Sztuczna Inteligencja

Soma Dutta

Wydział Matematyki i Informatyki, UWM w Olsztynie
soma.dutta@matman.uwm.edu.pl

Wykład - 8: Rachunek zdań

Semestr letni 2022

Podsumowanie

Logika ma dwa aspekty: Reprezentacja wiedzy i wnioskowanie.

- Wiedza jest reprezentowana przez język formalny i odnosi się do niektórych faktów na temat świata.
  - Składnia logiki: Dotyczy zasad formowania zdań
  - Semantyka logiki: Łączy symboliczną reprezentację zdań z prawdziwymi faktami, przypisując każdemu zdaniu wartość prawda lub fałsz w określonym modelu.
- Proces wnioskowania ma również dwa aspekty.
  - Konsekwencja składniowa ($\vdash_i$): Wnioskowanie po zastosowaniu pewnej metody dowodowej, która pociąga za sobą zdanie α ze zbioru zdań KB.
  - Konsekwencja semantyczna ($\models$): Zbiór zdań KB pociąga zdanie α jako konsekwencja, jeśli dowolne przypisanie lub model, który sprawia, że każde zdanie KB jest prawdziwe, czyni α również prawdziwym.

Wnioskowanie poprzez sprawdzenie modeli

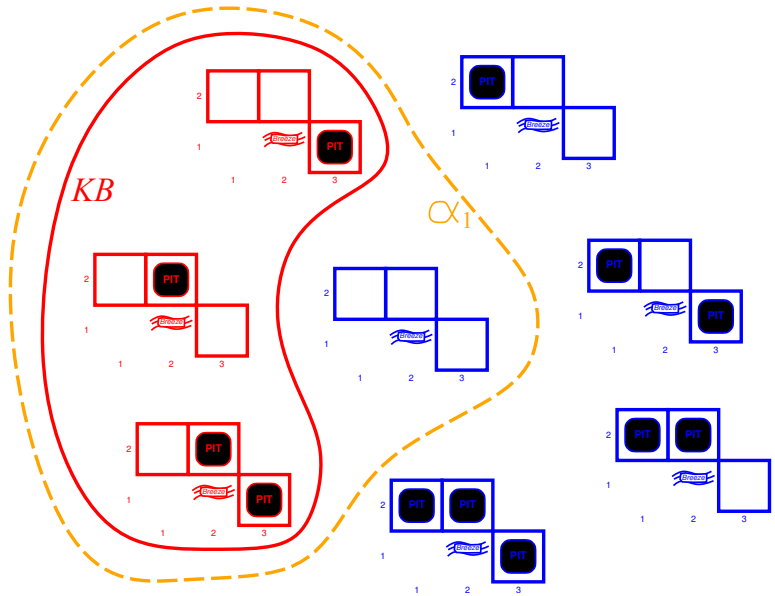
Przykład:

- Niech KB $= \{p, p \Rightarrow q \wedge r, q\}$ oraz $\alpha = r$.
- Chcielibyśmy sprawdzić, czy $KB \models \alpha$.

przypisanie	p	q	r	**p**	**p** $\Rightarrow$ **q** $\wedge$ **r**	**q**	r
v_1	0	0	0	0	1	0	0
v_2	0	0	1	0	1	0	1
v_3	0	1	0	0	1	1	0
v_4	0	1	1	0	1	1	1
v_5	1	0	0	1	0	0	0
v_6	1	0	1	1	0	0	1
v_7	1	1	0	1	0	1	0
v_8	1	1	1	1	1	1	1

- $M(KB) = \{v_8\}$, a $M(r) = \{v_2, v_4, v_6, v_8\}$. To potwierdza $M(KB) \subseteq M(r)$.
- Więc $KB \models r$.

Wracając do przykładu świata Wumpus



KB = reguły świata Wumpusa + obserwacje

α_1 = "[1,2] jest bezpieczny", $KB \models \alpha_1$, dowód przez sprawdzenie modeli

Wnioskowanie przez wyliczanie

Wyliczanie wszystkich modeli w głąb jest poprawne i pełne

```
function TT-ENTAILS(KB, α) returns true or false
    symbols ← a list of the proposition symbols in KB and α
    return TT-CHECK-ALL(KB, α, symbols, [])
end function

---

function TT-CHECK-ALL(KB, α, symbols, model) returns true or false
    if EMPTY(symbols) then
        if PL-TRUE(KB,model) then
            return PL-TRUE(α, model)
        else
            return true
        end if
    else
        P ← FIRST(symbols);
        rest ← REST(symbols)
        return TT-CHECK-ALL(KB, α, rest, EXTEND(P, true, model)) and
                TT-CHECK-ALL(KB, α, rest, EXTEND(P, false, model))
    end if
end function
```

$O(2^n)$ dla n symboli; problem jest co-NP-zupełny

Tautologie i spełnialność

- Zdanie jest tautologią (tautology), jeśli jest prawdziwe w każdym przypisaniu. Na przykład $p \Rightarrow (q \Rightarrow p)$.

przypisanie	p	q	$p \Rightarrow (q \Rightarrow p)$
v_1	0	0	$0 \Rightarrow (0 \Rightarrow 0) = 1$
v_2	0	1	$0 \Rightarrow (1 \Rightarrow 0) = 1$
v_3	1	0	$1 \Rightarrow (0 \Rightarrow 1) = 1$
v_4	1	1	$1 \Rightarrow (1 \Rightarrow 1) = 1$

- Zdanie jest spełnialne (satisfiable), zwane także niesprzecznym, jeśli istnieje co najmniej jedno przypisanie, w którym jest prawdziwe. Na przykład $p \Rightarrow (p \wedge q)$.

przypisanie	p	q	$p \Rightarrow (p \wedge q)$
v_1	0	0	$0 \Rightarrow 0 = 1$
v_2	0	1	$0 \Rightarrow 0 = 1$
v_3	1	0	$1 \Rightarrow 0 = 0$
v_4	1	1	$1 \Rightarrow 1 = 1$

- Zbiór zdań uważa się za spełnialny, jeśli istnieje przypisanie, zgodnie z którym każde zdanie zbioru jest prawdziwe.

Logiczna równoważność

- Dwa zdania są logicznie równoważne, jeśli mają taką samą wartość w ramach dowolnego przypisania; tzn., $\alpha \equiv \beta$ wtedy i tylko wtedy gdy $\alpha \models \beta$ oraz $\beta \models \alpha$. Na przykład $p \Rightarrow q \equiv \neg p \vee q$.

przypisanie	p	q	$p \Rightarrow q$	$\neg p \vee q$
v_1	0	0	1	$(0 \vee 1) = 1$
v_2	0	1	1	$(1 \vee 1) = 1$
v_3	1	0	0	$(0 \vee 0) = 0$
v_4	1	1	1	$(0 \vee 1) = 1$

- Kolejne zdania są logicznie równoważne.

- $\alpha \wedge \beta \equiv \beta \wedge \alpha$ || $\alpha \vee \beta \equiv \beta \vee \alpha$ **(Commutativity)**
- $\alpha \wedge (\beta \wedge \gamma) \equiv (\alpha \wedge \beta) \wedge \gamma$ || $\alpha \vee (\beta \vee \gamma) \equiv (\alpha \vee \beta) \vee \gamma$ **(Associativity)**
- $\neg\neg\alpha \equiv \alpha$ **(Double negation)**
- $\alpha \Rightarrow \beta \equiv \neg\beta \Rightarrow \neg\alpha$ **(Contraposition)**
- $\neg(\alpha \wedge \beta) \equiv (\neg\alpha \vee \neg\beta)$ || $\neg(\alpha \vee \beta) \equiv (\neg\alpha \wedge \neg\beta)$ **(De Morgan)**
- $\alpha \wedge (\beta \vee \gamma) \equiv (\alpha \wedge \beta) \vee (\alpha \wedge \gamma)$ || $\alpha \vee (\beta \wedge \gamma) \equiv (\alpha \vee \beta) \wedge (\alpha \vee \gamma)$ **(Distributivity)**
- $\alpha \vee \neg\alpha \equiv T$ **(Law of excluded middle)**
- $\alpha \wedge \neg\alpha \equiv F$ **(Law of contradiction)**

Związek między spełnialnością a konsekwencją semantyczną

- Zdanie α jest niespełnialne wtedy i tylko wtedy, gdy $\neg\alpha$ jest tautologią.
- Spełnialność jest związana z wnioskowaniem przez sprowadzanie do sprzeczności:
  $KB \models \alpha$ wtedy i tylko wtedy gdy $(KB \wedge \neg\alpha)$ jest niespełnialne.
  - Załóżmy, że $KB = \{\beta\}$. Więc
    $\beta \models \alpha$ wtedy i tylko wtedy gdy $(\beta \wedge \neg\alpha)$ jest niespełnialne.
  - $\beta \wedge \neg\alpha \equiv \neg\neg\beta \wedge \neg\alpha$ (double negation)
    $\equiv \neg(\neg\beta \vee \alpha)$ (De Morgan Law)
  - $\neg(\neg\beta \vee \alpha)$ jest niespełnialne wtedy i tylko wtedy gdy $(\neg\beta \vee \alpha)$ jest tautologią, tzn., $\beta \Rightarrow \alpha$ jest tatutologią.
  - Tak więc, $\beta \models \alpha$ wtedy i tylko wtedy gdy $(\beta \Rightarrow \alpha)$ jest tautologią.

SAT-problem

- Udowodnienie α z założenia β poprzez sprawdzenie niespełnienia $(\beta \wedge \neg\alpha)$ odpowiada dokładnie standardowa matematyczna technika reductio ad absurdum (dosłownie 'sprowadzenie do sprzeczności').
- Problem określania spełnienia zdań w rachunku zdań - znany jako SAT problem - jest pierwszym problemem, który okazał się NP-zupełny.

Logiczna konsekwencja przy użyciu metody dowodu

Zastosowanie reguł wnioskowania

- W metodzie dowodowej przyjmuje się pewną początkową wiedzę przedstawioną w zdaniach bez kwestionowania ich dowodu.
- Zbiór reguł, często nazywanych regułami wnioskowania, jest również ustalany na początku.
- Zadaniem jest poprawne generowanie nowych zdań ze starych za pomocą reguł.
- Dowód to łańcuch zdań, który prowadzi do pożądanego celu, w którym każde zdanie w łańcuchu jest albo zdaniem już udowodnionem, albo otrzymane jako konsekwencja przy użyciu reguły.

Reguły wnioskowania

Reguły wnioskowania: $\frac{\alpha_1,\ldots,\alpha_n}{\beta}$

Reguła jest poprawna, jeśli β jest prawdziwa w każdej interpretacji, w której prawdziwe są $\alpha_1,\ldots,\alpha_n$.

Przykłady poprawnych reguł wnioskowania

Reguła odrywania (modus ponens): $\frac{\alpha,\quad \alpha\Rightarrow\beta}{\beta}$

Reguła eliminacji koniunkcji: $\frac{\alpha_1\wedge\ldots\wedge\alpha_n}{\alpha_i}$

Reguła wprowadzenia koniunkcji: $\frac{\alpha_1,\ldots,\alpha_n}{\alpha_1\wedge\ldots\wedge\alpha_n}$

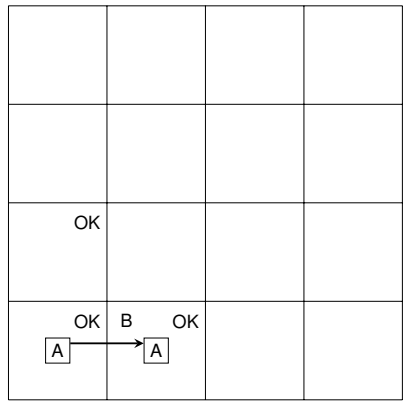
Reguła rezolucji: $\frac{\alpha\vee\beta,\quad \neg\beta\vee\gamma}{\alpha\vee\gamma}$

Reguły z logicznej równoważności

- $\frac{\alpha \Rightarrow \beta}{\neg\beta \Rightarrow \neg\alpha}$ (Reguła kontrapozycji)
- $\frac{\neg(\alpha \vee \beta)}{\neg\alpha \wedge \neg\beta}$ $\frac{\neg(\alpha \wedge \beta)}{\neg\alpha \vee \neg\beta}$ (Reguła De Morgana)

Świat Wumpusa: wnioskowanie za pomocą reguł

Reprezentacja KB



- R_1: $\neg P_{1,1}$ ($P_{x,y}$ oznacza w polu [x, y] jest dół))
- R_2: $B_{1,1} \Leftrightarrow P_{2,1} \vee P_{1,2}$ ($B_{x,y}$ oznacza w polu [x, y] jest bryza)
- R_3: $B_{2,1} \Leftrightarrow (P_{1,1} \vee P_{2,2} \vee P_{3,1})$

  $\vdots$

- R_4: $\neg B_{1,1}$.
- R_5: $B_{2,1}$.

Świat Wumpusa: wnioskowanie za pomocą reguł

(1) $(B_{1,1} \Rightarrow (P_{2,1} \vee P_{1,2})) \wedge ((P_{2,1} \vee P_{1,2}) \Rightarrow B_{1,1})$ (Reguła R_2)

(2) $((P_{2,1} \vee P_{1,2}) \Rightarrow B_{1,1})$ (Reguła eliminacji koniunkcji)
Nazwijmy tę nową wiedzę jako R_6.

(3) $\neg B_{1,1} \Rightarrow \neg(P_{2,1} \vee P_{1,2})$ (Reguła kontrapozycji)
Nazwijmy tę nową wiedzę jako R_7.

(4) $\neg B_{1,1}$ (Reguła R_4)

(5) $\neg(P_{2,1} \vee P_{1,2})$ (Reguła odrywania (4, 3))
Nazwijmy tę nową wiedzę jako R_8.

(6) $\neg P_{2,1} \wedge \neg P_{1,2}$ (Reguła De Morgana)
Nazwijmy tę nową wiedzę jako R_9.

(7) $\neg P_{1,2}$ (Reguła eliminacji koniunkcji)
Nazwijmy tę nową wiedzę jako R_{10}.

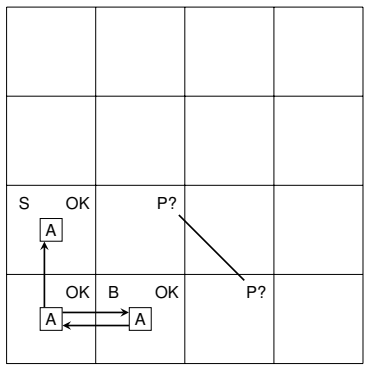
Tak więc, z zastosowaniem reguł zostało udowodnione: $KB \vdash \neg P_{1,2}$.

Czy dowodzenie z zastosowaniem reguł jest pełne?

- Przede wszystkim, reguły wnioskowania stosowane są poprawne. Równiez musimy mieć pełny algorytm wnioskowania, który ich używa.
- Możemy zastosować dowolny z algorytmów przeszukiwania omówionych wcześniej, znaleźć sekwencję kroków, która stanowi dowód.
  - Stan początkowy to opis KB.
  - Funkcję Action definiuje się za pomocą reguł. Zbiór działań składa się ze wszystkich reguł wnioskowania stosowanych do wszystkich zdań, które pasują do górnej połowy reguły wnioskowania.
  - Result jest zdefinowany przez dodanie zdania w dolnej części wnioskowania reguły.
  - Stan celu to zdanie, które chcemy udowodnić.
- Problem polega na tym, że jeśli dany zbiór reguł jest nieodpowiedni, algorytm może nie osiągnąć stanu celu, czyli zdania, które chcemy udowodnić.

Reguła rezolucji

- Reguła rezolucji jest taką regułą wnioskowania, która daje pełność algorytmu wnioskowania w połączeniu z dowolnym kompletnym algorytmem wyszukiwania.
- Wróćmy do przykładu Wumpusa: agent powrócił z pola [2, 1] do [1, 1] i idzie do [1, 2].



- R_{11}: $\neg B_{1,2}$
- R_{12}: $B_{1,2} \Leftrightarrow (P_{1,1} \vee P_{2,2} \vee P_{1,3})$
- Stosując tę samą metodę dowodów co poprzednio, możemy udowodnić R_{13}: $\neg P_{2,2}$ i R_{14}: $\neg P_{1,3}$.
- $B_{2,1} \Rightarrow (P_{1,1} \vee P_{2,2} \vee P_{3,1})$ (z R_3 po użyciu reguła eliminacji koniunkcji)
- $P_{1,1} \vee P_{2,2} \vee P_{3,1}$ (Reguła odrywania zastosowana do R_5)
- $P_{2,2} \vee P_{3,1}$ (Reguła rezolucji zastosowana do R_1)
- $P_{3,1}$ (Reguła rezolucji zastosowana do R_{13})

Reguła rezolucji

- Ostatnie dwa kroki to zastosowanie reguły rezolucji jednostkowej (unit resolution rule), która jest następująca.

$$\frac{l_1 \vee l_2 \vee \ldots \vee l_m \qquad m}{l_1 \vee \ldots \vee l_{i-1} \vee l_{i+1} \vee \ldots \vee l_m}$$

gdzie każdy l_j i m jest literałem (literals) (tzn., symbolem zdania atomowego, albo jego negacją), a l_i i m są literałami uzupełniającymi się (complementary literals) (tzn. jeden jest negacją drugiego).

- Skończona liczba literałów połączona przez alternatywę nazywa się klauzulą (clause).
- Uogólniona forma powyższej reguły jest następująca.

$$\frac{l_1 \vee l_2 \vee \ldots \vee l_m \qquad m_1 \vee m_2 \vee \ldots m_n}{l_1 \vee \ldots \vee l_{i-1} \vee l_{i+1} \vee \ldots \vee l_m \vee m_1 \vee \ldots \vee m_{j-1} \vee m_{j+1} \vee \ldots m_n}$$

gdzie l_i i m_j są literałami uzupełniającymi się.

CNF

- Sama reguła rezolucji uzasadnia, że jest poprawna. Co więcej, stanowi podstawę rodziny pełnych procedur wnioskowania. Weryfikator twierdzeń oparty na rezolucji może, dla dowolnych zdań α i β w rachunku zdań, rozstrzygać, czy $\alpha \models \beta$.
- Wydaje się, że reguła rezolucji ma zastosowanie tylko do klauzul (czyli alternatywy literałów) dotyczący tylko baz wiedzy i zapytań składających się z klauzul. Jak więc może prowadzić do pełnej procedury wnioskowania dla całego rachunku zdań?
- Odpowiedzią jest: każde zdanie rachunku zdań jest logicznie równoważne koniunkcji klauzul. Zdanie wyrażone w postaci koniunkcji klauzul nazywamy Koniunkcyjną Postacią Normalną (Conjunctive Normal Form) lub CNF.

Przykład

- Rozważajmy następujące zdanie:

$$
\begin{aligned}
& B_{1,1} \Leftrightarrow (P_{1,2} \vee P_{2,1}) \\
\equiv\ & (B_{1,1} \Rightarrow (P_{1,2} \vee P_{2,1})) \wedge ((P_{1,2} \vee P_{2,1}) \Rightarrow B_{1,1}) \\
\equiv\ & (\neg B_{1,1} \vee (P_{1,2} \vee P_{2,1})) \wedge (\neg(P_{1,2} \vee P_{2,1}) \vee B_{1,1}) && (\alpha \Rightarrow \beta \equiv \neg\alpha \vee \beta) \\
\equiv\ & (\neg B_{1,1} \vee P_{1,2} \vee P_{2,1}) \wedge ((\neg P_{1,2} \wedge \neg P_{2,1}) \vee B_{1,1}) && (\neg(\alpha \vee \beta) \equiv \neg\alpha \wedge \neg\beta) \\
\equiv\ & (\neg B_{1,1} \vee P_{1,2} \vee P_{2,1}) \wedge (\neg P_{1,2} \vee B_{1,1}) \wedge (\neg P_{2,1} \vee B_{1,1}) && ((\alpha \wedge \beta) \vee \gamma) \equiv (\alpha \vee \gamma) \wedge (\beta \vee \gamma))
\end{aligned}
$$

CNF, DNF

- Koniunkcyjna postać normalna (CNF) (ang. conjunctive normal form) - formuła zapisana w postaci koniunkcji klauzul, z których każda jest alternatywą literałów.

  $(l_{11} \vee l_{12} \vee \ldots \vee l_{1k}) \wedge (l_{21} \vee l_{22} \vee \ldots \vee l_{2s}) \wedge \ldots (l_{m1} \vee l_{m2} \vee \ldots \vee l_{mn})$
  gdzie każdy l_{ij} jest zdaniem atomowym lub jego negacją, i każde wyrażenie z $(l_{11} \vee l_{12} \vee \ldots \vee l_{1k})$, $\ldots (l_{m1} \vee l_{m2} \vee \ldots \vee l_{mn})$ jest klauzulą.

- Dysjunkcyjna postać normalna (DNF) (ang. disjunctive normal form) - formuła zapisana w postaci dysjunkcji (alternatywy) wyrażeń, z których każde jest koniunkcją literałów.

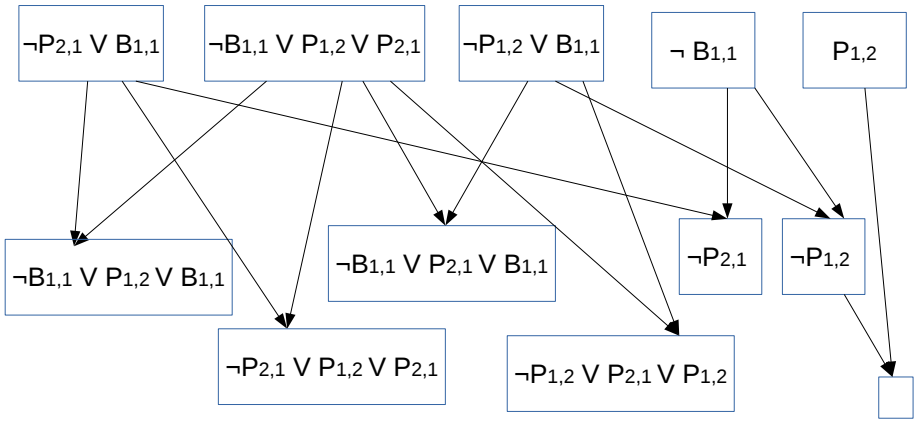
  $(l_{11} \wedge l_{12} \wedge \ldots \wedge l_{1k}) \vee (l_{21} \wedge l_{22} \wedge \ldots \wedge l_{2s}) \vee \ldots (l_{m1} \wedge l_{m2} \wedge \ldots \wedge l_{mn})$

Algorytm wnioskowania za pomocą reguły rezolucji

- Aby pokazać, że $KB \models \alpha$, pokazujemy, że $(KB \wedge \neg\alpha)$ jest niespełnialne.
- Najpierw musimy przekonwertować $(KB \wedge \neg\alpha)$ do jego CNF.
- Następnie reguła rezolucji jest stosowana do wynikowych klauzul.
- Każda para, która zawiera dopełniające się literały jest 'rozwiązywana' w celu utworzenia nowej klauzuli, która jest dodawana do zbioru, dopóki taka para jeszcze istnieje.
- Proces trwa, dopóki nie wystąpi jeden z dwóch faktów.
  - Rozwiązywanie dwóch klauzul prowadzi do pustej klauzuli, w którym to przypadku $KB \models \alpha$.
  - Nie ma nowych klauzul, które można by uzyskać przez zastosowanie reguły rezolucji, w takim przypadku $KB \not\models \alpha$.

Pseudokod

- Załóżmy $KB = R_2 \wedge R_4 = (B_{1,1} \Leftrightarrow (P_{1,2} \vee P_{2,1})) \wedge \neg B_{1,1}$ i $\alpha = \neg P_{1,2}$.



Dowód przez zaprzeczenie, tzn. pokazanie, że $KB \wedge \neg\alpha$ niespełnialne.

function PL-RESOLUTION(*KB*, α) **returns** *true* or *false*
  clauses $\leftarrow$ the set of clauses in the CNF representation of $KB \wedge \neg\alpha$
  loop
    new $\leftarrow \{\}$
    for each C_i, C_j in *clauses* **do**
      resolvents $\leftarrow$ PL-RESOLVE(C_i, C_j)
      if *resolvents* contains the empty clause **then**
        return *true*
      end if
      new $\leftarrow$ *new* $\cup$ *resolvents*
    end for
    if *new* $\subseteq$ *clauses* **then**
      return *false*
    end if
    clauses $\leftarrow$ *clauses* $\cup$ *new*
  end loop
end function

Pełność reguły rezolucji

Chcemy dowieść, że PL-RESOLUTION jest pełny. Rozważamy, że S to dane zdanie w formie CNF.

- Resolution closure: $RC(S)$ jest zbiorem wszystkich klauzul pochodnych poprzez wielokrotne stosowanie reguły rezolucji do klauzul w S lub ich pochodnych.
- Twierdzenie o pełności (completeness theorem) dla rezolucji w rachunku zdań jest nazywane podstawowym (bazowym) twierdzeniem o rezolucji (ground resolution theorem):
- Jeśli zbiór klauzul S jest niespełnialne, wówczas $RC(S)$ zawiera pustą klauzulę.

Funkcja Boolowska

- Każde odwzorowanie $f : \{0,1\}^n \mapsto \{0,1\}$ nazywamy funkcja Boolowska.
- Funkcje Boolowskie $\approx$ formuły Boolowskie:
  - Kanoniczna postać sumy (DNF), np. $f = xy\overline{z} + x\overline{y}z + \overline{x}yz + xyz$
  - Kanoniczna postać iloczynu (CNF), np. $f = (x + y + z)(\overline{x} + y + z)(\overline{x} + y + \overline{z})$

Implikant funkcji boolowskiej f to iloczyn literałów taki, że dla wszystkich wektorów binarnych x=(x1, ... , xn), dla których jest on równy jedności, funkcja f jest równa jedności.

$t = x_{i_1} \dots x_{i_m} \overline{x_{j_1}} \dots \overline{x_{j_k}}$; t nazywamy implikantem funkcji f jeśli

$$\forall_{a \in \{0,1\}^n} \quad t(a) = 1 \Rightarrow f(a) = 1$$

Implikantem pierwszym nazywamy taki implikant, który przestaje nim być po usunięciu dowolnego literału.

Kanoniczna postać Blake'a: każdą funkcję Boolowską można przedstawić w postaci sumy wszystkich jej implikantów pierwszych:

$$f = t_1 + t_2 + \ldots + t_n$$

- Przykład: Rozważajmy $p \Rightarrow (q \wedge r)$

model	p	q	r	$\mathbf{p \Rightarrow q \wedge r}$
v_1	0	0	0	1
v_2	0	0	1	1
v_3	0	1	0	1
v_4	0	1	1	1
v_5	1	0	0	0
v_6	1	0	1	0
v_7	1	1	0	0
v_8	1	1	1	1

- CNF: $p \Rightarrow (q \wedge r) \equiv \neg p \vee (q \wedge r) \equiv (\neg p \vee q) \wedge (\neg p \vee r)$
- Kanoniczna postać iloczynu (CNF):
  $(\neg p \vee q) \wedge (\neg p \vee r) \equiv (\neg p \vee q \vee (r \wedge \neg r)) \wedge (\neg p \vee (q \wedge \neg q) \vee r)$ $\quad [\alpha \wedge \neg\alpha \equiv F \text{ i } \alpha \vee F \equiv \alpha]$
  $\equiv (\neg p \vee q \vee r) \wedge (\neg p \vee q \vee \neg r) \wedge (\neg p \vee q \vee r) \wedge (\neg p \vee \neg q \vee r)$
  $\equiv (\neg p \vee q \vee r) \wedge (\neg p \vee q \vee \neg r) \wedge (\neg p \vee \neg q \vee r)$
- Powyższe wyrażenie możemy również napisać jako:
  $(\overline{p} + q + r).(\overline{p} + q + \overline{r}).(\overline{p} + \overline{q} + r)$
- Kanoniczna postać sumy (DNF): Otrzymujemy to skupiając się na 1:
  $(\neg p \wedge \neg q \wedge \neg r) \vee (\neg p \wedge \neg q \wedge r) \vee (\neg p \wedge q \wedge \neg r) \vee (\neg p \wedge q \wedge r) \vee (p \wedge q \wedge r)$
- Powyższe wyrażenie możemy również napisać jako:
  $(\overline{p}.\overline{q}.\overline{r}) + (\overline{p}.\overline{q}.r) + (\overline{p}.q.\overline{r}) + (\overline{p}.q.r) + (p.q.r)$
- $(\overline{p}.\overline{q}.\overline{r})$, $(\overline{p}.\overline{q}.r)$, $(\overline{p}.q.\overline{r})$, $(\overline{p}.q.r)$, $(p.q.r)$ są implikanty.
- $(\overline{p}.\overline{q})$, $(\overline{p}.q)$, $(p.q.r)$ są pierwsze implikanty.

Dziękuję za uwagę