n) \Leftrightarrow P^M(t_1^M, \ldots, t_n^M)$,

$M \models \neg A \Leftrightarrow M \not\models A$,

$M \models A \wedge B \Leftrightarrow M \models A$ i $M \models B$,

$M \models A \vee B \Leftrightarrow M \models A$ lub $M \models B$,

$M \models A \to B \Leftrightarrow (M \models A \Rightarrow M \models B)$,

$M \models A \leftrightarrow B \Leftrightarrow (M \models A \Leftrightarrow M \models B)$,

$M \models \forall x A \Leftrightarrow$ dla każdego $t \in \mathrm{TER}^u_{L_M}$ $M \models A[x/t]$,

$M \models \exists x A \Leftrightarrow$ istnieje $t \in \mathrm{TER}^u_{L_M}$ takie, że $M \models A[x/t]$.

Oznaczenie: $V(A)$ - zbiór zmiennych wolnych w formule A

Definicja 5. Niech M będzie interpretacją języka L. Niech A będzie formułą języka L taką, że $V(A) = \{x_1, \ldots, x_n\}$.

Mówimy, że formuła A jest *prawdziwa* w M (piszemy: $M \models A$), jeżeli $M \models A[x_1/t_1, \ldots, x_n/t_n]$ dla wszystkich termów $t_1, \ldots, t_n \in \mathrm{TER}^u_{L_M}$.

Wniosek. $M \models A$ wtw, gdy $M \models \forall x_1 \ldots \forall x_n\, A$.

Definicja 6. Podstawienie σ nazywamy *ustalonym dla formuły* A, jeżeli $A\sigma$ jest zdaniem.

Uwaga. Definicję 5 można sformułować tak:

$M \models A$, jeżeli $M \models A\sigma$ dla każdego podstawienia σ ustalonego dla A w języku L_M.

Oznaczenie: FOR_L - zbiór formuł języka L

Definicja 7. Mówimy, że formuła A jest *prawem* (albo: tautologią) KRP, jeżeli $M \models A$ dla każdej interpretacji M danego języka. Piszemy: $\models_{KRP} A$.

$$\models_{KRP} A \Leftrightarrow \forall_M M \models A$$

Definicja 8. Interpretację M nazywamy *modelem* zbioru formuł S, jeżeli $M \models A$ dla każdej formuły $A \in S$. Piszemy: $M \models S$.

$$M \models S \Leftrightarrow \forall_{A \in S} M \models A$$

Definicja 9. Mówimy, że formuła A *logicznie wynika* ze zbioru formuł S, jeżeli formuła A jest prawdziwa we wszystkich modelach zbioru S. Piszemy: $S \models_{KRP} A$.

$$S \models_{KRP} A \Leftrightarrow \forall_M (M \models S \Rightarrow M \models A)$$

Przykład. Następujące formuły są prawami KRP.

(1) $\forall x P(x) \to P(a)$ (prawo podstawiania)

(2) $P(a) \to \exists x P(x)$ (drugie prawo podstawiania)

(3) $\forall x(P(x) \wedge Q(x)) \leftrightarrow \forall x P(x) \wedge \forall x Q(x)$ (prawo rozdzielności $\forall$ względem $\wedge$)

(4) $\exists x(P(x) \vee Q(x)) \leftrightarrow \exists x P(x) \vee \exists x Q(x)$ (prawo rozdzielności $\exists$ względem $\vee$)

Prawdziwość tych praw w każdej interpretacji danego języka jest oczywista na podstawie znaczenia kwantyfikatorów i spójników logicznych. Istnieją różne *systemy dowodzenia* praw KRP; niektóre z nich poznamy w dalszym ciągu.

Fakt 1. Niech $A_1, \ldots, A_n, B$ będą zdaniami. Wtedy B logicznie wynika w KRP ze zbioru $\{A_1, \ldots, A_n\}$ wtw, gdy formuła $A_1 \wedge \cdots \wedge A_n \to B$ jest prawem KRP.

Przykład. Ponieważ (1) i (2) są prawami KRP, więc zdanie $P(a)$ logicznie wynika ze zdania $\forall x P(x)$ (tzn. ze zbioru $\{\forall x P(x)\}$), a zdanie $\exists x P(x)$ logicznie wynika ze zdania $P(a)$.

Definicja 10. Mówimy, że formuła A jest *logicznie równoważna* formule B w KRP, jeżeli formuła $A \leftrightarrow B$ jest prawem KRP.

Przykład. Ponieważ (3) i (4) są prawami KRP, więc formuła po lewej stronie danego prawa jest logicznie równoważna formule po prawej stronie tego prawa.

Podstawowe prawa KRP

(TL0) wszystkie formuły logicznie prawdziwe na gruncie KRZ

prawa podstawiania

(TL1a) $\forall x A \to A[x/t]$, (TL1e) $A[x/t] \to \exists x A$ (warunek: t jest podstawialne za x w A)

prawa zbędnego kwantyfikatora

(TL2a) $\forall x A \leftrightarrow A$, (TL2e) $\exists x A \leftrightarrow A$ (warunek: $x \notin V(A)$)

prawa dwustronnego dołączania kwantyfikatorów do implikacji

(TL3a) $\forall x(A \to B) \to (\forall x A \to \forall x B)$

(TL3e) $\forall x(A \to B) \to (\exists x A \to \exists x B)$

prawa jednostronnego dołączania kwantyfikatora do implikacji

(TL4a) $\forall x(A \to B) \to (A \to \forall x B)$ (warunek: $x \notin V(A)$)

(TL4e) $\forall x(A \to B) \to (\exists x A \to B)$ (warunek: $x \notin V(B)$)

prawa rozdzielności kwantyfikatorów

(TL5a) $\forall x(A \wedge B) \leftrightarrow \forall xA \wedge \forall xB$

(TL5e) $\exists x(A \vee B) \leftrightarrow \exists xA \vee \exists xB$

niepełne prawa rozdzielności kwantyfikatorów

(TL6a) $\forall xA \vee \forall xB \rightarrow \forall x(A \vee B)$

(TL6e) $\exists x(A \wedge B) \rightarrow \exists xA \wedge \exists xB$

prawa przestawiania kwantyfikatorów

(TL7a) $\forall x\forall yA \leftrightarrow \forall y\forall xA$

(TL7e) $\exists x\exists yA \leftrightarrow \exists y\exists xA$

niepełne prawo przestawiania kwantyfikatorów

(TL8) $\exists x\forall yA \rightarrow \forall y\exists xA$

prawa De Morgana dla kwantyfikatorów

(TL9a) $\neg\forall xA \leftrightarrow \exists x\neg A$, (TL9e) $\neg\exists xA \leftrightarrow \forall x\neg A$

prawa zamiany zmiennej związanej

(TL10a) $\forall xA \leftrightarrow \forall yA[x/y]$, (TL10e) $\exists xA \leftrightarrow \exists yA[x/y]$,

jeśli spełnione są warunki: (w1) $x \not\equiv y$, (w2) $y \notin V(A)$, (w3) y jest podstawialne za x w A.

Te warunki są spełnione, jeżeli y nie występuje w $\forall xA$ (jest tzw. nową zmienną).

prawa wyłączania kwantyfikatora przed nawias

(TL11a) $\forall xA * B \leftrightarrow \forall x(A * B)$ dla $* \in \{\wedge, \vee\}$, jeśli $x \notin V(B)$.

(TL11e) $\exists xA * B \leftrightarrow \exists x(A * B)$ przy tych samych zastrzeżeniach.

prawa ekstensjonalności dla kwantyfikatorów

(TL12a) $\forall x(A \leftrightarrow B) \rightarrow (\forall xA \leftrightarrow \forall xB)$

(TL12e) $\forall x(A \leftrightarrow B) \rightarrow (\exists xA \leftrightarrow \exists xB)$

Definicja 11. Zbiór formuł S nazywamy *spełnialnym*, jeżeli istnieje model zbioru S.

Fakt 2. Zdanie A jest prawem KRP wtw, gdy zdanie $\neg A$ nie jest spełnialne (tzn. zbiór $\{\neg A\}$ nie jest spełnialny).

Fakt 3. Niech A będzie zdaniem. $S \models_{KRP} A$ wtw, gdy zbiór $S \cup \{\neg A\}$ nie jest spełnialny.

Fakt 4. Niech A będzie formułą otwartą (tzn. bez kwantyfikatorów). Wtedy $A \models_{KRP} A\sigma$ dla dowolnego podstawienia σ.

Dowód. Zakładamy $M \models A$. Niech σ będzie podstawieniem. Niech η będzie podstawieniem ustalonym dla $A\sigma$ w języku L_M. Mamy $(A\sigma)\eta = A(\sigma\eta)$, więc $\sigma\eta$ jest podstawieniem ustalonym dla A. Wobec tego $M \models A(\sigma\eta)$, czyli $M \models (A\sigma)\eta$. Tak jest dla każdego podstawienia η ustalonego dla $A\sigma$, a więc $M \models A\sigma$. Q.E.D.

3.2. Interpretacje Herbranda i twierdzenie Herbranda

$H_L = \mathrm{TER}_L^u$ - *uniwersum Hebranda* dla języka L, B_L - zbiór ustalonych atomów języka L, *baza Hebranda* dla języka L

Definicja 12. Interpretację M języka L nazywamy *interpretacją Herbranda*, jeżeli spełnia następujące warunki:

(a) $U_M = H_L$,

(b) $a^M = a$ dla każdej stałej $a \in C_L$,

(c) $f^M(t_1, \ldots, t_n) = f(t_1, \ldots, t_n)$ dla każdego symbolu funkcyjnego $f^n \in F_L$ i wszystkich $t_1, \ldots, t_n \in H_L$.

Fakt 5. Niech M będzie interpretacją Herbranda języka L. Wtedy $t^M = t$ dla każdego $t \in H_L$.

Dowód. Indukcja po t. $a^M = a$ dla $a \in C_L$ na mocy Def. 12 (b).

$(f(t_1, \ldots, t_n))^M = f^M(t_1^M, \ldots, t_n^M) = f^M(t_1, \ldots, t_n) = f(t_1, \ldots, t_n)$ na mocy Def. 2, założenia indukcyjnego i Def. 12(c). Q.E.D.

UWAGA. Jeżeli M jest interpretacją Herbranda języka L, to $L_M = L$.

Lemat 1. Niech S będzie zbiorem zdań otwartych języka L. Następujące warunki są równoważne:

(a) zbiór S jest spełnialny w sensie KRP,

(b) zbiór S jest spełnialny w sensie KRZ,

(c) istnieje model Herbranda zbioru S, tzn. interpretacja Herbranda M języka L taka, że $M \models S$.

Dowód. Oczywiście (c) $\Rightarrow$ (a).

Wykażemy (a) $\Rightarrow$ (b). Zakładamy (a). Istnieje interpretacja M języka L taka, że $M \models S$. Określamy wartościowanie $w : B_L \mapsto \{0, 1\}$.

$w(A) = 1 \;\Leftrightarrow\; M \models A$ dla każdego $A \in B_L$.

Przez indukcję po złożoności A łatwo udowodnić, że ta równoważność zachodzi dla każdego otwartego zdania A. Ponieważ $M \models A$ dla każdego $A \in S$, więc w spełnia zbiór S.

Wykażemy (b) $\Rightarrow$ (c). Zakładamy (b). Istnieje wartościowanie $w : B_L \mapsto \{0, 1\}$, które spełnia zbiór S. Określamy interpretację Herbranda M języka L.

$P^M(t_1, \ldots, t_n) \Leftrightarrow w(P(t_1, \ldots, t_n)) = 1$

Wykażemy, że dla każdego otwartego zdania A:

$M \models A \Leftrightarrow w \models A$,

gdzie $w \models A$ znaczy $w(A) = 1$.

$M \models P(t_1, \ldots, t_n) \Leftrightarrow P^M(t_1^M, \ldots, t_n^M) \Leftrightarrow P^M(t_1, \ldots, t_n) \Leftrightarrow w \models P(t_1, \ldots, t_n)$

$M \models \neg A \Leftrightarrow M \not\models A \Leftrightarrow w \not\models A \Leftrightarrow w \models \neg A$

$M \models A \wedge B \Leftrightarrow (M \models A \text{ i } M \models B) \Leftrightarrow (w \models A \text{ i } w \models B) \Leftrightarrow w \models A \wedge B$

Podobnie dla $\vee, \rightarrow, \leftrightarrow$. Ponieważ w spełnia zbiór S, więc $M \models S$. Q.E.D.

Definicja 13. *Zdaniem uniwersalnym* nazywamy zdanie postaci $\forall x_1 \ldots \forall x_n B$, gdzie B jest formułą otwartą (zwaną *matrycą* tego zdania). Każde zdanie postaci $B\sigma$ nazywamy *ustaloną instancją* tego zdania uniwersalnego.

Oznaczenia: $gr(A)$ - zbiór wszystkich ustalonych instancji zdania uniwersalnego A

$gr(S) = \bigcup_{C \in S} gr(C)$ dla zbioru zdań uniwersalnych S

Lemat 2. Niech S będzie zbiorem zdań uniwersalnych. Następujące warunki są równoważne:

(a) zbiór S jest spełnialny w sensie KRP,

(b) zbiór $gr(S)$ jest spełnialny w sensie KRP (KRZ),

(c) istnieje model Herbranda zbioru S.

Dowód. Oczywiście (c) $\Rightarrow$ (a).

Wykażemy (a) $\Rightarrow$ (b). Zakładamy (a). Istnieje interpretacja M taka, że $M \models S$. Niech $C \in gr(S)$. Wtedy $C = B\sigma$, gdzie B jest matrycą zdania uniwersalnego $A \in S$. Mamy $M \models B$, skoro $M \models A$. Stąd $M \models B\sigma$ na mocy Faktu 4. Zatem $M \models C$. Wykazaliśmy $M \models gr(S)$.

Wykażemy (b) $\Rightarrow$ (c). Zakładamy (b). Na mocy Lematu 1 istnieje model Herbranda M zbioru $gr(S)$. Wykażemy $M \models S$. Niech $A \in S$. Wtedy $A = \forall x_1 \ldots \forall x_n B$, gdzie B jest formułą otwartą. Dla każdego podstawienia σ ustalonego dla B mamy $B\sigma \in gr(S)$, a więc $M \models B\sigma$. Zatem $M \models A$. Wykazaliśmy $M \models S$. Q.E.D.

Twierdzenie Herbranda. Niech S będzie zbiorem zdań uniwersalnych. Zbiór S nie jest spełnialny w sensie KRP wtw, gdy pewien skończony podzbiór zbioru $gr(S)$ nie jest spełnialny w sensie KRZ.

Dowód. Twierdzenie wynika z Lematu 2 i twierdzenia o zwartości dla KRZ.

Skolemizacja

Dla każdego zdania A możemy skonstruować zdanie uniwersalne A_u, spełniające warunek: A jest spełnialne wtw, gdy A_u jest spełnialne.

(1) Zdanie A przekształcamy w logicznie równoważne zdanie A' w *preneksowej postaci normalnej* $Q_1 x_1 \ldots Q_n x_n C$, gdzie C jest formułą otwartą, a $Q_1, \ldots, Q_n \in \{\forall, \exists\}$. A jest logicznie równoważne A', jeżeli:

$$\text{dla każdego } M,\ M \models A \Leftrightarrow M \models A'.$$

(2) Eliminujemy kwantyfikatory istnienia z A'.

Zdanie $\exists x D$ zastępujemy zdaniem $D[x/c]$, gdzie c jest nową stałą.

Zdanie $\forall x_1 \ldots \forall x_n \exists x D$ zastępujemy zdaniem $\forall x_1 \ldots \forall x_n D[x/f(x_1, \ldots, x_n)]$, gdzie f jest nowym symbolem funkcyjnym.

Po skończonej liczbie takich przekształceń otrzymamy zdanie A_u.

Podobnie dla każdego skończonego zbioru zdań S możemy skonstruować skończony zbiór zdań uniwersalnych S_u, spełniający warunek: zbiór S jest spełnialny wtw, gdy zbiór S_u jest spełnialny.

Fakt 5. Dla każdego zdania A następujące warunki są równoważne:

(a) zdanie A jest prawem rachunku predykatów,

(b) zdanie $\neg A$ jest niespełnialne,

(c) zdanie $(\neg A)_u$ jest niespełnialne,

(d) $gr((\neg A)_u) \vdash_{RZ} 0$.

Fakt 6. Niech S będzie skończonym zbiorem zdań, a A zdaniem. Następujące warunki są równoważne:

(a) $S \models_{KRP} A$,

(b) zbiór $S \cup \{\neg A\}$ jest niespełnialny,

(c) zbiór $(S \cup \{\neg A\})_u$ jest niespełnialny,

(d) $gr((S \cup \{\neg A\})_u) \vdash_{RZ} 0$.

Przykłady automatycznego dowodzenia twierdzeń metodą rezolucji zdaniowej.

I. Stałe a, b, c, d, e.

Relacja $R(x, y)$; sens: x jest rodzicem y.

Relacja $F(x)$; sens: x jest kobietą.

$S = \{R(a, b), R(b, c), R(c, d), R(d, e), F(a), F(c), F(e)\}$ (baza danych)

$A = \exists x(F(a) \wedge R(a, x) \wedge R(x, c))$, sens: a jest babcią c.

Wykażemy $S \models_{KRP} A$.

$\neg A$ jest równoważne zdaniu $\forall x(\neg F(a) \vee \neg R(a, x) \vee \neg R(x, c))$,
$(\neg A)_u = \neg A$.

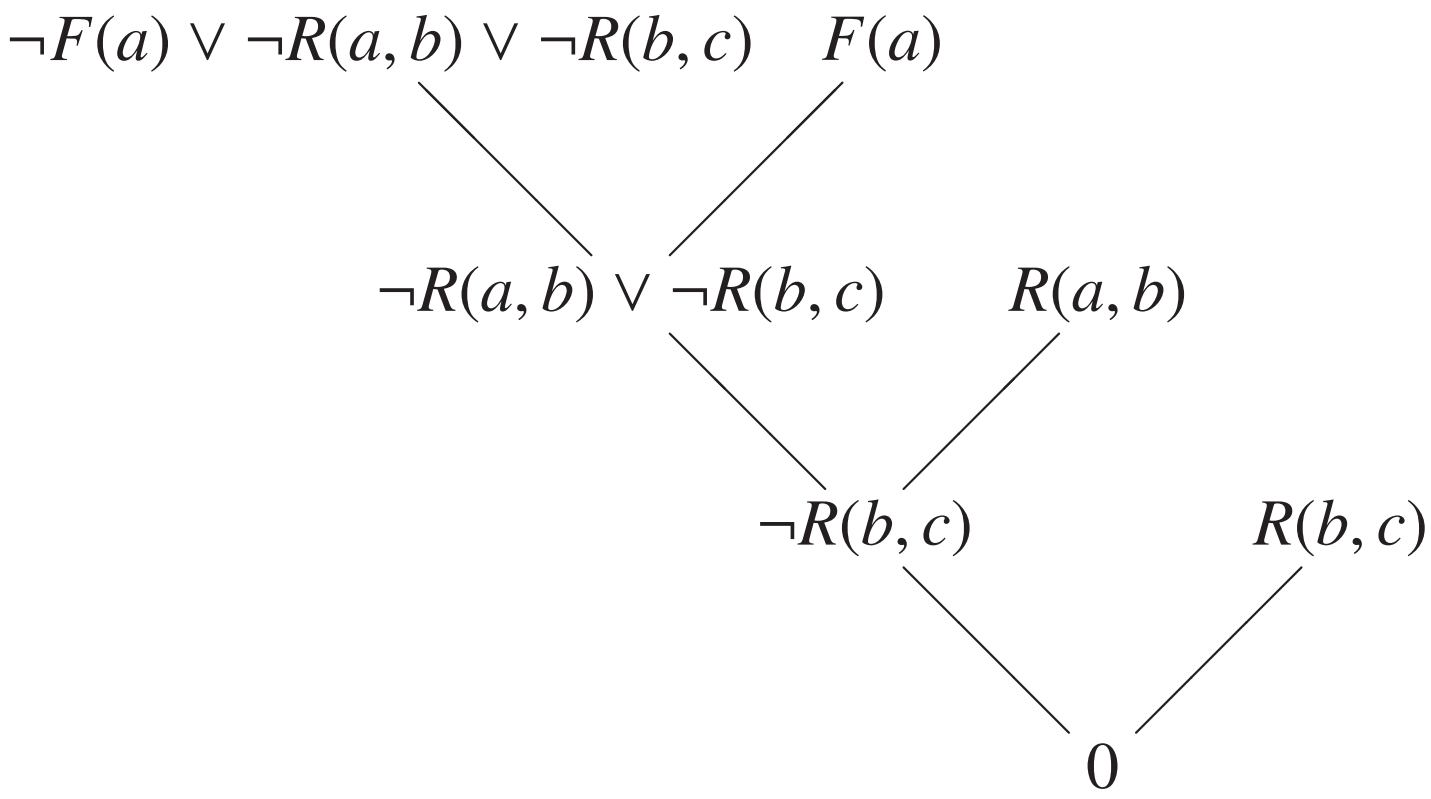
$gr((\neg A)_u)$ zawiera klauzulę: $\neg F(a) \vee \neg R(a, b) \vee \neg R(b, c)$.

Dowód rezolucyjny klauzuli 0 na podstawie

$gr(S \cup \{\neg A\}) = S \cup gr(\neg A)$

1.	$\neg F(a) \vee \neg R(a,b) \vee \neg R(b,c)$	(z $gr(\neg A)$)
2.	$F(a)$	(z S)
3.	$\neg R(a,b) \vee \neg R(b,c)$	(z 1,2)
4.	$R(a,b)$	(z S)
5.	$\neg R(b,c)$	(z 3,4)
6.	$R(b,c)$	(z S)
7.	0	(z 5,6)

Ten sam dowód w postaci drzewa:



II. $B = \exists x \forall y P(x, y) \rightarrow \forall y \exists x P(x, y)$

$\neg B$ jest równoważne zdaniu $\exists x \forall y P(x, y) \wedge \exists y \forall x \neg P(x, y)$.

Sprowadzamy $\neg B$ do preneksowej postaci normalnej.

$\exists x \forall y P(x, y) \wedge \exists u \forall z \neg P(z, u)$

$\exists x \exists u \forall y \forall z (P(x, y) \wedge \neg P(z, u))$

$(\neg B)_u = \forall y \forall z (P(a, y) \wedge \neg P(z, b))$

$gr((\neg B)_u)$ zawiera zdanie: $P(a, b) \wedge \neg P(a, b)$.

Zatem $gr((\neg B)_u) \vdash_{RZ} 0$.

Jeżeli w języku początkowym lub w wyniku skolemizacji pojawiają się symbole funkcyjne, to zbiór ustalonych instancji jest nieskończony; np. za x trzeba podstawiać a, $f(a)$, $f(f(a))$ itd. Opisana procedura daje tylko *algorytm pozytywny*.

Problem spełnialności zdań z dowolnymi kwantyfikatorami i zdań uniwersalnych z symbolami funkcyjnymi jest nierozstrzygalny.

4. Programy Horna. Semantyka deklaratywna.

Literałami nazywamy atomy i negacje atomów, np. $P(a)$, $Q(x, y)$, $\neg P(a)$, $\neg Q(x, y)$.

Klauzula jest to alternatywa skończenie wielu literałów:

$$\neg A_1 \vee \cdots \vee \neg A_n \vee B_1 \vee \cdots \vee B_m$$

gdzie $n, m \geq 0$ a A_i, B_j są atomami.

Powyższa klauzula jest logicznie równoważna implikacji:

$$A_1 \wedge \cdots \wedge A_n \rightarrow B_1 \vee \cdots \vee B_m$$

Klauzula Horna jest to klauzula, zawierająca dokładnie jeden literał pozytywny:

$$\neg A_1 \vee \cdots \vee \neg A_n \vee B \text{ albo } A_1 \wedge \cdots \wedge A_n \rightarrow B$$

Klauzule Horna dzielą się na:

fakty: B albo $\rightarrow B$ albo $B \leftarrow$,

reguły: $A_1 \wedge \cdots \wedge A_n \rightarrow B$ albo $B \leftarrow A_1, \ldots, A_n$,

gdzie $n \geq 1$. B nazywamy nagłówkiem, natomiast listę $A_1, \ldots, A_n$ nazywamy treścią tej reguły. W języku angielskim stosuje się terminy 'head' i 'body'.

W Prologu stosuje się notację : − zamiast $\leftarrow$.

Definicja 1. *Programem Horna* nazywamy skończony zbiór klauzul Horna.

Procedura programu Horna jest to zbiór wszystkich klauzul tego programu, których nagłówek ma ten sam symbol relacyjny. Ta procedura definiuje daną relację. Program Horna można rozumieć jako zbiór procedur.

Przykład. Prosty program definiujący relację dodawania liczb naturalnych suma(X, Y, Z), sens: $X + Y = Z$.

W języku jest stała 0 i jednoargumentowy symbol funkcyjny s - symbol następnika.

Liczby naturalne utożsamiamy z termami $0, s(0), s(s(0)), \ldots$.

Program SUMA zawiera jedną procedurę.

(P1) suma$(0, X, X) \leftarrow$

(P2) suma$(s(X), Y, s(Z)) \leftarrow$ suma(X, Y, Z)

Z tego programu logicznie wynika atom suma$(s(0), s(0), s(s(0)))$.

1. suma$(0, s(0), s(0))$ instancja (P1)
2. suma$(s(0), s(0), s(s(0))) \leftarrow$ suma$(0, s(0), s(0))$ instancja (P2)
3. suma$(s(0), s(0), s(s(0)))$ MP 2,1

Oznaczenie: $L(P)$ - język programu P. Zawiera wszystkie symbole pozalogiczne, występujące w P. Jeżeli $L(P)$ nie zawiera stałych indywiduowych, to dodajemy jedną stałą, żeby $H_{L(P)} \neq \emptyset$.

Aby określić interpretację Herbranda języka $L(P)$, wystarczy określić relacje P^M dla symboli relacyjnych P. Taka relacja jest jednoznacznie wyznaczona przez wszystkie atomy $P(t_1, \ldots, t_n)$ takie, że $P^M(t_1, \ldots, t_n)$, czyli $M \models P(t_1, \ldots, t_n)$. Zatem interpretację Herbranda M języka $L(P)$ możemy utożsamić ze zbiorem $\{A \in B_{L(P)} : M \models A\}$. To może być dowolny podzbiór bazy Herbranda $B_{L(P)}$. W konsekwencji interpretacje Herbranda języka $L(P)$ utożsamiamy z dowolnymi podzbiorami zbioru $B_{L(P)}$; piszemy $M \subseteq B_{L(P)}$. Dla $A \in B_{L(P)}$ mamy: $A \in M \Leftrightarrow M \models A$.

Relację $M_1 \subseteq M_2$ rozumiemy jako relację inkluzji między zbiorami atomów ustalonych.

Oznaczamy $H_P = H_{L(P)}$, $B_P = B_{L(P)}$.

Przykład. $\emptyset$ jest najmniejszą interpretacją Herbranda języka $L(P)$. W tej interpretacji wszystkie relacje są puste i wszystkie atomy z B_P są fałszywe.

B_P jest największą interpretacją Herbranda języka $L(P)$. W tej interpretacji wszystkie relacje są totalne i wszystkie atomy z B_P są prawdziwe.

Definicja 2. Niech P będzie programem Horna. Określamy interpretację Herbranda $M_P = \{A \in B_P : P \models_{KRP} A\}$.

Lemat 1. M_P jest najmniejszym modelem Herbranda programu P.

Dowód. Wykażemy $M_P \models P$.

Niech $B \in P$. Na mocy Faktu 3.4 $B \models_{KRP} B\sigma$ dla dowolnej ustalonej instancji $B\sigma$ faktu B. Stąd $P \models_{KRP} B\sigma$, a więc $B\sigma \in M_P$, czyli $M_P \models B\sigma$ dla każdej ustalonej instancji $B\sigma$ faktu B. Zatem $M_P \models B$ (patrz Def. 3.5).

Niech $B \leftarrow A_1, \ldots, A_n$ należy do P. Aby wykazać, że ta formuła jest prawdziwa w M_P, musimy wykazać, że jej każda ustalona instancja $(B \leftarrow A_1, \ldots, A_n)\sigma$ jest prawdziwa w M_P. Oczywiście ta instancja pokrywa się z $B\sigma \leftarrow A_1\sigma, \ldots, A_n\sigma$.

Zakładamy $M_P \models A_1\sigma \wedge \cdots \wedge A_n\sigma$. Na mocy Def. 3.4, $M_P \models A_i\sigma$ dla każdego $i = 1, \ldots, n$. Stąd $A_i\sigma \in M_P$, czyli $P \models_{KRP} A_i\sigma$ dla każdego $i = 1, \ldots, n$. Wobec tego $P \models_{KRP} A_1\sigma \wedge \cdots \wedge A_n\sigma$. Ponadto:

$$P \models_{KRP} A_1\sigma \wedge \cdots \wedge A_n\sigma \rightarrow B\sigma$$

na mocy Faktu 3.4. Stosując MP, otrzymujemy $P \models_{KRP} B\sigma$, a więc $B\sigma \in M_P$, czyli $M_P \models B\sigma$.

Niech M będzie modelem Herbranda programu P. Niech $A \in M_P$. Wtedy $P \models_{KRP} A$, więc $M \models A$, skoro $M \models P$. Zatem $A \in M$. Wykazaliśmy $M_P \subseteq M$. Q.E.D.

UWAGA. Ograniczenie do klauzul Horna jest istotne. Rozważmy program $\{P(a) \vee P(b)\}$. Dla tego programu $H_L = \{a, b\}$, $B_L = \{P(a), P(b)\}$. Istnieją dokładnie cztery interpretacje Herbranda:

$$M_1 = \emptyset,\ M_2 = \{P(a)\},\ M_3 = \{P(b)\},\ M_4 = B_L$$

M_2, M_3, M_4 są modelami Herbranda tego programu. Żaden z nich nie jest najmniejszym modelem Herbranda; M_2 i M_3 są minimalnymi modelami Herbranda tego programu.

Model M_P stanowi tzw. *semantykę deklaratywną* programu Horna P. Uważamy, że program P opisuje model M_P.

Na przykład, gdy pytamy dla jakich wartości x, y zachodzi $R(x, y)$ na gruncie programu P, to chodzi nam o ustalone termy t_1, t_2 takie, że $R(t_1, t_2)$ jest prawdziwe w M_P.

W przypadku programu SUMA M_P składa się ze wszystkich atomów suma$(s^m(0), s^n(0), s^{m+n}(0))$.

Atomy należące do M_P można generować w następujący sposób.

Określamy ciąg zbiorów $(M_{P,i})_{i\geq 0}$, $M_{P,i} \subseteq B_P$.

$M_{P,0} = \emptyset$

$M_{P,i+1}$ składa się ze wszystkich nagłówków ustalonych instancji klauzul z P, których wszystkie atomy w treści należą do $M_{P,i}$.

W szczególności $M_{P,1}$ składa się ze wszystkich ustalonych instancji faktów z P.

Mamy: $M_{P,i} \subseteq M_{P,i+1}$ dla każdego $i \geq 0$. Dowód stosuje indukcję względem i. Dla $i = 0$ mamy $M_{P,0} = \emptyset \subseteq M_{P,1}$.

Zakładamy $M_{P,i} \subseteq M_{P,i+1}$. Wykażemy $M_{P,i+1} \subseteq M_{P,i+2}$. Niech $A \in M_{P,i+1}$. Wtedy A jest nagłówkiem pewnej ustalonej instancji klauzuli z P, której wszystkie atomy w treści należą do $M_{P,i}$, a więc należą do $M_{P,i+1}$. Zatem $A \in M_{P,i+2}$.

Lemat 2. $M_P = \bigcup_{i\geq 0} M_{P,i}$.

Dowód. Oznaczmy $M = \bigcup_{i \geq 0} M_{P,i}$. Cel: $M_P = M$.

Dowodzimy $M \subseteq M_P$. Wystarczy wykazać $M_{P,i} \subseteq M_P$ dla każdego $i \geq 0$. Stosujemy indukcję względem i.

Dla $i = 0$ mamy $M_{P,0} = \emptyset \subseteq M_P$.

Zakładamy $M_{P,i} \subseteq M_P$. Wykazujemy $M_{P,i+1} \subseteq M_P$. Niech $A \in M_{P,i+1}$. Wtedy istnieje formuła $B\sigma \leftarrow A_1\sigma, \ldots, A_n\sigma$, będąca ustaloną instancją klauzuli $B \leftarrow A_1, \ldots, A_n$ z P ($n \geq 0$), taką że wszystkie atomy $A_j\sigma$ należą do $M_{P,i}$ oraz $A = B\sigma$. Z założenia indukcyjnego $A_j\sigma \in M_P$ dla wszystkich $1 \leq j \leq n$, czyli $M_P \models A_j\sigma$. Ponieważ $M_P \models P$, więc $M_P \models B\sigma \leftarrow A_1\sigma, \ldots, A_n\sigma$. W konsekwencji $M_P \models B\sigma$, czyli $B\sigma \in M_P$. Zatem $A \in M_P$.

Dowodzimy $M_P \subseteq M$. Wystarczy wykazać $M \models P$ i skorzystać z Lematu 1.

Niech B będzie faktem z P. Każda ustalona instancja $B\sigma$ faktu B należy do $M_{P,1}$, więc należy do M, a stąd $M \models B\sigma$. Zatem $M \models B$.

Niech $B \leftarrow A_1, \ldots, A_n$ będzie regułą z P. Niech $B\sigma \leftarrow A_1\sigma, \ldots, A_n\sigma$ będzie ustaloną instancją tej reguły. Wykażemy, że każda taka instancja jest prawdziwa w M.

Zakładamy $M \models A_1\sigma \wedge \cdots \wedge A_n\sigma$. Stąd $M \models A_j\sigma$, czyli $A_j\sigma \in M$ dla wszystkich $1 \leq j \leq n$. Wobec tego istnieją $k_j \geq 0$ takie, że $A_j\sigma \in M_{P,k_j}$ dla $j = 1, \ldots, n$. Niech k będzie największą z liczb k_j. Ponieważ $M_{P,k_j} \subseteq M_{P,k}$, więc $A_j\sigma \in M_{P,k}$ dla wszystkich $j = 1, \ldots, n$. Stąd $B\sigma \in M_{P,k+1}$, a więc $B\sigma \in M$, czyli $M \models B\sigma$.

Wykazaliśmy $M \models B \leftarrow A_1, \ldots, A_n$. Q.E.D.

Dla programu SUMA mamy:

$M_{P,1}$ składa się z atomów suma$(0, s^m(0), s^m(0))$ dla $m \geq 0$,

$M_{P,2}$ dokłada atomy suma$(s(0), s^m(0), s^{m+1}(0))$ dla $m \geq 0$,

$M_{P,i}$ dokłada atomy suma$(s^i(0), s^m(0), s^{m+i}(0))$ dla $m \geq 0$.

Definicja 3. *Zadaniem* (albo: zapytaniem, ang. goal) nazywamy niepustą klauzulę, w której wszystkie literały są negatywne.

$$\neg A_1 \vee \cdots \vee \neg A_m \text{ dla } m \geq 1\,,$$

gdzie $A_1, \ldots, A_m$ są atomami; w innej notacji:

$$\leftarrow A_1, \ldots, A_m$$

Niech $x_1, \ldots, x_n$ będą wszystkimi zmiennymi wolnymi w tej klauzuli.

Sens zapytania: *dla jakich wartości zmiennych* $x_1, \ldots x_n$ *prawdziwe są atomy* $A_1, \ldots, A_m$?

Definicja 4. Niech P będzie programem Horna, a G zadaniem. *Odpowiedzią poprawną* dla $P \cup \{G\}$ nazywamy podstawienie θ, ograniczone do zmiennych w $V(G)$ i takie, że $P \models_{KRP} (A_1 \wedge \cdots \wedge A_m)\theta$.

WYJAŚNIENIA.

Dalsze postępowanie polega na dodaniu zadania do programu P i wyprowadzeniu fałszu logicznego (klauzuli pustej). W ten sposób wykazujemy, że zbiór:

$$P \cup \{\forall x_1 \ldots \forall x_n(\neg A_1 \vee \cdots \neg A_m)\}$$

nie jest spełnialny. W konsekwencji zbiór:

$$P \cup \{\neg\exists x_1 \ldots \exists x_n(A_1 \wedge \cdots \wedge A_m)\}$$

nie jest spełnialny. Na mocy Faktu 2 z rozdziału 1

$$P \models_{KRP} \exists x_1 \ldots \exists x_n(A_1 \wedge \cdots \wedge A_m)$$

Dowód zdania egzystencjalnego $\exists x_1 \ldots \exists x_n(A_1 \wedge \cdots \wedge A_m)$ jest konstruktywny. Znajdujemy podstawienie $\theta = [x_1/t_1, \ldots, x_n/t_n]$ takie, że $P \models_{KRP} (A_1 \wedge \cdots \wedge A_m)\theta$.

5. Uzgadnianie

Termy i atomy nazywamy wyrażeniami prostymi.

Definicja 1. Niech S będzie niepustym zbiorem wyrażeń prostych. Podstawienie σ takie, że $E\sigma = E'\sigma$ dla wszystkich $E, E' \in S$, nazywamy *podstawieniem uzgadniającym* (albo: unifikatorem, ang. unifier) zbioru S. Podstawienie σ nazywamy *najogólniejszym podstawieniem uzgadniającym* (albo: najogólniejszym unifikatorem, ang. most general unifier) zbioru S, jeżeli σ jest podstawieniem uzgadniającym zbioru S oraz dla każdego podstawienia uzgadniającego η zbioru S istnieje podstawienie γ takie, że $\eta = \sigma\gamma$.

Oznaczamy $S\sigma = \{E\sigma : E \in S\}$.

Zamiast 'σ jest unifikatorem zbioru S' mówimy też: σ *unifikuje* (albo: uzgadnia) zbiór S.

Definicja 2. Zbiór S nazywamy *unifikowalnym* (albo: uzgadnialnym), jeżeli istnieje unifikator zbioru S.

Przykład. $S = \{f(x,a), f(g(y,a),z)\}$.

Podstawienie uzgadniające: $\sigma = [x/g(y,a), z/a]$.

Mamy: $S\sigma = \{f(g(y,a),a)\}$.

W dalszym ciągu często utożsamiamy zbiór $S\sigma$ z jedynym elementem tego zbioru, jeżeli σ unifikuje zbiór S.

Inne podstawienie uzgadniające: $\sigma' = [x/g(a,a), y/a, z/a]$.

Mamy: $S\sigma' = \{f(g(a,a),a)\}$.

Wykażemy dalej, że σ jest najogólniejszym unifikatorem zbioru S. σ' nie jest najogólniejszym unifikatorem zbioru S. Mamy $\sigma' = \sigma[y/a]$.

Definicja 3. Wyrażenia proste E_1, E_2 nazywamy *wariantami*, jeżeli istnieją podstawienia σ_1, σ_2 takie, że $E_2 = E_1\sigma_1$ i $E_1 = E_2\sigma_2$.

Definicja 4. Podstawienie σ nazywamy *przemianowaniem zmiennych* w wyrażeniu E, jeżeli σ jest bijekcją zbioru $V(E)$ na zbiór $V(E\sigma)$.

Fakt 1. Jeżeli wyrażenia proste E_1, E_2 są wariantami, to istnieją podstawienia η_1, η_2 takie, że $E_2 = E_1\eta_1$, $E_1 = E_2\eta_2$ oraz η_i jest przemianowaniem zmiennych w E_i dla $i = 1, 2$.

Dowód. Zakładamy, że E_1, E_2 są wariantami. Istnieją podstawienia σ_1, σ_2 takie, że $E_2 = E_1\sigma_1$, $E_1 = E_2\sigma_2$. Niech η_i będzie ograniczeniem podstawienia σ_i do zmiennych z $V(E_i)$.

Mamy $E_2 = E_1\eta_1$, $E_1 = E_2\eta_2$. Stąd $E_1 = E_1\eta_1\eta_2$, $E_2 = E_2\eta_2\eta_1$.

W konsekwencji $x\eta_1\eta_2 = x$ dla każdego $x \in V(E_1)$ oraz $y\eta_2\eta_1 = y$ dla każdego $y \in V(E_2)$. Wobec tego η_1 odwzorowuje $V(E_1)$ w $V(E_2)$ oraz η_2 odwzorowuje $V(E_2)$ w $V(E_1)$. Ponadto odwzorowanie $\eta_1\eta_2$ jest odwzorowaniem identycznościowym na $V(E_1)$; podobnie $\eta_2\eta_1$ jest odwzorowaniem identycznościowym na $V(E_2)$. Zatem η_1 i η_2 są bijekcjami. Q.E.D.

Fakt 2. Jeżeli σ_1, σ_2 są najogólniejszymi unifikatorami zbioru S, to $S\sigma_1$ i $S\sigma_2$ są wariantami.

Definicja 5. Niech $S = \{E_1, \ldots, E_n\}$, $n \geq 2$. Określamy *zbiór niezgodności* zbioru S, oznaczany przez D_S. Traktując wyrażenia E_i jako łańcuchy symboli, znajdujemy najmniejszą pozycję l taką, że istnieją dwa wyrażenia E_i, E_j mające różne symbole na pozycji l. W każdym wyrażeniu z S na pozycji l musi występować symbol znaczący: zmienna, stała, symbol funkcyjny lub symbol relacyjny. Zbiór D_S składa się ze wszystkich podwyrażeń sensownych (termów lub atomów) e_i wyrażeń E_i, które zaczynają się na pozycji l.

Przykład. $S = \{f(x, a), f(g(y, a), z)\}$.

Wyrażenia z tego zbioru mają rózne symbole na trzeciej pozycji, a równe symbole na pozycjach pierwszej i drugiej.

$D_S = \{x, g(y, a)\}$.

Zauważmy, że x jest zmienną, a $g(y, a)$ jest termem nie zawierającym zmiennej x.

Fakt 3. Jeżeli σ unifikuje zbiór S (przynajmniej dwuelementowy), to σ unifikuje zbiór D_S.

Fakt 4. Jeżeli zbiór D_S jest unifikowalny, to zawiera dwa wyrażenia, z których jedno jest zmienną, a drugie jest termem nie zawierającym tej zmiennej.

Dowód. Załóżmy, że σ unifikuje zbiór D_S. Wtedy istnieją różne wyrażenia $e_i, e_j \in D_S$ takie, że $e_i\sigma = e_j\sigma$. Te wyrażenia mają różne symbole na pierwszej pozycji. W następujących przypadkach nie istnieje unifikator zbioru $\{e_i, e_j\}$.

(a) e_i, e_j zaczynają sie od różnych symboli funkcyjnych, (b) e_i, e_j zaczynają się od różnych symboli relacyjnych, (c) jedno z tych wyrażeń jest termem, a drugie atomem, (d) jedno z tych wyrażeń jest stałą, a drugie nie jest zmienną, (e) jedno z tych wyrażeń jest zmienną, a drugie termem zawierającym tę zmienną.

Zatem $\{e_i, e_j\} = \{x, t\}$, przy czym $x \notin V(t)$. Q.E.D.

Algorytm unifikacji. (J. Robinson 1965)

Wejście: $S = \{E_1, \ldots, E_n\}$, $n \geq 1$, E_i - wyrażenia proste.

(0) Przyjmij $\sigma_0 = \varepsilon$. Dla $k = 0, 1, 2, \ldots$ wykonuj:

(1) Wyznacz zbiór $S\sigma_k$. Jeżeli $|S\sigma_k| = 1$, to STOP; wyjście σ_k. Jeżeli $|S\sigma_k| > 1$, to wyznacz $D_{S\sigma_k}$ i przejdź do (2).

(2) Jeżeli nie istnieją $x, t \in D_{S\sigma_k}$ takie, że $x \notin V(t)$, to STOP; wyjście NIE. W przeciwnym razie wybierz taką parę x, t i przyjmij $\sigma_{k+1} = \sigma_k[x/t]$; wróć do (1) przy $k := k + 1$.

Twierdzenie o unifikacji. Dla dowolnego niepustego, skończonego zbioru S algorytm unifikacji kończy pracę w skończonej liczbie kroków. Jeżeli zbiór S jest unifikowalny, to algorytm zwraca najogólniejszy unifikator zbioru S. Jeżeli zbiór S nie jest unifikowalny, to algorytm odpowiada NIE.

Wniosek. Jeżeli niepusty, skończony zbiór S jest unifikowalny, to istnieje najogólniejszy unifikator zbioru S.

Dowód. Niech $S = \{E_1, \ldots, E_n\}$, $n \geq 1$.

(W1) Algorytm uruchomiony na S kończy pracę po skończonej liczbie kroków.

Ponieważ $\sigma_{k+1} = \sigma_k[x/t]$, gdzie t jest podtermem pewnego wyrażenia z $S\sigma_k$, zmienna x występuje w $S\sigma_k$ oraz $x \notin V(t)$, więc w zbiorze $S\sigma_{k+1}$ zmienna x już nie występuje, a wszystkie zmienne występujące w $S\sigma_{k+1}$ występowały w $S\sigma_k$. Stąd $|V(S\sigma_{k+1})| < |V(S\sigma_k)|$. Zatem algorytm produkuje tylko skończenie wiele podstawień $\sigma_0, \sigma_1, \ldots, \sigma_m$, przy czym $m \leq |V(S)|$.

(W2) Jeżeli zbiór S nie jest unifikowalny, to algorytm uruchomiony na S odpowiada NIE.

Załóżmy, że zbiór S nie jest unifikowalny. Na mocy (W1) algorytm uruchomiony na S kończy pracę po skończonej liczbie kroków. Nie może skończyć pracy zgodnie z (1), ponieważ nie istnieje σ takie, że $|S\sigma| = 1$. Zatem kończy pracę zgodnie z (2).

(W3) Jeżeli zbiór S jest unifikowalny, to algorytm uruchomiony na S zwraca najogólniejszy unifikator zbioru S.

Zakładamy, że zbiór S jest unifikowalny. Algorytm produkuje podstawienia $\sigma_0, \sigma_1, \ldots, \sigma_m$. Najpierw wykażemy następującą własność.

(*) Dla każdego unifikatora η zbioru S oraz każdego $k \leq m$ istnieje podstawienie γ_k takie, że $\eta = \sigma_k\gamma_k$.

Dowodzimy (*) przez indukcję względem k. Ustalmy unifikator η zbioru S.

$k = 0$. Przyjmujemy $\gamma_0 = \eta$. Mamy $\eta = \varepsilon\eta = \sigma_0\gamma_0$.

ZI Istnieje γ_k takie, że $\eta = \sigma_k\gamma_k$ $(k < m)$.

TI Istnieje γ_{k+1} takie, że $\eta = \sigma_{k+1}\gamma_{k+1}$.

Skoro $k < m$, to istnieją $x, t \in D_{S\sigma_k}$ takie, że $x \notin V(t)$ i $\sigma_{k+1} = \sigma_k[x/t]$. η unifikuje zbiór S, więc γ_k unifikuje zbiór $S\sigma_k$ i zbiór $D_{S\sigma_k}$. Stąd $x\gamma_k = t\gamma_k$.

Określamy γ_{k+1}.

$x\gamma_{k+1} = x$. $y\gamma_{k+1} = y\gamma_k$ dla y różnych od x.

Wykażemy $\gamma_k = [x/t]\gamma_{k+1}$.

$x[x/t]\gamma_{k+1} = t\gamma_{k+1} = t\gamma_k$ (skoro $x \notin V(t)) = x\gamma_k$

$y[x/t]\gamma_{k+1} = y\gamma_{k+1} = y\gamma_k$

Ostatecznie $\eta = \sigma_k\gamma_k = \sigma_k[x/t]\gamma_{k+1} = \sigma_{k+1}\gamma_{k+1}$. Wykazaliśmy (*).

(**) σ_m jest unifikatorem zbioru S.

Przypuśćmy, że $|S\sigma_m| > 1$. Ponieważ zbiór S jest unifikowalny, więc istnieje unifikator η zbioru S. Na mocy (*) istnieje γ_m takie, że $\eta = \sigma_m\gamma_m$, a więc γ_m unifikuje zbiór $S\sigma_m$, czyli też zbiór $D_{S\sigma_m}$. Na mocy Faktu 4 istnieją $x, t \in D_{S\sigma_m}$ takie, że $x \notin V(t)$.

Zgodnie z (2) algorytm produkuje σ_{m+1}, co jest sprzeczne ze znaczeniem σ_m. Zatem $|S\sigma_m| = 1$.

Na mocy (**) algorytm zwraca σ_m, zgodnie z (1). Na mocy (*) i (**) σ_m jest najogólniejszym unifikatorem zbioru S, co kończy dowód (W3) i całego twierdzenia. Q.E.D.

Przykłady. I. $S = \{f(x,a),\ f(g(y,a),z)\}$.

$\sigma_0 = \varepsilon$. $S\sigma_0 = S$. $D_{S\sigma_0} = \{x, g(y,a)\}$

$\sigma_1 = [x/g(y,a)]$. $S\sigma_1 = \{f(g(y,a),a),\ f(g(y,a),z)\}$, $D_{S\sigma_1} = \{a,z\}$

$\sigma_2 = \sigma_1[z/a] = [x/g(y,a), z/a]$, $S\sigma_2 = \{f(g(y,a),a)\}$

STOP. Wyjście: σ_2.

II. $S = \{f(x,a),\ f(g(y,a),x)\}$.

Jak wyżej $\sigma_1 = [x/g(y,a)]$, więc
$S\sigma_1 = \{f(g(y,a),a), f(g(y,a),g(y,a))\}$.

$D_{S\sigma_1} = \{a, g(y,a)\}$. Zgodnie z (2) algorytm odpowiada NIE.

6. Rezolucja liniowa

Reguła rezolucji liniowej (RRL)

$$\begin{array}{ll} \leftarrow A_1, \ldots, A_{i-1}, A_i, A_{i+1}, \ldots, A_k & \text{(zadanie, } G\text{)} \\ B \leftarrow B_1, \ldots, B_m & \text{(wariant klauzuli z } P\text{)} \\ \hline (\leftarrow A_1, \ldots, A_{i-1}, B_1, \ldots, B_n, A_{i+1}, \ldots, A_k)\sigma & \text{(nowe zadanie, } G'\text{)} \end{array}$$

gdzie σ jest najogólniejszym unifikatorem (mgu) zbioru $\{A_i, B\}$.

Regułę (RRL) stosujemy do zadania G i wariantu C pewnej klauzuli programu P. *Wybrany atom* A_i zadania G uzgadniamy z nagłówkiem klauzuli C, wyznaczając mgu σ. Rezolwentą jest nowe zadanie G'.

Zauważmy, że G' logicznie wynika z G i C w KRP. Tak jest, ponieważ z G wynika $G\sigma$, z C wynika $C\sigma$, a z $G\sigma$ i $C\sigma$ wynika G' na mocy reguły rezolucji zdaniowej (RRZ).

Definicja 1. Niech P będzie programem Horna, a G zadaniem w języku programu P. *Wyprowadzeniem liniowym* dla $P \cup \{G\}$ nazywamy skończony lub nieskończony ciąg $(G_0, C_1, G_1, \ldots, C_n, G_n, \ldots)$, spełniający warunki:

(a) $G_0 = G$,

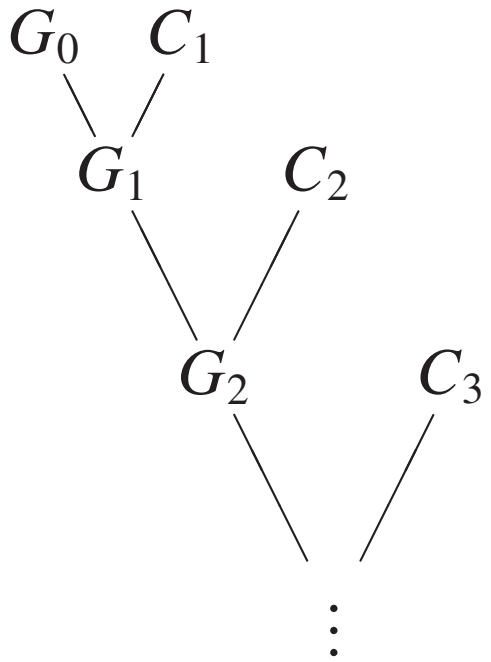
(b) dla każdego $i \geq 1$, jeżeli C_i występuje w ciągu, to C_i jest wariantem klauzuli z P, którego zmienne nie występują w $(G_0, C_1, G_1, \ldots, C_{i-1}, G_{i-1})$,

(c) dla każdego $i \geq 1$, jeżeli G_i występuje w ciągu, to G_i jest rezolwentą klauzul G_{i-1}, C_i na mocy (RRL).

Liczbę n nazywamy *długością* skończonego wyprowadzenia liniowego $(G_0, C_1, G_1, \ldots, C_n, G_n)$.

Definicja 2. Skończone wyprowadzenie liniowe dla $P \cup \{G\}$, kończące się klauzulą pustą, nazywamy *refutacją liniową* dla $P \cup \{G\}$.

Drzewo wyprowadzenia liniowego:



UWAGA. Klauzulę pustą oznaczaliśmy symbolem 0. W przyjętej tu notacji implikacyjnej będziemy oznaczać tę klauzulę symbolem $\leftarrow$.

UWAGA. Wymóg punktu (b) definicji wyprowadzenia liniowego można realizować przez *standaryzację* zmiennych. Przyjmujemy, że klauzula C_i zawiera zmienne indeksowane przez i, np. x_i, y_i, z_i.

Ponadto przyjmujemy, że żadne podstawienia stosowane przed $i-$tym krokiem wyprowadzenia liniowego nie wykorzystują zmiennych występujących w C_i.

Ten wymóg zapobiega pozornej nieuzgadnialności. Na przykład zadanie $\leftarrow P(x)$ i klauzula $P(f(x)) \leftarrow$ nie dadzą rezolwenty, ponieważ zbiór $\{P(x), P(f(x))\}$ nie jest unifikowalny. Konkretny kształt zmiennych w klauzulach programu nie ma znaczenia, więc możemy zastąpić $P(f(x)) \leftarrow$ przez $P(f(x_i)) \leftarrow$. Wtedy wyznaczymy mgu $\sigma = [x/f(x_i)]$.

Definicja 3. Niech $(G_0, C_1, G_1, \ldots, C_n, G_n)$ będzie refutacją liniową dla $P \cup \{G\}$. Niech $\sigma_1, \ldots, \sigma_n$ będą kolejnymi najogólniejszymi unifikatorami wyznaczonymi w tej refutacji. Podstawienie $\sigma_1\sigma_2 \cdots \sigma_n$ ograniczone do zmiennych występujących w G nazywamy *odpowiedzią obliczoną* dla $P \cup \{G\}$ (przez tę refutację).

Przykład. Rozważmy ponownie program SUMA.

$\text{suma}(0, X, X) \leftarrow$

$\text{suma}(s(X), Y, s(Z)) \leftarrow \text{suma}(X, Y, Z)$

Zadanie: $\leftarrow \text{suma}(s(0), s(0), Z)$

Refutacja liniowa dla SUMA $\cup\{\leftarrow \text{suma}(s(0), s(0), Z)\}$:

$G:$	$\leftarrow \text{suma}(s(0), s(0), Z)$	(zadanie)
$C_1:$	$\text{suma}(s(X_1), Y_1, s(Z_1)) \leftarrow \text{suma}(X_1, Y_1, Z_1)$	$\sigma_1 = [X_1/0, Y_1/s(0), Z/s(Z_1)]$
$G_1:$	$\leftarrow \text{suma}(0, s(0), Z_1)$	
$C_2:$	$\text{suma}(0, X_2, X_2) \leftarrow$	$\sigma_2 = [X_2/s(0), Z_1/s(0)]$
$G_2:$	$\leftarrow$	(klauzula pusta)

$Z\sigma_1\sigma_2 = s(Z_1)\sigma_2 = s(s(0))$.

Odpowiedź obliczona: $[Z/s(s(0))]$

Twierdzenie 1 (o poprawności rezolucji liniowej). Każda odpowiedź obliczona dla $P \cup \{G\}$ jest odpowiedzią poprawną dla $P \cup \{G\}$.

Dowód. Niech $(G, C_1, G_1, \ldots, C_n, G_n)$ będzie refutacją liniową dla $P \cup \{G\}$ z kolejnymi mgu $\sigma_1, \ldots, \sigma_n$. Niech σ będzie podstawieniem $\sigma_1 \cdots \sigma_n$ ograniczonym do zmiennych w G. Niech $G = (\leftarrow A_1, \ldots, A_k)$. Wykażemy:

$$(*)\ P \models_{KRP} (A_1 \wedge \cdots \wedge A_k)\sigma$$

przez indukcję względem n.

$n = 1$. Wtedy $k = 1$, $C_1 = (B \leftarrow)$, σ_1 jest mgu zbioru $\{A_1, B\}$. $B\sigma_1$ jest instancją faktu z P, więc $P \models_{KRP} B\sigma_1$. Zatem $P \models_{KRP} A_1\sigma_1$, skoro $A_1\sigma_1 = B\sigma_1$.

ZI: (*) zachodzi dla refutacji długości $n - 1$.

TI: (*) zachodzi dla refutacji długości n, $(n > 1)$.

Niech $C_1 = (B \leftarrow B_1, \ldots, B_m)$ a σ_1 będzie mgu zbioru $\{A_i, B\}$, gdzie A_i jest wybranym atomem z G. Wtedy G_1 jest klauzulą:

$$(\leftarrow A_1, \ldots, A_{i-1}, B_1, \ldots, B_m, A_{i+1}, \ldots, A_k)\sigma_1$$

Ciąg $(G_1, C_2, G_2, \ldots, C_n, G_n)$ jest refutacją liniową długości $n-1$ dla $P \cup \{G_1\}$ z kolejnymi mgu $\sigma_2, \ldots, \sigma_n$. Na mocy ZI:

$$P \models_{KRP} (A_1 \wedge \cdots \wedge B_1 \wedge \cdots \wedge B_m \wedge \cdots \wedge A_k)\sigma$$

skoro $\sigma = \sigma_1(\sigma_2 \cdots \sigma_n)$. Ponieważ:

$$P \models_{KRP} (B_1 \wedge \cdots \wedge B_m \rightarrow B)\sigma$$

więc $P \models_{KRP} (A_1 \wedge \cdots \wedge B \wedge \cdots \wedge A_k)\sigma$, czyli otrzymaliśmy (*), skoro $B\sigma = (A_i)\sigma$. Q.E.D.

Definicja 4. Jeżeli w regule (RRL) σ jest unifikatorem (niekoniecznie najogólniejszym) zbioru $\{A_i, B\}$, to wniosek nazywamy *nieograniczoną rezolwentą* przesłanek tej reguły. *Nieograniczoną refutację liniową* dla $P \cup \{G\}$ określamy jak refutację liniową za wyjątkiem tego, że dla każdego G_i może być nieograniczoną rezolwentą G_{i-1}, C_i dla każdego $i = 1, \ldots, n$.

Lemat 1 (o mgu). Niech $(G_0, C_1, G_1, \ldots, C_n, G_n)$ będzie nieograniczoną refutacją liniową dla $P \cup \{G\}$ z kolejnymi unifikatorami $\theta_1, \ldots, \theta_n$. Wtedy istnieje refutacja liniowa dla $P \cup \{G\}$ postaci $(G_0, C_1, (G_1)', \ldots, C_n, (G_n)')$ z kolejnymi mgu $\sigma_1, \ldots, \sigma_n$ taka, że $\theta_1 \cdots \theta_n = \sigma_1 \cdots \sigma_n \gamma$ dla pewnego podstawienia γ.

Dowód. Indukcja po długości nieograniczonej refutacji liniowej dla $P \cup \{G\}$. Niech $\leftarrow$ powstaje z $G = \leftarrow A$ i $B \leftarrow$ w jednym kroku, przy czym θ_1 jest unifikatorem zbioru $\{A, B\}$. Wtedy istnieją mgu σ_1 tego zbioru i podstawienie γ takie, że $\theta_1 = \sigma_1 \gamma$. Oczywiście $(\leftarrow A, B \leftarrow, \leftarrow)$ jest refutacją liniową dla $P \cup \{G\}$.

Zakładamy, że teza lematu jest prawdziwa dla nieograniczonych refutacji liniowych długości $n-1$ ($n > 1$). Niech $(G_0, C_1, G_1, \ldots, C_n, G_n)$ będzie taką refutacją długości n. Przyjmijmy, że $G_0 = \leftarrow A_1, \ldots, A_k$, $C_1 = B \leftarrow B_1, \ldots, B_m$, θ_1 jest unifikatorem zbioru $\{A_i, B\}$, a $G_1 = (\leftarrow A_1, \ldots, B_1, \ldots, B_m, \ldots, A_k)\theta_1$, gdzie ciąg $B_1, \ldots, B_m$ zajął miejsce A_i w $A_1, \ldots, A_k$.

Wtedy istnieją mgu σ_1 zbioru $\{A_i, B\}$ oraz podstawienie δ takie, że $\theta_1 = \sigma_1\delta$. Możemy przyjąć, że δ nie wykorzystuje zmiennych występujących w C_2. Określamy:

$(G_1)' = (\leftarrow A_1, \ldots, B_1, \ldots, B_m, \ldots, A_k)\sigma_1$.

$((G_1)', C_2, G_2, \ldots, C_n, G_n)$ jest nieograniczoną refutacją liniową dla $P \cup \{(G_1)'\}$ z kolejnymi unifikatorami $\delta\theta_2, \theta_3, \ldots, \theta_n$. Rzeczywiście, ponieważ θ_2 uzgadnia wybrany atom $A\theta_1$ w G_1 z nagłówkiem B' klauzuli C_2, czyli $A\theta_1\theta_2 = B'\theta_2$, więc mamy:

$A\sigma_1\delta\theta_2 = B'\theta_2 = B'\delta\theta_2$, czyli $\delta\theta_2$ uzgadnia $A\sigma_1$ z B'.

Na mocy założenia indukcyjnego, istnieją refutacja liniowa $((G_1)', C_2, (G_2)', \ldots, C_n, (G_n)')$ dla $P \cup \{(G_1)'\}$ z kolejnymi mgu $\sigma_2, \ldots, \sigma_n$ oraz podstawienie γ takie, że $\delta\theta_2 \cdots \theta_n = \sigma_2 \cdots \sigma_n\gamma$.

Wobec tego, $(G_0, C_1, (G_1)', C_2, (G_2)', \ldots, C_n, (G_n)')$ jest refutacją liniową dla $P \cup \{G\}$ z kolejnymi mgu $\sigma_1, \ldots, \sigma_n$. Mamy:

$\theta_1\theta_2 \cdots \theta_n = \sigma_1\delta\theta_2 \cdots \theta_n = \sigma_1\sigma_2 \cdots \sigma_n\gamma$. Q.E.D.

Lemat 2 (o przenoszeniu). Niech θ będzie podstawieniem niewykorzystującym zmiennych standaryzowanych. Niech $(G_0, C_1, G_1, \ldots, C_n, G_n)$ będzie refutacją liniową dla $P \cup \{G\theta\}$ z kolejnymi mgu $\eta_1, \ldots, \eta_n$. Wtedy istnieją refutacja liniowa $(G, C_1, (G_1)', \ldots, C_n, (G_n)')$ dla $P \cup \{G\}$ z kolejnymi mgu $\sigma_1, \ldots, \sigma_n$ oraz podstawienie γ takie, że $\theta\eta_1 \cdots \eta_n = \sigma_1 \cdots \sigma_n\gamma$.

Dowód. Zauważmy, że $(G, C_1, G_1, \ldots, C_n, G_n)$ jest nieograniczoną refutacją liniową dla $P \cup \{G\}$ z kolejnymi unifikatorami $\theta\eta_1, \eta_2, \ldots, \eta_n$. Korzystamy z lematu 1. Q.E.D.

Definicja 5. *Atomem sukcesu* programu P nazywamy ustalony atom $A \in B_P$ taki, że istnieje refutacja liniowa dla $P \cup \{\leftarrow A\}$. S_P oznacza zbiór wszystkich atomów sukcesu programu P.

Twierdzenie 2 (o słabej pełności rezolucji liniowej). $S_P = M_P$ dla dowolnego programu Horna P.

Lemat 3. Niech P będzie programem Horna, a A atomem języka L_P. Jeżeli $P \models_{KRP} A$, to ε jest odpowiedzią obliczoną dla $P \cup \{\leftarrow A\}$.

Twierdzenie 3 (o pełności rezolucji liniowej). Niech θ będzie odpowiedzią poprawną dla $P \cup \{G\}$. Wtedy istnieje odpowiedź obliczona σ dla $P \cup \{G\}$ taka, że $G\theta = G\sigma\gamma$ dla pewnego podstawienia γ.

Nieformalnie: każda odpowiedź poprawna jest instancją pewnej odpowiedzi obliczonej.

Dowody lematu 3 i twierdzeń 2 i 3 podano na wykładzie.

Regułą obliczeniową nazywamy przepis, zgodnie z którym wybieramy atom A_i zadania $\leftarrow A_1, \ldots, A_k$ w kolejnym zastosowaniu reguły (RRL). Regułę obliczeniową nazywamy *jednorodną*, jeżeli wybór atomu A_i zależy tylko od zadania $\leftarrow A_1, \ldots, A_k$ (nie zależy np. od dotychczasowego przebiegu wyprowadzenia). Jednorodną regułę obliczeniową można przedstawić jako funkcję R, która każdemu zadaniu $\leftarrow A_1, \ldots, A_k$ przyporządkowuje liczbę $i \in \{1, \ldots, k\}$. Przykładami jednorodnych reguł obliczeniowych są:

LEFT: $R(\leftarrow A_1, \ldots, A_k) = 1$ (zawsze wybieramy pierwszy atom zadania)

RIGHT: $R(\leftarrow A_1, \ldots, A_k) = k$ (zawsze wybieramy ostatni atom zadania)

W Prologu zwykle implementuje się regułę LEFT.

Wszystkie wyniki tego rozdziału pozostają prawdziwe, jeżeli ograniczymy się do wyprowadzeń liniowych, zgodnych z regułą LEFT.

SLD-rezolucja jest to rezolucja liniowa zgodna z daną regułą obliczeniową.

Przy ustalonej regule obliczeniowej, kolejne zadanie G_i wyprowadzenia liniowego $(G_0, C_1, G_1, \ldots)$ można na ogół wyznaczyć na kilka sposobów, w zależności od klauzuli programu, której nagłówek uzgadniamy z wybranym atomem zadania G_{i-1}.

Prolog traktuje program jako ciąg klauzul i wybiera klauzule w kolejności ich występowania w programie. Kiedy wybór zakończy się porażką, tzn. otrzymamy zadanie, w którym wybrany atom nie jest uzgadnialny z nagłówkiem żadnej klauzuli programu, następuje *nawracanie*: system unieważnia ostatnio wybraną klauzulę, uzgadnialną z odpowiednim atomem A i wybiera kolejną klauzulę programu, uzgadnialną z atomem A; w przypadku, gdy nie znajdzie takiej klauzuli, nawraca do poprzedniego wyboru klauzuli uzgadnialnej i kontynuuje tę procedurę.

Teoretycznym opisem tego postępowania jest SLD-drzewo, przedstawiające wszystkie wyprowadzenia liniowe dla $P \cup \{G\}$, zgodne z daną regułą obliczeniową. Wierzchołki drzewa etykietujemy zadaniami. Etykietą korzenia jest G. Jeżeli G' jest etykietą pewnego wierzchołka, to następniki tego wierzchołka (w porządku horyzontalnym) są etykietowane kolejnymi rezolwentami, które można otrzymać z G' i kolejnych klauzul programu.

Rozważmy program:

$$\begin{aligned}
&\text{(C1):} \quad p(a,b) \leftarrow \\
&\text{(C2):} \quad p(b,c) \leftarrow \\
&\text{(C3):} \quad q(x,y) \leftarrow p(x,y) \\
&\text{(C4):} \quad q(x,y) \leftarrow p(x,z), q(z,y)
\end{aligned}$$

Oznaczmy ten program przez P.

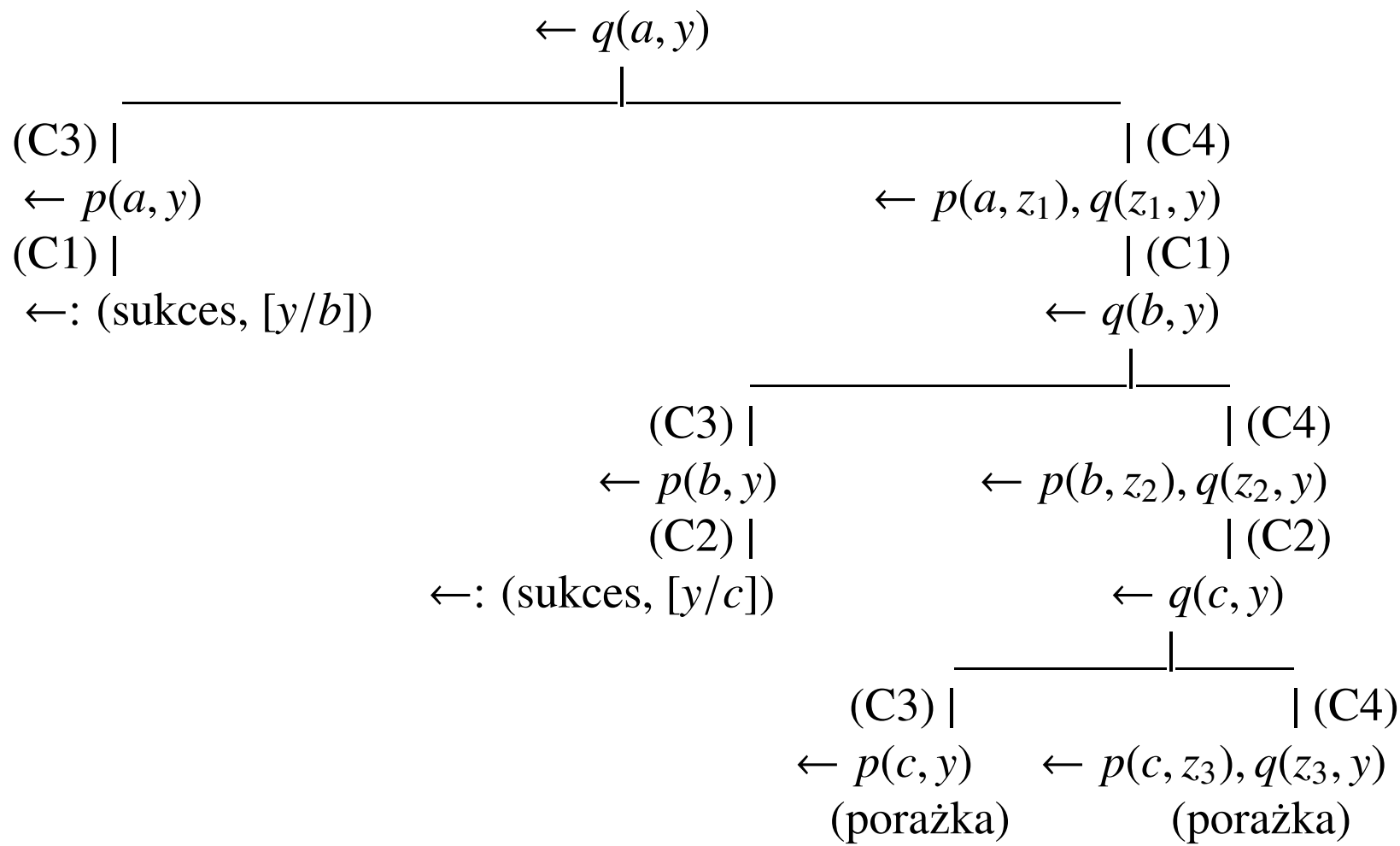
Predykat q jest przechodnim domknięciem predykatu p.

Na przykład dla p interpretowanego $p(x, y)$: *x jest rodzicem y*, interpretacją q będzie $q(x, y)$: *x jest przodkiem y*.

Niech G będzie zadaniem:

$\leftarrow q(a, y)$ (sens: *czyim przodkiem jest a*?).

Oto SLD-drzewo dla $P \cup \{G\}$, zgodne z LEFT.


← q(a, y)
(C3) |
← p(a, y)
(C1) |
←: (sukces, [y/b])
| (C4)
← p(a, z1), q(z1, y)
| (C1)
← q(b, y)
(C3) |
← p(b, y)
(C2) |
←: (sukces, [y/c])
| (C4)
← p(b, z2), q(z2, y)
| (C2)
← q(c, y)
(C3) |
← p(c, y)
(porażka)
| (C4)
← p(c, z3), q(z3, y)
(porażka)

W tym drzewie są dwie gałęzie sukcesu, kończące się klauzulą pustą $\leftarrow$, wyznaczające odpowiedzi obliczone $[y/b]$ i $[y/c]$; pozostałe dwie gałęzie kończą się porażką, tzn. pierwszy atom ostatniego zadania nie jest uzgadnialny z nagłówkiem żadnej klauzuli programu.

System przeszukuje SLD-drzewo metodą przeszukiwania w głąb (ang. *depth-first*), tzn. gałęziami od lewej strony.

Gdyby w programie P zmodyfikowano dwie ostatnie klauzule, przyjmując:

(C3): $q(x, y) \leftarrow q(x, z), p(z, y)$

(C4): $q(x, y) \leftarrow p(x, y)$

to SLD-drzewo dla $P \cup \{G\}$, zgodne z regułą LEFT, miałoby pierwszą (od lewej strony) gałąź nieskończoną. Wtedy system nie znalazłby żadnej odpowiedzi obliczonej.