KLASYCZNY RACHUNEK LOGICZNY:

UNIFIKACJA I REZOLUCJA

(LOGIKA MATEMATYCZNA: WYKŁADY 10, 24, 25)

2007–2008

JERZY POGONOWSKI

ZAKŁAD LOGIKI STOSOWANEJ UAM

http://www.logic.amu.edu.pl

LOGIKA MATEMATYCZNA

WYKŁAD 10: METODA REZOLUCJI W KRZ

10. Dowody rezolucyjne w KRZ

Pokażemy teraz działanie pewnej metody dowodowej, mającej istotne zastosowania m.in. w automatycznym dowodzeniu twierdzeń.

10.1. Przypomnienia i kilka definicji

Przypomnijmy, że:

Literałami nazywamy zmienne zdaniowe oraz negacje zmiennych zdaniowych.
Literałem *komplementarnym* do literału ℓ nazywamy literał $\overline{\ell}$, zdefiniowany następująco:

- jeśli ℓ jest literałem pozytywnym ℓ', to $\overline{\ell}$ jest literałem negatywnym $\neg\ell'$
- jeśli ℓ jest literałem negatywnym $\neg\ell'$, to $\overline{\ell}$ jest literałem pozytywnym ℓ'.

- *Koniunkcją elementarną* nazwiemy dowolną koniunkcję literałów.
- *Alternatywą elementarną* nazwiemy dowolną alternatywę literałów.
- *Alternatywną postacią normalną* (*apn*) nazwiemy dowolną alternatywę koniunkcji elementarnych.
- *Koniunkcyjną postacią normalną* (*kpn*) nazwiemy dowolną koniunkcję alternatyw elementarnych.
- Apn (odpowiednio: kpn) α nazywamy *istotną* i oznaczamy *iapn* (odpowiednio: *ikpn*), jeśli każda zmienna zdaniowa formuły α występuje w każdej elementarnej koniunkcji (odpowiednio: alternatywie) dokładnie raz, zaprzeczona bądź niezaprzeczona.
- Każdą apn (odpowiednio: kpn, iapn, ikpn) semantycznie równoważną danej formule α nazywamy *apn* (odpowiednio: *kpn, iapn, ikpn*) *formuły* α.

Pamiętamy, że każda formuła języka KRZ jest:

- semantycznie równoważna z pewną formułą w koniunkcyjnej postaci normalnej;
- semantycznie równoważna z pewną formułą w alternatywnej postaci normalnej;
- inferencyjnie równoważna z pewną formułą w koniunkcyjnej postaci normalnej;
- inferencyjnie równoważna z pewną formułą w alternatywnej postaci normalnej.

Pokazanie, że zachodzą równoważności, o których mowa powyżej, ma charakter ***algorytmiczny.***

Zachodzą implikacje:

- Jeśli α jest tautologią KRZ oraz $\alpha \sim \beta$ (α semantycznie równoważna z β), to także β jest tautologią KRZ.
- Jeśli α jest tezą KRZ oraz $\alpha \approx \beta$ (α inferencyjnie równoważna z β), to także β jest tezą KRZ.

Jeśli α jest kpn, to jest postaci: $\alpha_1 \wedge \alpha_2 \wedge \ldots \wedge \alpha_n$, gdzie każda formuła α_i jest alternatywą elementarną postaci:

$$\ell_1^i \vee \ell_2^i \vee \ldots \vee \ell_{m_i}^i,$$

gdzie z kolei każda formuła ℓ_j^i jest literałem.

Koniunkcja $\alpha_1 \wedge \alpha_2 \wedge \ldots \wedge \alpha_n$ jest tautologią KRZ wtedy i tylko wtedy, gdy wszystkie formuły α_i są tautologiami.

Formuła α_i (czyli formuła $\ell_1^i \vee \ell_2^i \vee \ldots \vee \ell_{m_i}^i$) jest tautologią KRZ wtedy i tylko wtedy, gdy wśród ℓ_1^i, ℓ_2^i, …, $\ell_{m_i}^i$ występuje co najmniej jedna para literałów komplementarnych.

Klauzulą nazwiemy dowolny skończony zbiór literałów.

Klauzule odpowiadają alternatywom elementarnym. Tak więc, jeśli $\ell_1 \vee \ell_2 \vee \ldots \vee \ell_n$ jest alternatywą elementarną, to odpowiadająca jej klauzula jest zbiorem $\{\ell_1, \ell_2, \ldots, \ell_n\}$. Umawiamy się, że literały, które (ewentualnie) występują więcej niż raz w danej alternatywie elementarnej zapisujemy tylko raz w odpowiadającej jej klauzuli. Ponieważ $(\alpha \vee \alpha) \equiv \alpha$ jest tezą KRZ, umowa ta niczego nie „psuje".

Zbiory klauzul są więc rodzinami zbiorów literałów. Każdej formule w kpn odpowiada pewien zbiór klauzul. Jeśli α jest kpn, to jest postaci: $\alpha_1 \wedge \alpha_2 \wedge \ldots \wedge \alpha_n$, gdzie każda formuła α_i jest alternatywą elementarną postaci:

$$\ell_1^i \vee \ell_2^i \vee \ldots \vee \ell_{m_i}^i,$$

gdzie z kolei każda formuła ℓ_j^i jest literałem. Formule α odpowiada wtedy zbiór klauzul:

$$\{\{\ell_1^1, \ell_2^1, \ldots, \ell_{m_1}^1\}, \{\ell_1^2, \ell_2^2, \ldots, \ell_{m_2}^2\}, \ldots, \{\ell_1^n, \ell_2^n, \ldots, \ell_{m_n}^n\}\}.$$

Umawiamy się, że alternatywy elementarne, które (ewentualnie) występują więcej niż raz w danej koniunkcyjnej postaci normalnej zapisujemy tylko raz w odpowiadającej jej rodzinie zbiorów. Również ta umowa jest poprawna.

Dla przykładu, formule w koniunkcyjnej postaci normalnej:

$$(p_1 \vee p_2 \vee \neg p_3) \wedge (p_3 \vee p_4) \wedge \neg p_1 \wedge (\neg p_2 \vee \neg p_4)$$

odpowiada następujący zbiór klauzul:

$$\{\{p_1, p_2, \neg p_3\}, \{p_3, p_4\}, \{\neg p_1\}, \{\neg p_2, \neg p_4\}\}.$$

Mówienie „formule α w kpn odpowiada zbiór klauzul S" oraz „zbiór klauzul S reprezentuje formułę α w kpn" jest nieco rozwlekłe. Można wprowadzić jakiś symbol relacyjny, powiedzmy $\rightleftharpoons$, pozwalający na skrótowe zapisywanie takich wypowiedzi:

- $\alpha \rightleftharpoons S$ czytamy: „formule α w kpn odpowiada zbiór klauzul S" ***lub, równoznacznie***
- $\alpha \rightleftharpoons S$ czytamy: „zbiór klauzul S reprezentuje formułę α w kpn".

Symbol $\rightleftharpoons$ należy oczywiście do metajęzyka.

Klauzulę pustą (nie zawierającą żadnych elementów) oznaczamy przez $\square$.

Klauzule zawierające najwyżej jeden literał pozytywny nazywamy ***klauzulami Hornowskimi***.

Są zatem trzy możliwości dla klauzuli Hornowskiej C:

- (a) C zawiera tylko literały negatywne;
- (b) C zawiera dokładnie jeden literał pozytywny i żadnych negatywnych;
- (c) C zawiera dokładnie jeden literał pozytywny oraz literały negatywne.

Będziemy posługiwać się ***stałymi zdaniowymi***:

- *falsum* $\bot$, reprezentującą dowolną kontrtautologię;
- *verum* $\top$, reprezentującą dowolną tautologię.

Zauważmy, że klauzula Hornowska:

- (a) postaci $\{\neg p_1, \neg p_2, \ldots, \neg p_n\}$ jest inferencyjnie równoważna formule:
  $(p_1 \wedge p_2 \wedge \ldots \wedge p_n) \rightarrow \bot$
- (b) postaci $\{p_1\}$ jest inferencyjnie równoważna formule: $\top \rightarrow p_1$
- (c) postaci $\{\neg p_1, \neg p_2, \ldots, \neg p_n, p_{n+1}\}$ jest inferencyjnie równoważna formule:
  $(p_1 \wedge p_2 \wedge \ldots \wedge p_n) \rightarrow p_{n+1}$.

Fakt powyższy można wykorzystać do podania ***szybkiego*** algorytmu dla sprawdzania, czy klauzula Hornowska odpowiada spełnialnej formule języka KRZ.

Przypomnijmy, że formuła α jest:

- *tautologią*, gdy ma wartość 1 przy *każdym* wzz;
- *kontrtautologią*, gdy ma wartość 0 przy *każdym* wzz;
- *spełnialna*, gdy ma wartość 1 przy *co najmniej jednym* wzz;
- *odrzucalna*, gdy ma wartość 0 przy *co najmniej jednym* wzz.

Tak więc, formuła α:

- jest tautologią, gdy nie jest odrzucalna;
- jest kontrtautologią, gdy nie jest spełnialna.

Ponadto, mamy oczywiście:

- α jest tautologią wtedy i tylko wtedy, gdy $\neg\alpha$ nie jest spełnialna.
- α jest kontrtautologią wtedy i tylko wtedy, gdy $\neg\alpha$ nie jest odrzucalna.

Pamiętamy, że algorytm ustalania, czy dana formuła języka KRZ jest tautologią ma złożoność wykładniczą: aby sprawdzić, czy formuła o n zmiennych zdaniowych jest tautologią KRZ trzeba sprawdzić, jaka jest jej wartość dla 2^n wzz.

Na mocy Twierdzenia o Pełności KRZ, jeśli formuła α ***nie jest*** spełnialna, to możemy to wykazać na drodze dedukcyjnej,

- pokazując, że: $\emptyset \vdash_{krz} \neg\alpha$ ***lub***
- pokazując, że: $\vdash_{jas} \neg\alpha$.

Nie możemy jednak, ani używając konsekwencji $\vdash_{krz}$, ani używając konsekwencji $\vdash_{jas}$ pokazać, że jakaś formuła ***jest*** spełnialna.
Podobnie, jeśli α wynika logicznie z X (czyli jeśli zachodzi $X \models_{KRZ} \alpha$), to możemy to wykazać,

- pokazując, że: $X \vdash_{krz} \alpha$ ***lub***
- pokazując, że: $X \vdash_{jas} \alpha$.

Jeśli jednak $X \nvdash \alpha$, (czyli gdy przy ***co najmniej jednym*** wartościowaniu h, $h[X] \subseteq \{1\}$ oraz $h(\alpha) = 0$), to nie mamy możliwości przedstawienia dowodu (w terminach konsekwencji $\vdash_{krz}$ lub $\vdash_{jas}$), że ***istnieje*** wartościowanie h takie, że $h[X] \subseteq \{1\}$ oraz $h(\alpha) = 0$.

Reguła rezolucji, którą omówimy za chwilę, dostarcza możliwości wykazywania środkami czysto syntaktycznymi, że dana formuła nie jest spełnialna.

10.2. Reguła rezolucji

DEFINICJA 10.2.1.

Niech C_1 i C_2 będą klauzulami i niech literał ℓ występuje w C_1, a literał $\overline{\ell}$ występuje w C_2. Wtedy każdą klauzulę postaci:

$$(C_1 - \{\ell\}) \cup (C_2 - \{\overline{\ell}\})$$

nazywamy ***rezolwentą*** klauzul C_1 i C_2. Zamiast *rezolwenta* używa się też terminu: ***rezolwent***. Logice jest oczywiście obojętny rodzaj gramatyczny. Jeśli C_1 i C_2 są powyższej postaci, to mówimy też, że C_1 i C_2 ***kolidują*** ze względu na literały ℓ oraz $\overline{\ell}$.

PRZYKŁAD 10.2.1.

Niech:

- $C_1 = \{p_1, \neg p_2, p_3\}$
- $C_2 = \{p_2, \neg p_3, p_4\}$.

Widać, że C_1 i C_2 kolidują ze względu na następujące pary literałów komplementarnych:

- (a) $(\neg p_2, p_2)$,
- (b) $(p_3, \neg p_3)$.

Wtedy rezolwentami C_1 i C_2 są klauzule:

- (a) $\{p_1, p_3, \neg p_3, p_4\}$
- (b) $\{p_1, p_2, \neg p_2, p_4\}$.

DEFINICJA 10.2.2.

(i) ***Dowodem rezolucyjnym*** klauzuli C ze zbioru klauzul S nazywamy każdy skończony ciąg klauzul $C_1, \ldots, C_n$ taki, że:

- C jest identyczna z C_n
- każda klauzula C_i $(1 \leqslant i \leqslant n)$ jest albo elementem zbioru S albo rezolwentą pewnych klauzul C_j oraz C_k dla $j, k < i$.

(ii) Jeśli istnieje dowód rezolucyjny C z S, to mówimy, że C jest ***rezolucyjnie dowodliwa*** (lub: ***rezolucyjnie wyprowadzalna***) z S i oznaczamy ten fakt przez $S \vdash_{res} C$.

(iii) Każdy dowód rezolucyjny klauzuli pustej $\square$ ze zbioru S nazywamy ***rezolucyjną refutacją*** S. Jeżeli istnieje rezolucyjna refutacja S, to mówimy, że S jest ***rezolucyjnie odrzucalny*** i oznaczamy ten fakt przez $S \vdash_{res} \square$.

(iv) Dla dowolnego zbioru klauzul S niech $res(S)$ będzie zbiorem wszystkich rezolwent wszystkich par elementów S. Zdefiniujmy:

- $res_0(S) = S$
- $res_n = res_{n-1}(S) \cup res(res_{n-1}(S))$ dla $n > 0$
- $\mathcal{R}(S) = \bigcup\{res_n(S) : n \in \mathcal{N}\}$.

Zbiór $\mathcal{R}(S)$ nazywamy ***domknięciem rezolucyjnym*** zbioru S.

(v) ***Rezolucyjnym drzewem dowodowym*** klauzuli C ze zbioru klauzul S nazywamy każde drzewo binarne T o następujących własnościach:

- korzeniem T jest C
- liśćmi T są pewne elementy zbioru S
- bezpośrednimi następnikami wierzchołka D niebędącego liściem są klauzule D_1 oraz D_2, których rezolwentą jest D.

Uwaga. Rozważamy drzewa, których wierzchołki są znakowane ***zbiorami*** literałów.

Uwaga. Dla dowolnego skończonego zbioru klauzul S istnieje liczba naturalna m taka, że $res_{m+1} = res_m$, czyli wszystkie wyrazy ciągu $res_n(S)$ są od pewnego miejsca identyczne. Jest to prosta konsekwencja faktu, że każdy skończony zbiór klauzul S zawiera tylko skończenie wiele literałów.

Uwaga. Nietrudno sprawdzić (korzystając z indukcji po długości dowodu rezolucyjnego), że zachodzi następująca równoważnośc:

- Istnieje rezolucyjne drzewo dowodowe dla C z S wtedy i tylko wtedy, gdy C jest rezolucyjnie dowodliwa z S, czyli gdy $S \vdash_{res} C$.

Uwaga. Często mówi się o dowodach rezolucyjnych *formuł* ze zbiorów *formuł*. Rozumiemy przez to, że wszystkie brane pod uwagę formuły:

- (1) zostały przekształcone do równoważnych im inferencyjnie kpn;
- (2) zostały zastąpione (przy uwzględnieniu (1)) odpowiadającymi im zbiorami klauzul.

Wtedy oczywiście należy powiedzieć, co rozumiemy przez dowód rezolucyjny ***zbioru klauzul*** ze ***zbiorów zbiorów*** klauzul. Jeśli piszemy skrótowo $S \vdash_{res} \alpha$, gdzie S jest zbiorem *formuł*, a α jest *formułą* to rozumiemy przez to, że:

- α została zastąpiona przez swoją kpn, a ta z kolei przez odpowiedni zbiór klauzul,
- każda formuła $\beta \in S$ została zastąpiona przez swoją kpn, a ta z kolei przez odpowiedni zbiór klauzul,
- $S \vdash_{res} \alpha$ oznacza, że ***każda*** klauzula występująca w zbiorze klauzul odpowiadającym kpn formuły α ma dowód rezolucyjny ze zbioru klauzul odpowiadającemu ***koniunkcji*** pewnych formuł z S.

Uwaga. Możemy rozważać ***dowolne*** zbiory klauzul jako poprzedniki relacji $\vdash_{res}$. Z Twierdzenia o Zwartości (zobacz Dodatek 4) oraz z Twierdzeń o Trafności i Pełności metody rezolucyjnej (które udowodnimy za chwilę) wynika, że jeśli $S \vdash_{res} \alpha$, to istnieje ***skończony*** zbiór $S' \subseteq S$ taki, że $S' \vdash_{res} \alpha$.

Koniec uwag.

PRZYKŁAD 10.2.2.

Niech $S = \{p_1 \to p_2, p_2 \to p_3, p_1, \neg p_3\}$ i niech $\bigwedge S$ będzie koniunkcją wszystkich formuł ze zbioru S. Pokażemy, że $\bigwedge S \vdash_{res} \square$. Formuła $\bigwedge S$ ma następującą kpn:

$$(\neg p_1 \vee p_2) \wedge (\neg p_2 \vee p_3) \wedge p_1 \wedge \neg p_3.$$

Odpowiada jej zatem zbiór klauzul:

$$\{\{\neg p_1, p_2\}, \{\neg p_2, p_3\}, \{p_1\}, \{\neg p_3\}\}.$$

A oto zapowiadany dowód rezolucyjny:

1.	$\{\neg p_1, p_2\}$	przesłanka
2.	$\{\neg p_2, p_3\}$	przesłanka
3.	$\{p_1\}$	przesłanka
4.	$\{\neg p_3\}$	przesłanka
5.	$\{\neg p_1, p_3\}$	rezolwenta (1) i (2)
6.	$\{p_3\}$	rezolwenta (3) i (5)
7.	$\square$	rezolwenta (4) i (6).

Zwykle takie dowody rezolucyjne zapisuje się w poniższej postaci:

1.	$\neg p_1 \vee p_2$	przesłanka
2.	$\neg p_2 \vee p_3$	przesłanka
3.	p_1	przesłanka
4.	$\neg p_3$	przesłanka
5.	$\neg p_1 \vee p_3$	rezolwenta (1) i (2)
6.	p_3	rezolwenta (3) i (5)
7.	$\square$	rezolwenta (4) i (6).

Informatycy stosują inne jeszcze skróty notacyjne, czym nie będziemy się tutaj przejmować.

Zauważmy, że $\{p_1 \to p_2, p_2 \to p_3, p_1\} \models_{KRZ} p_3$, co oznacza, że zbiór

$$\{p_1 \to p_2, p_2 \to p_3, p_1, \neg p_3\}$$

nie jest spełnialny (nie istnieje wartościowanie, przy którym wszystkie elementy tego zbioru mają wartość 1). Pokażemy za chwilę, że zbiór klauzul S jest rezolucyjnie odrzucalny dokładnie wtedy, gdy nie jest spełnialna formuła, której kpn odpowiada (skończonemu podzbiorowi) S.

PRZYKŁAD 10.2.3.

Pokażemy, że zbiór formuł

$$S = \{p_1 \to (\neg p_2 \vee (p_3 \wedge p_4)), p_1, p_2, \neg p_4\}$$

jest rezolucyjnie odrzucalny. Tworzymy koniunkcję $\bigwedge S$ wszystkich formuł z S:

$$(p_1 \to (\neg p_2 \vee (p_3 \wedge p_4))) \wedge p_1 \wedge p_2 \wedge \neg p_4,$$

a po przekształceniu tej formuły do kpn tworzymy odpowiadający jej zbiór klauzul:

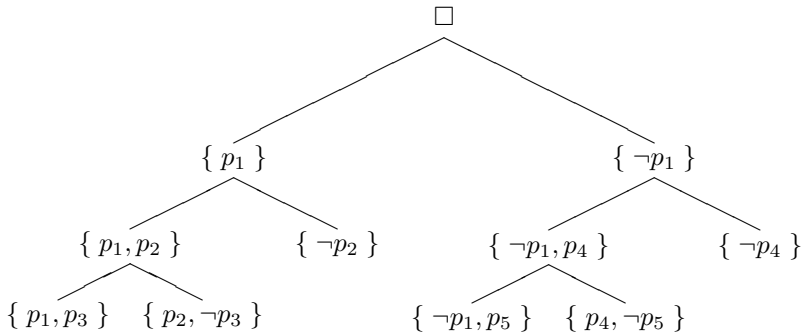
$$\{\{\neg p_1, \neg p_2, p_3\}, \{\neg p_1, \neg p_2, p_4\}, \{p_1\}, \{p_2\}, \{\neg p_4\}\}.$$

Dowód rezolucyjny zapiszemy korzystając z uproszczenia notacji zastosowanego w poprzednim przykładzie:

1.	$\neg p_1 \vee \neg p_2 \vee p_3$	przesłanka
2.	$\neg p_1 \vee \neg p_2 \vee p_4$	przesłanka
3.	p_1	przesłanka
4.	p_2	przesłanka
5.	$\neg p_4$	przesłanka
6.	$\neg p_1 \vee \neg p_2$	rezolwenta 2 i 5
7.	$\neg p_1$	rezolwenta 4 i 6
8.	$\square$	rezolwenta 3 i 7.

PRZYKŁAD 10.2.4.

Niech $S = \{\{p_1, p_3\}, \{p_2, \neg p_3\}, \{\neg p_2\}, \{\neg p_1, p_5\}, \{\neg p_4\}, \{p_4, \neg p_5\}\}$ będzie zbiorem klauzul. Poniższe drzewo jest rezolucyjnym drzewem dowodowym klauzuli $\square$ ze zbioru S:



Skoro $S \vdash_{res} \Box$, to zbiór S jest rezolucyjnie odrzucalny. Zauważmy, że:

$$\{p_1 \rightarrow p_3, p_5 \rightarrow p_4, p_3 \rightarrow p_2, \neg p_2, \neg p_4\} \models_{KRZ} \neg(p_1 \wedge p_3).$$

PRZYKŁAD 10.2.5.

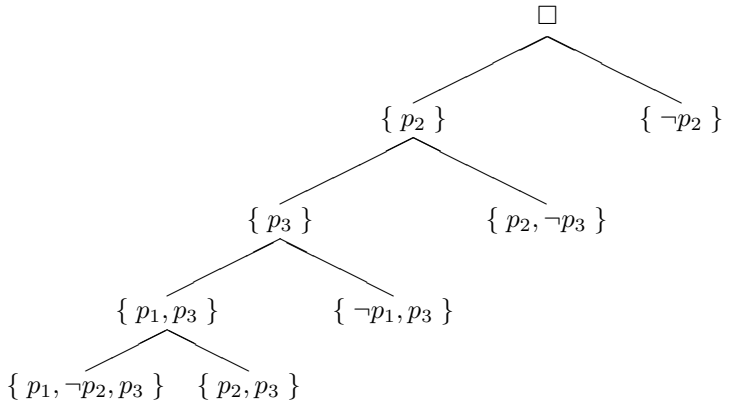
Pokażemy, że ze zbioru:

$$\{\{p_1, \neg p_2, p_3\}, \{p_2, p_3\}, \{\neg p_1, p_3\}, \{p_2, \neg p_3\}, \{\neg p_2\}\}$$

wyprowadzić można klauzulę pustą $\Box$.

1.	$\{p_1, \neg p_2, p_3\}$	przesłanka
2.	$\{p_2, p_3\}$	przesłanka
3.	$\{\neg p_1, p_3\}$	przesłanka
4.	$\{p_2, \neg p_3\}$	przesłanka
5.	$\{\neg p_2\}$	przesłanka
6.	$\{p_1, p_3\}$	rezolwenta 1 i 2
7.	$\{p_3\}$	rezolwenta 6 i 3
8.	$\{p_2\}$	rezolwenta 7 i 4
9.	$\Box$	rezolwenta 8 i 5.

Powyższe wyprowadzenie reprezentowane jest przez następujące rezolucyjne drzewo dowodowe:



Powyższe przykłady pokazują, że stosowanie reguły rezolucji jest banalnie proste. Mogą więc skłaniać do (pochopnej!) konkluzji, że reguła rezolucji może zastąpić wszelkie skomplikowane techniki dowodowe (metodę aksjomatyczną, dedukcję naturalną, itd.). Rzecz ma się następująco. Owszem, reguła rezolucji nie jest skomplikowana i — jak pokażemy za chwilę — jest trafna i pełna. Jednak owa prostota ma też swoją cenę: zbiory klauzul odpowiadają formułom w koniunkcyjnych postaciach normalnych, i choć istnieje algorytm znajdowania dla każdej formuły równoważnej jej inferencyjnie formuły w kpn, to postępowanie wedle jego zaleceń jest dla Człowieka wielce czasochłonne. Inaczej rzecz się ma z maszynami liczącymi, które stosunkowo szybko znajdują kpn, a potem przeprowadzają dowody rezolucyjne.

Tak więc, nie ma ucieczki: choć bezmyślną pracę można powierzyć Maszynom, to praca twórcza (np. znajdowanie dowodów) stale należy do Człowieka.

10.3. Trafność metody rezolucji

Trzeba pokazać, że metoda rezolucji jest ***trafna***, tj. pokazać, że jeśli klauzula pusta należy do rezolucyjnego domknięcia zbioru S, to S nie jest spełnialny.

Można to uczynić na dwa sposoby:

- (A) czysto syntaktyczny (odwołujący się do relacji konsekwencji $\vdash_{krz}$ lub $\vdash_{jas}$), a potem skorzystać z Twierdzenia o Pełności KRZ;

- (B) semantyczny, tj. odwołując się bezpośrednio do relacji $\models_{KRZ}$ wynikania logicznego w KRZ.

Ponieważ czujemy, że Audytorium tego oczekuje, pokażemy oba sposoby.

Sposób (A)

Wykorzystamy relację konsekwencji założeniowej $\vdash_{jas}$.
W Dodatku 4 udowodniono, że relacja konsekwencji $\vdash_{jas}$ ma podobne własności, jak relacja $\vdash_{krz}$, tj. pokazano, że:

Wniosek 8.1.

Dla dowolnych zbiorów formuł X, Y, Z oraz dowolnej formuły α zachodzą następujące warunki:

- (1) $\vdash_{jas}$ jest zwrotna: $X \vdash_{jas} X$
- (2) $\vdash_{jas}$ jest przechodnia: jeśli $X \vdash_{jas} Y$ oraz $Y \vdash_{jas} Z$, to $X \vdash_{jas} Z$
- (3) $\vdash_{jas}$ jest monotoniczna względem pierwszego argumentu:
  jeśli $X \vdash_{jas} Y$ oraz $X \subseteq Z$, to $Z \vdash_{jas} Y$
- (4) $\vdash_{jas}$ jest antymonotoniczna względem drugiego argumentu:
  jeśli $X \vdash_{jas} Y$ oraz $Z \subseteq Y$, to $X \vdash_{jas} Z$.

Tu potrzebne będą nam warunki zwrotności, przechodniości oraz monotoniczności relacji $\vdash_{jas}$.

Teraz możemy pokazać, że tworzenie dowodów rezolucyjnych można reprezentować w systemie dedukcji naturalnej (w systemie założeniowym) KRZ:

TWIERDZENIE 10.3.1.

Jeśli R jest rezolwentą klauzul C_1 i C_2, oraz $\vdash_{jas} C_1$ i $\vdash_{jas} C_2$, to $\vdash_{jas} R$. W konsekwencji, $\{C_1, C_2\} \vdash_{jas} R$.

DOWÓD.

Uwaga. Zapis $\{C_1, C_2\} \vdash_{jas} R$ rozumiemy w ten sposób, że relacja $\vdash_{jas}$ zachodzi pomiędzy kpn reprezentowaną przez zbiór $\{C_1, C_2\}$, a alternatywą elementarną reprezentowaną przez klauzulę R.

Po tym wyjaśnieniu, możemy już przystąpić do dowodu. Jeśli R jest rezolwentą C_1 i C_2, to istnieje zmienna zdaniowa p taka, że p jest literałem w jednej z klauzul C_1 i C_2, a $\neg p$ literałem w pozostałej z klauzul C_1 i C_2. Niech np. p będzie literałem występującym w C_1, a $\neg p$ niech występuje w C_2. Zatem klauzula C_1 reprezentuje alternatywę elementarną $\alpha \vee p$, a klauzula C_2 reprezentuje alternatywę elementarną $\alpha \vee \neg p$, dla pewnych alternatyw elementarnych α i β. Ponieważ R jest rezolwentą C_1 i C_2 (względem pary kolidujących literałów p i $\neg p$), to R reprezentuje alternatywę elementarną $\alpha \vee \beta$. Pokażemy, że: jeśli $\vdash_{jas} p \vee \alpha$ i $\vdash_{jas} \neg p \vee \beta$, to $\vdash_{jas} \alpha \vee \beta$.

1.	$\vdash_{jas} p \vee \alpha$	założenie
2.	$\vdash_{jas} \neg p \vee \beta$	założenie
3.	$\{\neg p\} \vdash_{jas} p \vee \alpha$	1, monotoniczność $\vdash_{jas}$
4.	$\{\neg p\} \vdash_{jas} \neg p$	zwrotność $\vdash_{jas}$
5.	$\{\neg p\} \vdash_{jas} \alpha$	OA: 4,3
6.	$\{\neg p\} \vdash_{jas} \alpha \vee \beta$	DA: 5
7.	$\{\neg\neg p\} \vdash_{jas} \neg p \vee \beta$	2, monotoniczność $\vdash_{jas}$
8.	$\{\neg\neg p\} \vdash_{jas} \neg\neg p$	zwrotność $\vdash_{jas}$
9.	$\{\neg\neg p\} \vdash_{jas} \beta$	OA: 7,8
10.	$\{\neg\neg p\} \vdash_{jas} \alpha \vee \beta$	DA: 9
11.	$\vdash_{jas} \neg p \rightarrow (\alpha \vee \beta)$	6, twierdzenie o dedukcji
12.	$\vdash_{jas} \neg\neg p \rightarrow (\alpha \vee \beta)$	10, twierdzenie o dedukcji
13.	$\vdash_{jas} \alpha \vee \beta$	11,12, reguła wtórna: $\frac{\gamma \rightarrow \delta, \neg\gamma \rightarrow \delta}{\delta}$ Q.E.D.

TWIERDZENIE 10.3.2. (***Trafność rezolucji w KRZ***)

Niech S będzie zbiorem klauzul (reprezentującym pewną formułę w kpn). Jeśli $\Box \in \mathcal{R}(S)$, to formuła reprezentowana przez S nie jest spełnialna.

DOWÓD.

Niech $\Box \in \mathcal{R}(S)$. Wtedy $\Box \in res_n(S)$ dla pewnego n (zob. uwagę po definicji 10.2.2.). Ponieważ $\Box \notin res_0(S)$ (bo $\Box$ nie jest klauzulą występującą w zbiorze S), więc istnieje liczba $m > 0$ taka, że:

- (a) $\Box \notin res_m(S)$
- (b) $\Box \in res_{m+1}(S)$.

Na mocy (b), $\Box$ jest rezolwentą dwóch klauzul z $res_m(S)$. Jednak $\Box$ może być rezolwentą jedynie pary literałów komplementarnych, czyli literałów postaci p i $\neg p$, gdzie p jest zmienną zdaniową. Nadto, zarówno p, jak i $\neg p$ są elementami $res_m(S)$. Na mocy:

- tego, że $m > 0$,
- definicji zbiorów $res_i(S)$,
- przechodniości relacji $\vdash_{jas}$,
- twierdzenia 10.3.1.,

zarówno p, jak i $\neg p$ są konsekwencjami (w sensie relacji $\vdash_{jas}$) formuły w kpn języka KRZ, której reprezentacją jest S. Stąd $p \wedge \neg p$ jest konsekwencją tej formuły. Ponieważ $p \wedge \neg p$ nie jest spełnialna, więc (na mocy Twierdzenia o Pełności KRZ) i owa formuła, reprezentowana przez S, nie jest spełnialna.

Q.E.D.

Pokazaliśmy zatem (korzystając z własności syntaktycznej relacji $\vdash_{jas}$ oraz z Twierdzenia o Pełności KRZ), że rezolucyjna odrzucalność zbioru klauzul reprezentującego formułę języka KRZ w koniunkcyjnej postaci normalnej implikuje niespełnialność tej formuły.

Sposób (B)

Twierdzenia o trafności metody rezolucyjnej możemy też dowieść „na drodze semantycznej", odwołując się bezpośrednio do wartościowań.

Zauważmy najpierw, że (na mocy stosownych definicji) zachodzi następująca równoważność:

- Klauzula C jest rezolucyjnie wyprowadzalna ze zbioru klauzul S wtedy i tylko wtedy, gdy $C \in \mathcal{R}(S)$.
- W szczególności, istnieje rezolucyjna refutacja S wtedy i tylko wtedy, gdy $\Box \in \mathcal{R}(S)$.

Zamiast twierdzenia 10.3.1. posłużymy się teraz jego semantycznym odpowiednikiem:

TWIERDZENIE 10.3.3.

Jeśli $S = \{C_1, C_2\}$ jest spełnialny oraz C jest rezolwentą C_1 i C_2, to C jest spełnialna w KRZ. Co więcej, każde wartościowanie zmiennych zdaniowych, które spełnia S, spełnia też C.

DOWÓD.

Uwaga. Mówiąc o spełnialności zbioru klauzul S mamy na myśli, że jeśli $\alpha \rightleftharpoons S$, to formuła α jest spełnialna. Podobnie dla (spełnialności) pojedynczych klauzul.

Jeśli C jest rezolwentą C_1 i C_2, to istnieje literał ℓ oraz klauzule D_1, D_2, takie, że :

- $C_1 = D_1 \cup \{\ell\}$
- $C_2 = D_2 \cup \{\overline{\ell}\}$
- $C = D_1 \cup D_2$.

Załóżmy, że istnieje wartościowanie h takie, że $h(C_1) = 1$ oraz $h(C_2) = 1$. Z definicji wartościowania, wykluczone są przypadki:

- $h(\ell) = 1$ i $h(\overline{\ell}) = 1$
- $h(\ell) = 0$ i $h(\overline{\ell}) = 0$.

Pozostają zatem możliwości:

- (a) $h(\ell) = 1$ i $h(\overline{\ell}) = 0$
- (b) $h(\ell) = 0$ i $h(\overline{\ell}) = 1$.

Załóżmy, że zachodzi przypadek (a). Wtedy, ponieważ $h(C_2) = 1$ i $h(\overline{\ell}) = 0$, więc musi być $h(D_2) = 1$. Wtedy oczywiście także $h(C) = h(D_1 \cup D_2) = 1$.

Załóżmy, że zachodzi przypadek (b). Wtedy, ponieważ $h(C_1) = 1$ i $h(\ell) = 0$, więc musi być $h(D_1) = 1$. Wtedy oczywiście także $h(C) = h(D_1 \cup D_2) = 1$.

Q.E.D.

Uwaga. Zauważmy, że w istocie mamy silniejszą wersję twierdzenia 10.3.3.: ***każde*** wartościowanie, które spełnia (tj. przypisuje wartość 1) zbiór $S = \{C_1, C_2\}$, spełnia także ***każdą*** rezolwentę klauzul C_1 i C_2.

TWIERDZENIE 10.3.4. (***Trafność rezolucji w KRZ***).

Jeżeli istnieje rezolucyjna refutacja S, to S nie jest spełnialny w KRZ.

DOWÓD.

Niech $C_1, C_2, \ldots, C_n$ będzie rezolucyjną refutacją S. Wtedy oczywiście C_n jest identyczna z klauzulą $\Box$.

Z twierdzenia 10.3.3. wynika natychmiast, przez indukcję po długości rezolucyjnej refutacji zbioru S, że każde wartościowanie, które spełnia S, spełnia też każdą klauzulę C_i dla $1 \leqslant i \leqslant n$.

Ponieważ żadne wartościowanie nie spełnia klauzuli pustej $\Box$, więc nie istnieje wartościowanie spełniające S.

Q.E.D.

10.4. Pełność metody rezolucji

Trzeba pokazać, że metoda rezolucji jest ***pełna***, tj. udowodnić, że jeśli zbiór S jest niespełnialny, to można z niego rezolucyjnie wyprowadzić klauzulę pustą $\Box$.

Podobnie jak w przypadku Twierdzenia o Trafności Rezolucji, można tego dokonać na dwa sposoby.

Sposób (A)

TWIERDZENIE 10.4.1. (***Pełność rezolucji w KRZ***).

Jeżeli S jest niespełnialny, to $\Box \in \mathcal{R}(S)$.

DOWÓD.

Niech $S = \{C_1, \ldots, C_k\}$. Możemy oczywiście założyć, że żadna C_i nie jest tautologią. W przeciwnym przypadku możemy usunąć wszystkie tautologie z S i wyprowadzić $\Box$ z klauzul pozostałych w S.

Przeprowadzimy dowód przez indukcję po n: liczbie zmiennych zdaniowych występujących w S.

POCZĄTKOWY KROK INDUKCJI. Niech $n = 1$. Niech p będzie jedyną zmienną zdaniową występującą w S. Są wtedy trzy możliwości:

- każda C_i jest postaci $\{p\}$
- każda C_i jest postaci $\{\neg p\}$
- każda C_i jest postaci $\{p, \neg p\}$.

Trzecią z tych możliwości wykluczyliśmy. Tak więc, jedynymi klauzulami w S są $\{p\}$ oraz $\{\neg p\}$. Są trzy możliwości:

- $S = \{\{p\}\}$
- $S = \{\{\neg p\}\}$
- $S = \{\{p\}, \{\neg p\}\}$.

W pierwszych dwóch S byłby spełnialny (a nie jest, z założenia). Tak więc, $S = \{\{p\}, \{\neg p\}\}$. Oczywiście $\Box \in \mathcal{R}(S)$.

NASTĘPNIKOWY KROK INDUKCJI. Załóżmy teraz, że jedynymi zmiennymi zdaniowymi występującymi w S są $p_1, p_2, \ldots, p_n, p_{n+1}$. Załóżmy też, że $\Box \in \mathcal{R}(T)$ dla każdego niespełnialnego zbioru T, w którym występują jedynie zmienne zdaniowe $p_1, p_2, \ldots, p_n$.

Zdefiniujmy następujące formuły:

- S^0 jest koniunkcją tych wszystkich C_i z S, które nie zawierają literału $\neg p_{n+1}$
- S^1 jest koniunkcją tych wszystkich C_i z S, które nie zawierają literału p_{n+1}.

S^0 i S^1 są formułami w kpn. Zauważmy, że $S = S^0 \cup S^1$, gdy S^0 i S^1 traktujemy jako ***zbiory*** klauzul. Gdyby było inaczej, to istniałaby klauzula C_i z S, która nie byłaby elementem $S^0 \cup S^1$. Wtedy C_i zawierałaby zarówno p_{n+1}, jaki $\neg p_{n+1}$, a więc (jako alternatywa elementarna) byłaby tautologią, co już wcześniej wykluczyliśmy. Dowodzi to inkluzji $S^0 \cup S^1 \subseteq S$. Ponieważ oczywiście $S \subseteq S^0 \cup S^1$, więc zachodzi równość $S = S^0 \cup S^1$.

Niech teraz:

- $S_0 = \{C_i - \{p_{n+1}\} : C_i \in S^0\}$
- $S_1 = \{C_i - \{\neg p_{n+1}\} : C_i \in S^1\}$.

Zauważmy, że:

- jeśli zastąpimy p_{n+1} w S przez $\bot$, to otrzymana formuła jest (semantycznie) równoważna z S_0
- jeśli zastąpimy p_{n+1} w S przez $\top$, to otrzymana formuła jest (semantycznie) równoważna z S_1.

Wynika stąd, że S jest (semantycznie) równoważny z $S_0 \vee S_1$. Ponieważ S jest niespełnialny, więc S_0 i S_1 są niespełnialne. W klauzulach z S_0 i z S_1 występują jedynie zmienne zdaniowe $p_1, p_2, \ldots, p_n$. Z założenia indukcyjnego, zachodzi zarówno $\Box \in \mathcal{R}(S_0)$, jak i $\Box \in \mathcal{R}(S_1)$.

S_0 utworzono z S^0 poprzez wyrzucenie literału p_{n+1} z każdej klauzuli w S^0. Ponieważ możemy wyprowadzić $\Box$ z S_0, więc z S^0 możemy wyprowadzić $\Box$ lub $\{p_{n+1}\}$.

Podobnie, S_1 utworzono z S^0 poprzez wyrzucenie literału $\neg p_{n+1}$ z każdej klauzuli w S^1. Ponieważ możemy wyprowadzić $\Box$ z S_1, więc z S^1 możemy wyprowadzić $\Box$ lub $\{\neg p_{n+1}\}$.

Jeśli możemy wyprowadzić $\{p_{n+1}\}$ z S_0, a $\{\neg p_{n+1}\}$ z S_1, to możemy wyprowadzić $\Box$ z $S^0 \cup S^1$. Ponieważ $S = S^0 \cup S^1$, więc ostatecznie $\Box \in \mathcal{R}(S)$.

Q.E.D.

TWIERDZENIE 10.4.2.

Niech α i β będą formułami języka KRZ i niech γ będzie koniunkcyjną postacią normalną formuły $\alpha \wedge \neg\beta$. Wtedy następujące warunki są równoważne:

- (1) $\alpha \models_{KRZ} \beta$
- (2) $\{\alpha\} \vdash_{krz} \beta$
- (3) $\{\alpha\} \vdash_{jas} \beta$
- (4) $\Box \in \mathcal{R}(\{\gamma\})$.

DOWÓD.

Równoważność (2) i (3) pokazano w Dodatku 4.

Implikacja (2)⇒(1) to Twierdzenie o Trafności w KRZ (udowodnione w Dodatku 3).

Implikacja (1)⇒(4) jest konsekwencją udowodnionego przed chwilą twierdzenia o pełności metody rezolucyjnej.

Pokażemy, że (4) implikuje (3).

Na mocy równości $\vdash_{krz} = \vdash_{jas}$ oraz twierdzenia 5.5. (zob. Dodatek 2), mamy: $\{\alpha \wedge \neg\beta\} \vdash_{jas} \gamma$. Reguła DK dołączania koniunkcji daje: $\{\alpha, \neg\beta\} \vdash_{jas} \alpha \wedge \neg\beta$. Z przechodniości $\vdash_{jas}$ (wniosek 8.1.(2)) mamy: $\{\alpha, \neg\beta\} \vdash_{jas} \gamma$.

Ponieważ $\Box \in \mathcal{R}(\{\gamma\})$, więc dla pewnej zmiennej zdaniowej p mamy: $p, \neg p \in \mathcal{R}(\{\gamma\})$. Stąd, na mocy twierdzenia 10.3.1.:

- $\{\gamma\} \vdash_{jas} p$ ***oraz***
- $\{\gamma\} \vdash_{jas} \neg p$.

Z przechodniości relacji $\vdash_{jas}$ otrzymujemy zatem:

- $\{\alpha, \neg\beta\} \vdash_{jas} p$ ***oraz***
- $\{\alpha, \neg\beta\} \vdash_{jas} \neg p$.

To oznacza, że:

- $\{\alpha\} \vdash_{jas} \neg\beta \rightarrow p$ ***oraz***
- $\{\alpha\} \vdash_{jas} \neg\beta\neg p$.

Na mocy tezy $(\alpha \rightarrow \beta) \rightarrow ((\alpha \rightarrow \neg\beta) \rightarrow \neg\alpha)$ otrzymujemy stąd: $\{\alpha\} \vdash_{jas} \neg\neg\beta$. Na mocy prawa opuszczania negacji mamy ostatecznie $\{\alpha\} \vdash_{jas} \beta$.

Q.E.D.

Sposób (B)

TWIERDZENIE 10.4.3.

Dla dowolnego zbioru klauzul T oraz dowolnego literału ℓ: jeśli T jest niespełnialny, to niespełnialny jest także zbiór $T(\ell) = \{C \in \mathcal{R}(T) \ : \ \ell, \overline{\ell} \notin C\}$.

DOWÓD.

Uwaga. Zapis $\ell \notin C$ oznacza, że literał ℓ nie występuje w klauzuli C.

Niech T będzie niespełnialny. W terminologii używanej w wykładach 3–4 powiedzielibyśmy, że T jest semantycznie sprzeczny.

Przypuśćmy, dla dowodu nie wprost, że h jest wartościowaniem takim, że h spełnia (tj. przyjmuje wartość 1) wszystkie elementy zbioru $T(\ell)$. Określmy wartościowania h_1 oraz h_2 tak, aby:

- $h_1(\ell) = 1$ oraz $h_2(\overline{\ell}) = 1$
- h_1 i h_2 przyjmowały takie same wartości jak h dla wszystkich pozostałych (tj. różnych od ℓ i $\overline{\ell}$) literałach występujących w T.

Ponieważ T jest niespełnialny, więc istnieją klauzule C_1 oraz C_2 w T takie, że:

- $h_1(C_1) = 0$
- $h_2(C_2) = 0$.

Ponieważ $h_1(\ell) = 1$, a $h_1(C_1) = 0$, więc ℓ nie występuje w C_1. Gdyby także $\overline{\ell}$ nie występował w C_1, to, na mocy definicji, mielibyśmy $C_1 \in T(\ell)$. Ponieważ przeczyłoby to założeniu, że h spełnia wszystkie elementy zbioru $T(\ell)$, więc $\overline{\ell}$ występuje w C_1. Analogicznie rozumując, pokazujemy, że ℓ występuje w C_2.

Pokazaliśmy, że klauzule C_1 i C_2 zawierają parę literałów komplementarnych. Możemy zatem otrzymać ich rezolwentę $D \in \mathcal{R}(T)$, która nie zawiera literału ℓ (oraz, oczywiście, nie zawiera też literału $\overline{\ell}$).

Z definicji wartościowania h mamy $h(D) = 1$. Z kolei, z definicji wartościowań h_1 i h_2 oraz z faktu, że $h_1(\ell) = 1$ i $h_2(\overline{\ell}) = 1$ wynika, że niemożliwe jest, aby $h(C_1) = 0$ ***oraz*** $h(C_2) = 0$. Zachodzi zatem alternatywa:

- $h(C_1) = 1$ ***lub***
- $h(C_2) = 1$

Otrzymujemy sprzeczność z założeniem, że T jest niespełnialny.

Q.E.D.

TWIERDZENIE 10.4.4. (***Pełność rezolucji w KRZ***).

Jeżeli S jest niespełnialny w KRZ, to istnieje rezolucyjna refutacja S.

DOWÓD.

Na mocy Twierdzenia o Zwartości (zobacz Dodatek 4, twierdzenie 8.5.), jeśli S jest niespełnialny, to istnieje skończony zbiór $S' \subseteq S$, który jest niespełnialny. Ponieważ każda rezolucyjna refutacja z S' jest też refutacją z S, możemy założyć, że S jest skończony. Skoro istnieje tylko skończenie wiele klauzul w S i każda klauzula jest zbiorem skończonym, więc istnieje tylko skończona liczba literałów, występujących w klauzulach z S. Niech będą to literały $\ell_1, \ell_2, \ldots, \ell_n$.

Niech S będzie niespełnialny. Pokażemy, że istnieje rezolucyjna refutacja S.

Niech $S_n = S(\ell) = \{C \in \mathcal{R}(S) : \ell, \overline{\ell} \notin C\}$. Z definicji, S_n jest zbiorem tych wszystkich konsekwencji rezolucyjnych S, które nie zawierają literałów ℓ_n oraz $\overline{\ell_n}$. Ponieważ S jest niespełnialny, więc na mocy twierdzenia 10.4.3., S_n również jest niespełnialny.

Niech z kolei $S_{n-1} = S(\ell_{n-1})$. Wtedy S_{n-1} jest zbiorem tych wszystkich konsekwencji rezolucyjnych S_n (a więc także S), które nie zawierają literałów ℓ_n, ℓ_{n-1}, $\overline{\ell_n}$ i $\overline{\ell_{n-1}}$.

Powtarzamy tę procedurę aż do otrzymania zbioru S_0, który jest niespełnialny i nie zawiera żadnych literałów. Jedynym takim zbiorem jest $\{\Box\}$. Pokazaliśmy zatem, że $\Box$ jest rezolucyjną konsekwencją S.

Q.E.D.

10.5. Dalsze przykłady

Skoro metoda rezolucji jest trafna i pełna, to można jej używać np. dla ustalania, czy:

- formuła języka KRZ jest tautologią KRZ
- formuła języka KRZ jest spełnialna
- formuła języka KRZ nie jest spełnialna
- formuła α wynika logicznie ze zbioru formuł X
- zbiór formuł X jest spełnialny
- zbiór formuł X nie jest spełnialny, itd.

PRZYKŁAD 10.5.1.

Rozważmy zbiór klauzul:

$$S = \{\{p_1, p_2, \neg p_3\}, \{p_3\}, \{p_1, \neg p_2, p_3\}, \{\neg p_3\}\}.$$

Zauważmy, że w zależności od kolejności doboru klauzul, do których stosujemy regułę rezolucji, możemy otrzymać różne wyniki końcowe:

(a)

1.	$\{p_1, p_2, \neg p_3\}$	przesłanka
2.	$\{p_3\}$	przesłanka
3.	$\{p_1, \neg p_2, p_3\}$	przesłanka
4.	$\{\neg p_3\}$	przesłanka
5.	$\square$	rezolwenta 2 i 4.

(b)

1.	$\{p_1, p_2, \neg p_3\}$	przesłanka
2.	$\{p_3\}$	przesłanka
3.	$\{p_1, \neg p_2, p_3\}$	przesłanka
4.	$\{\neg p_3\}$	przesłanka
5.	$\{p_1, p_2\}$	rezolwenta 1 i 2.
6.	$\{p_1, \neg p_2\}$	rezolwenta 3 i 4
7.	$\{p_1\}$	rezolwenta 5 i 6.

Tak więc, zbiór S ***nie jest*** spełnialny, ponieważ istnieje ***co najmniej jedno*** wyprowadzenie $\square$ ze zbioru S.

PRZYKŁAD 10.5.2.

Pokażemy, że

$$((\alpha \to \beta) \wedge (\beta \to \gamma) \wedge (\gamma \to \alpha) \wedge (\alpha \vee \beta \vee \gamma)) \to (\alpha \wedge \beta \wedge \gamma)$$

jest tautologią KRZ.

Jest tak dokładnie wtedy, gdy zbiór

$$\{\alpha \to \beta, \beta \to \gamma, \gamma \to \alpha, \alpha \vee \beta \vee \gamma, \neg(\alpha \wedge \beta \wedge \gamma)\}$$

jest semantycznie sprzeczny (nie jest spełnialny). To z kolei jest równoważne temu, że zbiór

$$\{\neg\alpha \vee \beta, \neg\beta \vee \gamma, \neg\gamma \vee \alpha, \alpha \vee \beta \vee \gamma, \neg\alpha \vee \neg\beta \vee \neg\gamma\}$$

nie jest spełnialny. Każda z formuł tego zbioru jest podstawieniem jakiejś alternatywy elementarnej: otrzymujemy je, gdy dokonamy np. podstawień $p_1/\alpha,\ p_2/\beta,\ p_3/\gamma$. W takich przypadkach usprawiedliwione jest pisanie dowodów rezolucyjnych z użyciem *metazmiennych* reprezentujących dowolne formuły języka KRZ i traktowanie pojedynczych metazmiennych jak literałów.

Na mocy pełności metody rezolucji wystarczy pokazać, że ze zbioru

$$\{\neg\alpha \vee \beta, \neg\beta \vee \gamma, \neg\gamma \vee \alpha, \alpha \vee \beta \vee \gamma, \neg\alpha \vee \neg\beta \vee \neg\gamma\}$$

można wyprowadzić klauzulę $\square$:

1.	$\neg\alpha \vee \beta$	przesłanka
2.	$\neg\beta \vee \gamma$	przesłanka
3.	$\neg\gamma \vee \alpha$	przesłanka
4.	$\alpha \vee \beta \vee \gamma$	przesłanka
5.	$\neg\alpha \vee \neg\beta \vee \neg\gamma$	przesłanka
6.	$\alpha \vee \beta$	rezolwenta 4 i 3
7.	β	rezolwenta 6 i 1
8.	γ	rezolwenta 7 i 2
9.	α	rezolwenta 8 i 3
10.	$\neg\beta \vee \neg\gamma$	rezolwenta 9 i 5
11.	$\neg\gamma$	rezolwenta 7 i 10
12.	$\square$	rezolwenta 8 i 11.

Stosujemy tu jeszcze jedno świństewko notacyjne, pisząc poszczególne alternatywy elementarne, a nie odpowiadające im klauzule. Inne z tego typu świństewek to milczące korzystanie z praw pochłaniania dla alternatywy: w wierszu 6 piszemy $\alpha \vee \beta$ zamiast $\alpha \vee \beta\alpha \vee \alpha$, a w wierszu 7 piszemy β zamiast $\beta \vee \beta$.

Takie (i dalsze jeszcze świństewka notacyjne) często spotykamy w niektórych podręcznikach. Czujemy się więc trochę usprawiedliwieni, także z nich korzystając. Jak pisał St.I. Witkiewicz:

Ciężko jest żyć w plugawej naszej atmosferze,
Czasami, ach, wprost nawet kogoś z boku litość bierze —
Pociecha w tym, że gorzej być plugawcem, ach, samemu,
Bo nic już nie pomoże, ach, takiemu.

PRZYKŁAD 10.5.3.

Pokażemy, że formuła:

$$(\bigstar) \quad \neg((\alpha \to \beta) \to ((\alpha \vee \gamma) \to (\beta \vee \gamma)))$$

nie jest spełnialna. Oznacza to, że formuła:

$$(\bigstar\bigstar) \quad (\alpha \to \beta) \to ((\alpha \vee \gamma) \to (\beta \vee \gamma))$$

jest tautologią KRZ.

W tym celu wystarczy pokazać, że ze zbioru klauzul otrzymanego z kpn formuły $(\bigstar)$ można wyprowadzić $\square$. Koniunkcyjną postacią normalną formuły $(\bigstar)$ jest:

$$(\neg\alpha \vee \beta) \wedge (\alpha \vee \gamma) \wedge (\neg\beta) \wedge (\neg\gamma).$$

Przeprowadzamy dowód rezolucyjny:

1.	$\neg\alpha \vee \beta$	przesłanka
2.	$\alpha \vee \gamma$	przesłanka
3.	$\neg\beta$	przesłanka
4.	$\neg\gamma$	przesłanka
5.	α	rezolwenta 2 i 4
6.	β	rezolwenta 1 i 5
9.	$\square$	rezolwenta 3 i 6.

PRZYKŁAD 10.5.4.

Pokażemy, że formuła γ wynika logicznie ze zbioru formuł:

$$S = \{\alpha, (\alpha \wedge \beta) \to \gamma, \tau \to \beta, \tau\}.$$

W tym celu wystarczy pokazać, że zbiór

$$\{\alpha, (\alpha \wedge \beta) \to \gamma, \tau \to \beta, \tau, \neg\gamma\}$$

nie jest spełnialny.

Każda formuła ze zbioru S jest równoważna alternatywie elementarnej:

1. α
2. $\neg\alpha \vee \neg\beta \vee \gamma$
3. $\neg\tau \vee \beta$
4. τ.

Pokazujemy, że z powyższych klauzul można wyprowadzić $\square$:

1.	α	przesłanka
2.	$\neg\alpha \vee \neg\beta \vee \gamma$	przesłanka
3.	$\neg\tau \vee \beta$	przesłanka
4.	τ	przesłanka
5.	$\neg\gamma$	przesłanka
6.	$\neg\alpha \vee \neg\beta$	rezolwenta 2 i 5
7.	$\neg\beta$	rezolwenta 6 i 1
8.	$\neg\tau$	rezolwenta 3 i 7
9.	$\square$	rezolwenta 4 i 8.

PRZYKŁAD 10.5.5.

Pokażemy, że formuła β wynika logicznie z następującego zbioru formuł:

$$S = \{\alpha \to \beta, (\gamma \wedge \delta) \to \alpha, (\tau \wedge \gamma) \to \delta, (\theta \wedge \alpha) \to \gamma, (\theta \wedge \tau) \to \gamma, \theta, \tau\}.$$

Każda formuła ze zbioru S jest równoważna alternatywie elementarnej:

1. $\neg\alpha \vee \beta$
2. $\neg\gamma \vee \neg\delta \vee \alpha$
3. $\neg\tau \vee \neg\gamma \vee \delta$
4. $\neg\theta \vee \neg\alpha \vee \gamma$
5. $\neg\theta \vee \neg\tau \vee \gamma$
6. θ
7. τ.

Budujemy rezolucyjne wyprowadzenie β z powyższych klauzul:

1.	$\neg\alpha \vee \beta$	przesłanka
2.	$\neg\gamma \vee \neg\delta \vee \alpha$	przesłanka
3.	$\neg\tau \vee \neg\gamma \vee \delta$	przesłanka
4.	$\neg\theta \vee \neg\alpha \vee \gamma$	przesłanka
5.	$\neg\theta \vee \neg\tau \vee \gamma$	przesłanka
6.	θ	przesłanka
7.	τ	przesłanka
8.	$\neg\tau \vee \gamma$	rezolwenta 5 i 6
9.	γ	rezolwenta 7 i 8
10.	$\neg\delta \vee \alpha$	rezolwenta 2 i 9
11.	$\neg\gamma \vee \delta$	rezolwenta 3 i 7
12.	$\neg\gamma \vee \alpha$	rezolwenta 2 i 11
13.	α	rezolwenta 9 i 12
14.	β	rezolwenta 1 i 13.

Ponieważ uzyskaliśmy rezolucyjne wyprowadzenie β z S, więc na mocy twierdzenia o pełności metody rezolucyjnej otrzymujemy, że $S \models_{KRZ} \beta$.

Dla porównania, przytoczmy jeszcze dowód założeniowy, że $S \vdash_{jas} \beta$:

1.	$\alpha \to \beta$	założenie
2.	$(\gamma \wedge \delta) \to \alpha$	założenie
3.	$(\tau \wedge \gamma) \to \delta$	założenie
4.	$(\theta \wedge \alpha) \to \gamma$	założenie
5.	$(\theta \wedge \tau) \to \gamma$	założenie
6.	θ	założenie
7.	τ	założenie
8.	$\theta \wedge \tau$	DK: 6,7
9.	γ	RO: 5,8
10.	$\tau \wedge \gamma$	DK: 7,9
11.	δ	RO: 3,10
12.	$\gamma \wedge \delta$	DK: 9,11
13.	α	RO: 2,12
14.	β	RO: 1,13.

Zauważmy, że dowód ten jest porównywalny — pod względem skomplikowania — z podanym wyżej dowodem rezolucyjnym.

Powyższe przykłady mogą osobie nieufnej nasunąć pytanie, po co właściwie zajmować się metodą rezolucji, skoro mamy inne, dobre metody dowodzenia tez. Podkreślamy, że metoda rezolucji znajduje zastosowanie przede wszystkim w automatycznym dowodzeniu twierdzeń. Przekształcenie nawet bardzo skomplikowanych formuł na równoważne im inferencyjnie kpn nie jest problemem dla szybkich maszyn liczących. Drugi krok w metodzie rezolucyjnej dowodzenia twierdzeń, czyli stosowanie samej reguły rezolucji, jest oczywiście także bardzo prostym zadaniem dla maszyn

liczących. Warto zatem wyobrazić sobie np. zbiór liczący tysiące skomplikowanych przesłanek i odetchnąć z ulgą, że możemy w takiej sytuacji powierzyć robotę dedukcyjną Maszynom.

10.6. Konsekwencja rezolucyjna

Jest jasne, jak zdefiniować operację C_{res} konsekwencji wyznaczoną przez metodę rezolucji:

$$C_{res}(X) = \{\alpha \in F_{KRZ} X \vdash_{res} \alpha\}.$$

Tak zdefiniowana operacja konsekwencji ma własności (C1)–(C4) podane na wykładach 5–7.

* * *

10.7. Uwagi końcowe

Uwaga. Jest wiele różnych, bardziej subtelnych od powyższego — całkowicie ogólnego — rodzajów rezolucji. Problematyka ta jest intensywnie badana, przede wszystkim w związku z zastosowaniami metody rezolucji w automatycznym dowodzeniu twierdzeń.

Dlaczego ważne są klauzule Hornowskie? Klauzulę Hornowską, w której występuje dokładnie jeden literał pozytywny i dowolna (skończona) liczba literałów negatywnych nazywamy *klauzulą programową*. Ów literał pozytywny nazywamy *nagłówkiem* klauzuli, a pozostałe literały negatywne *treścią klauzuli*. Zbiór klauzul o tym samym nagłowku jest *procedurą*, a zbiór procedur jest *programem*. Klauzulę Hornowską o jednym literale pozytywnym i bez literałów negatywnych nazywamy *faktem*, a klauzulę bez literału pozytywnego nazywamy *klauzulą negatywną*. Przez *regułę obliczeniową* rozumiemy regułę wybierania literału z klauzuli.

Język programowania PROLOG wykorzystuje właśnie takie konstrukcje syntaktyczne. Przypomnijmy, że jest to język *deklaratywny*. Wykorzystujemy w nim tzw. SLD-rezolucję (*Selection-rule driven Linear resolution for Definite clauses*), określoną dla programów w wyżej rozumianym sensie, a więc rodzin zbiorów klauzul Hornowskich.

Uwaga. Przypomnijmy następujące porównanie reguły rezolucji z regułą modus ponens w KRZ:

- REGUŁA REZOLUCJI: z formuł $\alpha \vee \gamma$ oraz $\neg\alpha \vee \beta$ wywnioskuj $\gamma \vee \beta$
- REGUŁA MODUS PONENS: z formuł $\alpha \to \beta$ oraz α wywnioskuj β (lub, w postaci równoważnej: z formuł $\neg\alpha \vee \beta$ oraz α wywnioskuj β).

Reguła *rezolucji* jest zatem szczególnym przypadkiem ogólniejszej reguły, tzw. *reguły cięcia*. W ramach niniejszego kursu nie przewiduje się omówienia tej problematyki.

Uwaga. Zainteresowanych zachęcamy do zajrzenia do rozdziału 3 książki Fitting 1990 (zwłaszcza do podrozdziału 3.3.), gdzie znajdujemy opis metody rezolucji w połączeniu z pewną (prostą i wielce naturalną!) techniką przekształcania formuł do równoważnych im inferencyjnie kpn oraz apn.

Uwaga historyczna. Church i Turing udowodnili podstawowe twierdzenia implikujące nierozstrzygalność klasycznego rachunku logicznego (logiki pierwszego rzędu). Z kolei, z wyników uzyskanych przez Herbranda i Skolema wynika, że rachunek ten jest półrozstrzygalny: jeśli jakaś formuła jest tautologią logiki pierwszego rzędu, to można tego dowieść (w skończonej liczbie kroków). Fakt ten, łącznie z powstaniem i rozwojem elektronicznych maszyn liczących inspirował do poszukiwania systemów automatycznego dowodzenia twierdzeń.

Algorytm zaproponowany przez Davisa i Putnama nawiązywał do twierdzenia Herbranda (redukującego, w ściśle określonym sensie, problem ustalania czy dana formuła jest tautologią logiki pierwszego rzędu do problemu tautologiczności pewnych formuł KRZ). Mówi się, że reguła rezolucji została po raz pierwszy wykorzystana przez J.A. Robinsona (1963–1964). Wiadomo też jednak, że już w XIX wieku używał jej konsekwentnie Lewis Carroll, w swoim algebraicznym ujęciu sylogistyki.

Więcej na ten temat np. w rozdziale siódmym monografii Marciszewski, Murawski 1995.

Uwaga. Należy być świadomym różnic pomiędzy:

- ***Wynikaniem logicznym*** a ***uzasadnianiem*** oraz ***uznawaniem*** zdań. Pierwsze z tych pojęć ma, w dzisiejszym rozumieniu, charakter obiektywny; drugie i trzecie mogą odwoływać się do różnych czynników, także natury pragmatycznej. Uzasadnianie może mieć postać precyzyjnego dowodu, ale może też odwoływać się do zabawnych reguł LOGIKI UZNANIOWEJ.

- ***Dowodzeniem*** a procedurami czysto ***algorytmicznymi***. Pierwsza z tych aktywności ma charakter ***twórczy***, drugie są działaniami wedle określonego przepisu.

Student, który zna jedynie KRZ i ma dopiero przed sobą wykład na temat KRP (Klasycznego Rachunku Predykatów) może odnieść (złudne i pochopne!) wrażenie, że dowodzenie ma w dowolnym systemie logicznym charakter czysto algorytmiczny i że w takim systemie zawsze istnieje efektywna metoda rozstrzygania, czy dana formuła jest tautologia tego systemu.

Zwróćmy uwagę, że jest zasadnicza różnica pomiędzy:

- ***sprawdzeniem***, że dany ciąg formuł jest dowodem jakiejś formuły α, ***a***

- ***znalezieniem*** dowodu formuły α.

LOGIKA MATEMATYCZNA (24–25)

KLASYCZNY RACHUNEK PREDYKATÓW: UNIFIKACJA I REZOLUCJA

Omówimy teraz konsekwencję ***rezolucyjną*** w KRP.

Wykład 24: Unifikacja

Pojęcia i metody wprowadzone w tym podrozdziale będą wykorzystane w podrozdziale następnym, w którym pokażemy działanie pewnej ważnej metody dowodowej (*metody rezolucji*), w pewnym sensie związanej z metodą tablic analitycznych. Nadto, pojęcie *unifikacji* jest samo w sobie interesujące. W ostatnich kilkudziesięciu latach nabrało istotnego znaczenia, np. w automatycznym dowodzeniu twierdzeń.

24.1. Definicje

Pracujemy teraz w KRP z identycznością oraz symbolami funkcyjnymi.

Literałami nazwiemy formuły atomowe oraz ich negacje. Formuły atomowe to ***literały pozytywne***, negacje formuł atomowych to ***literały negatywne***.

Terminu ***wyrażenie*** będziemy tu używać dla dowolnego termu lub literału.

Podstawieniem nazywamy każdą funkcję σ ze zbioru wszystkich zmiennych w zbiór wszystkich termów, która jest funkcją identycznościową prawie wszędzie, tj. dla wszystkich, oprócz skończonej liczby, zmiennych.

Ponieważ w zastosowaniach istotna będzie tylko skończona liczba wartości każdego podstawienia, więc czasem wygodnie będzie uważać za podstawienie dowolny skończony zbiór par uporządkowanych, których jednym elementem jest zmienna, a drugim term.

W takim przypadku podstawienia zapisywać możemy jako zbiory postaci $\{t_1/x_1, \ldots, t_n/x_n\}$, gdzie x_i są zmiennymi, a t_i termami. Inną często używaną notacją jest zapis: $\{x_1 \mapsto t_1, \ldots, x_n \mapsto t_n\}$. Ten zapis stosujemy poniżej.

Jeśli σ jest podstawieniem, a E wyrażeniem, to przez $E\sigma$ oznaczać będziemy wyrażenie powstające z E poprzez zastąpienie zmiennych występujących w E termami przyporządkowanymi im przez podstawienie σ. Indukcyjna definicja wyrażenia $E\sigma$ jest następująca:

- $E\sigma = x\sigma$, gdy E jest zmienną x,
- $E\sigma = f(t_1\sigma, \ldots, t_n\sigma)$, gdy E jest termem złożonym $f(t_1, \ldots, t_n)$,
- $E\sigma = R(t_1\sigma, \ldots, t_m\sigma)$, gdy E jest literałem pozytywnym $R(t_1, \ldots, t_m)$,
- $E\sigma = \neg R(t_1\sigma, \ldots, t_m\sigma)$, gdy E jest literałem negatywnym $\neg R(t_1, \ldots, t_m)$.

Dopuszczamy przy tym przypadek, gdy $n = 0$; wtedy E jest stałą indywiduową i przyjmujemy $E\sigma = E$.

Dziedziną podstawienia σ jest zbiór:

$$dm(\sigma) = \{x \ : \ x\sigma \neq x\},$$

a ***przeciwdziedziną*** (***zbiorem wartości***) podstawienia σ jest zbiór:

$$rg(\sigma) = \bigcup_{x \in dm(\sigma)} \{x\sigma\}.$$

Wreszcie, niech $var(\sigma)$ będzie zbiorem wszystkich zmiennych występujących w $rg(\sigma)$.

Ograniczeniem podstawienia σ do zbioru zmiennych X nazywamy podstawienie, które jest równe funkcji identycznościowej wszędzie poza zbiorem $X \cap dm(\sigma)$.

Jeśli S jest zbiorem wyrażeń, a σ podstawieniem, to przez $S\sigma$ oznaczać będziemy zbiór $\{E\sigma : \ E \in S\}$.

Ponieważ podstawienia są funkcjami, więc można na nich wykonywać operację złożenia. Zapis $E\sigma\theta$, gdzie E jest dowolnym wyrażeniem, należy odczytywać: wynik podstawienia złożonego $\sigma\theta$ na wyrażeniu E. Przy tym, wartość tę należy rozumieć jako wynik operacji $(E\sigma)\theta$.

Algorytm obliczania ***złożenia*** podstawień podamy dla przypadku, gdy rozważamy je jako skończone zbiory par (zmienna, term).

Niech $\sigma = \{x_1 \mapsto t_1, \ldots, x_n \mapsto t_n\}$ oraz $\theta = \{y_1 \mapsto s_1, \ldots, y_m \mapsto s_m\}$. Wtedy $\sigma\theta$ jest podstawieniem:

$$\{x_1 \mapsto t_1\theta, \ldots, x_n \mapsto t_n\theta, y_1 \mapsto s_1, \ldots, y_m \mapsto s_m\}$$

przy czym usuwamy te elementy $x_i \mapsto t_i\theta$ dla których $x_i = t_i\theta$ oraz te elementy $y_j \mapsto s_j$ dla których $y_j \in \{x_1, \ldots x_n\}$.

Podstawienie ***puste*** ϵ jest elementem neutralnym tej operacji, tj. $\theta\epsilon = \epsilon\theta = \theta$.

Przy tej definicji operacji złożenia można udowodnić, że operacja ta jest łączna, tj. że dla dowolnych podstawień θ, ψ oraz σ i dowolnego wyrażenia E:

- $(\psi\theta)\sigma = \psi(\theta\sigma)$.

Uwaga. Operacja złożenia nie jest przemienna, tj. nie zachodzi $\sigma\theta = \theta\sigma$ dla dowolnych θ oraz σ.

Powiemy, że podstawienie σ jest ***idempotentne***, gdy $\sigma\sigma = \sigma$. Można dowieść, że σ jest idempotentne wtedy i tylko wtedy, gdy $dm(\sigma) \cap rg(\sigma) = \emptyset$.

Niech $S = \{E_1, \ldots, E_n\}$ będzie zbiorem wyrażeń. Powiemy, że podstawienie σ jest ***unifikatorem*** dla S, gdy:

$$E_1\sigma = E_2\sigma = \ldots = E_n\sigma.$$

Zbiór S jest ***uzgadnialny***, jeśli istnieje unifikator dla S.

Unifikator θ dla S jest ***najbardziej ogólnym unifikatorem*** (***most general unifier***, w skrócie: ***mgu***), gdy dla każdego unifikatora σ dla S istnieje podstawienie λ takie, że $\theta\lambda = \sigma$.

Powiemy, że podstawienie λ ***przemianowuje zmienne***, jeśli $dm(\lambda) = rg(\lambda)$. Dla przykładu:

- podstawienie $\{x \mapsto y, y \mapsto z, z \mapsto x\}$ przemianowuje zmienne,
- podstawienia: $\{x \mapsto y\}$ oraz $\{x \mapsto z, y \mapsto z\}$ nie przemianowują zmiennych.

Jeśli $\lambda = \{x_1 \mapsto y_1, \ldots x_n \mapsto y_n\}$ jest podstawieniem przemianowującym zmienne, to podstawieniem do niego ***odwrotnym*** jest podstawienie $\lambda^{-1} = \{y_1 \mapsto x_1, \ldots y_n \mapsto x_n\}$. Oczywiście, jeśli λ przemianowuje zmienne, to λ^{-1} też.

Można dowieść, że jeśli θ oraz ψ są najbardziej ogólnymi unifikatorami dla S, to istnieją podstawienia przemianowujące zmienne σ oraz λ takie, że: $S\theta\sigma = S\psi$ oraz $S\theta = S\psi\lambda$.

Dwa podstawienia są równe, symbolicznie $\sigma = \theta$, jeśli $x\sigma = x\theta$ dla każdej zmiennej x. Mówimy, że σ jest ***bardziej ogólne*** niż θ, symbolicznie $\sigma \preceq \theta$, jeżeli istnieje λ takie, że $\theta = \sigma\lambda$. Relacja $\preceq$ jest częściowym porządkiem. Relacja $\doteq = \preceq \cap \preceq^{-1}$ jest oczywiście równoważnością. Można udowodnić, że $\sigma \doteq \theta$ wtedy i tylko wtedy, gdy istnieje podstawienie λ przemianowujące zmienne takie, że $\sigma = \theta\lambda$.

Dla przykładu, niech $\sigma_1 = \{x \mapsto f(g(a, h(z))), y \mapsto g(h(x), b), z \mapsto h(x)\}$ oraz $\sigma_2 = \{x \mapsto f(g(x, y)), y \mapsto g(z, b)\}$. Wtedy σ_2 jest bardziej ogólne niż σ_1, ponieważ $\sigma_1 = \sigma_2\tau$, gdzie $\tau = \{x \mapsto a, y \mapsto h(z), z \mapsto h(x)\}$.

Jeśli unifikator σ dla zbioru S ma tę własność, że dla dowolnego unifikatora τ dla S dziedzina $dm(\sigma)$ nie ma więcej elementów niż dziedzina τ, to σ nazywamy unifikatorem ***minimalnym*** dla S. Dla przykładu, jeżeli $S = \{x, f(y)\}$, to podstawienia $\sigma = \{y \mapsto x, x \mapsto f(x)\}$ oraz $\tau = \{x \mapsto f(y)\}$ są oba najbardziej ogólnymi unifikatorami dla S, ale tylko τ jest unifikatorem minimalnym dla S.

Można podać definicję mgu także w terminach porządku $\preceq$. Mianowicie σ jest najbardziej ogólnym unifikatorem (mgu) dla zbioru wyrażeń S, gdy $\sigma \preceq \theta$ dla każdego unifikatora θ dla S. Obie podane definicje są równoważne: $\sigma \preceq \theta$ dla każdego unifikatora θ dla S wtedy i tylko wtedy, gdy dla każdego unifikatora θ dla S istnieje podstawienie λ takie, że $\theta = \sigma\lambda$.

24.2. Przykłady

24.2.1. Rozważmy termy $f(a, x)$ oraz $f(y, b)$, gdzie a i b są stałymi indywiduowymi. Czy zbiór złożony z tych dwóch termów jest uzgadnialny? Inaczej mówiąc, czy można dokonać takiego podstawienia zmiennych, aby otrzymać z tych obu termów jeden i ten sam term? Odpowiedź jest prosta i twierdząca. Wystarczy dokonać podstawienia σ:

- $x \mapsto b$
- $y \mapsto a$.

Wtedy $f(a, x)\sigma = f(a, b)$ oraz $f(y, b)\sigma = f(a, b)$. Podstawienie $\{x \mapsto b, y \mapsto a\}$ nie jest w tym przypadku mgu dla rozważanego zbioru termów. Najbardziej ogólnym unifikatorem dla tego zbioru jest podstawienie $\theta = \{x \mapsto y\}$, jak łatwo widzieć, ponieważ dla dowolnego podstawienia $\sigma = \{x \mapsto t, y \mapsto t\}$ mamy: $\sigma = \theta\{y \mapsto t\}$.

Natomiast w przypadku termów $f(a, x)$ oraz $f(x, b)$ odpowiedź jest przecząca: nie istnieje podstawienie zmiennych, po dokonaniu którego otrzymalibyśmy jeden i ten sam term.

24.2.2. Ani zbiór $\{P(x, a), P(b, c)\}$ ani zbiór $\{P(f(x), z), P(a, w)\}$ nie jest uzgadnialny.

Niech $S_1 = \{P(x, c), P(b, c)\}$ oraz $S_2 = \{P(f(x), y), P(f(a), w)\}$. Wtedy zarówno S_1 jak i S_2 są uzgadnialne. Unifikatorem dla S_1 jest podstawienie $x \mapsto b$. Nadto, jest to jedyny unifikator dla S_1. Zbiór S_2 ma natomiast wiele różnych unifikatorów, np.:

- $\theta = \{x \mapsto a, y \mapsto w\}$,
- $\sigma = \{x \mapsto a, y \mapsto a, w \mapsto ba\}$,
- $\psi = \{x \mapsto a, y \mapsto b, w \mapsto b\}$.

W tym przypadku θ jest mgu dla S_2.

24.2.3. Niech $S_1 = \{f(x, g(x)), f(h(y), g(h(z)))\}$ oraz $S_2 = \{f(h(x), g(x)), f(g(x), h(x))\}$. Pokażemy, że S_1 jest uzgadnialny, natomiast S_2 nie jest.

W każdym z obu powyższych przypadków wszystkie literały rozpoczynają się od symbolu funkcyjnego f. Gdyby poszczególne literały (w S_1 lub w S_2) rozpoczynały się od różnych symboli funkcyjnych, to stosowne zbiory nie byłyby uzgadnialne, ponieważ podstawienia dotyczą jedynie zmiennych.

W każdym z rozważanych przypadków f jest dwuargumentowym symbolem funkcyjnym. Trzeba zatem przyjrzeć się pierwszemu i drugiemu argumentowi f. Jeśli uda się znaleźć podstawienie, które będzie uzgadniać oba te argumenty, to tym samym znajdziemy unifikator dla rozważanego zbioru literałów.

W przypadku S_1:

- pierwszymi argumentami f są: x i $h(y)$,
- drugimi argumentami f są: $g(x)$ i $g(h(z))$.

Aby uzgodnić pierwsze argumenty (w najbardziej ogólny sposób) powinniśmy dokonać podstawienia $x \mapsto h(y)$. Wtedy drugie argumenty literałów w S_1 przybiorą postać: $g(h(y))$ oraz $g(h(z))$, odpowiednio. Pierwszym miejscem, w którym różnią się te drugie argumenty jest wystąpienie y. Możemy uzgodnić drugie argumenty poprzez podstawienie $y \mapsto z$. Otrzymujemy wtedy jeden i ten sam literał dla drugich argumentów: $g(h(z))$. To podstawienie zastosować trzeba także do pierwszych argumentów, dla których otrzymujemy wtedy: $h(z)$. Ostatecznie mamy następujące wyrażenie, takie samo dla pierwszego i drugiego literału występującego w S_1: $f(h(z), g(h(z)))$. Unifikatorem poszukiwanym dla uzgodnienia zbioru S_1 jest więc złożenie podstawień: $\{x \mapsto h(y)\}$ oraz $\{y \mapsto z\}$.

W przypadku zbioru S_2 mamy następujące wartości dla pierwszych oraz drugich argumentów f:

- pierwszymi argumentami f są: $h(x)$ i $g(x)$,
- drugimi argumentami f są: $g(x)$ i $h(x)$.

Dla pierwszych argumentów f pierwszym symbolem, którym się one różnią, jest symbol funkcyjny, a nie zmienna. Tak więc, próba ich uzgodnienia kończy się niepowodzeniem. (Podobnie dla drugich argumentów, ale to już bez znaczenia, ponieważ pierwsze argumenty nie mogą zostać uzgodnione.) Widzimy zatem, że S_2 nie jest uzgadnialny.

24.2.4. Niech $S = \{R(f(g(x)), a, x), R(f(g(a)), a, b), R(f(y), a, z)\}$. Pokażemy, że S nie jest uzgadnialny.

Każdy z literałów w S rozpoczyna się od ciągu symboli $R(f($. W drugim z literałów następuje potem $g(a)$, a w trzecim zmienna y. Term $g(a)$ nie zawiera zmiennej y. Dokonujemy podstawienia $y \mapsto g(a)$ i otrzymujemy:

$$\{R(f(g(x)), a, x), R(f(g(a)), a, b), R(f(a), a, z)\}.$$

Mamy więcej niż jeden literał. Teraz wszystkie literały rozpoczynają się od ciągu symboli $R(f(g($. W pierwszym z literałów jest dalej zmienna x, a w drugim term a, który nie zawiera tej zmiennej. Dokonujemy podstawienia $x \mapsto a$ i otrzymujemy:

$$\{R(f(g(a)), a, a), R(f(g(a)), a, b), R(f(a), a, z)\}.$$

W dalszym ciągu mamy więcej niż jeden literał. Każdy literał rozpoczyna się teraz od ciągu symboli $R(f(g(a), a,$. W pierwszym literale mamy dalej term a, natomiast w drugim term b. Żaden z tych termów nie jest zmienną, a więc nie ma podstawienia, które uzgadniałoby te termy. W konsekwencji, wyjściowy zbiór S nie jest uzgadnialny.

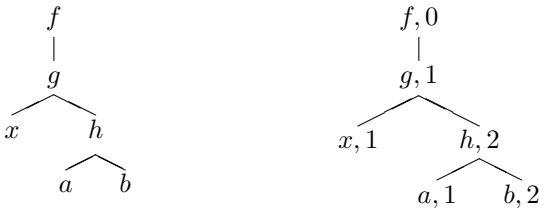
24.2.5. Zbiór $\{P(x, y), P(x, f(y))\}$ nie jest uzgadnialny. Niezależnie od tego, co podstawimy za zmienne x oraz y, w drugim literale jest jedno więcej wystąpienie symbolu f niż w pierwszym.

Powyższe przykłady (zaczerpnięte z: Baader, Snyder 2001, Hedman 2004 oraz Nerode, Shore 1997) dobrane są tak, aby zilustrować algorytmiczny sposób odnajdywania mgu dla zbioru literałów (lub pokazania, że zbiór literałów nie jest uzgadnialny). W tym celu potrzebne będą jeszcze następujące definicje.

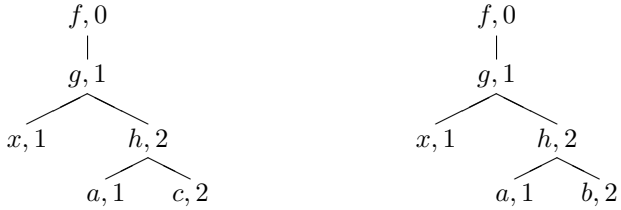
Niech S będzie skończonym niepustym zbiorem wyrażeń. ***Zbiorem niezgodności*** (niektórzy używają terminu: ***para niezgodności***) dla S nazywamy każdy dwuelementowy zbiór wyrażeń $\{E_1, E_2\}$ taki, że symbole (funktory) główne w E_1 i E_2 są różne oraz E_1 i E_2 występują na tych samych pozycjach jako podwyrażenia dwóch wyrażeń w S. Dla przykładu, gdy $S = \{x, g(a, y, u), g(z, b, v)\}$, to zbiorami niezgodności dla S są:

$$\{a, z\}, \{y, b\}, \{u, v\}, \{x, g(a, y, u)\}, \{x, g(z, b, v)\}.$$

Zbiory niezgodności łatwo sobie wyobrazić, gdy uwzględnimy syntaktyczną budowę termów. Termy możemy traktować jako ***drzewa znakowane***. Przy tym, znakowanie wierzchołków takiego drzewa może uwzględniać, oprócz poszczególnych symboli występujących w termie, także ***pozycje*** tych symboli w termie (np. numer argumentu funktora). Dla przykładu, term $f(g(x, h(a, b)))$ ma następujące reprezentacje:

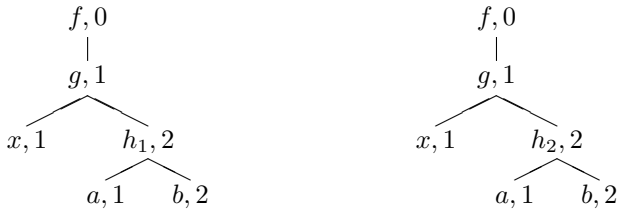


Lewe drzewo to po prostu drzewo syntaktyczne termu $f(g(x, h(a, b)))$. W drzewie prawym zaznaczono pozycje poszczególnych argumentów. Poniższe rysunki pokazują zbiory niezgodności dla termów reprezentowanych przez drzewa:



Zbiór niezgodności: $\{c, b\}$.

Zwróćmy uwagę, że do c prowadzi w powyższych drzewach taka sama droga, jak do b, a mianowicie droga: $(f, 0), (g, 1), (h, 2)$.



Zbiór niezgodności: $\{h_1(a, b), h_2(a, b)\}$.

Zwróćmy uwagę, że do $h_1(a, b)$ prowadzi w powyższych drzewach taka sama droga, jak do $h_2(a, b)$, a mianowicie droga: $(f, 0), (g, 1)$.

Jeśli S jest skończonym zbiorem wyrażeń takim, że jednym z jego zbiorów niezgodności jest $\{x, t\}$ (gdzie x jest zmienną, a t termem nie zawierającym zmiennej x), to mówimy, że $S\{x \mapsto t\}$ jest otrzymany z S przez ***eliminację zmiennej*** względem $\{x \mapsto t\}$.

Można udowodnić, że (Letz 1999, strony 160–161):

- Jeśli σ jest unifikatorem dla S, a D jest jednym ze zbiorów niezgodności dla S, to:
  - σ jest unifikatorem dla D,
  - każdy element D jest termem,
  - jednym z elementów D jest zmienna, która nie występuje w drugim elemencie D.
- Niech σ będzie unifikatorem dla zbioru S zawierającego co najmniej dwa elementy i niech $\{x, t\}$ będzie zbiorem niezgodności takim, że $x \neq x\sigma$. Jeśli $\tau = \sigma - \{x \mapsto x\sigma\}$, to $\sigma = \{x \mapsto t\}\tau$.
- Niech S będzie dowolnym zbiorem wyrażeń (termów lub formuł) bez kwantyfikatorów i niech V_S będzie zbiorem wszystkich zmiennych występujących w S. Wtedy:
  - Jeśli S jest uzgadnialny, to uzgadnialny jest też każdy zbiór otrzymany z S przez eliminację zmiennych.
  - Przez eliminację zmiennych można z S otrzymać jedynie skończenie wiele zbiorów.
  - Jeśli S' otrzymano z S przez eliminację zmiennych względem $\{x \mapsto t\}$, to moc zbioru S' jest mniejsza niż moc S oraz $V_{S'} = V_S - \{x\}$.
  - Przechodnie domknięcie relacji zachodzącej między zbiorami S' i S wtedy i tylko wtedy, gdy S' jest otrzymany (w jednym kroku) z S przez eliminację zmiennej, jest dobrze ufundowane, tj. nie istnieją nieskończone ciągi zbiorów otrzymywanych przez kolejne eliminacje zmiennych.

Definicja ***obliczonego unifikatora*** ma postać indukcyjną (względem mocy dziedziny unifikatora):

- $\emptyset$ jest (jedynym) obliczonym unifikatorem dla dowolnego jednoelementowego zbioru wyrażeń bez kwantyfikatorów.
- Jeśli podstawienie σ takie, że moc $dm(\sigma)$ równa jest n jest obliczonym unifikatorem dla skończonego zbioru S' oraz S' można otrzymać z S przez eliminacje zmiennej względem $\{x \mapsto t\}$, to podstawienie $\sigma \cup \{x \mapsto t\sigma\} = \{x \mapsto t\}\sigma$ o mocy dziedziny równej $n + 1$ jest obliczonym unifikatorem dla S.

Pojęcie obliczonego unifikatora zostanie użyte w omówieniu pewnego uogólnienia oryginalnego algorytmu unifikacji Robinsona.

Można udowodnić (zob. np. Letz 1999, strona 162), że:

- Jeśli zbiór S jest uzgadnialny, to σ jest minimalnym unifikatorem dla S wtedy i tylko wtedy, gdy σ jest obliczonym unifikatorem dla S.
- Jeśli zbiór S jest uzgadnialny, to obliczony unifikator dla S jest najbardziej ogólnym unifikatorem dla S.

Dla sformułowania jednego z algorytmów unifikacji użyteczne będzie następujące pojęcie. Niech S będzie skończonym niepustym zbiorem wyrażeń. Traktujemy S jako zbiór uporządkowany liniowo. Znajdujemy pierwszą (z lewej) pozycję, na której nie wszystkie elementy S mają ten sam symbol. Zbiór podwyrażeń każdego wyrażenia $E \in S$, które zaczynają się od tej pozycji jest oznaczamy przez $D(S)$.

Dla przykładu, w 24.2.3. powyżej rozważaliśmy zbiory:

$$S_1 = \{f(x, g(x)), f(h(y), g(h(z)))\} \text{ oraz } S_2 = \{f(h(x), g(x)), f(g(x), h(x))\}.$$

Mamy tutaj: $D(S_1) = \{x, h(y)\}$ oraz $D(S_2) = \{h(x), g(x)\}$.
Dla $S_1\{x \mapsto h(y)\}$ mamy: $D(S_1\{x \mapsto h(y)\}) = D(\{f(h(y), g(h(y))), f(h(y), g(h(z)))\}) = \{y, z\}$.
Zauważmy, że dowolny unifikator zbioru wyrażeń S musi uzgadniać zbiór $D(S)$.

24.3. Algorytmy unifikacji

Pojęcie unifikacji odnaleźć można już w pracach Herbranda. Podaje on również nieformalny opis algorytmu unifikacji, choć bez dowodu jego poprawności. Sam termin *unifikacja* po raz pierwszy został użyty przez J.A. Robinsona, który wykorzystywał pojęcie unifikacji w badaniach reguły rezolucji oraz pokazał, że uzgadnialny zbiór termów ma mgu i podał algorytm znajdowania tego mgu.

Poniżej omawiamy kilka algorytmów unifikacji.

24.3.1. Algorytm $\mathfrak{A}_1$: algorytm naiwny

W wielu podręcznikach przedstawiany jest następujący algorytm unifikacji $\mathfrak{A}_1$.

Niech dany będzie zbiór literałów S. Próba jego unifikacji polega na znalezieniu ciągu podstawień, których złożenie jest mgu dla S lub orzeczeniu, że S nie jest uzgadnialny, w przypadku gdy taki ciąg nie istnieje.

Krok 0**.** Niech $S_0 = S$ oraz $\sigma_0 = \epsilon$.

Krok $k + 1$**.** Jeśli zbiór S_k ma tylko jeden element, to algorytm kończy pracę: złożenie $\sigma_0\sigma_1 \ldots \sigma_k$ jest mgu dla S.

W przeciwnym przypadku sprawdzamy czy istnieje zmienna x oraz term t nie zawierający zmiennej x takie, że $x \in D(S_k)$ oraz $t \in D(S_k)$:

- Jeśli nie, to algorytm kończy pracę: S nie posiada mgu.
- Jeśli tak, to niech x oraz t będą najmniejszą taką parą termów (w ustalonym porządku termów). Niech $\sigma_{k+1} = \{x \mapsto t\}$ oraz $S_{k+1} = S_k\sigma_{k+1}$ i przechodzimy do kroku $k + 2$.

Rozważmy (za Nerode, Shore 1997) przykład ilustrujący działanie tego algorytmu. Niech:

$$S = \{P(f(y, g(z)), h(b)), P(f(h(w), g(a)), t), P(f(h(b), g(z)), y)\}.$$

Krok 0**.** $S = S\epsilon = S_0\sigma_0$.

Krok 1**.** S_0 nie jest zbiorem jednoelementowym. Mamy: $D(S_0) = \{y, h(w), h(b)\}$. W zależności od określenia uporządkowania termów, są dwie możliwości dla σ_1:

- $\sigma_1 = \{y \mapsto h(w)\}$,
- $\sigma_1 = \{y \mapsto b\}$.

Przypuśćmy, że wybierzemy pierwszą możliwość (choć druga jest lepsza, jak zobaczymy w kroku 2). Jeśli $\sigma_1 = \{y \mapsto h(w)\}$, to:

$$S_1 = S_0\sigma_1 = \{P(f(h(w), g(z)), h(b)), P(f(h(w), g(a)), t), P(f(h(b), g(z)), h(w))\}.$$

Krok 2. Mamy: $D(S_1) = \{w, b\}$, więc niech $\sigma_2 = \{w \mapsto b\}$. Wtedy:

$$S_2 = S_1\sigma_2 = \{P(f(h(b), g(z)), h(b)), P(f(h(b), g(a)), t), P(f(h(b), g(z)), h(b))\}.$$

Krok 3. Mamy $D(S_2) = \{z, a\}$, a więc $\sigma_3 = \{z \mapsto a\}$. Wtedy:

$$S_3 = S_2\sigma_3 = \{P(f(h(b), g(a)), h(b)), P(f(h(b), g(a)), t), P(f(h(b), g(a)), h(b))\}.$$

Krok 4. Mamy $D(S_3) = \{t, h(b)\}$. Wtedy $\sigma_4 = \{t \mapsto h(b)\}$ i otrzymujemy:

$$S_{34} = S_3\sigma_4 = \{P(f(h(b), g(a)), h(b)), P(f(h(b), g(a)), h(b)), P(f(h(b), g(a)), h(b))\}.$$

Krok 5. S_4 jest zbiorem jednoelementowym, a mgu dla S_4 jest:

$$\sigma_1\sigma_2\sigma_3\sigma_4 = \{y \mapsto h(w)\}\{w \mapsto b\}\{z \mapsto a\}\{t \mapsto h(b)\} = \{y \mapsto h(b)\}\{w \mapsto b\}\{z \mapsto a\}\{t \mapsto h(b)\}.$$

Można udowodnić, że opisany wyżej algorytm $\mathfrak{A}_1$ jest poprawny:

TWIERDZENIE 24.3.1. Dla dowolnego zbioru S algorytm $\mathfrak{A}_1$ kończy pracę w pewnym kroku $k + 1$ podając prawidłową odpowiedź, tj.:

- albo S nie jest uzgadnialny, albo
- $\psi = \sigma_0\sigma_1 \ldots \sigma_k$ jest mgu dla S.

Nadto, dla dowolnego unifikatora θ dla S mamy: $\theta = \psi\theta$.

DOWÓD. Po pierwsze, algorytm oczywiście zatrzymuje się, ponieważ w każdym kroku (poza ostatnim) eliminujemy wszystkie wystąpienia jednej ze ***skończonej*** liczby zmiennych w S. Po drugie, jest także oczywiste, że jeśli algorytm daje odpowiedź, iż S nie jest uzgadnialny, to S nie jest uzgadnialny. Tym, co być może nie jest oczywiste, jest to, że $\psi = \sigma_0\sigma_1 \ldots \sigma_k$ jest unifikatorem dla S. Niech θ bedzie dowolnym unifikatorem dla S. Musimy pokazać, że $\theta = \psi\theta$. Dokonamy tego przez indukcję, pokazując, że dla każdego i mamy: $\theta = \sigma_0\sigma_1 \ldots \sigma_i\theta$.

Dla $i = 0$ powyższe stwierdzenie oczywiście zachodzi. Przypuśćmy, że mamy $\theta = \sigma_0\sigma_1 \ldots \sigma_i\theta$ oraz że $\sigma_{i+1} = \{v \mapsto t\}$. Wystarczy pokazać, że podstawienia $\sigma_{i+1}\theta$ oraz θ są równe. Pokażemy, że dają one ten sam wynik dla każdej zmiennej. Dla $x \neq v$ $x\sigma_{k+1}\theta$ oraz $x\theta$ są rzecz jasna równe. Dla v mamy: $v\sigma_{i+1}\theta = t\theta$. Ponieważ θ jest unifikatorem dla $S\sigma_0\sigma_1 \ldots \sigma_i$, a v oraz t należą do $D(S\sigma_0\sigma_1 \ldots \sigma_i)$, więc θ musi być unifikatorem również dla v oraz t, czyli $t\theta = v\theta$. To kończy dowód.

Algorytm $\mathfrak{A}_1$ jest prosty w opisie, ale jednocześnie bardzo mało efektywny. Istotnie, pracuje on w czasie wykładniczym, co nie jest w żadnym rozsądnym rozumieniu efektywne.

24.3.2. Algorytm $\mathfrak{A}_2$: algorytm Robinsona

Oryginalny algorytm Robinsona także opisywany jest w wielu podręcznikach, zob. np.: Ben-Ari 2005 (strony 120–121), Baader, Snyder 2001 (strony 453–454). Uogólnienie tego algorytmu, z użyciem pojęcia obliczonego unifikatora podaje np. Letz 1999, strona 162:

Niech S będzie skończonym zbiorem wyrażeń (bez kwantyfikatorów). Niech $\sigma_0 = \emptyset$, $S_0 = S$ oraz $k = 1$. Przejdź do (1).

(1) Jeśli S_k jest zbiorem jednoelementowym, to podaj σ_k jako obliczony unifikator dla S. W przeciwnym przypadku wybierz zbiór niezgodności D_k dla S_k i przejdź do (2).

(2) Jeśli D_k jest postaci $\{x, t\}$, gdzie t jest termem nie zawierającym zmiennej x, to niech $\sigma_{k+1} = \sigma_k\{x \mapsto t\}$ oraz $S_{k+1} = S_k\{x \mapsto t\}$. Powiększ k o 1 i przejdź do (1). W przeciwnym przypadku daj odpowiedź: S nie jest uzgadnialny.

24.3.3. Algorytm $\mathfrak{A}_3$: algorytm Herbranda

Algorytm, który jest bardzo bliski oryginalnym pomysłom Herbranda opisano np. w artykule Baader, Snyder 2001 (strony 454–458). Przedstawiony jest on jako pewien system reguł inferencji.

Wykorzystuje się przy tym fakt, że każde podstawienie idempotentne może być reprezentowane przez pewien układ równań w postaci *rozwiązanej* (tj. takiej, że każda zmienna występuje tylko raz w tym układzie, jako jedna ze stron równości).

Problem unifikacji przekształca się, z pomocą wspomnianych reguł, w problem rozwiązania układu równań termów.

24.3.4. Inne algorytmy

Przegląd niektórych dalszych algorytmów unifikacji dla klasycznej logiki pierwszego rzędu podano np. w *Handbook of automated reasoning*. W cytowanym już kilkakrotnie artykule Baader, Snyder 2001 z tego podręcznika znajdujemy np. opisy następujących algorytmów:

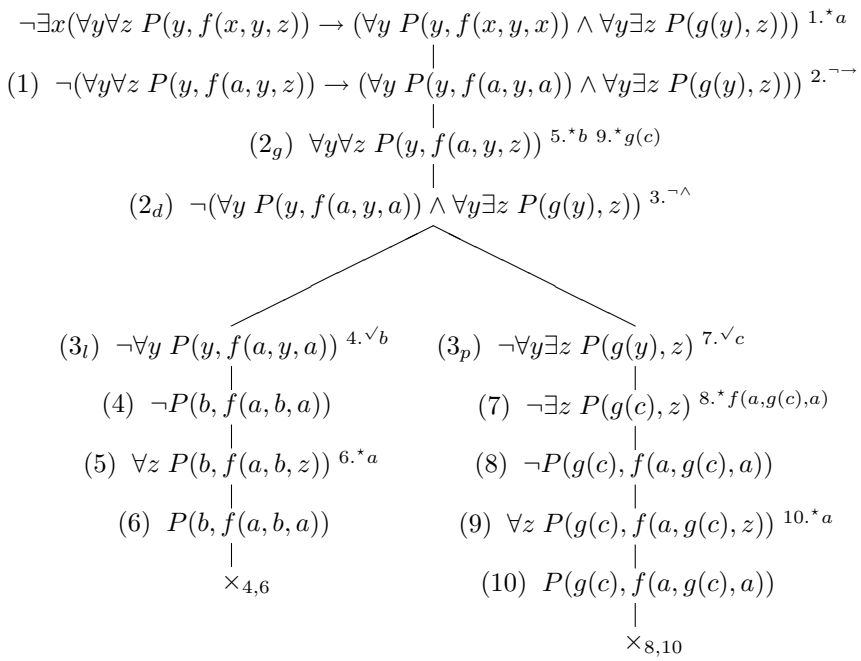
- unifikacja tzw. *dag*-ów termów (dag jest drzewową reprezentacją budowy termu),
- unifikacja prawie liniowa (wykorzystująca pewne relacje równoważności na termach),
- różne rodzaje E-unifikacji, tj. unifikacji wykorzystującej zbiory identyczności termów.

Nie opisujemy dokładniej żadnego z wymienionych wyżej algorytmów, jako że nie jest to potrzebne dla celów niniejszego skryptu. To, co naprawdę istotne, to samo pojęcie unifikacji. Będzie ono wykorzystane poniżej, w opisie pewnego rodzaju drzew semantycznych.

24.4. Tablice analityczne ze zmiennymi wolnymi

Pamiętamy, że reguły $R(\forall)$ oraz $R(\neg\exists)$ powinny być zastosowane dla ***każdego*** termu (bez zmiennych), występującego na rozważanej gałęzi tablicy analitycznej. Fakt ten jest kłopotliwy ze względu na efektywność procesu zamykania gałęzi. Często nie jest od razu widoczne, które zastosowania reguł $R(\forall)$ oraz $R(\neg\exists)$ do stosownych termów wystarczą do zamknięcia rozważanej gałęzi. Z problemem tym zetknęliśmy się kilkakrotnie poprzednio. Dowody przeprowadzane przy użyciu tablic analitycznych niekoniecznie są czymś w rodzaju procedury algorytmicznej — czasem pomysłowe dobranie kolejnych kroków dowodowych znacznie upraszcza pracę. Problematyka ta jest również istotna w zastosowaniach tablic analitycznych w *automatycznym dowodzeniu twierdzeń*. Nawet dla najszybszego komputera może być kłopotliwe wykonywanie ***wszystkich*** możliwych w danym momencie kroków. Jednym z przydatnych rozwiązań są tzw. *tablice analityczne ze zmiennymi wolnymi*, które krótko opiszemy poniżej.

Dla zilustrowania problemów, o których mowa, rozważmy następujące drzewo semantyczne (za Letz 1999, strona 147):



Drzewo ma wszystkie gałęzie zamknięte. Zwróćmy uwagę na następujące rzeczy:

- Formuła w korzeniu drzewa jest zanegowaną formułą egzystencjalną i nie zawiera żadnej stałej indywiduowej. W takim przypadku stosujemy regułę $R(\neg\exists)$ dla ***dowolnej*** stałej indywiduowej. Ponieważ zakładamy, że w sygnaturze rozważanego języka KRP mamy do dyspozycji przeliczalny zbiór stałych indywiduowych, krok taki jest z definicji wykonalny.
- Gałąź prawą zamknięto wykorzystując zastosowania reguł $R(\forall)$ (krok 9.) oraz $R(\neg\exists)$ (krok 8.). Istotne przy tym było trafne dobranie termów: w kroku 8 termu $f(a,g(c),a)$, a w kroku 9 termu $g(c)$.
- Stosowana przez nas notacja ma pewien minus (w odróżnieniu od notacji Letza). W naszej notacji, wynik zastosowania kroku 9 (czyli formuła $\forall z\ P(g(c),f(a,g(c),z))$) powinien zostać wpisany na ***obu*** gałęziach drzewa. Widać, że formuła (9) nie ma żadnego wpływu na zamknięcie gałęzi lewej. Zastosowaliśmy (nielegalne!) uproszczenie, nie wpisując (9) na lewej gałęzi. Nadto, w kroku 7 wprowadziliśmy nową stałą c, choć równie dobrze można było (jak u Letza) posłużyć się stałą b. Notacja Letza różni się od naszej tym, że informacja o wykonywanym kroku umieszczana jest na ***gałęzi*** drzewa, przed wynikiem wykonania tego kroku (a nie z prawej strony formuły, do której stosujemy rozważany krok dowodowy). Tak więc, informacja o kroku 9 byłaby u Letza umieszczona na krawędzi między formułami o numerach 8 i 9. Wtedy jest wyraźnie widoczne, że wynik wykonania kroku 9 dotyczy tylko prawej gałęzi drzewa. W rozważanym tu przypadku nasza notacja nie prowadzi do błędu logicznego, ale każe zapisywać nieistotną (dla zamknięcia drzewa) informację na gałęzi lewej. Z drugiej strony, można zastanawiać się, czy notacja Letza nie gubi jakiejś istotnej informacji — skoro dokonujemy pewnej operacji na formule należącej do pnia drzewa, to wynik tej operacji ***powinien*** być znaczący dla wszystkich formuł „potomnych".

Do tego przykładu powrócimy niebawem, pokazując, jak można uzasadnić taki, a nie inny dobór termów w krokach 8 i 9.

24.4.1. Definicje

Przypominamy, że w podrozdziale 18.6.1. omówiono pojęcie ***skolemizacji***. Będzie ono potrzebne poniżej.

Podstawowa idea wprowadzania tablic analitycznych ze zmiennymi wolnymi jest następująca. Zamiast reguł $R(\forall)$ oraz $R(\neg\exists)$ wprowadzamy regułę pozwalającą przejść od formuły generalnie skwantyfikowanej (lub negacji formuły

egzystencjalnie skwantyfikowanej) do formuły bez kwantyfikatora ogólnego (lub zanegowanego kwantyfikatora egzystencjalnego), w której zmienną dotąd wiązaną przez opuszczany kwantyfikator zastępujemy nową zmienną wolną. Zamiast reguł $R(\exists)$ oraz $R(\neg\forall)$ wprowadzamy regułę pozwalającą przejść od formuły egzystencjalnie skwantyfikowanej (lub negacji formuły generalnie skwantyfikowanej) do formuły bez kwantyfikatora egzystencjalnego (lub zanegowanego kwantyfikatora generalnego), w której zmienną dotąd wiązaną przez opuszczany kwantyfikator zastępujemy termem złożonym: nowym symbolem funkcyjnym od argumentów, które są wszystkimi zmiennymi wolnymi na rozważanej gałęzi. Wreszcie, dodajemy regułę *domknięcia*, pozwalającą dodać do każdej otwartej gałęzi drzewa dowolne podstawienie, które jest wolne dla wszystkich formuł z tej gałęzi. W ten sposób problem wielokrotnego stosowania reguł $R(\forall)$ oraz $R(\neg\exists)$ redukujemy do problemu znalezienia unifikatora dla zbioru literałów (na danej gałęzi). Nie musimy stosować reguł $R(\forall)$ oraz $R(\neg\exists)$ „na ślepo", wystarczy, że potrafimy znaleźć taki unifikator, który pozwoli zamknąć rozważaną gałąź (o ile istotnie daje się on zamknąć).

Jeśli σ jest podstawieniem, to dla dowolnej zmiennej x zdefiniujmy σ_x w sposób następujący:

- $y\sigma_x = y\sigma$, jeśli $y \neq x$,
- $y\sigma_x = x$, jeśli $y = x$.

Podstawienia mogą zostać rozszerzone (z odwzorowań ze zbioru zmiennych w zbiór formuł) w następujący znany sposób:

- $A(t_1, \ldots, t_n)\sigma = A(t_1\sigma, \ldots, t_n\sigma)$, gdy A jest formułą atomową.
- $(\neg A)\sigma = \neg(A\sigma)$.
- $(A\S B)\sigma = A\sigma\S B\sigma$ dla $\S \in \{\wedge, \vee, \rightarrow, \equiv\}$.
- $(\forall x A)\sigma = \forall x(A\sigma_x)$.
- $(\exists x A)\sigma = \exists x(A\sigma_x)$.

Indukcyjna definicja podstawienia σ ***wolnego dla formuły*** A ma postać następującą:

- Jeśli A jest formułą atomową, to σ jest wolne dla A.
- σ jest wolne dla $\neg B$ wtedy i tylko wtedy, gdy σ jest wolne dla B.
- σ jest wolne dla $B\S C$ wtedy i tylko wtedy, gdy σ jest wolne dla B oraz σ jest wolne dla C, dla $\S \in \{\wedge, \vee, \rightarrow, \equiv\}$.
- σ jest wolne dla $\forall x A$ oraz $\exists x A$ o ile: σ_x jest wolne dla A oraz jeśli y jest zmienną wolną w A różną od x, to $y\sigma$ nie zawiera x.

Niech σ będzie podstawieniem, a T drzewem semantycznym. Przez $T\sigma$ rozumiemy drzewo semantyczne powstające z T poprzez zastąpienie wszystkich formuł A w T przez formuły $A\sigma$.

Mówimy, że podstawienie σ jest ***wolne*** dla drzewa T, jeśli σ jest wolne dla wszystkich formuł w T.

Możemy teraz podać formalne wersje potrzebnych reguł:

- ***Reguła dla formuł generalnie skwantyfikowanych:***

$$R(\forall) \qquad \begin{array}{c} \forall x\ A(x) \\ | \\ A(y/x) \end{array}$$

dla nowej zmiennej y, która nie jest związana w formułach drzewa.

- ***Reguła dla formuł egzystencjalnie skwantyfikowanych:***

$$R(\exists) \qquad \begin{array}{c} \exists x\ A(x) \\ | \\ A(f(x_1,\ldots,x_n)/x) \end{array}$$

dla nowego symbolu funkcyjnego f oraz wszystkich zmiennych wolnych występujących dotąd na rozważanej gałęzi.

- ***Reguła dla negacji formuł generalnie skwantyfikowanych:***

$$R(\neg\forall) \qquad \begin{array}{c} \neg\forall x\ A(x) \\ | \\ \neg A(f(x_1,\ldots,x_n)/x) \end{array}$$

dla nowego symbolu funkcyjnego f oraz wszystkich zmiennych wolnych występujących dotąd na rozważanej gałęzi.

- ***Reguła dla negacji formuł egzystencjalnie skwantyfikowanych:***

$$R(\neg\exists) \qquad \begin{array}{c} \neg\exists x\ A(x) \\ | \\ \neg A(y/x) \end{array}$$

dla nowej zmiennej y, która nie jest związana w formułach drzewa.

- ***Reguła podstawień:***

Jeśli σ jest wolne dla drzewa T,
to drzewo T można rozszerzyć do drzewa $T\sigma$.

Zastosowania reguł $R(\forall)$ oraz $R(\neg\exists)$ dające w wyniku wprowadzenie nowych zmiennych wolnych będziemy zaznaczać z prawej strony odnośnej formuły, w górnej frakcji. Wprowadzenie w kroku $n.$ zmiennej x będzie zaznaczane przy tym przez symbol $^{n.^{*}x}$. Proszę zwrócić uwagę na różnicę kształtu symboli: $\star$ oraz $*$.

Zastosowania reguł $R(\exists)$ oraz $R(\neg\forall)$ dające w wyniku wprowadzenie nowych symboli funkcyjnych będziemy zaznaczać z prawej strony odnośnej formuły, w górnej frakcji. Wprowadzenie w kroku $n.$ symbolu funkcyjnego f będzie zaznaczane przy tym przez symbol $^{n.^{\checkmark}f}$. Wprowadzenie zeroargumentowego symbolu funkcyjnego, czyli stałej indywiduowej będzie zaznaczane jak poprzednio, symbolem $\surd$ w górnej frakcji. Proszę zwrócić uwagę na różnicę kształtu symboli: $\checkmark$ oraz $\surd$.

Zastosowanie reguły podstawień będziemy zaznaczać wpisując w każdej otwartej gałęzi drzewa stosowne podstawienie. Ponieważ podstawienie dotyczy całego drzewa, więc numer tego kroku można byłoby wpisywać (umownie) z prawej strony wierzchołka drzewa, wraz ze zmienną, której dotyczy podstawienie. Wybierzemy jednak inne rozwiązanie. Wynik działania tego kroku, tj. stosowne podstawienie, otrzyma swój numer ***z kropką*** z lewej strony. W ten sposób, zarówno wszystkie kroki, jak i ich wyniki będą jednoznacznie rozpoznawalne w drzewie. Wreszcie, z prawej strony wiersza zawierającego podstawienie pisać będziemy numery formuł, do których podstawienie stosujemy.

Jak poprzednio, gałąź tablicy jest ***zamknięta***, gdy występuje na niej para formuł wzajem sprzecznych: A oraz $\neg A$. Tablica jest ***zamknięta***, jeśli wszystkie jej gałęzie są zamknięte.

Rozważmy proste przykłady.

24.4.2. Przykłady

Przykład 24.4.2.1. (Fitting 1990, 153–154).

Pokażemy, że formuła:

$$(\bigstar) \quad \exists w \forall x\ R(x, w, f(x, w)) \to \exists w \forall x \exists y\ R(x, w, y)$$

jest tautologią KRP. W tym celu budujemy tablice analityczną jej negacji:

$$\neg\exists w \forall x\ R(x, w, f(x, w)) \to \exists w \forall x \exists y\ R(x, w, y)\ ^{1.\neg\to}$$
$$|$$
$$(1_g)\ \ \exists w \forall x\ R(x, w, f(x, w))\ ^{2.\checkmark a}$$
$$|$$
$$(1_d)\ \ \neg\exists w \forall x \exists y\ R(x, w, y)\ ^{3.* v_1}$$
$$|$$
$$(2)\ \ \forall x\ R(x, a, f(x, a))\ ^{5.* v_2}$$
$$|$$
$$(3)\ \ \neg\forall x \exists y\ R(x, v_1, y)\ ^{4.\checkmark g}$$
$$|$$
$$(4)\ \ \neg\exists y\ R(g(v_1), v_1, y)\ ^{6.* v_3}$$
$$|$$
$$(5)\ \ R(v_2, a, f(v_2, a))\ ^{10.8.}$$
$$|$$
$$(6)\ \ \neg R(g(v_1), v_1, v_3)\ ^{11.7.,9.}$$
$$|$$
$$(7.)\ \ v_1 \mapsto a$$
$$|$$
$$(8.)\ \ v_2 \mapsto g(a)$$
$$|$$
$$(9.)\ \ v_3 \mapsto f(g(a), a)$$
$$|$$
$$(10)\ \ R(g(a), a, f(g(a), a))$$
$$|$$
$$(11)\ \ \neg R(g(a), a, f(g(a), a))$$
$$|$$
$$\times_{10,11}$$

Jedyna gałąź tej tablicy jest zamknięta. Tak więc, nie istnieje interpretacja, w której formuła w korzeniu byłaby prawdziwa, a stąd formuła $(\bigstar)$ jest tautologia KRP. Zauważmy, że do zamknięcia jedynej gałęzi rozważanej tablicy wykorzystano:

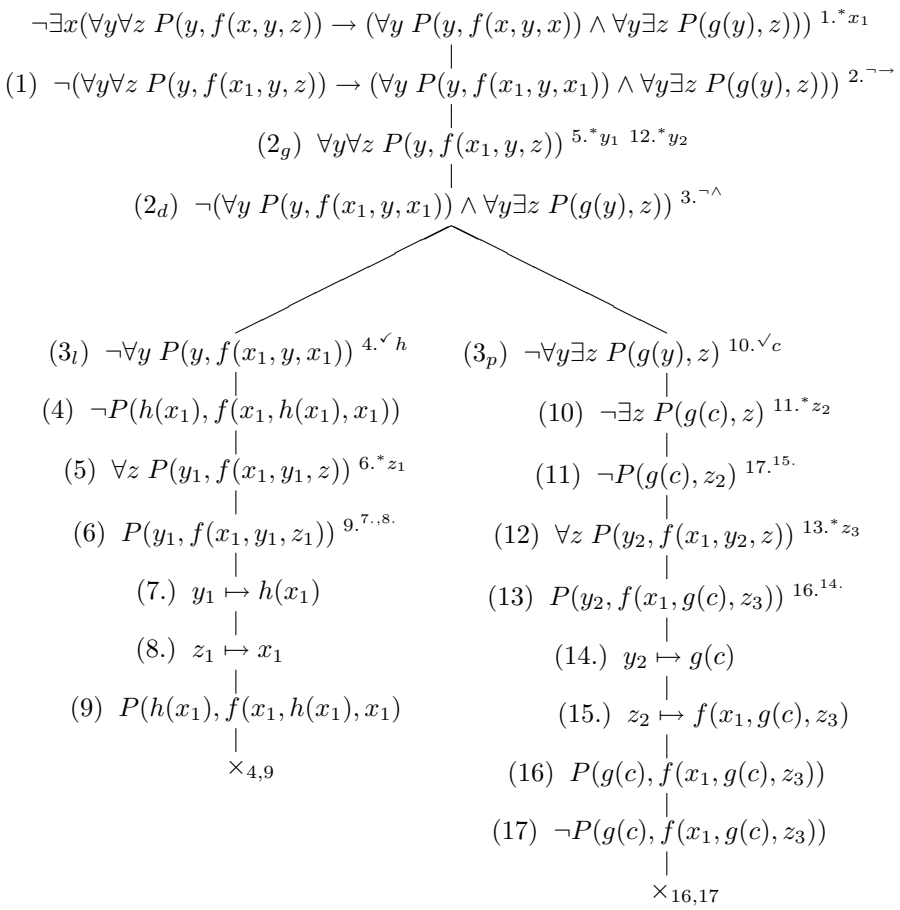
- wprowadzenie nowych zmiennych wolnych,
- podstawienia odpowiednich termów za zmienne wolne.

W kroku 11 dokonaliśmy jednocześnie stosownych podstawień za zmienne v_1 oraz v_3 (zamiast rozłożyć ten krok na dwa kroki elementarne, każdy z podstawieniem za jedną zmienną).

Uwaga. Można stosować pewne uproszczenia używanej dotąd notacji. Dla przykładu, można łączyć w jedno podstawienie kilka podstawień za poszczególne zmienne. Ważne przy tego rodzaju uproszczeniach jest oczywiście to, aby: nie popełnić błędu logicznego, aby zastosowanie uproszczeń było rozpoznawalne, aby otrzymany diagram był przejrzysty, itp.

Przykład 24.4.2.2. (Letz 1999, 168).

Wrócimy teraz do przykładu rozważanego na początku 24.4.



Mogłoby się wydawać, że wprowadzenie zmiennych wolnych do tablic analitycznych tylko utrudnia dowodzenie, zamiast je ułatwiać. Powyższe drzewo ma więcej wierzchołków niż oryginalne drzewo rozważane na początku III.7.3. Jest jednak inaczej. Zauważmy, że:

- zastosowania reguł $R(\forall)$ oraz $R(\neg\exists)$ zostały ograniczone do minimum; m.in. nie stosujemy tych reguł dla ***każdego*** termu na rozważanej gałęzi;
- nowe funkcje wprowadzone przez reguły $R(\exists)$ oraz $R(\neg\forall)$ mają prostą, naturalną interpretację: są funkcjami wprowadzanymi przez ***skolemizację***;
- wreszcie, to co najważniejsze: problem zamykania gałęzi drzewa został sprowadzony do problemu znalezienia ***unifikatora*** zbioru literałów; jak wiemy z III.7.2., ten ostatni problem jest rozwiązywalny w sposób algorytmiczny; widać więc tu chyba wyraźnie, że dobór termów bez zmiennych w podstawieniach nie jest przypadkowy.

Warto próbować sobie wyobrazić bardziej skomplikowane przykłady formuł, np. z wielokrotnymi kwantyfikatorami generalnymi oraz z dużą liczbą symboli funkcyjnych. W takich przypadkach tablice analityczne ze zmiennymi wolnymi są istotnie bardziej przydatne od „zwykłych" tablic analitycznych.

* * *

Jest nieprzebrane mnóstwo różnych rodzajów tablic analitycznych. Nie jest celem tego skryptu opisywanie tego bogactwa. Zainteresowanych zapraszamy do czytania literatury przedmiotu.

Wykład 25: Rezolucja w KRP

Pokażemy teraz działanie pewnej metody dowodowej, związanej z metodą tablic analitycznych i mającej istotne zastosowania m.in. w automatycznym dowodzeniu twierdzeń.

25.1. Definicje

Jeśli P jest predykatem, to stosujemy skrótowe zapisy $P(\vec{x})$ oraz $P(\vec{t})$ dla formuł atomowych utworzonych z predykatu P oraz stosownej liczby jego argumentów — zmiennych (w pierwszym przypadku) lub termów (w drugim przypadku).

Klauzulą nazwiemy dowolny skończony zbiór literałów.

Literałem ***komplementarnym*** do literału ℓ nazywamy literał $\overline{\ell}$, zdefiniowany następująco:

- jeśli ℓ jest literałem pozytywnym ℓ', to $\overline{\ell}$ jest literałem negatywnym $\neg\ell'$
- jeśli ℓ jest literałem negatywnym $\neg\ell'$, to $\overline{\ell}$ jest literałem pozytywnym ℓ'.

Klauzulę pustą (nie zawierającą żadnych elementów) oznaczamy przez $\square$.

Klauzule zawierające najwyżej jeden literał pozytywny nazywamy ***klauzulami Hornowskimi***.

Klauzulą programową nazywamy każdą klauzulę z dokładnie jednym literałem pozytywnym.

Jeśli klauzula programowa zawiera jakieś literały negatywne, to nazywamy ją ***regułą***; w przeciwnym przypadku nazywamy ją ***faktem***.

Klauzulą celową nazywamy klauzulę bez literałów pozytywnych.

Programem nazywamy zbiór klauzul programowych (reguł lub faktów). Programy odpowiadają programom rozważanym w PROLOGu.

Klauzule reprezentują formuły w skolemowej postaci normalnej. Tak więc, np. klauzula $\{\neg P(x), Q(x)\}$ reprezentuje formułę $\forall x \forall y\ (\neg P(x) \vee Q(x))$ lub, co na jedno wychodzi, formułę $\forall x \forall y\ (P(x) \to Q(x))$.

Niech C_1 i C_2 będą dwiema klauzulami, które nie mają żadnych wspólnych zmiennych i są postaci:

- $D_1 \cup \{P(\overrightarrow{t_1}), \ldots, P(\overrightarrow{t_n})\}$ oraz
- $D_2 \cup \{\neg P(\overrightarrow{s_1}), \ldots, \neg P(\overrightarrow{s_m})\}$, odpowiednio.

Jeśli σ jest najbardziej ogólnym unifikatorem dla $\{P(\overrightarrow{t_1}), \ldots, P(\overrightarrow{t_n}), P(\overrightarrow{s_1}), \ldots, P(\overrightarrow{s_m})\}$, to $D_1\sigma \cup D_2\sigma$ jest ***rezolwentą*** dla C_1 i C_2. Czasem mówi się wtedy także, że $D_1\sigma \cup D_2\sigma$ jest ***dzieckiem*** swoich ***rodziców*** C_1 oraz C_2.

Dowodem rezolucyjnym klauzuli C ze zbioru formuł S nazywamy każdy skończony ciąg klauzul $C_1, \ldots, C_n$ taki, że:

- C jest identyczna z C_n
- każda klauzula C_i $(1 \leqslant i \leqslant n)$ jest albo elementem zbioru S albo rezolwentą pewnych klauzul C_j oraz C_k dla $j, k < i$.

Jeśli istnieje dowód rezolucyjny C z S, to mówimy, że C jest ***rezolucyjnie dowodliwa*** z S i oznaczamy ten fakt przez $S \vdash_R C$.

Każdy dowód rezolucyjny klauzuli pustej $\square$ ze zbioru S nazywamy ***rezolucyjną refutacją*** S. Jeżeli istnieje rezolucyjna refutacja S, to mówimy, że S jest ***rezolucyjnie odrzucalny*** i oznaczamy ten fakt przez $S \vdash_R \square$.

Rezolucyjnym drzewem dowodowym klauzuli C ze zbioru S nazywamy każde drzewo binarne T o następujących własnościach:

- korzeniem T jest C
- liśćmi T są pewne elementy zbioru S
- pozostałe (oprócz korzenia i liści) wierzchołki T są klauzulami
- bezpośrednimi następnikami wierzchołka D niebędącego liściem są klauzule D_1 oraz D_2, których rezolwentą jest D.

Niech $res(S)$ będzie zbiorem zawierający wszystkie elementy S oraz rezolwenty wszystkich par elementów S. Dla $n \geqslant 1$, niech $res_{n+1} = res(res_n(S))$. Wreszcie, niech $\mathcal{R}(S)$ będzie sumą wszystkich zbiorów $res(S)$. Zbiór $\mathcal{R}(S)$ nazywamy ***domknięciem rezolucyjnym*** zbioru S.

Uwaga! W definicji dowodu rezolucyjnego oraz rezolucyjnego drzewa dowodowego dopuszczamy (jako przesłanki dowodu lub liście drzewa, odpowiednio) ***formuły otrzymane z elementów zbioru S przez zastosowanie podstawień przemianowujących zmienne.***

Uwaga. Rozważamy teraz drzewa, których wierzchołki są znakowane ***zbiorami*** formuł.

25.2. Przykłady dowodów rezolucyjnych

PRZYKŁAD 25.2.1. (Nerode, Shore 1997: 146–147).

Rezolwentą klauzul:

- $C_1 = \{Q(x), \neg R(y), P(x, y), P(f(z), f(z))\}$
- $C_2 = \{\neg N(u), \neg R(w), \neg P(f(a), f(a)), \neg P(f(w), f(w))\}$

jest klauzula:

$$C_3 = \{Q(f(a)), \neg R(f(a)), \neg N(u), \neg R(a)\}.$$

Aby się o tym przekonać, należy:

- zauważyć, że $C_1 = \{Q(x), \neg R(y)\} \cup \{P(x, y), P(f(z), f(z))\}$
- zauważyć, że $C_2 = \{\neg N(u), \neg R(w)\} \cup \{\neg P(f(a), f(a)), \neg P(f(w), f(w))\}$
- zastosować najbardziej ogólny unifikator $\sigma = \{x \mapsto f(a), y \mapsto f(a), z \mapsto a, w \mapsto a\}$ dla uzgodnienia zbioru literałów $\{P(x, y), P(f(z), f(z)), P(f(a), f(a)), P(f(w), f(w))\}$
- $\{Q(x), \neg R(y)\}\sigma = \{Q(f(a)), \neg R(f(a))\}$
- $\{\neg N(u), \neg R(w)\}\sigma = \{\neg N(u), \neg R(a)\}$.

PRZYKŁAD 25.2.2. (Nerode, Shore 1997: 147).

Pokażemy, że warunki przechodniości i symetrii, tj. warunki:

$$\forall x \forall y \forall z((P(x, y) \wedge P(y, z)) \rightarrow P(x, z))$$

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

implikują następujący warunek (euklidesowości):

$$\forall x \forall y \forall z((P(x, y) \wedge P(z, y)) \rightarrow P(x, z)).$$

Powyższe warunki mają następujące reprezentacje w postaci klauzul (po rozdzieleniu zmiennych):

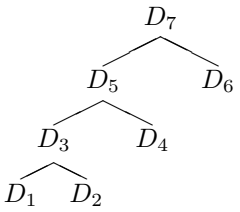
- $C_1 = \{\neg P(x,y), \neg P(y,z), P(x,z)\}$
- $C_2 = \{\neg P(u,v), P(v,u)\}$
- $C_3 = \{\neg P(x,y), \neg P(z,y), P(x,z)\}$.

Chcemy zatem uzyskać dowód rezolucyjny C_3 z C_1 oraz C_2. Będzie on się składał z trzech kroków. W każdym z nich z pary klauzul otrzymamy rezolwentę tej pary. W każdym kroku podkreślamy ten literał, względem którego dokonujemy rezolucji (tj. ten, który eliminujemy w wyniku danego kroku).

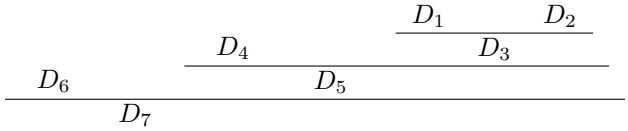
Tak więc, dowód rezolucyjny C_3 z C_1 oraz C_2 jest następującym ciągiem klauzul $D_1, \ldots D_7$:

- $D_1 = C_1 = \{\neg P(x,y), \neg P(y,z), \underline{P(x,z)}\}$
- $D_2 = C_2\{u \mapsto x, v \mapsto z\} = \{\neg \underline{P(u,v)}, P(v,u)\}\{u \mapsto x, v \mapsto z\} = \{\underline{\neg P(x,z)}, P(z,x)\}$
- $D_3 = \{\neg P(x,y), \neg P(y,z), \underline{P(z,x)}\}$ rezolwenta D_1 oraz D_2
- $D_4 = C_2\{u \mapsto z, v \mapsto x\} = \{\neg \underline{P(u,v)}, P(v,u)\}\{u \mapsto z, v \mapsto x\} = \{\underline{\neg P(z,x)}, P(x,z)\}$
- $D_5 = \{\neg P(x,y), \underline{\neg P(y,z)}, P(x,z)\}$ rezolwenta D_3 i D_4
- $D_6 = C_2\{u \mapsto z, v \mapsto y\} = \{\neg P(u,v), \underline{P(v,u)}\}\{u \mapsto z, v \mapsto y\} = \{\neg P(z,y), \underline{P(y,z)}\}$
- $D_7 = \{\neg P(x,y), \neg P(z,y), P(x,z)\} = C_3$ rezolwenta D_5 i D_6.

Rezolucyjne drzewo dowodowe wygląda w tym przypadku następująco:



Zwykle rezolucyjne drzewa dowodowe przedstawia się „korzeniem w dół, liśćmi do góry”. W takiej notacji rozważane rezolucyjne drzewo dowodowe wygląda następująco:



PRZYKŁAD 25.2.3. (Hedman 2004: 124-125).

Niech:

- $C_1 = \{Q(x,y), P(f(x),y)\}$
- $C_2 = \{R(x,c), \neg P(f(c),x), \neg P(f(y),h(z))\}$.

Chcemy znaleźć rezolwentę C_1 oraz C_2. Najpierw dokonamy przemianowania zmiennych (ponieważ pewne zmienne występują zarówno w C_1, jak i w C_2). Mamy: $C_1\{x \mapsto u, y \mapsto v\} = \{Q(u,v), P(f(u),v)\}$. Widać, że $C_1\{x \mapsto u, y \mapsto v\}$ oraz C_2 nie mają wspólnych zmiennych.

Zauważmy, że predykat P występuje zarówno w literałach pozytywnych, jak i negatywnych rozpatrywanych klauzul. Dla zbioru:

$$\{P(f(u),v), \neg P(f(c),x), \neg P(f(y),h(z))\}$$

będziemy zatem szukać najbardziej ogólnego unifikatora. Po zastosowaniu (ćwiczenie!) algorytmu unifikacji widzimy, że takim mgu jest:

$$\sigma = \{u \mapsto c, v \mapsto h(z), x \mapsto h(z)\}.$$

Tak więc, rezolwentą klauzul C_1 oraz C_2 jest klauzula:

$$R = (C_1\sigma - \{P(f(u), v)\}) \cup (C_2\sigma - \{\neg P(f(c), x), \neg P(f(y), h(z))\}) = \\ = \{Q(u, v), R(x, c)\}\sigma = \{Q(c, h(z)), R(h(z), c)\}.$$

Sprawdzimy, że R jest logiczną konsekwencją C_1 oraz C_2. Zgodnie z przyjętymi umowami notacyjnymi, klauzule reprezentują formuły w skolemowej postaci normalnej. Tak więc:

- C_1 reprezentuje formułę (1): $\forall x \forall y\ (Q(x, y) \vee P(f(x), y))$
- C_2 reprezentuje formułę (2): $\forall x \forall y \forall z\ (R(x, c) \vee \neg P(f(c), x) \vee \neg P(f(y), h(z)))$
- R reprezentuje formułę (3): $\forall z\ (Q(c, h(z)) \vee R(h(z), c))$.

Mamy pokazać, że dla dowolnej interpretacji $\mathfrak{M}$, jeśli formuły reprezentowane przez C_1 oraz C_2 są prawdziwe w $\mathfrak{M}$, to formuła reprezentowana przez R jest prawdziwa w $\mathfrak{M}$.

Przypuśćmy, że (1) i (2) są prawdziwe w interpretacji $\mathfrak{M}$. Ponieważ są to zdania uniwersalne, więc w $\mathfrak{M}$ prawdziwe są też wszystkie podstawienia dowolnych termów (bez zmiennych) za zmienne x oraz y w (1) i (2). W szczególności:

- (3) $\forall z\ (Q(c, h(z)) \vee P(f(c), h(z)))$, czyli formuła reprezentowana przez klauzulę $C_1\{x \mapsto u, y \mapsto v\}\sigma$
- (4) $\forall z\ (R(h(z), c) \vee \neg P(f(c), h(z)))$, czyli formuła reprezentowana przez klauzulę $C_2\sigma$

są obie prawdziwe w $\mathfrak{M}$. Łatwo zauważyć, że formuły (3) i (4) są, odpowiednio, równoważne z (5) oraz (6):

- (5) $\forall z\ (\neg Q(c, h(z)) \rightarrow P(f(c), h(z)))$
- (6) $\forall z\ (P(f(c), h(z)) \rightarrow R(h(z), c))$

Z (5) i (6), na mocy praw KRP, otrzymujemy: (7) $\forall z\ (\neg Q(c, h(z)) \rightarrow R(h(z), c))$. Z kolei, formuła (7) jest równoważna z (8): $\forall z\ (Q(c, h(z)) \vee R(h(z), c))$.

Pokazaliśmy, że jeśli (1) i (2) są prawdziwe w dowolnej interpretacji $\mathfrak{M}$, to także (8) jest prawdziwa w interpretacji $\mathfrak{M}$.

Formuła (8) jest reprezentowana przez klauzulę R, a więc zakończyliśmy dowód.

25.3. Przypomnienie: modele Herbranda

Przypomnijmy niektóre potrzebne pojęcia (uniwersa Herbranda, modele Herbranda).

Jeśli S jest dowolnym zbiorem formuł języka KRP (ustalonej sygnatury), to przez ***uniwersum Herbranda*** dla S rozumiemy zbiór H_S określony indukcyjnie następująco:

- (i) jeśli stała indywiduowa a_k występuje w jakiejś formule ze zbioru S, to $a_k \in H_S$
- (ii) jeśli $t_1, \ldots, t_{n_j}$ są dowolnymi termami należącymi do H_S, to $f_j^{n_j}(t_1, \ldots, t_{n_j})$ także należy do H_S, dla dowolnego symbolu funkcyjnego $f_j^{n_j}$.

Jeśli w formułach z S nie występuje żadna stała indywiduowa, to warunek (i) definicji zbioru H_S zastępujemy warunkiem: $a_k \in H_S$ dla dowolnie wybranej stałej indywiduowej a_k.

Jeśli w formułach z S występuje co najmniej jeden symbol funkcyjny, to H_S jest zbiorem nieskończonym.

Uniwersum Herbranda dla danego zbioru formuł S jest zatem zbiorem wszystkich termów bez zmiennych utworzonych (z użyciem symboli funkcyjnych) ze stałych indywiduowych występujących w formułach zbioru S.

Interpretacją Herbranda dla zbioru formuł S nazywamy interpretację $\langle H_S, \Delta_S \rangle$ spełniającą następujące warunki:

- $\Delta_S(a_k) = a_k$ dla dowolnej stałej indywiduowej a_k należącej do H_S;
- $\Delta_S(f_j^{n_j}(t_1, \ldots, t_{n_j})) = f_j^{n_j}(t_1, \ldots, t_{n_j})$ dla dowolnych termów $t_1, \ldots, t_{n_j}$ należących do H_S.

Modelem Herbranda dla zbioru formuł S nazywamy każdą interpretację Herbranda dla S, w której prawdziwe są wszystkie formuły z S.

Zauważmy, że uniwersa Herbranda tworzone są z wyrażeń języka KRP. ***Alfabetem Herbranda*** dla zbioru formuł S nazywamy zbiór wszystkich stałych pozalogicznych występujących w formułach z S (jeśli w S nie występuje żadna stała indywiduowa, to dodajemy dowolną ustaloną stałą indywiduową). Niech V_S oznacza alfabet Herbranda dla S.

Ważną konsekwencją twierdzenia Herbranda jest możliwość wykazania niespełnialności zbioru formuł języka KRP w KRZ.

TWIERDZENIE 25.3.1. Niech $\Gamma = \{A_1, A_2, \ldots, A_n \ldots\}$ będzie zbiorem formuł w skolemowych postaciach normalnych nie zawierających wystąpień symbolu identyczności. Wtedy: Γ jest spełnialny wtedy i tylko wtedy, gdy Γ ma model Herbranda.

DOWÓD. Jeśli Γ ma model Herbranda, to Γ jest oczywiście spełnialny. Pozostaje udowodnić implikację w drugą stronę.

Przypuśćmy, że Γ jest spełnialny. Niech $\mathfrak{N}$ będzie dowolną strukturą sygnatury V_S taką, że $\mathfrak{N} \models \Gamma$. Niech $\mathfrak{M}'$ będzie interpretacją Herbranda. Zbudujemy interpretację $\mathfrak{M}$ sygnatury V_S taką, że $\mathfrak{M} \models \Gamma$.

Uniwersum dla $\mathfrak{M}$ jest uniwersum Herbranda H_Γ. Trzeba podać interpretację w $\mathfrak{M}$ symboli funkcyjnych oraz predykatów. Mówiąc intuicyjnie:

- $\mathfrak{M}$ interpretuje symbole funkcyjne tak, jak robi to $\mathfrak{M}'$ (co nie wymaga uściśleń, ponieważ mowa tu o interpretacjach Herbranda: $\mathfrak{M}$ jest interpretacją Herbranda sygnatury V_Γ, tak samo jak $\mathfrak{M}'$);
- $\mathfrak{M}$ interpretuje predykaty tak, jak robi to $\mathfrak{N}$ (co wymaga uściślenia, bo uniwersa struktur $\mathfrak{M}$ oraz $\mathfrak{N}$ mogą być różne).

Dla dowolnego n-argumentowego predykatu R w V_Γ oraz termów $t_1, \ldots, t_n$ należących do H_Γ musimy określić, która z poniższych (nawzajem się wykluczających oraz dopełniających) możliwości zachodzi:

- $R(t_1, \ldots, t_n)$
- $\neg R(t_1, \ldots, t_n)$.

Ponieważ każdy z powyższych termów t_i jest termem bez zmiennych, a $\mathfrak{N}$ jest strukturą sygnatury V_Γ, więc zachodzi dokładnie jedno z dwojga:

- $\mathfrak{N} \models R(t_1, \ldots, t_n)$
- $\mathfrak{N} \models \neg R(t_1, \ldots, t_n)$.

Definiujemy interpretację predykatu R w $\mathfrak{M}$ w ten sposób, aby zachodziła równoważność:

$$\mathfrak{M} \models R(t_1, \ldots, t_n) \text{ wtedy i tylko wtedy, gdy } \mathfrak{N} \models R(t_1, \ldots, t_n).$$

Trzeba teraz pokazać, że $\mathfrak{M} \models \Gamma$. Dowód przeprowadzimy w trzech krokach:

- (1) pokażemy, że dla dowolnego zdania A języka sygnatury V_Γ, które nie zawiera ani kwantyfikatorów, ani znaku identyczności zachodzi: $\mathfrak{M} \models A$ wtedy i tylko wtedy, gdy $\mathfrak{N} \models A$.
- (2) pokażemy, że z (1) wynika, że dla dowolnego zdania A języka sygnatury V_Γ w skolemowej postaci normalnej, które nie zawiera znaku identyczności zachodzi: jeśli $\mathfrak{N} \models A$, to $\mathfrak{M} \models A$.
- (3) pokażemy, że z (2) wynika $\mathfrak{M} \models \Gamma$.

DOWÓD (1).

Niech A będzie formułą bez kwantyfikatorów. Pokażemy, że $\mathfrak{M} \models A$ wtedy i tylko wtedy, gdy $\mathfrak{N} \models A$ przez indukcję po złożoności A.

Jeśli A jest formułą atomową, to — ponieważ nie zawiera wystąpień predykatu identyczności — musi być postaci $R(t_1, \ldots, t_n)$ dla pewnego predykatu n-argumentowego R z alfabetu V_Γ oraz termów $t_1, \ldots, t_n$. Ponieważ A jest zdaniem, więc żaden z termów t_i nie może zawierac zmiennych. Oznacza to, że wszystkie termy t_i są elementami H_Γ. Z definicji $\mathfrak{M}$ otrzymujemy wtedy, że $\mathfrak{M} \models A$ wtedy i tylko wtedy, gdy $\mathfrak{N} \models A$.

Przypuśćmy, że:

- $\mathfrak{M} \models A_1$ wtedy i tylko wtedy, gdy $\mathfrak{N} \models A_1$
- $\mathfrak{M} \models A_2$ wtedy i tylko wtedy, gdy $\mathfrak{N} \models A_2$.

Wtedy oczywiście także:

- $\mathfrak{M} \models \neg A_1$ wtedy i tylko wtedy, gdy $\mathfrak{N} \models \neg A_1$
- $\mathfrak{M} \models A_1 \wedge A_2$ wtedy i tylko wtedy, gdy $\mathfrak{N} \models A_1 \wedge A_2$.

(podobnie dla innych spójników zdaniowych). To kończy dowód (1).

DOWÓD (2).

Dowód przeprowadzimy przez indukcję względem liczby kwantyfikatorów w A. Jeśli A nie zawiera żadnych kwantyfikatorów, to na mocy (1) mamy: $\mathfrak{M} \models A$ wtedy i tylko wtedy, gdy $\mathfrak{N} \models A$.

Przypuśćmy, że A jest postaci $\forall x_1 \ldots \forall x_n\ B$, gdzie B nie zawiera ani kwantyfikatorów, ani predykatu identyczności. Założenie indukcyjne głosi, że (2) zachodzi dla wszystkich formuł, które mają mniej niż n kwantyfikatorów. Niech $C(x_1)$ będzie formułą otrzymaną z $\forall x_1 \ldots \forall x_n\ B$ poprzez opuszczenie pierwszego kwantyfikatora. Niech t będzie termem bez zmiennych z języka o sygnaturze V_Γ (oznacza to, że t jest elementem H_Γ). Mamy:

- jeśli $\mathfrak{N} \models C(x_1)$, to $\mathfrak{N} \models C(t/x_1)$ (z definicji relacji $\models$),
- jeśli $\mathfrak{N} \models C(t/x_1)$, to $\mathfrak{M} \models C(t/x_1)$ (z założenia indukcyjnego).

Tak więc, jeśli $\mathfrak{N} \models C(x_1)$, to $\mathfrak{M} \models C(t/x_1)$. Term t był dowolnie wybranym elementem zbioru H_Γ. Mamy zatem: jeśli $\mathfrak{N} \models A$, to $\mathfrak{M} \models C(t/x_1)$ dla wszystkich $t \in H_\Gamma$. Ponieważ H_Γ jest uniwersum struktury $\mathfrak{M}$, więc z definicji relacji $\models$ otrzymujemy: $\mathfrak{M} \models \forall x_1 C(x_1)$. Ponieważ A jest identyczne z $\forall x_1 C(x_1)$, dowód (2) został zakończony.

DOWÓD (3).

Dla każdego $A_i \in \Gamma$ mamy:

- $\mathfrak{N} \models A_i$
- A_i jest w skolemowej postaci normalnej
- A_i nie zawiera predykatu identyczności.

Spełnione są zatem wszystkie założenia (2). Tak więc, $\mathfrak{N} \models \Gamma$. To kończy dowód (3), a zarazem całego twierdzenia 25.3.1.

Z powyższego twierdzenia wynika w szczególności, że:

- (†) Jeśli A jest formułą w skolemowej postaci normalnej bez predykatu identyczności, to: A jest spełnialna wtedy i tylko wtedy, gdy A ma model Herbranda.

Dlaczego w twierdzeniu 25.3.1. istotne było założenie, że Γ nie zawiera wystąpień znaku identyczności? Poniższy przykład stanowi odpowiedź na to pytanie.

PRZYKŁAD 25.3.1. (Hedman 2004: 115).

Rozważmy zdanie A postaci $\forall x\ ((f(x) \neq x) \wedge (f(f(x)) = x))$. Słownikiem Herbranda dla A jest zbiór $\{c, f\}$, gdzie c jest dowolną stałą indywiduową.

Uniwersum Herbranda dla A jest zbiorem nieskończonym: $H_A = \{c, f(c), f(f(c)), f(f(f(c))), \ldots\}$.

W każdej interpretacji Herbranda elementy c oraz $f(f(c))$ są oczywiście różne. Ponieważ konsekwencją A jest zdanie $\forall x\ f(f(x)) = x$, więc w szczególności $c = f(f(c))$ także jest konsekwencją A. Zdanie A nie może zatem posiadać żadnego modelu Herbranda. Łatwo jednak zobaczyć, że zbiór $\{A\}$ jest spełnialny: modelem dla A jest np. zbiór wszystkich liczb całkowitych (bez zera), w którym symbol funkcyjny f interpretujemy tak, aby $f(x) = -x$.

Jak radzić sobie w przypadku, gdy rozważany zbiór formuł zawiera wystąpienia znaku identyczności? Oto stosowna procedura.

Przypuśćmy, że A jest formułą w skolemowej postaci normalnej i że w A występuje predykat identyczności. Wtedy (†) nie zachodzi dla A. Zdefiniujemy formułę A^* taką, że:

- (†) zachodzi dla A^*
- (‡) A jest spełnialna wtedy i tylko wtedy, gdy A^* jest spełnialna.

Niech E będzie dwuargumentowym predykatem nie występującym w V_A. Poszukiwana formuła A^* jest koniunkcją formuł A_1, A_2, A_3 oraz A_4, zdefiniowanych następująco:

- A_1 jest formułą powstającą z A poprzez zastąpienie każdej równości termów $t_1 = t_2$ występującej w A przez formułę $E(t_1, t_2)$.
- A_2 jest koniunkcją warunków stwierdzających, że E denotuje relację równoważności (tj. zwrotną, przechodnią i symetryczną).
- Dla każdego n-argumentowego predykatu R z V_A niech A_R będzie formułą:

$$\forall x_1 \ldots \forall x_n \forall y_1 \ldots \forall y_n (\bigwedge_{i=1}^{n} (E(x_i, y_i \wedge R(x_1, \ldots, x_n))) \rightarrow R(y_1, \ldots, y_n)).$$

  Niech A_3 będzie koniunkcją wszystkich formuł A_R.

- Dla każdego n-argumentowego symbolu funkcyjnego f z V_A niech A_f będzie formułą:

$$\forall x_1 \ldots \forall x_n \forall y_1 \ldots \forall y_n (\bigwedge_{i=1}^{n} E(x_i, y_i) \rightarrow E(f(x_1, \ldots, x_n), f(y_1, \ldots, y_n))).$$

  Niech A_4 będzie koniunkcją wszystkich formuł A_f.

Niech A^* będzie skolemową postacią normalną koniunkcji formuł A_1, A_2, A_3 oraz A_4. Z konstrukcji A^* widać, że nie zawiera ona predykatu identyczności. Tak więc, (†) zachodzi dla A^*. Niech wykazanie, że zachodzi także (‡) będzie ćwiczeniem dla czytelniczek. Wskazówka: należy oczywiście rozważyć model ilorazowy.

Rozważmy jeszcze raz formułę A postaci $\forall x\ ((f(x) \neq x) \wedge (f(f(x)) = x))$ z przykładu 8.3.1.1. i zastosujmy do niej powyżej opisaną procedurę. Ponieważ formuła A^* spełnia warunek (†), więc A^* posiada model Herbranda o uniwersum $H_A = \{c, f(c), f(f(c)), f(f(f(c))), \ldots\}$. Możemy interpretować E w tym modelu jako relację równoważności o dwóch klasach: w jednej są termy zawierające parzystą liczbę wystąpień symbolu f, a w drugiej pozostałe termy. Wynika stąd również, że formuła A ma model dwuelementowy o uniwersum np. postaci $\{a, b\}$, w którym symbol funkcyjny f interpretujemy tak, aby: $f(a) = b$ oraz $f(b) = a$.

Teraz możemy opisać procedurę pozwalającą ustalać, czy dowolna formuła języka KRP jest niespełnialna.

Niech A będzie dowolną formułą języka KRP w skolemowej postaci normalnej, nie zawierającą predykatu identyczności. Tak więc, A jest postaci $\forall x_1 \ldots \forall x_n\ B(x_1, \ldots, x_n)$, gdzie B nie zawiera ani kwantyfikatorów, ani predykatu identyczności. Przypominamy, że H_A jest uniwersum Herbranda dla A. Zdefiniujmy zbiór:

$$E(A) = \{B(t_1, \ldots, t_n)\ :\ t_1, \ldots, t_n \in H_A\}.$$

Zbiór $E(A)$ otrzymujemy zatem przez podstawienia wszelkich możliwych termów z H_A za zmienne w B, na wszelkie możliwe sposoby. Niech $\{A_1, A_2, \ldots\}$ będzie wyliczeniem wszystkich elementów zbioru $E(A)$. Pokażemy, że A jest spełnialna wtedy i tylko wtedy, gdy $E(A)$ jest spełnialny.

LEMAT 25.3.2. Niech A będzie dowolną formułą języka KRP w skolemowej postaci normalnej, nie zawierającą predykatu identyczności. Wtedy: A jest spełnialna wtedy i tylko wtedy, gdy $E(A)$ jest spełnialny.

DOWÓD.

Dowód implikacji z lewa na prawo jest dość prosty. Jeśli $\mathfrak{M}$ jest modelem dla A, to:

$$\mathfrak{M} \models \forall x_1 \ldots \forall x_n\ B(x_1, \ldots, x_n).$$

W szczególności, $\mathfrak{M} \models B(t_1, \ldots, t_n)$ dla wszystkich $t_1, \ldots, t_n \in H_A$. Oznacza to, że $\mathfrak{M} \models A_i$ dla wszystkich i, a więc $\mathfrak{M} \models E(A)$.

Dla dowodu implikacji w drugą stronę załóżmy, że $E(A)$ jest spełnialny. Wtedy, na mocy twierdzenia 8.3.1., $E(A)$ ma model Herbranda $\mathfrak{M}$. Alfabet Herbranda dla $E(A)$ jest taki sam jak alfabet Herbranda dla A. Tak więc, uniwersum $\mathfrak{M}$ jest równe H_A. Dla wszystkich $t_1, \ldots, t_n \in H_A$, mamy $\mathfrak{M} \models B(t_1, \ldots, t_n)$, ponieważ $B(t_1, \ldots, t_n) \in E(A)$. Z definicji relacji $\models$ otrzymujemy, że $\mathfrak{M} \models \forall x_1 \ldots \forall x_n\ B(x_1, \ldots, x_n)$. Oznacza to, że $\mathfrak{M} \models A$, czyli że A jest spełnialna.

Z powyższego wynika, że A ***nie*** jest spełnialna wtedy i tylko wtedy, gdy $E(A)$ ***nie*** jest spełnialny. Ponieważ $E(A)$ zawiera jedynie zdania bez kwantyfikatorów, więc możemy uważać $E(A)$ za zbiór formuł języka KRZ (po stosownych podstawieniach zmiennych zdaniowych za formuły atomowe). Ponieważ A jest w skolemowej postaci normalnej, więc każda formuła w $E(A)$ jest w koniunkcyjnej postaci normalnej. W rozdziale II udowodniono, że zbiór S formuł języka KRZ jest niespełnialny wtedy i tylko wtedy, gdy $\square \in \mathcal{R}(S)$ (zob. też Lemat 25.4.2. poniżej). Wiemy zatem, że $E(A)$ jest niespełnialny wtedy i tylko wtedy, gdy $\square \in \mathcal{R}(E(A))$. Z TWIERDZENIA O ZWARTOŚCI dla KRZ (również udowodnionego w rozdziale II) wynika, że $E(A)$ jest niespełnialny wtedy i tylko wtedy, gdy pewien skończony podzbiór $\{A_1, \ldots, A_m\}$ zbioru $E(A)$ jest niespełnialny. Tak więc, jeśli A jest niespełnialna, to $\square \in \mathcal{R}(\{A_1, \ldots, A_m\})$ dla pewnego m. Zauważmy, że $\square \in \mathcal{R}(\{A_1, \ldots, A_m\})$ jest zbiorem ***skończonym***.

Procedura powyższa dostarcza metody ustalania, że A jest niespełnialna. Sprawdzamy, czy dla pewnego m zachodzi $\square \in \mathcal{R}(\{A_1, \ldots, A_m\})$. Jeśli odpowiedź jest twierdząca, to A jest niespełnialna. W przeciwnym przypadku sprawdzamy, czy $\square \in \mathcal{R}(\{A_1, \ldots, A_m, A_{m+1}\})$, itd. Jeśli A ***jest*** niespełnialna, to ta procedura poda tę odpowiedź po skończonej liczbie kroków. Jeśli natomiast A ***jest*** spełnialna, to omawiana procedura nie zakończy się.

Zauważmy, że nie ma żadnego ograniczenia (z góry) liczby kroków, w której powyższa procedura ewentualnie się zakończy.

25.4. Trafność i pełność rezolucyjna

Pokażemy teraz, że metoda rezolucji ma „porządne" własności metalogiczne, tj. że jest (w ściśle określonym sensie) trafna oraz pełna.

Przypuśćmy, że literały pozytywne $C_1, \ldots, C_n$ nie zawierają zmiennych wolnych. Niech S będzie dowolnym zbiorem klauzul. Chcielibyśmy, aby rezolucyjny dowód klazuli pustej $\square$ ze zbioru $S \cup \{\neg C_1, \ldots, \neg C_n\}$ implikował, że wszystkie C_i $(1 \leqslant i \leqslant n)$ wynikają logicznie ze zbioru S. Będzie to konsekwencją podanego niżej twierdzenia o pełności metody rezolucji.

Jeśli P jest programem, a $G = \{\neg A_1, \ldots \neg A_n\}$ klauzulą celową, to powiemy, że podstawienie θ (za zmienne z G) jest ***podstawieniem wyznaczającym poprawną odpowiedź***, gdy $(A_1 \wedge \ldots \wedge A_n)\theta$ wynika logicznie z (uniwersalnego domknięcia) P. W rozdziale IV pokażemy, że jeśli program $P \cup \{G\}$ nie jest spełnialny, to istnieje podstawienie wyznaczające poprawną odpowiedź o wartościach w zbiorze termów bez zmiennych. Będzie to konsekwencją twierdzenia Herbranda.

W wykładzie 10 omówiono metodę rezolucji dla KRZ. Pojęcia: dowodu rezolucyjnego, rezolucyjnej refutacji, rezolucyjnego drzewa dowodowego, domknięcia rezolucyjnego są w KRZ analogiczne, jak podane wyżej dla KRP (z oczywistymi modyfikacjami). Przypomnimy niżej (bez dowodów) pewne twierdzenia dotyczące rezolucji w KRZ. Będą one potrzebne w dowodach trafności i pełności rezolucji w KRP.

- LEMAT 25.4.1. Istnieje rezolucyjne drzewo dowodowe dla C z S wtedy i tylko wtedy, gdy C jest rezolucyjnie dowodliwa z S w KRZ.
- LEMAT 25.4.2. C jest rezolucyjnie dowodliwa z S w KRZ wtedy i tylko wtedy, gdy $C \in \mathcal{R}(S)$. W szczególności, istnieje rezolucyjna refutacja S wtedy i tylko wtedy, gdy $\Box \in \mathcal{R}(S)$.
- LEMAT 25.4.3. Jeśli $S = \{C_1, C_2\}$ jest spełnialny w KRZ oraz C jest rezolwentą C_1 i C_2, to C jest spełnialna w KRZ. Co więcej, każde wartościowanie zmiennych zdaniowych, które spełnia S, spełnia też C.
- LEMAT 25.4.4. Dla dowolnego zbioru formuł T języka KRZ oraz dowolnego literału ℓ: jeśli T jest niespełnialny, to niespełnialny jest także zbiór $\{C \in \mathcal{R}(T) \ : \ \ell, \overline{\ell} \notin C\}$.
- TWIERDZENIE 25.4.5. (***Trafność rezolucji w KRZ***). Jeżeli istnieje rezolucyjna refutacja S, to S nie jest spełnialny w KRZ.
- TWIERDZENIE 25.4.6. (***Pełność rezolucji w KRZ***). Jeżeli S jest niespełnialny w KRZ, to istnieje rezolucyjna refutacja S.

Dowody tych twierdzeń podano w wykładzie 10. Przypomnijmy jeszcze następujące porównanie reguły rezolucji z regułą modus ponens w KRZ:

- REGUŁA REZOLUCJI: z formuł $A \vee C$ oraz $\neg A \vee B$ wywnioskuj $C \vee B$
- REGUŁA MODUS PONENS: z formuł $A \rightarrow B$ oraz A wywnioskuj B (lub, w postaci równoważnej: z formuł $\neg A \vee B$ oraz A wywnioskuj B).

Najpierw udowodnimy trafność metody rezolucji, tj. pokażemy, że jeśli klauzula pusta należy do rezolucyjnego domknięcia zbioru S, to S nie jest spełnialny.

TWIERDZENIE 25.4.7. (***Trafność metody rezolucji w KRP***). Jeśli $\Box \in \mathcal{R}(S)$, to S nie jest spełnialny.

DOWÓD. Niech $\Box \in \mathcal{R}(S)$. Przypuśćmy, dla dowodu nie wprost, że S jest spełnialny, czyli że istnieje interpretacja $\mathfrak{M}$ taka, że $\mathfrak{M} \models S$.

Dla udowodnienia twierdzenia wystarczy pokazać, że:

- $(\bigstar)$ jeśli $\mathfrak{M} \models C_1$ i $\mathfrak{M} \models C_2$, to $\mathfrak{M} \models C$, gdzie C jest rezolwentą C_1 oraz C_2.

Istotnie, jeśli zachodzi $(\bigstar)$, to można pokazać (przez indukcję po złożoności formuł), że $\mathfrak{M} \models C$ dla wszystkich $C \in \mathcal{R}(S)$. Ponieważ $\Box \in \mathcal{R}(S)$, więc byłoby wtedy $\mathfrak{M} \models \Box$, a to jest sprzeczność.

Dowód $(\bigstar)$ jest uogólnieniem procedury, którą wykonywaliśmy w przykładzie 25.2.3.

Jeśli C jest rezolwentą C_1 i C_2, to C ma postać $D_1\sigma \cup D_2\sigma$ (gdzie D_1, D_2 oraz σ są oznaczeniami takimi, jakie występują w definicji rezolwenty). Przypominamy, że klauzula odpowiada formule generalnie skwantyfikowanej.

Dla każdego podstawienia τ, którego wartościami są termy bez zmiennych mamy:

- $\mathfrak{M} \models D_1\sigma\tau$ albo
- $\mathfrak{M} \models D_2\sigma\tau$.

W konsekwencji, także $C\tau$, które jest sumą $D_1\tau$ oraz $D_1\tau$, jest prawdziwe w $\mathfrak{M}$.

Tak więc, jeśli ze zbioru S klauzul da się rezolucyjnie wyprowadzić klauzulę pustą $\Box$ (reprezentującą sprzeczność), to zbiór S jest niespełnialny, nie ma modelu.

Pokażemy teraz dowód implikacji odwrotnej, tj. tego, że jeśli zbiór S jest niespełnialny, to można z niego rezolucyjnie wyprowadzić klauzulę pustą $\Box$.

Niech A będzie dowolną formułą języka KRP w skolemowej postaci normalnej, nie zawierającą predykatu identyczności. W Lemacie 25.3.2. pokazano, ze wtedy: A jest spełnialna wtedy i tylko wtedy, gdy $E(A)$ jest spełnialny. Ponieważ elementami zbioru $E(A)$ są zdania (!) otrzymane z A przez opuszczenie kwantyfikatorów oraz zastąpienie wszystkich zmiennych termami z H_A, więc elementy $E(A)$ można traktować jak formuły języka KRZ.

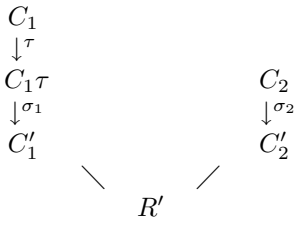
Przypuśćmy, że D_1 oraz D_2 są elementami $E(A)$ i że R' jest rezolwentą D_1 oraz D_2 ***w sensie rezolucji w KRZ***. Istnieją wtedy klauzule C_1 oraz C_2 otrzymane z A (tj. klauzule odpowiadające dwóm członom koniunkcyjnej postaci A) takie, że $D_1 = C_1\sigma_1$ oraz $D_2 = C_2\sigma_2$ dla pewnych σ_1 oraz σ_2. Pokażemy w następnym lemacie, że istnieje wtedy rezolwenta ***w sensie rezolucji w KRP*** R dla C_1 oraz C_2 oraz podstawienie σ takie, że $R\sigma = R'$. Mówiąc nie całkiem precyzyjnie, lemat ten stwierdza, że cokolwiek może zostać wyprowadzone rezolucyjnie *w sensie KRZ* z $E(A)$, może też zostać wyprowadzone rezolucyjnie *w sensie KRP* z A.

LEMAT 25.4.8. (***Lemat o podnoszeniu***). Niech A będzie dowolną formułą języka KRP w skolemowej postaci normalnej. Jeśli $R' \in res(E(A))$, to istnieje $R \in res(A)$ taka, że $R\sigma' = R'$.

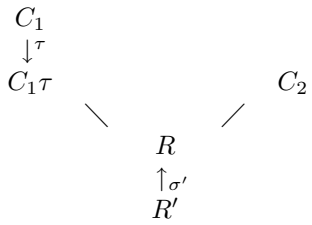
DOWÓD.

Może, przed właściwym dowodem, pożyteczna będzie pewna ilustracja (wyjaśniająca jednocześnie nazwę lematu).

Niech A będzie dowolną formułą języka KRP w skolemowej postaci normalnej. Niech C_1 i C_2 będą dwiema klauzulami z A. Niech τ będzie podstawieniem takim, że $C_1\tau$ oraz C_2 nie mają żadnych wspólnych zmiennych. Niech C_1' i C_2' będą takimi elementami $E(A)$, że $C_1\tau\sigma_1 = C_1'$ oraz $C_2\sigma_2 = C_2'$ dla pewnych podstawień σ_1 oraz σ_2. Wreszcie, niech R' będzie rezolwentą (w sensie KRZ) dla C_1' i C_2'. Sytuacje tę przedstawia diagram (kreski ukośne odpowiadają rezolucji):



Teza lematu o podnoszeniu mówi wtedy, że istnieje rezolwenta R dla $C_1\tau$ oraz C_2 (w sensie KRP) taka, że $R\sigma' = R'$, co symbolizuje poniższy diagram:



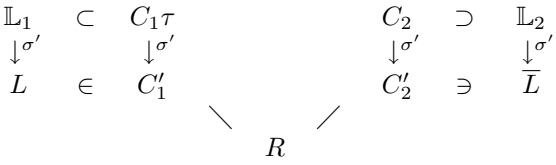
Pierwszy z powyższych diagramów dotyczy rezolucji w KRZ, drugi rezolucji w KRP.

Przejdźmy do dowodu. Przypuśćmy, że zachodzą założenia lematu (zob. pierwszy diagram). Wtedy, na mocy definicji rezolucji w KRZ, musi istnieć literał $L \in C_1'$ taki, że $\overline{L} C_2'$ oraz $R' = (C_1' - \{L\}) \cup (C_2' - \overline{L})$.

Niech $\sigma' = \sigma_1\sigma_2$. Ponieważ $C_1\tau$ oraz C_2 nie mają żadnych wspólnych zmiennych, więc $C_1\tau\sigma' = C_1\tau\sigma_1 = C_1'$ oraz $C_2\sigma' = C_2\sigma_2 = C_2'$.

Niech $\mathbb{L}_1 = \{L_1, \ldots, L_n\}$ będzie zbiorem tych wszystkich literałów L_i z $C_1\tau$, dla których $L_i\sigma' = L$. Podobnie, niech $\mathbb{L}_2 = \{L_1', \ldots, L_m'\}$ będzie zbiorem tych wszystkich literałów L_i z C_2, dla których $L_i'\sigma' = \overline{L}$.

Zilustrujmy dokonane konstrukcje diagramem:



Niech $\mathbb{L} = \mathbb{L}_1 \cup \mathbb{L}_2$. Zbiór $\mathbb{L}$ jest uzgadnialny, ponieważ $\mathbb{L}\sigma' = \{\overline{L}\}$. Niech σ będzie najbardziej ogólnym unifikatorem dla $\mathbb{L}$. Z definicji rezolucji w KRP znajdujemy rezolwentę R dla C_1 oraz C_2:

$$R = ((C_1\tau - \mathbb{L}_1) \cup (C_2 - \mathbb{L}_2))\sigma.$$

Trzeba jeszcze pokazać, że R' może zostać otrzymana z R przez podstawienie. Pokażemy, że $R\sigma' = R$. Zwróćmy uwagę, że ponieważ σ' jest unifikatorem dla $\mathbb{L}$, a σ jest najbardziej ogólnym unifikatorem dla $\mathbb{L}$, więc $\sigma\sigma' = \sigma'$. Mamy następujący ciąg równości:

$$\begin{aligned} R\sigma' &= ((C_1\tau - \mathbb{L}_1) \cup (C_2 - \mathbb{L}_2))\sigma\sigma' = \\ &= ((C_1\tau - \mathbb{L}_1) \cup (C_2 - \mathbb{L}_2))\sigma' = \\ &= (C_1\tau\sigma' - \mathbb{L}_1\sigma') \cup (C_2\sigma' - \mathbb{L}_2\sigma') = \\ &= (C_1' - \{L\}) \cup (C_2' - \{\overline{L}\}) = \\ &= R' \end{aligned}$$

Tym samym dowód lematu o podnoszeniu został zakończony. Konsekwencją tego lematu jest lemat następujący.

LEMAT 25.4.9. Niech A będzie dowolnym zdaniem języka KRP w skolemowej postaci normalnej. Jeśli $C' \in \mathcal{R}(E(A))$, to istnieje $C \in \mathcal{R}(A)$ oraz podstawienie σ' takie, że $C\sigma' = C$.

DOWÓD.

Jeśli $C' \in \mathcal{R}(E(A))$, to $C' \in res_n(E(A))$ dla pewnego n. Dowód przeprowadzimy przez indukcję po n.

Gdy $n = 0$, to $C\sigma' \in E(A)$. Wtedy, z definicji $E(A)$, $C\sigma'$ otrzymujemy przez podstawienie za zmienne termów bez zmiennych w pewnej klauzuli C z A.

W kroku indukcyjnym skorzystamy z lematu o podnoszeniu. Przypuśćmy, że dla pewnego m każda klauzula w $res_m(E(A))$ otrzymana jest z pewnej klauzuli z $\mathcal{R}(A)$ poprzez podstawienie. Niech $\widetilde{A} \subset \mathcal{R}(A)$ będzie taka, że każda klauzula z $res_m(E(A))$ otrzymana jest z pewnej klauzuli z $\widetilde{A}$. Wtedy $res_m(E(A)) \subset E(\widetilde{A})$. Jeśli $C' \in res_{m+1}(E(A))$, to $C' \in res(E(\widetilde{A}))$. Na mocy lematu o podnoszeniu, istnieje $C \in res(\widetilde{A})$ taka, że $C\sigma' = C'$ dla pewnego podstawienia σ'. Ponieważ $\widetilde{A} \subset \mathcal{R}(A)$, więc $C \in \mathcal{R}(A)$. To kończy dowód.

Powyższe lematy są potrzebne do udowodnienia najważniejszego wyniku w tym podrozdziale, a mianowicie twierdzenia o pełności rezolucji.

TWIERDZENIE 25.4.10. (***Twierdzenie o pełności rezolucji w KRP***). Niech A będzie dowolnym zdaniem języka KRP w skolemowej postaci normalnej. Jeśli A jest niespełnialna, to $\Box \in \mathcal{R}(A)$.

DOWÓD.

Na mocy lematu 25.4.9., jeśli $\Box \in \mathcal{R}(E(A))$, to istnieje $C \in \mathcal{R}(A)$ taka, że $C\sigma' = \Box$ dla pewnego podstawienia σ'. To jednak jest możliwe jedynie wtedy, gdy $C = \Box$. Tak więc, jeśli $\Box \in \mathcal{R}(E(A))$, to $\Box \in \mathcal{R}(A)$. Wnioskujemy stąd, że jeśli A jest niespełnialna, to $\Box \in \mathcal{R}(A)$.

Uwaga. Czasem przez twierdzenie o pełności rezolucji w KRP rozumie się łącznie twierdzenia 25.4.7. oraz 25.4.10.

* * *

Jest wiele różnych, bardziej subtelnych od powyższego — całkowicie ogólnego — rodzajów rezolucji. Problematyka ta jest intensywnie badana, przede wszystkim w związku z zastosowaniami metody rezolucji w automatycznym dowodzeniu twierdzeń.

Uwagi końcowe

Skrypt nie jest pomyślany jako standardowy podręcznik logiki. Ma być jedynie prezentacją określonej metody dowodowej — metody drzew semantycznych.

* * *

Jak wiadomo, LOGIKA zajmuje się pojęciami: DOWODU oraz WYNIKANIA LOGICZNEGO. Związki między tymi pojęciami ustalają twierdzenia metalogiczne: TWIERDZENIE O TRAFNOŚCI oraz TWIERDZENIE O PEŁNOŚCI. Pewne TWIERDZENIA LIMITACYJNE określają samoograniczenia stosowalności metod KRP. Tak więc, KLASYCZNY RACHUNEK LOGICZNY jest dyscypliną o dobrze rozwiniętej metodologii. Należy jednak pamiętać, że LOGIKA nie jest dyscypliną zamkniętą. Nowe inspiracje dla niej znajdujemy na kilku obszarach. Wymieńmy trzy z nich:

- PRAKTYKA BADAWCZA MATEMATYKI. Pojęcie DOWODU rozważane w logice jest tylko idealizacją (normatywną względem *Przeszłości*?) praktyki matematycznej. Przy tym, to owa praktyka matematyków jest fundamentalna dla tworzenia pojęć logicznych (a nie na odwrót). Tak więc, rozwój logiki może być inspirowany przez badania matematyczne. W ten sposób powstały np. logiki wyższych rzędów, logiki infinitarne, logiki z uogólnionymi kwantyfikatorami, logiki intuicjonistyczne, itd.
- TEORETYCZNE PODSTAWY INFORMATYKI. Z oczywistych powodów badania informatyczne muszą być wspomagane badaniami logicznymi. Źródeł informatyki teoretycznej poszukiwać należy przecież w rozważaniach logicznych. Rozwój informatyki inspiruje z kolei nowe badania logiczne. W ten sposób powstały np. logiki algorytmiczne, nowe interpretacje dla logik modalnych, itd.
- PRAGMATYKA LOGICZNA. Pojęcie NIEZAWODNEJ REGUŁY WNIOSKOWANIA zostało wyabstrahowane (w cywilizacji Zachodu) z rozumowań przeprowadzanych w językach naturalnych. W tej postaci, w jakiej stosowane jest ono obecnie (odwołującej się do czysto FORMALNYCH, składniowych, własności komunikatów oraz do znaczenia ustalonego zestawu STAŁYCH LOGICZNYCH) jest ono adekwatne do opisu tworzenia i przekształcania WIEDZY w aparaturze pojęciowej poszczególnych nauk. Jest problemem otwartym, czy ***obecnie*** znane systemy logiczne potrafią trafnie reprezentować wszelkie rozumowania przeprowadzane w języku naturalnym, którym chcielibyśmy nadać walor — jakoś pragmatycznie rozumianej — prawomocności. Stąd kolejna inspiracja dla badań logicznych.

* * *

Przypomnijmy, że jednym ze skromnych celów niniejszego skryptu jest to, aby uświadomić ewentualnym czytelniczkom różnice między:

- ***Wynikaniem logicznym*** a ***uzasadnianiem*** oraz ***uznawaniem*** zdań. Pierwsze z tych pojęć ma, w dzisiejszym rozumieniu, charakter obiektywny; drugie i trzecie mogą odwoływać się do różnych czynników, także natury pragmatycznej. Uzasadnianie może mieć postać precyzyjnego dowodu, ale może też odwoływać się do zabawnych reguł LOGIKI UZNANIOWEJ.
- ***Dowodzeniem*** a procedurami czysto ***algorytmicznymi***. Pierwsza z tych aktywności ma charakter ***twórczy***, drugie są działaniami wedle określonego przepisu.

Jeśli lektura skryptu nie przeszkodzi w osiągnięciu tych celów, to pozwolimy sobie uznać, że praca nad nim miała SENS.

Wykorzystywana literatura

Baader, F., Snyder, W. 2001. Unification theory. W: *Handbook of Automated Reasoning.*, 446–533.

Bachmair, L., Ganzinger, H. 2001. Resolution theorem proving. W: *Handbook of Automated Reasoning.*, 19–99.

Bartley, W.W., III. 1977. *Lewis Carroll's Symbolic Logic.* Clarkson N. Potter, New York.

Ben-Ari, M. 2005. *Logika matematyczna w informatyce.* Wydawnictwa Naukowo Techniczne.

Fitting, M. 1990. *First-Order Logic and Automated Theorem Proving.* Springer Verlag, New York Berlin Heidelberg London Paris Tokyo Hong Kong.

Handbook of Automated Reasoning. 2001. A. Robinson, A. Voronkov (eds.), Elsevier, Amsterdam London New York Oxford Paris Shannon Tokyo, The MIT Press, Cambridge, Massachusetts.

Handbook of Tableau Methods. 1999. Edited by: D'Agostino, M., Gabbay, D.M., Hähnle, R., Posegga, J., Kluwer Academic Publishers, Dordrecht Boston London.

Hedman, S. 2004. *A first course in logic.* Oxford University Press.

Letz, R. 1990. First-order tableau methods. W: *Handbook of Tableau Methods*, 125–196.

Marciszewski, W., Murawski, R. 1995. *Mechanization of Reasoning in a Historical Perspective.* Rodopi, Amsterdam – Atlanta.

Nerode, A., Shore, R.A. 1997. *Logic for applications.* Springer.

* * *

JERZY POGONOWSKI
Zakład Logiki Stosowanej UAM
www.logic.amu.edu.pl