

PROVER

Joanna Kulesza

Kamil Łopata

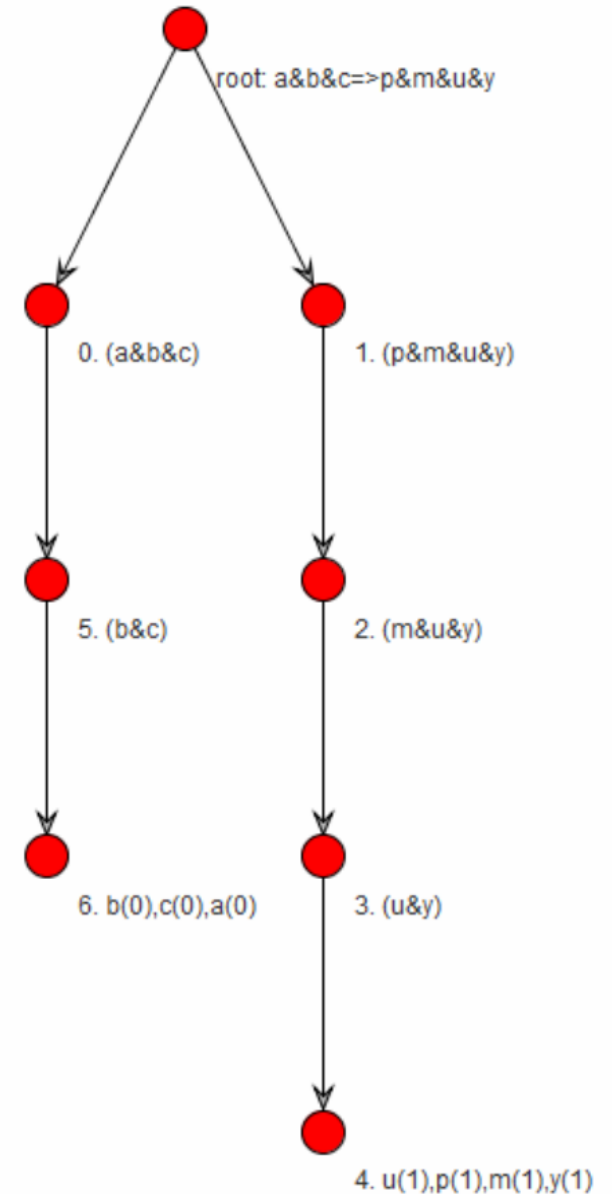
PROBLEM:

sprawdzenie poprawności
formuły logicznej

(z logiką temporalną)

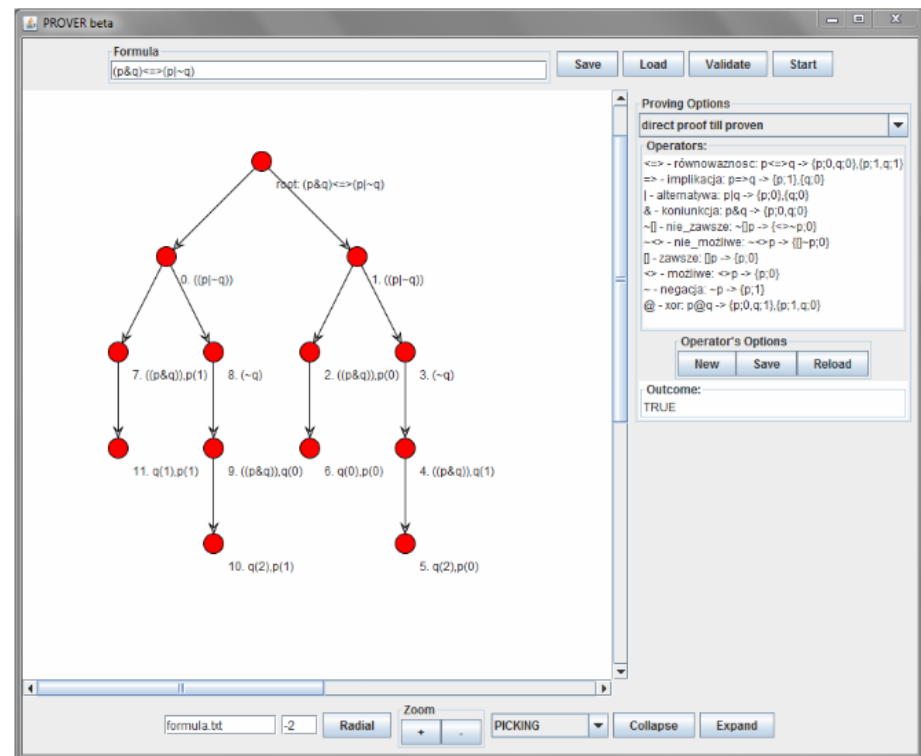
ROZWIĄZANIE:

**metoda
tablic semantycznych**



IMPLEMENTACJA:

JAVA
+
JUNG



Formula

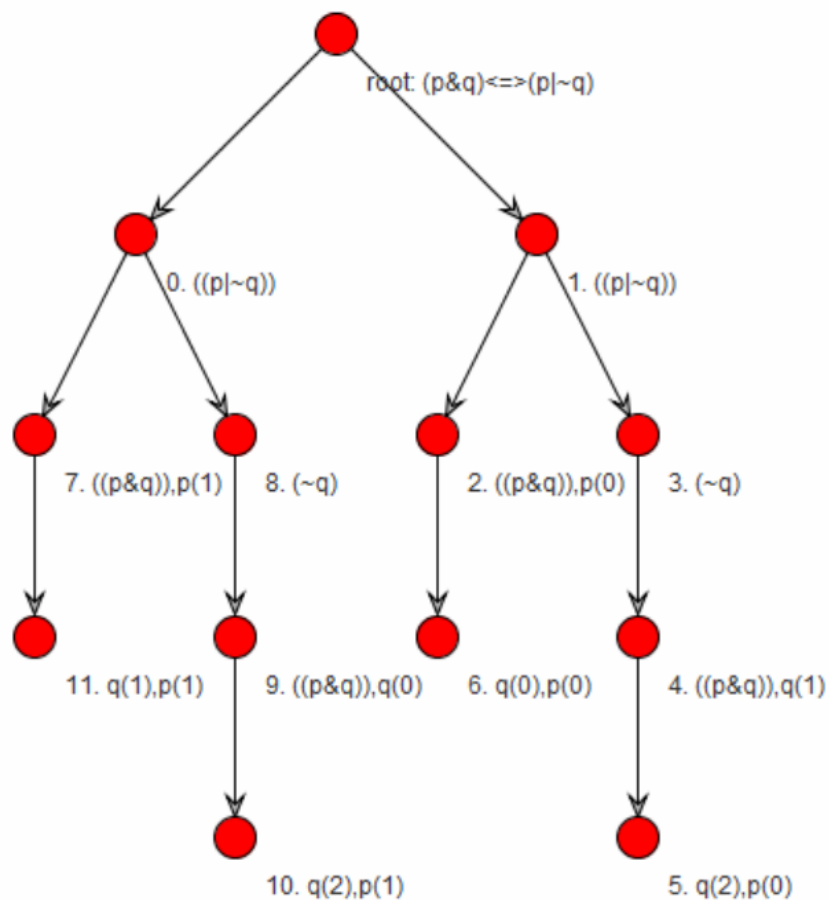
 $(p \& q) \Leftrightarrow (p | \sim q)$

Save

Load

Validate

Start



Proving Options

direct proof till proven

Operators:

 \Leftrightarrow - równowaznosc: $p \Leftrightarrow q \rightarrow \{p;0,q;0\}, \{p;1,q;1\}$ \Rightarrow - implikacja: $p \Rightarrow q \rightarrow \{p;1\}, \{q;0\}$ $|$ - alternatywa: $p | q \rightarrow \{p;0\}, \{q;0\}$ $\&$ - koniunkcja: $p \& q \rightarrow \{p;0,q;0\}$ $\sim[]$ - nie_zawsze: $\sim[]p \rightarrow \{\<p;0\}$ $\sim\<$ - nie_możliwe: $\sim\<p \rightarrow \{[]\sim p;0\}$ $[]$ - zawsze: $[]p \rightarrow \{p;0\}$ $\<$ - możliwe: $\<p \rightarrow \{p;0\}$ \sim - negacja: $\sim p \rightarrow \{p;1\}$ $@$ - xor: $p @ q \rightarrow \{p;0,q;1\}, \{p;1,q;0\}$

Operator's Options

New

Save

Reload

Outcome:

TRUE

formula.txt

-2

Radial

Zoom

+

-

PICKING

Collapse

Expand

jakie operatory zastosować?

logika klasyczna

negacja
koniunkcja
alternatywa
implikacja
równoważność

logika temporalna

zawsze
możliwe

& &

//

~

<=>

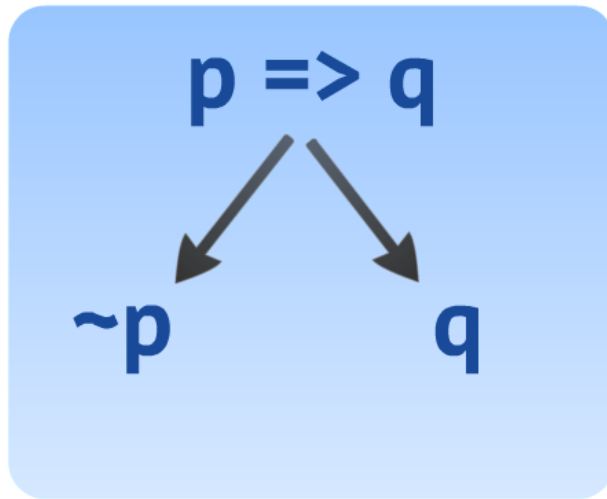
**jakie symbole
operatorów?**

AND

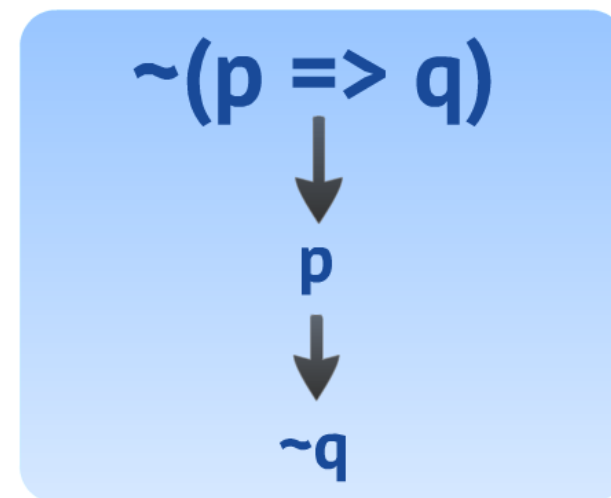
'

OR

=



jak rozłożyć działania?



jaka kolejność działań?

**ile użytkownik
może zmienić?**



WSZYSTKO!

**unikalny system
tworzenia nowych
operatorów**

dodawanie

edytowanie

usuwanie

zmiana kolejności

| | | | | |
|-----|--------------|---|---|------------------------------|
| => | implikacja | p | q | $\{p;1\}, \{q;0\}$ |
| <=> | równowaznosc | p | q | $\{p;0, q;0\}, \{p;1, q;1\}$ |
| | alternatywa | p | q | $\{p;0\}, \{q;0\}$ |
| & | koniunkcja | p | q | $\{p;0, q;0\}$ |
| ~[] | nie_zawsze | p | | $\{<>\sim p;0\}$ |
| ~<> | nie_możliwe | p | | $\{[]\sim p;0\}$ |
| [] | zawsze | p | | $\{p;0\}$ |
| <> | możliwe | p | | $\{p;0\}$ |
| ~ | negacja | p | | $\{p;1\}$ |

~ negacja p {p;1}

~p



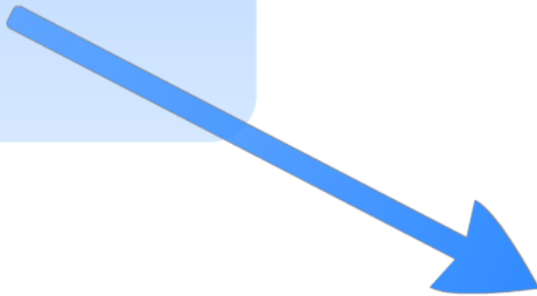
p;1

przyspieszenie
przetwarzania
negacji

$\sim p$



$p;1$



bit znaku

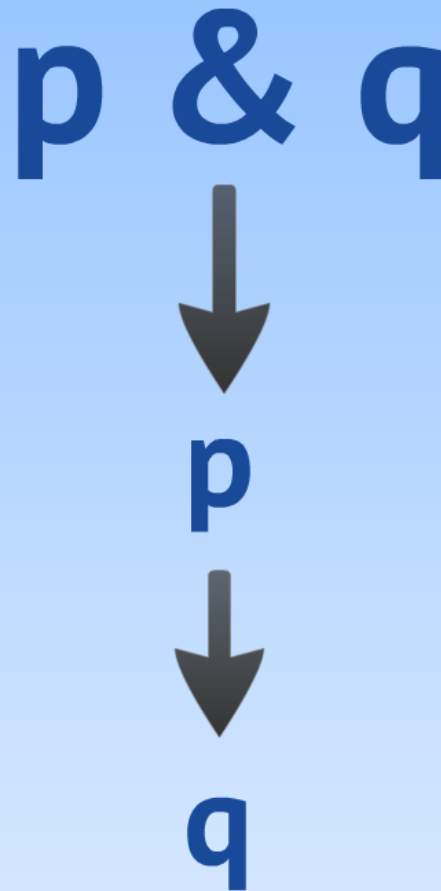
~ negacja p {p;1}

~p

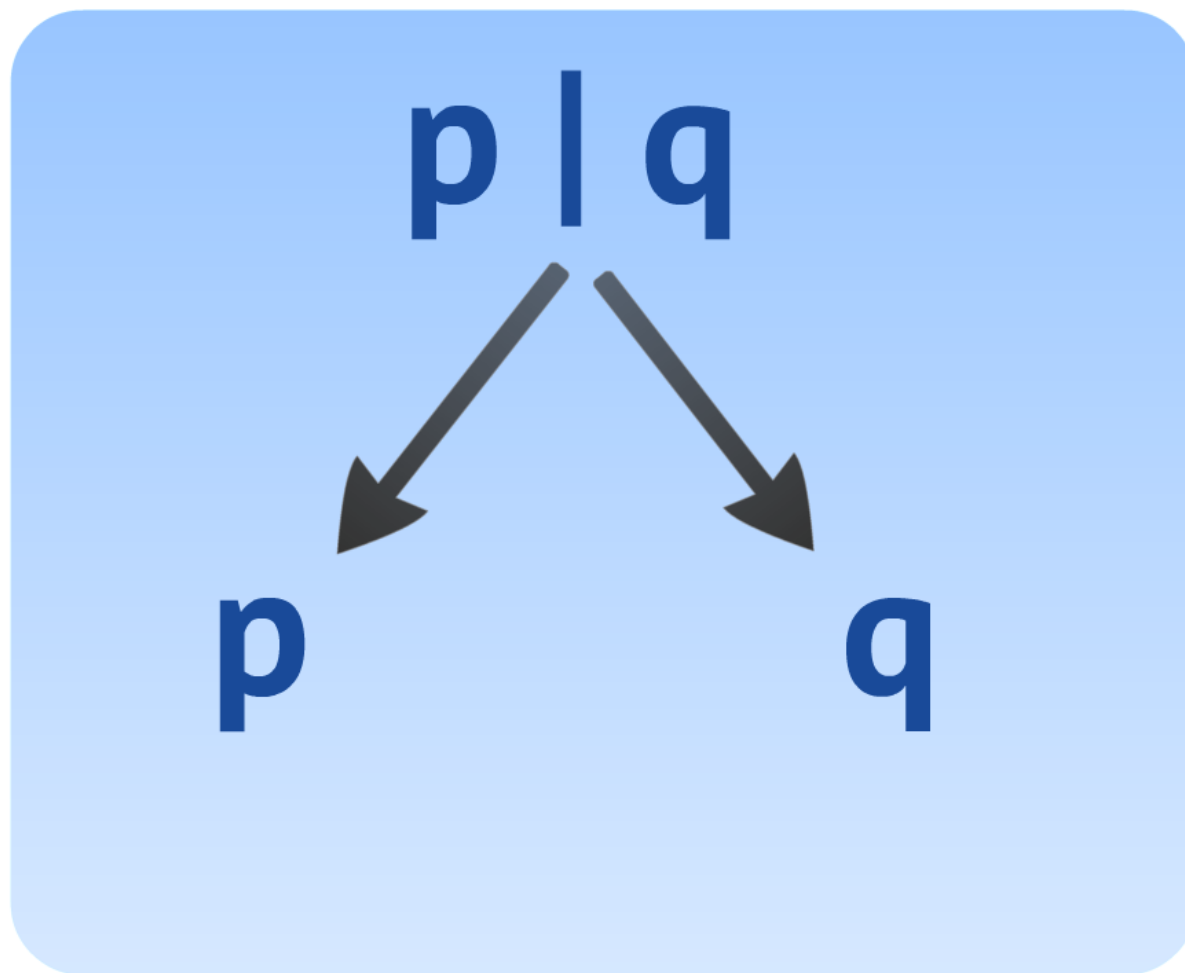


p;1

& koniunkcja p q {p;0,q;0}



| alternatywa p q {p;0},{q;0}

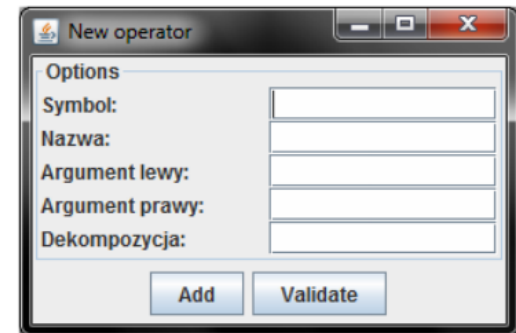


| | | | | |
|-----|--------------|---|---|---------------------|
| => | implikacja | p | q | {p;1},{q;0} |
| <=> | równowaznosc | p | q | {p;0,q;0},{p;1,q;1} |
| | alternatywa | p | q | {p;0},{q;0} |
| & | koniunkcja | p | q | {p;0,q;0} |
| ~[] | nie_zawsze | p | | {<>~p;0} |
| ~<> | nie_możliwe | p | | {[]~p;0} |
| [] | zawsze | p | | {p;0} |
| <> | możliwe | p | | {p;0} |
| ~ | negacja | p | | {p;1} |

prosto i szybko
stwórz własny operator!

tworzenie nowego operatora:

1. wybierz symbol
2. wybierz nazwę
3. wybierz nazwy argumentów
4. wpisz dekompozycje
5. kliknij 'Add'
6. **ciesz się nowym operatorem!**





New operator



Options

Symbol:

Nazwa:

Argument lewy:

Argument prawy:

Dekompozycja:

Add

Validate

**na operatorach świat się
nie kończy**

inne funkcjonalności

dowód klasyczny/nie wprost

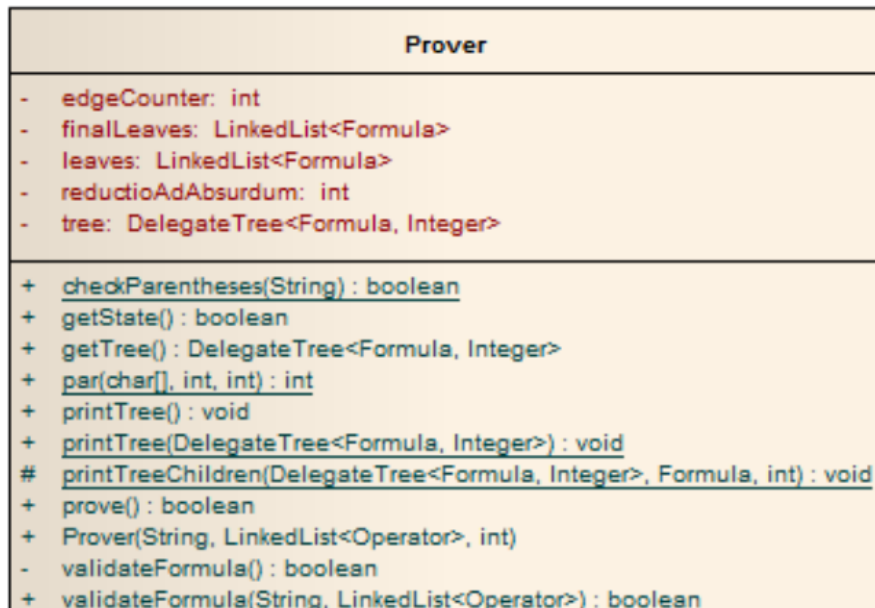
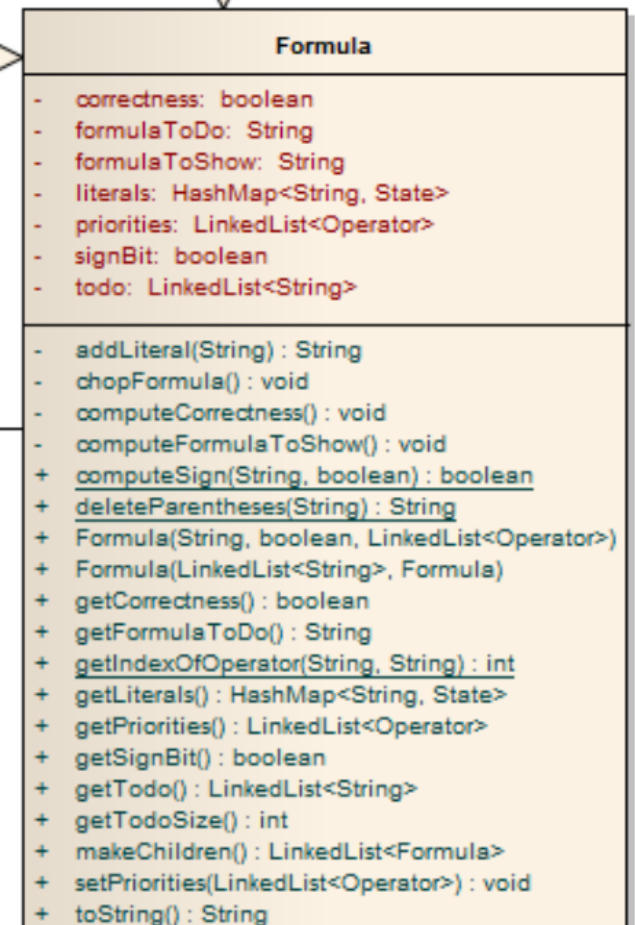
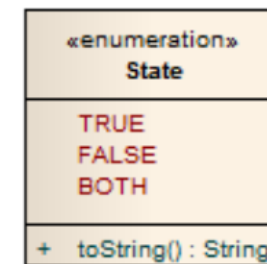
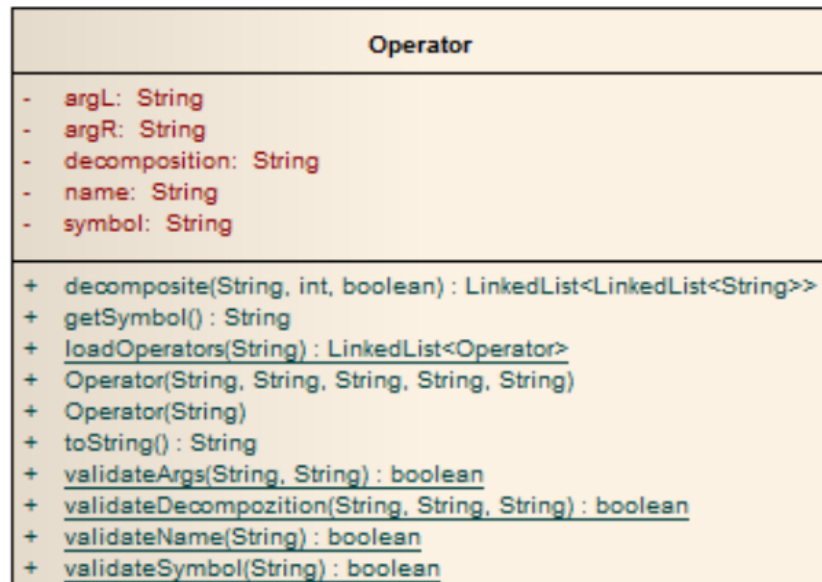
do udowodnienia / do końca

zwijanie wierzchołków

zapis/wczytanie formuły

zapis/wczytanie operatorów

class Class Model





**NOW YOU HAVE
THE POWER
TO PROVE**

**THE POWER TO PROVE
NOW YOU HAVE**



© (2010) Lucasfilm Ltd. & TM.