Rozdział 1

Rachunek zdań

Rachunek zdań jest działem logiki rozważającym prawa rządzące zdaniami logicznymi. Bada on własności spójników logicznych służących budowaniu zdań złożonych.

1.1 Pojęcie zdania

Przez zdanie rozumieć będziemy wypowiedź, która orzeka o stanie świata. Wypowiedź ta może być zgodna z faktycznym stanem i wtedy mamy do czynienia ze zdaniem prawdziwym. W przeciwnym przypadku, gdy wypowiedź nie jest zgodna ze stanem faktycznym, mamy do czynienia ze zdaniem fałszywym.

Zdaniem jest zatem tylko taka wypowiedź, co do której możemy stwierdzić jej prawdziwość lub fałszywość.

Zdaniem w sensie rachunku zdań nie jest zatem pytanie, choć oczywiście jest ono zdaniem języka naturalnego.

Ciekawym zdaniem twierdzącym języka polskiego, które nie jest zdaniem w sensie rachunku zdań jest „*To zdanie jest fałszywe*". Nie można określić jego prawdziwości, bo jeśli jest prawdziwe, to zgodnie z tym co twierdzi, jest fałszywe. Nie jest również fałszywe, bo zgodnie z tym co twierdzi, byłoby prawdziwe.

Przykłady zdań:

- *Zbiór pusty zawiera w sobie tylko zbiór pusty.* (prawda)
- *Nie ma największej liczby całkowitej.* (prawda)
- *Nie ma największej liczby całkowitej w typie* **int**[1]. (fałsz)

Ze zdań możemy budować zdania złożone łącząc je spójnikami: „i", „lub", „jeśli, to", „wtedy i tylko wtedy". Szczególnym spójnikiem jest „nieprawda, że" ponieważ nie łączy on dwóch zdań ale stoi przed jednym zdaniem.

Przykłady zdań złożonych:

[1] Typ dostępny między innymi w języku C i C++.

- *Pada deszcz* **lub** *świeci słońce.*
- **Jeśli** *pada deszcz*, **to** *ulice są mokre.*
- **Nieprawda, że** *rzeka Wisła jest dłuższa od rzeki Nil.*
- `NULL`[2] *jest pustym wskaźnikiem* **i** `null`[3] *jest pustym wskaźnikiem.*

1.2 Język formalny

Rachunek zdań posługuje się formalnym językiem, na który składają się:

1. Spójniki zdaniowe $\wedge$ (koniunkcja, czytamy „i"), $\vee$ (alternatywa, czytamy „lub"), $\rightarrow$ (implikacja, czytamy „jeśli, to"), $\neg$ (negacja, czytamy „nieprawda, że").

2. Zmienne zdaniowe $p, q, r, s, t, p_1, p_2, \ldots$ (nieskończony ale przeliczalny zbiór zmiennych).

Za pomocą spójników i zmiennych zdaniowych można budować schematy zdań. Schemat zdania mówi nam jak zbudowane jest zdanie, przy czym zmiennym zdaniowym odpowiadają dowolne zdania.

Przykłady schematów i odpowiadających im zdań:

Schemat	**Zdanie**
$p \wedge q$	Pada deszcz **i** jest pochmurno.
$p \vee q$	Jest pochmurno **lub** świeci słońce.
$\neg p$	**Nieprawda, że** świeci słońce.
$p \vee \neg p$	Jest dobrze **lub nieprawda, że** jest dobrze.
$p \rightarrow q$	**Jeżeli** pada deszcz, **to** ulice są mokre.

Prawdziwość zdań zbudowanych ze spójników zależy jedynie od prawdziwości łączonych tymi spójnikami zdań.

Na przykład jeśli wiemy, że zdanie Z_1 jest prawdziwe a zdanie Z_2 jest fałszywe, to zdanie Z_1 **i** Z_2 jest fałszywe (koniunkcja jest prawdziwa tylko wtedy, gdy oba jej człony są prawdziwe).

1.3 Stałe logiczne i wartościowanie

Oznaczmy przez stałą T wartość logiczną „prawda" a przez stałą F wartość logiczną „fałsz".

[2] Stała w języku programowania C

[3] Słowo kluczowe języka programowania Ada

Gdy mówimy, że p ma wartość T, to rozumiemy przez to, że zdanie odpowiadające zmiennej zdaniowej p jest prawdziwe. W przeciwnym przypadku p ma wartość F, co oznacza, że zdanie odpowiadające zmiennej p jest fałszywe.

Do wyliczania wartości logicznej formuł zbudowanych ze spójników, można posłużyć się poniższą tabelką:

p	q	$\neg p$	$\neg q$	$p \wedge q$	$p \vee q$	$p \to q$
F	F	T	T	F	F	T
F	T	T	F	F	T	T
T	F	F	T	F	T	F
T	T	F	F	T	T	T

Jak widać formuła $\neg p$ ma wartość logiczną do wartości formuły p. Formuła $p \wedge q$ ma wartość T wtedy, i tylko wtedy, gdy zarówno p jak i q mają wartości T. Formuła $p \vee q$ ma wartość F wtedy, i tylko wtedy, gdy zarówno p jak i q mają wartość F. Formuła $p \to q$ jest fałszywa wtedy, i tylko wtedy, gdy p jest prawdziwe a q fałszywe.

Przypisanie wartości logicznych T i F wszystkim zmiennym zdaniowym występującym w formule nazywamy wartościowaniem.

Przykłady wartościowań zmiennych p, q, r i odpowiadające im wartości logiczne formuły $p \wedge (q \vee \neg r)$:

wartościowanie	$p \wedge (q \vee \neg r)$
$p = T, q = F, r = F$	T
$p = T, q = F, r = T$	F
$p = F, q = T, r = F$	F

1.4 Spełnialność

Literami greckimi oznaczać będziemy formuły rachunku zdań.

Ciekawym zagadnieniem jest stwierdzenie czy istnieją takie przypisania wartości T i F do zmiennych występujących w formule α, że wartością logiczną formuły α jest T.

Jeśli takie przypisanie wartości istnieje, to mówimy, że formuła α jest spełnialna.

Rozpatrzmy na przykład formułę $\alpha = p \to (p \vee q)$. Ma ona wartość T w przypadku gdy poprzednik implikacji p ma wartość F (bez względu na wartość logiczną następnika implikacji $p \vee q$). Zatem formuła α jest spełniona np. przez wartościowanie $p = F, q = T$.

1.5 Tautologie

Rachunek zdań bada, między innymi, czy każde zdanie jakie odpowiada danemu schematowi jest prawdziwe. O takich schematach mówi się, że są tautologiami.

Tautologią rachunku zdań jest zatem taka formuła α, że przy każdym wartościowaniu ma ona wartość logiczną T.

Przykłady tautologii:

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

1.6 Dowodzenie tautologii

Tautologie są prawami rachunku zdań i jako takie wymagają udowodnienia. Przedstawione zostaną dwie metody dowodzenia tautologii.

1.6.1 Metoda zero-jedynkowa

Metoda zero-jedynkowa polega na utworzeniu tabelki, w której dla każdej zmiennej zdaniowej występującej w dowodzonej formule jest kolumna. Tabela taka ma tyle wierszy ile jest możliwych wartościowań. Jeśli w formule występuje n różnych zmiennych zdaniowych, to tabela będzie miała 2^n wierszy (każda zmienna może przyjąć dwie wartości więc liczba wartościowań jest równa liczbie wariacji z powtórzeniami ciągu długości n złożonego z dwóch wartości F i T).

Dowodzona formuła jest tautologią, wtedy i tylko wtedy, gdy przy każdym z 2^n wartościowań ma ona wartość logiczną T.

Rozpatrzmy dla przykładu formułę $(p \rightarrow \neg p) \rightarrow \neg p$:

p	$\neg p$	$p \rightarrow \neg p$	$(p \rightarrow \neg p) \rightarrow \neg p$
F	T	T	T
T	F	F	T

W powyższej tabeli jest jedna kolumna dla zmiennej p. W kolejnych kolumnach wymieniono podformuły składające się na dowodzoną formułę i ją samą. Są tylko dwa możliwe wartościowania $p = F$ i $p = T$ i w każdym wierszu im odpowiadającemu w ostatniej kolumnie znajduje się wartość T, co dowodzi, że formuła $(p \rightarrow \neg p) \rightarrow \neg p$ jest tautologią.

Rozpatrzmy jeszcze jeden przykład, tym razem z większą liczbą możliwych wartościowań:

p	q	r	$p \wedge q$	$p \vee r$	$(p \wedge q) \rightarrow (p \vee r)$
F	F	F	F	F	T
F	F	T	F	T	T
F	T	F	F	F	T
F	T	T	F	T	T
T	F	F	F	T	T
T	F	T	F	T	T
T	T	F	T	T	T
T	T	T	T	T	T

Jak widać w ostatniej kolumnie odpowiadającej formule $(p \wedge q) \rightarrow (p \vee r)$ są same wartości T, zatem formuła ta jest tautologią.

Główną zaletą metody zero-jedynkowej jest jej prostota. Ma jednak ona zasadniczą wadę. Otóż aby sprawdzić czy formuła zawierająca n zmiennych jest tautologią należy, w najgorszym przypadku, wypełnić 2^n wierszy tabeli. Oczywiście jeśli formuła nie jest tautologią, to możemy natrafić dużo wcześniej na wartościowanie, przy którym formuła ma wartość F ale w przypadku tautologią, trzeba przejrzeć wszystkie 2^n wierszy. Już dla $n = 30$ liczba wierszy może przekroczyć miliard.

1.6.2 Tabele semantyczne

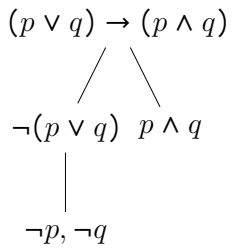
Za pomocą metody tabel semantycznych można sprawdzić czy dla danej formuły istnieje wartościowanie, przy którym ma ona wartość równą T, tzn. sprawdzić spełnialność tej formuły.

Metoda ta polega na tworzeniu drzewa, którego wierzchołkami są sekwencje formuł (ciągi formuł oddzielonych przecinkami). Wierzchołek zawierający sekwencję formuł $\alpha_1, \alpha_2, \ldots, \alpha_n$ odpowiada poszukiwaniu wartościowania, które jednocześnie spełnia formuły $\alpha_1, \alpha_2, \ldots, \alpha_n$, tzn. przy tym wartościowaniu wszystkie te formuły mają wartość logiczną T.

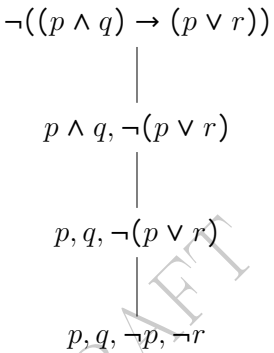
Zaprezentujemy metodę tabeli semantycznej na przykładzie badania spełnialności formuły $(p \vee q) \rightarrow (p \wedge q)$.

Na rysunku 1.1 przedstawiono konstrukcję tabeli semantycznej. W korzeniu znajduje się formuła $(p \vee q) \rightarrow (p \wedge q)$. Chcemy znaleźć dla niej wartościowanie ją spełniające. Pamiętamy, że implikacja $\alpha \rightarrow \beta$ ma wartość T jeśli formuła α ma wartość F lub formuła β ma wartość T. Zatem szukamy wartościowanie spełniające formułę $\neg(p \vee q)$, a jeśli to się nie powiedzie, to zaczniemy szukać wartościowania spełniającego formułę $p \wedge q$. W tabeli semantycznej zaznaczamy to rysując rozgałęzienie prowadzące od formuły $(p \vee q) \rightarrow (p \wedge q)$ do dwóch formuł $\neg(p \vee q)$ i $p \wedge q$.

Zajmijmy się teraz formułą $\neg(p \vee q)$. Wiemy, że formuła $\alpha \vee \beta$ ma wartość F wtedy, i tylko wtedy, gdy zarówno formuła α jak i formuła β mają wartość F. W tabeli semantycznej zaznaczamy to rysując gałąź prowadzącą od formuły $\neg(p \vee q)$



Rysunek 1.1: Tabela semantyczna dla formuły $(p \vee q) \to (p \wedge q)$



Rysunek 1.2: Tabela semantyczna dla formuły $\neg((p \wedge q) \to (p \vee r))$

do sekwencji dwóch formuł $\neg p, \neg q$ (obie te formuły muszą być spełnione jednocześnie).

Zauważmy, że wszystkie formuły znajdujące się w sekwencji $\neg p, \neg q$ są albo zmiennymi albo negacjami zmiennych. Co więcej, nie występuje w tej sekwencji żadna zmienna jednocześnie zanegowana i niezanegowana (nie ma zmiennej, która jednocześnie musiałaby mieć wartość F i T). Zatem sekwencji tej odpowiada wartościowanie $p = F, q = F$. Łatwo sprawdzić, że przy tym wartościowaniu formuła $(p \vee q) \to (p \wedge q)$ ma wartość T zatem jest spełnialna.

Jeśli umiemy zastosować metodę tabel semantycznych do badania spełnialności dowolnej formuły α, to możemy zastosować ją do sprawdzania czy formuła α jest tautologią. W tym celu zamiast stosować ją dla formuły α należy spróbować znaleść wartościowanie spełniające $\neg\alpha$. Jeśli się to powiedzie, to znaczy, że w pewnym wartościowaniu formuła α ma wartość F a zatem nie jest tautologią. Jeśli natomiast nie znajdziemy wartościowania spełniającego formułę $\neg\alpha$, to świadczyć będzie o tym, że formuła α jest tautologią.

Sprawdźmy czy formuła $(p \wedge q) \to (p \vee r)$ jest tautologią. Na rysunku 1.2 przedstawiono tabelę semantyczną dla formuły $\neg((p \wedge q) \to (p \vee r))$.

Sekwencja $p, q, \neg p, \neg r$ z ostatniego wierzchołka w drzewie zawiera zarówno

zmienną p jak i jej negację $\neg p$, zatem nie ma wartościowania odpowiadającego temu wierzchołkowi. Zbadaliśmy wszystkie możliwe wierzchołki w drzewie zatem nie ma wartościowania spełniającego formułę $\neg((p \wedge q) \rightarrow (p \vee r))$. Wynika z tego, że formuła $(p \wedge q) \rightarrow (p \vee r)$ jest tautologią.

Podsumowując:

1. Metoda tabel semantycznych służy znajdowaniu wartościowana spełniającego zadaną formułę.

2. Stosując metodę dla formuły $\neg\alpha$ można rozstrzygnąć czy formuła α jest tautologią (nie może istnieć wartościowanie spełniające formułę $\neg\alpha$, tzn. każda gałąź drzewa kończy się wierzchołkiem zawierającym w sekwencji formuł pewną zmienną i jej negację).

3. Przy dużej liczbie zmiennych, metoda tabeli semantycznej może być dużo bardziej efektywna (szybsza) niż metoda zero-jedynkowa.

1.7 Związki rachunku zdań z teorią informatyki

Zagadnienie rozstrzygania spełnialności formuły rachunku zdań odgrywa bardzo ważną rolę w informatyce.

W tym punkcie będziemy rozważać abstrakcyjne modele obliczeń, tzn. nie mamy na myśli konkretnych komputerów ale chcemy powiedzieć coś ogólnego o bardzo szerokiej klasie problemów rozwiązywanych przy użyciu klasycznych komputerów.

Na początek należy wyjaśnić pojęcie *deterministycznej* maszyny obliczeniowej. Jest nią maszyna wykonująca program, w którym dalszy tok obliczeń zależy tylko od historii obliczeń przeprowadzonych do danego miejsca. Nie ma zatem podczas jej działania takich sytuacji, że dochodzi ona do pewnego miejsca i dalsze obliczenia mogą potoczyć się na conajmniej dwa sposoby i żaden warunek możliwy do sprawdzenia przez maszynę nie determinuje, którą ze ścieżek obliczeń wybrać.

Dla przykładu, w języku C/C++ możliwe jest rozgałęzienie ścieżki obliczeń za pomocą instrukcji warunkowych (`if` i `switch`) ale to którą ścieżką obliczenia potoczą się dalej jest wyznaczone na podstawie warunku lub wyrażenia, których wartości zależą tylko od bieżącego stanu obliczeń (od bieżących wartości danych).

Podobnie w języku maszynowym możliwe jest wykonywanie skoków warunkowych (wykonają się lub nie) ale zależy to od bitów rejestru zawierającego znaczniki ustalone w sposób deterministyczny podczas wcześniejszych obliczeń.

Maszyną *niedeterministyczną* jest maszyna, podczas działania której może, ale nie musi, dochodzić do sytuacji gdy maszyna musi wybrać kolejną instrukcję programu spośród dwóch możliwych i wcześniejsze obliczenia nie determinują tego wyboru.

Możemy sobie wyobrazić, że w języku C/C++ na taką niedeterministyczną maszynę, są dodatkowe dwa słowa kluczowe `select` i `or`. Dzięki tym słowom zapisywałoby się instrukcję `select A or B`, która nakazywałaby komputerowi wybór między jedną z dwóch instrukcji `A` i `B`.

Rozpatrzmy następującą funkcję:

```
int nondet(int x)
{
  int y;
  select y = x - 1 or y = x + 1;
  return y;
}
```

Funkcja taka zwracałaby wartość o jeden mniejszą albo o jeden większą od argumentu `x` i nic nie determinowałoby, którą z tych dwóch wartości zwróci.

Należy zauważyć, że nie jest to zwrócenie jednej z dwóch wylosowanych wartości. Raczej można założyć, że pewna dobra wróżka[4] podpowiada maszynie, którą z dwóch instrukcji wykonać.

Co mają wspólnego maszyny niedeterministyczne i spełnialność formuł zdaniowych? Otóż maszyna niedeterministyczna mogłaby za pomocą niedeterministycznych wyborów odgadywać wartościowanie a następnie bardzo szybko (w czasie liniowo zależnym od długości formuły) sprawdzać czy odgadnięte wartościowanie faktycznie spełnia formułę (czy ma ona wartość T w tym wartościowaniu).

Oto program pseudokodzie na maszynę niedeterministyczną, który w czasie liniowym w stosunku do długości formuły rozstrzyga czy formuła α jest spełnialna:

```
for all variable X in formula Alpha do
  select X := T or X := F;

if Alpha then
  formuła jest spełnialna
else
  formuła nie jest spełnialna
```

Informatycy symbolem $\mathcal{P}$ oznaczają klasę wszystkich problemów decyzyjnych, dla których odpowiedź **TAK** albo **NIE** maszyna deterministyczna znajduje w czasie wielomianowym w stosunku do rozmiaru danych tego problemu.

[4] Pomysł z wróżką nie jest tak idiotyczny jakim początkowo się wydaje. Komputery kwantowe zamiast wróżki starają się w obliczeniach uwzględnić wszystkie takie rozgałęzienia na jakie natrafiły w trakcie obliczeń. Można sobie wyobrazić, że komputer kwantowy wykonując instrukcję `select y = x - 1 or y = x + 1` podstawiłby pod zmienną `y` wartość będącą jednocześnie o jeden mniejszą od `x` i o jeden większą od `x`. Do zrozumienia tego trzeba zainteresować się fizyką kwantową.

Przykładem problemu z klasy $\mathcal{P}$ jest np. znalezienie odpowiedzi na pytanie, czy można dojechać z Jeleniej Góry do Warszawy po trasie krótszej niż 470 km.

Z kolei symbolem $\mathcal{NP}$ oznaczamy klasę wszystkich problemów decyzyjnych, dla których odpowiedź **TAK** albo **NIE** maszyna niedeterministyczna znajduje w czasie wielomianowym w stosunku do rozmiaru danych tego problemu[5].

Jedyne czego jesteśmy pewni, to że $\mathcal{P} \subseteq \mathcal{NP}$. Sformułowano jednak hipotezę, że

$$\mathcal{P} \neq \mathcal{NP}.$$

Według tej hipotezy maszyny niedeterministyczne potrafią w czasie wielomianowym rozwiązać takie problemy decyzyjne, których maszyny deterministyczne nie potrafią rozwiązać w czasie wielomianowym.

Problem rozstrzygnięcia spełnialności formuł rachunku zdań, oznaczany skrótem SAT, należy do klasy problemów $\mathcal{NP}$. Innym przykładem problemów z tej klasy jest decyzyjny problem komiwojażera, w którym poszukujemy odpowiedzi na pytanie czy można objechać zadane miasta (przejeżdżając przez każde miasto dokładnie raz) i wrócić na początek trasy pokonując odległość krótszą od podanego limitu.

Problem SAT ma zaskakującą własność. Otóż każdy problem decyzyjny $P \in \mathcal{NP}$ można w czasie wielomianowym sprowadzić deterministycznie do rozstrzygnięcia problemu SAT dla pewnej formuły α zależnej od problemu P (można na podstawie problemu P zbudować w czasie wielomianowym tę formułę). Zatem gdybyśmy umieli w czasie wielomianowym na maszynie deterministycznej rozstrzygać problem SAT, to umielibyśmy w czasie wielomianowym rozwiązywać na maszynie deterministycznej każdy problem z klasy $\mathcal{NP}$. Obalilibyśmy w ten sposób hipotezę $\mathcal{P} \neq \mathcal{NP}$.

[5] Skrót NP jest od **Nondeterministic Polynomial**. Nie jest zatem prawdą, jak można napotkać w wielu publikacjach, że oznacza on problemy niewielomianowe.