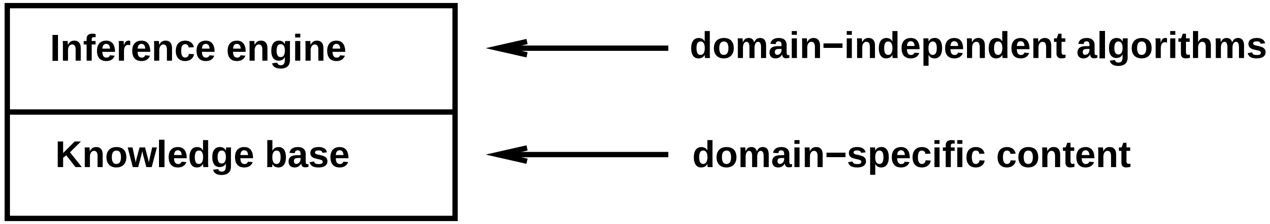
Sztuczna Inteligencja i Systemy Doradcze

Wnioskowanie w rachunku zdań

Bazy wiedzy



Baza wiedzy = zbiór zdań w języku formalnym

Deklaratywne podejście do budowania systemu:

POWIEDZ systemowi to co potrzebuje wiedzieć

Potem 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

Swiat 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

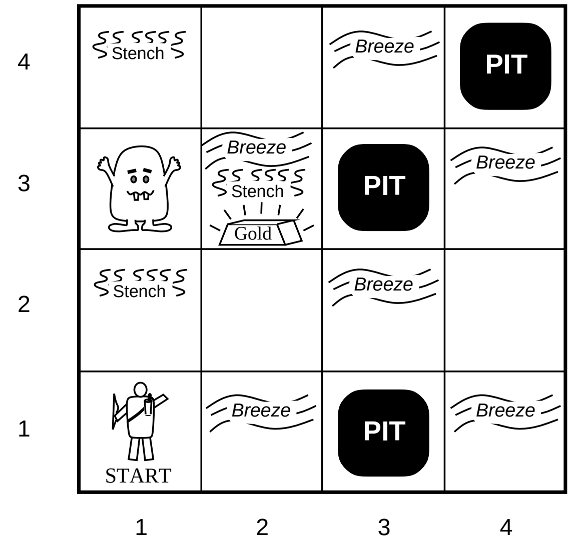
Złoto się błyszczy

Strzał w kierunku Wumpusa zabija go

Strzał wykorzystuje jedyną strzałę

Podniesienie powoduje zabranie złota,
jeśli jest na tym samym polu

Upuszczenie powoduje pozostawienie złota



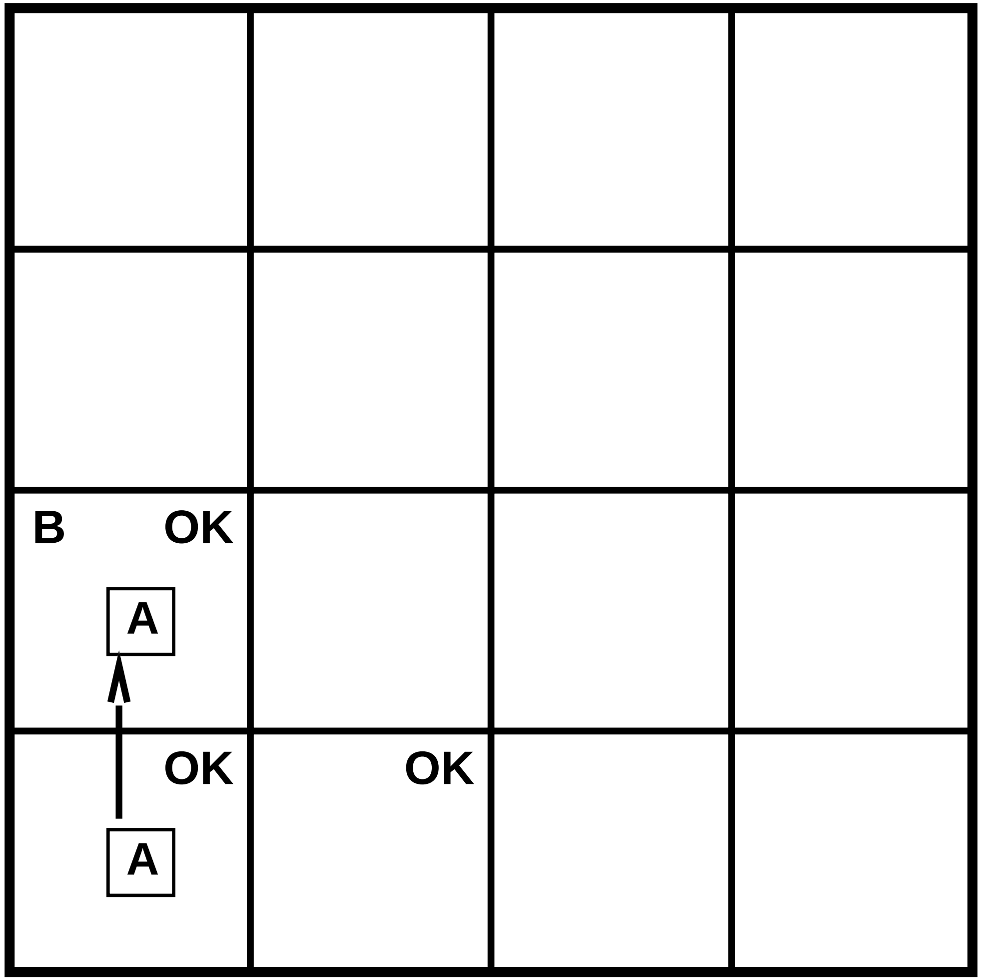
Obserwacje Wiatr, Błysk, Zapach

Działania Skręć w lewo, Skręć w prawo,
Naprzód, Podnieś, Upuść, Strzał

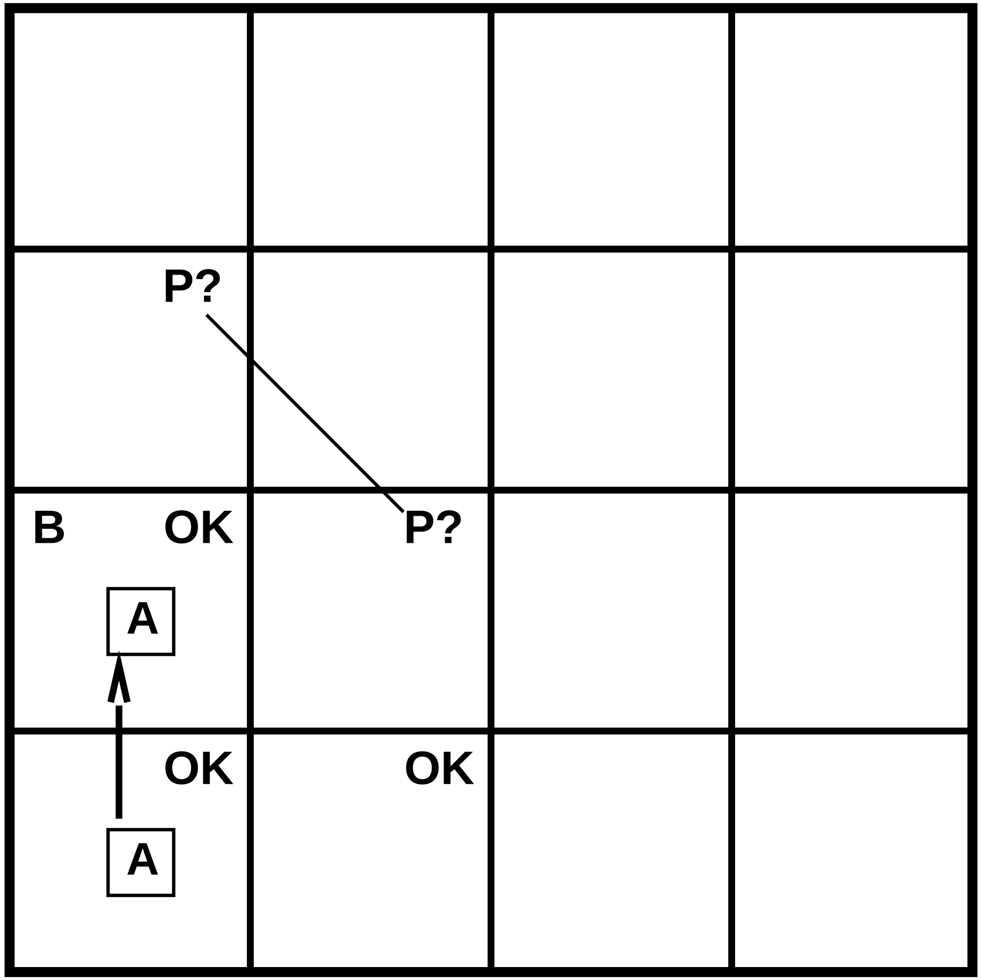
Swiat Wumpusa: eksploracja

OK			
OK $\boxed{A}$	OK		

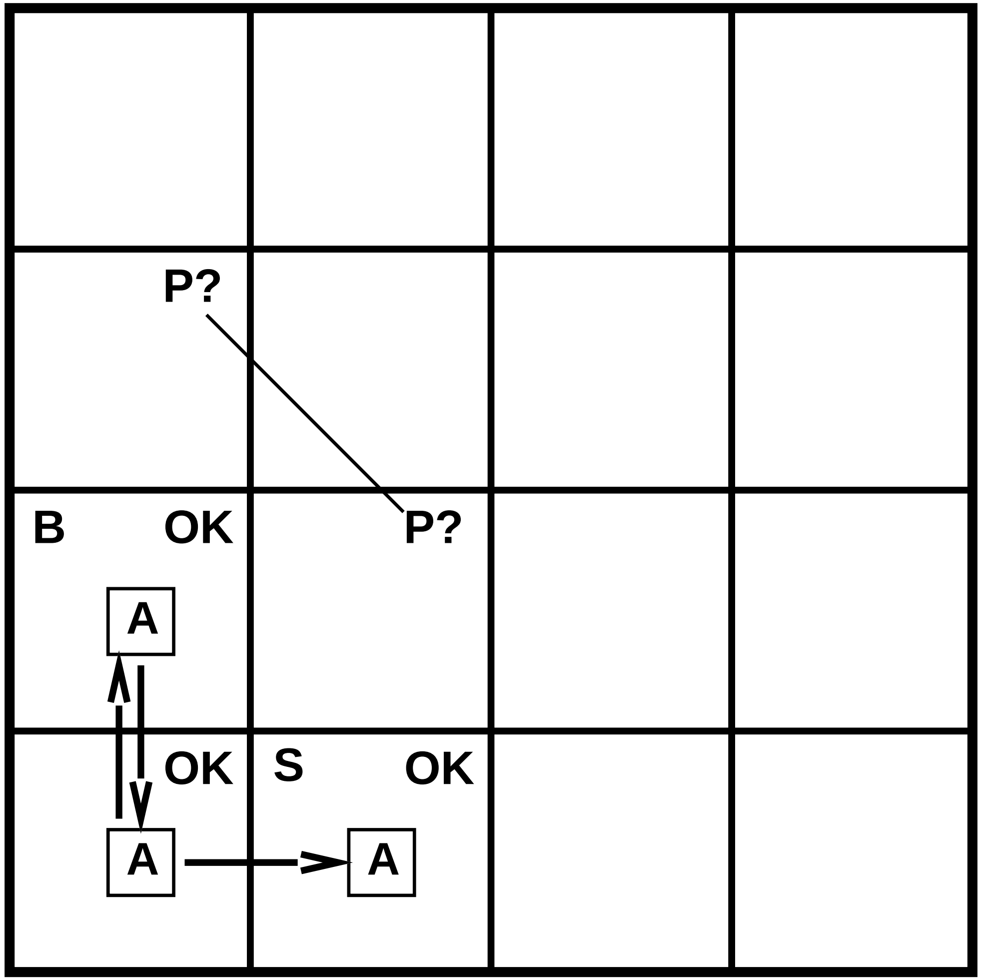
Swiat Wumpusa: eksploracja



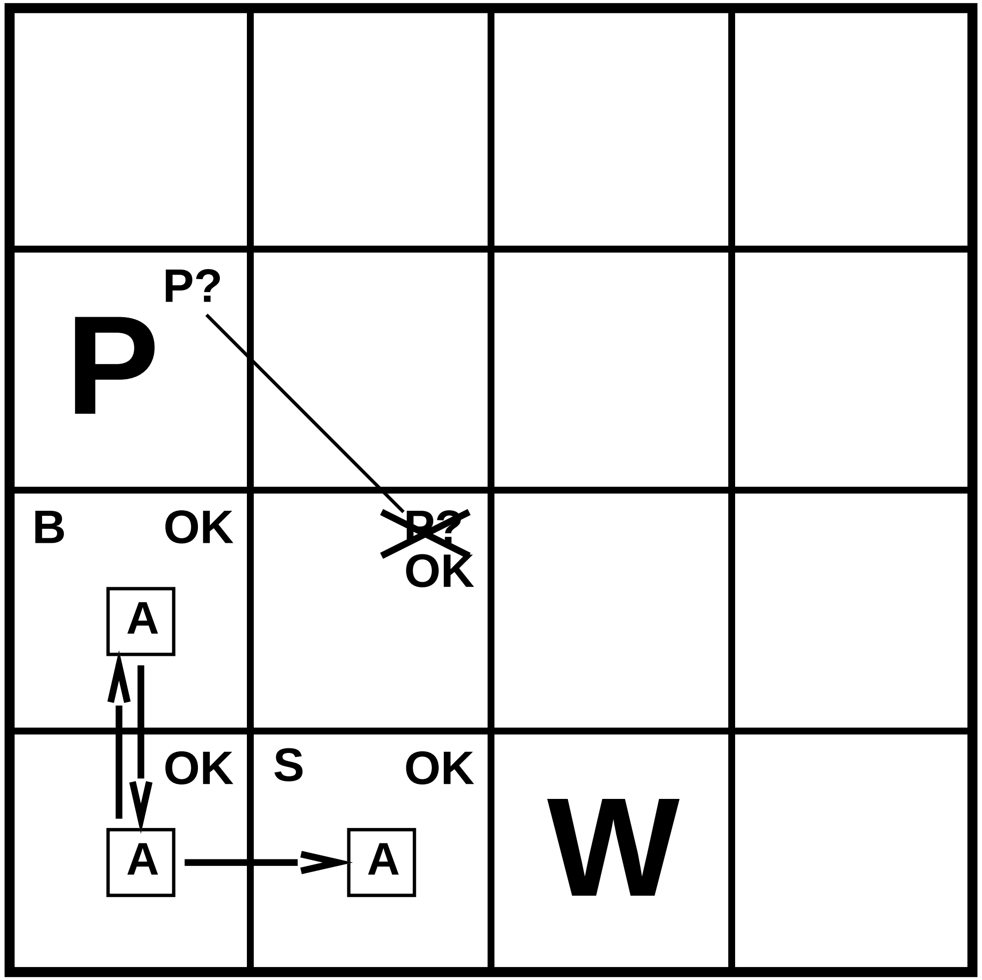
Swiat Wumpusa: eksploracja



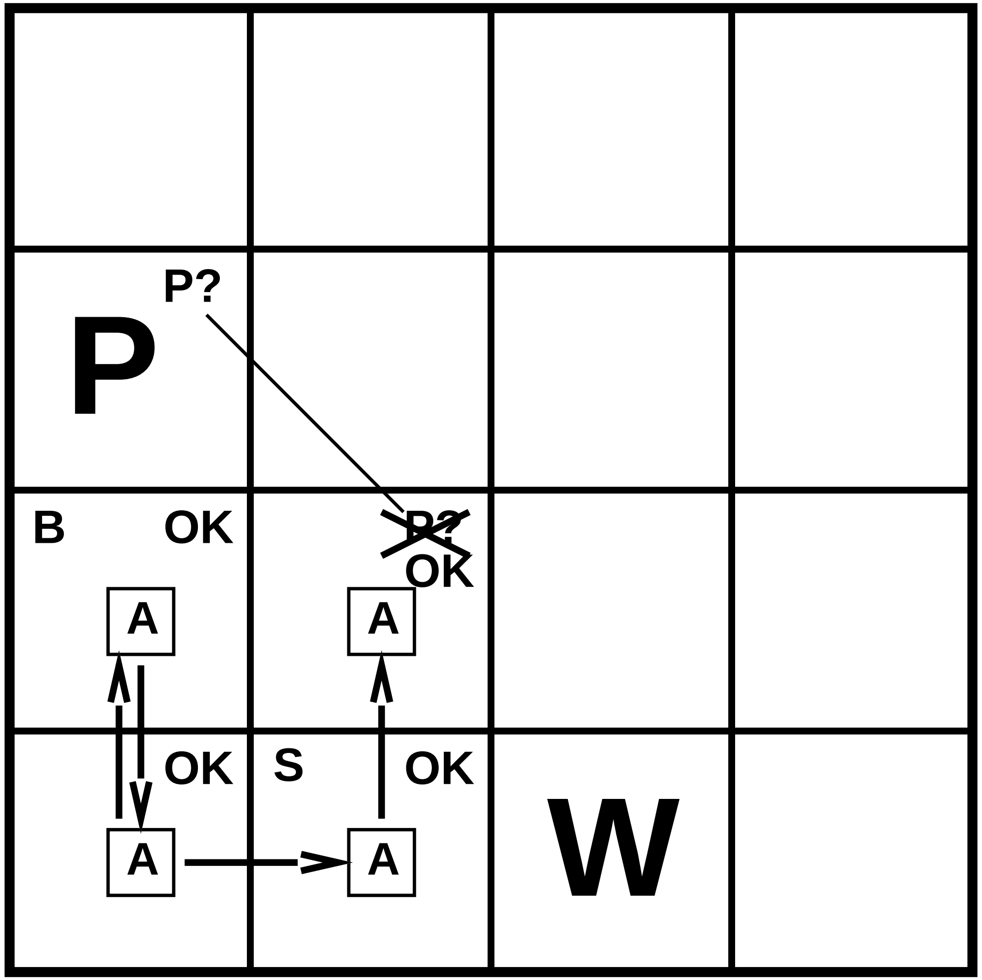
Swiat Wumpusa: eksploracja



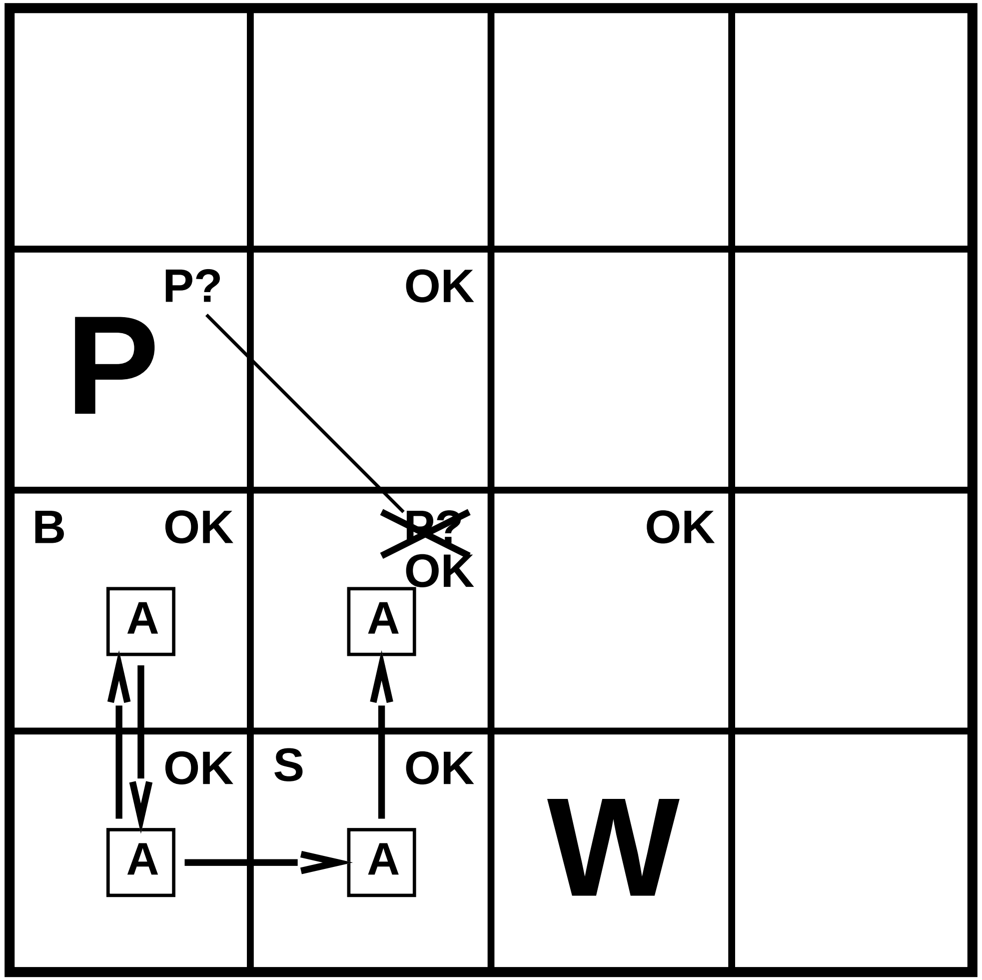
Swiat Wumpusa: eksploracja



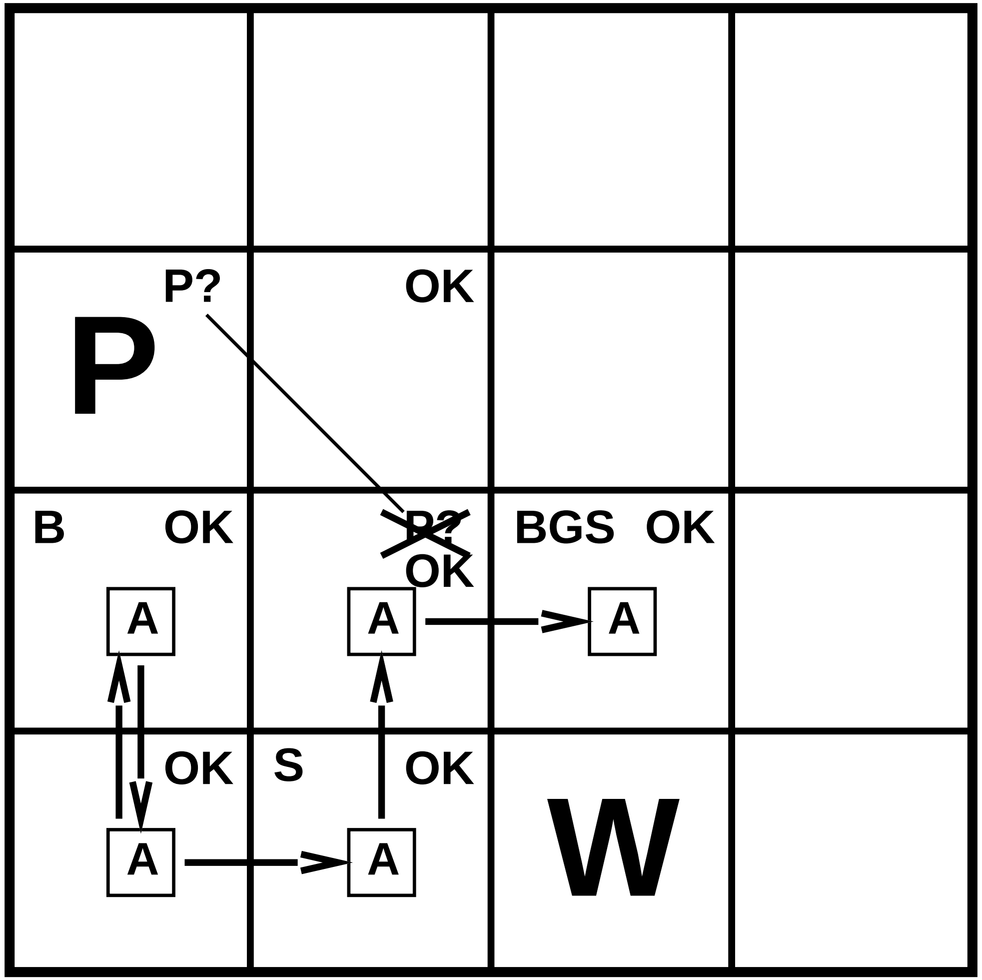
Swiat Wumpusa: eksploracja



Swiat Wumpusa: eksploracja



Swiat Wumpusa: eksploracja



Logika

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

Syntaktyka definiuje zdania w języku

Semantyka definiuje "znaczenie" zdań;
tzn. definiuje prawdziwość zdań w opisywanym świecie

Np. język arytmetyki

$x + 2 \geq y$ jest zdaniem; $x2 + y >$ nie jest zdaniem

$x + 2 \geq y$ jest prawdziwe wtw $x + 2$ jest nie mniejsze niż liczba y

$x + 2 \geq y$ jest prawdziwe w świecie, gdzie $x = 7$, $y = 1$
$x + 2 \geq y$ jest nieprawdziwe w świecie, gdzie $x = 0$, $y = 6$

Logiczna konsekwencja

Logiczna konsekwencja oznacza, że jeden fakt *wynika* z innego:

$$KB \models \alpha$$

α jest logiczną konsekwencją bazy wiedzy KB
wtedy i tylko wtedy
α jest prawdziwe we wszystkich światach, w których KB jest prawdziwe

Np. logiczną konsekwencją baza wiedzy KB zawierającej "Giants wygrali" i "Reds wygrali" jest zdanie "Giants lub Reds wygrali"

Np. $4 = x + y$ jest logiczną konsekwencją $x + y = 4$

Logiczna konsekwencja jest relacją pomiędzy zdaniami (*syntaktyczną*) która opiera się na *semantyce*

Uwaga: umysł analizuje *syntaktykę* (pewnego rodzaju)

Modele

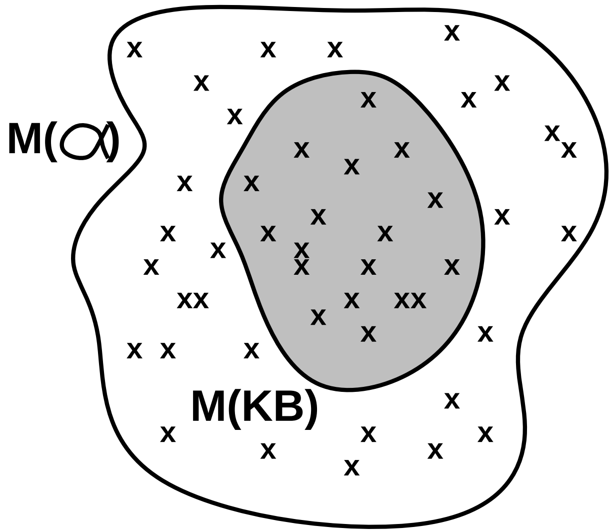
Logicy myślą zazwyczaj w terminach modeli, które formalnie są ustrukturalnionymi światami względem których można wyznaczać prawdziwość

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

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

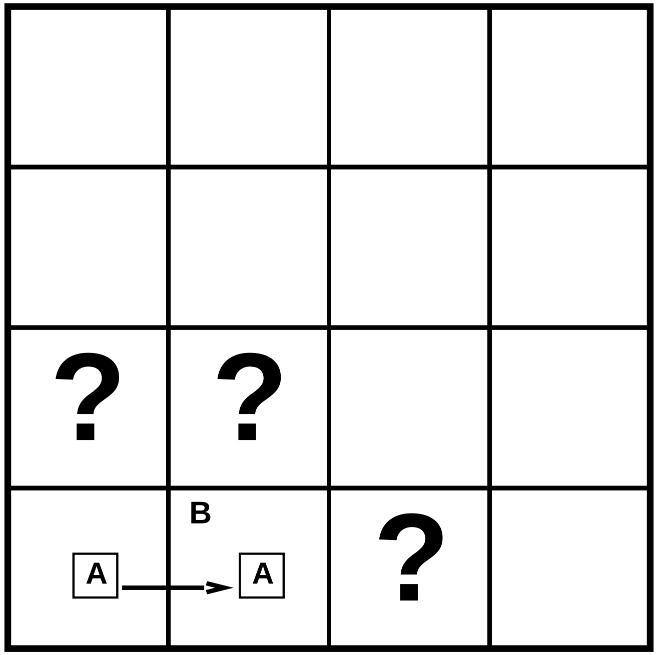
Wtedy $KB \models \alpha$ wtw $M(KB) \subseteq M(\alpha)$

Np. $KB =$ Giants i Reds wygrali
$\alpha =$ Giants wygrali



Swiat Wumpusa: logiczna konsekwencja

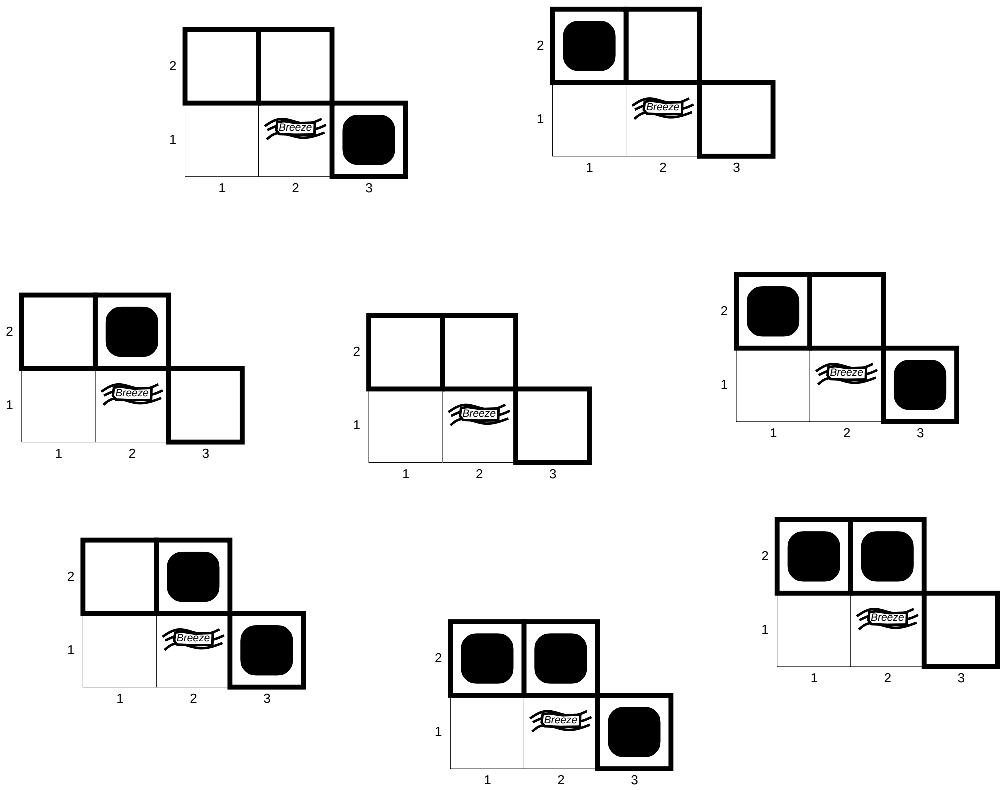
Sytuacja po zbadaniu pola [1,1], przesunięciu w prawo i wykryciu wiatru w [2,1]



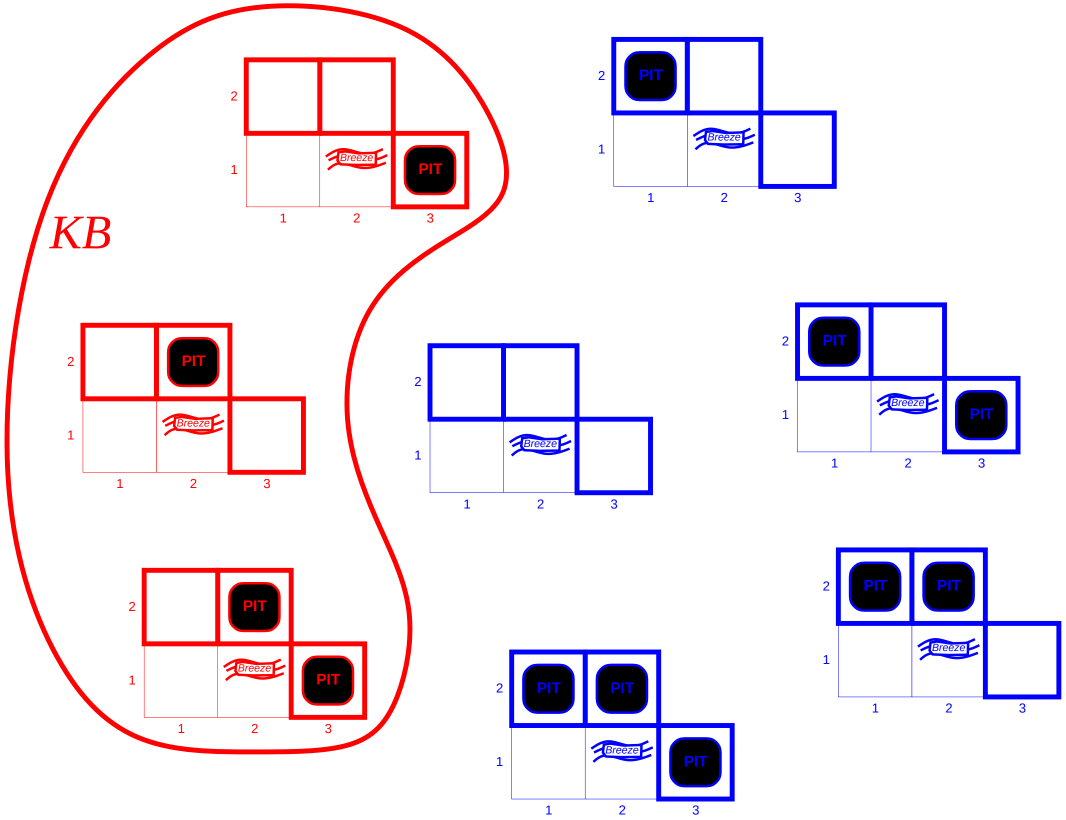
Rozważamy możliwe modele dla pól '?' dotyczące informacji, czy na tych polach są pułapki

3 binarne wybory $\Rightarrow$ 8 możliwych modeli

Modele w swiecie Wumpusa

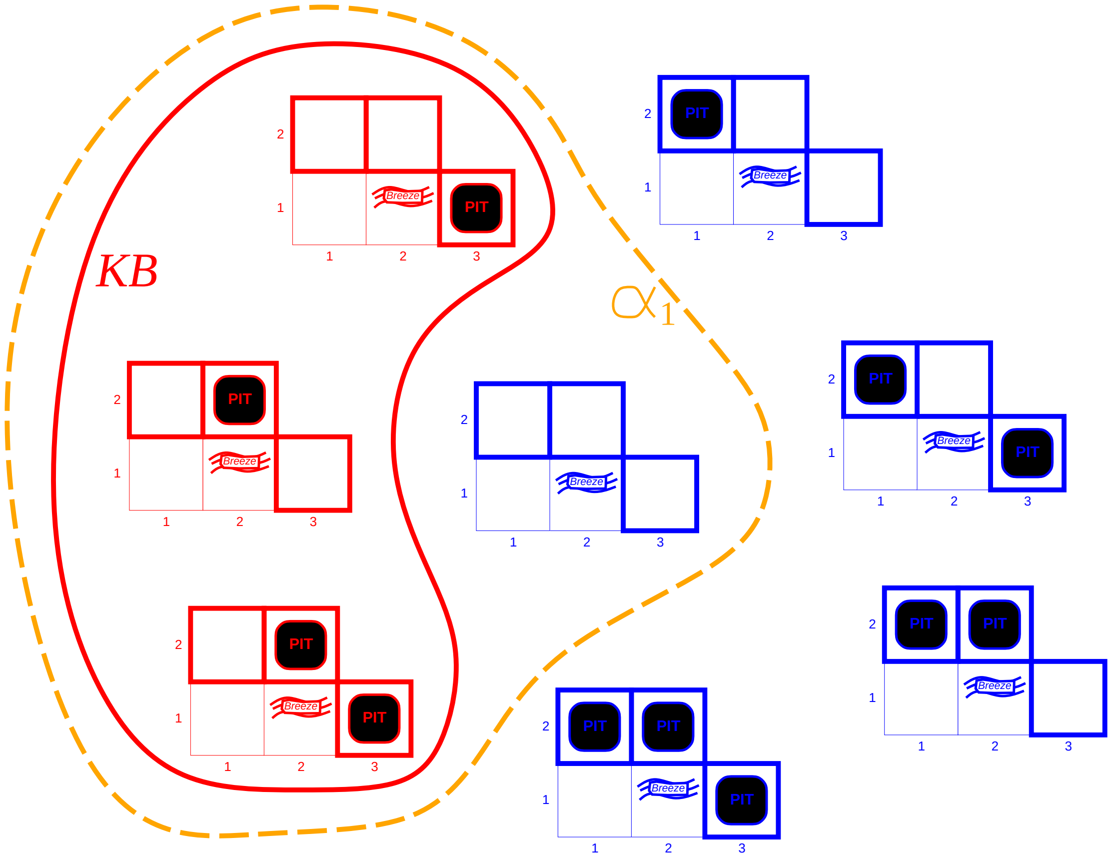


Modele w swiecie Wumpusa



KB = reguły świata Wumpusa + obserwacje

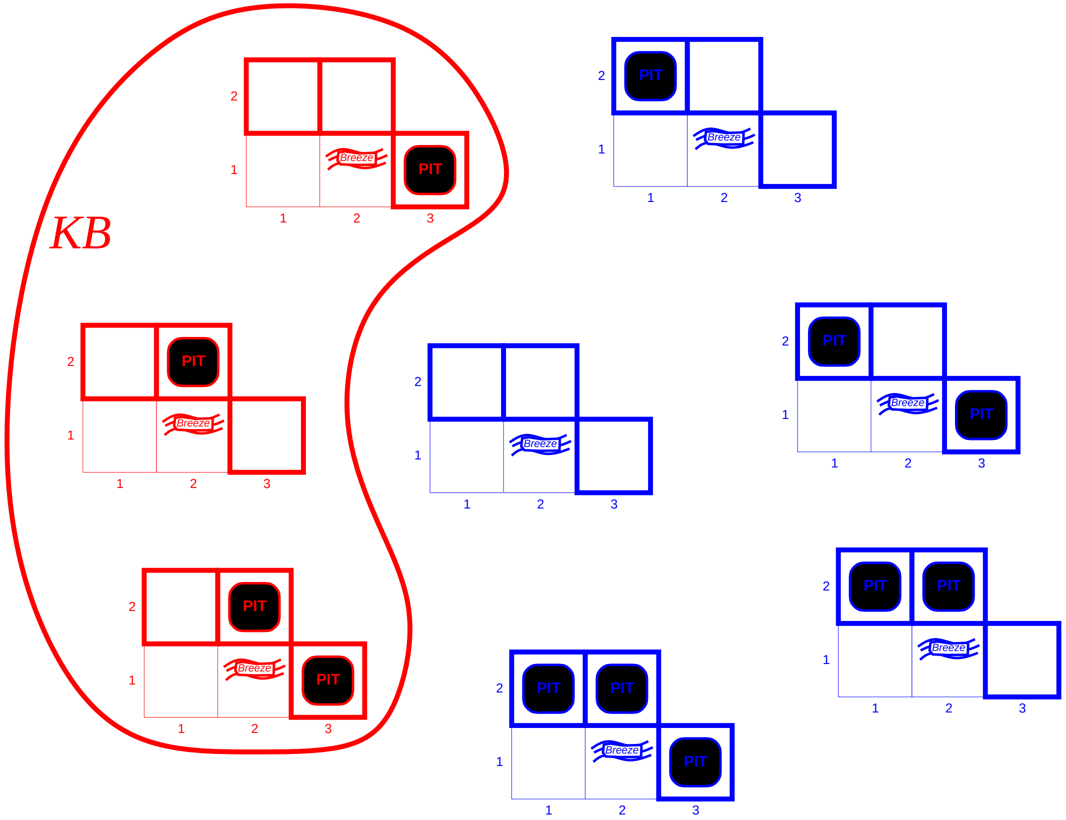
Modele w swiecie Wumpusa



KB = reguły świata Wumpusa + obserwacje

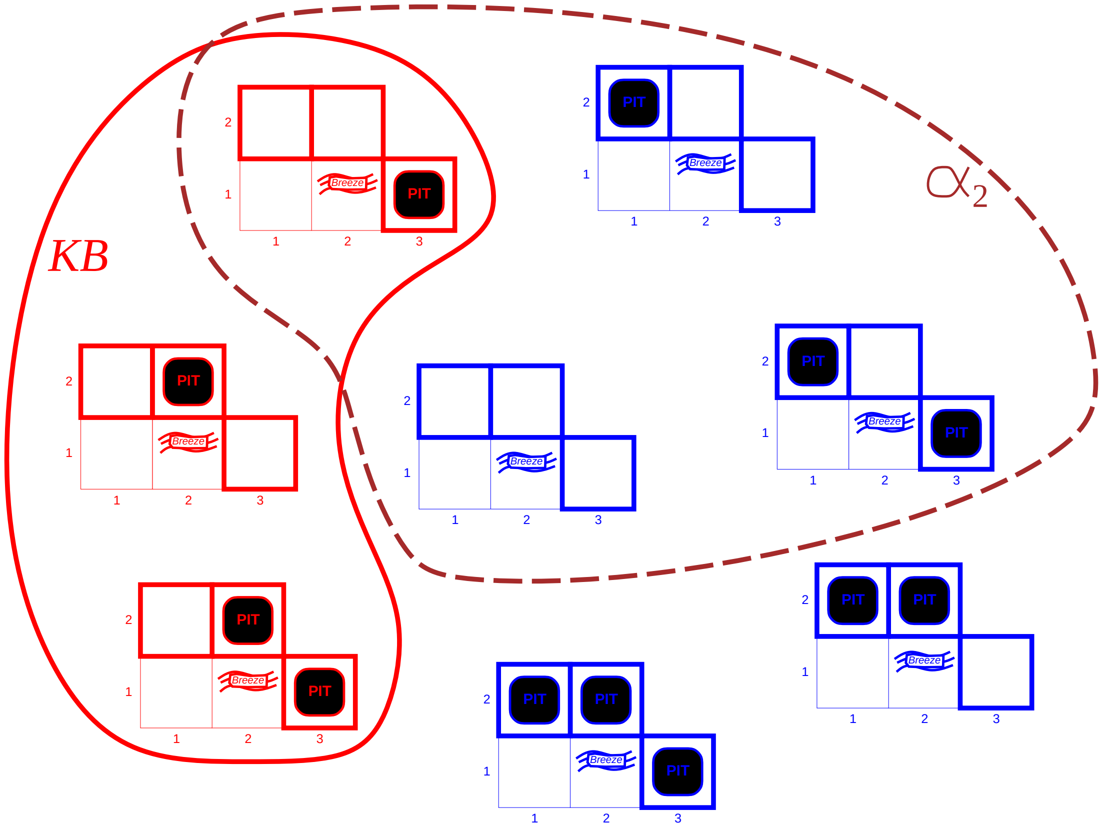
α_1 = "[1,2] jest bezpieczny", $KB \models \alpha_1$, dowód przez sprawdzenie modeli

Modele w swiecie Wumpusa



KB = reguły świata Wumpusa + obserwacje

Modele w swiecie Wumpusa



KB = reguły świata Wumpusa + obserwacje

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

Wnioskowanie

$KB \vdash_i \alpha$ = zdanie α może być wyprowadzone z KB procedurą i

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

Poprawność: i jest poprawne, jeśli
zawsze kiedy $KB \vdash_i \alpha$, to też $KB \models \alpha$

Pełność: i jest pełne jeśli
zawsze kiedy $KB \models \alpha$, to też $KB \vdash_i \alpha$

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.

Logika zdaniowa: syntaktyka

Logika zdaniowa jest najprostszą logiką — ilustruje podstawowe pomysły

Symbole zdaniowe P_1, P_2 itd. są zdaniami

Jeśli S jest zdaniem, $\neg S$ jest zdaniem (negacja)

Jeśli S_1 i S_2 są zdaniami, $S_1 \wedge S_2$ jest zdaniem (koniunkcja)

Jeśli S_1 i S_2 są zdaniami, $S_1 \vee S_2$ jest zdaniem (alternatywa)

Jeśli S_1 i S_2 są zdaniami, $S_1 \Rightarrow S_2$ jest zdaniem (implikacja)

Jeśli S_1 i S_2 są zdaniami, $S_1 \Leftrightarrow S_2$ jest zdaniem (równoważność)

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:

$\neg S$	jest prawdziwe wtw	S	jest nieprawdziwe		
$S_1 \wedge S_2$	jest prawdziwe wtw	S_1	jest prawdziwe i	S_2	jest prawdziwe
$S_1 \vee S_2$	jest prawdziwe wtw	S_1	jest prawdziwe lub	S_2	jest prawdziwe
$S_1 \Rightarrow S_2$	jest prawdziwe wtw	S_1	jest nieprawdz. lub	S_2	jest prawdziwe
tzn.	jest nieprawdz. wtw	S_1	jest prawdziwe i	S_2	jest nieprawdz.
$S_1 \Leftrightarrow S_2$	jest prawdziwe wtw	$S_1 \Rightarrow S_2$	jest prawdziwe i	$S_2 \Rightarrow S_1$	jest prawdziwe

Prosty rekurencyjny proces określający prawdziwość dowolnego zdania, np.
$\neg P_{1,2} \wedge (P_{2,2} \vee P_{3,1}) = true \wedge (false \vee true) = true \wedge true = true$

Tabela prawdziwosci dla lacznikow zdaniowych

P	Q	$\neg P$	$P \wedge Q$	$P \vee Q$	$P \Rightarrow Q$	$P \Leftrightarrow Q$
$false$	$false$	$true$	$false$	$false$	$true$	$true$
$false$	$true$	$true$	$false$	$true$	$true$	$false$
$true$	$false$	$false$	$false$	$true$	$false$	$false$
$true$	$true$	$false$	$true$	$true$	$true$	$true$

Zdania w swiecie 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"

Zdania w swiecie 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"

$$\begin{aligned} B_{1,1} &\Leftrightarrow (P_{1,2} \vee P_{2,1}) \\ B_{2,1} &\Leftrightarrow (P_{1,1} \vee P_{2,2} \vee P_{3,1}) \end{aligned}$$

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

Tabela prawdziwosci dla wnioskowania

$B_{1,1}$	$B_{2,1}$	$P_{1,1}$	$P_{1,2}$	$P_{2,1}$	$P_{2,2}$	$P_{3,1}$	KB	α_1
$false$	$false$	$false$	$false$	$false$	$false$	$false$	$false$	$true$
$false$	$false$	$false$	$false$	$false$	$false$	$true$	$false$	$true$
$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$
$false$	$true$	$false$	$false$	$false$	$false$	$false$	$false$	$true$
$false$	$true$	$false$	$false$	$false$	$false$	$true$	$\underline{true}$	$\underline{true}$
$false$	$true$	$false$	$false$	$false$	$true$	$false$	$\underline{true}$	$\underline{true}$
$false$	$true$	$false$	$false$	$false$	$true$	$true$	$\underline{true}$	$\underline{true}$
$false$	$true$	$false$	$false$	$true$	$false$	$false$	$false$	$true$
$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$
$true$	$true$	$true$	$true$	$true$	$true$	$true$	$false$	$false$

Rownowaznosc logiczna

Dwa zdania są logicznie równoważne wtw prawdziwe w tych samych modelach:

$\alpha \equiv \beta$ wtedy i tylko wtedy $\alpha \models \beta$ and $\beta \models \alpha$

$$
\begin{aligned}
(\alpha \land \beta) &\equiv (\beta \land \alpha) \quad \text{przemienność } \land \\
(\alpha \lor \beta) &\equiv (\beta \lor \alpha) \quad \text{przemienność } \lor \\
((\alpha \land \beta) \land \gamma) &\equiv (\alpha \land (\beta \land \gamma)) \quad \text{łączność } \land \\
((\alpha \lor \beta) \lor \gamma) &\equiv (\alpha \lor (\beta \lor \gamma)) \quad \text{łączność } \lor \\
\neg(\neg\alpha) &\equiv \alpha \quad \text{elimnacja podwójnej negacji} \\
(\alpha \Rightarrow \beta) &\equiv (\neg\beta \Rightarrow \neg\alpha) \quad \text{zaprzeczenie} \\
(\alpha \Rightarrow \beta) &\equiv (\neg\alpha \lor \beta) \quad \text{eliminacja implikacji} \\
(\alpha \Leftrightarrow \beta) &\equiv ((\alpha \Rightarrow \beta) \land (\beta \Rightarrow \alpha)) \quad \text{eliminacja równoważności} \\
\neg(\alpha \land \beta) &\equiv (\neg\alpha \lor \neg\beta) \quad \text{prawo de Morgana} \\
\neg(\alpha \lor \beta) &\equiv (\neg\alpha \land \neg\beta) \quad \text{prawo de Morgana} \\
(\alpha \land (\beta \lor \gamma)) &\equiv ((\alpha \land \beta) \lor (\alpha \land \gamma)) \quad \text{rozdzielność } \land \text{ względem } \lor \\
(\alpha \lor (\beta \land \gamma)) &\equiv ((\alpha \lor \beta) \land (\alpha \lor \gamma)) \quad \text{rozdzielność } \lor \text{ względem } \land
\end{aligned}
$$

Tautologie i spelnialnosc

Zdanie jest tautologią jeśli jest prawdziwe we *wszystkich* modelach,

np. $True$, $A \vee \neg A$, $A \Rightarrow A$, $(A \wedge (A \Rightarrow B)) \Rightarrow B$

Tautologie są zwiazązne z Twierdzeniem o Dedukcji:

$KB \models \alpha$ wtedy i tylko wtedy $(KB \Rightarrow \alpha)$ 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. $A \wedge \neg A$

Spełnialność jest związana z wnioskowaniem przez *sprowadzenie do sprzeczności*:

$KB \models \alpha$ wtedy i tylko wtedy $(KB \wedge \neg\alpha)$ jest niespełnialne

Metody dowodzenia

Metody dowodzenia można podzielić na dwie kategorie:

Sprawdzanie modeli

– Przeszukiwanie przestrzeni wartościowań

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

Metody dowodzenia: algorytmy

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

Zastosowanie reguł wnioskowania
- Forward chaining (ograniczone do klauzul Horna)
- Backward chaining (ograniczone do klauzul Horna)
- Rezolucja

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($KB, \alpha, symbols, [\,]$)

---

function TT-CHECK-ALL($KB, \alpha, symbols, model$) **returns** *true* or *false*
  if EMPTY?(*symbols*) **then**
    if PL-TRUE?($KB, model$) **then return** PL-TRUE?($\alpha, model$)
    else return *true*
  else do
    P ← FIRST(*symbols*); *rest* ← REST(*symbols*)
    return TT-CHECK-ALL($KB, \alpha, rest$, EXTEND($P, true, model$)) **and**
        TT-CHECK-ALL($KB, \alpha, rest$, EXTEND($P, false, model$))

$O(2^n)$ 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) $\Rightarrow$ symbol

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

Modus Ponens (dla postaci Horna): pełne dla baz wiedzy Horna

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

Można użyć algorytmów forward chaining lub backward chaining.
Oba algorytmy są naturalne i wykonują się w czasie *liniowym*

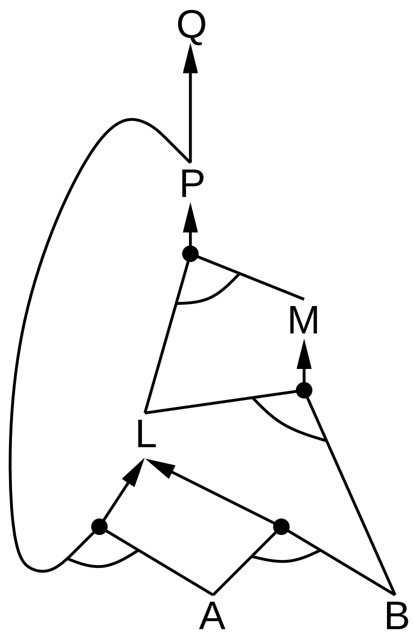
Forward chaining: algorytm

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

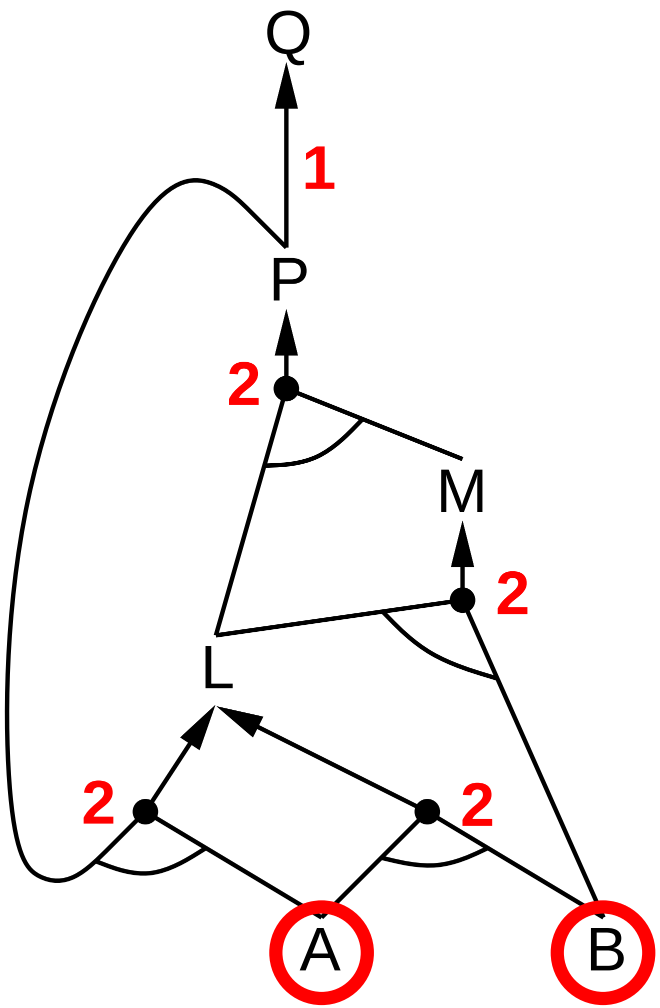
```
function PL-FC-ENTAILS?(KB, q) returns true or false
   local variables: count, a table, indexed by clause, initially the number of premises
                    inferred, a table, indexed by symbol, each entry initially false
                    agenda, a list of symbols, initially the symbols known to be true

   while agenda is not empty do
       p ← POP(agenda)
       unless inferred[p] do
           inferred[p] ← true
           for each Horn clause c in whose premise p appears do
               decrement count[c]
               if count[c] = 0 then do
                   if HEAD[c] = q then return true
                   PUSH(HEAD[c], agenda)
   return false
```

Forward chaining: przyklad

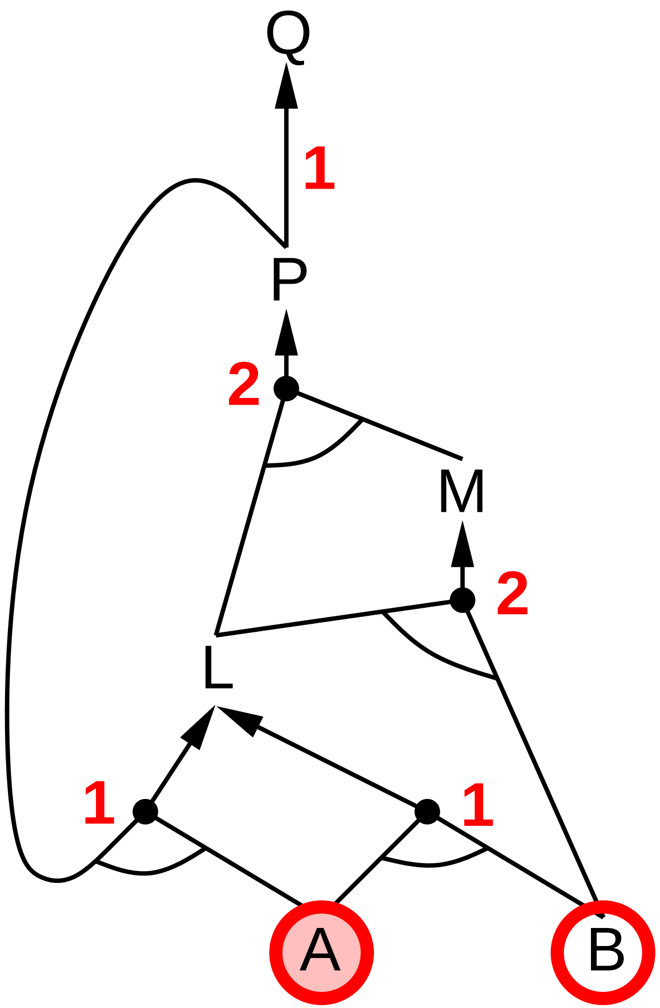
$$P \Rightarrow Q$$
$$L \wedge M \Rightarrow P$$
$$B \wedge L \Rightarrow M$$
$$A \wedge P \Rightarrow L$$
$$A \wedge B \Rightarrow L$$
$$A$$
$$B$$



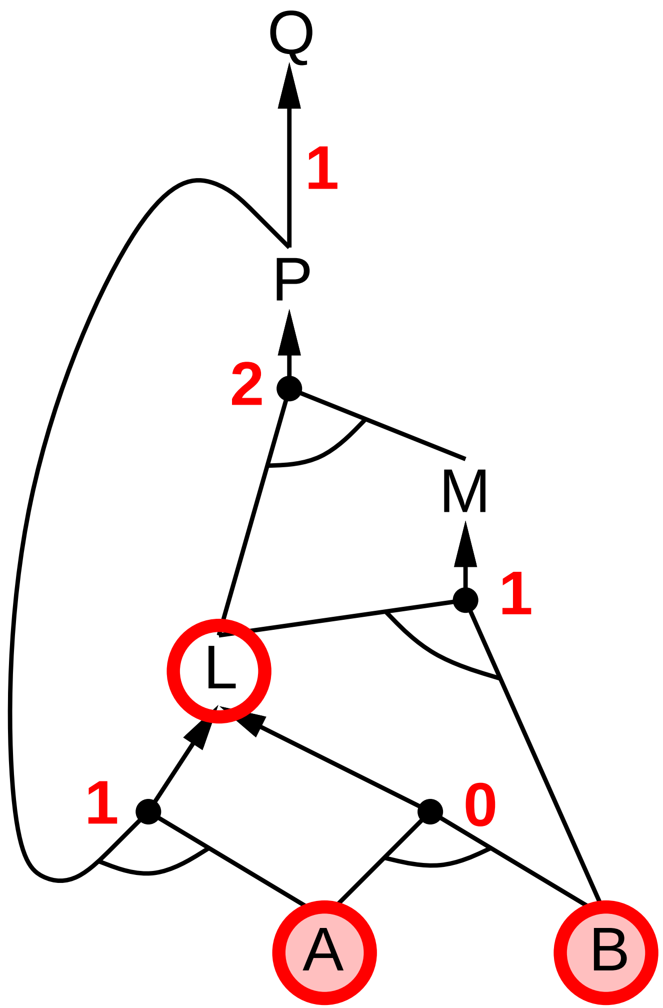
Forward chaining: przyklad



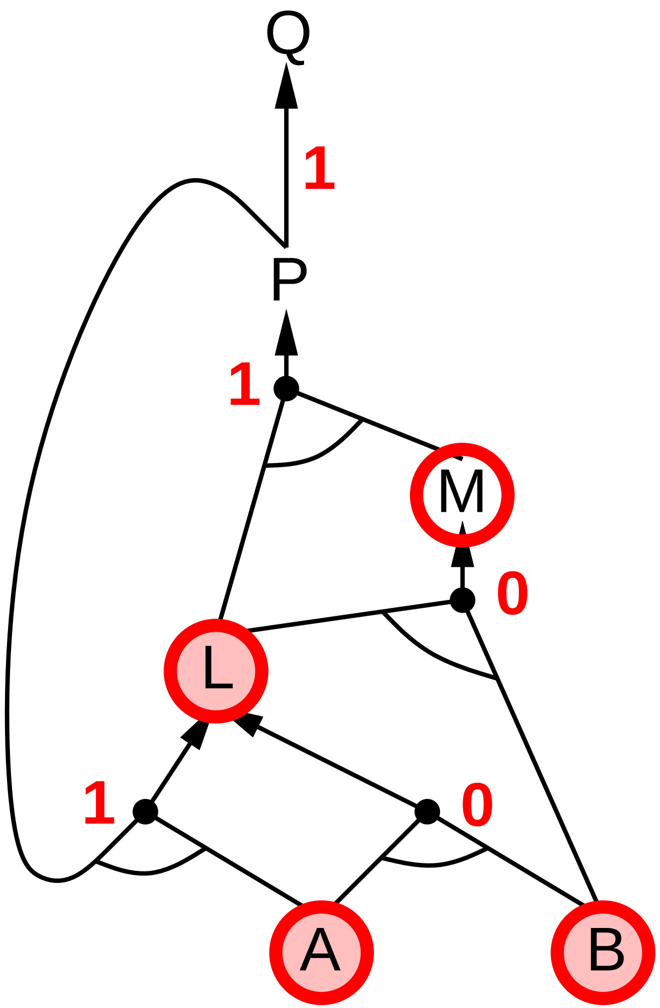
Forward chaining: przyklad



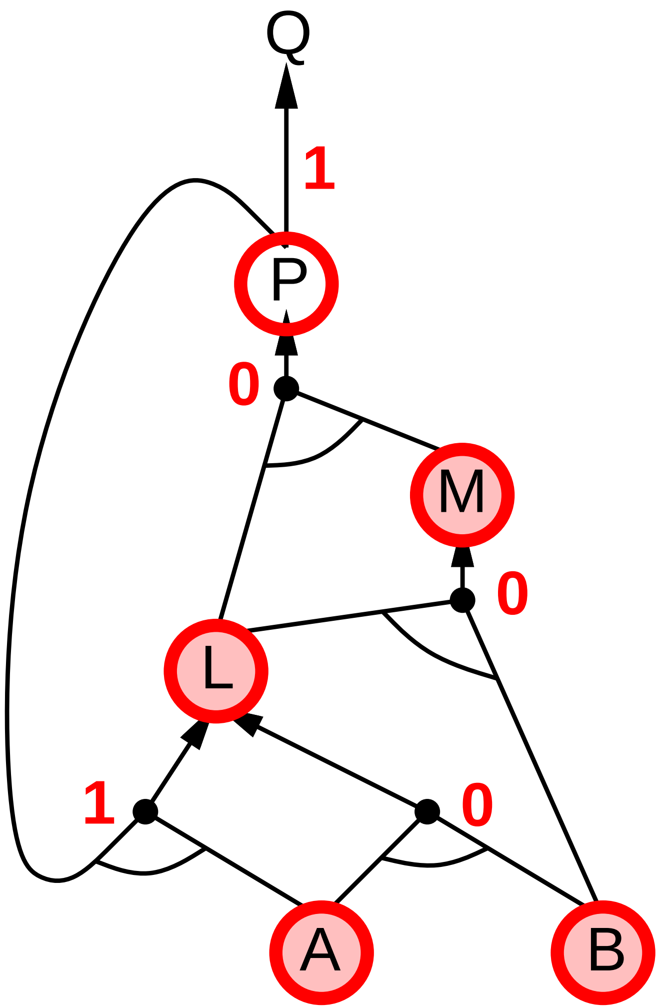
Forward chaining: przyklad



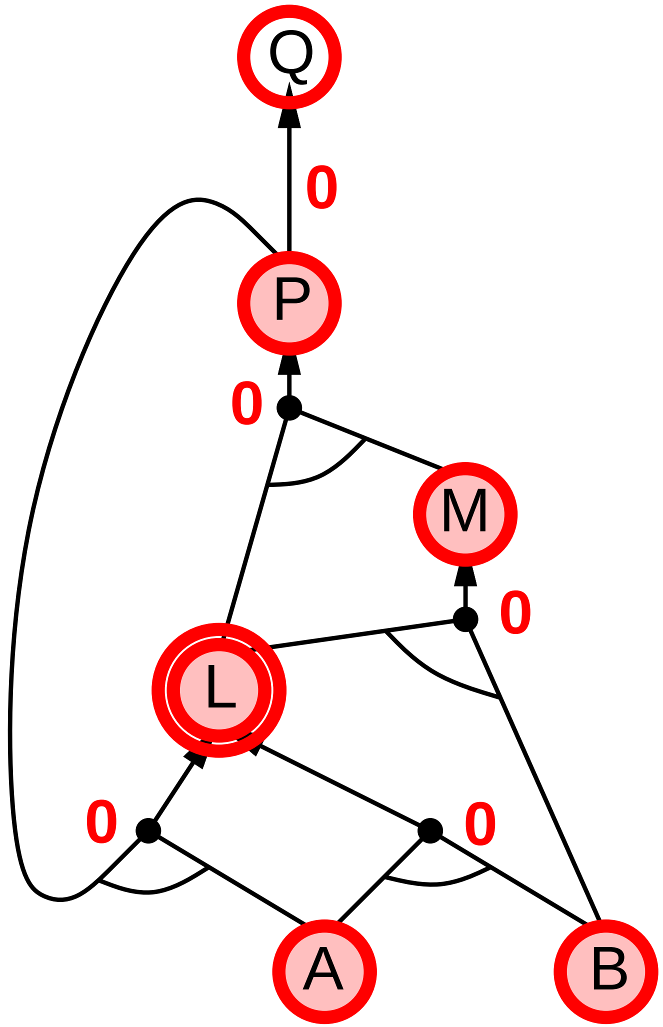
Forward chaining: przyklad



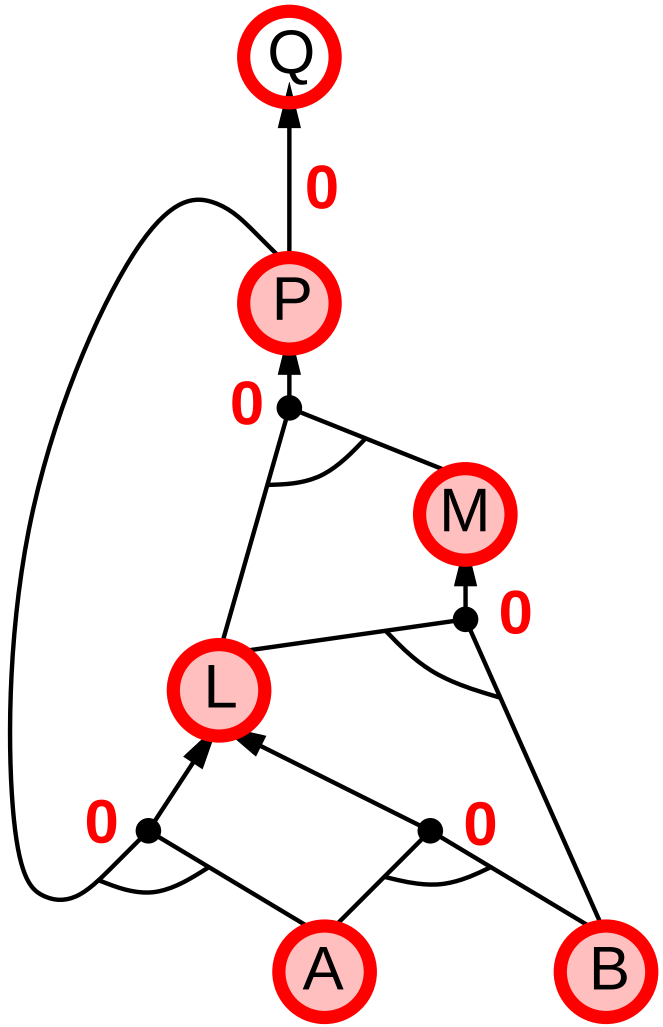
Forward chaining: przyklad



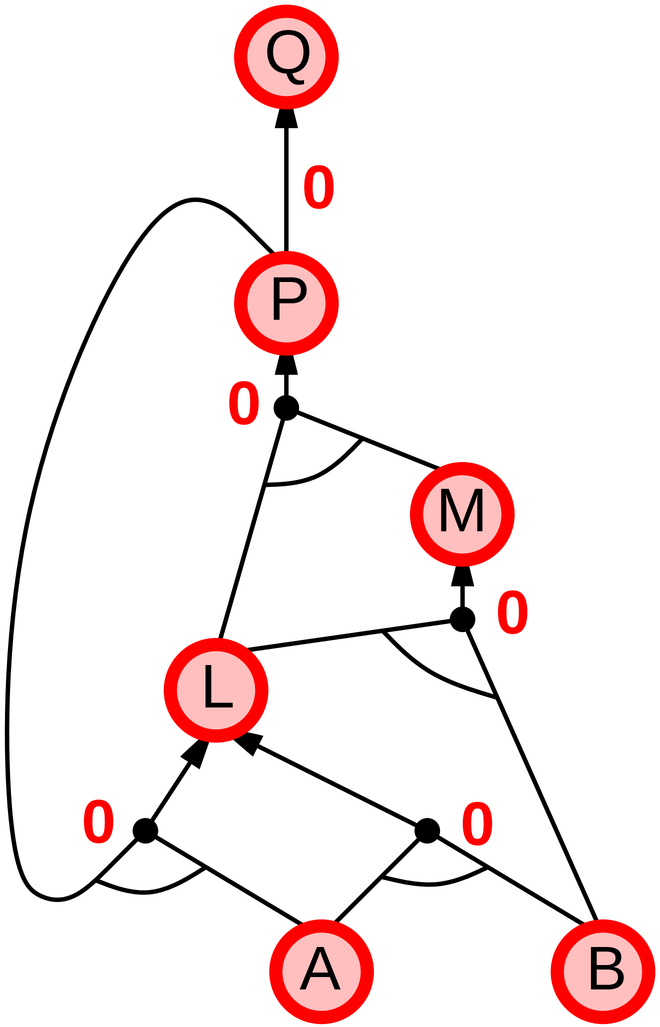
Forward chaining: przyklad



Forward chaining: przyklad



Forward chaining: przyklad



Dowod pelnosci

Forward Chaining wnioskuje każde atomowe zdanie, które jest logiczną konsekwencją KB

1. Algorytm osiąga punkt stały gdzie nie można wywnioskować żadnego nowego zdania

2. Rozważmy stan końcowy jako model m, przypisujący prawdę/fałsz do symboli

3. Każda klauzula w oryginalnej KB jest prawdziwa w m
   Dowód: Przypuśćmy $a_1 \wedge \ldots \wedge a_k \Rightarrow b$ jest nieprawdziwe w m
   Wtedy $a_1 \wedge \ldots \wedge a_k$ jest prawdziwe w m i b jest nieprawdziwe w m
   Zatem algorytm nie osiągnął punktu stałego!

4. Stąd m jest modelem KB

5. Jeśli $KB \models q$, q jest prawdziwe w *każdym* modelu KB, również w m

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

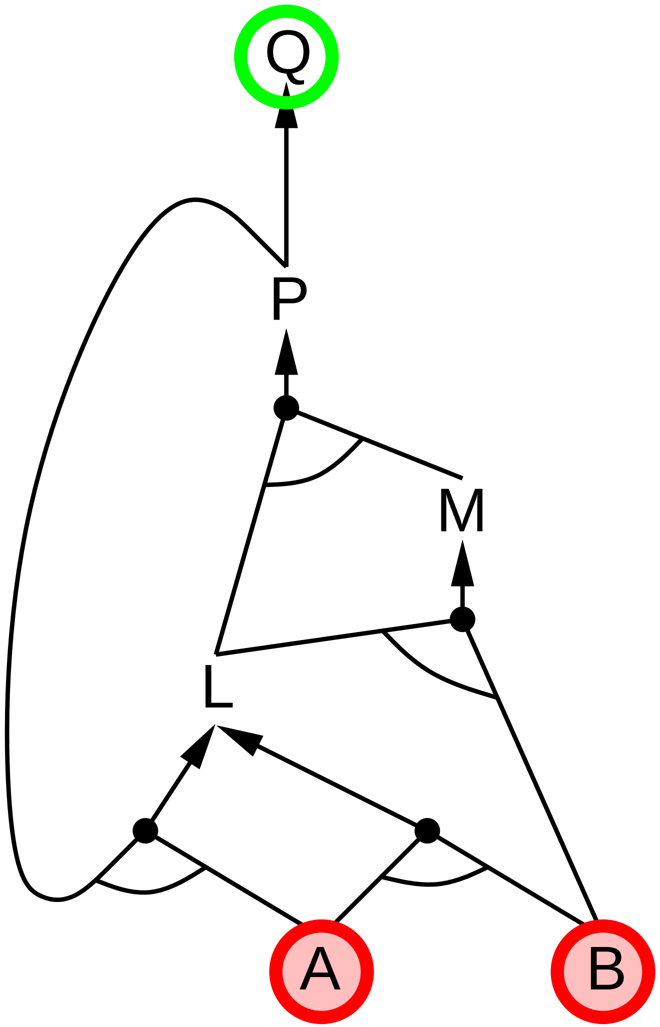
Unikanie pętli: sprawdza, czy nowy podcel nie był już wcześniej wygenerowany

Unikanie powtórzeń: sprawdza, czy dla nowego podcelu

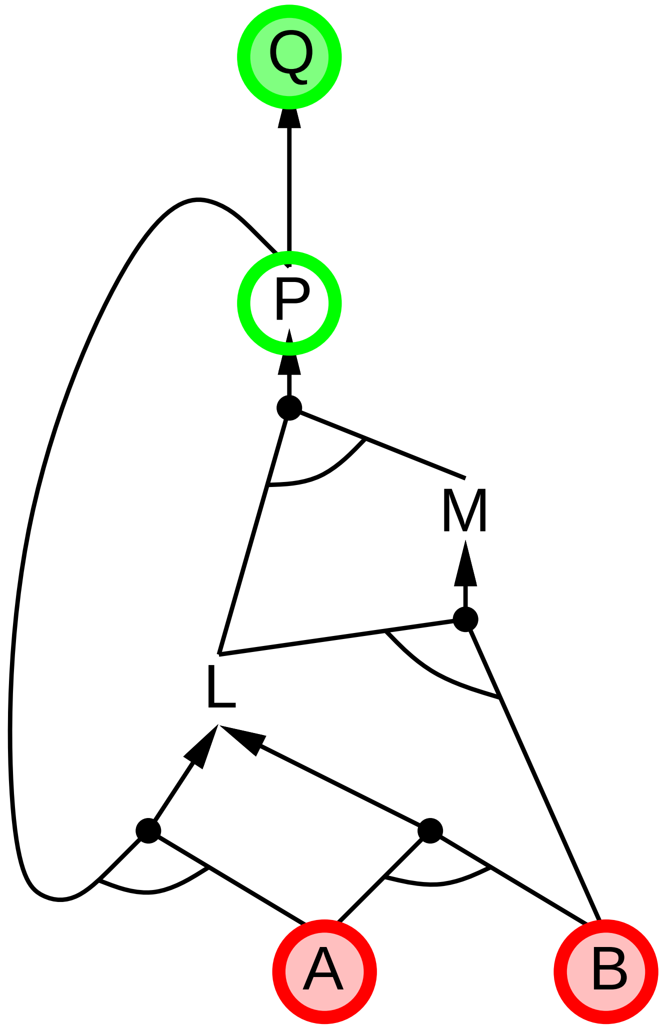
1) była już udowodniona prawdziwość, lub

2) dowód był już podjęty wcześniej i zakończył się porażką

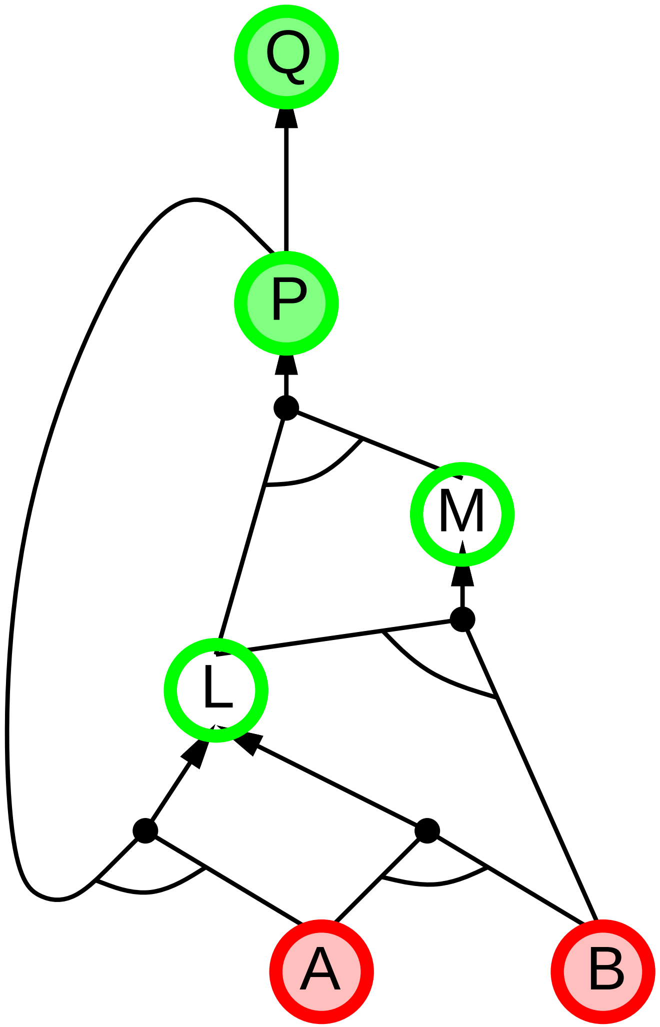
Backward chaining: przyklad



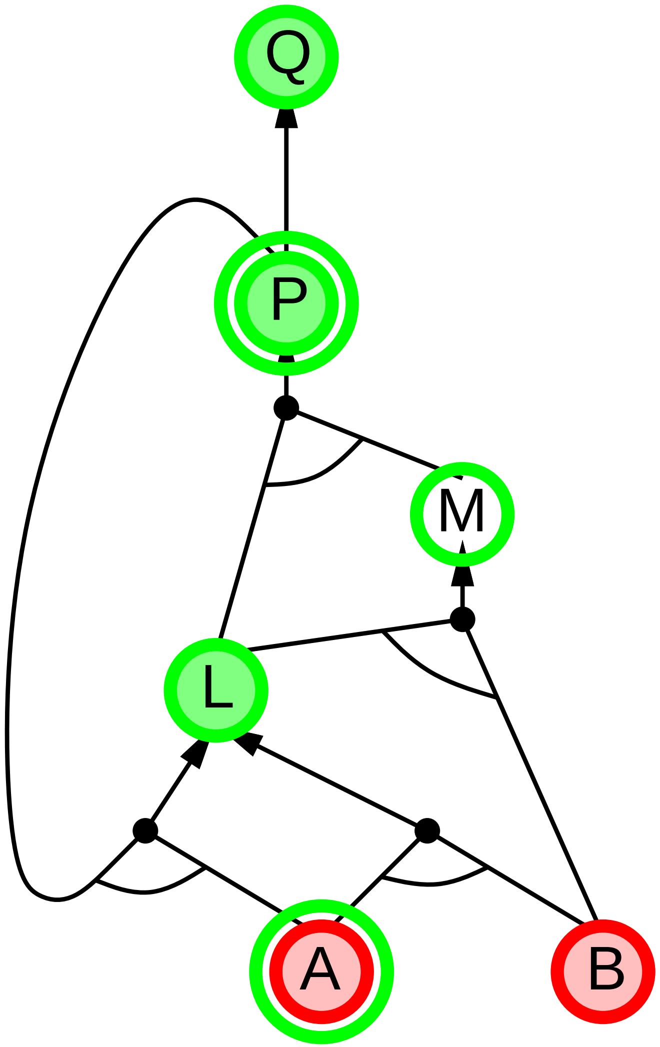
Backward chaining: przyklad



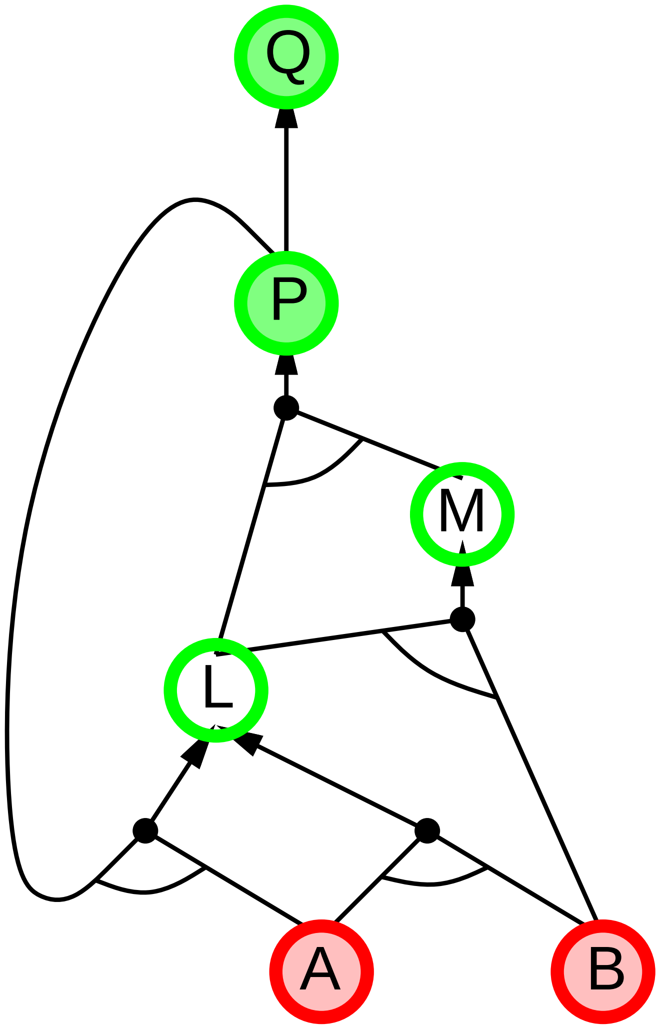
Backward chaining: przyklad



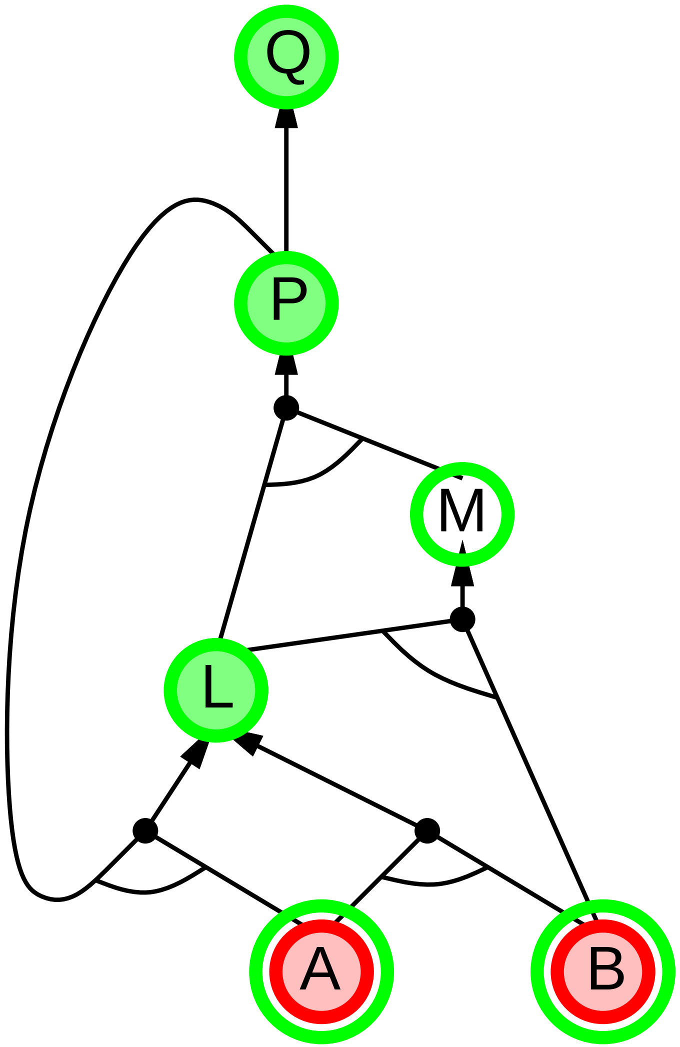
Backward chaining: przyklad



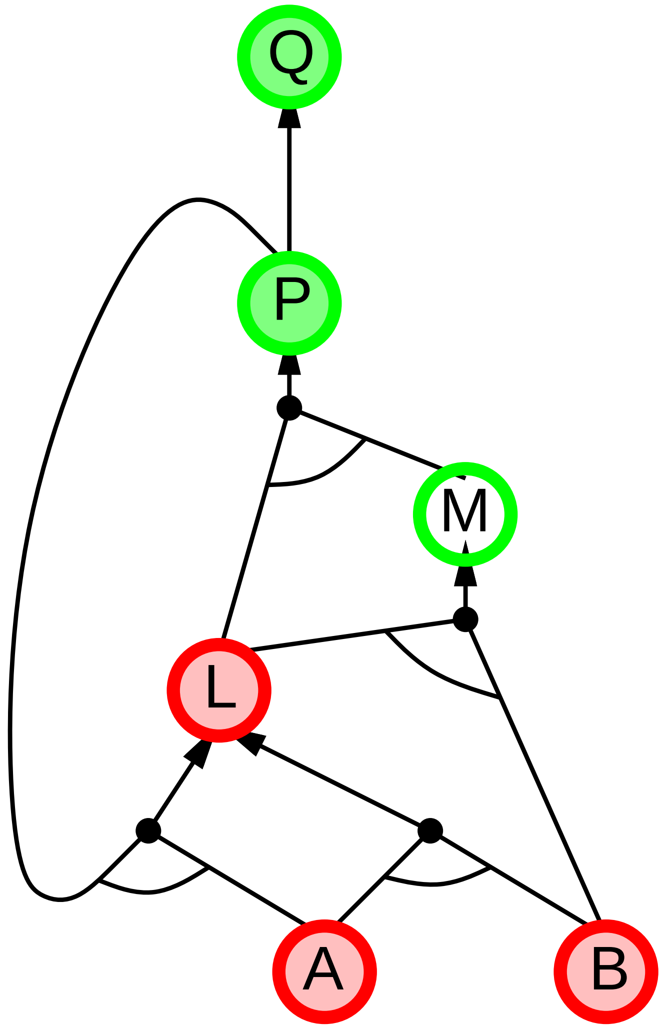
Backward chaining: przyklad



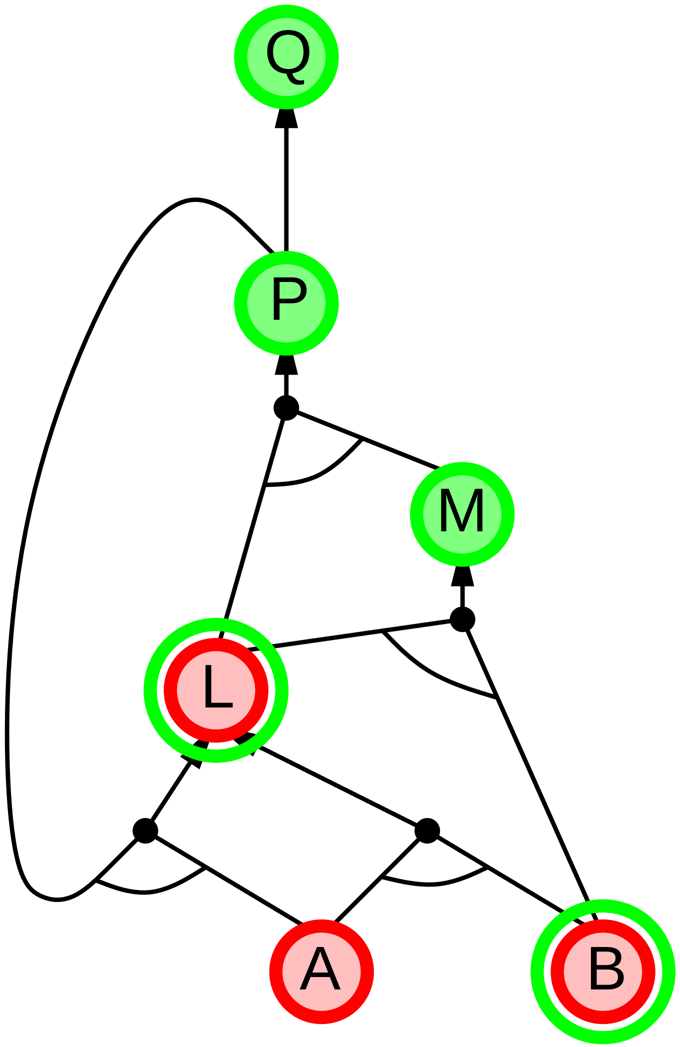
Backward chaining: przyklad



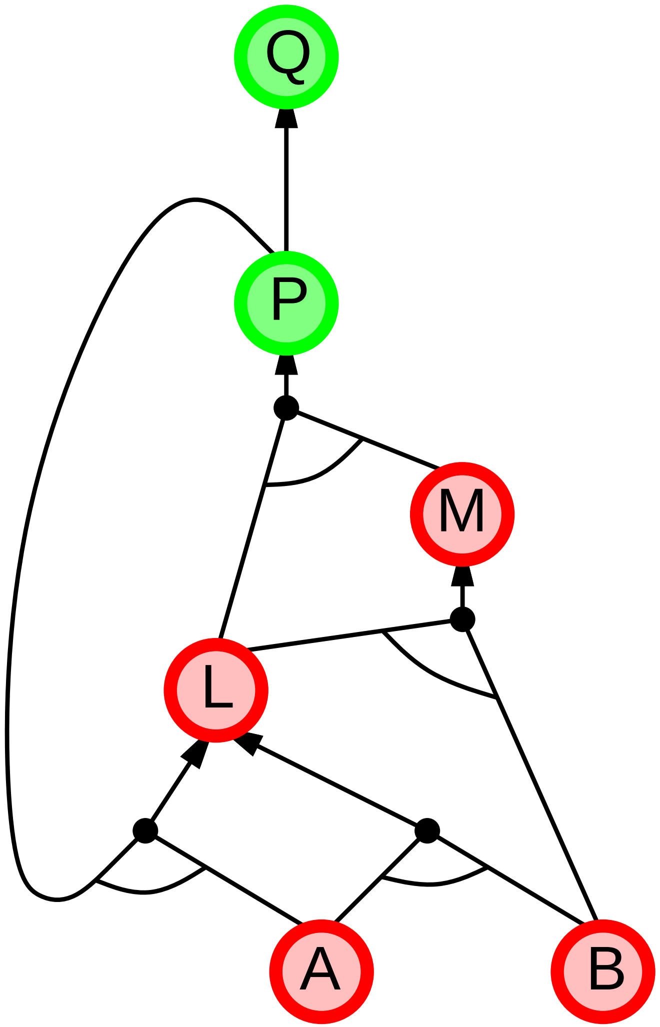
Backward chaining: przyklad



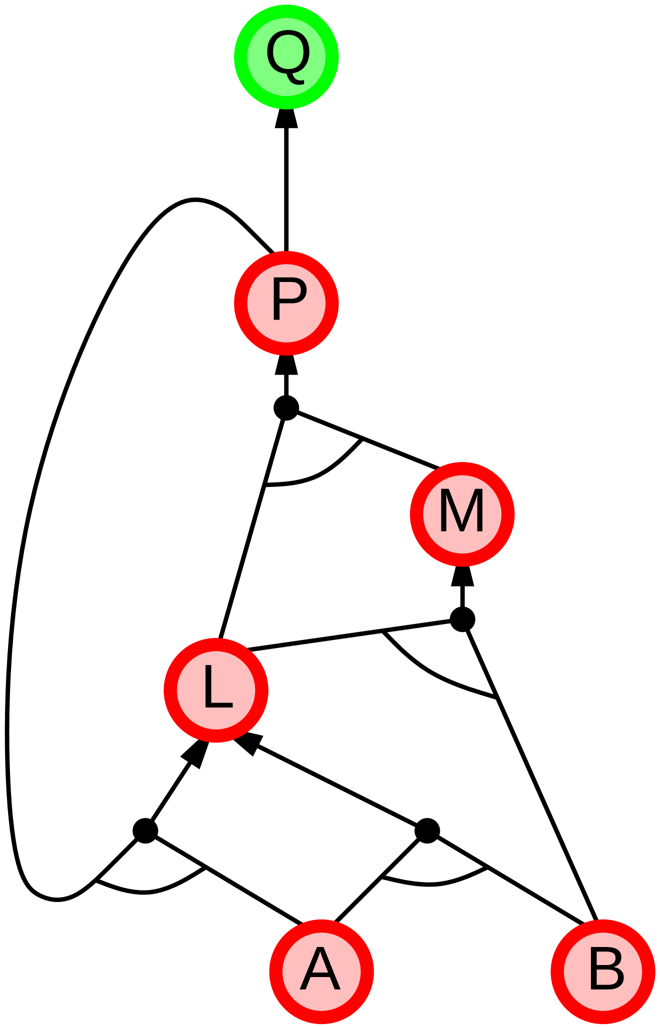
Backward chaining: przyklad



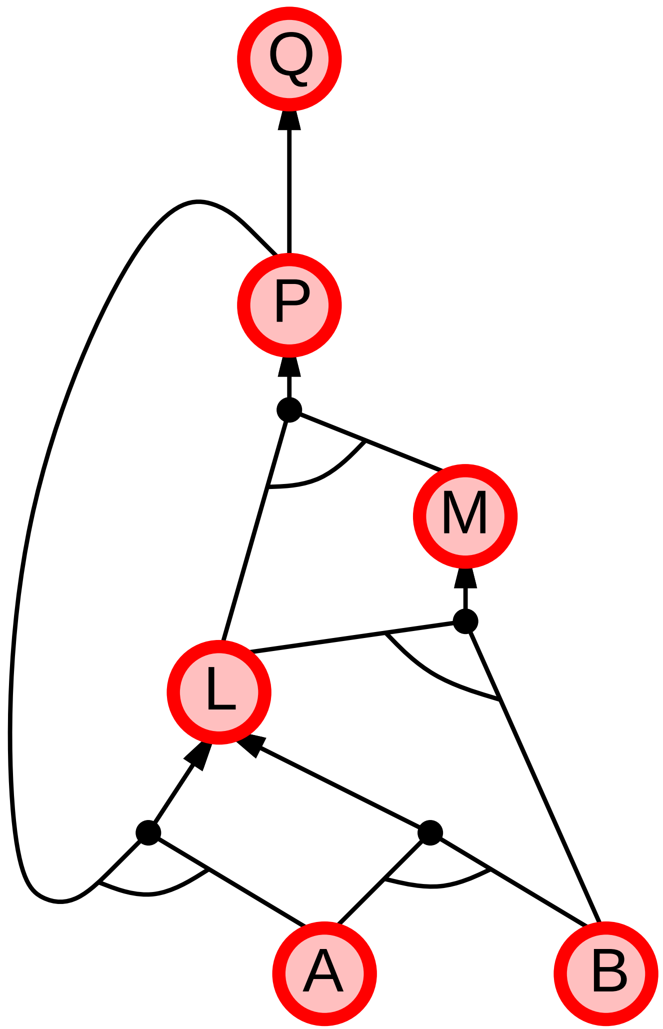
Backward chaining: przyklad



Backward chaining: przyklad



Backward chaining: przyklad



Forward chaining vs. 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

Postać normalna koniunkcyjna (CNF — uniwersalna)

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

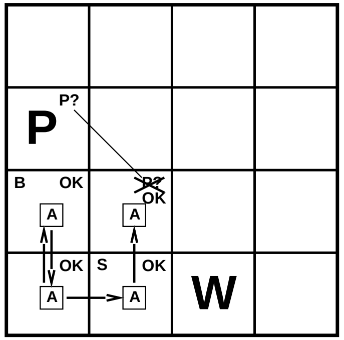
Np. $(A \vee \neg B) \wedge (B \vee \neg C \vee \neg D)$

Rezolucyjna reguła wnioskowania (dla CNF):

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

gdize 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 jest poprawna i pełna dla logiki zdaniowej

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 elim. 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 $\wedge$):

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

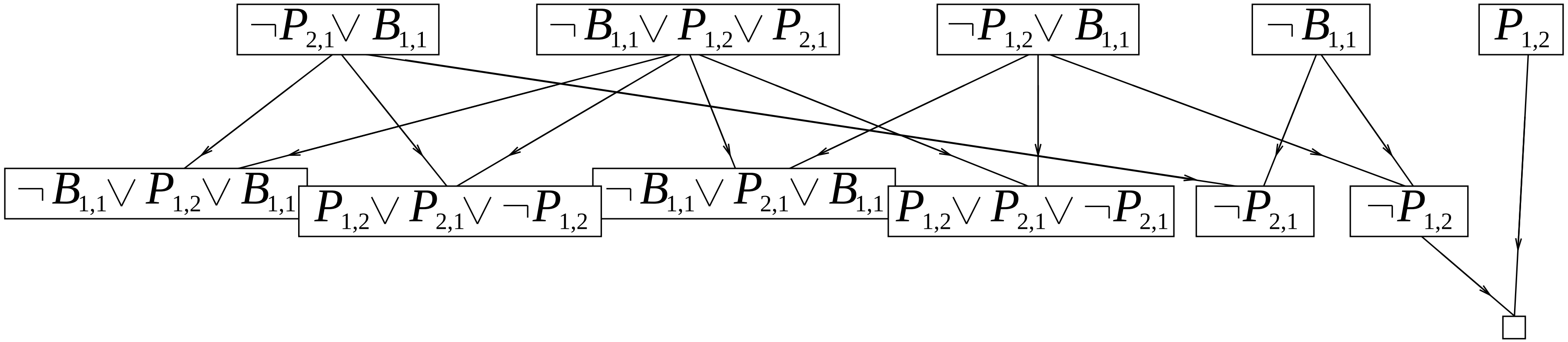
Rezolucja: algorytm

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 do
        new ← { }
        for each Ci, Cj in clauses do
            resolvents ← PL-RESOLVE(Ci, Cj)
            if resolvents contains the empty clause then return true
            new ← new ∪ resolvents
        if new ⊆ clauses then return false
        clauses ← clauses ∪ new
```

Rezolucja: przyklad

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



Procedura Davisa-Putnama

Podobnie jak przy rezolucji:

– Sprowadzenie $KB \wedge \neg\alpha$ do postaci normalnej koniunkcyjnej

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

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)
$sets \leftarrow \{clauses\}$ $\quad newsets \leftarrow \{\,\}$
while *sets* is not empty **do**
  if $\{\,\} \in sets$ **then return** *false*
  $sets \leftarrow \{clauses \in sets$: *clauses* doesn't 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 $\neg l$
    while some l occurs in *clauses* but $\neg l$ doesn't **do** remove clauses containing l
    $p \leftarrow$ 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**
      $clauses(\ldots) \leftarrow \{C_i \in clauses(\ldots) : C_i$ isn't superclause of any $C_j \in clauses(\ldots)\}$
      $newsets \leftarrow newsets \cup \{clauses(\ldots)\}$
  $sets \leftarrow newsets \quad newsets \leftarrow \{\,\}$
return *true*