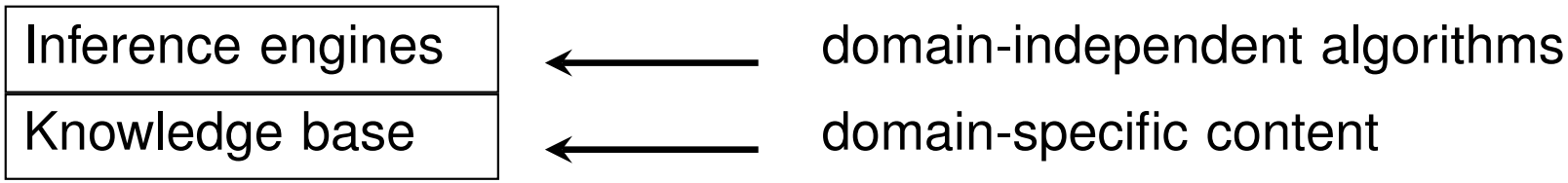
SID – Wykład 5
Wnioskowanie w rachunku zdań

Dominik Ślęzak

Wydział Matematyki, Informatyki i Mechaniki UW
slezak@mimuw.edu.pl

Bazy wiedzy



Baza wiedzy = zbiór faktów o świecie, zdania zapisane w języku formalnym

Deklaratywne podejście do budowania systemu:

powiedz systemowi to co potrzebuje wiedzieć.

System może zapytać się co robić – odpowiedzi powinny wynikać z bazy wiedzy

Poziom wiedzy:

to co jest wiadome, niezależnie od tego jak jest zaimplementowane

Poziom implementacji:

struktury danych w bazie wiedzy i algorytmy manipulowania nimi

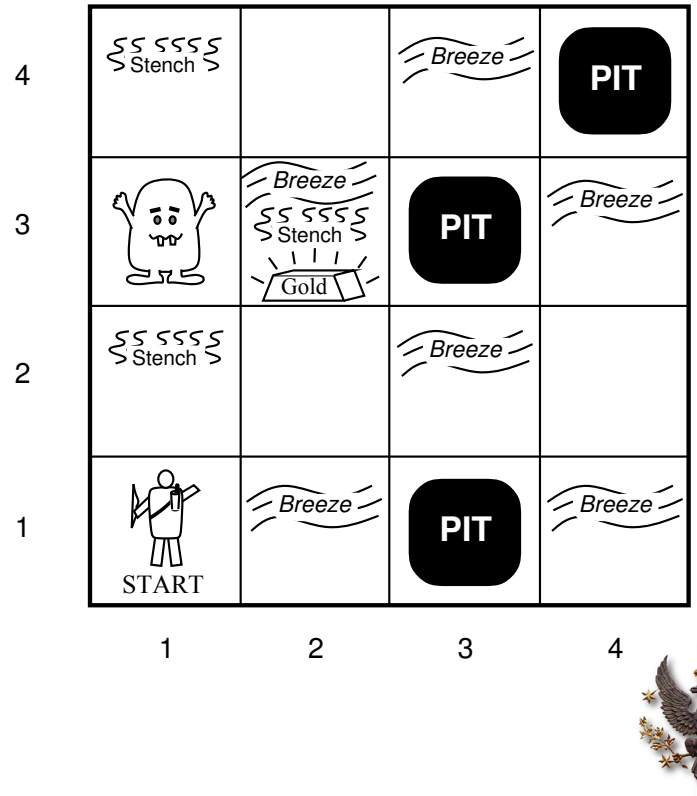
Świat Wumpusa: opis

Wartości wypłaty:

- złoto +1000
- śmierć -1000
- -1 za krok
- -10 za użycie strzały

Reguły:

- Pola sąsiadujące z Wumpusem mają zapach
- Pola wokół pułapek są wietrzne
- Złoto się błyszczy
- Strzał w kierunku Wumpusa zabija go
- Strzał wykorzystuje jedyną strzałę
- Podniesienie powoduje zabranie złota, jeśli jest na tym samym polu
- Upuszczenie powoduje pozostawienie złota



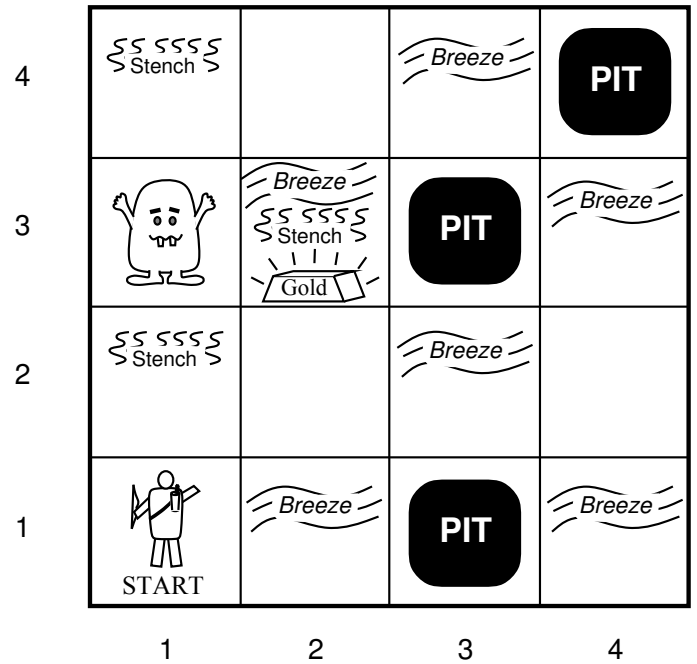
Świat Wumpusa: opis

Obserwacje:

- wiatr
- błysk
- zapach

Działania:

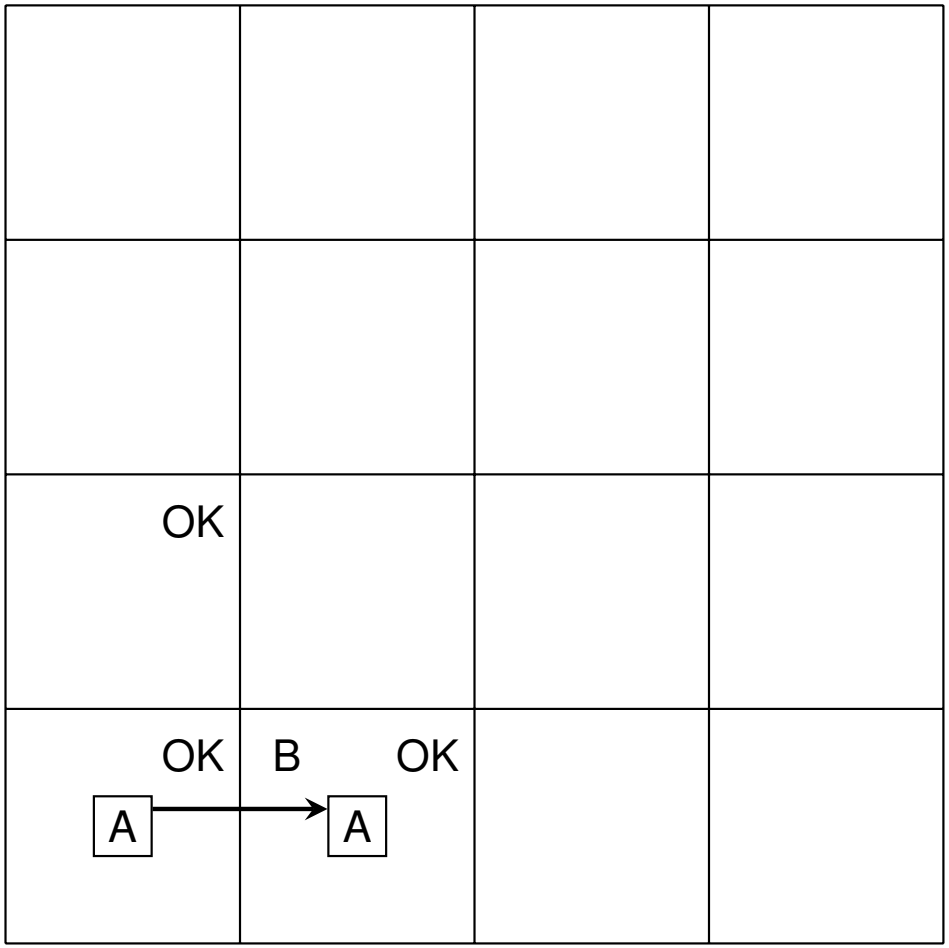
- skręć w lewo
- skręć w prawo
- naprzód
- podnieś
- upuść
- strzał



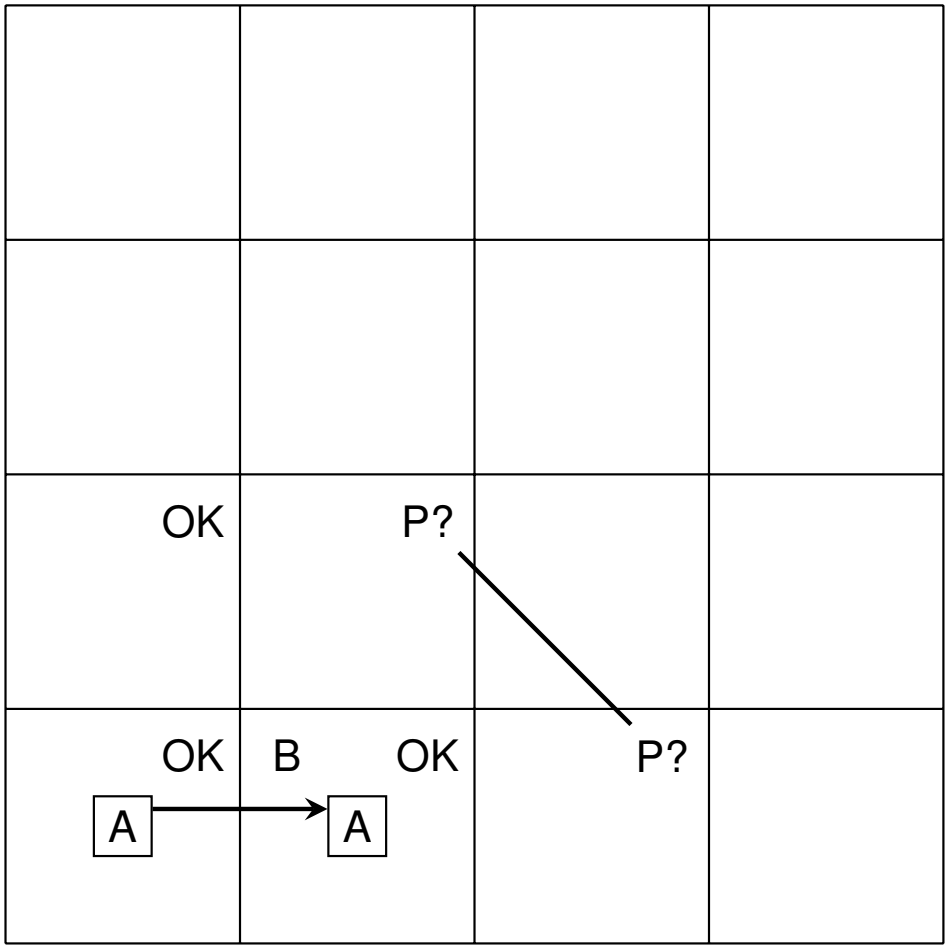
Świat Wumpusa: eksploracja

OK			
OK A	OK		

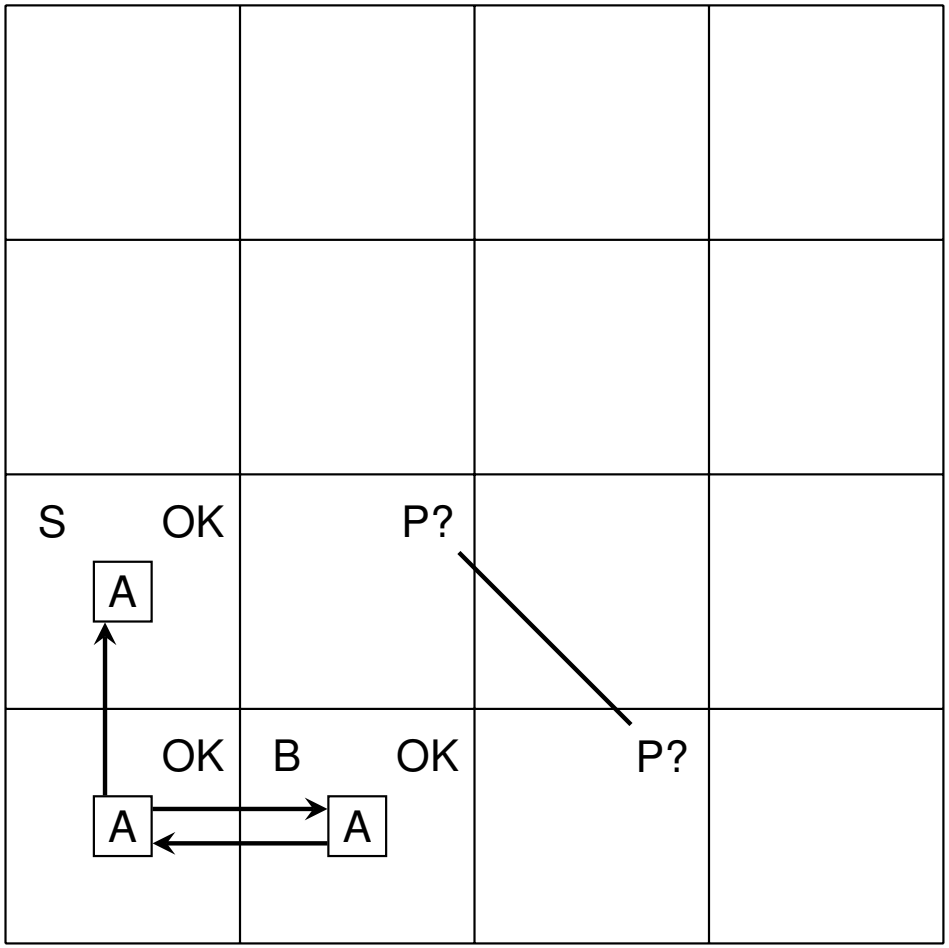
Świat Wumpusa: eksploracja



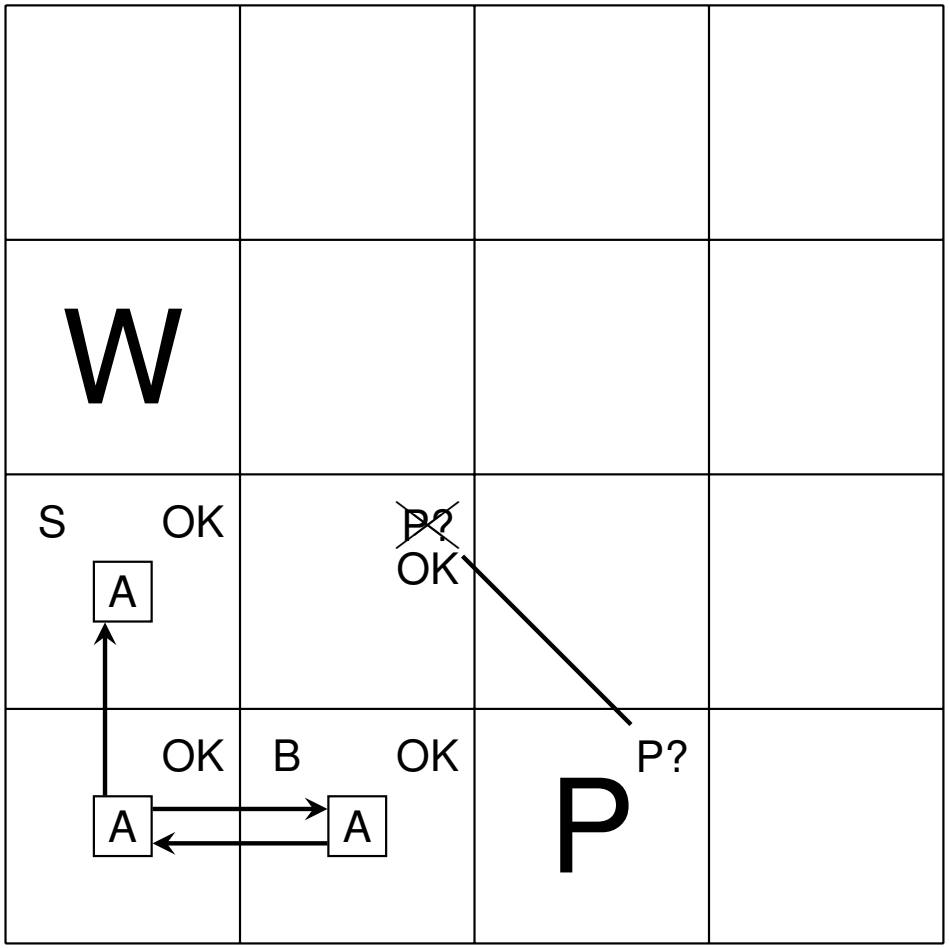
Świat Wumpusa: eksploracja



Świat Wumpusa: eksploracja



Świat Wumpusa: eksploracja



Logika

Logika jest formalnym językiem reprezentacji informacji
takim, w którym mogą być wyciągane wnioski

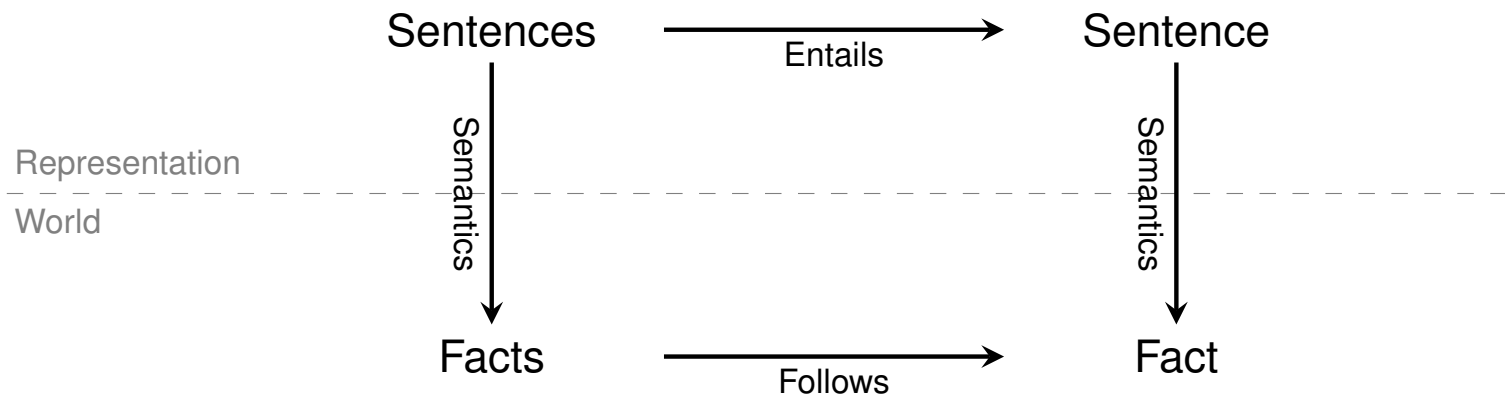
Syntaktyka – opisuje budowę zdań

Semantyka – opisuje związek pomiędzy zdaniami i odpowiadającymi im faktami zachodzącymi w świecie

Np. język arytmetyki
$x + 2 \geq y$ jest zdaniem; $x2 + y >$ nie jest zdaniem
$x + 2 \geq y$ jest prawdziwe $\Leftrightarrow$ $x + 2$ jest nie mniejsze niż liczba y
$x + 2 \geq y$ jest nieprawdziwe w świecie, gdzie $x = 0,\ y = 6$

Wnioskowanie: proces wyprowadzania nowych zdań ze zdań przyjętych jako prawdziwe (tzn. reprezentujących prawdziwe fakty). Poprawne wnioskowanie powinno zapewniać prawdziwość wyprowadzonych zdań.

Wnioskowanie



Model dla bazy wiedzy *KB*: każdy świat, w którym prawdziwe są wszystkie zdania z *KB*.

Wynikanie (entailment): Zdanie α wynika z bazy wiedzy *KB*, $KB \models \alpha$, jeśli α jest prawdziwe w każdym modelu dla *KB*.

Wyprowadzalność (derivability): zdanie α jest wyprowadzalne z bazy wiedzy *KB* przy użyciu procedury dowodzenia *i*, $KB \vdash_i \alpha \Leftrightarrow$ *i* znajduje dowód zdania α ze zdań zbioru *KB*.

Wnioskowanie

Konsekwencje KB to stóg siana, a α to igła.
Logiczna konsekwencja = igła w stogu siana
Wnioskowanie = metoda na jej znalezienie

Poprawność: procedura dowodzenia i jest poprawna $\Leftrightarrow$ dla każdej bazy wiedzy KB i każdego zdania α, $KB \vdash_i \alpha$ pociąga $KB \models \alpha$.

Pełność: procedura dowodzenia i jest pełna $\Leftrightarrow$ dla każdej bazy wiedzy KB i każdego zdania α, $KB \models \alpha$ pociąga $KB \vdash_i \alpha$.

Cel: zdefiniować logikę, w której można wyrazić możliwie jak najwięcej i dla której istnieje poprawna i pełna procedura dowodzenia.

Tzn. ta procedura odpowie na każde pytanie, które wynika z tego, co wiadomo w bazie wiedzy KB.

Logiczna konsekwencja

Logiczna konsekwencja oznacza, że jeden fakt wynika z innego:

$$KB \models \alpha$$

α jest logiczną konsekwencją bazy wiedzy *KB*

$\Leftrightarrow$

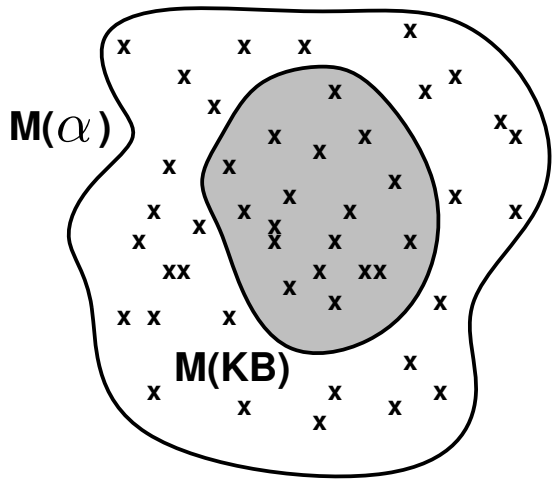
α jest prawdziwe we wszystkich światach, w których *KB* jest prawdziwe

Mówimy, że *m* jest modelem zdania α jeśli
α jest prawdziwe w *m*

$M(\alpha)$ jest zbiorem wszystkich modeli α

Wtedy $KB \models \alpha \Leftrightarrow M(KB) \subseteq M(\alpha)$

Np. *KB* = Giants i Reds wygrali
α = Giants wygrali

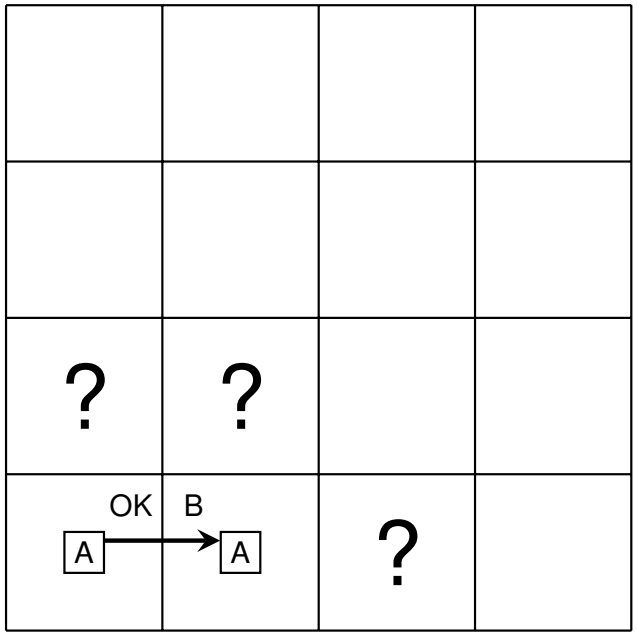


Świat Wumpusa: logiczna konsekwencja

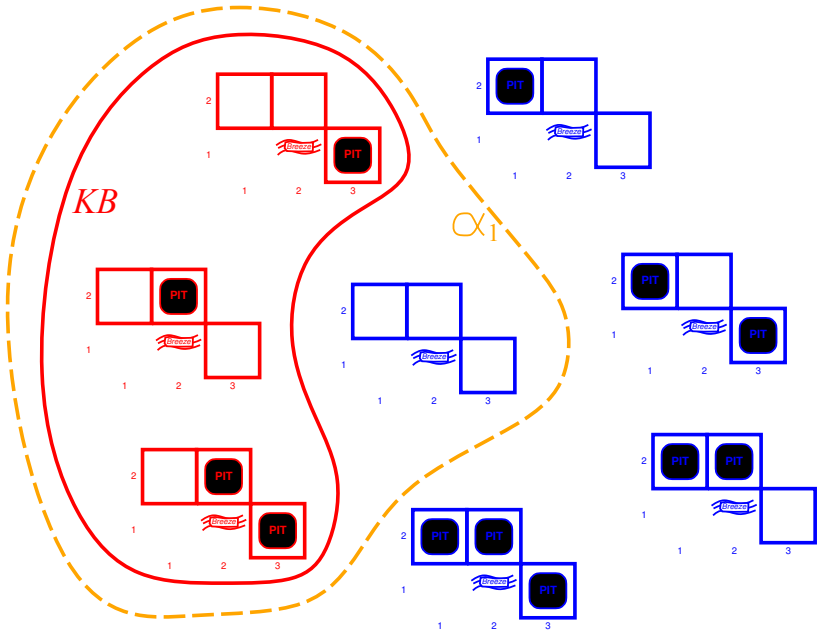
Sytuacja po zbadaniu pola [1,1], przesunięciu w prawo i wykryciu wiatru w [2,1]

Rozważamy możliwe modele dla pól '?' dotyczące informacji, czy na tych polach są pułapki

3 binarne wybory $\Longrightarrow$ 8 możliwych modeli



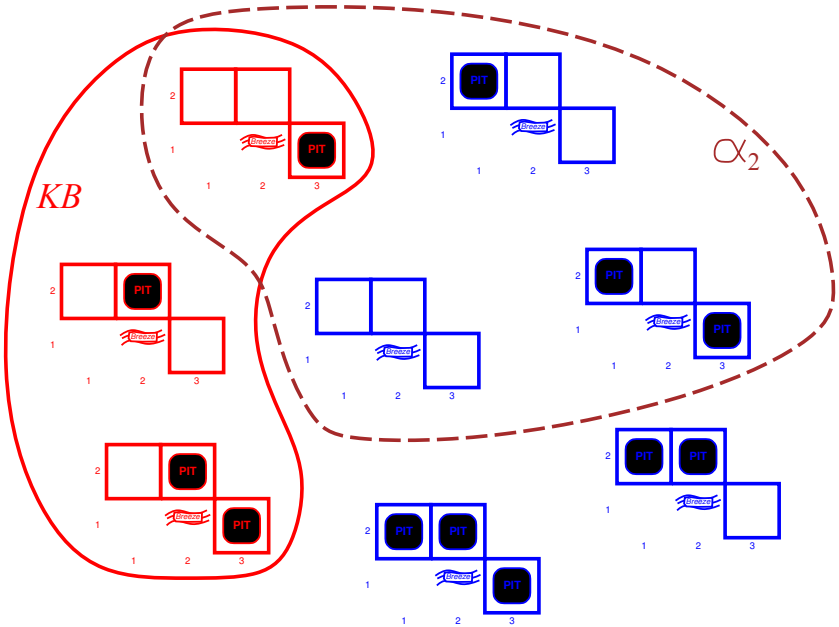
Modele w świecie Wumpusa



KB = reguły świata Wumpusa + obserwacje

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

Modele w świecie Wumpusa



KB = reguły świata Wumpusa + obserwacje

α_2 = “[2,2] jest bezpieczne”, $KB \not\models \alpha_2$

Logika zdaniowa: semantyka

Każdy model określa wartość prawda/fałsz dla każdego symbolu zdaniowego

Np.	$P_{1,2}$	$P_{2,2}$	$P_{3,1}$
	true	*true*	*false*

(Dla tych symboli 8 możliwych modeli, mogą być wyliczone automatycznie.)

Reguły do określenia prawdziwości zdań względem modelu m:

$$\neg S \text{ jest prawdziwe} \iff S \text{ jest nieprawdziwe}$$
$$S_1 \wedge S_2 \text{ jest prawdziwe} \iff S_1 \text{ jest prawdziwe i } S_2 \text{ jest prawdziwe}$$
$$S_1 \vee S_2 \text{ jest prawdziwe} \iff S_1 \text{ jest prawdziwe lub } S_2 \text{ jest prawdziwe}$$
$$S_1 \Rightarrow S_2 \text{ jest prawdziwe} \iff S_1 \text{ jest nieprawdziwe lub } S_2 \text{ jest prawdziwe}$$
$$\text{tzn. jest nieprawdziwe} \iff S_1 \text{ jest prawdziwe i } S_2 \text{ jest nieprawdziwe}$$
$$S_1 \Leftrightarrow S_2 \text{ jest prawdziwe} \iff S_1 \Rightarrow S_2 \text{ jest prawdziwe i } S_2 \Rightarrow S_1 \text{ jest prawdziwe}$$

Prosty rekurencyjny proces określający prawdziwość dowolnego zdania, np.

$$\neg P_{1,2} \wedge (P_{2,2} \vee P_{3,1}) = \mathit{true} \wedge (\mathit{false} \vee \mathit{true}) = \mathit{true} \wedge \mathit{true} = \mathit{true}$$

Zdania w świecie Wumpusa

Niech $P_{i,j}$ jest prawdziwe jeśli w $[i,j]$ jest pułapka.
Niech $B_{i,j}$ jest prawdziwe jeśli w $[i,j]$ jest wiatr.

$\neg P_{1,1}$

$\neg B_{1,1}$

$B_{2,1}$

Pułapki wywołują wiatr na sąsiednich polach

$$B_{1,1} \iff (P_{1,2} \vee P_{2,1})$$
$$B_{2,1} \iff (P_{1,1} \vee P_{2,2} \vee P_{3,1})$$

"Na polu jest wiatr wtedy i tylko wtedy gdy w sąsiedztwie jest pułapka"

CNF, DNF

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

$$(p_{11} \vee p_{12} \vee \ldots \vee p_{1k_1}) \wedge (p_{21} \vee p_{22} \vee \ldots \vee p_{2k_2}) \wedge \ldots \wedge (p_{n1} \vee p_{n2} \vee \ldots \vee p_{nk_n})$$

Problem znajdowania wartościowania spełniającego formułę w postaci CNF jest NP-zupełny.

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

$$(p_{11} \wedge p_{12} \wedge \ldots \wedge p_{1k_1}) \vee (p_{21} \wedge p_{22} \wedge \ldots \wedge p_{2k_2}) \vee \ldots \vee (p_{n1} \wedge p_{n2} \wedge \ldots \wedge p_{nk_n})$$

Problem znajdowania wartościowania spełniającego formułę w postaci DNF - istnieją algorytmy wielomianowe.

Implikanty

Każde odwzorowanie $f : \{0,1\}^n \to \{0,1\}$ nazywamy funkcją Boolowską.

Funkcje Boolowskie $\equiv$ formuły Boolowskie:

- Kanoniczna postać sumy (DNF), np. $f = xy\overline{z} + x\overline{y}z + \overline{x}yz + xyz$
- Kanoniczna postać iloczynu (CNF), np. $f = (x + y + z)(\overline{x} + y)$

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.

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

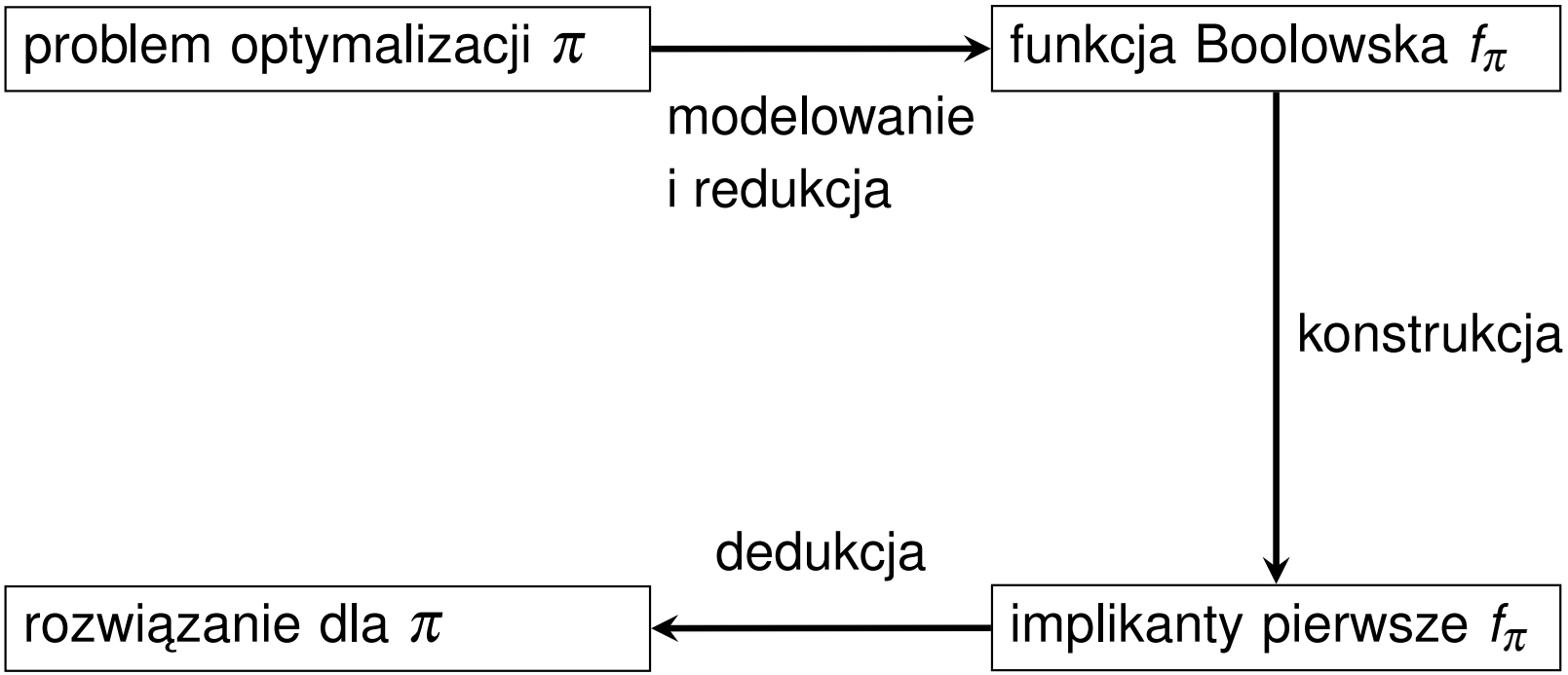
$$\forall_{a \in \{0,1\}^n} \quad t(a) = 1 \Rightarrow f(a) = 1$$

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:

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

Wnioskowanie Boolowskie



Tautologie i spełnialność

Zdanie jest tautologią jeśli jest prawdziwe we wszystkich modelach,
np. $True$, $A \vee \neg A$, $A \implies A$, $(A \wedge (A \implies B)) \implies B$

Tautologie są związane z Twierdzeniem o Dedukcji:
$KB \models \alpha$ wtedy i tylko wtedy gdy $(KB \implies \alpha)$ jest tautologią

Zdanie jest spełnialne jeśli jest prawdziwe w niektórych modelach,
np. $A \vee B$, C

Zdanie jest niespełnialne jeśli nie jest prawdziwe w żadnym modelu,
np. $A \wedge \neg A$

Spełnialnoć jest związana z wnioskowaniem przez sprowadzenie do sprzeczności:
$KB \models \alpha$ wtedy i tylko wtedy gdy $(KB \wedge \neg \alpha)$ jest niespełnialne

Metody dowodzenia

Metody dowodzenia można podzielić na dwie kategorie:
Sprawdzanie modeli

- Przeszukiwanie przestrzeni wartościowań
  - wyliczanie tabeli prawdziwości (zawsze wykładnicze od n)
  - poprawiony backtracking, alg. Davis–Putnam–Logemann–Loveland
  - przeszukiwanie heurystyczne w przestrzeni modeli (poprawne, ale niepełne), np. algorytmy hill-climbing podobne do min-conflicts

Zastosowanie reguł wnioskowania

- Poprawne generowanie nowych zdań ze starych
- Dowód = ciąg zastosowań reguł wnioskowania
  Można użyć reguł jako operatorów w standardowych algorytmach przeszukiwania
- Wymaga zazwyczaj przekształcenia zdań do postaci normalnej

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

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

Forward chaining i backward chaining

Postać Horna (ograniczona)

KB = koniunkcja klauzul Horna

Klauzula Horna =

- symbol zdaniowy; lub
- (koniunkcja symboli) $\Longrightarrow$ symbol

Np. $C \wedge (B \Longrightarrow A) \wedge (C \wedge D \Longrightarrow B)$

Modus Ponens (dla postaci Horna):

$$\frac{\alpha_1, \ldots, \alpha_n \qquad \alpha_1 \wedge \cdots \wedge \alpha_n \Longrightarrow \beta}{\beta}$$

Opisane dalej algorytmy forward i backward chaining wykonują się dla takiej postaci w czasie liniowym.

Forward chaining: algorytm

Pomysł: stosuje dowolną regułę, której przesłanki są spełnione w *KB*, dodaje jej wniosek do *KB* i powtarza aż znajdzie odpowiedź.

```
function PL-FC-ENTAILS(KB, q) returns true or false
    local variables: count - a table, indexed by clause, initially the number of premises
                     inferred - a table, indexed by symbol, each entry initially false
                     agenda - a list of symbols, initially the symbols known to be true
    while agenda is not empty do
        p ← POP(agenda)
        if notinferred[p] then
            inferred[p] ← true
            for each Horn clause c in whose premise p appears do
                decrement count[c]
                if count[c] = 0 then
                    if HEAD[c] = q then
                        return true
                    end if
                    PUSH(HEAD[c], agenda)
                end if
            end for
        end if
    end while
end function
```

Backward chaining

Pomysł: wyprowadzanie wstecz od zapytania q:

dowód q w backward chaining przez

- sprawdzenie, czy q jest już znane, lub
- udowodnienie wszystkich przesłanek pewnej reguły, która pociąga q

Unikanie pętli: sprawdza, czy nowy podcel nie był już wcześniej wygenerowany

Unikanie powtórzeń: sprawdza, czy dla nowego podcelu

1) była już udowodniona prawdziwość, lub

2) dowód był już podjęty wcześniej i zakończył się porażką

Forward chaining a backward chaining

Forward chaining jest sterowany danymi, por. automatyczne, nieświadome przetwarzanie, np. rozpoznawanie obiektów, rutynowe decyzje.

Może wykonać dużo pracy nieistotnej dla osiągnięcia celu.

Backward chaining jest nakierowany na cel, dobry do rozwiązywania problemów, np. Gdzie są moje klucze? Jak dostanę się na studia?

Koszt backward chaining może być dużo mniejszy niż liniowy względem rozmiaru bazy wiedzy *KB*.

Rezolucja

Problem dowodzenia twierdzeń jest równoważny problemowi spełnialności: Dla danej bazy wiedzy KB i danej formuły α próbujemy stwierdzić, czy zbiór $KB \cup \{\neg\alpha\}$ jest niespełnialny.

Ponieważ bazy wiedzy są skończone, $KB \cup \{\neg\alpha\}$ można zawsze traktować jako pojedynczą formułę.

Metoda rezolucji dla danej formuły β próbuje stwierdzić, czy β jest niespełnialna.

Rezolucja

Postać normalna koniunkcyjna (CNF — uniwersalna)

koniunkcja $\underbrace{\text{alternatyw literałów}}_{\text{klauzule}}$

Np. $(A \vee \neg B) \wedge (B \vee \neg C \vee \neg D)$

Rezolucyjna reguła wnioskowania (dla CNF):

$$\frac{\ell_1 \vee \cdots \vee \ell_k \qquad m_1 \vee \cdots \vee m_n}{\ell_1 \vee \cdots \vee \ell_{i-1} \vee \ell_{i+1} \vee \cdots \vee \ell_k \vee m_1 \vee \cdots \vee m_{j-1} \vee m_{j+1} \vee \cdots \vee m_n}$$

gdzie ℓ_i i m_j są dopełniającymi się literałami, np.

$$\frac{P_{1,3} \vee P_{2,2}, \qquad \neg P_{2,2}}{P_{1,3}}$$

Rezolucja: przekształcanie zdania do CNF

$B_{1,1} \Leftrightarrow (P_{1,2} \vee P_{2,1})$

1. Eliminacja $\Leftrightarrow$ poprzez zastąpienie $\alpha \Leftrightarrow \beta$ przez $(\alpha \Rightarrow \beta) \wedge (\beta \Rightarrow \alpha)$.

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

2. Eliminacja $\Rightarrow$ poprzez zastąpienie $\alpha \Rightarrow \beta$ przez $\neg\alpha \vee \beta$.

$$(\neg B_{1,1} \vee P_{1,2} \vee P_{2,1}) \wedge (\neg(P_{1,2} \vee P_{2,1}) \vee B_{1,1})$$

3. Przesunięcie $\neg$ do wewnątrz (prawa de Morgana i eliminacja podwójnej negacji):

$$(\neg B_{1,1} \vee P_{1,2} \vee P_{2,1}) \wedge ((\neg P_{1,2} \wedge \neg P_{2,1}) \vee B_{1,1})$$

4. Spłaszczenie przy pomocy rozdzielności ($\vee$ względem $\wedge$):

$$(\neg B_{1,1} \vee P_{1,2} \vee P_{2,1}) \wedge (\neg P_{1,2} \vee B_{1,1}) \wedge (\neg P_{2,1} \vee B_{1,1})$$

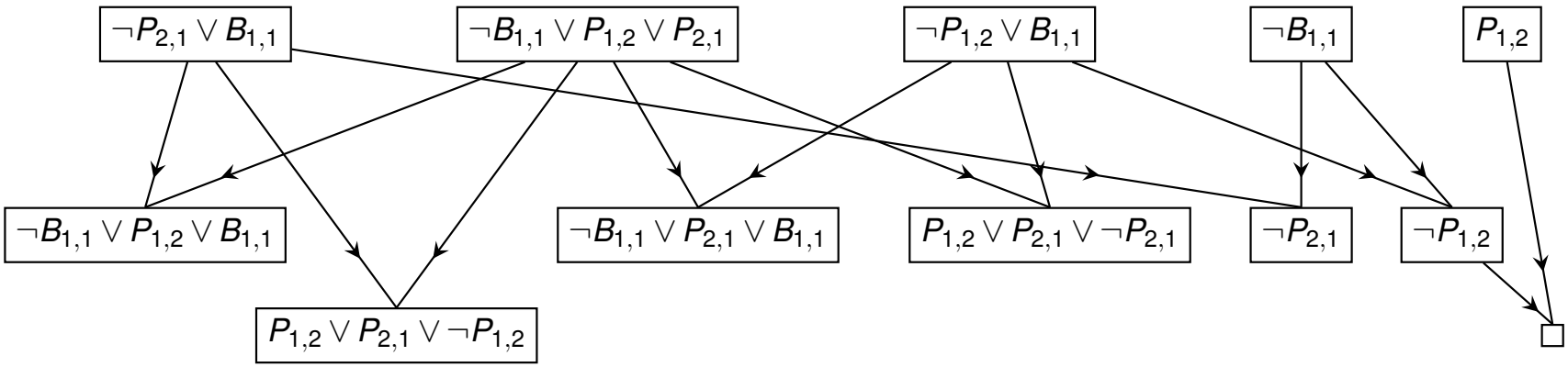
Rezolucja: algorytm

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

```
function PL-RESOLUTION(KB, α) returns true or false
    clauses ← the set of clauses in the CNF representation of KB ∧ ¬α
    loop
        new ← {}
        for each C_i, C_j in clauses do
            resolvents ← PL-RESOLVE(C_i, C_j)
            if resolvents contains the empty clause then
                return true
            end if
            new ← new ∪ resolvents
        end for
        if new ⊆ clauses then
            return false
        end if
        clauses ← clauses ∪ new
    end loop
end function
```

Rezolucja: przykład

$KB = (B_{1,1} \Leftrightarrow (P_{1,2} \vee P_{2,1})) \wedge \neg B_{1,1}$
$\alpha = \neg P_{1,2}$



- Sprowadzenie $KB \wedge \neg\alpha$ do postaci normalnej koniunkcyjnej
- Dowód przez zaprzeczenie, tzn. pokazanie, że $KB \wedge \neg\alpha$ niespełnialne
- Tak rozumiana rezolucja jest pełna

Procedura Davisa-Putnama

function PL-DAVIS-PUTNAM(KB, α) **returns** *true* or *false*
  $clauses \leftarrow$ the set of CNF clauses representing $KB \wedge \neg\alpha$ (tautologies removed)
  $sets \leftarrow \{clauses\}$; $newsets \leftarrow \{\}$
  while *sets* is not empty **do**
    if $\{\} \in sets$ **then return** *false*
    end if
    $sets \leftarrow \{clauses \in sets : clauses$ does not have contradictory unary clauses p and $\neg p\}$
    for each $clauses \in sets$ **do**
      while $\exists$ unary $l \in clauses$ **do**
        remove clauses containing l and occurences of $\neg l$
      end while
      while some l occurs in *clauses* but $\neg l$ does not **do**
        remove clauses containing l
      end while
      $p \leftarrow$ any propositional variable occuring in *clauses*
      $clauses(p = true) \leftarrow \{C_i' : C_i \in clauses \wedge p \notin C_i \wedge C_i'$ is C_i with removed $\neg p\}$
      $clauses(p = false) \leftarrow \{C_i' : C_i \in clauses \wedge \neg p \notin C_i \wedge C_i'$ is C_i with removed $p\}$
      for each $clauses(\ldots) \in \{clauses(p = true), clauses(p = false)\}$ **do**
        $clauses(\ldots) \leftarrow \{C_i \in clauses(\ldots) : C_i$ is not superclause of any $C_j \in clauses(\ldots)\}$
        $newest \leftarrow newest \cup \{clauses(\ldots)\}$
      end for
    end for
    $sets \leftarrow newsets$; $newsets \leftarrow \{\}$
  end while
  return *true*
end function

Dziękuję za uwagę!