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 wykładzie będą wykorzystane w wykładzie 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 $\{E\sigma : \quad E \in S\}$.

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.

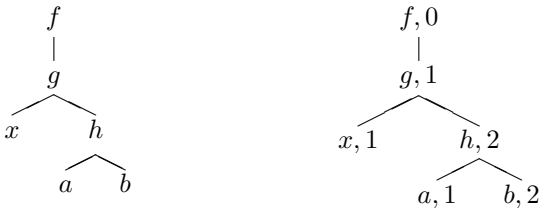
24.2.5. Zbiór $\{P(x, y), P(x, f(y))\}$ nie jest uzgadnialny. Niezależnie od tego, co podstawimy za zmienne x oraz y, w drugim literale jest jedno więcej wystąpienie symbolu f niż w pierwszym.

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.

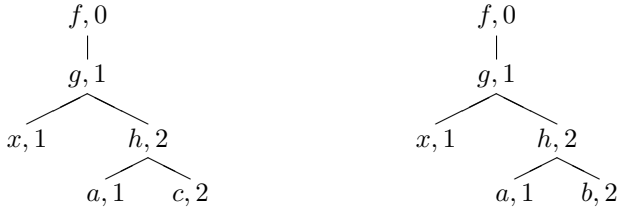
Niech S będzie skończonym niepustym zbiorem wyrażeń. ***Zbiorem niezgodności*** (niektórzy używają terminu: ***para niezgodności***) dla S nazywamy każdy dwuelementowy zbiór wyrażeń $\{E_1, E_2\}$ taki, że symbole (funktory) główne w E_1 i E_2 są różne oraz E_1 i E_2 występują na tych samych pozycjach 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:

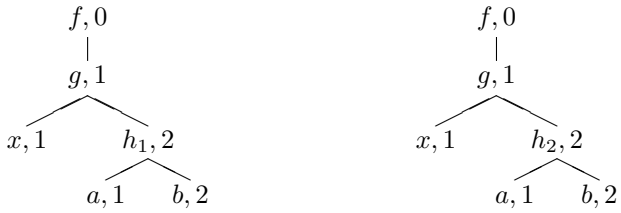


Lewe drzewo to po prostu drzewo syntaktyczne termu $f(g(x, h(a, b)))$. W drzewie prawym zaznaczono pozycje poszczególnych argumentów. Poniższe rysunki pokazują zbiory niezgodności dla termów reprezentowanych przez drzewa:



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:

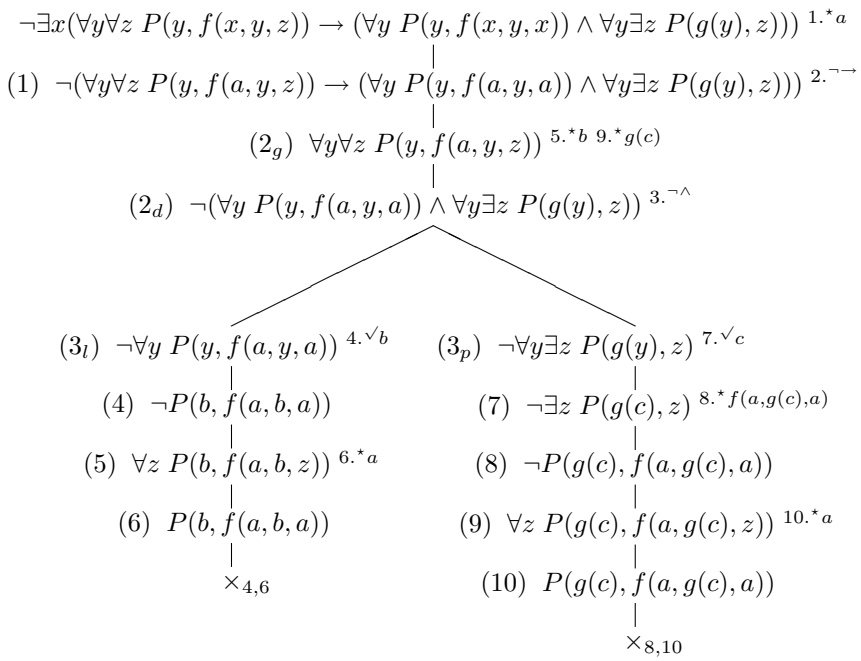
- unifikacja tzw. *dag*-ów termów (dag jest drzewową reprezentacją budowy termu),
- unifikacja prawie liniowa (wykorzystująca pewne relacje równoważności na termach),
- różne rodzaje E-unifikacji, tj. unifikacji wykorzystującej zbiory identyczności termów.

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:

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

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

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

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:

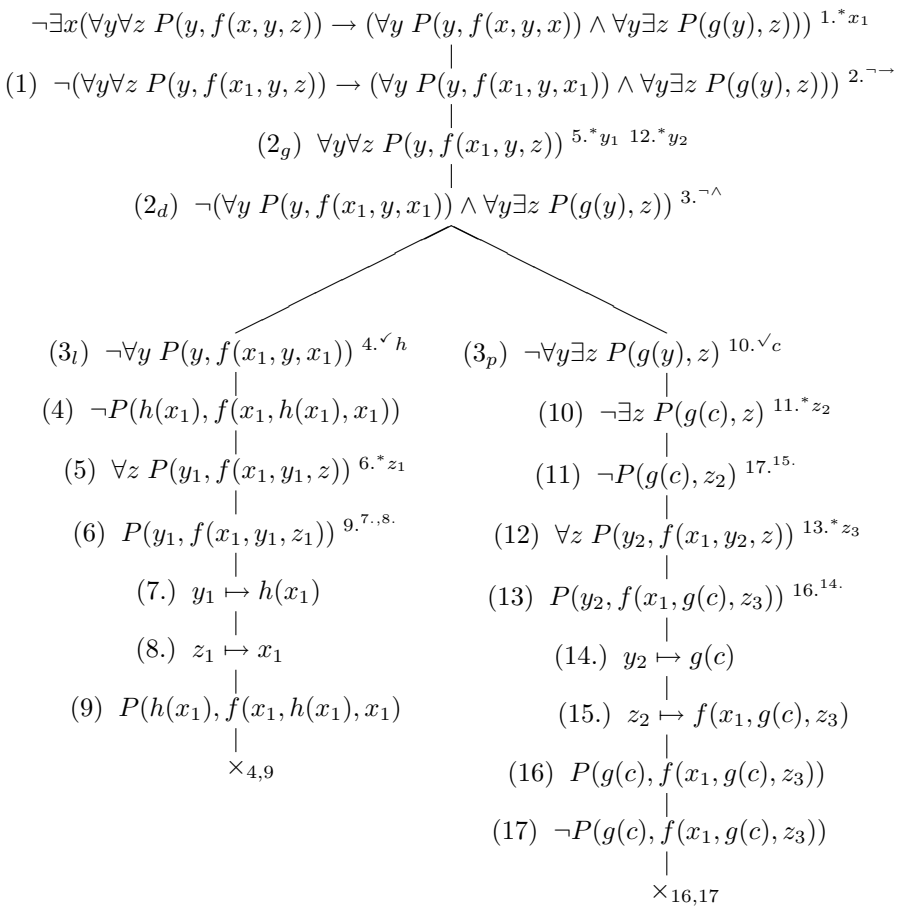
- wprowadzenie nowych zmiennych wolnych,
- podstawienia odpowiednich termów za zmienne wolne.

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 tych wykładów 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):

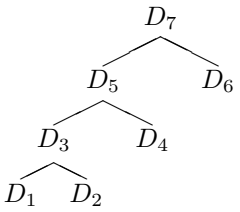
- $C_1 = \{\neg P(x,y), \neg P(y,z), P(x,z)\}$
- $C_2 = \{\neg P(u,v), P(v,u)\}$
- $C_3 = \{\neg P(x,y), \neg P(z,y), P(x,z)\}$.

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:

$$\dfrac{\dfrac{\dfrac{}{D_1} \quad \dfrac{}{D_2}}{D_3} \quad \dfrac{}{D_4}}{D_5} \quad \dfrac{}{D_6}$$
$$D_7$$

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.

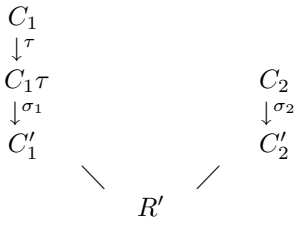
Przypuśćmy, że D_1 oraz D_2 są elementami $E(A)$ i że R' jest rezolwentą D_1 oraz D_2 ***w sensie rezolucji w KRZ***. Istnieją wtedy klauzule C_1 oraz C_2 otrzymane z A (tj. klauzule odpowiadające dwóm członom koniunkcyjnej postaci A) takie, że $D_1 = C_1\sigma_1$ oraz $D_2 = C_2\sigma_2$ dla pewnych 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 $R\sigma' = R'$.

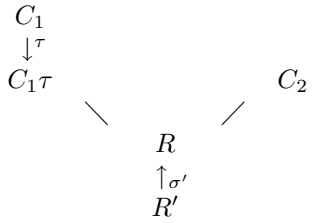
DOWÓD.

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:

$$\begin{array}{ccccccc} \mathbb{L}_1 & \subset & C_1\tau & & & C_2 & \supset & \mathbb{L}_2 \\ \downarrow^{\sigma'} & & \downarrow^{\sigma'} & & & \downarrow^{\sigma'} & & \downarrow^{\sigma'} \\ L & \in & C_1' & & & C_2' & \ni & \overline{L} \end{array}$$

$$\diagdown \quad \diagup$$

$$R$$

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:

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

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

Wykłady 26–30 będą poświęcone w roku akademickim 2007–2008 powtórce całego materiału oraz przygotowaniu do egzaminu. W następnym roku akademickim, za przyzwoleniem Losu, na wykładach 26–30 omawiane będą: rachunki sekwentów, matematyczne reprezentacje pojęcia obliczalności, elementy metalogiki, teoria modeli oraz logiki modalne.

Powyższe notatki nie są pomyślane jako standardowy podręcznik logiki. Mają być jedynie prezentacją kilku metod dowodowych. Notatki te są obecnie poprawiane i rozszerzane. Celem jest przygotowanie, w miarę nowoczesnego podręcznika logiki, na który mają złożyć się m.in. teksty wykładów, zamieszczane na stronach internetowych Zakładu Logiki Stosowanej UAM.

* * *

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 etnicznych. 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 językach etnicznych, którym chcielibyśmy nadać walor — jakoś pragmatycznie rozumianej — prawomocności. Stąd kolejna inspiracja dla badań logicznych.

* * *

Przypomnijmy, że jednym ze skromnych celów niniejszych notatek 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 niniejszych notatek nie przeszkodzi w osiągnięciu tych celów, to pozwolimy sobie uznać, że nasza praca miała SENS.

Literatura wykorzystywana w wykładach 24–25

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