Logika dla informatyków

Andrzej M. Borzyszkowski

Instytut Informatyki
Uniwersytet Gdański

inf.ug.edu.pl/~amb

Semantyka rachunku predykatów

Modele

- semantyka rachunku zdań startowała z wartościowania formuł atomowych
- w rachunku predykatów te formuły mają swoją strukturę – predykaty dotyczące termów

Definicja

Model rachunku predykatów nad sygnaturą $\mathcal{F}$,$\mathcal{P}$ składa się z:

- *niepusty zbiór A elementów*
- *dla każdego symbolu funkcyjnego $f \in \mathcal{F}$ funkcja f^A o właściwej liczbie argumentów (dotyczy to też stałych)*
- *dla każdego symbolu predykatu $P \in \mathcal{P}$ predykat P^A o odpowiedniej liczbie argumentów (tzn. podzbiór A^k)*

Modele i interpretacja termów

- w modelu $\mathcal{M} = \left(A, \left\{f^A \mid f \in \mathcal{F}\right\}, \left\{P^A \mid P \in \mathcal{P}\right\}\right)$ interpretacja stałych i symboli funkcyjnych daje semantykę termów nie zawierających zmiennych
- fragment definicji interpretacji termów
  - $[\![c]\!]^A = c^A$
  - $[\![f(t_1, \ldots, t_n)]\!]^A = f^A([\![t_1]\!]^A, \ldots, [\![t_n]\!]^A)$
- przykłady
  - $A = \mathbb{N}$, $[\![0]\!]^A = 0$, $[\![t_1 + t_2]\!]^A = [\![t_1]\!]^A + [\![t_2]\!]^A$, itd.
  - $A = \mathcal{P}(U)$, $[\![\emptyset]\!]^A = \emptyset$, $[\![t_1 \cup t_2]\!]^A = [\![t_1]\!]^A \cup [\![t_2]\!]^A$, $[\![t^c]\!]^A = U \setminus [\![t]\!]^A$ itd.
  - $A = \{0, 1\}$, $[\![\bot]\!]^A = 0$, $[\![\neg t]\!]^A = 1 - [\![t]\!]^A$, $[\![t_1 \vee t_2]\!]^A = \max([\![t_1]\!]^A, [\![t_2]\!]^A)$ itd.

Środowiska i interpretacja termów

Definicja

Środowiskiem dla modelu $\mathcal{M}$ o nośniku A nazywamy funkcję częściową $\ell : Var \rightharpoonup A$ o skończonej dziedzinie

- interpretacja zmiennych w modelu jest zadana przez środowisko, środowisko pozwala na interpretację dowolnych termów, również zawierających zmienne
- $[\![x]\!]^A_\ell = \ell(x)$ gdzie x musi być w dziedzinie ℓ
- $[\![f(t_1, \ldots, t_n)]\!]^A_\ell = f^A([\![t_1]\!]^A_\ell, \ldots, [\![t_n]\!]^A_\ell)$

Semantyka formuł

- definiowana jest poprzez relację o trzech argumentach: model $\mathcal{M}$, formuła φ oraz środowisko ℓ, zapis $\mathcal{M} \models_\ell \varphi$
- będziemy zakładać, że równość w modelu $M = \left(A, \left\{f^A \mid f \in \mathcal{F}\right\}, \left\{P^A \mid P \in \mathcal{P}\right\}\right)$ ma ustaloną interpretację: „prawdziwą" równość $\{(x, x) \mid x \in A\}$.
- środowisko ℓ musi być określone dla każdej zmiennej wolnej formuły φ
- okaże się, że nie jest istotne czy i jak określone jest środowisko dla zmiennych innych niż zmienne wolne formuły φ

Semantyka formuł, definicja

Definicja

Dany model $\mathcal{M}$ i interpretacja zmiennych $\ell : Var \rightharpoonup A$.
Semantyka formuł określona jest następująco:

- $\mathcal{M} \models_\ell P(t_1, \ldots, t_n)$ *w.t.w. w modelu zachodzi* $P^A([\![t_1]\!]^A_\ell, \ldots, [\![t_n]\!]^A_\ell)$
- $\mathcal{M} \models_\ell \forall\, x.\varphi$ *w.t.w.* $\mathcal{M} \models_{\ell[x \mapsto a]} \varphi$ *dla wszystkich elementów modelu,* $a \in A$,
- $\mathcal{M} \models_\ell \exists\, x.\varphi$ *w.t.w.* $\mathcal{M} \models_{\ell[x \mapsto a]} \varphi$ *dla pewnego* $a \in A$,
- $\mathcal{M} \models_\ell \varphi \wedge \psi$ *w.t.w.* $\mathcal{M} \models_\ell \varphi$ *oraz* $\mathcal{M} \models_\ell \psi$
- *i dla pozostałych spójników podobnie jak w rachunku zdań*

$\ell[x \mapsto a]$ oznacza środowisko, w którym zmiennej x przypisano wartość a, a pozostałe zmienne są określone tak jak w ℓ, ℓ może być dla x określone lub nie, oryginalna wartość jest utracona

Interpretacja termów i semantyka formuł

Twierdzenie

- $[\![t_1[t/x]]\!]^A_\ell = [\![t_1]\!]^A_{\ell[x \mapsto [\![t]\!]^A_\ell]}$
- *Jeśli term t jest wolny dla x w formule φ, to* $\mathcal{M} \models_\ell \varphi[t/x]$ *w.t.w.* $\mathcal{M} \models_{\ell[x \mapsto [\![t]\!]^A_\ell]} \varphi$

Własności te pozwolą uzasadnić poprawność reguł wprowadzanie i eliminacji kwantyfikatorów.

Twierdzenie

Jeśli ℓ i m są identycznie określone dla zmiennych wolnych formuły φ, to $\mathcal{M} \models_\ell \varphi$ *w.t.w.* $\mathcal{M} \models_m \varphi$
W szczególności, dla formuły zamkniętej interpretacja w modelu nie wymaga w ogóle środowiska

Implikacja semantyczna

$\varphi_1, \ldots, \varphi_n, \psi$ są formułami nad sygnaturą $(\mathcal{F}, \mathcal{P})$

Definicja

- $\varphi_1, \ldots, \varphi_n \models \psi$ *jeśli dla wszystkich modeli* $\mathcal{M}$ *i środowisk* ℓ *zachodzi implikacja*

$$\mathcal{M} \models_\ell \psi \text{ o ile tylko } \mathcal{M} \models_\ell \varphi_i \text{ dla } i = 1, \ldots, n$$

- ψ *jest prawdziwa jeśli* $\mathcal{M} \models_\ell \psi$ *dla każdego modelu i każdego środowiska*
- ψ *jest spełnialna jeśli* $\mathcal{M} \models_\ell \psi$ *dla pewnego modelu i pewnego środowiska*
- $\varphi_1, \ldots, \varphi_n$ *są niesprzeczne jeśli istnieje model i środowisko takie, że* $\mathcal{M} \models_\ell \varphi_i$ *dla wszystkich* φ_i

Uwaga: modeli jest potencjalnie nieskończenie wiele

Poprawność i pełność rachunku predykatów

Definicja

- *system reguł logiki jest poprawny jeśli* $\vdash \varphi$ *pociąga* $\mathcal{M} \models \varphi$ *dla wszystkich modeli* $\mathcal{M}$
- *system reguł logiki jest pełny jeśli zachodzi odwrotna implikacja*

Twierdzenie (K. Gödel)

Klasyczny rachunek predykatów z regułami naturalnej dedukcji jest poprawny i pełny.

- poprawność każdej z reguł można łatwo sprawdzić,
- dowód pełności jest znacznie trudniejszy niż dla logiki zdaniowej.

Obliczalność/ rozstrzygalność

Obliczalność

- modele obliczeń
  - maszyny Turinga
  - lambda rachunek (używany np. w Pythonie)
  - dowolny (prawie) język programowania
- teza A. Churcha–A. Turinga: jest tylko jedno pojęcie obliczalności
- problem decyzyjny: podzbiór pewnego uniwersum (np. formuły prawdziwe) – pytamy, czy dany element należy do tego zbioru
- rozstrzygalność: program komputerowy p rozwiązujący ten problem decyzyjny

$$[\![p]\!](input) = \begin{cases} \text{TAK} & \text{o ile } input \text{ spełnia warunek} \\ \text{NIE} & \text{w p.p.} \end{cases}$$

Problem stopu

- *input* jest parą (p, d) – dowolny program i jego dane, chcemy sprawdzać, czy ten program na tych danych się zatrzyma

Twierdzenie

Problem stopu jest nierozstrzygalny.

Problem stopu c.d.

Dowód.

Niech *stop* jest superprogramem rozstrzygającym problem stopu.
Budujemy złośliwy program q, który

- czyta na wejściu jakiś program r
- wywołuje program *stop* z danymi (r, r)
- zapętla się, gdy *stop* zwraca wynik pozytywny
- co to za wartość $[\![q]\!](q)$?
- $[\![q]\!](q)$ nie da się obliczyć – paradoks kłamcy
- program q **nie może** być zdefiniowany w zależności od wyniku programu, który jeszcze nie mógł być wykonany
- tzn. program *stop* nie może istnieć

□

Jest to generyczny przykład nierozstrzygalności.

Rozstrzygalność rachunku zdań

Twierdzenie

- *klasyczny rachunek zdań jest rozstrzygalny*
- *intuicjonistyczny rachunek zdań jest rozstrzygalny*

szkic dowodu:

- rachunek klasyczny: algorytm sprawdzenia wszystkich wartościowań jest na pewno skuteczny (problemem praktycznym jest złożoność)
- rachunek intuicjonistyczny: poszukiwanie dowodu jest sterowane składnią formuł i musi skończyć się powodzeniem, jeśli formuła jest prawdziwa

Nierozstrzygalność logiki predykatów

Twierdzenie

Logika predykatów nie jest rozstrzygalna –
nie istnieje algorytm pozwalający stwierdzić czy dana formuła φ (bez zmiennych wolnych) jest twierdzeniem logiki $\vdash \varphi$.

- poprawność i pełność logiki predykatów oznacza, że $\vdash \varphi$ w.t.t. $\mathcal{M} \models \varphi$ dla wszystkich $\mathcal{M}$.
  Ale modeli jest nieskończenie wiele i wyczerpujące badanie nie wchodzi w grę.
- pytanie, czy istnieje jakiś inny algorytm?

„Dowód" nierozstrzygalności

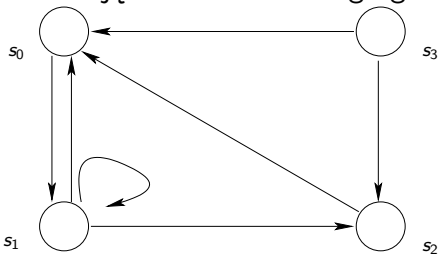
- wystarczy znaleźć redukcję problemu stopu do problemu rozstrzygalności logiki predykatów
- opisuje się dowolny program komputerowy w języku logiki predykatów
  - „zawartość komórki pamięci jest równa ...",
  - „kolejną instrukcją jest ...",
  - „jeśli zachodzi warunek, to ...",
  - „program zakończył działania i dał wynik ...", itd
- własność stopu programu p redukuje się do problemu czy formuła „program p ma własność stopu" jest prawdziwa

Niezupełność – Tw. K. Gödela

- dla każdego modelu $\mathcal{M}$ wystarczająco bogatego (np. umożliwiającego arytmetykę) i każdego (poprawnego) systemu logicznego istnieje formuła φ prawdziwa w modelu, ale niewyprowadzalna w logice: $\mathcal{M} \models \varphi$ ale $\mathcal{M} \not\vdash \varphi$
- w dowodzie kodujemy formuły i ich dowody jako liczby i konstruujemy formułę „nie można mnie dowieść"
- oczywiście nie można jej dowieść i oczywiście jest prawdziwa
- dodanie dowolnych problematycznych formuł jako aksjomatów nie może pomóc – nadal będą spełnione założenia twierdzenia i nadal będą istnieć formuły prawdziwe i niedowodliwe
- dodanie *wszystkich* prawdziwych formuł nie wchodzi w grę – w systemie logicznym musi być rozstrzygalny problem czy formuła jest aksjomatem

Wyrażalność rachunku predykatów

- rachunek predykatów nie może spełniać założeń tw. Gödela o niezupełności
- wiele problemów w informatyce ma postać grafu skierowanego (automaty, systemy tranzycji, itd)
- interesują nas własności tego grafu, np. spójność, osiągalność



- czy istnieje możliwość/niebezpieczeństwo przejścia ze stanu s_0 w stan s_3?

Nieskończoność a logika predykatów

- niech E(u,v) oznacza, że istnieje krawędź od u do v
- wówczas $u = v \vee E(u,v) \vee \exists\, w(E(u,w) \wedge E(w,v))$ oznacza, że istnieje ścieżka długości najwyżej dwa od u do v
- można zakodować predykat istnienia ścieżki ustalonej długości
- czy można zdefiniować predykat „istnieje jakaś ścieżka"?

Twierdzenie (o zwartości)

Jeśli skończone podzbiory zbioru zdań Γ mają model, to cały zbiór też ma.

1. pełność rachunku – istnienie modelu jest związane z możliwością przeprowadzenia dowodu
2. każdy dowód jest skończony i musi odwoływać się do skończonej liczby założeń

Niewyrażalność nieskończonej dyzjunkcji

- niech $\varphi_n(u, v)$ oznacza, że istnieje ścieżka długości n od u do v, $\varphi(u, v)$ – że istnieje jakaś ścieżka
- zbiór formuł $\{\neg\varphi_n \mid n = 0, 1, \dots\} \cup \{\varphi\}$ nie ma modelu – nie może jednocześnie istnieć jakaś ścieżka i żadna o konkretnej długości
- a więc istnieje (jakoby) N takie, że zbiór $\{\neg\varphi_n \mid n = 0, 1, \dots, N\} \cup \{\varphi\}$ nie ma modelu
- ale oczywiście istnieją modele, w których istnieje ścieżka od u do v, ale bardzo długa
- tzn. nie ma możliwości zdefiniowania φ powyżej

Przykład: SQL

```
create table lubi (kto varchar(11), kogo varchar(11));
insert into lubi (kto,kogo) values ('Ewa','Paweł'),
('Paweł','Ewa'), ('Paweł','Marta'), ('Marta','Ewa'),
('Tomasz','Ewa'), ('Tomasz','Marta');
-- ścieżki długości 1
select kto, kogo from lubi;
-- ścieżki długości 2
select distinct Z1.kto, Z2.kogo
from lubi Z1 join lubi Z2 on Z1.kogo=Z2.kto;
-- ścieżki długości 1 lub 2
select kto, kogo from lubi
union
select distinct Z1.kto, Z2.kogo
from lubi Z1 join lubi Z2 on Z1.kogo=Z2.kto;
```