# Jak wnioskują maszyny

**Andrzej Szałas**

Instytut Informatyki, Uniwersytet Warszawski

andrzej.szalas@mimuw.edu.pl

**Streszczenie**

Wykład jest poświęcony wprowadzeniu do logiki z perspektywy jej zastosowań w informatyce i sztucznej inteligencji. Poruszane treści obejmują następujące zagadnienia:

- wprowadzenie do logiki jako nauki o modelowaniu świata rzeczywistego i wnioskowaniu o nim;
- klasyczny rachunek zdań (składnia, semantyka spójników logicznych);
- odniesienie do zbiorów i użycie diagramów Venna, jako metody wnioskowania;
- wyszukiwanie a wnioskowanie na przykładzie wyszukiwarki internetowej Google;
- automatyczne wnioskowanie (informacja o metodzie rezolucji dla rachunku zdań).

Wszystkie zagadnienia są ilustrowane przykładami, w tym związanymi z robotyką i sztuczną inteligencją.

Od słuchaczy nie wymaga się żadnej wstępnej wiedzy z zakresu logiki i matematyki.

**Spis treści**





## 1 WPROWADZENIE

Działalność człowieka – począwszy od tej codziennej po najbardziej zaawansowane badania i konstrukcje – zaczyna się od rozpoznawania odpowiedniego zestawu zjawisk, a następnie ich modelowania, po to by uzyskany i sprawdzony model później wykorzystywać. Gdy poruszamy się po własnym mieszkaniu, posługujemy się jego modelem stworzonym w czasie rozpoznawania tegoż mieszkania. Model ten mamy zwykle w głowie, niemniej jednak w porównaniu z rzeczywistością jest on bardzo uproszczony. Nie bierze pod uwagę przepływu cząstek powietrza, zachowania elektronów w poszczególnych cząstkach występujących w stropie i ścianach, nie dbamy o skomplikowane procesy przepływu wody w rurach wodociągowych itd. Prostota schematu pozwala jednak na skuteczne wnioskowanie, poruszanie się po pomieszczeniach i działanie. W informatyce też dba się o to, by dla danego zjawiska czy problemu obliczeniowego wybrać możliwie jak najprostszy, ale zarazem skuteczny model.

Wyobraźmy sobie zadanie polegające na opracowaniu bardzo prostego robota przemysłowego, który „obserwuje" taśmę produkcyjną i którego zadaniem jest przestawianie z niej przedmiotów zielonych na taśmę znajdującą się po lewej stronie, a czerwonych – na taśmę znajdującą się po prawej stronie. Tworzymy więc model – trzy taśmy produkcyjne. Nad środkową taśmą czuwa robot wyposażony w:

- czujniki rozpoznające kolor zielony i czerwony,
- chwytaki służące do chwytania przedmiotów i przestawiania ich na lewą lub prawą taśmę.

Mając ten model możemy teraz opisać działanie robota dwoma prostymi regułami:

- jeśli obserwowany obiekt jest zielony, to przenieś go na taśmę z lewej strony,
- jeśli obserwowany obiekt jest czerwony, to przenieś go na taśmę z prawej strony.

Powyższe reguły nie gwarantują przeniesienia wszystkich przedmiotów zielonych na lewą stronę, a czerwonych na prawą, bo zależy to od prędkości taśm, akcji rozpoznawania koloru i akcji przenoszenia przedmiotów. Zaczynamy jednak już mieć do czynienia z logiką. Przytoczone dwie reguły odzwierciedlają bardzo proste wnioskowanie. W opisanym przypadku niekoniecznie skuteczne, ale za to skuteczne i wystarczające w innym modelu. Mianowicie, jeśli założymy, że robot nie myli się w rozpoznawaniu kolorów i niezawodnie przenosi obiekty oraz że środkowa taśma zatrzymuje się na czas rozpoznawania koloru i przenoszenia przedmiotu przez robota, a na dodatek zatrzymuje każdy przedmiot w zasięgu czujników i chwytaków robota – to takie proste wnioskowanie będzie skuteczne i można je formalnie wykazać. Daje to wiedzę o warunkach, jakie powinno spełniać środowisko robota, by ten mógł działać skutecznie wykorzystując swoje możliwości.

Możemy więc zauważyć, że skuteczność wnioskowań zależy od przyjętego modelu rzeczywistości. I znów zauważmy, że omawiane modele biorą pod uwagę jedynie to, co niezbędne dla skutecznego rozwiązania problemu, zaniedbując to, co z punktu widzenia tej skuteczności nie jest wymagane.

Aby modelować rzeczywistość zwykle zaczynamy od identyfikacji przedmiotów (**obiektów**), rodzajów obiektów (**pojęć**), ich cech (**atrybutów**) i związków między nimi. Tak postępuje się np. w projektowaniu relacyjnych baz danych (jak Access, Oracle, MySQL itp.). Każda baza danych jest modelem pewnej rzeczywistości, a wyniki zapytań kierowanych do baz danych – uzyskanymi informacjami, prawdziwymi w tej rzeczywistości. Podobnie postępuje się w wielu innych obszarach informatyki, w tym choćby w projektowaniu obiektowym, niesłychanie ważnym we współczesnych systemach.

Załóżmy teraz, że zidentyfikowaliśmy dwa rodzaje owoców: *cytryny* i *figi*. W zależności od potrzeb wynikających z rozwiązywanego zadania możemy określać ich atrybuty, jak rodzaj, kolor, smak itp. Powstaje w ten sposób pewna baza danych, zilustrowana w tabeli 1 (oczywiście różne rodzaje obiektów mogą mieć różne atrybuty, co prowadzi do wielu tabel – na przykład, gdyby wśród obiektów występowali ludzie, atrybut *smak* mógłby mieć w ich przypadku mało sensu).

*Tabela 1.*
Przykładowe obiekty i ich atrybuty

| | atrybuty | | |
|---|---|---|---|
| obiekt | *rodzaj* | *smak* | *kolor* |
| o1 | *cytryna* | *kwaśny* | *żółty* |
| o2 | *figa* | *słodki* | *brązowy* |
| o3 | *cytryna* | *kwaśny* | *zielony* |

Tabelę 1 możemy przekształcić w tabelę 2, w której atrybutami stają się wartości rodzaju, smaku, koloru, zaś wartościami 0 i 1, gdzie 0 oznacza fałsz, zaś 1 – prawdę:

*Tabela 2.*
Obiekty i ich atrybuty jako wartości

| | atrybuty | | | | | | |
|---|---|---|---|---|---|---|---|
| obiekt | *cytryna* | *figa* | *kwaśny* | *słodki* | *żółty* | *brązowy* | *zielony* |
| o1 | 1 | 0 | 1 | 0 | 1 | 0 | 0 |
| o2 | 0 | 1 | 0 | 1 | 0 | 1 | 0 |
| o3 | 1 | 0 | 1 | 0 | 0 | 0 | 1 |

Zapytanie *cytryna* AND *żółty* wybierze w wyniku obiekt o1, gdyż tylko on jest jednocześnie *cytryną* i jest *żółty*. Zapytanie *cytryna* OR *żółty* wybierze obiekty o1, o3, bowiem wybieramy będące *cytryną* lub mające kolor *żółty*. Natomiast zapytanie ¬*cytryna* wybierze obiekt o2, bowiem symbol '¬' oznacza negację (zaprzeczenie).

Możemy też określić związki między zidentyfikowanymi pojęciami, np:

jeśli dany obiekt jest *cytryną*, to jest *kwaśny* i nie jest *figą*
jeśli dany obiekt jest *żółty*, to jest *cytryną*
jeśli dany obiekt jest *figą*, to nie jest *cytryną*

Znów mamy do czynienia z pewnymi wyrażeniami typu „jeśli ..., to ...", stosowanymi we wnioskowaniu o rzeczywistości złożonej z *cytryn* i *fig*. Dodatkowo takie związki często mogą służyć do uproszczenia tabeli. Na przykład, mając powyższe zależności można opuścić kolumnę *figa*, gdyż jest ona zdefiniowana jako ¬*cytryna* (dlaczego?), a więc występujące w niej wartości można obliczyć na podstawie wartości z kolumny *cytryna*.

Badaniem metod wnioskowania zajmuje się **logika** (od greckiego słowa *logos*, oznaczającego rozum, słowo, myśl). Z jednej strony analizuje się w niej poprawność wnioskowań, a z drugiej strony – dostarcza metod i algorytmów wnioskowania. Korzenie logiki sięgają starożytnej Grecji, ale też Chin czy Indii. Odgrywała istotną rolę w średniowieczu, burzliwy jej rozwój datuje się od końca XIX wieku. George Boole, Charles Sanders Peirce, John Venn, potem Bertrand Russell czy Gottlob Frege są wybitnymi logikami z tego okresu. Pojęcie **obliczalności** jest też ściśle związane z badaniami logicznymi dotyczącymi rozstrzygalności teorii matematycznych, czyli szukania metod algorytmicznych dla automatycznego znajdowania dowodów twierdzeń tych teorii. Wybitnymi przedstawicielami tego kierunku są: Richard Dedekind, Giuseppe Peano, David Hilbert, Arend Heyting, Ernst Zermelo, John von Neumann, Kurt Gödel czy



Alfred Tarski. Pierwsze matematyczne modele maszyn matematycznych zawdzięczamy kontynuacji prac tych logików, prowadzonych przez Emile'a Posta, Alonza Churcha (prekursorzy języków funkcyjnych), jak również Stephena C. Kleenego, Alana M. Turinga czy Claude'a E. Shannona (prekursorzy języków imperatywnych).

## 2 WYSZUKIWANIE I WNIOSKOWANIE

Można napisać, że logika jest we wnioskowaniu wszechobecna, gdyż fałsz, prawda, wnioskowanie, model, pojęcie, związek (relacja), reguła czy spójniki (AND, OR, ¬) są podstawowymi konceptami rozważanymi w logice. Zaczęliśmy od baz danych i wyszukiwania obiektów mających interesujące nas właściwości. Przejdźmy teraz do wyszukiwarek internetowych. Internet jest ogromną bazą danych. Zasadniczą różnicą w porównaniu z tradycyjnymi bazami danych, zorganizowanymi w bardzo uporządkowany sposób, jest w Internecie ogromna różnorodność zasobów i brak ich jednolitej struktury.

Załóżmy, że interesującymi nas obiektami z Internetu są strony WWW. Wpisując w okienko wyszukiwarki Google zestaw słów, pytamy o te strony, na których występują wszystkie wypisane słowa kluczowe. Jest to tzw. **wyszukiwanie AND**. W języku polskim „and" znaczy „i". W logice taki spójnik nazywamy **koniunkcją** i często w literaturze oznaczamy symbolem ∧.

Aby koniunkcja *p* AND *q* była prawdziwa, prawdziwe muszą być oba zdania składowe: zdanie *p* i zdanie *q*. Jeśli np. wpiszemy dwa słowa: *logika informatyka,* to z punktu widzenia wyszukiwarki Google oznacza to wpisanie wyrażenia *logika* AND *informatyka*, czyli wyszukanie stron (naszych obiektów), na których występuje słowo *logika* i słowo *informatyka*. Tak naprawdę nie zawsze pojawią się oba słowa, co odbiega od logicznego rozumienia koniunkcji. Aby mieć „prawdziwą" koniunkcję powinniśmy wpisać wyrażenie + *logika* + *informatyka* (operator + umieszczony przed danym słowem oznacza, że musi ono wystąpić na wyszukanej stronie).

Co jeszcze pojawia się w Google? Twórcy tej wyszukiwarki oferują też **wyszukiwanie OR**. W języku polskim „or" to „lub", czyli logiczny spójnik **alternatywy**, w literaturze często oznaczany symbolem ∨. Aby alternatywa *p* OR *q* była prawdziwa, prawdziwe musi być co najmniej jedno ze zdań składowych: zdanie *p* lub zdanie *q* (lub oba te zdania). Na przykład wpisanie w Google wyrażenia *logika* OR *informatyka* spowoduje wyszukanie stron na których występuje słowo *logika* lub słowo *informatyka* lub oba te słowa. Spójnik OR wiąże silniej niż spójnik AND[5]. Oznacza to, że wyrażenie

*lekcja informatyka* OR *logika*

Google rozumie jako

*lekcja* AND (*informatyka* OR *logika*)

a nie jako (*lekcja* AND *informatyka*) OR *logika*. Wyszukane zostaną więc strony, na których pojawia się słowo *lekcja* oraz co najmniej jedno ze słów *informatyka* lub *logika*.

I mamy jeszcze operator negacji ¬, który umieszczony przed słowem oznacza, że *nie* może ono wystąpić na wyszukanej stronie. W logice spójnik „*nie*" nazywamy **negacją** i oznaczamy często symbolem ¬ (a w Google symbolem –). Negacja ¬*p* jest prawdziwa, gdy zdanie *p nie* jest prawdziwe. Na przykład wpisanie do wyszukiwarki wyrażenia – *logika* spowoduje wyszukanie tych stron WWW, na których nie występuje słowo *logika*. Czyli Google posługuje się logiką, interpretując wyrażenia logiczne i wyszukując zgodnie z nimi interesujące nas zasoby.

[5] Ta konwencja jest charakterystyczna dla Google. Tradycyjnie koniunkcja wiąże silniej niż alternatywa.

Ta logika to uproszczona wersja **klasycznego rachunku zdań**, zajmującego się w swoim podstawowym wariancie spójnikami logicznymi negacji, alternatywy, koniunkcji oraz **implikacji** i **równoważności**. Implikacja to wyrażenie postaci „*jeśli p to q*”, zaś równoważność to wyrażenie postaci „*p wtedy i tylko wtedy, gdy q*”. O tym dokładniej poniżej.

**Zadania**

1. Przetłumacz na język logiki zdanie określające zbiór obiektów *czerwonych* i *zielonych*, ale takich, które nie są *słodkie*.

   Przykładowe rozwiązanie: *czerwony* AND *zielony* AND -*słodki*.

   Wpisz powyższe wyrażenie do przeglądarki Google i porównaj je z wynikami wyszukiwania dla wyrażenia +*czerwony* AND +*zielony* AND -*słodki*. Zauważ dużą różnicę w liczbie wyszukanych stron. Zinterpretuj tę różnicę.
2. Napisz wyrażenie wyszukujące w Google strony WWW o *restauracjach* lub *barach* na Mazurach, ale nie zawierających słowa *Rzym*.

   *Przykładowe rozwiązanie: +Mazury -Rzym restauracja* OR *bar*.

   Jakie mogą być inne rozwiązania?

I do samodzielnego rozwiązania:

3. Przetłumacz na język logiki zdanie określające jeden z dni roboczych w tygodniu. Jakie wyrażenie określi wszystkie dni robocze?
4. Napisz wyrażenie wyszukujące w Google strony o prowadzonych kierunkach studiów z informatyki lub biologii, ale nie z matematyki i nie z filologii klasycznej.

## 3 KLASYCZNY RACHUNEK ZDAŃ

Klasyczny rachunek zdań zajmuje się badaniem prawdziwości zdań złożonych na podstawie zdań składowych i w konsekwencji – badaniem poprawności wnioskowania.

Aby wprowadzić rachunek zdań wprowadza się **zmienne zdaniowe** reprezentujące wartości logiczne *prawda*, *fałsz*, a zarazem zbiory obiektów mających cechy opisywane tymi zmiennymi (w tym ujęciu zmienne zdaniowe odpowiadają cechom, czyli atrybutom obiektów). Bardziej złożone wyrażenia (zwane **formułami**) uzyskujemy stosując **spójniki logiczne** negacji, koniunkcji, alternatywy, implikacji i równoważności. Czasem wprowadza się też inne spójniki. Tak naprawdę wszystkie możliwe spójniki można zdefiniować przy pomocy np. negacji i koniunkcji, jednak przyjęty przez nas zestaw spójników, choć z tego punktu widzenia nadmiarowy, jest z jednej strony prosty i naturalny, a z drugiej wystarczający w wielu typowych zastosowaniach.

Znaczenie (**semantykę**) spójników logicznych podaje się często przy pomocy tablic logicznych, w których w kolumnach podaje się wartości poszczególnych wyrażeń. Przyjmujemy, że wartościami tymi mogą być jedynie 0, 1; 0 – to **fałsz**, a 1 – to **prawda**, patrz tabele 3 i 4.

*Tabela 3.*
Tablica dla negacji:

| $p$ | $\neg p$ |
|---|---|
| 0 | 1 |
| 1 | 0 |



*Tabela 4.*
Tablica dla koniunkcji, alternatywy, implikacji i równoważności

| | | koniunkcja | alternatywa | implikacja | równoważność |
|---|---|---|---|---|---|
| $p$ | $q$ | $p$ AND $q$ | $p$ OR $q$ | $p \Rightarrow q$ | $p \Leftrightarrow q$ |
| 0 | 0 | 0 | 0 | 1 | 1 |
| 0 | 1 | 0 | 1 | 1 | 0 |
| 1 | 0 | 0 | 1 | 0 | 0 |
| 1 | 1 | 1 | 1 | 1 | 1 |

Na tej podstawie mamy bardzo skuteczny mechanizm sprawdzania poprawności wnioskowania dla formuł z niewielką liczbą zmiennych zdaniowych. Mianowicie konstruujemy tablice logiczne, w których w pierwszych kolumnach są zmienne zdaniowe, zaś w kolejnych – wyrażenia występujące w badanej formule ułożone w ten sposób, by wartość danego wyrażenia można było policzyć na podstawie wcześniej występujących wyrażeń. Wiersze w tabeli wypełnia się najpierw wszystkimi możliwymi układami wartości logicznych, a następnie wylicza wartości wyrażeń w kolejnych kolumnach.

Formuła nazywa się **tautologią**, jeśli przyjmuje wartość 1 (prawda) niezależnie od wartości wchodzących w jej skład zmiennych zdaniowych. Jest ona **spełnialna**, gdy przyjmuje wartość 1 co najmniej dla jednej kombinacji wartości zmiennych zdaniowych. Jeśli zawsze przyjmuje wartość 0 (fałsz), jest nazywana **kontrtautologią**.

Dla przykładu sprawdźmy jakie wartości przyjmuje formuła $\neg(p \text{ OR } q) \Rightarrow \neg p$ (tab. 5).

*Tabela 5.*
Tablica logiczna dla przykładowej formuły

| $p$ | $q$ | $p$ OR $q$ | $\neg(p \text{ OR } q)$ | $\neg p$ | $\neg(p \text{ OR } q) \Rightarrow \neg p$ |
|---|---|---|---|---|---|
| 0 | 0 | 0 | 1 | 1 | 1 |
| 0 | 1 | 1 | 0 | 1 | 1 |
| 1 | 0 | 1 | 0 | 0 | 1 |
| 1 | 1 | 1 | 0 | 0 | 1 |

Badana formuła jest zawsze prawdziwa, jest więc tautologią (każda tautologia jest również spełnialna). Formuły występujące we wcześniejszych kolumnach są spełnialne, ale nie są tautologiami.

Sprawdźmy jeszcze **zasadę dowodzenia przez doprowadzanie do sprzeczności**. Została ona odkryta już przed niemal 2,5 tysiącami lat (przypisuje się ją Zenonowi z Elei). Jednak jest ważna współcześnie. Stosuje się ją w matematyce, ale też odgrywa zasadniczą rolę we wnioskowaniu z baz danych wiedzy. Będziemy z niej korzystać przy omawianiu metody rezolucji, najpopularniejszej współczesnej metody automatycznego wnioskowania. Zasada ta mówi, że w celu wykazania implikacji $p \Rightarrow q$, zaprzeczamy $q$ i wykazujemy, że prowadzi to do fałszu. Innymi słowy, chcemy wykazać formułę:

$$(p \Rightarrow q) \Leftrightarrow ((p \text{ AND } \neg q) \Rightarrow 0).$$

Konstruujemy tablicę logiczną jak w tabeli 6.

*Tabela 6.*
Tablica logiczna dla formuły uzasadniającej zasadę dowodzenia przez sprzeczność

| $p$ | $q$ | $p \Rightarrow q$ | $\neg q$ | $p$ AND $\neg q$ | ($p$ AND $\neg q$) $\Rightarrow 0$ | ($p \Rightarrow q$) $\Leftrightarrow$ (($p$ AND $\neg q$) $\Rightarrow 0$) |
|---|---|---|---|---|---|---|
| 0 | 0 | 1 | 1 | 0 | 1 | 1 |
| 0 | 1 | 1 | 0 | 0 | 1 | 1 |
| 1 | 0 | 0 | 1 | 1 | 0 | 1 |
| 1 | 1 | 1 | 0 | 0 | 1 | 1 |

Badana formuła jest więc rzeczywiście tautologią.

W Internecie można znaleźć wiele appletów konstruujących tablice logiczne dla zadanych formuł. Przykładowy applet można znaleźć pod adresem:
http://highered.mcgraw-hill.com/sites/0072880082/student_view0/chapter1/interactive_demonstration_applet__truth_tables_.html.

**Zadania (do samodzielnego rozwiązania)**

1. Zbuduj tablice logiczne dla poniższych formuł i stwierdź, które z nich są tautologiami:
   a. ($p \Rightarrow q$) $\Leftrightarrow$ ($\neg p$ OR $q$)
   b. $\neg$($p$ OR $q$) $\Leftrightarrow$ ($\neg p$ AND $\neg q$).
2. Zapisz prawe strony równoważności tak, by były one tautologiami i sprawdź rozwiązanie używając tablic logicznych:
   a. $\neg$($p$ AND $q$) $\Leftrightarrow$.....
   b. $\neg$($p \Rightarrow q$) $\Leftrightarrow$.....

**Dlaczego metoda tablic logicznych nie jest dobra dla większych zadań?**
W praktycznych zastosowaniach często trzeba sprawdzać spełnialność formuł zawierających dużą liczbę zmiennych. Potrafi ona dochodzić do tysięcy. Załóżmy, że mamy formułę mającą 100 zmiennych. Tabela logiczna będzie więc miała $2^{100}$ wierszy (dlaczego?). Ile czasu spędziłby na obliczeniach bardzo szybki komputer, wykonujący powiedzmy $2^{34}$ operacji na sekundę (to więcej niż $10^{10}$ operacji na sekundę)? Aby wygenerować $2^{100}$ wierszy potrzebujemy więcej niż $2^{100}/2^{34}$ (= $2^{66}$) sekund, czyli więcej niż $2^{66}/60$ minut. To więcej niż $2^{66}/2^6$ (= $2^{60}$) minut i więcej niż $2^{60}/2^6$ (= $2^{54}$) godzin i więcej niż $2^{54}/2^5$ (= $2^{49}$) dób. To z kolei więcej niż $2^{40}$ lat (czyli więcej niż $10^{12}$ lat!). Wiek wszechświata szacuje się na 13-14 miliardów lat (czyli nie więcej niż $14^*10^9$ lat).

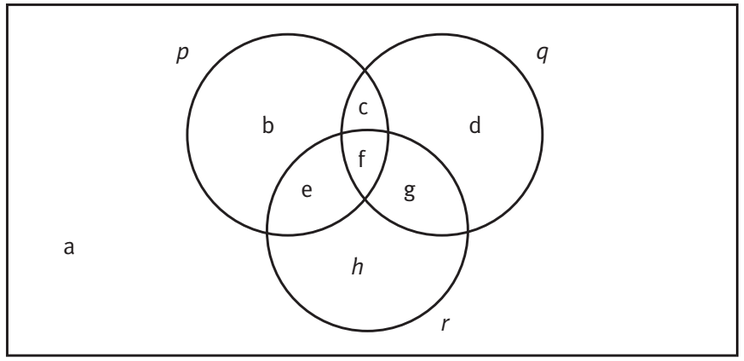
## 4 DIAGRAMY VENNA

**Diagramy Venna** ilustrują zależności pomiędzy zbiorami obiektów. Ponieważ na zmienne logiczne możemy patrzeć jako na cechy obiektów, możemy też im przypisać zbiory obiektów mających te cechy. Diagramy Venna mogą ilustrować zbiory obiektów i posługując się nimi można znajdować zależności między rozważanymi pojęciami. Zbiory na tych diagramach reprezentujemy przy pomocy kół: z każdą rozważaną zmienną zdaniową związujemy koło. Jeśli nie mamy żadnych dodatkowych założeń, umieszczamy koła w ten sposób, by wydzieliły wszystkie możliwe zależności (obszary) na danej płaszczyźnie. Obszary te oznaczamy kolejnymi literami lub liczbami. Zbiór wszystkich obiektów jest reprezentowany przez prostokąt otaczający wszystkie koła.



Diagram dla trzech zmiennych zdaniowych $p$, $q$, $r$ jest przedstawiony na rysunku 1. Na tym diagramie:

- zbiór obszarów przypisanych zmiennej $p$ to {b, c, e, f}
- zbiór obszarów przypisanych zmiennej $q$ to {c, d, f, g}
- zbiór obszarów przypisanych zmiennej $r$ to {e, f, g, h}.




*Rysunek 1.*
Diagram Venna dla trzech zbiorów

Zauważmy, że obszar oznaczony przez a reprezentuje wszystkie obiekty leżące poza kołami reprezentującymi $p$, $q$ oraz $r$.

Spójniki negacji, alternatywy i koniunkcji na diagramach Venna interpretujemy następująco:

- negacja formuły jest reprezentowana przez zbiór wszystkich obszarów niebędących obszarami reprezentującymi daną formułę; na przykład $\neg p$ reprezentujemy przez zbiór tych obszarów, które leżą poza $p$, czyli wykluczamy obszary b, c, e, f, otrzymując {a, d, g, h}
- alternatywa dwóch formuł jest reprezentowana przez zbiór obszarów reprezentujących pierwszą *lub* drugą formułę; na przykład $p$ OR $r$ reprezentujemy przez zbiór obszarów {b, c, e, f, g, h}
- koniunkcja dwóch formuł jest reprezentowana przez zbiór obszarów wspólnych dla pierwszej *i* drugiej formuły; na przykład $p$ AND $q$ reprezentujemy przez zbiór obszarów {c, f}.

Spójnik implikacji odzwierciedla zawieranie się zbiorów: $p \Rightarrow q$ oznacza, że zbiór obiektów reprezentujących $p$ zawiera się w zbiorze obiektów reprezentujących $q$.

Spójnik równoważności odzwierciedla równość zbiorów: $p \Leftrightarrow q$ oznacza, że zbiór obiektów reprezentujących $p$ jest taki sam, jak zbiór obiektów reprezentujących $q$.

**W jaki sposób wnioskujemy stosując diagramy Venna**
Diagramy Venna mogą być wykorzystane do znajdowania zależności między formułami.

Jako pierwszy przykład rozważmy formułę ($p$ AND $q$) $\Rightarrow p$. Użyjmy poprzedniego diagramu (oczywiście koło reprezentujące $r$ moglibyśmy pominąć, gdyż $r$ nie występuje w naszej formule). Już zauważyliśmy, że $p$ AND $q$ reprezentujemy przez zbiór obszarów {c, f} oraz $p$ reprezentujemy przez {b, c, e, f}. Oczywiście zbiór {c, f} zawiera się w zbiorze {b, c, e, f}, a więc badana implikacja jest prawdziwa.

Jako drugi przykład rozważmy formułę (($p$ AND $q$) OR $r$) $\Leftrightarrow$ (($p$ OR $r$) AND ($q$ OR $r$)):

- formuła $p$ AND $q$ jest reprezentowana przez zbiór {c, f}
- formuła $r$ jest reprezentowana przez zbiór {e, f, g, h}

- formuła ((*p* AND *q*) OR *r* jest więc reprezentowana przez zbiór {c, e, f, g, h}
- formuła (*p* OR *r*) jest reprezentowana przez zbiór {b, c, e, f, g, h}
- formuła (*q* OR *r*) jest reprezentowana przez zbiór {c, d, e, f, g, h}
- formuła ((*p* OR *r*) AND (*q* OR *r*)) jest więc reprezentowana przez zbiór {c, e, f, g, h},
  zatem zbiory reprezentujące lewą i prawą stronę równoważności są identyczne, co oznacza, że równoważność ta jest prawdziwa.

**Zadania (do samodzielnego rozwiązania)**

1. Zbuduj diagramy Venna dla poniższych formuł i stwierdź, które z nich są tautologiami:
   a. (*p* AND *q*) ⇔ ¬(¬*p* OR ¬*q*)
   b. ¬(*p* AND *q*) ⇔ (¬*p* OR ¬*q*).
2. Zapisz prawe strony równoważności tak, by były one tautologiami i sprawdź rozwiązanie używając tablic logicznych:
   a. ¬(*p* OR ¬*q*) ⇔.....
   b. ¬(*p* AND ¬*q*) ⇔.....

**Przykład**

Rozważmy sytuację, w której:

- robot wybiera obiekt duży lub mały: *p* OR *q*
- robot nie może wybrać jednocześnie obiektu dużego lub małego: ¬(*p* AND *q*)
- jeśli podłoże jest śliskie, to nie wybiera dużych obiektów: *r* ⇒ ¬*p*, czyli ¬*r* OR ¬*p* (dlaczego?)
- jeśli podłoże nie jest śliskie, to wybiera małe obiekty: ¬*r* ⇒ *q*, czyli *r* OR *q*.

Czy koniunkcja powyższych formuł implikuje, że zostanie wybrany mały obiekt? Innymi słowy, pytamy, czy prawdziwa jest formuła:

((*p* OR *q*) AND ¬(*p* AND *q*) AND (¬*r* OR ¬*p*) AND (*r* OR *q*)) ⇒ *q*.

Znów korzystamy z wcześniejszego diagramu:

- formuła *p* OR *q* jest reprezentowana przez {b, c, e, f, g}
- formuła *p* AND *q* jest reprezentowana przez {c, f}, a więc formuła ¬(*p* AND q) jest reprezentowana przez {a, b, d, e, g, h}
- formuła ¬*r* jest reprezentowana przez {a, b, c, d}
- formuła ¬*p* jest reprezentowana przez {a, d, g, h}, a więc formuła (¬*r* OR ¬*p*) jest reprezentowana przez {a, b, c, d, g, h}
- formuła (*r* OR *q*) jest reprezentowana przez {c, d, e, f, g, h}
- koniunkcja występująca z lewej strony implikacji jest więc reprezentowana przez część wspólną zbiorów {b, c, e, f, g}, {a, b, d, e, g, h}, {a, b, c, d, g, h},
  {c, d, e, f, g, h}, czyli przez {g}
- formuła *q* jest reprezentowana przez zbiór {c, d, f, g}, w którym {g} się zawiera, a więc odpowiedź na pytanie o prawdziwość badanej implikacji jest twierdząca.

Metoda diagramów Venna jest bardzo atrakcyjna ze względu na łatwość wizualnego odkrywania stosunkowo złożonych praw logicznych. Doczekała się wielu badań i uogólnień. Więcej na jej temat można znaleźć np. na bardzo interesującej stronie: http://www.combinatorics.org/Surveys/ds5/VennEJC.html, na której przedstawiono m.in., jak można konstruować diagramy Venna dla więcej niż trzech zbiorów, czyli w taki sposób, aby diagram jednego zbioru miał część wspólną z diagramem każdego innego zbioru.



## 5 AUTOMATYCZNE WNIOSKOWANIE

Dotychczas omawialiśmy metody wnioskowania skuteczne w niewielkich przykładach. Rzeczywiste systemy obsługujące klasyczny rachunek zdań, nazywane SAT Solvers (SAT pochodzi od angielskiego *satisfiability*, czyli **spełnialność**), potrafią sobie radzić z bardzo dużymi formułami występującymi w niemałym obszarze zastosowań (z formułami mającymi kilkaset, czasem nawet tysiące zmiennych). Ich siła polega na stosowaniu dużej liczby algorytmów skutecznych dla wybranych rodzajów formuł. Poniżej przedstawimy jeden z takich algorytmów, bardzo skuteczny i powszechnie stosowany w informatyce i sztucznej inteligencji, zwany **metodą rezolucji**. Metoda rezolucji działa na koniunkcjach klauzul, przy czym klauzula jest alternatywą zmiennych zdaniowych lub ich negacji. Na przykład klauzulą jest:

*p* OR ¬q OR ¬*r* OR *s*

zaś nie jest: *p* OR ¬¬*q* (dlaczego?) ani też *p* AND ¬*r* (dlaczego?).

**UWAGA:** pusta klauzula (niemająca żadnych wyrażeń) jest równoważna fałszowi (czyli 0).

**Dlaczego klauzule są ważne**

Wiedza w systemach sztucznej inteligencji, w tym w bazach wiedzy, systemach eksperckich itd., zwykle ma postać klauzulową. Mianowicie implikacja w postaci:

($p_1$ AND $p_2$ AND ... AND $p_k$ ) ⇒ ($r_1$ OR $r_2$ OR ... OR $r_m$)

jest równoważna klauzuli (dlaczego?):

¬$p_1$ OR ¬$p_2$ OR ...OR ¬$p_k$ OR $r_1$ OR $r_2$ OR ...OR $r_m$.

Implikacje wspomnianej postaci odzwierciedlają reguły, jakimi posługujemy się codziennie, np.:

*gorączka* AND *kaszel* ⇒ *przeziębienie* OR *grypa*
*deszcz* AND *bezwietrznie* ⇒ *parasol* OR *kurtka_z_kapturem*
*deszcz* AND *wiatr* ⇒ *samochód*

**Przekształcanie formuł do postaci klauzulowej**

Aby przekształcić formuły do **postaci klauzulowej** zamieniamy występujące w niej podformuły zgodnie z regułami podanymi poniżej aż do momentu uzyskania koniunkcji klauzul.

Reguły (jako zadanie przekonaj się, stosując metodę tablic logicznych, że formuła zastępowana jest zawsze równoważna formule zastępującej):

1. zastąp (A⇔B) przez (¬A OR B) AND (A OR¬B)
2. zastąp (A⇒B) przez (¬A OR B)
3. zastąp¬¬A przez A
4. zastąp¬(A AND B) przez (¬A OR¬B)
5. zastąp¬(A OR B) przez (¬A AND¬B)
6. zastąp A OR (B AND C) przez (A OR B) AND (A OR C)
7. zastąp (B AND C) OR A przez (A OR B) AND (A OR C).

Zakładamy, że usuwane są też zbędne nawiasy, np. ((A)) można zastąpić przez (A), zaś (A OR B) OR C – przez A OR B OR C oraz powtórzenia wyrażeń, np. (A OR A OR B) można zastąpić przez równoważną formułę (A OR B). Na przykład rozważmy formułę: (¬(*p* AND¬*q*) OR (*p* ⇒ *r*) ) OR (*r* ⇔ ¬*s*). Jej sprowadzenie do postaci klauzulowej może składać się z kroków:

- ((¬*p* OR¬¬*q*) OR (*p* ⇒ *r*) ) OR (*r* ⇔ ¬*s*)
- ((¬*p* OR¬¬*q*) OR (¬*p* OR *r*) ) OR (*r* ⇔ ¬*s*)
- (¬*p* OR¬¬*q* OR¬*p* OR *r*) OR (*r* ⇔ ¬*s*)
- (¬*p* OR *q* OR¬*p* OR *r*) OR (*r* ⇔ ¬*s*)
- (¬*p* OR *q* OR¬*p* OR *r*) OR ((¬*r* OR¬*s*) AND (*r* OR¬¬*s*))

- $(\neg p$ OR $q$ OR $\neg p$ OR $r)$ OR $((\neg r$ OR $\neg s)$ AND $(r$ OR $s))$
- $(\neg p$ OR $q$ OR $r)$ OR $((\neg r$ OR $\neg s)$ AND $(r$ OR $s))$
- $(\neg p$ OR $q$ OR $r$ OR $\neg r$ OR $\neg s)$ AND $(\neg p$ OR $q$ OR $r$ OR $r$ OR $s)$
- $(\neg p$ OR $q$ OR $r$ OR $\neg r$ OR $\neg s)$ AND $(\neg p$ OR $q$ OR $r$ OR $s)$.

Powyższą formułę można z łatwością uprościć (jak?).

**Zadania (do samodzielnego rozwiązania)**

1. Przekształć do postaci klauzulowej formuły:
   a. $(p$ AND $q) \Leftrightarrow \neg(\neg p$ OR $\neg q)$
   b. $\neg(p$ AND $q) \Leftrightarrow (\neg p$ OR $\neg q)$.
2. Zastąp uzyskane klauzule równoważnymi im implikacjami postaci rozważanej w podrozdziale „Dlaczego klauzule są ważne".

**Metoda rezolucji**

Metoda rezolucji wykorzystuje zasadę dowodzenia przez doprowadzanie do sprzeczności. Jest ona na w swojej istocie oparta na przechodniości implikacji, czyli na regule mówiącej, że z $(p \Rightarrow q)$ oraz $(q \Rightarrow r)$ mamy prawo wnioskować $(p \Rightarrow r)$. Sformułowanie tej reguły w postaci klauzulowej jest następujące:

$$(\neg p \text{ OR } q) \text{ AND } (\neg q \text{ OR } r) \Rightarrow (\neg p \text{ OR } r).$$

Uogólniając, na dowolne klauzule uzyskamy regułę rezolucji mówiącej, że z klauzul:

$$\neg p_1 \text{ OR} \neg p_2 \text{ OR} \ldots \text{OR} \neg p_k \text{ OR } r_1 \text{ OR } r_2 \text{ OR} \ldots \text{OR } r_m$$
$$p_1 \text{ OR} \neg q_1 \text{ OR} \ldots \text{OR} \neg q_n \text{ OR } s_1 \text{ OR } s_2 \text{ OR} \ldots \text{OR } s_m$$

uzyskujemy klauzulę:

$$\neg p_2 \text{ OR} \ldots \text{OR} \neg p_k \text{ OR } r_1 \text{ OR } r_2 \text{ OR} \ldots \text{OR } r_m \text{ OR} \neg q_1 \text{ OR} \ldots \text{OR} \neg q_n \text{ OR } s_1 \text{ OR } s_2 \text{ OR} \ldots \text{OR } s_m,$$

czyli usuwamy jedną ze zmiennych występującą w pierwszej klauzuli wraz z jej negacją występującą w drugiej klauzuli. Dla wygody zapisaliśmy je jako pierwsze, ale miejsce wystąpienia w klauzulach nie ma znaczenia – ważne jest tylko, by wystąpiły one w dwóch klauzulach.

Jak wspomnieliśmy, metoda rezolucji to zasada wnioskowania przez doprowadzanie do sprzeczności. Oznacza to, że najpierw negujemy sprawdzaną formułę, a następnie staramy się z tej negacji uzyskać fałsz, a więc klauzulę pustą. Jeśli się to uda, początkowa formuła jest tautologią. Jeśli pustej klauzuli nie da się w żaden sposób uzyskać, formuła tautologią nie jest.

Ważnym przykładem jest wykazywanie, że pewna formuła wynika z bazy danych wiedzy, w której wiedza jest reprezentowana jako zbiór klauzul. Chcemy sprawdzić czy $D \Rightarrow q$, gdzie $D$ jest bazą wiedzy, a $q$ jest formułą wykazywaną przy założeniu, że wiedza zawarta w $D$ odzwierciedla interesującą nas rzeczywistość. W metodzie rezolucji zaprzeczamy implikacji $(D \Rightarrow q)$. Zaprzeczenie to jest równoważne formule $(D$ AND $\neg q)$. Czyli $\neg q$ dokładamy do bazy wiedzy $D$ (przekształcając $\neg q$ do postaci klauzulowej) i staramy się wyprowadzić klauzulę pustą. Baza $D$ nie zmienia się! Z wydajnościowego punktu widzenia jest to bardzo ważne, bo zwykle taka baza danych jest duża, podczas gdy badane wnioski zwykle stosunkowo małe, gdyż reprezentują one typowe zapytania użytkowników. Zwykle zapytania mają bardzo niewielkie rozmiary w porównaniu z rozmiarem bazy danych.

Dla przykładu wykażmy, że z koniunkcji klauzul $(p \Rightarrow q)$ AND $(\neg p \Rightarrow q)$ można wywnioskować $q$. Jest to formalizacja wnioskowania przez przypadki, bo spełniony jest warunek $p$ albo warunek $\neg p$. Bez względu na to, który z nich jest spełniony, konsekwencją jest $q$. W życiu często stosujemy takie wnioskowanie. Na przykład, gdy chcemy zabezpieczyć się przed zmoknięciem, bierzemy parasol. W tej sytuacji stosujemy wnioskowanie:

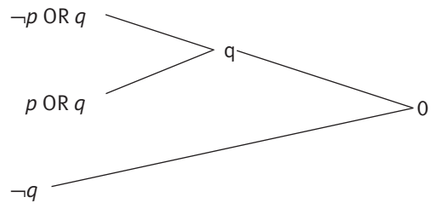
$$(\neg \textit{deszcz} \Rightarrow \neg \textit{zmoknę}) \text{ AND } (\textit{deszcz} \Rightarrow \neg \textit{zmoknę}),$$



a więc wnioskuję, że nie zmoknę bez względu na to, czy będzie deszcz, czy nie.

Zasadę wnioskowania przez przypadki można zapisać w postaci klauzulowej jako:
$(\neg p$ OR $q)$ – pierwsze założenie w postaci klauzulowej
$(p$ OR $q)$ – drugie założenie w postaci klauzulowej
$\neg$q – zaprzeczona konkluzja.

Stosując regułę rezolucji do dwóch pierwszych klauzul uzyskujemy klauzulę $(q$ OR $q)$, usuwamy zbędne powtórzenie $q$, uzyskujemy więc klauzulę zawierającą jedynie $q$. Teraz z tej klauzuli oraz z $\neg q$ uzyskujemy klauzulę pustą. Graficznie to wnioskowanie można przedstawić jak na rysunku 2.




*Rysunek 2.*
Graficzna reprezentacja wnioskowania rezolucyjnego

**Zadania (do samodzielnego rozwiązania)**

1. Korzystając z metody rezolucji wykaż, że:
   a. $((p$ OR $q)$ AND $r) \Rightarrow ((p$ AND $r)$ OR $(q$ AND $r))$
   b. $((p$ AND $q)$ OR $r) \Rightarrow ((p$ OR $r)$ AND $(q$ OR $r))$.

**Przykład**

Załóżmy, że mamy następującą bazę wiedzy:

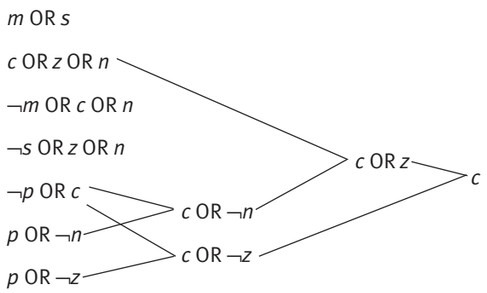
- Pudełka są małe lub średnie $(m$ OR $s)$.
- Każde pudełko jest czerwone, zielone lub niebieskie $(c$ OR $z$ OR $n)$.
- Małe pudełka są czerwone lub niebieskie $(m \Rightarrow (c$ OR $n))$.
- Średnie pudełka są zielone lub niebieskie $(s \Rightarrow (z$ OR $n))$.
- Do przewozu robot wybiera czerwone pudełka $(p \Rightarrow c)$.
- Do aktywności innych niż przewóz robot nie wybiera pudełek niebieskich ani zielonych $(\neg p \Rightarrow (\neg n$ AND $\neg z))$.

Jakie pudełko wybierze robot? Mamy bazę wiedzy zawierającą klauzule (dlaczego?):

$$(m \text{ OR } s), (c \text{ OR } z \text{ OR } n), (\neg m \text{ OR } c \text{ OR } n), (\neg s \text{ OR } z \text{ OR } n), (\neg p \text{ OR } c), (p \text{ OR} \neg n), (p \text{ OR} \neg z).$$

Przykładowe rozumowanie rezolucyjne wykorzystujące te klauzule jest przedstawione na rysunku 3. Z tego rozumowania widzimy, że uzupełnienie bazy wiedzy o klauzulę $\neg c$ dałoby natychmiastowy wywód klauzuli pustej, mamy już bowiem wyprowadzoną klauzulę zawierającą jedynie $c$. Stąd wnioskujemy, że z naszej bazy wiedzy wynika $c$ (w metodzie rezolucji taka konkluzja – po zaprzeczeniu – trafia do bazy wiedzy).

Zauważmy, że dowód rezolucyjny jest tu naprawdę bardzo krótki. Dla porównania – korzystanie z tablic logicznych wymagałoby skonstruowania $2^6 = 64$ wierszy, mamy bowiem 6 różnych zmiennych.




*Rysunek 3.*
Przykładowe wnioskowanie rezolucyjne

**Przykład**

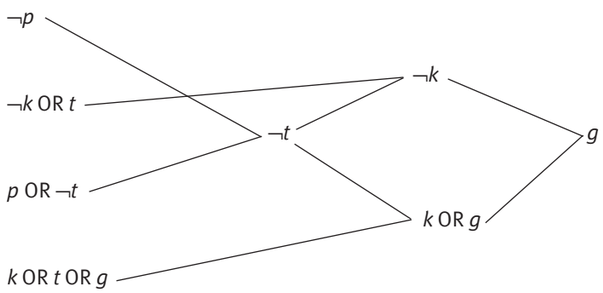
Załóżmy, że nasza baza danych zawiera formuły:

- K twierdzi, że Jan pracuje w soboty ($p$).
- K twierdzi także, że w soboty Jan czyta książki i nie ogląda telewizji ($k$ AND $\neg t$).
- P twierdzi, że gdy Jan nie pracuje, nie ogląda również telewizji ($\neg p \Rightarrow \neg$t).
- P twierdzi także, że w soboty Jan czyta książki, ogląda telewizję lub gotuje ($k$ OR $t$ OR $g$).

Wiedząc, że K zawsze kłamie, a P zawsze mówi prawdę, odgadnijmy, co Jan robi w soboty i wykażmy to stosując metodę rezolucji. Skoro K kłamie, a P mówi prawdę, mamy bazę danych klauzul (dlaczego?):

$$\neg p, (\neg k \text{ OR } t), (p \text{ OR} \neg t), (k \text{ OR } t \text{ OR } g).$$

Wnioskowanie metodą rezolucji jest przedstawione na rysunku 4.




*Rysunek 4.*
Wnioskowanie rezolucyjne dla przykładu o kłamcy i prawdomównym

Z powyższego wywodu uzyskaliśmy jako wniosek $g$, czyli fakt, że w soboty Jan gotuje. Formalne wykazanie polegałoby na dodaniu zaprzeczonej konkluzji (czyli $\neg g$) do zbioru klauzul i wyprowadzeniu klauzuli pustej, co w obecności klauzuli zawierającej jedynie $g$ jest natychmiastowe.

I znów wnioskowanie rezolucyjne jest krótkie. Oczywiście robot, stosując regułę „na ślepo", może błądzić zanim znajdzie rozwiązanie. Jednak nadal proces dowodzenia jest tu bardzo wydajny. Warto jednak podkreślić, że w pesymistycznym przypadku wymaga on $2^n$ kroków, gdzie $n$ jest liczbą zmiennych występujących w formule. Nie jest znany żaden algorytm działający efektywnie dla każdej formuły wejściowej.



## 6 I CO DALEJ?

W wykładzie pojawiła się jedna logika – klasyczny rachunek zdań. Trudno przecenić jego rolę i zakres zastosowań. Wielu naukowców poświęca całą swoją aktywność badawczą np. na poszukiwanie efektywnych systemów wnioskowania dla wybranych rodzajów formuł, pojawiających się w danych zastosowaniach w wyniku modelowania lub tłumaczenia łatwiejszych w użyciu formalizmów. W końcu jeden z kilku problemów milenijnych, za rozwiązanie którego czeka nagroda w wysokości miliona USD, dotyczy wykazania istnienia lub nieistnienia efektywnego algorytmu badania, czy dana formuła klasycznego rachunku zdań jest spełnialna. Niemniej jednak z punktu widzenia wielu ważnych zastosowań jest to bardzo prosty formalizm i – jako taki – nie jest w stanie dobrze modelować złożonej rzeczywistości systemów informatycznych, języka naturalnego, reprezentacji wiedzy czy wnioskowania o świecie rzeczywistym.

W minionych czterdziestu latach w informatyce prowadzono bardzo intensywne badania nad logikami, np. modelującymi wnioskowanie człowieka lepiej niż klasyczny rachunek zdań (znanymi w sztucznej inteligencji jako wnioskowanie zdroworozsądkowe lub niemonotoniczne). Dziedzina ta nadal intensywnie się rozwija, bowiem szuka się coraz lepszych metod modelujących inteligentne zachowania w systemach autonomicznych, jak bezzałogowe helikoptery czy złożone systemy robotyki.

Ale o tym kiedy indziej, na przykład na studiach informatycznych...

## LITERATURA

1. Ben-Ari M., *Logika matematyczna w informatyce,* WNT, Warszawa 2004
2. Shannon C.E, *A Symbolic Analysis of Rely and Switching Circuits*, M. Sc. Thesis, MIT 1938

## ZAŁĄCZNIK

Poniżej przedstawiamy krótki fragment pracy magisterskiej C.E. Shannona z 1940 roku. Uchodzi ona za najlepszą pracę magisterską XX wieku i przekłada język logiki na obwody, z jakich korzystają inżynierowie konstruujący także współczesne komputery [2, s. 11].

TABLE I

Analogue Between the Calculus of Propositions and the Symbolic Relay Analysis

| Symbol | Interpretation in relay circuits | Interpretation in the Calculus of Propositions |
|---|---|---|
| X | The circuit X. | The proposition X. |
| 0 | The circuit is closed. | The proposition is false. |
| 1 | The circuit is open. | The proposition is true. |
| X + Y | The series connection of circuits X and Y | The proposition which is true if either X or Y is true. |
| XY | The parallel connection of circuits X and Y | The proposition which is true if both X and Y are true. |
| $X'$ | The circuit which is open when X is closed, and closed when X is open. | The contradictory of proposition X. |
| = | The circuits open and close simultaneously. | Each proposition implies the other. |