Zagadnienia podstawowe dotyczące metod formalnych w informatyce

❖ ***Logika***

Analiza języka i czynności badawczych (np. rozumowanie, definiowanie, klasyfikowanie) w celu poznania takich reguł posługiwania się językiem i wykonywania owych czynności, które uczyniłyby tę działalność możliwie najbardziej skuteczną.

❖ ***Logika formalna***

Schematy rozumowań niezawodnych

prawdziwe przesłanki $\rightarrow$ *prawdziwe wnioski*

❖ ***Podział logiki formalnej*:**

- logika tradycyjna
- współczesna

❖ ***Logika współczesna*** **obejmuje:**

- rachunek zdań
- rachunek kwantyfikatorów
- wraz z symbolem identyczności i symbolami funkcyjnymi
- zawiera wszystkie tautologie logiczne.

❖ ***Teoria mnogości****:* pozostaje w bliskim związku z logiką. Ta część teorii, która da się sformułować w terminach logicznych, obejmuje algebrę zbiorów Bool'a.

❖ ***Jedyna nauka wcześniejsza – logika.***
Stanowi ona podstawę do budowania innych nauk

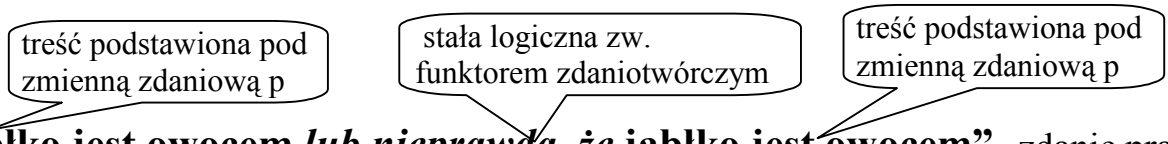
❖ ***Logika w informatyce umożliwia między innymi:***

- badanie własności języków programowania
- badanie oprogramowania
- badanie własności metodologii tworzenia oprogramowania
- specyfikację poszczególnych produktów tworzonych w kolejnych etapach tworzenia oprogramowania
- programowanie.

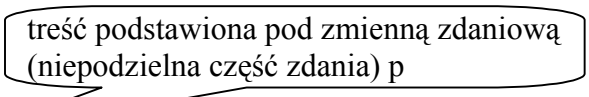
1.1. Rachunek zdań

Zdania logiczne i formuły

Przykład 1.1



(1) **„Jabłko jest owocem *lub nieprawda, że* jabłko jest owocem”**- zdanie prawdziwe

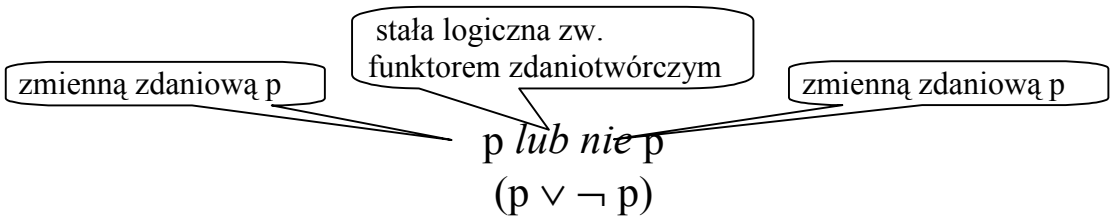


(2) **„Jabłko jest owocem”** - zdanie prawdziwe

Zdanie (2) można przekształcić **zawsze w zdanie fałszywe**:
„Jabłko jest warzywem” zdanie fałszywe
„Kwiat jest owocem” zdanie fałszywe

Zdanie (1) **jest zawsze prawdziwe**
„Kwiat jest owocem” *lub nieprawda , że* kwiat jest owocem”

Zdanie (1) *to* ***formuła zw. prawem wyłączonego środka***



Formuła generująca zawsze zdania prawdziwe po podstawieniu za zmienną zdaniową(nazwową) określonej treści nazywa się ***prawem logicznym*.**

Przykład 1.2

- „*Jeśli* (jest tak, że) *jeśli* grzmi, to błyska, to *jeśli nie* błyska, to *nie* grzmi”
- *jeśli (jeśli* p *to* q) *to (jeśli nie* q *to nie* p)
- (p→q) → (¬ q→ ¬ p) ***formuła zw. prawem transpozycji***

Symbole stałe rachunku zdań (symbole funktorów zdaniotwórczych):

∧ -koniunkcja	**∨ alternatywa**	**¬ negacja**	**→ implikacja**	**↔ równoważność**
i	lub	nie	jeśli, to	wtedy i tylko wtedy, gdy
p∧q	p ∨ q	¬ p	p → q	p ↔ q
2>0 i 2 <3	x=1 lub x=-1	-1 nie jest dodatnią liczbą całkowitą	z tego, że x>0 wynika 2x>0	z tego, że x>0 wynika 2x>0 i na odwrót

Rachunek zdań jest *dwuwartościowy:*

- logiczna prawda oznacza **1** (true)
- logiczny fałsz oznacza **0** (false)

Język rachunku zdań

❖ ***Alfabet rachunku zdań:***

1) symbole stałe logiczne: **1** (true), **0** (false*)*
2) *symbole zmiennych zdaniowych:* **p**, **q**, **r**, ...
3) *symbole spójników logicznych*: ¬ , ∧ , ∨ , → , ↔
4) *symbole pomocnicze*: lewy nawias **"("** oraz prawy nawias **")"** .

❖ ***Zasady budowania formuł***

Formuły poprawnie zbudowane, czyli *wyrażenia sensowne* tego rachunku są:

1) wyrażenia proste: zmienne zdaniowe **p**, **q**, **r**, ...
2) wyrażenia złożone:
   a) jeśli **φ** jest wyrażeniem sensownym, to **¬φ** jest wyrażeniem sensownym
   b) jeśli **φ** jest wyrażeniem sensownym i **ψ** jest wyrażeniem sensownym, to wyrażenia **φ∧ψ**, **φ∨ψ**, **φ→ψ**, **φ≡ψ** są także wyrażeniami sensownym
3) tylko formuły 1) i 2) są sensowne
4) jeśli formuła **φ** posiada zmienną zdaniową **p**, a **ψ** jest inną formułą, to przez zastąpienie każdego wystąpienia **p** formułą **ψ** otrzymujemy formułę oznaczoną **φ[ψ/p]**.

Przykład 1.3

❖ Wyrażenia sensowne: **p →(q∧r)** , **(q∨p)∧r**
❖ Wyrażenia sensowne po zastąpieniu: **(p∨¬p)→(q∧r)** , **((p∨¬p)∨p)∧r**
❖ Wyrażenia, które nie są sensowne: **p ¬r** , **p ∧(∨q)**

Tabela funktorów zdaniotwórczych (istniejących i możliwych do zdefiniowania):

- funktory jednoargumentowe (dane tabeli są wartościami wyrażeń np. **¬1**)

α	**¬**	B	C	D
1	**0**	1	1	0
0	**1**	1	0	0

- funktory dwuargumentowe (dane tabeli oznaczają wynik wyrażenia np. **1∨1**)

α	**β**	d1	**∨**	d3	d4	**→**	d6	**↔**	**∧**	d9	d10	d11	d12	d13	d14	d15	d16
1	**1**	1	**1**	1	1	**1**	1	**1**	**1**	0	0	0	0	0	0	0	0
1	**0**	1	**1**	1	1	**0**	0	**0**	**0**	1	1	1	1	0	0	0	0
0	**1**	1	**1**	0	0	**1**	1	**0**	**0**	1	1	0	0	1	1	0	0
0	**0**	1	**0**	1	0	**1**	0	**1**	**0**	1	0	1	0	1	0	1	0

Przykład 1.4

❖ ***Nowe funktory jednoargumentowe***

B α - nadawanie prawdy każdemu zdaniu logicznemu

```
#include <stdio.h>
int B (int);                    //operator nadawania prawdy
void main()
{int a;
 for (a=0; a<=1;a++)
    printf("\n B %i = %i",a,B(a));
}
int B(int)
 {return 1;}
```

❖ ***Nowe funktory dwuargumentowe***

α **d10** β - operator logicznej różnicy symetrycznej

```
#include <stdio.h>
int d10 (int,int);   //operator logicznej różnicy symetrycznej
void main()
{int a, b;
 for (a=0;a<=1;a++)
  for (b=0;b<=1;b++)
   printf("\n %i d10 %i =%i",a,b,d10(a,b));
}
int d10 (int a, int b)
 {return !(a==b);}
```

Tautologie rachunku zdań

Poprawnie zbudowane formuły rachunku zdań przy pewnych podstawieniach w miejsce zmiennych zdaniowych stają się zdaniami *prawdziwymi* lub *fałszywymi.*

Wyrażenia, które przy dowolnych podstawieniach stają się zdaniami prawdziwymi, są ***tautologiami rachunku zdań*** lub ***prawami rachunku zdań***.

Metody wyznaczania wartości formuł logicznych= metody dowodzenia

- *Metoda zero-jedynkowa* .
- *Metoda skrócona zero-jedynkowa*

Wybrane tautologie:

1.	$p \rightarrow p$	
2.	$\neg(p \wedge \neg p)$	zasada sprzeczności
3.	$p \vee \neg p$	zasada wyłączonego środka
4.	$p \rightarrow \neg(\neg p)$	zasady podwójnego zaprzeczenia
5.	$\neg(\neg p) \rightarrow p$	””
6.	$(\neg p \rightarrow p) \rightarrow p$	prawo Claviusa
7.	$(p \rightarrow \neg p) \rightarrow \neg p$	prawo redukcji do absurdu
8.	$\neg p \rightarrow (p \rightarrow q)$	prawo Dunsa Szkota
9.	$\neg(p \vee q) \leftrightarrow \neg p \wedge \neg q$	prawa De Morgana
10.	$\neg(p \wedge q) \leftrightarrow \neg p \vee \neg q$	””
11.	$(\neg p \rightarrow \neg q) \rightarrow (q \rightarrow p)$	prawa transpozycji prostej
12.	$(p \rightarrow q) \rightarrow (\neg q \rightarrow \neg p)$	””
13.	$(p \rightarrow q) \wedge (p \rightarrow \neg q) \rightarrow \neg p$	prawo redukcji do absurdu z dwiema zmiennymi
14.	$q \rightarrow (p \rightarrow q)$	prawo symplikacji
15.	$p \wedge q \leftrightarrow q \wedge p$	prawa przemienności
16.	$p \vee q \leftrightarrow q \vee p$	””
17.	$(p \leftrightarrow q) \leftrightarrow (q \leftrightarrow p)$	””
18.	$(p \rightarrow q) \wedge p \rightarrow q$	modus ponens (prawo odrywania)
19.	$(p \rightarrow q) \wedge \neg q \rightarrow \neg p$	modus tollens
20.	$(p \rightarrow q) \rightarrow [(q \rightarrow r) \rightarrow (p \rightarrow r)]$	prawo sylogizmu hipotetycznego
21.	$(p \wedge q \rightarrow r) \rightarrow [p \rightarrow (q \rightarrow r)]$	prawo eksportacji
22.	$[p \rightarrow (q \rightarrow r)] \rightarrow (p \wedge q \rightarrow r)$	prawo importacji
23.	$(p \rightarrow q) \rightarrow [(\neg p \rightarrow q) \rightarrow q]$	prawo dylematu konstrukcyjnego
24.	$(p \rightarrow q) \rightarrow [(p \rightarrow \neg q) \rightarrow \neg p]$	prawo dylematu destrukcyjnego

Przykład 1.5

❖ Dowód zerojedynkowy prawa wyłączonego środka: **p *lub nie* p**

```
#include <stdio.h>
int tautologia_1(int);
void main()
{int p;
  for (p=0;p<=1;p++)
   printf ("\n %i lub nie %i = %i",p,!p,tautologia_1(p));
}
int tautologia_1(int p)
 { return p || !p; }
```

❖ Dowód zerojedynkowy prawa transpozycji:
jeśli (jeśli **p** to **q**) to (jeśli **nie q** to **nie p**)

```
#include <stdio.h>
int tautologia_2(int,int);
void main()
{int p,q;
  for (p=0;p<=1;p++)
   for (q=0;q<=1;q++)
   printf ("\n(%i wynika %i) wynika (%i wynika %i) = %i",
             p,q,q!,p!,tautologia_2(p,q));
}
int implikacja(int p, int q)
 { switch (p)
    { case 1: return 1==q;
      case 0: return 1;    }
    return-1; }
int tautologia_2(int p, int q)
 {int wynik1=implikacja(p,q);
  int wynik2=implikacja(!q,!p);
  return implikacja(wynik1,wynik2);}
```

❖ Dowód skróconą metodą zerojedynkową prawa transpozycji

```
#include <stdio.h>
int implikacja(int,int);
void main()
{int p,q;
  for (p=0;p<=1;p++)
   for (q=0;q<=1;q++)
    if (implikacja(!q,!p)==0)
     if (implikacja(p,q)==1) {printf ("\n falsz");return;}
     else printf ("\n prawda");
 }
```

1.2. Rachunek kwantyfikatorów I-rzędu

Z terminów i twierdzeń rachunku zdań robi się użytek w teorii zwanej *rachunkiem kwantyfikatorów.* W rachunku kwantyfikatorów jest jednak uwidoczniona wewnętrzna struktura zdania prostego.

Wyróżnia się:

- ❖ *podmiot* np. *zmienna nazwowa* **x**
- ❖ oraz *orzeczenie* czyli ***predykat*** jako **L**.

Przykład 1.6

Zdanie proste „Russell jest logikiem" zapisano w języku rachunku kwantyfikatorów w postaci *formuły* :

L(x) lub **Lx** tzn. (**x jest logikiem**)

Ta formuła staje się zdaniem w wyniku:

- podstawienia jakiejś nazwy za **x**
- poprzedzenia danej formuły symbolami, reprezentującymi słówka *każdy* i *niektóry* czyli: $\forall$ i $\exists$

(3) *Każdy x* <u>jest owocem</u> $\forall_x$ **L(x)**

(4) *Pewien y* <u>jest owocem</u> $\exists_y$ **L(y)**

Terminologia rachunku kwantyfikatorów

- $\forall_x$ i $\exists_y$ - *kwantyfikatory określające ilość*
- zmienne **x**, **y** - *zmienne związane (indywiduowe)*
- rachunek I-rzędu – predykaty nie są związane kwantyfikatorem
- *zmienne wolne,* gdy nie odnoszą się do kwantyfikatorów.

Prawdziwość zdań tak zbudowanych zależy od przedmiotów, do których odnoszą się zmienne nazwowe np. **x**, **y**, **z**.

Zdanie (3) jest fałszywe dla zbioru ludzi: *Każdy* <u>człowiek</u> jest lekarzem
Zdanie (4) jest prawdziwe dla zbioru ludzi: *Pewien* <u>człowiek</u> jest lekarzem

Prawami logicznymi **rachunku kwantyfikatorów** są jedynie te zdania zbudowane z predykatów, zmiennych nazwowych oraz kwantyfikatorów, które są prawdziwe w *każdym* niepustym zbiorze przedmiotów, a nie wybranym zbiorze np. ludzi, liczb. Stąd (3) i (4) nie są prawami

Przykład 1.7

Przykłady wyrażeń prawdziwych ze względu na formę logiczną:

(5) $\forall_x \mathbf{Q(x)} \rightarrow \exists_x \mathbf{Q(x)}$

(6) $\forall_x [\mathbf{P(x)} \rightarrow \mathbf{Q(x)}] \wedge \forall_x [\mathbf{Q(x)} \rightarrow \mathbf{S(x)}] \rightarrow \forall_x [\mathbf{P(x)} \rightarrow \mathbf{S(x)}]$

Zdania (5) i (6) są zawsze prawdziwe nie dzięki treści predykatów czy treści zmiennych nazwowych, lecz układowi stałych symboli logicznych $\forall x$, $\exists x$ oraz $\rightarrow$. Te i pozostałe symbole rachunku zdań wyznaczają *formę logiczną* wyrażenia w rachunku kwantyfikatorów.

Język rachunku kwantyfikatorów (nadzbiór języka rachunku zdań) jest zdefiniowany przez zbiór termów i formuł nad pewnym alfabetem oraz zasad ich tworzenia.

***Alfabet języka rachunku kwantyfikatorów*:**

1) nieskończenie wiele zmiennych zdaniowych p, q, r
2) nieskończenie wiele zmiennych indywiduowych, x,y,z...
3) nieskończenie wiele parametrów indywiduowych
4) nieskończenie wiele predykatów n-argumentowych oznaczanych przez L, P,..., gdzie n=1,2,3,...
5) nieskończenie wiele symboli funkcyjnych n-argumentowych (symbole operacji), f,g,h,... gdzie n=1,2,...
6) stałe logiczne: $\neg,\wedge,\vee,\rightarrow,\leftrightarrow$
7) kwantyfikator ogólny (uogólnienie koniunkcji) $\forall$ oraz kwantyfikator szczegółowy lub egzystencjalny (uogólnienie alternatywy) $\exists$
8) symbol identyczności =
5) symbole pomocnicze: lewy nawias (oraz prawy nawias).

Zasady tworzenia formuł poprawnie zbudowanych:

1) ***Formuła atomowa*** czyli ***term*** składa się z n-argumentowego predykatu wraz z następującymi po nim n-argumentami np **Lx**, **Lxy**,...
2) ***Formuła złożona*** powstaje z posiadanych już termów **L**, **Q** następująco:
   2.1) przez poprzedzenie **L** znakiem negacji
   2.2) przez połączenie **L** i **Q** znakiem koniunkcji lub alternatywy lub implikacji lub równoważności
   2.3) przez poprzedzenie **L** kwantyfikatorem

Aksjomaty i reguły wnioskowania **-** ujęcie Hilberta i Bernaysa

Założenia:

❖ Aksjomatami są wszystkie twierdzenia rachunku zdań oraz wszystkie formuły podpadające pod schematy:

$$(A1) \quad \forall_x L(x) \rightarrow Q(p)$$
$$(A2) \quad Q(p) \rightarrow \exists_x L(x)$$

❖ Regułami pierwotnymi wnioskowania są: reguła dołączania kwantyfikatora ogólnego DU i reguła dołączania kwantyfikatora egzystencjalnego DE

$$(DU) \quad Z \quad Q \rightarrow L(x) \quad \textit{wolno wnosić} \quad Q \rightarrow \forall_x L(x)$$
$$(DE) \quad Z \quad L(x) \rightarrow Q \quad \textit{wolno wnosić} \quad \exists_x L(x) \rightarrow Q$$

w formule Q x nie może być zmienną wolną

Przykłady fałszywych formuł:

- $(x<2) \rightarrow (x<3)$, wówczas z DU mamy $(1<2) \rightarrow \forall_x (x<3)$ - implikacja fałszywa
- $(x<2) \rightarrow (x<3)$, wówczas x DE mamy $\exists_x (x<2) \rightarrow (4<3)$ - implikacja fałszywa

❖ Reguły wtórne wnioskowania:

Przykład 1.8.

Wyprowadzenia wtórnej reguły

Z L(x) wolno wnosić $\forall_x L(x)$

(1)	L(x)	- założenie
(2)	$L(x) \rightarrow ((p \vee \neg p) \rightarrow L(x))$	- prawo wyłączonego środka
(3)	$(p \vee \neg p) \rightarrow L(x)$	- prawo odrywania z (1) i (2)
(4)	$(p \vee \neg p) \rightarrow \forall_x L(x)$	- reguła dołączania DU
(5)	$\forall_x L(x)$	- prawo odrywania z $p \vee \neg p$ i 4

Przykład 1.9.

Reguły, których prawdziwość zależy od wartości zmiennych wolnych

```
#include <conio.h>
#include <stdlib.h>
const N=10;
void usun(char tab[],int x, int& ile);
void dodaj(char tab[], int &ile);
void wyswietl(char tab[], int ile);

void main()
{ char tab[N]; int ile=0,x;
  do
  { switch(getch()
   {case '1':dodaj(tab,ile); wyswietl(tab,ile); break;
    case '2':x=random(ile); usun(tab,x,ile);
             wyswietl(tab,ile); break;
    case '3':return;}
  }while (1);
}

void usun(char tab[],int x, int& ile)
   // reguła 1 ∃0 ≤ x ≤ ile-1 ∀x ≤ i < ile-1 (tab[i]=tab[i+1])
   // reguła prawdziwa dla zmiennej wolnej 0 < ile≤ N
   // istnieje takie x>=0 i x<=ile-1;          (x-zmienna związana przez ∃)
   // dla każdego i, gdy i>=x oraz i<ile-1, że  (i-zmienna związana przez ∀)
   // jest tab[i]=tab[i+1]                          (predykat)
{if (ile==0) return;
  for (int i=x; i<ile-1;i++)  tab[i]=tab[i+1];
     ile--;}

void dodaj(char tab[], int &ile)
   // reguła 2 ∃0≤ile<N  tab[ile++]=48+ile)
   // reguła prawdziwa dla zmiennej wolnej N>0
   // istnieje takie ile, że ile>=0 i ile<N, że     (ile-zmienna związana przez ∃)
   // jest tab[ile]=48+ile                          (predykat)
{if (ile<N) tab[ile++]=48+ile;}


void wyswietl(char tab[], int ile)
   // reguła 3 ∀0≤i<ile (putch(tab[i]))
   // reguła prawdziwa dla zmiennej wolnej 0 < ile ≤ N
   // dla każdego i, gdy i>=0 i i<ile, że           (i-zmienna związana przez ∀)
   // wyswietl tab[i]                               (predykat)
{for (int i=0; i<ile;i++)  putch(tab[i]);
  putch(' ');}
```

Najczęściej stosowane twierdzenia z predykatami jednoargumentowymi:

1) $\forall_x L(x) \leftrightarrow \neg\exists_x \neg L(x)$ definiowanie $\forall$ za pomocą $\exists$
2) $\exists_x L(x) \leftrightarrow \neg\forall_x \neg L(x)$ definiowanie$\exists$ za pomocą $\forall$
3) $\forall_x (L(x) \wedge Q(x)) \leftrightarrow (\forall_x L(x) \wedge \forall_x Q(x))$
4) $\exists_x (L(x) \wedge Q(x)) \rightarrow (\exists_x L(x) \wedge \exists_x Q(x))$
5) $\exists_x (L(x) \vee Q(x) \leftrightarrow (\exists_x L(x) \vee \exists_x Q(x))$
6) $(\forall_x (L(x) \vee \forall_x Q(x)) \rightarrow \forall_x (L(x) \vee Q(x))$
7) $\forall_x (L(x) \rightarrow Q(x)) \rightarrow (\forall_x L(x) \rightarrow \forall_x Q(x))$
8) $\forall_x (L(x) \rightarrow Q(x)) \leftrightarrow \neg\exists_x (L(x) \wedge \neg Q(x))$

oraz dwuargumentowymi:

9) $\forall_x \forall_y A(xy) \leftrightarrow \forall_y \forall_x A(xy)$
10) $\exists_x \exists_y A(xy) \leftrightarrow \exists_y \exists_x A(xy)$
11) $\exists_y \forall_x A(xy) \rightarrow \forall_x \exists_y A(xy)$

Bezpośrednie konsekwencje 1) i 2) są prawa De Morgana dla kwantyfikatorów:

$\neg\forall_x L(x) \leftrightarrow \exists_x \neg L(x)$
$\neg\exists_x L(x) \leftrightarrow \forall_x \neg L(x)$

Rachunek zdań z identycznością i funkcjami

Jest to relacja między indywiduami określana jako relacja równoważności (materiał dotyczący relacji)

$$(x = y) \rightarrow (L(x) \rightarrow L(y))$$

Korzystając z symbolu identyczności i odpowiednich predykatów można wprowadzać *definicyjne symbole funkcyjne*.

np.

a) Z predykatu

„x jest następnikiem y” czyli L(x,y)

można przejść do funkcji następnika symbolizowanym literą s:

$x = s(y) \leftrightarrow L(xy)$

b) Z predykatu

„ x jest sumą x i z” czyli L(xyz)

można przejść do zapisu funkcyjnego

$x = y + z \leftrightarrow L(xyz)$

Cokolwiek da się wyrazić za pomocą formuł funkcyjnych, da się wyrazić za pomocą formuł predykatowych.

1.3. Teoria schematów wnioskowania – logika przedmiotów

- Formuły, które są prawdziwe ze względu na swoją formę logiczną, są spełniane przez wszystkie przedmioty z dowolnej (niepustej) dziedziny rozważań.
- Można je traktować jako *schematy zdań prawdziwych* w dowolnej dziedzinie przedmiotów.
- Każdemu twierdzeniu logiki przedmiotowej w postaci okresu warunkowego przyporządkowany jest jednocześnie pewien schemat poprawnego wnioskowania, opisany pewną regułą.

Przykład 1.10

Formuła (5)

poprzednik (założenie)		*następnik (teza)*
$\forall_x Q(x)$	$\rightarrow$	$\exists_x Q(x)$

generuje schemat wnioskowania

$\forall_x Q(x)$	*Przesłanka*
$\exists_x Q(x)$	*Wniosek*

Formuła (6) $\forall_x [P(x) \rightarrow Q(x)] \wedge \forall_x [Q(x) \rightarrow S(x)] \quad \rightarrow \quad \forall_x[P(x) \rightarrow S(x)]$
generuje schemat rozumowania

$$\frac{\forall_x [P(x) \rightarrow Q(x)] \quad \forall_x [Q(x) \rightarrow S(x)]}{\forall_x[P(x) \rightarrow S(x)]}$$

Każdemu schematowi wnioskowania jest przyporządkowane jednoznacznie twierdzenie w formie implikacji, której poprzednikiem jest koniunkcja przesłanek, zaś następnikiem zdanie występujące w schemacie jako wniosek.
Przykładem jest rachunek sekwencji Gentzena.

Twierdzenie o dedukcji

Głosi ono, że **dla udowodnienia implikacji wystarczy z jej poprzednika wyprowadzić następnik, stosując logiczne reguły wnioskowania.**

Niech **X** będzie zbiorem jakichś formuł teorii **T**, zaś **A** formułą zamkniętą (tj. bez zmiennych wolnych) tejże teorii; jeśli z **X** i **A**, przy zastosowaniu reguł właściwych teorii **T**, da się wydedukować formuła **B**, to z **X** da się wydedukować implikacja **A→B**, czyli

jeśli **X**, **A** ⊢ **B**, to **X** ⊢ (A→B)

Jeśli X jest zbiorem pustym mamy: jeśli A ⊢ B, to ⊢ A→B

Odwrotne twierdzenie o dedukcji: jeśli X ⊢ A→B, to X, A⊢ B

Pozwala wyprowadzać nowe reguły dedukcyjne na podstawie udowodnionych już formuł logicznych

np. Jeśli mamy udowodnione lub przyjęte aksjomatycznie: A→¬(¬A)

wówczas można przyjąć regułę: A ⊢ ¬ (¬A) :Z A można wydedukować ¬¬ A

Dedukcja naturalna

Intencją jej jest jak najdalej idące przybliżenie logicznej teorii dowodu do rzeczywistej praktyki dowodowej w matematyce i i innych naukach.

	Reguły wprowadzania			Reguły opuszczania	
(∧)	$\dfrac{AB}{A\wedge B}$			$\dfrac{A\wedge B}{A}$	$\dfrac{A\wedge B}{B}$
(∨)	$\dfrac{A}{A\vee B}$	$\dfrac{B}{A\vee B}$		$\dfrac{A\vee B\ A\vdash C\ B\vdash C}{C}$	
(¬)	$\dfrac{A\vdash 0}{\neg A}$			$\dfrac{A\neg A}{0}$	$\dfrac{\neg\neg A}{A}$
(→)	$\dfrac{A\vdash B}{A\to B}$	*odpowiednik twierdzenia o dedukcji*		$\dfrac{A\ A\to B}{B}$	
(∀)	$\dfrac{B(a)}{\forall_x B(x)}$			$\dfrac{\forall_x B(x)}{B(a)}$	
(∃)	$\dfrac{B(a)}{\exists_x B(x)}$			$\dfrac{\exists_x B(x)\ B(a)\vdash A}{A}$	

gdzie a – zmienna nie związana (występuje w roli parametru) ,
x - zmienna związana,
O- formuła fałszywa logicznie

oraz reguła

$$\frac{0}{A}$$

Dowody przeprowadzane metodą dedukcji naturalnej to *dowody założeniowe* (kierując się regułami wyprowadza się wnioski z założeń przyjętych na próbę):

- *dowód wprost*
- *dowód nie wprost.*

Przykład 1.11

Dana jest formuła $(p\rightarrow q)\wedge(r\rightarrow s)\wedge(p\wedge r) \rightarrow (q\wedge s)$.
Należy udowodnić, że formuła ta jest prawem

(1)	$p\rightarrow q$	założenie uzyskane z reguły opuszczania ($\wedge$)
(2)	$r\rightarrow s$	””
(3)	$p\wedge r$	””
(4)	p	z reguły opuszczania ($\wedge$) z (3)
(5)	r	z reguły opuszczania ($\wedge$) z (3)
(6)	q	z reguły opuszczania ($\rightarrow$) z (1),(4)
(7)	s	z reguły opuszczania ($\rightarrow$) z (2),(5)
(8)	$q\wedge s$	z reguły dołączania ($\wedge$) z (6) ,(7)

Stąd mamy: $(p\rightarrow q)\wedge(r\rightarrow s)\wedge(p\wedge r)\rightarrow(q\wedge s)$

z reguły dołączania ($\rightarrow$) z (1),(2),(3),(8)

Zagadnienie zupełności systemu dedukcyjnego

Definicje zupełności systemu dedukcyjnego:

1) System jest zupełnym zbiorem zdań zawierającym terminy stałe **P1**,...**Pn**, wtedy i tylko wtedy, gdy dla każdego zdania **A** zawierającego jako symbole stałe jedynie wyrażenia spośród **P1**,...,**Pn** prawdą jest , że $\mathbf{A} \in \mathbf{S}$ lub $\mathbf{(\neg A)} \in \mathbf{S}$. Oznacza to że albo zdanie A albo jego negacja jest twierdzeniem systemu. Dotyczy to zdań, w których nie występują zmienne wolne.
2) System jest zupełny wtedy i tylko wtedy, gdy każda poprawnie zbudowana formuła bądź jest twierdzeniem systemu, bądź po dołączeniu do jego aksjomatów wprowadzi doń sprzeczność. Każda formuła rachunku zdań spełnia ten warunek. Rachunek kwantyfikatorów nie spełnia obu warunków zupełności
3) System dedukcyjny logiki jest pełny wtedy i tylko wtedy, gdy z jego aksjomatów dadzą się wywieść wszystkie zdania będące zdaniami prawdziwymi w każdym modelu. Warunek ten spełniają rachunek zdań oraz rachunek kwantyfikatorów I-rzędu.

Niesprzeczność

1) Sprzeczność to stosunek między dwoma zdaniami,
   z których jedno stanowi negację drugiego.
2) Sprzeczność to własność zbioru zdań, polegająca na tym,
   że w zbiorze tym znajdują się lub dadzą się z niego wyprowadzić zdania sprzeczne.
3) System jest niesprzeczny, gdy nie ma w nim żadnego takiego wyrażenia Φ,
   że Φ jest tezą oraz $\neg\Phi$ jest tezą.
4) System jest niesprzeczny (przepełniony), gdy nie wszystkie wyrażenia sensowne są w nim tezami. Systemy zawierające klasyczny rachunek zdań spełniają warunek niesprzeczności.

Niesprzeczność danego systemu można dowieść przez interpretację na gruncie innego systemu, którego niesprzeczność jest zagwarantowana. Dla systemów prostych wskazuje się taką cechę, która przysługuje wszystkim aksjomatom systemu np. dla rachunku zdań jest nią tautologia. Systemami niesprzecznymi są rachunek zdań i rachunek kwantyfikatorów.

Nierozstrzygalność

Teoria nazywa się *rozstrzygalną*, gdy istnieje metoda pozwalająca o każdym wyrażeniu tej teorii rozstrzygnąć za pomocą skończonej liczby prób, czy jest ono twierdzeniem danej teorii.

Przykład: Rachunek zdań jest teorią rozstrzygalną, gdyż efektywnym sposobem rozstrzygania jest metoda zerojedynkowa.

Rachunek kwantyfikatorów jest nierozstrzygalny, jedynie jego pewne fragmenty są rozstrzygalne: np. rachunek jednoargumentowy, pewne klasy twierdzeń określone przez swoją postać normalną(np. nie zawierające kwantyfikatorów ogólnych lub szczegółowych, lub ani jednego szczegółowego stojącego przed ogólnym).