Wykład 2.

2.1 Programy w logice

Atomem nazywamy dowolną formułę atomową. Zauważmy, że **true** oraz **false** nie są atomami. Atomy i ich negacje nazywamy literałami. Tak więc literały mogą być negatywne (zanegowane) lub pozytywne (niezanegowane). Klauzulami są dowolne formuły postaci

$$\forall x_1 \ \ldots \ \forall x_n \ (L_1 \vee \ldots \vee L_m), \tag{1}$$

gdzie $L_1, \ldots, L_m$ są literałami, a $\{x_1, \ldots, x_n\}$ jest zbiorem zmiennych występujących w $L_1, \ldots, L_m$.

Rozbijmy literały $L_1, \ldots, L_m$ z klauzuli (1) na dwie części: $A_1, \ldots, A_k$ i $\neg B_1, \ldots, \neg B_l$, gdzie $A_1, \ldots, A_k$ oraz $B_1, \ldots, B_l$ są atomami. Klauzulę (1) będziemy zapisywać w postaci

$$A_1, \ldots, A_k \leftarrow B_1, \ldots, B_l.$$

Niezanegowane atomy w klauzuli będziemy nazywać wnioskami, zanegowane – przesłankami. Klauzule z jednym wnioskiem nazywamy klauzulami programowymi. Klauzula programowa bez przesłanek nazywa się klauzulą jednostkową lub faktem. Takie klauzule mają postać $A \leftarrow$. Klauzula z pustym zbiorem przesłanek nazywa się klauzulą negatywną lub celem.

Jeżeli klauzula zarówno ma pusty zbiór przesłanek jak i wniosków, to określamy ją jako pustą i oznaczamy symbolem $\Box$. Symbol $\Box$ można uważać za alternatywę nie zawierającą członów. Przyjmujemy, że taka alternatywa jest fałszywa przy każdym wartościowaniu. Interpretujemy ją jako sprzeczność.

Programem w logice nazywamy dowolny skończony zbiór klauzul programowych. Jest to podklasa ogólnych programów w logice. Ogólne programy w logice to skończone zbiory klauzul.

2.2 Rezolucja

Niech P będzie programem w logice i niech $N = \leftarrow A_1, \ldots, A_n$ będzie negatywną klauzulą. Rozważmy klauzulę $C = A \leftarrow B_1, \ldots, B_k \in P$. Przypuśćmy, że dla pewnego i atomy A_i oraz A można zunifikować i θ jest najogólniejszym unifikatorem tej pary. Wtedy klauzulę

$$N' = \leftarrow (A_1, \ldots, A_{i-1}, B_1, \ldots, B_k, A_{i+1}, \ldots, A_n)\theta$$

nazywamy rezolwentą klauzul N i C.

Definicję rezolwenty można uważać za sposób tworzenia pewnej klauzuli wykorzystujący dane klauzule. Ten sposób tworzenia nowej klauzuli nazywamy regułą rezolucji.

Regułę rezolucji można uważać za regułę dowodzenia. Korzystając z zapisu charakterystycznego dla logiki możemy ją przedstawić w postaci

$$\frac{\leftarrow A_1, \ldots, A_n, \quad A \leftarrow B_1, \ldots, B_k}{\leftarrow (A_1, \ldots, A_{i-1}, B_1, \ldots, B_k, A_{i+1}, \ldots, A_n)\theta},$$

gdzie θ jest tak określone, jak to zostało podane wyżej. Jeżeli wrócimy do symboliki logicznej, to reguła ta przyjmie postać

$$\frac{\neg A_1 \vee \ldots \vee \neg A_n, \quad A \vee \neg B_1 \vee \ldots \vee \neg B_k}{(\neg A_1 \vee \ldots \vee \neg A_{i-1} \vee \neg B_1 \vee \ldots \vee \neg B_k \vee \neg A_{i+1} \vee \ldots \vee \neg A_n)\theta}.$$

Stosowanie reguły rezolucji składa się z 4 kroków:

1) wybieramy atom A_i,
2) próbujemy zunifikować klauzule A oraz A_i, a jeżeli jest to możliwe, to znajdujemy najogólniejszy unifikator θ tych klauzul,
3) jeżeli klauzule A oraz A_i dają się zunifikować, to zastępujemy w klauzuli N atom A_i klauzulami $B_1, \ldots, B_k$,
4) a następnie w otrzymanej w ten sposób klauzuli wykonujemy podstawienie θ.

Klauzulę C' nazywamy wariantem klauzuli C, jeżeli $C' = C\theta$ dla pewnego podstawienia θ przyporządkowującego wzajemnie jednoznacznie zmiennym występującym w C (inne) zmienne.

Niech P będzie programem (w logice). Ciąg klauzul $N_0, N_1, \ldots$, nazywamy *SLD*-wyprowadzeniem, jeżeli istnieją ciągi $C_0, C_1, \ldots$ klauzul i $\theta_0, \theta_1, \ldots$ podstawień takie, że dla wszystkich $i = 0, 1, \ldots$

1) N_{i+1} jest rezolwentą klauzul N_i oraz C_i,
2) θ_i jest najogólniejszym unifikatorem znalezionym w 2) kroku reguły rezolucji, która pozwoliła wywnioskować klauzulę N_{i+1} z klauzul N_i oraz C_i,
3) klauzula C_i jest wariantem pewnej klauzuli z P (zawierającym wyłącznie zmienne nie występujące w N_i).

O wyprowadzeniu $N_0, N_1, \ldots$, mówimy, że jest wyprowadzeniem ze zbioru P.

Rezolucja w wersji opisanej wyżej jest określana jako *SLD*-rezolucja (Linear resolution with Selection rule for Definite clauses).

2.3 Ogólna reguła rezolucji

W logice jest wykorzystywana ogólniejsza reguła rezolucji

$$\frac{C_1, \ldots, C_m \leftarrow A_1, \ldots, A_n, \qquad A'_1, \ldots, A'_{n'} \leftarrow B_1, \ldots, B_k}{(C_1, \ldots, C_m, A'_1, \ldots, A'_{j-1}, A'_{j+1}, \ldots, A'_{n'} \leftarrow A_1, \ldots, A_{i-1}, B_1, \ldots, B_k, A_{i+1}, \ldots, A_n)\theta},$$

gdzie θ jest najogólniejszym unifikatorem atomów A_i oraz A'_j. Jeżeli zapiszemy ją w zwyczajnej symbolice logicznej, to przyjmie ona postać

$$\frac{C_1 \vee \ldots \vee C_m \vee \neg A_1 \vee \ldots \vee \neg A_n, \qquad A'_1 \vee \ldots \vee A'_{n'} \vee \neg B_1 \vee \ldots \vee \neg B_k}{\begin{array}{l}(C_1 \vee \ldots \vee C_m \vee A'_1 \vee \ldots A'_{j-1} \vee A'_{j+1} \vee \ldots \vee A'_{n'} \vee \\ \qquad \vee \neg A_1 \vee \ldots \vee \neg A_{i-1} \vee \neg B_1 \vee \ldots \vee \neg B_k \vee \neg A_{i+1} \vee \ldots \vee \neg A_n)\theta\end{array}}$$

lub w skrócie

$$\frac{\varphi \vee \neg A_i \quad A'_j \vee \psi}{(\varphi \vee \psi)\theta}.$$

2.4 Poprawność reguły rezolucji

Lemat 2.1 *Jeżeli φ jest formułą bez kwantyfikatorów, θ – podstawieniem, a h – dowolnym wartościowaniem w strukturze $\mathcal{A}$, i jeżeli h' jest wartościowaniem takim, że $h'(x) = \theta(x)[h]$ dla zmiennych x należących do dziedziny θ oraz $h'(x) = h(x)$ dla pozostałych zmiennych, to*

$$\mathcal{A} \models \varphi\theta[h] \Leftrightarrow \mathcal{A} \models \varphi[h'].$$

Dowód. Lemat ten można dowieść przez indukcję ze względu na budowę formuły. Dowód powinien zostać poprzedzony sprawdzeniem analogicznej własności dla termów, a więc równości $t\theta[h] = t[h']$, zachodzącej dla dowolnego termu t. □

Lemat 2.2 *Jeżeli Φ_0 jest rezolwentą formuł Φ i Ψ oraz w strukturze $\mathcal{A}$ jest spełniona formuła Ψ (a więc $\mathcal{A} \models \Psi$), to*

$$\mathcal{A} \models \Phi\theta \to \Phi_0.$$

Dowód. Przypuśćmy, że $\Phi = \varphi \vee \neg A$, $\Psi = A' \vee \psi$. Jeżeli stosując regułę rezolucji (także ogólną) do formuł Φ i Ψ unifikujemy atomy A i A', to rezolwentą jest formuła $\Phi_0 = (\varphi \vee \psi)\theta$, gdzie θ jest najbardziej ogólnym unifikatorem atomów A i A'.

Weźmy dowolne wartościowanie h w strukturze $\mathcal{A}$ i załóżmy, że

$$\mathcal{A} \models \Phi\theta[h]. \tag{2}$$

Aby dowieść lemat wystarczy wykazać, że

$$\mathcal{A} \models \Phi_0[h].$$

Niech h' oznacza wartościowanie, które spełnia równości $h'(x) = \theta(x)[h]$ dla zmiennych x należących do dziedziny θ oraz $h'(x) = h(x)$ dla pozostałych zmiennych.

Z założenia, że formuła Ψ jest spełniona w strukturze $\mathcal{A}$ otrzymujemy, że

$$\mathcal{A} \models \Psi[h'], \tag{3}$$

a stąd i z lematu 2.1 wynika, że

$$\mathcal{A} \models \Psi\theta[h].$$

Jeżeli wykonamy podstawienie θ w formułach Φ i Ψ i skorzystajmy z tego, że θ jest unifikatorem A i A', to własności (2) i (3) przyjmą postać

$$\mathcal{A} \models (\varphi\theta \vee \neg A\theta)[h] \text{ oraz } \mathcal{A} \models (A\theta \vee \psi\theta)[h].$$

Są możliwe dwa przypadki: albo formuła $A\theta$ jest spełniona w strukturze $\mathcal{A}$ przy wartościowaniu h, albo nie. Bez względu na to, który z tych przypadków zachodzi, przy wartościowaniu h w tej strukturze spełniony jest jeden z członów alternatywy $\varphi\theta \vee \psi\theta = \Phi_0$. Wobec tego spełniona jest też alternatywa, czyli formuła Φ_0. Tak więc

$$\mathcal{A} \models \Phi_0[h].$$

□

Twierdzenie 2.3 *Przyjmijmy, że P jest programem w logice, $N = \leftarrow A_1, \ldots, A_k$ jest celem oraz w strukturze $\mathcal{A}$ są spełnione wszystkie formuły ze zbioru P i formuła N. Jeżeli istnieje wyprowadzenie z programu P klauzuli pustej $\Box$, rozpoczynające się klauzulą N, a $\theta_1, \ldots, \theta_n$ jest ciągiem podstawień z tego wyprowadzenia, to*

$$\mathcal{A} \models (A_1 \wedge \ldots \wedge A_k)\theta_1 \ldots \theta_n.$$

Dowód. Niech $N_0, \ldots, N_n$ będzie wyprowadzeniem z programu P klauzuli pustej $\Box$, rozpoczynającym się klauzulą N, z ciągiem podstawień $\theta_1, \ldots, \theta_n$. Pokazujemy przez indukcję, że dla $m = 0, \ldots, n$

$$\mathcal{A} \models \neg(A_1 \wedge \ldots \wedge A_k)\theta_1 \ldots \theta_m \to N_m. \tag{4}$$

Własność dowodzona w pierwszym kroku indukcyjnym, a więc

$$\mathcal{A} \models \neg(A_1 \wedge \ldots \wedge A_k) \to N_0.$$

jest konsekwencją prawa de Morgana.

Aby sprawdzić drugi krok indukcyjny wystarczy powołać się na dwa fakty. Po pierwsze, z założenia indukcyjnego (a więc z wzoru (4)) oraz z lematu 2.1 otrzymujemy, że

$$\mathcal{A} \models \neg(A_1 \wedge \ldots \wedge A_k)\theta_1 \ldots \theta_m\theta_{m+1} \to N_m\theta_{m+1}.$$

Drugim potrzebnym faktem jest własność

$$\mathcal{A} \models N_m\theta_{m+1} \to N_{m+1}$$

wynikająca z poprzedniego lematu.

Ponieważ rozważane wyprowadzenie zawiera klauzulę pustą (czyli $N_n = \Box$), więc otrzymujemy, że

$$\mathcal{A} \models \neg(A_1 \wedge \ldots \wedge A_k)\theta_1 \ldots \theta_n \to \Box.$$

Stąd i np. z prawa kontrapozycji wynika, że

$$\mathcal{A} \models (A_1 \wedge \ldots \wedge A_k)\theta_1 \ldots \theta_n.$$

$\Box$