Negacyjna postać normalna

Marcin Kostrzewski, 444409

17 Czerwca, 2019r

1 Wprowadzenie

Ten program sprowadza formułę logiki w dowolnej postaci do negacyjnej postaci normalnej. Został napisany w języku **C++** i bazuje na różnych algorytmach i operacjach na stringach, czyli program bezpośrednio sprawdza i modyfikuje surowy tekst z formułą.

2 Negacyjna postać normalna

Negacyjna postać normalna formuły logicznej *(ang. negation normal form)*, to taka postać, w której negacja ($\neg$) stoi tylko przed zmiennymi, oraz jedynymi operatorami w formule jest koniunkcja ($\wedge$) i alternatywa ($\vee$).

3 Dlaczego C++?

Wybór tego języka uzasadniam tym, że jest jest mi on najbardziej znajomy. C++ znakomicie nadaje się do matematyki i w dodaku działa szybciej niż inne języki. Początkowo program miał zostać napisany w C, ale implementacja stringów w C++ zdecydowanie ułatwiała manipulację w przypadku mojego algorytmu. Standardowa biblioteka również dostarcza wiele przydatnych narzędzi, które również usprawniają pracę nad kodem oraz sprawiają, że jest on bardziej czytelny.

4 Algorytm

Program działa w oparciu o manipulacje na surowym tekście, którym jest formuła logiczna. Program nie będzie działał poprawnie, jeżeli formuła, którą wpiszemy będzie niepoprawna (np. jeżeli nawiasy nie będą się poprawnie domykać). Program ma również pewną limitację, mianowicie musimy explicite wziąć w nawiasy operacje, które mają wyższą moc wiązania, czyli:

$$p \vee q \Rightarrow r$$

Musimy zapisać jako:

$$(p \vee q) \Rightarrow r$$

Skracamy również zapisywanie predykatów do zapisania tylko jego symbolu, bez nawiasu i zmiennej. Zdecydowałem się na to uproszczenie, gdyż zdecydowanie ułatwia to pracę z algorytmem i przede wszystkim nie ma konieczności używania pełnego zapisu, bowiem sprowadzanie formuły logicznej do negacyjnej postaci normalej nie wpływa w żaden sposób na zmienne wolne, itp.
Zatem formułę:

$$\forall_{x \in X}(S(x) \wedge Q(x))$$

Musimy zapisać jako:

$$\forall(S \wedge Q)$$

Algorytm można podzielić na trzy główne funkcje:

- Zredukowanie operatorów
- Usunięcie podwójnych negacji
- Zastosowanie praw DeMorgana

Zredukowanie operatorów polega na wyszukiwaniu operatora do zamiany ($\Leftarrow$, $\Leftrightarrow$) i zastosowaniu odpowiedniego prawa logiki aby je przedstawić za pomocą ($\wedge$,$\vee$), czyli:

$$p \Rightarrow q \Leftrightarrow \neg p \vee q$$

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

Algorytm wyszukuje liniowo indeks operatora do zamiany i wywołuje procedurę zamiany. Operacje zamiany zamieniają fragment formuły i na inny fragment, gdzie operator został już przekształcony. Następnie algorytm zapętla się, aż wszystkie operatory zostaną zamienione.

Złożoność:

$$T(n,k) = \Theta(n * k)$$

n - długość formuły, k - ilość operatorów do zamiany

Usunięcie podwójnych negacji wyszukuje indeks najbliższej negacji i sprawdza, czy następny znak w tekście jest również negacją. Jeżeli tak, to jest ona usuwana z tekstu i algorytm zapętla się.

Złożoność:

$$T(n) = O(n)$$

n - długość formuły

Zastosowanie praw DeMorgana rozpoczyna od znalezienia indeksu najbliższej negacji. Następnie sprawdza, czy negacja jest przed nawiasem lub kwantyfikatorem. Jeżeli tak, to na fragmencie formuły stosuje prawa:

$$\neg\forall_{x\in X} S(x) \Leftrightarrow \exists_{x\in X}\neg S(x)$$

$$\neg\exists_{x\in X} S(x) \Leftrightarrow \forall_{x\in X}\neg S(x)$$

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

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

Złożoność:

$$T(n,k) = O(k * n^2)$$

n - długość formuły, k - ilość negacji

5 Wejście / Wyjście

Formułe do programu wpisujemy w jednej linii, nie używając spacji. Koniec formuły oznaczamy klawiszem ENTER.
Pamiętamy o wspomnianych wcześniej zasadach, czyli pozbywamy się zmiennych z predykatów i spod kwantyfikatorów i zapisujemy dodatkowe nawiasy.
Stosujemy również określony sposób zapisywania operatorów:

- $\neg p$ zaposujemy jako ! p
- $p \wedge q$ zapsujemy jako p ^ q
- $p \vee q$ zapisujemy jako pvq
- $p \Rightarrow q$ zapisujemy jako p>q
- $p \Leftrightarrow q$ zapisujemy jako p=q
- $\forall_{x\in X} S(x)$ zapisujemy jako VS
- $\exists_{x\in X} S(x)$ zapisujemy jako ES

Przykładowe wejście:

$$\neg\exists_{x\in X}(P(x) \vee Q(x) \Rightarrow P(x))$$

Zapiszemy jako:

$$!E((PvQ) > P)$$

Na wyjściu otrzymamy:

$$V((PvQ) \hat{} !P)$$

Odczytujemy jako:

$$\forall_{x\in X}(P(x) \vee Q(x) \Rightarrow \neg P(x))$$