III.7. 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 drzew semantycznych. Nadto, pojęcie *unifikacji* jest samo w sobie interesujące. W ostatnich kilkudziesięciu latach nabrało istotnego znaczenia, np. w automatycznym dowodzeniu twierdzeń.

III.7.1. 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 $\sigma\sigma = \sigma$. Mozna 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 $\theta = \sigma\lambda$.

Przykłady. III.7.1.

III.7.1.1. 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.

III.7.1.2. 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.

III.7.1.3. 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.

III.7.1.4. 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:

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

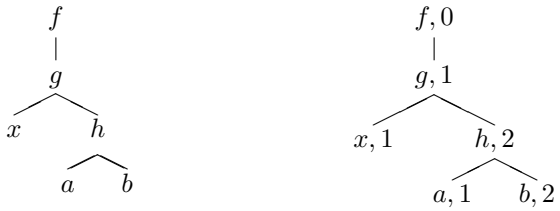
W dalszym ciągu mamy więcej niż jeden literał. Każdy literał rozpoczyna się teraz od ciągu symboli $R(f(g(a), a,$. W pierwszym literale mamy dalej term a, natomiast w drugim term b. Żaden z tych termów nie jest zmienną, a więc nie ma podstawienia, które uzgadniałoby te termy. W konsekwencji, wyjściowy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

III.7.1.5. 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 III.7.1.3. powyżej rozważaliśmy zbiory $S_1 = \{f(x, g(x)), f(h(y), g(h(z)))\}$ oraz $S_2 = \{f(h(x), g(x)), f(g(x), h(x))\}$. 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 $D(S)$.

III.7.2. 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.

7.2.1. 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 7.2.1. 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.

7.2.2. 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.

7.2.3. 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.

7.2.4. 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:

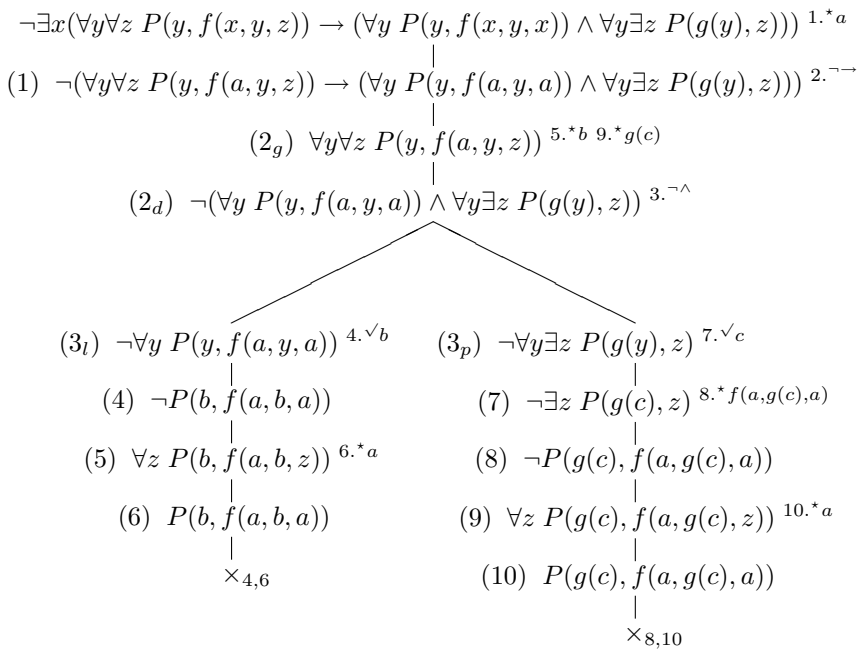
- unifikacja tzw. *dag*-ów termów (dag jest drzewową reprezentacją budowy termu),
- unifikacja prawie liniowa (wykorzystująca pewne relacje równoważności na termach),
- różne rodzaje E-unifikacji, tj. unifikacji wykorzystującej zbiory identyczności termów.

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.

III.7.3. Drzewa semantyczne ze zmiennymi wolnymi

Pamiętamy, że reguły $R(\forall)$ oraz $R(\neg\exists)$ powinny być zastosowane dla ***każdego*** termu (bez zmiennych), występującego na rozważanej gałęzi drzewa semantycznego. 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 w różnych podrozdziałach rozdziału III. Dowody przeprowadzane przy użyciu drzew semantycznych niekoniecznie są czymś w rodzaju procedury algorytmicznej — czasem pomysłowe dobranie kolejnych kroków dowodowych znacznie upraszcza pracę. Problematyka ta jest również istotna w zastosowaniach drzew semantycznych w *automatycznym dowodzeniu twierdzeń*. Nawet dla najszybszego komputera może być kłopotliwe wykonywanie ***wszystkich*** możliwych w danym momencie kroków. Jednym z przydatnych rozwiązań są tzw. *drzewa semantyczne 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.

III.7.3.1. Definicje

Przypominamy, że w podrozdziale III.6.3. omówiono pojęcie ***skolemizacji***. Będzie ono potrzebne poniżej.

Podstawowa idea wprowadzania drzew semantycznych 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 drzewa jest ***zamknięta***, gdy występuje na niej para formuł wzajem sprzecznych: A oraz $\neg A$.

Drzewo jest ***zamknięte***, jeśli wszystkie jego gałęzie są zamknięte.

Rozważmy proste przykłady.

III.7.3.3. Przykłady

Przykład III.7.3.3.1. (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 drzewo semantyczne jej negacji:

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

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

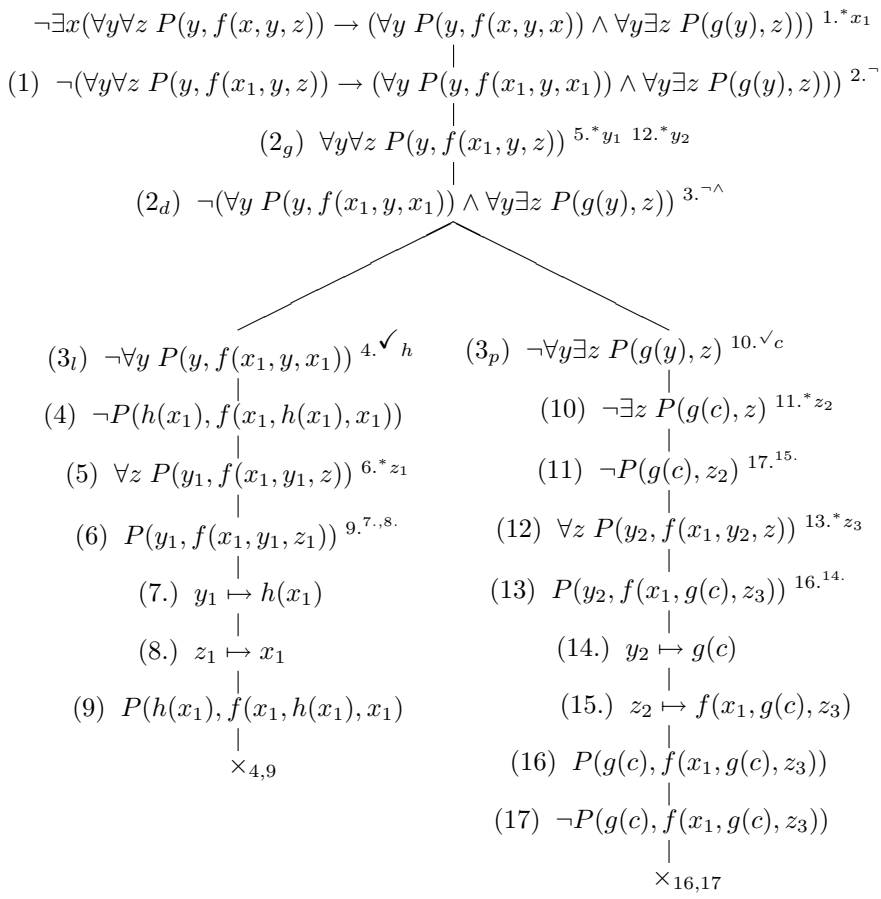
- wprowadzenie nowych zmiennych wolnych,
- podstawienia odpowiednich termów za zmienne wolne.

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 III.7.3.3.2. (Letz 1999, 168).

Wrócimy teraz do przykładu rozważanego na początku III.7.3.



Mogłoby się wydawać, że wprowadzenie zmiennych wolnych do drzew semantycznych 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 drzewa semantyczne ze zmiennymi wolnymi są istotnie bardziej przydatne od „zwykłych" drzew semantycznych.

* * *

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

Odnośniki bibliograficzne do podrozdziału III.7.

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

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.

Jeffrey, R. 1991. *Formal Logic: Its Scope and Limits.* McGraw-Hill, New York.

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

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

* * *

Jerzy Pogonowski
Zakład Logiki Stosowanej UAM
www.logic.amu.edu.pl