Programowanie w logice i funkcyjne (prezentacja niekompletna i przed korektą)

Konrad Zdanowski

Outline

Literatura

Pozycje dosępne (do ściągnięcia)z adresów UKSW przez stronę `springer.com`.

- Clocksin, Mellish, Programming in Prolog, Springer, 2003.
- Cocksin, Clause and Effect, Springer, 1997.
- Deransart, Ed-Dbali, Cervoni, Prolog. The Standard, Springer, 1996.
- Serrano, Beginning Haskell, Springer, 2014.

Pozycje online:

- Nilsson, Maluszynski, Logic, Programming and Prolog (2ed), `https://www.ida.liu.se/~ulfni53/lpp/index.html`,
- Lipovaca, Learn You a Haskell for Great Good!, `http://learnyouahaskell.com/chapters`.

Paradygmaty programowania

- paradygmat imperatywny
  - języki proceduralne
  - języki obiektowe
- paradygmat deklaratywny
  - prograwmowanie funkcyjne
  - programowanie w logice
  - języki kwerend do bazy danych (kwerenda SELECT w SQL)
  - HTML

Cechy paradygmatu deklaratywnego

- użytkownik opisuje jakie własności ma rozwiązanie problemu
- za obliczenie/wyszukanie rozwiązania lub obliczenie wartości funkcji odpowiada mechanizm zaimplementowany w kompilatorze lub interpreterze

Cechy paradygmatu deklaratywnego - przykład

Znajdź wszystkich autorów i napisane przez nich książki, gdzie autorzy urodzili się po 1989 roku i napisali przynajmniej 5 książek.

```
SELECT Osoby.nazwisko, Ksiazki.tytul
FROM Osoby join Ksiazki on Osoby.id=Ksiazki.autor_id
WHERE Osoby.rok_urodzenia >= 1990
HAVING count(*)>=5
GROUP BY Osoby.nazwisko;
```

Cechy paradygmatu deklaratywnego

- zaimplementowany mechanizm obliczeniowy jest bardzo ogólny
  - wnioskowanie w logice pierwszego rzędu (programowanie w logice)
  - defniowanie funkcji skończonych rzędów, rekursja (programowanie funkcyjne)
- skonstruowany algorytm nie musi być optymalny ale
  - program może być bardziej czytelny, krótszy
  - analiza programu jest prostsza (przejrzystość referenacycyjna)
- wybór paradygmatu programowania i języka powinien zależeć od problemu do rozwiązania

Outline

Programowanie w logice - PwL

- Program jest bazą wiedzy, w którym reprezentujemy fakty
- Zapytania do tej bazy generują obliczenia, które szukają odpowiedzi
- Mechanizm obliczeniowy bazuje na rezolucji dla logiki pierwszego rzędu

Wobec tego aby poznać PwL trzeba

- poznać rezolucyjny system dowodzenia,
- poznać mechanizm kontroli tego wnioskowania używany w PwL.

Programowanie w logice

- Mechanizm dowodzenia w logice jest bardzo silnym mechanizmem obliczeniowym.
- Problem tautologii (bycia dowodliwą formułą) logiki pierwszego rzędu jest nierozstrzygalny.
- Bardzo ważny jest system kontroli przebiegu wnioskowania w programie.

Outline

Klasyczny rachunek zdań

Zbiór zmiennych: p,q,r,. . .

Zbiór spójników zdaniowych: $\wedge$, $\vee$, $\rightarrow$, $\neg$

Czy wszyscy pamiętają tabelki prawdziwościowe spójników zdaniowych?

Definicja 1

Wartościowanie v jest funkcją ze zbioru zmiennych zdaniowych w $\{0, 1\}$. Dla każdego wartościowania v i formuły KRZ φ możemy wyliczyć wartość φ przy v.

Rozszerzamy więc wartościowania na zbiór wszystkich formuł i przez $v(\varphi)$ oznaczamy wartość formuły φ przy wartościowaniu v.

Przykład 2

Niech $v(p) = v(q) = 1$, $v(r) = 0$. Niech $\varphi = (p \vee \neg q) \wedge (\neg p \vee r)$.
Ile wynosi $v(\varphi)$?
$v(\neg q) = 0$, $v(p \vee \neg q) = 1$, $v(\neg p) = v(r) = 0$, $v((\neg p \vee r) = 0$, $v(\varphi) = 0$.

Piszemy też

$$v \models \varphi, \quad \text{gdy } v(\varphi) = 1,$$
$$v \not\models \varphi, \quad \text{gdy } v(\varphi) = 0.$$

Fakt 1

Jeśli zmienne φ zawierają się w zbiorze $\{p_1, \ldots, p_n\}$ to $v(\varphi)$ zależy tylko od $v(p_1), \ldots, v(p_n)$.

Definicja 3

Formuły φ i ψ są równoważne, ozn. $\varphi \equiv \psi$, gdy dla dowolnego wartościowania v, $v(\varphi) = v(\psi)$.

Definicja 4

Formuła jest tautologią, jeśli dla każdego v, $v \models \varphi$.
Formuła jest spełnialna, jeśli dla pewnego v, $v \models \varphi$.
Formuła jest kontrtautologią, jeśli dla każdego v, $v \not\models \varphi$.

Definicja 5

Reguła wnioskowania to para $\frac{\varphi_1,\ldots,\varphi_n}{\varphi}$.
Reguła wnioskowania jest poprawna jeśli dla każdego wartościowania v*, jeśli* $v(\varphi_i) = 1$*, dla wszystkich* $i \leqslant n$*, to* $v(\varphi) = 1$.

Poprawna reguła wnioskowania prowadzi od zdań prawdziwych do zdań prawdziwych, od tautologii do tautologii.

Definicja 6

Reguła rezolucji ma postać

$$\frac{\varphi \vee p, \psi \vee \neg p}{\varphi \vee \psi}.$$

Fakt 2

Reguła rezolucji jest poprawną regułą wnioskowania.

Rezolucja – rachunek zdań

Definicja 7

Literał to zmienna lub jej negacja.

Przykład. p, q, $\neg p$, ...

Definicja 8

Formuła KRZ φ jest w postaci koniunkcyjno–alternatywnej jeśli $\varphi = \varphi_1 \wedge \cdots \wedge \varphi_n$, gdzie każde φ_i to alternatywa literałów.

Przykład. $(p \vee \neg q) \wedge (\neg p \vee \neg q \vee r)$.
Przykład. $(p \vee q)$, $\neg p$, $p \wedge \neg q$.

Twierdzenie 9

Każda formuła KRZ jest równoważna formule w postaci KA.

Reguły wykorzystywane do otrzymania postaci KA:

- $\varphi \rightarrow \psi \equiv \neg\varphi \vee \psi$,
- $\neg(\varphi \vee \psi) \equiv (\neg\varphi \wedge \neg\psi)$,
- $\neg(\varphi \wedge \psi) \equiv (\neg\varphi \vee \neg\psi)$,
- $\neg\neg\varphi \equiv \varphi$,
- $\varphi \vee (\psi \wedge \gamma) \equiv (\varphi \vee \psi) \wedge (\varphi \vee \gamma)$.

Outline

Definicja 10

Klauzula to zbiór literałów.
Niech $\varphi = l_1 \vee \ldots l_n$*, gdzie* l_i *literał.* $K(\varphi) = \{l_1, \ldots, l_n\}$.

Definicja 11

Niech v wartościowanie, K klauzula. $v(K) = 1$ *(*$v \models K$*) jeśli istnieje* $l \in K$*,* $v(l) = 1$.

Czyli jeśli $K = \{l_1, \ldots, l_n\}$, to

$$v \models K \iff v \models (l_1 \vee \cdots \vee l_n).$$

Uwaga. $v \not\models \emptyset$!

Definicja 12

Niech v wartościowanie, $\mathcal{K}$ *zbiór klauzul.* $v(\mathcal{K}) = 1$ *(*$v \models \mathcal{K}$*) jeśli dla wszystkich* $K \in \mathcal{K}$*,* $v(K) = 1$.

Definicja 13 (Reguła rezolucji dla klauzul)

Niech $p \in K_1, \neg p \in K_2$. *Resolwent* K_1, K_2 *względem* p *to*

$$\mathrm{Res}(K_1, K_2, p) = K_1 \setminus \{p\} \cup K_2 \setminus \{\neg p\}.$$

Przykład. $K_1 = \{t, q, \neg r\}$, $K_2 = \{t, w, \neg q\}$, $\mathrm{Res}(K_1, K_2, q) = \{t, \neg r, w\}$.
Jeśli $v \models K_1$, $v \models K_2$, to $v \models \mathrm{Res}(K_1, K_2, p)$.

Definicja 14

$\mathcal{K}$ zbiór klauzul. Dowód klauzuli L ze zbioru $\mathcal{K}$ to ciąg $K_1, \ldots, K_n$ taki, że

- $K_n = L$,
- *dla każdego* $i \leqslant n$
  - $K_i \in \mathcal{K}$ *lub*
  - *istnieje zmienna p oraz istnieją* $j, k < i$ *takie, że* $K_i = \mathrm{Res}(K_j, K_k, p)$.

Jeśli istnieje dowód L z $\mathcal{K}$ to piszemy $\mathcal{K} \vdash L$.

Fakt 3

- *Jeśli* $\mathcal{K} \vdash L$ *i* $v \models \mathcal{K}$, *to* $v \models L$.
- *Jeśli* $\mathcal{K} \vdash \emptyset$, *to* $\mathcal{K}$ *nie ma modelu.*

Przykład dowodu

Twierdzenie 15 (Twierdzenie o pełności)

Równoważne są

- $\mathcal{K} \vdash \emptyset$,
- $\mathcal{K}$ *jest niespełnialne.*

Definicja 16

Klauzula jest Hornowska jeśli zawiera najwyżej jeden literał bez negacji. Formuła jest Hornowska jeśli jest postaci $p_1 \wedge \cdots \wedge p_n \rightarrow q$ (koniunkcja zmiennych w poprzedniku implikacji i jedna zmienna w następniku).

Przykład. $\{p, \neg q, \neg r\}$, $\{\neg q, \neg r\}$, $\emptyset$. **Uwaga.** Nie wszystko da się wyrazić formułą Hornowską, np. $p \vee q$.

Formule Hornowskiej odpowiada klauzula Hornowska.

$$v \models (p_1 \wedge \cdots \wedge p_n \rightarrow q) \iff v \models \{\neg p_1, \ldots, \neg p_n, q\}.$$

SLD rezoulcja

- S (unique selection) – jednoznaczny wybór zmiennej, w każdym resolwencie,
- L (linear) – liniowość,
- D (definite clauses) – z klauzul, które zawierają dokładnie jedno wystąpienie pozytywne.

SLD rezoulcja

Definicja 17

Klauzula jest negatywna jeśli zawiera same negacje.
Klauzula jest implikacyjna jeśli zawiera dokładnie jedną zmienną pozytywną (bez negacji).

Definicja 18

Niech $\mathcal{K}$ zbiór klauzul Hornowskich takich, że jedna i tylko jedna, jest negatywna. Niech $G \in \mathcal{K}$ będzie taką klauzulą (klauzulą celu).

Wywód rezolucyjny jest SLD jeśli w każdym kroku bierze udział klauzula negatywna. Co więcej, w każdym kroku, z wyjątkiem pierwszego, bierze udział klauzula negatywna powstała w poprzednim kroku.

Jeśli L ma SLD dowód z $\mathcal{K}$ to piszemy $\mathcal{K} \vdash_{SLD} L$.

Przykład. Rysunek dowodu SLD.
Uwaga. Dlaczego dowody SLD są prostsze?

Fakt 4

Problem istnienia wywodu w SLD rezolucji dla KRZ ma złożoność wielomianową.

Twierdzenie 19 (Twierdzenie o pełności SLD rezolucji)

Niech $\mathcal{K}$ zbiór klauzul Hornowskich z jedną klauzulą negatywną. Równoważne są

- $\mathcal{K} \vdash_{SLD} \emptyset$,
- $\mathcal{K}$ *jest niespełnialne.*

Czyli dla formuł Hornowskich możemy ograniczyć się do SLD rezolucji.

Outline

Klasyczny rachunek predykatów (KRP) – składnia

Termy

- Zbiór zmiennych $\mathbb{V} = \{x_0, x_1, \ldots, x, y, z, \ldots\}$
- Zbiór stałych $\mathbb{C}$: a, b, c, ala, kot, ...
  Zakładamy niepustość zbioru stałych, $\mathbb{C} \neq \emptyset$.
- Zbiór symboli funkcyjnych $\mathbb{F}$: f, g, ...
- Z każdym symbolem funkcyjnym związana jest jego arność, która mówi z iloma argumentami występuje f. Niech $\mathrm{ar}(f)$ to arność f.
  Przyjmujemy, że arność stałych to zero.

Definicja 20

Zbiór termów $\mathbb{T}$ to domknięcie $\mathbb{C} \cup \mathbb{V}$ na stosowanie funkcji czyli to najmniejszy zbiór X taki, że

- $\mathbb{C} \cup \mathbb{V} \subseteq X$,
- *dla dowolnego $f \in \mathbb{F}$ i dowolnych $t_1, \ldots, t_{\mathrm{ar}(f)} \in X$,*

$$f(t_1, \ldots, t_{\mathrm{ar}(f)}) \in X.$$

- Termy są zdefiniowane relatywnie do zbiorów $\mathbb{C}$ i $\mathbb{F}$ czyli relatywnie do słownika. Inną częścią słownika będą predykaty.
- Termy są nazwami obiektów w uniwersum.
- W Prologu termy będą rozwiązaniami, których będzie szukał program podczas swojego wykonania.

Definicja 21

Term domknięty to term, w którym nie występują zmienne.

Definicja 22

Uniwersum Herbranda dla danego słownika σ to zbiór wszystkich termów domkniętych nad tym słownikiem.

- W uniwersum Hebranda utożsamiamy obiekty z ich nazwami (termami).
- Możemy myśleć o uniwersum Herbranda jako o obiektach w pewnych świecie (modelu).
- W interesującym nas przypadku są to wszystkie obiekty modelu.

Zbiór formuł

Niech $\mathbb{P}$ to zbiór predykatów. Niech $\mathrm{ar}(P)$ to arność predykatu P.

Definicja 23

Formuła atomowa ma postać

$$P(t_1, \ldots, t_{\mathrm{ar}(P)}),$$

gdzie $t_1, \ldots, t_{\mathrm{ar}(P)}$ *to termy.*

Uwaga. Jeśli w języku jest równość, to drugim rodzajem formuły atomowej jest równość dwóch termów, $t = s$. My rozważamy język bez równości.

Definicja 24

Zbiór formuł to domknięcie zbioru formuł atomowych na spójniki zdaniowe oraz kwantyfikację.

Definicja 25

Baza Herbranda (dla ustalonego słownika) to zbiór wszystkich formuł atomowych postaci $P(t_1, \ldots, t_n)$, gdzie $t_1, \ldots, t_n$ to termy domknięte.

- A priori, nie możemy zakładać, że którekolwiek dwa termy oznaczają ten sam obiekt.
- Brak równości w języku sprawia, że nie możemy nawet tego wyrazić.
- Wobec tego każde dwie formuły z bazy Herbranda są od siebie niezależne (o ile nie założymy dodatkowej teorii).

Model dla KRP

Dla uproszczenia rozważmy język z jednym predykatem $P(x, y)$ oraz dwoma symbolami funkcyjnymi $f(x), g(x, y)$ i stałą c.

Definicja 26

Modelem M dla powyższego słownika będzie n-tka postaci

$$M = (U, R_P, F_f, F_g, a_c),$$

gdzie

- $U \neq \emptyset$,
- $P \subseteq U^2$,
- $F_f : U \to U$,
- $F_g : U^2 \to U$,
- $a_c \in U$.

Model dla KRP

Funkcja $v\colon \mathbb{V} \to U$ to wartościowanie.
Wartościowanie v rozszerzamy na zbiór wszystkich termów, przez indukcję po budowie termu:

- $v(c) = a_c$,
- $v(f(t)) = F_f(v(t))$,
- $v(g(t,s)) = F_g(v(t), v(s))$.

Model M z wartościowaniem v spełnia formułę atomową $P(t,s)$, ozn. $M \models P(t,s)[v]$, jeśli $(v(t), v(s)) \in R_P$.

$$M \models P(t,s)[v] \iff (v(t), v(s)) \in R_P.$$

Wtedy

$$\begin{aligned} M \models \neg P(t,s)[v] &\iff \text{nieprawda, że } M \models P(t,s)[v] \\ &\iff (v(t), v(s)) \notin R_P. \end{aligned}$$

Formuły Hornowskie w KRP

Definicja 27

Formuła Hornowska ma postać

$$\forall \bar{x}(\varphi_1 \wedge \cdots \wedge \varphi_n \rightarrow \varphi),$$

gdzie $\varphi_1, \ldots, \varphi_n, \varphi$ *to formuły atomowe postaci* $P(\bar{t})$ *a* $\bar{x}$ *to wszystkie zmienne w nich występujące.*
Dla formuły Hornowskiej ψ*, jak wyżej, defniujemy klauzlę* $K(\psi)$ *postaci*

$$K(\psi) = \{\neg\varphi_1, \ldots, \neg\varphi_n, \varphi\}.$$

Dla uproszczenia nie będziemy pisać ogólnych kwantyfikatorów ale domyślnie wszystkie zmienne w formule Hornowskiej są skwantyfikowane.

- Jak zdefiniować pojęcie spełniania dla formuł Hornowskich?
- Jak wnioskować w oparciu o formuły Hornowskie?
- Jak wykorzystać rezolucję?

Model dla KRP

Definicja 28

Niech $\psi = \forall \bar{x}(\varphi_1 \wedge \cdots \wedge \varphi_n \rightarrow \varphi)$, formuła Hornowska.
M spełnia ψ ($M \models \psi$) gdy

dla każdego wartościowania v, jeśli $M \models \varphi_i[v]$, dla $i \leqslant n$, to $M \models \varphi[v]$.

Niech T zbiór formuł Hornowskich. M spełnia T ($M \models T$) jeśli spełnia wszystkie formuły z T.

Model *M* spełnia ψ gdy

- dla dowolnego wartościowania *v*, *M* spełnia (w sensie rachunku zdań) $\varphi_1 \wedge \cdots \wedge \varphi_n \rightarrow \varphi$ przy wartościowaniu *v*.
- dla dowolnego wartościowania *v* istnieje $\gamma \in K(\psi)$ takie, że $M \models \gamma[v]$.

Niesprzeczność

Definicja 29

Zbiór formuł Hornowskich T jest niesprzeczny jeśli ma model.

Model Herbranda

Zakładamy, że słownik zawiera przynajmniej jedną stałą. Dla uproszczenia przyjmujemy słownik taki jak wcześniej (P, f, g, c).

Definicja 30 (Model Herbranda)

Niech U będzie zbiorem termów domkniętych dla danego słownika.
Niech R będzie podzbiorem bazy Herbranda (formuł atomowych z domkniętymi termami).
Modelem Herbranda $M_H(R)$ dla danego słownika i zbioru R nazywamy model postaci $M_H(R) = (U, P(R), F, G, c)$, gdzie

- $(t, s) \in P(R) \iff P(t, s) \in R$,
- $F(t) = f(t)$,
- $G(t, s) = g(t, s)$.

Model Herbranda dla teorii

- Niech T zbiór formuł Hornowskich.
- Niech R zbiór formuł z bazy Herbranda, które wynikają z T.
- Modelem Herbranda dla T, $M_H(T)$, nazywamy model $M_H(R)$.
- Nie wiemy jednak co oznacza termin „wynikają”.

Podstawienie

Definicja 31

Podstawienie θ to dowolna funkcja ze zbioru zmiennych w zbiór termów.

Jeśli $t = t(x_1, \ldots, x_n)$ jest termem, to $t\theta$ to term postaci

$$t(\theta(x_1), \ldots, \theta(x_n)).$$

Dla $\varphi = P(t_1, \ldots, t_n)$, formuły atomowej, $\varphi\theta$ ma postać

$$P(t_1\theta, \ldots, t_n\theta).$$

Podobnie $\neg\varphi\theta = \neg(\varphi\theta)$.

Dla klauzuli $K = \{\varphi_1, \ldots, \varphi_n\}$, $K\theta$ to

$$\{\varphi_1\theta, \ldots, \varphi_n\theta\}.$$

Model Herbranda dla teorii

Definicja 32

Podstawienie bazowe to dowolna funkcja θ ze zbioru zmiennych w termy domknięte.

Definicja 33

Dla formuły Hornowskiej $\psi = \forall \bar{x}(\varphi_1 \wedge \cdots \wedge \varphi_n \rightarrow \varphi)$ i podstawienia bazowego θ definiujemy $\psi\theta$ jako formułę

$$\varphi_1\theta \wedge \cdots \wedge \varphi_n\theta \rightarrow \varphi\theta.$$

Definicja 34

Dla teorii T składającej się ze zdań Hornowskich definiujemy T_{KRZ} jako zbiór wszystkich formuł postaci $\psi\theta$, gdzie $\psi \in T$ a θ to podstawienie bazowe,

$$T_{KRZ} = \{\psi\theta : \psi \in T \text{ i } \theta \text{ to podstawienie bazowe}\}.$$

Twierdzenie 35

Niech T to zbiór formuł Hornowskich. Niech R_T to zbiór formuł z bazy Herbranda, które wynikają z T_{KRZ} w sensie rachunku zdań.
Jeśli T ma model, to $M_H(R_T) \models T$.

Uwaga. Model Herbranda dla teorii T to właśnie model skonstruowany w powyższym twierdzeniu. Model ten będziemy oznaczać przez $M_H(T)$.

Model Herbranda cd.

Definicja 36

Dla teorii T składającej się ze zdań Hornowskich definiujemy T_{KRZ} jako zbiór wszystkich formuł postaci $\psi\theta$, gdzie $\psi \in T$ a θ to podstawienie bazowe,

$$T_{KRZ} = \{\psi\theta \colon \psi \in T \text{ i } \theta \text{ to podstawienie bazowe}\}.$$

Niech BH to baza Herbranda czyli zbiór formuł atomowych bez zmiennych, czyli formuł postaci $P(t_1, \ldots, P_n)$, gdzie każdy t_i to term domknięty.

- Niech $R_0 = T_{KRZ} \cap \text{BH}$.
- Niech

$$R_{i+1} = R_i \cup \{\varphi \in \text{BH} \colon \text{istnieje } \varphi_1 \wedge \cdots \wedge \varphi_n \to \varphi \in T_{KRZ} \quad \{\varphi_1, \ldots, \varphi_n\} \subseteq R_i\}.$$

- Niech $R_T = \bigcup_{i \in \mathbb{N}} R_i$.

Twierdzenie 37

Jeśli T ma model, to $M_H(R_T)$ jest modelem Herbranda dla T.

Model Herbranda cd.

- Model $M_H(T)$ jest, w pewnym sensie, modelem minimalnym dla T.
- Jego uniwersum jest minimalne bo składa się tylko z termów, które nazywają elementy modelu.
- Prawdziwe są w nim tylko te fakty z bazy Herbranda, które mają dowód na gruncie T.

Wniosek Formuła $P(a) \vee P(b)$ nie jest równoważna żadnej teorii wyrażonej przez formuły Herbranda.

Model Herbranda cd.

- Model skonstruowany na poprzednich slajdach ma znaczenie dla semantyki Prologu.
- Prolog przyjmuje, że wszystkie obiekty to termu z uniwersum Herbranda.
- Wnioskowanie, które wykonuje Prolog to wnioskowanie podobne do powyższego ale
  - ze zdefiniowaną kolejnością wykonywania inferencji,
  - lepszym wyborem podstawieniem, używane są nie tylko podstawienia zmiennych za domknięte termy ale podstawienia bardziej ogólne (MGU).
- Zajmiemy się więc problemem jak wnioskować używając formuł Hornowskich?

Przykład wnioskowania

Założenia:

- $\forall x \forall y (P(f(x)) \wedge S(f(y)) \rightarrow H(x, g(y))$,
- $\forall x (S(x) \rightarrow P(f(x)))$,
- $\forall x (P(x) \rightarrow S(f(x)))$,
- $S(alan)$.

Czy istnieją a, b takie, że $H(a, b)$?

- Skoro $S(alan)$ to $P(f(alan))$.
  Używamy drugiego założenia dla $x = alan$.
- Skoro $P(f(alan))$ to $S(f(f(alan)))$.
  Używamy trzeciego założenia dla $x = f(alan)$.
- Skoro $P(f(alan))$ i $S(f(f(alan)))$, to $H(alan, g(f(alan)))$.
  Używamy pierwszego założenia dla $x = alan$ i $y = f(alan)$.

Wniosek: $H(alan, g(f(alan)))$!

Przykład wnioskowania – rezolucja

Zapiszemy założenia bez kwantyfikatorów:

- $\neg P(f(x)) \vee \neg S(f(y) \vee H(x, g(y))$,
- $\neg S(x) \vee P(f(x))$,
- $\neg P(x) \vee S(f(x))$,
- $S(alan)$.

Odpowiadają im klauzule:

- $\{\neg P(f(x)), \neg S(f(y), H(x, g(y)\}$,
- $\{\neg S(x), P(f(x))\}$,
- $\{\neg P(x), S(f(x))\}$,
- $\{S(alan)\}$.

Ale jak wykonać rezolucję?

Klauzule są prawdziwe dla wszystkich podstawień za zmienne!

Przykład wnioskowania – rezolucja

- $K_1 = \{\neg P(f(x)), \neg S(f(y), H(x, g(y)\}$,
- $K_2 = \{\neg S(x), P(f(x))\}$,
- $K_3 = \{\neg P(x), S(f(x))\}$,
- $K_4 = \{S(alan)\}$.

- Niech, w K_3, $x = f(z)$ i, w K_2, $x = z$.
  Wtedy z $\{\neg S(z), P(f(z))\}$ i $\{\neg P(f(z)), S(f(f(z)))\}$ wnioskujemy
  $K_5 = \{\neg S(z), S(f(f(z)))\}$.
- Niech $z = alan$ w K_5.
  Wtedy z K_4 i z K_5 wnioskujemy $K_6 = \{S(f(f(alan)))\}$.
- Niech $x = alan$ w K_2.
  Z K_2 i z K_4 wnioskujemy $K_7 = \{P(f(alan))\}$.
- Niech $x = alan$ i $y = f(alan)$ w K_1.
  Z K_1, K_6, K_7 wnioskujemy $H(alan, g(f(alan)))$.

- Potrzebny jest algorytm, który wyszukuje takie podstawienie θ, które pozwoli wykonać rezolucję.
- Dla klauzul K_1, K_2 szukamy θ takiego, że istnieje φ, dla którego $\varphi \in K_1\theta$ oraz $\neg\varphi \in K_2\theta$.
- Możemy założyć, że klauzule mają różne zbiory zmiennych oraz, że wartości θ mają tylko nowe zmienne.

Unifikacja

Definicja 38

Niech $\alpha = (t_1, s_1), \ldots, (t_n, s_n)$ *to ciąg par termów. Podstawienie* θ *jest unifikatorem* α *jeśli dla każdego* $i \leqslant n$,

$$t_i\theta = s_i\theta.$$

- Mając dane $\varphi = P(t_1, \ldots, t_n)$ i $\psi = \neg P(s_1, \ldots, s_n)$ będziemy szukać unifikatora dla $(t_1, s_1), \ldots, (t_n, s_n)$.
- Jeśli znajdziemy taki unifikator θ to wtedy $\neg(\varphi\theta) = \psi\theta$.

Przykład. $(f(x, g(y)), f(y, g(h(z))), (f(z, z), f(w, h(c)))$.
Przykład. Jak wyglada unifikator dla pary $(f(x), g(x))$?
Przykład. Jak wyglada unifikator dla pary $(f(x), x)$?

Najbardziej ogólny unifikator – MGU

MGU – most general unifier

Definicja 39

Niech $\alpha = (t_1, s_1), \ldots, (t_n, s_n)$. Podstawienie θ jest najbardziej ogólnym unifikatorem α (MGU) jeśli θ jest unifikatorem α i dowolne podstawienie σ, które jest unifkatorem α da się przedstawić jako

$$\sigma = \theta \circ \tau,$$

gdzie τ jest podstawieniem.

Uwaga. Przyjmujemy konwencję, $t(\theta \circ \tau) = (t\theta)\tau$.

Twierdzenie 40

Niech α będzie ciągiem par termów. Jeśli α posiada unifikator, to α posiada MGU.

Twierdzenie 41

Najbardziej ogólny unifkator jest wyznaczony jednoznacznie z dokładnością do permutacji zmiennych. Tzn. jeśli θ, σ to MGU ciągu termów α, to istnieje taka permutacja zmiennych τ, że $\theta = \sigma \circ \tau$.

Najbardziej ogólny unifikator – algorytm

Wejście: $(t_1, s_1), \ldots, (t_n, s_n)$
Zakładamy, że zmienne są symbolami jednoliterowymi. Stałe traktujemy jako symbole funkcyjne zeroargumentowe.

1. $\theta := \emptyset$.
2. Rozważamy ciągi $\bar{t} = t_1 \ldots t_n$ i $\bar{s} = s_1 \ldots s_n$.
3. Znajdujemy pierwszą pozycję i, na której występuje różnica w $\bar{t}$ i $\bar{s}$.
   1. Jeśli $\bar{t}[i]$ i $\bar{s}[i]$ to dwa różne symbole funkcyjne, to zwracamy „brak MGU”.
   2. Jeśli jedno z $\bar{t}[i]$ i $\bar{s}[i]$ to zmienna, powiedzmy x a drugie to początek termu, w którym x występuje, to zwracamy „brak MGU”.
   3. Teraz jedno z $\bar{t}[i]$ i $\bar{s}[i]$ to zmienna, powiedzmy x a drugie to początek termu, w którym x nie występuje, powiedzmy t.
      $\theta := \theta \circ \{(x, t)\}$ i podstawiamy term t za x w ciągach $\bar{t}$ i $\bar{s}$.
   4. Wracamy do **??**.
4. Jeśli nie ma różnic, zwróć θ.

Najbardziej ogólny unifikator – przykład

Reguła rezolucji w KRP

Definicja 42 (Reguła rezolucji KRP)

Niech K_1, K_2 dwie klauzule o rozłącznych zmiennych. Niech θ będzie takim podstawieniem, że istnieje taka formuła φ, że

$$\varphi \in K_1\theta, \quad \neg\varphi \in K_2\theta.$$

Wtedy, przez $\mathrm{Res}(K_1, K_2, \varphi, \theta)$ *oznaczamy*

$$K_1\theta \setminus \{\varphi\} \cup K_2\theta \setminus \{\neg\varphi\}.$$

Definicja 43 (Dowód rezolucyjny)

$\mathcal{K}$ zbiór klauzul nad formułami KRP. Dowód klauzuli L ze zbioru $\mathcal{K}$ to ciąg $K_1, \ldots, K_n$ taki, że

- $K_n = L$,
- *dla każdego* $i \leqslant n$
  - $K_i \in \mathcal{K}$ *lub*
  - *istnieje formuła* φ, *permutacja zmiennych* τ *i podstawienie* θ *oraz istnieją* $j, k < i$ *takie, że* $K_i = \mathrm{Res}(K_j\tau, K_k, \varphi, \theta)$.

Jeśli istnieje dowód L z $\mathcal{K}$ to piszemy $\mathcal{K} \vdash L$.

Uwaga. Podstawienie τ służy tylko urozłącznieniu zmiennych klauzul użytych w regule rezolucji.

Twierdzenie 44 (Twierdzenie o zupełności dla rezolucji)

Niech $\mathcal{K}$ będzie zbiorem klauzul w logice pierwszego rzędu. Równoważne są:

- $\mathcal{K}$ *jest niespełnialny (sprzeczny semantycznie),*
- $\mathcal{K} \vdash \emptyset$ *($\mathcal{K}$ jest sprzeczny syntaktycznie).*

SLD rezolucja w KRP

- Na wejściu mamy zbiór klauzul Hornowskich.
- Dokładnie jedna z klauzul jest negatywna (same zanegowane formuły atomowe).
- Przyjmujemy, że zmienne w każdej klauzuli są rozłączne.
  Gdyby tak nie było, możemy wykonać podstawienie urozłączniające zmienne. W praktyce podstawienie takie wykonywane jest przed użyciem każdej reguły rezolucji przez interpreter Prologu.
- Ma to uzasadnienie w tym, że każda formuła, z której „powstały” klauzule jest skwantyfikowana ogólnie.

SLD rezolucja a Prolog

- Baza wiedzy (program) T w Prologu składa się z formuł Hornowskich. T tłumaczymy na zbiór klauzul $K(T)$.
- Zapytanie do bazy wiedzy ma postać $P(X)$.
- Prolog stara się znaleźć takie podstawienie t za zmienną X, że $T \vdash P(t)$ (term t może zawierać zmienne).
- Tworzymy klauzulę $\{\neg P(X)\}$ i staramy się wykazać sprzeczność $K(T) \cup \{K(T)\}$ przy pomocy SLD rezolucji.

SLD rezolucja a Prolog

- $\{\neg P(X)\}$ to jedyna negatywna klauzula w $K(T) \cup \{K(T)\}$.
- Prolog próbuje znaleźć formułę w T postaci

$$\forall \bar{x}(\varphi_1 \wedge \cdots \wedge \varphi_n \rightarrow P(t))$$

  i podstawienie θ (w tym przypadku $X \mapsto t$) takie, że $P(X)\theta = P(t)\theta$.
- Po wykonaniu rezolucji otrzymujemy negatywną klauzulę postaci

$$\{\varphi_1\theta, \ldots, \varphi_n\theta\}.$$

- Wybieramy jedno $\varphi_i\theta$ i szukamy formuły w T

$$\forall \bar{x}(\psi_1 \wedge \cdots \wedge \psi n \rightarrow \psi)$$

  i podstawienie τ takich, że $\psi\tau = \varphi_i\theta\tau$ (wcześniej urozłączniamy zmienne).

SLD rezolucja a Prolog

- Obliczenie Prologu kończy się sukcesem, jeśli zostanie otrzymana pusta klauzula.
- Podstawienie σ złożone z kolejnych podstawień podczas obliczenia podaje nam świadka $t = \sigma(X)$, dla którego T i $\neg P(t)$ są sprzeczne.
- Wobec tego $T \vdash P(t)$ a t jest rozwiązaniem, które obliczył Prolog.

Twierdzenie 45 (O pełności)

Niech $\mathcal{K}$ zbiór klauzul Hornowskich w języku logiki pierwszego rzędu. $\mathcal{K}$ jest sprzeczny (semantycznie) wtedy, i tylko wtedy, gdy $\mathcal{K} \vdash_{SLD} \emptyset$.

Wniosek 46

Niech T zbiór formuł Hornowskich, niech φ formuła atomowa. $T \models \varphi$ wtedy, i tylko wtedy gdy $K(T) \cup \{\neg\varphi\} \vdash_{SLD} \emptyset$.

Twierdzenie 47

Problem sprzeczności dla zbiórów klauzul Hornowskich jest nierozstrzygalny.

SLD rezolucja a Prolog – przykład

Rozważmy bazę danych T:

- $\forall x Add(x, 0, x)$,
- $\forall x \forall y \forall z (Add(x, y, z) \rightarrow Add(x, s(y), s(z)))$.

Czy $T \vdash \exists x Add(s(0), s(s(0)), x)$?

Dodajemy do T negację powyższej formuły czyli $\forall x \neg Add(s(0), s(s(0)), x)$.

Otrzymujemy zbiór klauzul:

- $K_1 = \{Add(x, 0, x)\}$,
- $K_2 = \{\neg Add(x, y, z), Add(x, s(y), s(z))\}$,
- $K_3 = \{\neg Add(s(0), s(s(0)), x)\}$.

- $K_1 = \{Add(x, 0, x)\}$,
- $K_2 = \{\neg Add(x, y, z), Add(x, s(y), s(z))\}$,
- $K_3 = \{\neg Add(s(0), s(s(0)), x)\}$.

Wyprowadzamy

- $\tau_1 = \{x \mapsto x_1\}$, $\theta_1 = \{y \mapsto s(0), x \mapsto s(0), x_1 \mapsto s(z)\}$
  $K_3\tau_1 = \{\neg Add(s(0), s(s(0)), x_1)\}$,
  $K_3\tau_1\theta_1 = \{\neg Add(s(0), s(s(0)), s(z))\}$,
  $K_2\theta_1 = \{\neg Add(s(0), s(0), z), Add(s(0), s(s(0)), s(z))\}$,
  $K_4 = Res(K_3\tau_1, K_2, \ldots, \theta_1) = \{\neg Add(s(0), s(0), z)\}$.

- $K_1 = \{Add(x, 0, x)\}$,
- $K_2 = \{\neg Add(x, y, z), Add(x, s(y), s(z))\}$,
- $K_3 = \{\neg Add(s(0), s(s(0)), x)\}$,
- $K_4 = \{\neg Add(s(0), s(0), z)\}$.

Wyprowadzamy

- $\tau_2 = \{z \mapsto z_2\}$, $\theta_2 = \{z_2 \mapsto s(z), y \mapsto 0, x \mapsto s(0)\}$
  $K_2\theta_2 = \{\neg Add(s(0), 0, z), Add(s(0), s(0), s(z))\}$,
  $K_4\tau\theta_2 = \{\neg Add(s(0), s(0), s(z))\}$
  $K_5 = Res(K_4\tau_2, K_2, \ldots, \theta_2) = \{\neg Add(s(0), 0, z)\}$.

- $K_1 = \{Add(x, 0, x)\}$,
- $K_2 = \{\neg Add(x, y, z), Add(x, s(y), s(z))\}$,
- $K_3 = \{\neg Add(s(0), s(s(0)), x)\}$,
- $K_4 = \{\neg Add(s(0), s(0), z)\}$,
- $K_5 = \{\neg Add(s(0), 0, z)\}$.

Wyprowadzamy

- $\theta_3 = \{z \mapsto x, x \mapsto s(0)\}$.
  $Res(K_1, K_5, \ldots, \theta) = \emptyset$.

- $K_1 = \{Add(x, 0, x)\}$,
- $K_2 = \{\neg Add(x, y, z), Add(x, s(y), s(z))\}$,
- $K_3 = \{\neg Add(s(0), s(s(0)), x)\}$,
- $K_4 = \{\neg Add(s(0), s(0), z)\}$,
- $K_5 = \{\neg Add(s(0), 0, z)\}$,
- $Res(K_1, K_5, \ldots, \theta) = \emptyset$.

Na co przepisaliśmy zmienną x z K_3?

- $\tau_1 = \{x \mapsto x_1\}$, $\theta_1 = \{y \mapsto s(0), x \mapsto s(0), x_1 \mapsto s(z)\}$,
- $\tau_2 = \{z \mapsto z_2\}$, $\theta_2 = \{z_2 \mapsto s(z), y \mapsto 0, x \mapsto s(0)\}$,
- $\theta_3 = \{z \mapsto x, x \mapsto s(0)\}$.

$x, x_1, s(z), s(z_2), s(s(z)), s(s(x)), s(s(s(0)))$.
Wniosek. $Add(s(0), s(s(0)), s(s(s(0))))$ czyli $1 + 2 = 3$!

Dlaczego $Add(s(0), s(s(0)), s(s(s(0))))$?

Z klauzulami K_1, K_2 sprzeczna jest nie tylko K_3 ale także jej podstawienie

$$\{\neg Add(s(0), s(s(0)), s(s(s(0))))\}$$

czyli

$$K_3\tau_1\theta_1\tau_2\theta_2\theta_3.$$

Rysunek na tablicy, tłumaczący tę sytuację.
Odwołanie do MGU.

Outline

Jak to robimy w Prologu?

Baza wiedzy:

- add(X,zero,X).
- add(X,s(Y),s(Z)):-add(X,Y,Z).

Zapytanie ma postać

$$:-add(s(zero), s(s(zero)), X).$$

Odpowiedź:

$$X = s(s(s(zero)))$$

Jak to robimy w Prologu?

Baza wiedzy:

- add(X,zero,X). czyli $\forall x add(x, 0, x)$.
- add(X,s(Y),s(Z)):-add(X,Y,Z). czyli $\forall x \forall y \forall z (add(x, y, z) \rightarrow add(x, s(y), s(z))$.

Zapytanie ma postać

$$: -add(s(zero), s(s(zero)), X).$$

Odpowiada to dodaniu do teorii T zdania $\forall x \neg add(s(0), s(s(0)), x)$.

Odpowiedź:

$$X = s(s(s(zero))), czyli T \vdash add(s(0), s(s(0)), s(s(s(0)))).$$

Prolog – składnia

- Stałe
  - atomy – zaczynają się od małej litery: zero, ala, slonce, jan_kowalski, ...
  - dowolne ciągi znaków w apostrofach: 'Ala', ...
  - stałe liczbowe: 20, 1.23, -1, 4.32e4, 3.45e-2, ...
- zmienne – zaczynają się dużą literą lub podkreśleniem: X, Y, Z, Ala, _X, _Y, ...
- zmienna specjalna: podkreślenie _, używamy jej, kiedy nazwa zmiennej nie ma znaczenia.
- Predykaty i symbole funkcyjne zaczynają się w Prologu małą literą, jak stałe.
- atomy specjalne
  -
  - :- – operator ..., odpowiada implikacji,
  - ?- – „symbol zachęty".
- symbole funkcyjne i predykaty –
- struktury

Koniec