METODY DOWODZENIA TWIERDZEŃ I AUTOMATYZACJA ROZUMOWAŃ

WYKŁAD 9: UNIFIKACJA

III rok kognitywistyki UAM, 2016–2017

1 Wstęp

Ten wykład stanowi uzupełnienie do wykładów dotyczących tablic analitycznych oraz metody rezolucji. Jak pamiętamy, w metodzie tablic analitycznych pewien kłopot sprawiały γ-formuły: reguła ich dotycząca nakazywała zastąpić każdą γ-formułę przez wszystkie jej instancje, z użyciem dowolnych termów domkniętych. Do zamykania gałęzi tablicy analitycznej wystarcza jednak znalezienie pary formuł wzajem sprzecznych, a więc w odniesieniu do γ-formuł oznacza to znalezienie takich ich instancji, które wystarczają do zamknięcia gałęzi. Proponowanym rozwiązaniem jest rozważanie tablic analitycznych ze zmiennymi wolnymi, a następnie znajdowanie takich podstawień (termów domkniętych za zmienne), aby możliwe było zamykanie gałęzi.

Metoda rezolucji w KRZ była dosyć banalna. Jest ona o wiele ciekawsza w logice pierwszego rzędu, wymaga jednak skorzystania z procedur *unifikacji* (*uzgadniania*). Dysponujemy różnymi algorytmami unifikacji dla logiki klasycznej. Unifikacja oraz rezolucja są fundamentalnymi pojęciami, wykorzystywanymi w automatycznym dowodzeniu twierdzeń w językach pierwszego rzędu.

Plan na dziś:

1. Opiszemy procedurę *unifikacji*.
2. Poznamy regułę rezolucji dla logiki pierwszego rzędu.
3. Podamy reguły dla metody tablic analitycznych ze zmiennymi wolnymi.
4. Sformułujemy kilka twierdzeń metalogicznych dotyczących unifikacji i rezolucji w logice pierwszego rzędu.
5. Pobawimy się przykładami dowodów rezolucyjnych w logice pierwszego rzędu.

2 Podstawienia

1. *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.

2. Zapis: $\{x_1 \mapsto t_1, \ldots, x_n \mapsto t_n\}$ oznacza podstawienie termów t_i za zmienne x_i $(1 \leqslant i \leqslant n)$.

Jeśli σ jest podstawieniem, a E wyrażeniem (termem lub literałem), to $E\sigma$ definiujemy przez indukcję:

1. $E\sigma = x\sigma$, gdy E jest zmienną x,

2. $E\sigma = f(t_1\sigma, \ldots, t_n\sigma)$, gdy E jest termem złożonym $f(t_1, \ldots, t_n)$,

3. $E\sigma = R(t_1\sigma, \ldots, t_m\sigma)$, gdy E jest literałem pozytywnym $R(t_1, \ldots, t_m)$,

4. $E\sigma = \neg R(t_1\sigma, \ldots, t_m\sigma)$, gdy E jest literałem negatywnym $\neg R(t_1, \ldots, t_m)$.

Dziedziną podstawienia σ jest zbiór: $dm(\sigma) = \{x \ : \ x\sigma \neq x\}$, a *przeciwdziedziną* (*zbiorem wartości*) podstawienia σ jest zbiór: $rg(\sigma) = \bigcup\limits_{x \in dm(\sigma)} \{x\sigma\}$.

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 $(E\sigma)\theta$.

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

   $$\{x_i \mapsto t_i\sigma \ : \ x_i \in dm(\theta) \wedge x_i \neq t_i\sigma\} \cup \{y_j \mapsto s_j \ : \ y_j \in dm(\sigma) - dm(\theta)\}$$

   (czyli stosujemy σ do termów t_i otrzymanych w θ [wykluczając $x_i \mapsto x_i$], a potem dodajemy z σ podstawienia dla tych zmiennych, którym θ nie nadaje wartości).

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

Operacja złożenia jest łączna, ale nie jest przemienna.

Rozważmy przykładowe obliczenie złożenia podstawień. Rozważmy mianowicie literał E: $P(x, y, w, u)$ oraz podstawienia (tu a, b, c stałe):

1. $\theta = \{x \mapsto f(y), y \mapsto g(z), w \mapsto v\}$
2. $\sigma = \{x \mapsto a, y \mapsto b, z \mapsto f(y), v \mapsto w, u \mapsto c\}$
3. $E\theta = P(f(y), g(z), v, u)$
4. $(E\theta)\sigma = P(f(b), g(f(y)), w, c)$

Obliczamy złożenie $\theta\sigma$:

1. x zamieniamy na $f(y)$, potem y zamieniamy na b, czyli $x \mapsto f(b)$
2. y zamieniamy na $g(z)$, potem z zamieniamy na $f(y)$, czyli $y \mapsto g(f(y))$
3. w zamieniamy na v, potem v zamieniamy na w, czyli opuszczamy $w \mapsto w$
4. w $E\theta$ nie ma x, więc $x \mapsto a$ nie działa
5. θ nie zmienia u, więc zostaje $u \mapsto c$

Tak więc: $\theta\sigma = \{x \mapsto f(b), y \mapsto g(f(y)), u \mapsto c, z \mapsto f(y), v \mapsto w\}$

3 Unifikacja

3.1 Definicje

1. 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.

2. 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 $\theta\lambda = \sigma$.

3. Podstawienie λ *przemianowuje zmienne*, jeśli $dm(\lambda) = rg(\lambda)$. Dla przykładu:

   (a) podstawienie $\{x \mapsto y, y \mapsto z, z \mapsto x\}$ przemianowuje zmienne,

   (b) 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 $\theta = \sigma\lambda$.

3.2 Przykłady

Przykład 1. Czy zbiór $\{\ f(a, x),\ f(y, b)\ \}$ jest uzgadnialny, tj. czy można dokonać takiego podstawienia zmiennych, aby otrzymać z tych obu termów jeden i ten sam term? Tak, wystarczy dokonać podstawienia σ:

1. $x \mapsto b$
2. $y \mapsto a$.

Wtedy $f(a, x)\sigma = f(a, b)$ oraz $f(y, b)\sigma = f(a, b)$.

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

Przykłady 2 i 3. 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.:

1. $\theta = \{x \mapsto a, y \mapsto w\}$,
2. $\sigma = \{x \mapsto a, y \mapsto a, w \mapsto a\}$,
3. $\psi = \{x \mapsto a, y \mapsto b, w \mapsto b\}$.

W tym przypadku θ jest mgu dla S_2.

Przykład 4. Niech:

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

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.

1. pierwszymi argumentami f są: x i $h(y)$,
2. 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:

1. pierwszymi argumentami f są: $h(x)$ i $g(x)$,
2. 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.

Przykład 5. 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. Podstawiamy $y \mapsto g(a)$:

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

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 x. Podstawiamy $x \mapsto a$ i otrzymujemy:

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

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.

Przykład 6. 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.

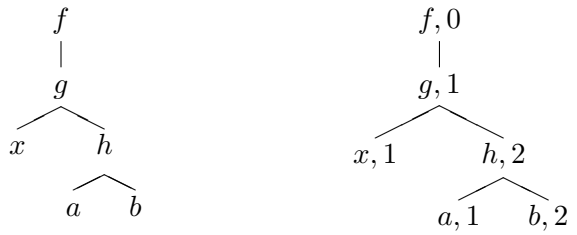
Powyższe przykłady (zaczerpnięte z: Baader, Snyder 2001, Hedman 2004 oraz Nerode, Shore 1997) dobrane są tak, aby zilustrować algorytmiczny sposób odnajdywania mgu dla zbioru literałów (lub pokazania, że zbiór literałów nie jest uzgadnialny).

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

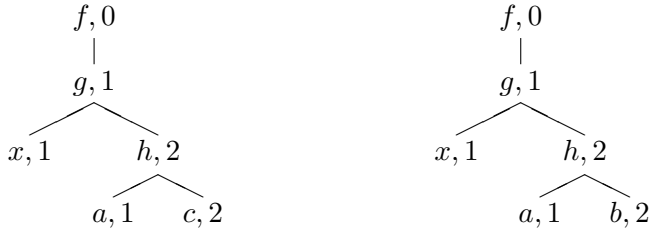
E_2 są różne oraz E_1 i E_2 występują na tych samych pozycjach jako podwyrażenia dwóch wyrażeń w S. Dla przykładu, gdy $S = \{x, g(a, y, u), g(z, b, v)\}$, to zbiorami niezgodności dla S są: $\{a, z\}, \{y, b\}, \{u, v\}, \{x, g(a, y, u)\}, \{x, g(z, b, v)\}$

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:

Term $f(g(x, h(a, b)))$:

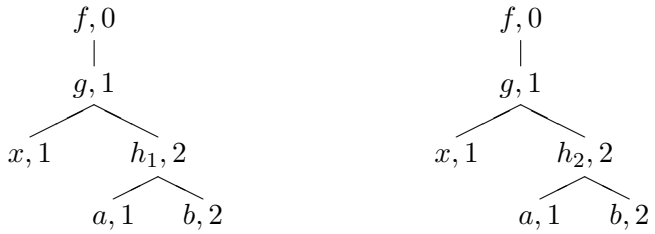


Lewe drzewo to po prostu drzewo syntaktyczne termu $f(g(x, h(a, b)))$. W drzewie prawym zaznaczono pozycje poszczególnych argumentów.



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 $\{x \mapsto t\}$. Dowodzi się, że:

1. Jeśli σ jest unifikatorem dla S, a D jest jednym ze zbiorów niezgodności dla S, to:
   (a) σ jest unifikatorem dla D,
   (b) każdy element D jest termem,
   (c) jednym z elementów D jest zmienna, która nie występuje w drugim elemencie D.
2. 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 skończonym zbiorem wyrażeń (termów lub formuł) bez kwantyfikatorów i niech V_S będzie zbiorem wszystkich zmiennych występujących w S. Dowodzi się, że:

1. Jeśli S jest uzgadnialny, to uzgadnialny jest też każdy zbiór otrzymany z S przez eliminację zmiennych.
2. Przez eliminację zmiennych można z S otrzymać jedynie skończenie wiele zbiorów.
3. Jeśli S' otrzymano z S przez eliminację zmiennych względem $\{x \mapsto t\}$, to moc zbioru S' jest mniejsza niż moc S oraz $V_{S'} = V_S - \{x\}$.
4. 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 ufundowane, tj. nie istnieją nieskończone ciągi zbiorów otrzymywanych przez kolejne eliminacje zmiennych.

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

1. $\emptyset$ jest (jedynym) obliczonym unifikatorem dla dowolnego jednoelementowego zbioru wyrażeń bez kwantyfikatorów.

2. 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.

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

1. Jeśli zbiór S jest uzgadnialny, to σ jest minimalnym unifikatorem dla S wtedy i tylko wtedy, gdy σ jest obliczonym unifikatorem dla S.

2. Jeśli zbiór S jest uzgadnialny, to obliczony unifikator dla S jest najbardziej ogólnym unifikatorem dla S.

Dla sformułowania algorytmu 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 $D(S)$.

W Przykładzie 4 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 $D(S)$.

4 Algorytm unifikacji

4.1 Algorytm

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.

W wielu podręcznikach przedstawiany jest następujący algorytm unifikacji $\mathfrak{A}$. Niech dany będzie zbiór literałów S języka pierwszego rzędu. 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.

Struktura algorytmu unifikacji.

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 $t \in D(S_k)$:

1. Jeśli nie, to algorytm kończy pracę: S nie posiada mgu.
2. 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 $k+2$.

4.2 Przykład

Przykład 7. Niech: $S = \{P(f(y, g(z)), h(b)), P(f(h(w), g(a)), u), P(f(h(b), g(z)), y)\}$.

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:

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

Przypuśćmy, że wybierzemy pierwszą możliwość. Jeśli $\sigma_1 = \{y \mapsto h(w)\}$, to: $S_1 = S_0\sigma_1 = \{P(f(h(w), g(z)), h(b)), P(f(h(w), g(a)), u), P(f(h(b), g(z)), h(w))\}$

Krok 2. Mamy: $D(S_1) = \{w, b\}$, więc niech $\sigma_2 = \{w \mapsto b\}$. Wtedy: $S_2 = S_1\sigma_2 = \{P(f(h(b), g(z)), h(b)), P(f(h(b), g(a)), u), P(f(h(b), g(z)), h(b))\}$

Krok 3. Mamy $D(S_2) = \{z, a\}$, a więc $\sigma_3 = \{z \mapsto a\}$. Wtedy: $S_3 = S_2\sigma_3 = \{P(f(h(b), g(a)), h(b)), P(f(h(b), g(a)), u), P(f(h(b), g(a)), h(b))\}$

Krok 4. Mamy $D(S_3) = \{u, h(b)\}$. Wtedy $\sigma_4 = \{u \mapsto h(b)\}$ i otrzymujemy: $S_4 = S_3\sigma_4 = \{P(f(h(b), g(a)), h(b)), P(f(h(b), g(a)), h(b)), P(f(h(b), g(a)), h(b))\}$

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

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

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

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

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

Opracowano cały szereg bardziej subtelnych algorytmów (zob. pozycje podane w bibliografii).

5 Rezolucja w logice pierwszego rzędu

5.1 Reguła rezolucji

Przypominamy niektóre wcześniej omawiane pojęcia:

1. Literał pozytywny: formuła atomowa.
2. Literał negatywny: negacja formuły atomowej.
3. Literały komplementarne: para wzajem sprzecznych literałów.
4. Klauzula: (uogólniona) alternatywa literałów.
5. Klauzula pusta: pusta alternatywa, oznaczana np. $\Box$.
6. Klauzula Hornowska: klauzula zawierająca co najwyżej jeden literał pozytywny.
7. Klauzula programowa: klauzula z dokładnie jednym literałem pozytywnym.
8. Reguła: klauzula programowa zawierająca jakieś literały negatywne.
9. Fakt: klauzula programowa bez literałów negatywnych.
10. Klauzula celowa: klauzula bez literałów pozytywnych.
11. Program: zbiór klauzul programowych (reguł lub faktów).

Klauzule (w języku pierwszego rzędu) reprezentują formuły w skolemowej postaci normalnej (zob. wykład dotyczący postaci normalnych i prefiksowych).

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 $\forall x \forall y \ (P(x) \rightarrow Q(x))$.

Plan dalszych działań jest następujący:

1. Podamy regułę rezolucji dla języka logiki pierwszego rzędu.
2. Zakładamy, że w języku tym występują stałe indywidualne, predykaty oraz symbole funkcyjne.
3. Nie uwzględniamy predykatu identyczności (wymaga on specjalnego potraktowania).

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

1. $D_1 \cup \{P(t_1^1, \ldots, t_1^k), \ldots, P(t_n^1, \ldots, t_n^k)\}$ oraz
2. $D_2 \cup \{\neg P(s_1^1, \ldots, s_1^k), \ldots, \neg P(s_m^1, \ldots, s_m^k)\}$, odpowiednio.

Jeśli σ jest najbardziej ogólnym unifikatorem dla

$$\{P(t_1^1, \ldots, t_1^k), \ldots, P(t_n^1, \ldots, t_n^k), P(s_1^1, \ldots, s_1^k), \ldots, P(s_m^1, \ldots, s_m^k)\},$$

to $D_1\sigma \cup D_2\sigma$ jest *rezolwentą* dla C_1 i C_2.

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

1. C jest identyczna z C_n
2. 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 dwójkowe T o następujących własnościach:

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

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

5.2 Przykłady

Przykład 8. Rezolwentą klauzul:

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

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

1. $C_1 = \{Q(x), \neg R(y)\} \cup \{P(x, y), P(f(z), f(z))\}$
2. $C_2 = \{\neg N(u), \neg R(w)\} \cup \{\neg P(f(a), f(a)), \neg P(f(w), f(w))\}$
3. zastosowanie najbardziej ogólnego unifikatora $\sigma = \{x \mapsto f(a), y \mapsto f(a), z \mapsto a, w \mapsto a\}$ dla uzgodnienia zbioru literałów

$$\{P(x, y), P(f(z), f(z)), P(f(a), f(a)), P(f(w), f(w))\}$$

daje:

4. $\{Q(x), \neg R(y)\}\sigma = \{Q(f(a)), \neg R(f(a))\}$
5. $\{\neg N(u), \neg R(w)\}\sigma = \{\neg N(u), \neg R(a)\}$.

Przykład 9. 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):

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

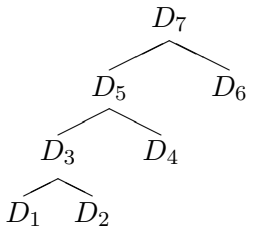
Powyższe warunki mają następujące reprezentacje w postaci klauzul (po rozdzieleniu zmiennych w przesłankach):

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

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

1. $D_1 = C_1 = \{\neg P(x,y), \neg P(y,z), \underline{P(x,z)}\}$
2. $D_2 = C_2\{u \mapsto x, v \mapsto z\} = \{\neg \underline{P(u,v)}, P(v,u)\}\{u \mapsto x, v \mapsto z\} = \{\underline{\neg P(x,z)}, P(z,x)\}$
3. $D_3 = \{\neg P(x,y), \neg P(y,z), \underline{P(z,x)}\}$ rezolwenta D_1 oraz D_2
4. $D_4 = C_2\{u \mapsto z, v \mapsto x\} = \{\neg \underline{P(u,v)}, P(v,u)\}\{u \mapsto z, v \mapsto x\} = \{\underline{\neg P(z,x)}, P(x,z)\}$
5. $D_5 = \{\neg P(x,y), \underline{\neg P(y,z)}, P(x,z)\}$ rezolwenta D_3 i D_4
6. $D_6 = C_2\{u \mapsto z, v \mapsto y\} = \{\neg P(u,v), \underline{P(v,u)}\}\{u \mapsto z, v \mapsto y\} = \{\neg P(z,y), \underline{P(y,z)}\}$
7. $D_7 = \{\neg P(x,y), \neg P(z,y), P(x,z)\} = C_3$ rezolwenta D_5 i D_6.

Przykład 9: rezolucyjne drzewo dowodowe. Dowód rezolucyjny z tego przykładu reprezentuje następujące drzewo:



Często przedstawia się tego typu drzewa w takiej postaci:

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

6 Fakty metalogiczne

1. **Twierdzenie Herbranda**. Zbiór S klauzul (języka pierwszego rzędu) jest niespełnialny wtedy i tylko wtedy, gdy niespełnialny jest pewien skończony zbiór klauzul, będących bazowymi instancjami (klauzulami ustalonymi) klauzul z S.

2. Twierdzenie Herbranda pozwala zatem redukować problem spełnialności w logice pierwszego rzędu do problemu spełnialności w rachunku zdań.

3. **Twierdzenie o trafności rezolucji w logice pierwszego rzędu**. Dla dowolnego zbioru formuł S, jeśli $\Box \in \mathcal{R}(S)$, to S nie jest spełnialny.

4. **Twierdzenie o pełności rezolucji w logice pierwszego rzędu**. Niech A będzie dowolnym zdaniem języka logiki pierwszego rzędu w skolemowej postaci normalnej. Jeśli A jest niespełnialne, to $\Box \in \mathcal{R}(A)$.

7 Tablice analityczne ze zmiennymi wolnymi

7.1 Reguły

Z procedury unifikacji korzystamy m.in. w metodzie tablic analitycznych ze zmiennymi wolnymi.

1. Jeśli T jest tablicą analityczną, a σ podstawieniem wolnym dla wszystkich formuł w T, to mówimy, że σ jest *wolne dla* T.

2. **Reguła podstawień**. Jeśli T jest tablicą analityczną, a σ jest wolne dla T, to $T\sigma$ też jest tablicą analityczną (tu $T\sigma = \{\Phi\sigma : \Phi \in T\}$).

3. **Reguła dla γ-formuł**: $\frac{\gamma}{\gamma(x)}$ dla zmiennej wolnej x (która nie występuje jako związana w tablicy).

4. **Reguła dla δ-formuł**: $\frac{\delta}{\overline{\delta(f(x_1,\ldots,x_n))}}$, gdzie f jest nowym symbolem funkcyjnym, zaś $x_1, \ldots, x_n$ są wszystkimi zmiennymi wolnymi na rozważanej gałęzi.

7.2 Przykład: Fitting 1990, 153

1.	$\neg(\exists z \forall x\ R(x,z,f(x,z)) \rightarrow \exists z \forall x \exists y\ R(x,z,y))$	
2.	$\exists z \forall x\ R(x,z,f(x,z))$	$1 : \alpha$
3.	$\neg\exists z \forall x \exists y\ R(x,z,y))$	$1 : \alpha$
4.	$\forall x\ R(x,a,f(x,a))$	$2 : \delta, a$
5.	$\neg\forall x \exists y\ R(x,x_1,y))$	$3 : \gamma, x_1$
6.	$\neg\exists y\ R(g(x_1),x_1,y))$	$5 : \delta, g$
7.	$R(x_2,a,f(x_2,a))$	$4 : \gamma, x_2$
8.	$\neg R(g(x_1),x_1,x_3))$	$6 : \gamma, x_3$

Podstawienie $\sigma = \{x_1 \mapsto a, x_2 \mapsto g(a), x_3 \mapsto f(g(a),a)\}$ przekształca formuły 7 i 8 w parę zdań wzajem sprzecznych, co pozwala zamknąć gałąź (a więc i całą tablicę).

Warto próbować wyobrazić sobie 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.

8 Wykorzystywana literatura

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

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

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

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

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

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

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

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

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

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

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

JERZY POGONOWSKI
Zakład Logiki i Kognitywistyki UAM
www.kognitywistyka.amu.edu.pl
http://logic.amu.edu.pl/index.php/Dydaktyka
pogon@amu.edu.pl