

Dokumentacja końcowa projektu z Podstaw Sztucznej Inteligencji:

0. Spis treści:

1. Skład zespołu
2. Zadanie
3. Treść zadania
 - 3.1. Maszyna wnioskująca
4. Podstawowe wymagania i uściślenia
5. Sposób reprezentacji danych
6. Konceptualne rozwiązanie problemu
7. Sposób działania algorytmu
 - 7.1. Struktury danych
 - 7.2. Podejście
 - 7.3. Działanie
8. Wynik działania algorytmu
 - 8.1. Wyjście
9. Uruchomienie
 - 9.1. Argumenty uruchomienia
 - 9.2. Przykład uruchomienia algorytmu
10. Przykład działania
 - 10.1 constants.txt
 - 10.2 variables.txt
 - 10.3 knowledge_base.txt
 - 10.4 argument.txt
 - 10.5 Wynik

1. Skład zespołu:

- Kurek Wojciech
- Belniak Michał
- Szachewicz Jakub

2. Zadanie:

- PW.W.2

3. Treść zadania:

3.1 Maszyna wnioskująca

Napisać program, który przyjmuje:

- formuły koniunkcyjno-implikacyjne bez negacji w rachunku predykatów pierwszego rzędu
- tezę do udowodnienia.

Następnie, sprowadza formuły do postaci klauzul, dalej do formuł koniunkcyjnoimplikacyjnych i przeprowadza wnioskowanie wstecz dowodzące zadanej tezy.

Program może nie obsługiwać funkcji, czyli może nie akceptować formuł postaci $PRED(F(x))$. Wynik wnioskowania zostaje zademonstrowany graficznie.

4. Podstawowe wymagania i uściślenia:

Projekt zakłada napisanie programu odczytującego formuły koniunkcyjno-implikacyjne z pliku tekstowego oraz dający w rezultacie swojego działania tekstowo-graficzną reprezentację wynikowego grafu powstałego w ramach procedury wnioskowania wstecz przeprowadzonego na podstawie odczytanych formuł koniunkcyjno-implikacyjnych.

5. Sposób reprezentacji danych:

Formuły koniunkcyjno-implikacyjne mają być zapisane za pomocą specjalnej notacji tekstowej pozwalającej reprezentować m.in. symbol implikacji (\Rightarrow) oraz symbol koniunkcji (\wedge).

6. Koceptualne rozwiązanie problemu:

Częścią składową programu jest parser umożliwiający odzwierciedlenie tekstowej reprezentacji koniunkcyjno-implikacyjnych w ramach modelu obiektowego.

Każda poprawnie zdefiniowana formuła ma zostać zmapowana na obiekt klasy `Closure`. Obiekty klasy `Closure` stanowią bazę wiedzy wykorzystywaną w procesie wnioskowania wstecz. Częścią programu jest obiektowa reprezentacja poszczególnych części formuły koniunkcyjno-implikacyjnej. Uwzględnione są typy podstawowe takich jak `model.Constant`, `model.Variable`.

Predykat jest reprezentowany przez klasę `model.Predicate` zawierającą w sobie obiekty typu `model.Constant` i `model.Variable`.

Nazwę predykatu traktujemy jako stałą (`model.Constant`). Reprezentację obiektową posiadają również operatory – AND i NOT (operator implikacji traktowane są oddzielnie).

Ich implementacja zakłada możliwość pobierania pierwszego operandu operacji oraz pozostałych operandów operatora. Operator implikacji jest zdefiniowany w postaci klasy `Closure`, zakłada on jako pierwszy operand obiekt klasy `model.Predicate`, drugi operand może być operacją lub predykatem.

7. Sposób działania algorytmu:

7.1 Struktury danych:

Implementacja algorytmu wnioskowania wstecz zakłada wykorzystanie struktury grafu.

7.2 Podejście:

Wykorzystywany jest mechanizm rekurencji.

7.3 Działanie

Kolejne stany algorytmu w których dla określonego (zaprzeczonego) predykatu poszukujemy formuły koniunkcyjno-implikacyjnej z unifikowalną z nim konkluzją reprezentujemy jako węzeł w grafie.

W kolejnym kroku algorytmu analizujemy ogon zdania, które zostało poddane unifikacji. W sytuacji, gdy ogon zdania był w postaci koniunkcji predykatów to też reprezentujemy go początkowo jako jeden węzeł, który później będzie rozbijany na następne węzły grafu poprzez kolejne wydzielanie pierwszego operandu z koniunkcji predykatów.

Z każdym węzłem jest związany zestaw niewykorzystanych jeszcze obiektów typu `Closure` (baza wiedzy), sumaryczny zestaw podstawień użyty do uzyskania danego węzła oraz id obiektu `Closure` w wyniku unifikacji którego powstał ten węzeł.

8. Wynik działania algorytmu:

8.1 Wyjście:

Program na wyjściu drukuje wynikowy graf powstały w ramach przebiegu algorytmu.

Istnieje konieczność zapewnienia możliwości znakowej reprezentacji poszczególnych węzłów grafu wynikowego. Wynik programu ma być wyświetlony w jak najprostszej i czytelnej dla użytkownika końcowego formie.

9. Uruchomienie:

9.1 Argumenty uruchomienia

Program przyjmuje dane od użytkownika w formie **paramterów uruchomieniowych** cztery parametry.

- Baza wiedzy w postaci formuł koniunkcyjno-implikacyjnych. Baza wiedzy jest przekazywana w formie pliku, do którego ścieżkę przekazujemy programowi parametrem `-knowledgeBase`. Kolejne formuły pisane jedna pod drugą. Implikacja oznaczona poprzez symbol `=>`.
- Użyte w bazie wiedzy stałe (w tym nazwy predykatów). Stałe są przekazywane w formie pliku, do którego ścieżkę przekazujemy programowi parametrem `-const` lub `-constants`. Kolejne stałe są pisane w pliku jedna pod drugą.
- Użyte w bazie wiedzy zmienne. Zmienne są przekazywane w formie pliku, do którego ścieżkę przekazujemy programowi parametrem `-var` lub `-variables`. Kolejne zmienne są pisane w pliku jedna pod drugą.
- Teza do udowodnienia przez algorytm. Teza jest przekazywana w formie pliku, do którego ścieżkę przekazujemy programowi parametrem `-arg` lub `-argument`. Waże by stałe oraz nazwa predykatu użyte w tezie były również uwzględnione w plikach ze stałymi.

9.2 Przykład uruchomienia algorytmu:

```
$ java -jar ./MaszynaWnioskujaca.jar -knowledgeBase knowledge_base.txt -var  
variables.txt -const constants.txt -arg argument.txt
```

10. Przykłady działania

Przykłady większe:

Przykład 6:

“The law says that it is a crime for an American to sell weapons to hostile nations. The country Nono, an enemy of America, has some missiles, and all of its missiles were sold to it by Colonel West, who is American.”

They said: **“Prove that Colonel West is a criminal”**

Knowledge Base:

- . . . it is a crime for an American to sell weapons to hostile nations:

$\text{American}(x) \wedge \text{Weapon}(y) \wedge \text{Sells}(x, y, z) \wedge \text{Hostile}(z) \Rightarrow \text{Criminal}(x)$

- Nono . . . has some missiles

$\text{Owns}(\text{Nono}, M1) \text{ and } \text{Missile}(M1)$

- . . . all of its missiles were sold to it by Colonel West

$\forall x \text{ Missile}(x) \wedge \text{Owns}(\text{Nono}, x) \Rightarrow \text{Sells}(\text{West}, x, \text{Nono})$

- Missiles are weapons:

$\text{Missile}(x) \Rightarrow \text{Weapon}(x)$

- An enemy of America counts as “hostile”:

$\text{Enemy}(x, \text{America}) \Rightarrow \text{Hostile}(x)$

- West, who is American . . .

$\text{American}(\text{West})$

- The country Nono, an enemy of America . . .

$\text{Enemy}(\text{Nono}, \text{America})$

Do udowodnienia:

$\text{Criminal}(\text{West})$

Pliki przekazywane programowi:

argument.txt

CRIMINAL (WEST)

knowledge_base.txt

(AMERICAN (x0) ^WEAPON (x1) ^SELLS (x0, x1, x2) ^HOSTILE (x2)) =>CRIMINAL (x0)
(MISSILE (x3) ^OWNS (NONO, x3)) =>SELLS (WEST, x3, NONO)
MISSILE (x4) =>WEAPON (x4)
ENEMY (x5, AMERICA) =>HOSTILE (x5)
AMERICAN (WEST)
ENEMY (NONO, AMERICA)
OWNS (NONO, M)
MISSILE (M)

variables.txt

x0
x1
x2
x3
x4
x5

constants.txt

AMERICAN
WEAPON
SELLS
HOSTILE
CRIMINAL
MISSILE
ENEMY
OWNS
NONO
M
WEST
AMERICA

Wynik działania algorytmu:

Baza wiedzy:

```
K1: AMERICAN(x0_0) ^ WEAPON(x1_1) ^ SELLS(x0_0, x1_1, x2_2) ^ HOSTILE(x2_2) =>
CRIMINAL(x0_0)
K2: MISSILE(x3_3) ^ OWNS(NONO, x3_3) => SELLS(WEST, x3_3, NONO)
K3: MISSILE(x4_4) => WEAPON(x4_4)
K4: ENEMY(x5_5, AMERICA) => HOSTILE(x5_5)
K5: AMERICAN(WEST)
K6: ENEMY(NONO, AMERICA)
K7: OWNS(NONO, M)
K8: MISSILE(M)
K9: WEAPON(x1_7) ^ SELLS(x0_7, x1_7, x2_7) ^ HOSTILE(x2_7) ^ ~CRIMINAL(x0_7) =>
~AMERICAN(x0_7)
K10: AMERICAN(x0_8) ^ SELLS(x0_8, x1_8, x2_8) ^ HOSTILE(x2_8) ^ ~CRIMINAL(x0_8) =>
~WEAPON(x1_8)
K11: AMERICAN(x0_9) ^ WEAPON(x1_9) ^ HOSTILE(x2_9) ^ ~CRIMINAL(x0_9) => ~SELLS(x0_9,
x1_9, x2_9)
K12: AMERICAN(x0_10) ^ WEAPON(x1_10) ^ SELLS(x0_10, x1_10, x2_10) ^ ~CRIMINAL(x0_10) =>
~HOSTILE(x2_10)
K13: OWNS(NONO, x3_11) ^ ~SELLS(WEST, x3_11, NONO) => ~MISSILE(x3_11)
K14: MISSILE(x3_12) ^ ~SELLS(WEST, x3_12, NONO) => ~OWNS(NONO, x3_12)
K15: ~WEAPON(x4_13) => ~MISSILE(x4_13)
K16: ~HOSTILE(x5_14) => ~ENEMY(x5_14, AMERICA)
```

Teza: CRIMINAL(WEST)

Zbiór podstawień: {x0_0/WEST, x5_5/NONO, x4_4/M, x3_3/x4_4, x1_1/x4_4, x2_2/NONO}

Użyte klauzule:

```
K6: ENEMY(NONO, AMERICA)
K4: ENEMY(x5_5, AMERICA) => HOSTILE(x5_5)
K7: OWNS(NONO, M)
K8: MISSILE(M)
K2: MISSILE(x3_3) ^ OWNS(NONO, x3_3) => SELLS(WEST, x3_3, NONO)
K8: MISSILE(M)
K3: MISSILE(x4_4) => WEAPON(x4_4)
K5: AMERICAN(WEST)
K1: AMERICAN(x0_0) ^ WEAPON(x1_1) ^ SELLS(x0_0, x1_1, x2_2) ^ HOSTILE(x2_2) =>
CRIMINAL(x0_0)
```

Graf wnioskowania:

```
(CRIMINAL(WEST))
^
|
(AMERICAN(WEST) ^ WEAPON(M) ^ SELLS(WEST, M, NONO) ^ HOSTILE(NONO))
^
|
(AMERICAN(WEST))
^
|
(MISSILE(M))
^
|
(MISSILE(M) ^ OWNS(NONO, M))
^
|
(MISSILE(M))
^
|
(OWNS(NONO, M))
^
|
(ENEMY(NONO, AMERICA))
```

Przykład 3

“Każdy kocha wspinaczkę lub narciarstwo. Miłośnicy wspinaczki nie lubią deszczu, miłośnicy narciarstwa zaś lubią śnieg. Abacki lubi to, czego nie lubi Babacki, zaś Babacki to, czego nie lubi Abacki. Abacki lubi deszcz oraz śnieg.”

Powiedzieli: „**Udowodnij, że Babacki kocha Wspinaczkę**”

Knowledge Base:

- Każdy kocha wspinaczkę lub narciarstwo

$\sim \text{Kocha}(x, \text{Wspin}) \Rightarrow \text{Kocha}(x, \text{Narty})$

- Miłośnicy wspinaczki nie lubią deszczu,

$\text{Kocha}(x, \text{Wspin}) \Rightarrow \sim \text{Lubi}(x, \text{Deszcz})$

- . . . miłośnicy narciarstwa zaś lubią śnieg.

$\text{Kocha}(x, \text{Narty}) \Rightarrow \text{Lubi}(x, \text{Śnieg})$

- Abacki lubi to, czego nie lubi Babacki,

$\text{Lubi}(\text{Abacki}, x) \Rightarrow \sim \text{Lubi}(\text{Babacki}, x)$

- . . . zaś Babacki to, czego nie lubi Abacki.

$\sim \text{Lubi}(\text{Babacki}, x) \Rightarrow \text{Lubi}(\text{Abacki}, x)$

- Abacki lubi deszcz. . .

$\text{Lubi}(\text{Abacki}, \text{Deszcz})$

- Abacki lubi (. . .) śnieg.

$\text{Lubi}(\text{Abacki}, \text{Śnieg})$

Do udowodnienia:

$\text{Kocha}(\text{Babacki}, \text{Wspin})$

Pliki przekazywane programowi:

argument.txt

KOCHA (Babacki, Wspin)

knowledge_base.txt

~KOCHA (x0, Wspin) => KOCHA (x0, Narty)
KOCHA (x1, Wspin) => ~LUBI (x1, Descz)
KOCHA (x2, Narty) => LUBI (x2, Snieg)
LUBI (Abacki, x3) => ~LUBI (Babacki, x3)
~LUBI (Babacki, x4) => LUBI (Abacki, x4)
LUBI (Abacki, Descz)
LUBI (Abacki, Snieg)

variables.txt

x0
x1
x2
x3
x4

constants.txt

KOCHA
LUBI
~KOCHA
~LUBI
Wspin
Narty
Descz
Snieg
Abacki
Babacki

Baza wiedzy:

```

K1: ~KOCHA(x0_0, Wspin_4) => KOCHA(x0_0, Narty_5)
K2: KOCHA(x1_1, Wspin_4) => ~LUBI(x1_1, Descz_6)
K3: KOCHA(x2_2, Narty_5) => LUBI(x2_2, Snieg_7)
K4: LUBI(Abacki_8, x3_3) => ~LUBI(Babacki_9, x3_3)
K5: ~LUBI(Babacki_9, x4_4) => LUBI(Abacki_8, x4_4)
K6: LUBI(Abacki_8, Descz_6)
K7: LUBI(Abacki_8, Snieg_7)
K8: ~KOCHA(x0_6, Narty_5) => KOCHA(x0_6, Wspin_4)
K9: LUBI(x1_7, Descz_6) => ~KOCHA(x1_7, Wspin_4)
K10: ~LUBI(x2_8, Snieg_7) => ~KOCHA(x2_8, Narty_5)
K11: LUBI(Babacki_9, x3_9) => ~LUBI(Abacki_8, x3_9)
K12: ~LUBI(Abacki_8, x4_10) => LUBI(Babacki_9, x4_10)

```

Zbiór podstawień: {x0_0/Abacki_8, x2_2/Abacki_8, x3_3/Snieg_7, x1_7/Abacki_8, x0_6/Babacki_9, x2_8/Babacki_9}

```
K6: LUBI(Abacki_8, Descz_6)
K9: LUBI(x1_7, Descz_6) => ~KOCHA(x1_7, Wspin_4)
K1: ~KOCHA(x0_0, Wspin_4) => KOCHA(x0_0, Narty_5)
K3: KOCHA(x2_2, Narty_5) => LUBI(x2_2, Snieg_7)
K4: LUBI(Abacki_8, x3_3) => ~LUBI(Babacki_9, x3_3)
K10: ~LUBI(x2_8, Snieg_7) => ~KOCHA(x2_8, Narty_5)
K8: ~KOCHA(x0_6, Narty_5) => KOCHA(x0_6, Wspin_4)
```

$$\begin{array}{c} (\text{KOCHA}(\text{Babacki_9}, \text{Wspin_4})) \\ \wedge \\ | \\ (\sim \text{KOCHA}(\text{Babacki_9}, \text{Narty_5})) \\ \wedge \\ | \\ (\sim \text{LUBI}(\text{Babacki_9}, \text{Snieg_7})) \\ \wedge \\ | \\ (\text{LUBI}(\text{Abacki_8}, \text{Snieg_7})) \\ \wedge \\ | \\ (\text{KOCHA}(\text{Abacki_8}, \text{Narty_5})) \\ \wedge \\ | \\ (\sim \text{KOCHA}(\text{Abacki_8}, \text{Wspin_4})) \\ \wedge \\ | \\ (\text{LUBI}(\text{Abacki_8}, \text{Descz_6})) \end{array}$$

Przykład 7

“Wolves, foxes, birds, caterpillars, and snails are animals. Also there are some grains, and grains are plants. Every animal either likes to eat plants or animals much smaller than itself that like to eat some plants. Caterpillars and snails are much smaller than birds, which are much smaller than foxes, which are in turn much smaller than wolves. Birds do not like to eat grain. Wolves do not like to eat foxes or grains, while birds do not like to eat snails. Snails like to eat some plants and caterpillars like to eat grains. We know, that Animal1 is a bird, Grain1 is a grain and Animal2 is a caterpillar.”

They said: **“Prove, that Animal1 eats Animal2.”**

Knowledge Base:

- Wolves, (. .) are animals.

Wolf(x) \Rightarrow Animal(x)

- . . . foxes, (. .) are animals.

Fox(x) \Rightarrow Animal(x)

- (. .) birds (. .) are animals.

Bird(x) \Rightarrow Animal(x)

- (. .) caterpillars, (. .) are animals.

Caterpillar(x) \Rightarrow Animal(x)

- (. .) and snails are animals.

Snail(x) \Rightarrow Animal(x)

- , and grains are plants(. .)

Grain(x) \Rightarrow Plant(x)

- Every animal either likes to eat plants or animals much smaller than itself that like to eat some plants.

$$\text{Animal}(x) \wedge \text{Plant}(y) \wedge \sim \text{Eats}(x,y) \wedge \text{Animal}(z) \wedge \text{Smaller}(z,x) \wedge \text{Plant}(u) \wedge \text{Eats}(z,u) \Rightarrow \text{Eats}(x,z)$$

- Caterpillars and snails are much smaller than birds, which are much smaller than foxes, which are in turn much smaller than wolves.

$$\text{Caterpillar}(x) \wedge \text{Bird}(y) \Rightarrow \text{Smaller}(x,y)$$

$$\text{Snail}(x) \wedge \text{Bird}(y) \Rightarrow \text{Smaller}(x,y)$$

$$\text{Bird}(x) \wedge \text{Fox}(y) \Rightarrow \text{Smaller}(x,y)$$

$$\text{Fox}(x) \wedge \text{Wolf}(y) \Rightarrow \text{Smaller}(x,y)$$

- Birds do not like to eat grain.

$$\text{Bird}(x) \wedge \text{Grain}(y) \Rightarrow \sim \text{Eats}(x,y)$$

- caterpillars like to eat grains

$$\text{Caterpillar}(x) \wedge \text{Grain}(y) \Rightarrow \text{Eats}(x,y)$$

- Snails like to eat some plants,

$$\text{Snail}(x) \wedge \text{Plant}(y) \Rightarrow \text{Eats}(x,y)$$

- Wolves do not like to eat foxes (. . .)

$$\text{Wolf}(x) \wedge \text{Fox}(y) \Rightarrow \sim \text{Eats}(x,y)$$

- Wolves do not like to eat (. . .) grains,

$$\text{Wolf}(x) \wedge \text{Grain}(y) \Rightarrow \sim \text{Eats}(x,y)$$

- while birds do not like to eat snails.

$$\text{Bird}(x) \wedge \text{Snail}(y) \Rightarrow \sim \text{Eats}(x,y)$$

- Animal1 is a bird

Bird(Animal1)

- Grain1 is a grain

Grain(Grain1)

- Animal2 is a caterpillar.

Caterpillar(Animal2)

Do udowodnienia:

Eats(Animal1, Animal2)

Pliki przekazywane programowi:

argument.txt

```
EATS(Animal1,Animal2)
```

knowledge_base.txt

```
WOLF(x1)=>ANIMAL(x1)
FOX(x2)=>ANIMAL(x2)
BIRD(x3)=>ANIMAL(x3)
CATERPILLAR(x4)=>ANIMAL(x4)
SNAIL(x5)=>ANIMAL(x5)
GRAIN(x6)=>PLANT(x6)
GRAIN(x7)=>PLANT(x7)
(ANIMAL(x8)^PLANT(y8)^~EATS(x8,y8)^ANIMAL(z8)^SMALLER(z8,x8)^PLANT(u8)^EATS(z8,u8))=>EATS(x8,z8)
(CATERPILLAR(x9)^BIRD(y9))=>SMALLER(x9,y9)
(SNAIL(x10)^BIRD(y10))=>SMALLER(x10,y10)
(BIRD(x11)^FOX(y11))=>SMALLER(x11,y11)
(FOX(x13)^WOLF(y13))=>SMALLER(x13,y13)
(BIRD(x14)^GRAIN(y14))=>~EATS(x14,y14)
(CATERPILLAR(x15)^GRAIN(y15))=>EATS(x15,y15)
(SNAIL(x16)^PLANT(y16))=>EATS(x16,y16)
(WOLF(x17)^FOX(y17))=>~EATS(x17,y17)
(WOLF(x18)^GRAIN(y18))=>~EATS(x18,y18)
(BIRD(x19)^SNAIL(y19))=>~EATS(x19,y19)
BIRD(Animal1)
GRAIN(Grain1)
CATERPILLAR(Animal2)
```

variables.txt

```
x0
x1
x2
x3
x4
x5
x6
x7
x8
x9
x10
x11
x13
x14
x15
x16
x17
x18
x19
y8
y9
y10
y11
y13
y14
y15
y16
```

y17
y18
y19
z8
u8

constants.txt

WOLF
FOX
BIRD
CATERPILLAR
SNAIL
GRAIN
ANIMAL
EATS
~EATS
SMALLER
PLANT
Animal1
Animal2
Grain1

Rezultat działania programu:

Baza wiedzy:

K1: WOLF(x1_1) => ANIMAL(x1_1)
K2: FOX(x2_2) => ANIMAL(x2_2)
K3: BIRD(x3_3) => ANIMAL(x3_3)
K4: CATERPILLAR(x4_4) => ANIMAL(x4_4)
K5: SNAIL(x5_5) => ANIMAL(x5_5)
K6: GRAIN(x6_6) => PLANT(x6_6)
K7: GRAIN(x7_7) => PLANT(x7_7)
K8: ANIMAL(x8_8) ^ PLANT(y8_19) ^ ~EATS(x8_8, y8_19) ^ ANIMAL(z8_30) ^ SMALLER(z8_30, x8_8) ^ PLANT(u8_31) ^ EATS(z8_30, u8_31) => EATS(x8_8, z8_30)
K9: CATERPILLAR(x9_9) ^ BIRD(y9_20) => SMALLER(x9_9, y9_20)
K10: SNAIL(x10_10) ^ BIRD(y10_21) => SMALLER(x10_10, y10_21)
K11: BIRD(x11_11) ^ FOX(y11_22) => SMALLER(x11_11, y11_22)
K12: FOX(x13_12) ^ WOLF(y13_23) => SMALLER(x13_12, y13_23)
K13: BIRD(x14_13) ^ GRAIN(y14_24) => ~EATS(x14_13, y14_24)
K14: CATERPILLAR(x15_14) ^ GRAIN(y15_25) => EATS(x15_14, y15_25)
K15: SNAIL(x16_15) ^ PLANT(y16_26) => EATS(x16_15, y16_26)
K16: WOLF(x17_16) ^ FOX(y17_27) => ~EATS(x17_16, y17_27)
K17: WOLF(x18_17) ^ GRAIN(y18_28) => ~EATS(x18_17, y18_28)
K18: BIRD(x19_18) ^ SNAIL(y19_29) => ~EATS(x19_18, y19_29)
K19: BIRD(Animal1_11)
K20: GRAIN(Grain1_13)
K21: CATERPILLAR(Animal2_12)
K22: ~ANIMAL(x1_33) => ~WOLF(x1_33)
K27: ~PLANT(x6_38) => ~GRAIN(x6_38)
K29: PLANT(y8_40) ^ ~EATS(x8_40, y8_40) ^ ANIMAL(z8_40) ^ SMALLER(z8_40, x8_40) ^ PLANT(u8_40) ^ EATS(z8_40, u8_40) ^ ~EATS(x8_40, z8_40) => ~ANIMAL(x8_40)
K30: ANIMAL(x8_41) ^ ~EATS(x8_41, y8_41) ^ ANIMAL(z8_41) ^ SMALLER(z8_41, x8_41) ^ PLANT(u8_41) ^ EATS(z8_41, u8_41) ^ ~EATS(x8_41, z8_41) => ~PLANT(y8_41)
K31: ANIMAL(x8_42) ^ PLANT(y8_42) ^ ANIMAL(z8_42) ^ SMALLER(z8_42, x8_42) ^ PLANT(u8_42) ^ EATS(z8_42, u8_42) ^ ~EATS(x8_42, z8_42) => EATS(x8_42, y8_42)
K32: ANIMAL(x8_43) ^ PLANT(y8_43) ^ ~EATS(x8_43, y8_43) ^ SMALLER(z8_43, x8_43) ^ PLANT(u8_43) ^ EATS(z8_43, u8_43) ^ ~EATS(x8_43, z8_43) => ~ANIMAL(z8_43)
K33: ANIMAL(x8_44) ^ PLANT(y8_44) ^ ~EATS(x8_44, y8_44) ^ ANIMAL(z8_44) ^ PLANT(u8_44) ^ EATS(z8_44, u8_44) ^ ~EATS(x8_44, z8_44) => ~SMALLER(z8_44, x8_44)
K34: ANIMAL(x8_45) ^ PLANT(y8_45) ^ ~EATS(x8_45, y8_45) ^ ANIMAL(z8_45) ^ SMALLER(z8_45, x8_45) ^ EATS(z8_45, u8_45) ^ ~EATS(x8_45, z8_45) => ~PLANT(u8_45)
K35: ANIMAL(x8_46) ^ PLANT(y8_46) ^ ~EATS(x8_46, y8_46) ^ ANIMAL(z8_46) ^ SMALLER(z8_46, x8_46) ^ PLANT(u8_46) ^ ~EATS(x8_46, z8_46) => ~EATS(z8_46, u8_46)

K36: BIRD(y9_47) ^ ~SMALLER(x9_47, y9_47) => ~CATERPILLAR(x9_47)
 K37: CATERPILLAR(x9_48) ^ ~SMALLER(x9_48, y9_48) => ~BIRD(y9_48)
 K39: SNAIL(x10_50) ^ ~SMALLER(x10_50, y10_50) => ~BIRD(y10_50)
 K40: FOX(y11_51) ^ ~SMALLER(x11_51, y11_51) => ~BIRD(x11_51)
 K42: WOLF(y13_53) ^ ~SMALLER(x13_53, y13_53) => ~FOX(x13_53)
 K44: GRAIN(y14_55) ^ EATS(x14_55, y14_55) => ~BIRD(x14_55)
 K45: BIRD(x14_56) ^ EATS(x14_56, y14_56) => ~GRAIN(y14_56)
 K46: GRAIN(y15_57) ^ ~EATS(x15_57, y15_57) => ~CATERPILLAR(x15_57)
 K47: CATERPILLAR(x15_58) ^ ~EATS(x15_58, y15_58) => ~GRAIN(y15_58)
 K49: SNAIL(x16_60) ^ ~EATS(x16_60, y16_60) => ~PLANT(y16_60)
 K50: FOX(y17_61) ^ EATS(x17_61, y17_61) => ~WOLF(x17_61)
 K51: WOLF(x17_62) ^ EATS(x17_62, y17_62) => ~FOX(y17_62)
 K54: SNAIL(y19_65) ^ EATS(x19_65, y19_65) => ~BIRD(x19_65)

Teza: EATS(Animal1_11, Animal2_12)

Zbiór podstawień: {x15_14/Animal2_12, x9_9/Animal2_12, x4_4/Animal2_12, y8_19/x6_6, u8_31/x7_7, y9_20/Animal1_11, x3_3/Animal1_11, x6_6/Grain1_13, x8_8/Animal1_11, x14_13/Animal1_11, y15_25/x7_7, y14_24/x6_6, x7_7/Grain1_13, z8_30/Animal2_12}

Użyte klauzule:

K20: GRAIN(Grain1_13)
 K21: CATERPILLAR(Animal2_12)
 K14: CATERPILLAR(x15_14) ^ GRAIN(y15_25) => EATS(x15_14, y15_25)
 K20: GRAIN(Grain1_13)
 K7: GRAIN(x7_7) => PLANT(x7_7)
 K19: BIRD(Animal1_11)
 K21: CATERPILLAR(Animal2_12)
 K9: CATERPILLAR(x9_9) ^ BIRD(y9_20) => SMALLER(x9_9, y9_20)
 K21: CATERPILLAR(Animal2_12)
 K4: CATERPILLAR(x4_4) => ANIMAL(x4_4)
 K20: GRAIN(Grain1_13)
 K19: BIRD(Animal1_11)
 K13: BIRD(x14_13) ^ GRAIN(y14_24) => ~EATS(x14_13, y14_24)
 K20: GRAIN(Grain1_13)
 K6: GRAIN(x6_6) => PLANT(x6_6)
 K19: BIRD(Animal1_11)
 K3: BIRD(x3_3) => ANIMAL(x3_3)
 K8: ANIMAL(x8_8) ^ PLANT(y8_19) ^ ~EATS(x8_8, y8_19) ^ ANIMAL(z8_30) ^ SMALLER(z8_30, x8_8) ^ PLANT(u8_31) ^ EATS(z8_30, u8_31) => EATS(x8_8, z8_30)

Graf wnioskowania:

(EATS(Animal1_11, Animal2_12))
 ^
 |
 (ANIMAL(Animal1_11) ^ PLANT(Grain1_13) ^ ~EATS(Animal1_11, Grain1_13) ^
 ANIMAL(Animal2_12) ^ SMALLER(Animal2_12, Animal1_11) ^ PLANT(Grain1_13) ^
 EATS(Animal2_12, Grain1_13))
 ^
 |
 (BIRD(Animal1_11))
 ^
 |
 (GRAIN(Grain1_13))
 ^
 |
 (BIRD(Animal1_11) ^ GRAIN(Grain1_13))
 ^
 |
 (BIRD(Animal1_11))
 ^
 |
 (GRAIN(Grain1_13))

^
|
(CATERPILLAR(Animal2_12))
^
|
(CATERPILLAR(Animal2_12) ^ BIRD(Animal1_11))
^
|
(CATERPILLAR(Animal2_12))
^
|
(BIRD(Animal1_11))
^
|
(GRAIN(Grain1_13))
^
|
(CATERPILLAR(Animal2_12) ^ GRAIN(Grain1_13))
^
|
(CATERPILLAR(Animal2_12))
^
|
(GRAIN(Grain1_13))

10.2. Przykłady mniejsze:

Przykład 1

Pliki przekazywane programowi:

argument.txt

ZAWOD (Cabacki, Inf)

knowledge_base.txt

TS (x0, x0)

TS (x1, x2) => TS (x2, x1)

~TS (Abacki, Babacki)

~TS (Abacki, Cabacki)

~TS (Babacki, Cabacki)

~TS (Architekt, Inf)

~TS (Architekt, Lekarz)

~TS (Inf, Lekarz)

~TS (Muzyka, Szachy)

~TS (Muzyka, Wedka)

~TS (Szachy, Wedka)

(~ZAWOD (x3, Lekarz) ^ ~ZAWOD (x3, Inf)) => ZAWOD (x3, Architekt)

(ZAWOD (x4, x6) ^ ZAWOD (x5, x6)) => TS (x4, x5)

(ZAWOD (x7, x8) ^ ZAWOD (x7, x9)) => TS (x8, x9)

(~HOBBY (x10, Muzyka) ^ ~HOBBY (x10, Szachy)) => HOBBY (x10, Wedka)

(HOBBY (x11, x13) ^ HOBBY (x12, x13)) => TS (x11, x12)

(HOBBY (x14, x15) ^ HOBBY (x14, x16)) => TS (x15, x16)

ZAWOD (Abacki, Lekarz)

ZAWOD (x17, Architekt) => HOBBY (x17, Szachy)

HOBBY (Cabacki, Muzyka)

variables.txt

x0

x1

x2

x3

x4

x5

constants.txt

TS

~TS

~ZAWOD

ZAWOD

HOBBY

~HOBBY

Abacki

Babacki

Cabacki

Lekarz

Inf

Architekt

Muzyka

Szachy

Wedka

Wynik działania programu:

Argument can not be proved by algorithm.

Przykład 2

Pliki przekazywane programowi:

argument.txt

MUZYK (Babacki)

knowledge_base.txt

```
(UMIE_NUTY(x0) ^ GRA_NA(x0,x1)) => MUZYK(x0)
(~UMIE_NUTY(x2) ^ GRA_NA(x2,x3)) => AMATOR(x2)
AMATOR(Abacki) => MUZYK(Babacki)
GRA_NA(Babacki, Trabka)
GRA_NA(Abacki, Gitara)
UMIE_TEORIE(x4) => UMIE_NUTY(x4)
UMIE_NUTY(x5) => UMIE_TEORIE(x5)
~UMIE_TEORIE(Abacki)
OSOBA(Abacki)
OSOBA(Babacki)
INSTRUMENT(Trabka)
INSTRUMENT(Gitara)
GRA_NA(x6,x7) => INSTRUMENT(x7)
GRA_NA(x8,x9) => OSOBA(x8)
```

variables.txt

```
x0
x1
x2
x3
x4
x5
x6
x7
x8
x9
```

constants.txt

```
UMIE_NUTY
~UMIE_NUTY
GRA_NA
MUZYK
AMATOR
UMIE_TEORIE
~UMIE_TEORIE
OSOBA
INSTRUMENT
Abacki
Babacki
Trabka
Gitara
```

Wynik działania programu:

Baza wiedzy:

```

K1: UMIE_NUTY(x0_0) ^ GRA_NA(x0_0, x1_1) => MUZYK(x0_0)
K2: ~UMIE_NUTY(x2_2) ^ GRA_NA(x2_2, x3_3) => AMATOR(x2_2)
K3: AMATOR(Abacki_9) => MUZYK(Babacki_10)
K4: GRA_NA(Babacki_10, Trabka_11)
K5: GRA_NA(Abacki_9, Gitara_12)
K6: UMIE_TEORIE(x4_4) => UMIE_NUTY(x4_4)
K7: UMIE_NUTY(x5_5) => UMIE_TEORIE(x5_5)
K8: ~UMIE_TEORIE(Abacki_9)
K9: OSOBA(Abacki_9)
K10: OSOBA(Babacki_10)
K11: INSTRUMENT(Trabka_11)
K12: INSTRUMENT(Gitara_12)
K13: GRA_NA(x6_6, x7_7) => INSTRUMENT(x7_7)
K14: GRA_NA(x8_8, x9_9) => OSOBA(x8_8)
K15: GRA_NA(x0_11, x1_11) ^ ~MUZYK(x0_11) => ~UMIE_NUTY(x0_11)
K16: UMIE_NUTY(x0_12) ^ ~MUZYK(x0_12) => ~GRA_NA(x0_12, x1_12)
K17: GRA_NA(x2_13, x3_13) ^ ~AMATOR(x2_13) => UMIE_NUTY(x2_13)
K18: ~UMIE_NUTY(x2_14) ^ ~AMATOR(x2_14) => ~GRA_NA(x2_14, x3_14)
K19: ~MUZYK(Babacki_10) => ~AMATOR(Abacki_9)
K20: ~UMIE_NUTY(x4_16) => ~UMIE_TEORIE(x4_16)
K21: ~UMIE_TEORIE(x5_17) => ~UMIE_NUTY(x5_17)
K22: ~INSTRUMENT(x7_18) => ~GRA_NA(x6_18, x7_18)
K23: ~OSOBA(x8_19) => ~GRA_NA(x8_19, x9_19)

```

Teza: MUZYK (Babacki 10)

Zbiór podstawień: {x3 3/Gitara 12, x5 17/Abacki 9, x2 2/Abacki 9}

Użyte klauzule:

```
K5: GRA_NA(Abacki_9, Gitara_12)
K8: ~UMIE_TEORIE(Abacki_9)
K21: ~UMIE_TEORIE(x5_17) => ~UMIE_NUTY(x5_17)
K2: ~UMIE_NUTY(x2_2) ^ GRA_NA(x2_2, x3_3) => AMATOR(x2_2)
K3: AMATOR(Abacki_9) => MUZYK(Babacki_10)
```

Graf wnioskowania:

$$\begin{array}{c} (\text{MUZYK}(\text{Babacki}_{10})) \\ \wedge \\ | \\ (\text{AMATOR}(\text{Abacki}_{9})) \\ \wedge \\ | \\ (\sim \text{UMIE_NUTY}(\text{Abacki}_{9}) \wedge \text{GRA_NA}(\text{Abacki}_{9}, \text{Gitara}_{12})) \\ \wedge \\ | \\ (\sim \text{UMIE_TEORIE}(\text{Abacki}_{9})) \\ \wedge \\ | \\ (\text{GRA_NA}(\text{Abacki}_{9}, \text{Gitara}_{12})) \end{array}$$

Przykład 4

Pliki przekazywane programowi:

argument.txt

```
LUBI(Wojtek,Python)
```

knowledge_base.txt

```
(LUBI(Michal,x9)^LUBI(Jakub,x9))=>LUBI(Wojtek,x9)
LUBI(x10,Java)=>LUBI(x10,Python)
LUBI(Michal,Java)
LUBI(Jakub,Python)
~LUBI(x11,Python)=>~LUBI(x11,Java)
```

variables.txt

```
x7
x8
x9
x10
x11
```

constants.txt

```
LUBI
~LUBI
Java
Python
Jakub
Michal
Wojtek
```

Wynik działania programu:

Baza wiedzy:

```
K1: LUBI(Michal_5, x9_2) ^ LUBI(Jakub_4, x9_2) => LUBI(Wojtek_6, x9_2)
K2: LUBI(x10_3, Java_2) => LUBI(x10_3, Python_3)
K3: LUBI(Michal_5, Java_2)
K4: LUBI(Jakub_4, Python_3)
K5: ~LUBI(x11_4, Python_3) => ~LUBI(x11_4, Java_2)
K6: LUBI(Jakub_4, x9_6) ^ ~LUBI(Wojtek_6, x9_6) => ~LUBI(Michal_5, x9_6)
K7: LUBI(Michal_5, x9_7) ^ ~LUBI(Wojtek_6, x9_7) => ~LUBI(Jakub_4, x9_7)
```

Teza: LUBI(Wojtek_6, Python_3)

Zbiór podstawień: {x9_2/Python_3, x10_3/Michal_5}

Użyte klauzule:

```
K4: LUBI(Jakub_4, Python_3)
K3: LUBI(Michal_5, Java_2)
K2: LUBI(x10_3, Java_2) => LUBI(x10_3, Python_3)
K1: LUBI(Michal_5, x9_2) ^ LUBI(Jakub_4, x9_2) => LUBI(Wojtek_6, x9_2)
```

Graf wnioskowania:

```
(LUBI(Wojtek_6, Python_3))
  ^
  |
(LUBI(Michal_5, Python_3) ^ LUBI(Jakub_4, Python_3))
  ^
  |
(LUBI(Michal_5, Java_2))
  ^
  |
(LUBI(Jakub_4, Python_3))
```

Przykład 5

Pliki przekazywane programowi:

argument.txt

CIEZAROWIEC (Syn)

knowledge_base.txt

```
OSOBA(x0,Wzrost,Wysoki)^OSOBA(x0,Wiek,Mlody)=>KOSZYKARZ(x0)
OSOBA(x1,Wzrost,Niski)^OSOBA(x1,Wiek,Mlody)=>CIEZAROWIEC(x1)
(RODZIC(x2,x3)^OSOBA(x2,Wzrost,Wysoki))=>OSOBA(x3,Wzrost,Wysoki)
RODZIC(x4,x5)=>~RODZIC(x5,x4)
(RODZIC(x6,x7)^OSOBA(x6,Wzrost,Niski))=>OSOBA(x7,Wzrost,Niski)
RODZIC(x8,x9)=>OSOBA(x8,Wiek,Stary)
RODZIC(x10,x11)=>OSOBA(x11,Wiek,Mlody)
RODZIC(Tata,Syn)
OSOBA(Tata,Wzrost,Niski)
```

variables.txt

```
x0
x1
x2
x3
x4
x5
x6
x7
x8
x9
x10
x11
```

constants.txt

```
OSOBA
KOSZYKARZ
CIEZAROWIEC
RODZIC
~RODZIC
Wzrost
Wysoki
Niski
Wiek
Mlody
Stary
Tata
Syn
```

Wynik działania programu:

Baza wiedzy:

```
K1: OSOBA(x0_0, Wzrost_5, Wysoki_6) ^ OSOBA(x0_0, Wiek_8, Mlody_9) => KOSZYKARZ(x0_0)
K2: OSOBA(x1_1, Wzrost_5, Niski_7) ^ OSOBA(x1_1, Wiek_8, Mlody_9) => CIEZAROWIEC(x1_1)
K3: RODZIC(x2_2, x3_3) ^ OSOBA(x2_2, Wzrost_5, Wysoki_6) => OSOBA(x3_3, Wzrost_5,
Wysoki_6)
K4: RODZIC(x4_4, x5_5) => ~RODZIC(x5_5, x4_4)
K5: RODZIC(x6_6, x7_7) ^ OSOBA(x6_6, Wzrost_5, Niski_7) => OSOBA(x7_7, Wzrost_5,
Niski_7)
K6: RODZIC(x8_8, x9_9) => OSOBA(x8_8, Wiek_8, Stary_10)
K7: RODZIC(x10_10, x11_11) => OSOBA(x11_11, Wiek_8, Mlody_9)
K8: RODZIC(Tata_11, Syn_12)
K9: OSOBA(Tata_11, Wzrost_5, Niski_7)
K10: OSOBA(x0_13, Wiek_8, Mlody_9) ^ ~KOSZYKARZ(x0_13) => ~OSOBA(x0_13, Wzrost_5,
Wysoki_6)
K11: OSOBA(x0_14, Wzrost_5, Wysoki_6) ^ ~KOSZYKARZ(x0_14) => ~OSOBA(x0_14, Wiek_8,
Mlody_9)
K12: OSOBA(x1_15, Wiek_8, Mlody_9) ^ ~CIEZAROWIEC(x1_15) => ~OSOBA(x1_15, Wzrost_5,
Niski_7)
K13: OSOBA(x1_16, Wzrost_5, Niski_7) ^ ~CIEZAROWIEC(x1_16) => ~OSOBA(x1_16, Wiek_8,
Mlody_9)
K14: OSOBA(x2_17, Wzrost_5, Wysoki_6) ^ ~OSOBA(x3_17, Wzrost_5, Wysoki_6) =>
~RODZIC(x2_17, x3_17)
K15: RODZIC(x2_18, x3_18) ^ ~OSOBA(x3_18, Wzrost_5, Wysoki_6) => ~OSOBA(x2_18,
Wzrost_5, Wysoki_6)
K17: OSOBA(x6_20, Wzrost_5, Niski_7) ^ ~OSOBA(x7_20, Wzrost_5, Niski_7) =>
~RODZIC(x6_20, x7_20)
K18: RODZIC(x6_21, x7_21) ^ ~OSOBA(x7_21, Wzrost_5, Niski_7) => ~OSOBA(x6_21, Wzrost_5,
Niski_7)
K19: ~OSOBA(x8_22, Wiek_8, Stary_10) => ~RODZIC(x8_22, x9_22)
K20: ~OSOBA(x11_23, Wiek_8, Mlody_9) => ~RODZIC(x10_23, x11_23)
```

Teza: CIEZAROWIEC(Syn_12)

Zbiór podstawień: {x11_11/Syn_12, x10_10/Tata_11, x7_7/Syn_12, x6_6/Tata_11, x1_1/Syn_12}

Użyte klauzule:

```
K8: RODZIC(Tata_11, Syn_12)
K7: RODZIC(x10_10, x11_11) => OSOBA(x11_11, Wiek_8, Mlody_9)
K9: OSOBA(Tata_11, Wzrost_5, Niski_7)
K8: RODZIC(Tata_11, Syn_12)
K5: RODZIC(x6_6, x7_7) ^ OSOBA(x6_6, Wzrost_5, Niski_7) => OSOBA(x7_7, Wzrost_5,
Niski_7)
K2: OSOBA(x1_1, Wzrost_5, Niski_7) ^ OSOBA(x1_1, Wiek_8, Mlody_9) => CIEZAROWIEC(x1_1)
```

Graf wnioskowania:

```
(CIEZAROWIEC(Syn_12))
^
|
(OSOBA(Syn_12, Wzrost_5, Niski_7) ^ OSOBA(Syn_12, Wiek_8, Mlody_9))
^
(RODZIC(Tata_11, Syn_12) ^ OSOBA(Tata_11, Wzrost_5, Niski_7))
^
(RODZIC(Tata_11, Syn_12))
^
(OSOBA(Tata_11, Wzrost_5, Niski_7))
^
(RODZIC(Tata_11, Syn_12))
^
(LUBI(Jakub_4, Python_3))
```

Przykład 8- Historia kryminalna (trudna do rozwiązania przez człowieka)

Pliki przekazywane programowi:

argument.txt

OSKARZONY (Abacki)

knowledge_base.txt

```
(~BYL_WIDZIANY(x1) ^ ~MA_MOTYW(x1) ^ MA_ALIBI(x1)) => ~PODEJRZANY(x1)
(RODZINA(x2,x3) ^ OFIARA(x3)) => POWIAZANY(x2)
RODZINA(x4,x5) => RODZINA(x5,x4)
(BYL_WIDZIANY(x6) ^ ~MA_ALIBI(x6)) => PODEJRZANY(x6)
(MA_MOTYW(x7) ^ ~MA_ALIBI(x7)) => PODEJRZANY(x7)
(PODEJRZANY(x8) ^ RODZINA(x8,x9) ^ ~NIENAWIDZI(x8,x9) ^ OFIARA(x9)) => OCZYSZCZONY(x8)
(OCZYSZCZONY(x10) ^ OFIARA(x11)) => ~OSKARZONY(x10,x11)
(~POWIAZANY(x12) ^ OFIARA(x13)) => ~OSKARZONY(x12,x13)
PODEJRZANY(x14) => POWIAZANY(x14)
(KOCHA(x15,x17) ^ KOCHA(x16,x17) ^ OFIARA(x16)) => MA_MOTYW(x15)
(KOCHA(x19,x20) ^ KOCHA(x18,x20)) => NIENAWIDZI(x19,x18)
NIENAWIDZI(x21,x22) => NIENAWIDZI(x22,x21)
(PODEJRZANY(x23) ^ NIENAWIDZI(x23,x24) ^ ~OCZYSZCZONY(x23) ^ OFIARA(x24)) => OSKARZONY(x23)
~PODEJRZANY(x25) => ~OCZYSZCZONY(x25)
(~RODZINA(x26,x27) ^ OFIARA(x27)) => ~OCZYSZCZONY(x26)
(NIENAWIDZI(x28,x29) ^ OFIARA(x29)) => ~OCZYSZCZONY(x28)
~RODZINA(Abacki,Babacki)
KOCHA(Babacki,Cabacka)
KOCHA(Abacki,Cabacka)
~MA_ALIBI(Abacki)
OFIARA(Babacki)
```

variables.txt

```
x1
x2
x3
x4
x5
x6
x7
x8
x9
x10
x11
x12
x13
x14
x15
x16
x17
x18
x19
x20
x21
x22
x23
x24
x25
x26
x27
x28
x29
```


constants.txt

```
BYL_WIDZIANY
~BYL_WIDZIANY
MA_MOTYW
~MA_MOTYW
MA_ALIBI
~MA_ALIBI
PODEJRZANY
~PODEJRZANY
RODZINA
~RODZINA
OFIARA
~OFIARA
POWIAZANY
~POWIAZANY
NIENAWIDZI
~NIENAWIDZI
OCZYSZCZONY
~OCZYSZCZONY
KOCHA
OSKARZONY
~OSKARZONY
Abacki
Babacki
Cabacka
```

Wynik działania programu:

Baza wiedzy:

```
K1: ~BYL_WIDZIANY(x1_0) ^ ~MA_MOTYW(x1_0) ^ MA_ALIBI(x1_0) => ~PODEJRZANY(x1_0)
K2: RODZINA(x2_1, x3_2) ^ OFIARA(x3_2) => POWIAZANY(x2_1)
K3: RODZINA(x4_3, x5_4) => RODZINA(x5_4, x4_3)
K4: BYL_WIDZIANY(x6_5) ^ ~MA_ALIBI(x6_5) => PODEJRZANY(x6_5)
K5: MA_MOTYW(x7_6) ^ ~MA_ALIBI(x7_6) => PODEJRZANY(x7_6)
K6: PODEJRZANY(x8_7) ^ RODZINA(x8_7, x9_8) ^ ~NIENAWIDZI(x8_7, x9_8) ^ OFIARA(x9_8) =>
OCZYSZCZONY(x8_7)
K7: OCZYSZCZONY(x10_9) ^ OFIARA(x11_10) => ~OSKARZONY(x10_9, x11_10)
K8: ~POWIAZANY(x12_11) ^ OFIARA(x13_12) => ~OSKARZONY(x12_11, x13_12)
K9: PODEJRZANY(x14_13) => POWIAZANY(x14_13)
K10: KOCHA(x15_14, x17_16) ^ KOCHA(x16_15, x17_16) ^ OFIARA(x16_15) => MA_MOTYW(x15_14)
K11: KOCHA(x19_18, x20_19) ^ KOCHA(x18_17, x20_19) => NIENAWIDZI(x19_18, x18_17)
K12: NIENAWIDZI(x21_20, x22_21) => NIENAWIDZI(x22_21, x21_20)
K13: PODEJRZANY(x23_22) ^ NIENAWIDZI(x23_22, x24_23) ^ ~OCZYSZCZONY(x23_22) ^
OFIARA(x24_23) => OSKARZONY(x23_22)
K14: ~PODEJRZANY(x25_24) => ~OCZYSZCZONY(x25_24)
K15: ~RODZINA(x26_25, x27_26) ^ OFIARA(x27_26) => ~OCZYSZCZONY(x26_25)
K16: NIENAWIDZI(x28_27, x29_28) ^ OFIARA(x29_28) => ~OCZYSZCZONY(x28_27)
K17: ~RODZINA(Abacki_21, Babacki_22)
K18: KOCHA(Babacki_22, Cabacka_23)
K19: KOCHA(Abacki_21, Cabacka_23)
K20: ~MA_ALIBI(Abacki_21)
K21: OFIARA(Babacki_22)
K22: ~MA_MOTYW(x1_30) ^ MA_ALIBI(x1_30) ^ PODEJRZANY(x1_30) => BYL_WIDZIANY(x1_30)
K23: ~BYL_WIDZIANY(x1_31) ^ MA_ALIBI(x1_31) ^ PODEJRZANY(x1_31) => MA_MOTYW(x1_31)
K24: ~BYL_WIDZIANY(x1_32) ^ ~MA_MOTYW(x1_32) ^ PODEJRZANY(x1_32) => ~MA_ALIBI(x1_32)
K25: OFIARA(x3_33) ^ ~POWIAZANY(x2_33) => ~RODZINA(x2_33, x3_33)
K26: RODZINA(x2_34, x3_34) ^ ~POWIAZANY(x2_34) => ~OFIARA(x3_34)
K27: ~RODZINA(x5_35, x4_35) => ~RODZINA(x4_35, x5_35)
K28: ~MA_ALIBI(x6_36) ^ ~PODEJRZANY(x6_36) => ~BYL_WIDZIANY(x6_36)
K29: BYL_WIDZIANY(x6_37) ^ ~PODEJRZANY(x6_37) => MA_ALIBI(x6_37)
K31: MA_MOTYW(x7_39) ^ ~PODEJRZANY(x7_39) => MA_ALIBI(x7_39)
```

K32: $\text{RODZINA}(x8_40, x9_40) \wedge \sim \text{NIENAWIDZI}(x8_40, x9_40) \wedge \text{OFIARA}(x9_40) \wedge \sim \text{OCZYSZCZONY}(x8_40) \Rightarrow \sim \text{PODEJRZANY}(x8_40)$
 K33: $\text{PODEJRZANY}(x8_41) \wedge \sim \text{NIENAWIDZI}(x8_41, x9_41) \wedge \text{OFIARA}(x9_41) \wedge \sim \text{OCZYSZCZONY}(x8_41) \Rightarrow \sim \text{RODZINA}(x8_41, x9_41)$
 K34: $\text{PODEJRZANY}(x8_42) \wedge \text{RODZINA}(x8_42, x9_42) \wedge \text{OFIARA}(x9_42) \wedge \sim \text{OCZYSZCZONY}(x8_42) \Rightarrow \text{NIENAWIDZI}(x8_42, x9_42)$
 K35: $\text{PODEJRZANY}(x8_43) \wedge \text{RODZINA}(x8_43, x9_43) \wedge \sim \text{NIENAWIDZI}(x8_43, x9_43) \wedge \sim \text{OCZYSZCZONY}(x8_43) \Rightarrow \sim \text{OFIARA}(x9_43)$
 K36: $\text{OFIARA}(x11_44) \wedge \text{OSKARZONY}(x10_44, x11_44) \Rightarrow \sim \text{OCZYSZCZONY}(x10_44)$
 K37: $\text{OCZYSZCZONY}(x10_45) \wedge \text{OSKARZONY}(x10_45, x11_45) \Rightarrow \sim \text{OFIARA}(x11_45)$
 K39: $\sim \text{POWIAZANY}(x12_47) \wedge \text{OSKARZONY}(x12_47, x13_47) \Rightarrow \sim \text{OFIARA}(x13_47)$
 K40: $\sim \text{POWIAZANY}(x14_48) \Rightarrow \sim \text{PODEJRZANY}(x14_48)$
 K41: $\text{KOCHA}(x16_49, x17_49) \wedge \text{OFIARA}(x16_49) \wedge \sim \text{MA_MOTYW}(x15_49) \Rightarrow \sim \text{KOCHA}(x15_49, x17_49)$
 K43: $\text{KOCHA}(x15_51, x17_51) \wedge \text{KOCHA}(x16_51, x17_51) \wedge \sim \text{MA_MOTYW}(x15_51) \Rightarrow \sim \text{OFIARA}(x16_51)$
 K44: $\text{KOCHA}(x18_52, x20_52) \wedge \sim \text{NIENAWIDZI}(x19_52, x18_52) \Rightarrow \sim \text{KOCHA}(x19_52, x20_52)$
 K46: $\sim \text{NIENAWIDZI}(x22_54, x21_54) \Rightarrow \sim \text{NIENAWIDZI}(x21_54, x22_54)$
 K47: $\text{NIENAWIDZI}(x23_55, x24_55) \wedge \sim \text{OCZYSZCZONY}(x23_55) \wedge \text{OFIARA}(x24_55) \wedge \sim \text{OSKARZONY}(x23_55) \Rightarrow \sim \text{PODEJRZANY}(x23_55)$
 K48: $\text{PODEJRZANY}(x23_56) \wedge \sim \text{OCZYSZCZONY}(x23_56) \wedge \text{OFIARA}(x24_56) \wedge \sim \text{OSKARZONY}(x23_56) \Rightarrow \sim \text{NIENAWIDZI}(x23_56, x24_56)$
 K49: $\text{PODEJRZANY}(x23_57) \wedge \text{NIENAWIDZI}(x23_57, x24_57) \wedge \text{OFIARA}(x24_57) \wedge \sim \text{OSKARZONY}(x23_57) \Rightarrow \text{OCZYSZCZONY}(x23_57)$
 K50: $\text{PODEJRZANY}(x23_58) \wedge \text{NIENAWIDZI}(x23_58, x24_58) \wedge \sim \text{OCZYSZCZONY}(x23_58) \wedge \sim \text{OSKARZONY}(x23_58) \Rightarrow \sim \text{OFIARA}(x24_58)$
 K51: $\text{OCZYSZCZONY}(x25_59) \Rightarrow \text{PODEJRZANY}(x25_59)$
 K52: $\text{OFIARA}(x27_60) \wedge \text{OCZYSZCZONY}(x26_60) \Rightarrow \text{RODZINA}(x26_60, x27_60)$
 K53: $\sim \text{RODZINA}(x26_61, x27_61) \wedge \text{OCZYSZCZONY}(x26_61) \Rightarrow \sim \text{OFIARA}(x27_61)$
 K55: $\text{NIENAWIDZI}(x28_63, x29_63) \wedge \text{OCZYSZCZONY}(x28_63) \Rightarrow \sim \text{OFIARA}(x29_63)$

Teza: $\text{OSKARZONY}(\text{Abacki_21})$

Zbiór podstawień: $\{x27_26/\text{Babacki_22}, x23_22/\text{Abacki_21}, x15_14/\text{Abacki_21}, x26_25/\text{Abacki_21}, x7_6/\text{Abacki_21}, x18_17/\text{Babacki_22}, x20_19/\text{Cabacka_23}, x17_16/\text{Cabacka_23}, x19_18/\text{Abacki_21}, x24_23/x18_17, x16_15/\text{Babacki_22}\}$

Użyte klauzule:

K21: $\text{OFIARA}(\text{Babacki_22})$
 K21: $\text{OFIARA}(\text{Babacki_22})$
 K17: $\sim \text{RODZINA}(\text{Abacki_21}, \text{Babacki_22})$
 K15: $\sim \text{RODZINA}(x26_25, x27_26) \wedge \text{OFIARA}(x27_26) \Rightarrow \sim \text{OCZYSZCZONY}(x26_25)$
 K18: $\text{KOCHA}(\text{Babacki_22}, \text{Cabacka_23})$
 K19: $\text{KOCHA}(\text{Abacki_21}, \text{Cabacka_23})$
 K11: $\text{KOCHA}(x19_18, x20_19) \wedge \text{KOCHA}(x18_17, x20_19) \Rightarrow \text{NIENAWIDZI}(x19_18, x18_17)$
 K20: $\sim \text{MA_ALIBI}(\text{Abacki_21})$
 K21: $\text{OFIARA}(\text{Babacki_22})$
 K18: $\text{KOCHA}(\text{Babacki_22}, \text{Cabacka_23})$
 K19: $\text{KOCHA}(\text{Abacki_21}, \text{Cabacka_23})$
 K10: $\text{KOCHA}(x15_14, x17_16) \wedge \text{KOCHA}(x16_15, x17_16) \wedge \text{OFIARA}(x16_15) \Rightarrow \text{MA_MOTYW}(x15_14)$
 K5: $\text{MA_MOTYW}(x7_6) \wedge \sim \text{MA_ALIBI}(x7_6) \Rightarrow \text{PODEJRZANY}(x7_6)$
 K13: $\text{PODEJRZANY}(x23_22) \wedge \text{NIENAWIDZI}(x23_22, x24_23) \wedge \sim \text{OCZYSZCZONY}(x23_22) \wedge \text{OFIARA}(x24_23) \Rightarrow \text{OSKARZONY}(x23_22)$

Graf wnioskowania:

$(\text{OSKARZONY}(\text{Abacki_21}))$
 \wedge
 $|$
 $(\text{PODEJRZANY}(\text{Abacki_21}) \wedge \text{NIENAWIDZI}(\text{Abacki_21}, \text{Babacki_22}) \wedge \sim \text{OCZYSZCZONY}(\text{Abacki_21}) \wedge \text{OFIARA}(\text{Babacki_22}))$
 \wedge
 $|$
 $(\text{MA_MOTYW}(\text{Abacki_21}) \wedge \sim \text{MA_ALIBI}(\text{Abacki_21}))$

$$\begin{aligned}
& \wedge \\
& | \\
& (\text{KOCHA}(\text{Abacki_21}, \text{Cabacka_23}) \wedge \text{KOCHA}(\text{Babacki_22}, \text{Cabacka_23}) \wedge \text{OFIARA}(\text{Babacki_22})) \\
& \wedge \\
& | \\
& (\text{KOCHA}(\text{Abacki_21}, \text{Cabacka_23})) \\
& \wedge \\
& | \\
& (\text{KOCHA}(\text{Babacki_22}, \text{Cabacka_23})) \\
& \wedge \\
& | \\
& (\text{OFIARA}(\text{Babacki_22})) \\
& \wedge \\
& | \\
& (\sim \text{MA_ALIBI}(\text{Abacki_21})) \\
& \wedge \\
& | \\
& (\text{KOCHA}(\text{Abacki_21}, \text{Cabacka_23}) \wedge \text{KOCHA}(\text{Babacki_22}, \text{Cabacka_23})) \\
& \wedge \\
& | \\
& (\text{KOCHA}(\text{Abacki_21}, \text{Cabacka_23})) \\
& \wedge \\
& | \\
& (\text{KOCHA}(\text{Babacki_22}, \text{Cabacka_23})) \\
& \wedge \\
& | \\
& (\sim \text{RODZINA}(\text{Abacki_21}, \text{Babacki_22}) \wedge \text{OFIARA}(\text{Babacki_22})) \\
& \wedge \\
& | \\
& (\sim \text{RODZINA}(\text{Abacki_21}, \text{Babacki_22})) \\
& \wedge \\
& | \\
& (\text{OFIARA}(\text{Babacki_22})) \\
& \wedge \\
& | \\
& (\text{OFIARA}(\text{Babacki_22}))
\end{aligned}$$