Programowanie deklaratywne i logika obliczeniowa

Programowanie deklaratywne i logika obliczeniowa

- Wykład logika – 12 godzin
  - Dr hab. inż. Joanna Józefowska, prof. PP
    dyżur: poniedziałek 9.30 - 11.00
    p. 10, Wieniawskiego 17/19
    e-mail:
    joanna.jozefowska@cs.put. poznan.pl
  - materiały do wykładów:
    http://www.cs.put.poznan.pl/jjozefowska/
    hasło: w2005
- Laboratorium – 10 godzin
  - Dr inż. Tomasz Łukaszewski
  - Mgr inż. Agnieszka Ławrynowicz
- Sprawdzian na ostatnim wykładzie (10 punktów)
- Sprawdzian na laboratorium
- Czas pracy poza zajęciami: ok. 30 h

- Wykład programowanie – 18 godzin
  - Dr inż. Artur Michalski
    e-mail: artur.michalski@cs.put poznan.pl
- Laboratorium – 20 godzin
  - Dr inż. Artur Michalski
  - Mgr inż. Agnieszka Ławrynowicz
- Sprawdzian na ostatnim wykładzie (20 punktów)
- Zaliczenie laboratorium

Literatura

- M. Ben-Ari, Logika matematyczna w informatyce, WNT, Warszawa 2005.
- I. A. Ławrow, Ł. L. Maksimowa, Zadania z teorii mnogości, logiki matematycznej i teorii algorytmów, PWN, Warszawa 2004.
- Franz Baader, Diego Calvanese, Deborah McGuinness, Daniele Nardi, Peter Patel-Schneider (eds.), The Description Logic Handbook Theory, Implementation and Applications, Cambridge University Press, Cambridge 2003.
- A. V. Aho, J. D. Ullman, Wykłady z informatyki z przykładami w języku C, Helion, Gliwice 2003.
- R. L. Epstein, W. A. Carnelli, Computability, Wadsworth, Belmont, 2000.
- T. Batóg, Podstawy logiki, Wydawnictwo naukowe UAM, Poznań 1999.

Plan wykładu

- Powtórka: rachunek zdań i rachunek predykatów, rezolucja.
- Rachunek zdań: systemy dowodzenia (gentzenowski, hilbertowski),
- Rachunek predykatów: systemy dowodzenia, rezolucja
- Programowanie w logice: formuły jako programy, semantyka języków programowania, weryfikacja programów
- Logiki deskrypcyjne: podstawowe definicje i

Dowodzenie twierdzeń, systemy logiczne

Wykład 1

Plan wykładu

- Logika
- Dowodzenie twierdzeń
- Systemy logiczne
- System gentzenowski

Logika a informatyka

Logika stanowi matematyczne podstawy oprogramowania, jest używana do:

- formalnego definiowania semantyki języków programowania,
- tworzenia formalnych specyfikacji programów,
- weryfikacji ich poprawności.

Zasada sylogizmu

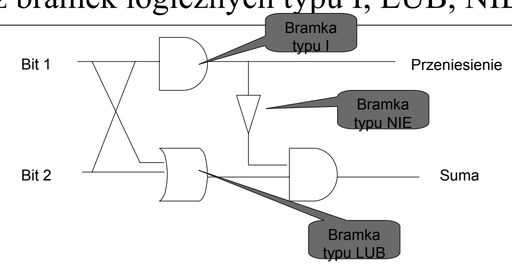
Przesłanka: Wszyscy ludzie są śmiertelni.
Przesłanka: X jest człowiekiem.
Wniosek: A zatem X jest śmiertelny.

~~**Przesłanka:** Jakieś samochody terkoczą.~~
~~**Przesłanka:** Mój samochód jest jakimś samochodem.~~
~~**Wniosek:** A zatem mój samochód terkocze.~~

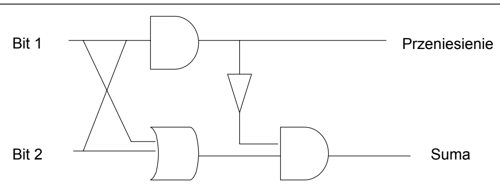
Logika matematyczna

1. Każde zdanie, które można udowodnić jest prawdziwe.
2. Jeżeli pewne zdanie jest prawdziwe, to istnieje jego dowód (być może jeszcze nieznany).

Sumator jednocyfrowy zbudowany z bramek logicznych typu I, LUB, NIE

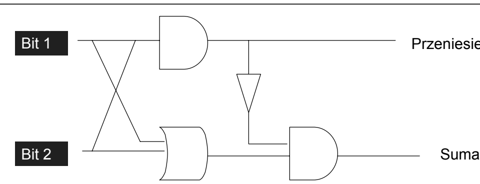


Sumator jednocyfrowy zbudowany z bramek logicznych typu I, LUB, NIE



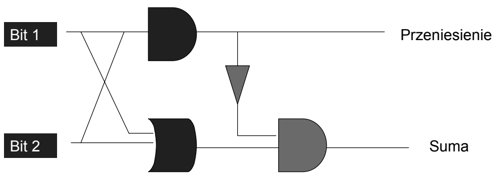
Suma $\leftrightarrow \neg$ (Bit1 $\wedge$ Bit2) $\wedge$ (Bit1 $\vee$ Bit2)
Przeniesienie $\leftrightarrow$ Bit1 $\wedge$ Bit2

Sumator jednocyfrowy zbudowany z bramek logicznych typu I, LUB, NIE



Suma $\leftrightarrow \neg$ (Bit1 $\wedge$ Bit2) $\wedge$ (Bit1 $\vee$ Bit2)
Przeniesienie $\leftrightarrow$ Bit1 $\wedge$ Bit2

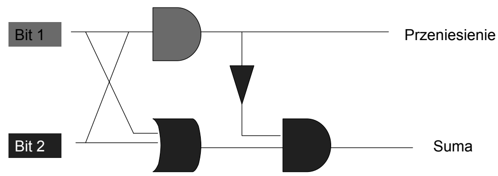
Sumator jednocyfrowy zbudowany z bramek logicznych typu I, LUB, NIE



Suma $\leftrightarrow \neg$ (Bit1 $\wedge$ Bit2) $\wedge$ (Bit1 $\vee$ Bit2)

Przeniesienie $\leftrightarrow$ Bit1 $\wedge$ Bit2

Sumator jednocyfrowy zbudowany z bramek logicznych typu I, LUB, NIE



Suma $\leftrightarrow \neg$ (Bit1 $\wedge$ Bit2) $\wedge$ (Bit1 $\vee$ Bit2)

Przeniesienie $\leftrightarrow$ Bit1 $\wedge$ Bit2

Rachunek zdań

- Wyrażenie dwuwartościowe: prawda i fałsz
  - Zmienne zdaniowe
  - Operatory logiczne
  - Reguły składniowe
  - Semantyka (interpretacja)
  - Dowód
- Komputery cyfrowe pracują z dwoma poziomami napięcia: 0 i 1

Rachunek predykatów

System logiczny dopuszczający funkcje o wartościach logicznych to rachunek predykatów lub logika pierwszego rzędu.

Wystarcza do sformalizowania algebry, arytmetyki, ...

Rachunek zdań – systemy dowodzenia

Formuła A1$\wedge$...$\wedge$An$\rightarrow$A jest prawdziwa

- Twierdzenia teorii $\mathcal{T}(\mathcal{U})$ to logiczne konsekwencje zbioru aksjomatów $\mathcal{U}$.
- $\mathcal{U} \models$ A (formuła A jest konsekwencją zbioru aksjomatów $\mathcal{U}$) w.t.w. gdy $\boxed{\models A_1 \wedge \ldots \wedge A_n \rightarrow A}$, gdzie $\mathcal{U} = \{A_1, ..., A_n\}$ jest zbiorem aksjomatów.
- $\mathcal{U} \models$ A gdy procedura decyzyjna rozwiązująca problem prawdziwości formuł udzieli dla formuły A odpowiedzi „TAK”.

Metoda tabel semantycznych

- Algorytm badania spełnialności formuł rachunku zdań.
- Zasada: należy systematycznie poszukiwać modelu.

Metoda tabel semantycznych

- Wartości zapamiętujemy w strukturze drzewa.
- Pierwotna formuła jest umieszczana w korzeniu.
- Liście zawierają zbiory literałów, które powinny być spełnione.
- Liść zawierający literały komplementarne oznaczymy x, a liść zawierający zbiór literałów spełnialnych ⊙.

Metoda tabel semantycznych

- Tworzenie tabel semantycznych nie jest jednoznaczne.
- Są dwie klasy reguł: typu α i typu β.

Reguły tworzenia tabel semantycznych

α	α_1	α_2
$\neg\neg A_1$	A_1	
$A_1 \wedge A_2$	A_1	A_2
$\neg(A_1 \vee A_2)$	$\neg A_1$	$\neg A_2$
$\neg(A_1 \rightarrow A_2)$	A_1	$\neg A_2$
$\neg(A_1 \uparrow A_2)$	A_1	A_2
$A_1 \downarrow A_2$	$\neg A_1$	$\neg A_2$
$A_1 \leftrightarrow A_2$	$A_1 \rightarrow A_2$	$A_2 \rightarrow A_1$
$\neg(A_1 \oplus A_2)$	$A_1 \rightarrow A_2$	$A_2 \rightarrow A_1$

Reguły tworzenia tabel semantycznych

α	α_1	α_2	β	β_1	β_2
$\neg\neg A_1$	A_1				
$A_1 \wedge A_2$	A_1	A_2	$\neg(B_1 \wedge B_2)$	$\neg B_1$	$\neg B_2$
$\neg(A_1 \vee A_2)$	$\neg A_1$	$\neg A_2$	$B_1 \vee B_2$	B_1	B_2
$\neg(A_1 \rightarrow A_2)$	A_1	$\neg A_2$	$B_1 \rightarrow B_2$	$\neg B_1$	B_2
$\neg(A_1 \uparrow A_2)$	A_1	A_2	$B_1 \uparrow B_2$	$\neg B_1$	$\neg B_2$
$A_1 \downarrow A_2$	$\neg A_1$	$\neg A_2$	$\neg(B_1 \downarrow B_2)$	B_1	B_2
$A_1 \leftrightarrow A_2$	$A_1 \rightarrow A_2$	$A_2 \rightarrow A_1$	$\neg(B_1 \leftrightarrow B_2)$	$\neg(B_1 \rightarrow B_2)$	$\neg(B_2 \rightarrow B_1)$
$\neg(A_1 \oplus A_2)$	$A_1 \rightarrow A_2$	$A_2 \rightarrow A_1$	$B_1 \oplus B_2$	$\neg(B_1 \rightarrow B_2)$	$\neg(B_2 \rightarrow B_1)$

Algorytm

- Jeżeli $U(l)$ (zbiór formuł w wierzchołku l) jest zbiorem literałów, to sprawdź, czy zawiera on parę literałów komplementarnych. Jeżeli tak, to oznakuj go jako domknięty x, jeżeli nie, to oznakuj go jako otwarty ⊙.
- Jeżeli $U(l)$ nie jest zbiorem literałów, to wybierz dowolną formułę z tego zbioru, niebędącą literałem.
  - Jeżeli formuła jest typu α, to utwórz nowy wierzchołek l' jako potomka wierzchołka l i umieść w nim zbiór formuł:
    $U(l')=(U(l) - \{\alpha\}) \cup \{\alpha_1, \alpha_2\}$
    (Jeżeli formuła ma postać $\neg\neg A_1$, to nie ma formuły α_2).
  - Jeżeli formuła jest typu β, to utwórz dwa nowe wierzchołki l' i l'' jako następniki wierzchołka l'. W wierzchołku l' umieść zbiór formuł:
    $U(l')=(U(l) - \{\beta\}) \cup \{\beta_1\}$
    a w wierzchołku l'' zbiór:
    $U(l')=(U(l) - \{\beta\}) \cup \{\beta_2\}$

Przykład

$p \wedge (\neg q \vee \neg p)$

α	α_1	α_2
$\neg\neg A_1$	A_1	
$A_1 \wedge A_2$	A_1	A_2
$\neg(A_1 \vee A_2)$	$\neg A_1$	$\neg A_2$
$\neg(A_1 \rightarrow A_2)$	A_1	$\neg A_2$
$\neg(A_1 \uparrow A_2)$	A_1	A_2
$A_1 \downarrow A_2$	$\neg A_1$	$\neg A_2$
$A_1 \leftrightarrow A_2$	$A_1 \rightarrow A_2$	$A_2 \rightarrow A_1$
$\neg(A_1 \oplus A_2)$	$A_1 \rightarrow A_2$	$A_2 \rightarrow A_1$

Przykład

$p \wedge (\neg q \vee \neg p)$

$\downarrow$

$p, \neg q \vee \neg p$

α	α_1	α_2
$\neg\neg A_1$	A_1	
$A_1 \wedge A_2$	A_1	A_2
$\neg(A_1 \vee A_2)$	$\neg A_1$	$\neg A_2$
$\neg(A_1 \rightarrow A_2)$	A_1	$\neg A_2$
$\neg(A_1 \uparrow A_2)$	A_1	A_2
$A_1 \downarrow A_2$	$\neg A_1$	$\neg A_2$
$A_1 \leftrightarrow A_2$	$A_1 \rightarrow A_2$	$A_2 \rightarrow A_1$
$\neg(A_1 \oplus A_2)$	$A_1 \rightarrow A_2$	$A_2 \rightarrow A_1$

Przykład

$p \wedge (\neg q \vee \neg p)$

$\downarrow$

$p, \neg q \vee \neg p$

$p, \neg q$ $p, \neg p$

$\odot$ x

β	β_1	β_2
$\neg(B_1 \wedge B_2)$	$\neg B_1$	$\neg B_2$
$B_1 \vee B_2$	B_1	B_2
$B_1 \rightarrow B_2$	$\neg B_1$	B_2
$B_1 \uparrow B_2$	$\neg B_1$	$\neg B_2$
$\neg(B_1 \downarrow B_2)$	B_1	B_2
$\neg(B_1 \leftrightarrow B_2)$	$\neg(B_1 \rightarrow B_2)$	$\neg(B_2 \rightarrow B_1)$
$B_1 \oplus B_2$	$\neg(B_1 \rightarrow B_2)$	$\neg(B_2 \rightarrow B_1)$

Tabele semantyczne

- Tabelę semantyczną, której tworzenie zakończono nazywamy zakończoną.
- Tabelę zakończoną nazywamy domkniętą, jeśli wszystkie liście są oznakowane jako domknięte.
- Jeżeli istnieje liść otwarty, to tabelę nazywamy otwartą.
- Algorytm tworzenia tabeli semantycznej zatrzymuje się.

Tabele semantyczne

- Formuła A jest niespełnialna wtw, gdy zakończona tabela $\mathcal{T}$ dla formuły A jest domknięta.
- Formuła A jest spełnialna wtw, gdy $\mathcal{T}$ jest otwarta.
- Formuła A jest prawdziwa wtw tabela semantyczna dla formuły $\neg$A jest domknięta.
- Metoda tabel semantycznych jest procedurą decyzyjną rozstrzygającą prawdziwość formuł rachunku zdań.

Problemy

- Zbiór aksjomatów może być nieskończony.
  - np. $(x=y) \rightarrow (x+1=y+1)$ (arytmetyka)
- Dla nielicznych systemów logicznych istnieją procedury decyzyjne takie, jak dla rachunku zdań.
- Procedura decyzyjna może nie dawać wglądu w związki między aksjomatami i twierdzeniem.
- Procedura decyzyjna daje tylko odpowiedzi „TAK” i „NIE”, czyli nie możemy poznać wyników pośrednich (lematów).

Rozwiązanie

W logice stosuje się podejście zwane

wyprowadzaniem formuł

polegające na wyborze reguł składniowych służących do wyprowadzania

nowych formuł ze zbioru aksjomatów.

System dowodzenia

Składa się ze zbioru aksjomatów oraz zbioru reguł dowodzenia.

Wyprowadzeniem (dowodem) w systemie dowodzenia nazywamy ciąg zbiorów formuł takich, że każda formuła należąca do jednego z tych zbiorów jest aksjomatem lub może być wyprowadzona z poprzednich formuł ciągu przy użyciu pewnej reguły dowodzenia.

Jeśli ostatnim elementem ciągu jest {A}, to A nazywamy twierdzeniem, a ciąg zbiorów formuł nazywamy dowodem formuły A.

Mówimy też, że A jest wyprowadzalne, co zapisujemy $\vdash A$

Zalety wyprowadzania

- Liczba aksjomatów może być nieskończona.
- Każdy dowód składa się ze skończonego zbioru formuł.
- Łatwo sprawdzić poprawność na podstawie składni formuł.
- Z dowodu wynika jakich aksjomatów, twierdzeń oraz reguł użyto i w jakim kroku.
- Wzorzec dowodu można przenieść na podobne dowody.
- Udowodnione twierdzenie może być wykorzystane w kolejnych dowodach.

Problemy wyprowadzania

- Nie poddaje się systematycznemu przeszukiwaniu.
- Wymaga pomysłowości, a nie „siłowego" przeszukiwania.
- Można wykorzystać heurystyki.

System gentzenowski $\mathcal{G}$

Gerhard Gentzen (24 listopada 1909 - 4 sierpnia 1945) - niemiecki matematyk, zasłużony w badaniach nad logiką i podstawami matematyki. Miał duży wpływ na powstanie systemów dowodzenia twierdzeń, tworząc między innymi system sekwentów.

Po zajęciu Pragi przez wojska radzieckie został aresztowany razem z pozostałymi profesorami niemieckiego uniwersytetu i po trzech miesiącach przebywania w tragicznych warunkach zmarł.

System gentzenowski $\mathcal{G}$

- System gentzenowski jest systemem
- A… matami są zbiory for… zawierające pa… literałów komplemen… rnych.
- Reguły dowodzenia są następujące:

Formuła typu α

Formuła typu β

$$\frac{\vdash U_1 \cup \{\alpha_1, \alpha_2\}}{\vdash U_1 \cup \{\alpha\}} \qquad \frac{\vdash U_1 \cup \{\beta_1\} \quad \vdash U_2 \cup \{\beta_2\}}{\vdash U_1 \cup U_2 \cup \{\beta\}}$$

System gentzenowski

Przesłanka

$$\frac{\vdash U_1 \cup \{\alpha_1, \alpha_2\}}{\vdash U_1 \cup \{\alpha\}}$$

Konkluzja

System gentzenowski

α	α_1	α_2
$\neg(A_1 \wedge A_2)$	$\neg A_1$	$\neg A_2$
$A_1 \vee A_2$	A_1	A_2
$A_1 \to A_2$	$\neg A_1$	A_2
$A_1 \uparrow A_2$	$\neg A_1$	$\neg A_2$
$\neg(A_1 \downarrow A_2)$	A_1	A_2
$\neg(A_1 \leftrightarrow A_2)$	$\neg(A_1 \to A_2)$	$\neg(A_2 \to A_1)$
$A_1 \oplus A_2$	$\neg(A_1 \to A_2)$	$\neg(A_2 \to A_1)$

β	β_1	β_2
$\neg\neg B_1$	B_1	
$B_1 \wedge B_2$	B_1	B_2
$\neg(B_1 \vee B_2)$	$\neg B_1$	$\neg B_2$
$\neg(B_1 \to B_2)$	B_1	$\neg B_2$
$\neg(B_1 \uparrow B_2)$	B_1	B_2
$B_1 \downarrow B_2$	$\neg B_1$	$\neg B_2$
$B_1 \leftrightarrow B_2$	$B_1 \to B_2$	$B_2 \to B_1$
$\neg(B_1 \oplus B_2)$	$B_1 \to B_2$	$B_2 \to B_1$

Przykład: $\vdash (p \vee q) \to (q \vee p)$

1. $\vdash \neg p, q, p$ — aksjomat
2. $\vdash \neg q, p, q$ — aksjomat
3. $\vdash \neg(p \vee q), q, p$ — $\beta\vee$: 1, 2
4. $\vdash \neg(p \vee q), (q \vee p)$ — $\alpha\vee$: 3
5. $\vdash \neg(p \vee q) \to (q \vee p)$ — $\alpha\to$: 4

Przykład: $\vdash (p \vee q) \to (q \vee p)$

$\neg p, q, p$ $\qquad$ $\neg q, p, q$

$\neg(p \vee q), q, p$

$\neg(p \vee q), (q \vee p)$

$\neg(p \vee q) \to (q \vee p)$

Odwrócenie do góry nogami tabeli semantycznej i zamiana znaków wszystkich formuł występujących w wierzchołkach.

Twierdzenia

Czytamy: U jest wyprowadzalne (w systemie gentzenowskim)

- Niech U będzie dowolnym zbiorem formuł, a U' zbiorem dopełnień formuł ze zbioru U. Wówczas $\boxed{\vdash U}$ wtedy i tylko wtedy, gdy istnieje domknięta tabela semantyczna dla U'.
- W systemie gentzenowskim $\vdash A$ wtw gdy istnieje domknięta tabela semantyczna dla $\neg A$.
- $\models A$ wtw gdy $\vdash A$ w systemie $\mathcal{G}$.