Agent wnioskujący oparty o logikę

wykład 6

dr inż. Joanna Kołodziejczyk

jkolodziejczyk@wi.ps.pl

Zakład Sztucznej Inteligencji ISZiMM

Plan wykładu

- Agent zawierający wiedzę
- Świat wumpusów „hunt the wumpus”
- Podstawy logiki - modele i interpretacja
- Rachunek zdań
- Równoważność, wiarygodność, spełnialność
- Reguły wnioskowania i dowodzenie twierdzeń
  - wnioskowanie w przód
  - wnioskowanie wstecz
  - rezolucja

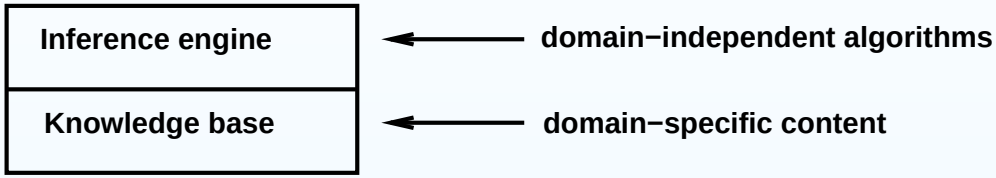
Wymaganie wiedzy w SI

Człowiek ma wiedzę i na jej postawie wykonuje rozumowanie. W SI zawarcie wiedzy w systemie może zapewnić lepsze jego zachowania.

Agent problem-solving posiadał wiedzę o oczekiwanym wyjściu lub dodatkową informację w postaci funkcji heurystycznej. Im więcej wiedzy w systemie tym bardziej złożone zadania można z jego pomocą rozwiązywać.

Agent z wiedzą będzie dobrym rozwiązaniem do nie w pełni obserwowalnego środowiska. Może dokonywać uogólnień doświadczeń z przestrzeni obserwowalnej. np. diagnoza lekarska lub rozumienie języka naturalnego.

Wiedza - podstawy



Baza wiedzy (KB) = zbiór zdań w języku formalnym lub inaczej w języku reprezentacji wiedzy.

Deklaratywne podejście ma zapewnić wykonywanie następujących zadań:

- TELL — możliwość poinformowania systemu o nowej wiedzy (wprowadzanie nowych zdań).
- ASK — odpytywanie systemu co jest mu wiadome (odpowiedź powinna wynikać z KB).

ASK i TELL mogą wymagać wnioskowania czyli wyprowadzania nowych zdań na podstawie starych.

Agent z bazą wiedzy

```
1  function KB-Agent (percept) return an action
2     static: KB a knowledge base
3     t, a counter, initially 0, indicating time
4
5     Tell(KB, Make-Percept-Sentence( percept, t))
6     action = Ask(KB, Make-Action-Query( t))
7     Tell(KB, Make-Action-Sentence( action, t))
8     t = t + 1
9     return action
```

KB — knowledge base na początku musi zawierać informacje zwane wiedzą podstawową.

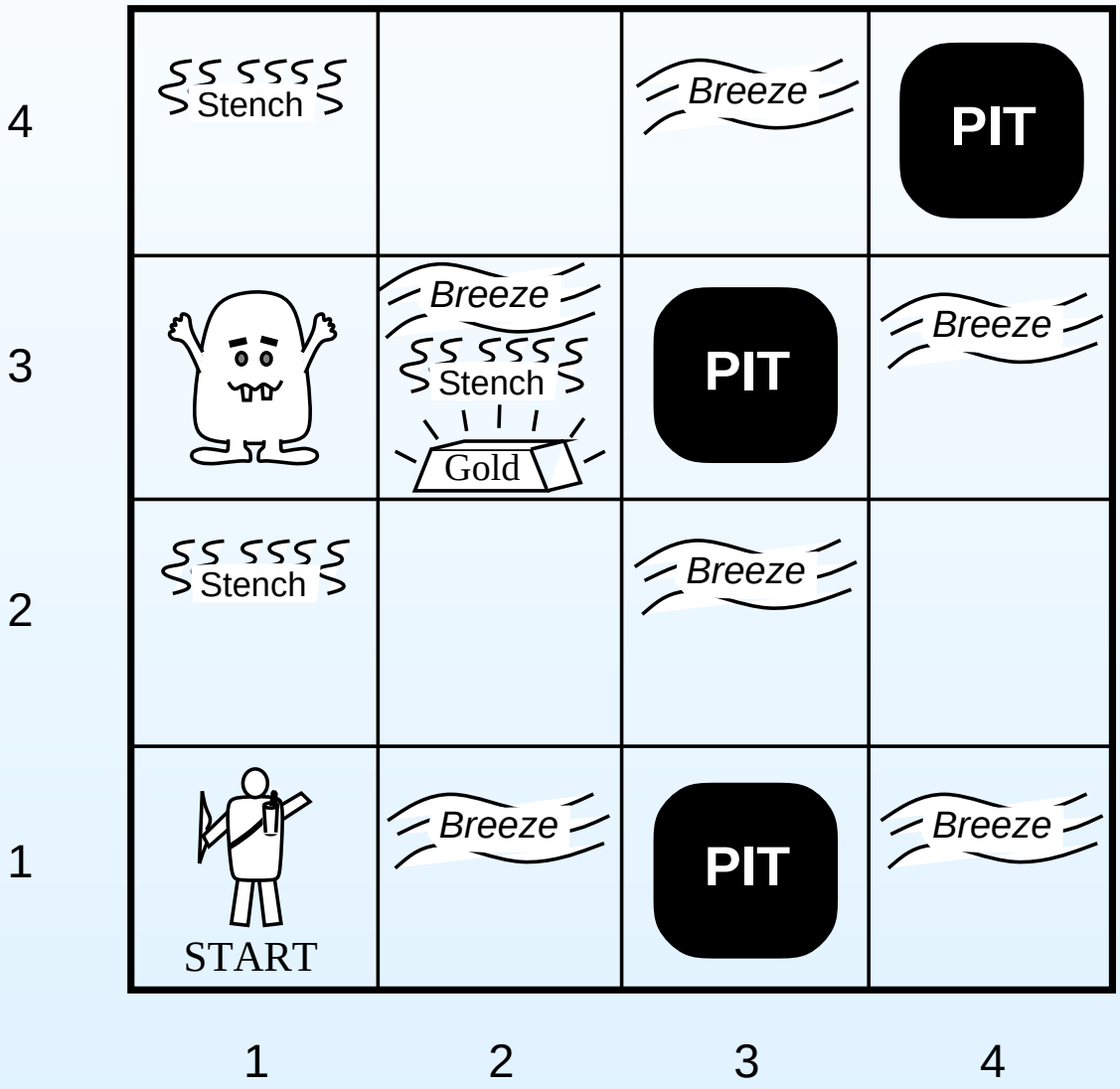
Wszystkie funkcje MAKE- tworzą zdania bądź do wpisania do bazy, bądź jako zapytanie.

Cechy agenta KB

Agenta z bazą wiedzy musi:

- Reprezentować stany i wykonywane akcje w postaci zdań w języku formalnym.
- Włączać do KB nową wiedzą napływającą z otoczającego go świata.
- Odświeżać wewnętrzną reprezentację świata.
- Dedukować ukryte zależności o świecie.
- Wybierać odpowiednie do stanu bieżącego akcje.

Wumpus by Michael Genesereth



Wumpus PEAS

(Performace Enviroment Actuators Sensors)

Miara osiągów	złoto +1000; śmierć -1000; -1 za każdy ruch; -10 za użycie strzały
Środowisko	Tablica 4×4 pomieszczeń. Agent zaczyna w polu $[1, 1]$ twarzą skierowaną w prawo. Położenie wumpusa i złota wybierane jest losowo z pominięciem pola startowego. Każde z pól z prawdopodobieństwem 0.2 może być dołkiem
Aktualizatory	Skręć w lewo, skręć w prawo, idź, chwytaj, upuść, strzelaj (do końca wiersza lub kolumny, lub wumpusa), umiera
Czujniki	Pola przylegające do wumpusa cuchną (stench). W polach przylegających do dołka czuć powiew (breeze). Blask jest w polu gdzie znajduje się złoto (glitter). Jeżeli uderzy się w ścianę to jest to sygnalizowane. Zabicie wumpusa jest oznajmiane jego krzykiem. $[Stench, Breez, None, None, None]$

Charakterystyka otoczenia dla gry wumpus

Obserwowalne: Otoczenie nie jest w pełni widoczne. Możemy mówić tylko o lokalnej widoczności.

Deterministyczne:

Epizodyczne:

Statyczność:

Dyskretne:

Pojedynczy agent:

Charakterystyka otoczenia dla gry wumpus

Obserwowalne: Otoczenie nie jest w pełni widoczne. Możemy mówić tylko o lokalnej widoczności.

Deterministyczne: Tak, kolejny stan jest dokładnie określony.

Epizodyczne:

Statyczność:

Dyskretne:

Pojedynczy agent:

Charakterystyka otoczenia dla gry wumpus

Obserwowalne: Otoczenie nie jest w pełni widoczne. Możemy mówić tylko o lokalnej widoczności.

Deterministyczne: Tak, kolejny stan jest dokładnie określony.

Epizodyczne: Otoczenie jest sekwencyjne — wykonywanie kolejnych akcji.

Statyczność:

Dyskretne:

Pojedynczy agent:

Charakterystyka otoczenia dla gry wumpus

Obserwowalne: Otoczenie nie jest w pełni widoczne. Możemy mówić tylko o lokalnej widoczności.

Deterministyczne: Tak, kolejny stan jest dokładnie określony.

Epizodyczne: Otoczenie jest sekwencyjne — wykonywanie kolejnych akcji.

Statyczność: Tak, wumpus i dołki nie zmieniają położenia.

Dyskretne:

Pojedynczy agent:

Charakterystyka otoczenia dla gry wumpus

Obserwowalne: Otoczenie nie jest w pełni widoczne. Możemy mówić tylko o lokalnej widoczności.

Deterministyczne: Tak, kolejny stan jest dokładnie określony.

Epizodyczne: Otoczenie jest sekwencyjne — wykonywanie kolejnych akcji.

Statyczność: Tak, wumpus i dołki nie zmieniają położenia.

Dyskretne: Tak

Pojedynczy agent:

Charakterystyka otoczenia dla gry wumpus

Obserwowalne: Otoczenie nie jest w pełni widoczne. Możemy mówić tylko o lokalnej widoczności.

Deterministyczne: Tak, kolejny stan jest dokładnie określony.

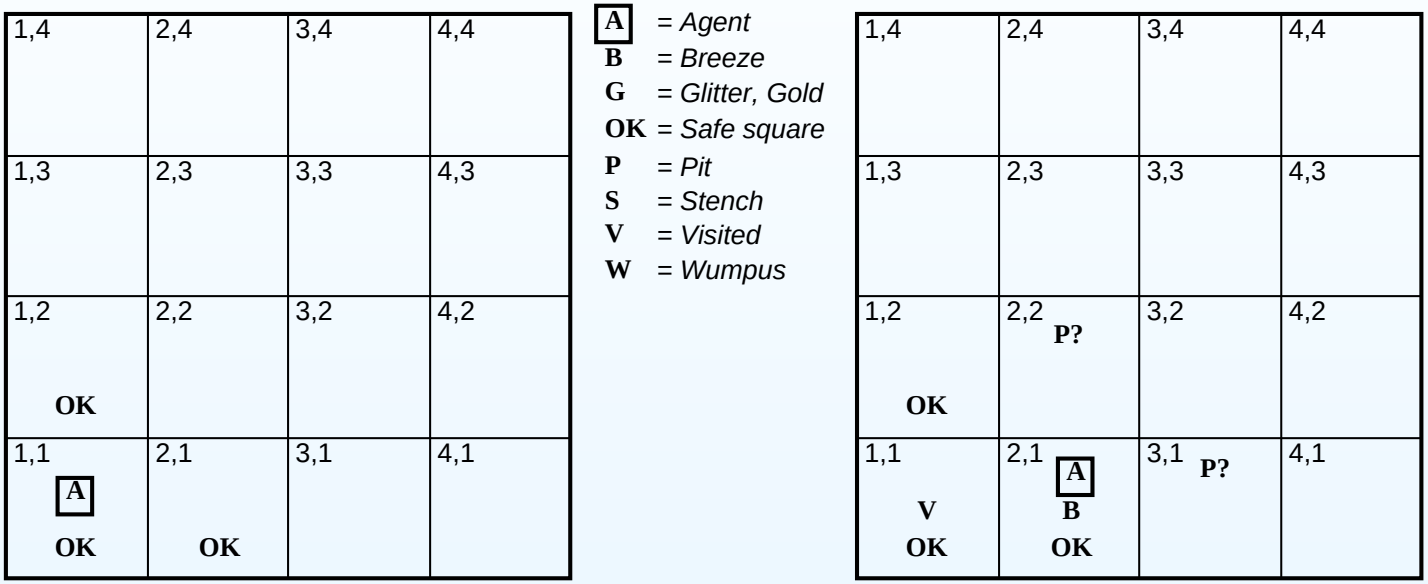
Epizodyczne: Otoczenie jest sekwencyjne — wykonywanie kolejnych akcji.

Statyczność: Tak, wumpus i dołki nie zmieniają położenia.

Dyskretne: Tak

Pojedynczy agent: Zasadniczo wumpus jest tylko częścią otoczenia.

Przemierzanie świata wumpusa



(a) (b)

$[1,1]$ KB początkowo zawiera reguły otoczenia. Pierwszy stan jest bezpieczny i można zapisać jako listę odczytów z sensorów $[None, None, None, None, None]$, przesuń się do bezpiecznego pola albo $[2,1]$ albo $[1,2]$

$[2,1]$ $[None, Breeze, None, None, None]$ oznacza, że musi być dołek w polu $[2,2]$ lub $[3,1]$, wróć zatem do $[1,1]$ by wypróbować kolejne bezpieczne pole.

Przemierzanie świata wumpusa

(a) (b)

$[1,2]$ $[Stench, None, None, None, None]$ oznacza, że wumpus jest albo w polu $[1,3]$ albo $[2,2]$

choć na pewno nie w $[1,1]$

choć nie może też być w $[2,2]$ bo cuchnęłoby w polu $[2,1]$

stąd wumpus jest w polu $[1,3]$

stąd pole $[2,2]$ jest bezpieczne bo w $[1,2]$ nie czuć powiewu

stąd dołek jest w $[1,3]$

przejdź do bezpiecznego pola $[2,2]$

Przemierzanie świata wumpusa

(a) (b)

$[2,2]$ $[None, None, None, None, None]$ czysto — przesuń się do $[2,3]$

$[2,3]$ $[Stench, Breeze, Glitter, None, None]$

stąd wnioskujemy podnieś złoto

stąd wnioskujemy dziura w $[3,3]$ lub $[2,4]$

Fundamentalna zasada wnioskowania w logice

Za każdym razem gdy agent tworzył wnioski z dostępnych informacji zapisanych w bazie wiedzy poprawność wnioskowania była zagwarantowana faktem, iż dostępna informacja była poprawna.

W dalszej części wykładu podane zostanie jak zbudować agenta opartego na logice potrafiącego wyciągać właściwe wnioski.

Logika - podstawy

Logika jest formalnym językiem reprezentowania informacji, z której można wyciągać wnioski (konkluzje).

Język formalny:

- Składnia określa budowę zdań w danym języku. Wnioskowanie musi uwzględniać manipulowanie i generowanie różnych zdań w określonej składni.
- Semantyka określa „znaczenie” wyrażenia. W logice semantyka definiuje prawdziwość każdego zdania w odniesieniu do rozpatrywanej rzeczywistości.

Przykład, język arytmetyki

$x + 2 \geq y$ jest poprawnym wyrażeniem a $x2 + y >$ nie jest poprawnym wyrażeniem
$x + 2 \geq y$ jest prawdą witw, gdy liczba $x + 2$ jest nie mniejsza niż liczba y
$x + 2 \geq y$ jest prawdziwe w takiej rzeczywistości, gdzie $x = 7$ i $y = 1$
$x + 2 \geq y$ jest fałszywe w takiej rzeczywistości, gdzie $x = 0$ i $y = 6$

Model

Model to formalną nazwa tego, co wcześniej nazywane było pewną rzeczywistością.
O ile „pewna rzeczywistość” może być potencjalnie rzeczywistym otoczeniem, w którym agent może lub nie być osadzony, o tyle

Model to pojęcie matematyczne, w którym zachowana jest prawdziwość lub fałsz każdego zdania.

Zdanie „m jest modelem α” należy rozumieć, że zdanie α jest prawdziwe w modelu m.

Nieformalnie, możemy myśleć o x i y jako liczbie kobiet i mężczyzn grających w brydża i prawdziwe będzie zdanie $x + y = 4$, bo liczba graczy jest 4.

Formalnie, wszystkie możliwe modele, to wszystkie możliwe przypisania wartości dla x i y o ile takie przypisanie zachowuje prawdziwość zdania wyrażonego językiem arytmetyki.

Logiczna konsekwencja

Logiczna konsekwencja — wynikanie dwóch zdań to oznacza, że jedno zdanie wynika logicznie z innego zdania. Co zapisujemy

$$\alpha \models \beta$$

i czytamy, że α pociąga za sobą β.

Definicja
$\alpha \models \beta$ wtedy i tylko wtedy, gdy w każdym modelu, w którym α jest prawdziwe β jest również prawdziwe.
Inaczej można powiedzieć jeżeli α jest prawdziwe, to β musi być prawdziwe.

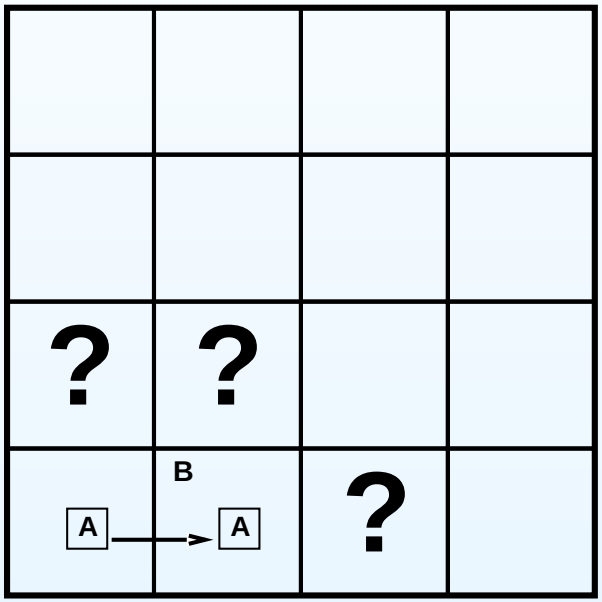
np., $x + y = 4$ pociąga za sobą $4 = x + y$ bo wkażdym modelu, w którym prawdziwe jest zdanie lewe prawdziwe jest również prawe.

Konsekwencja logiczna jest związkiem pomiędzy zdaniami,

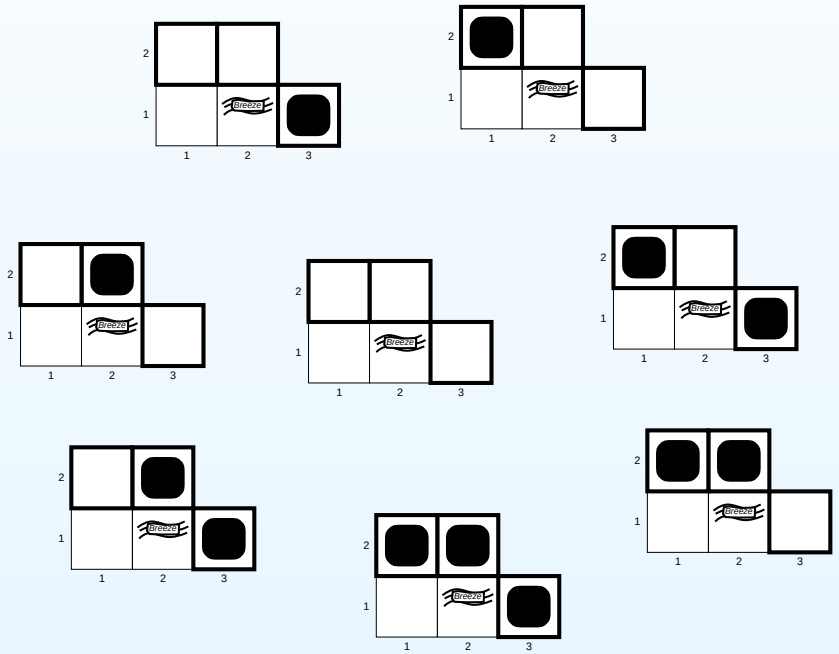
Analiza wnioskowania w świecie wumpusa

KB to: reguły zapisane w PEAS oraz wiedza, że pole [1,1] jest czyste, a po przesunięciu w prawo, stwierdzono powiew w polu [2,1].

Agent chce wiedzieć, czy pola oznaczone „?” zawierają dołki. Każde z trzech pól może zawierać dołek lub nie. Zatem istnieje $2^3 = 8$ możliwych modeli.

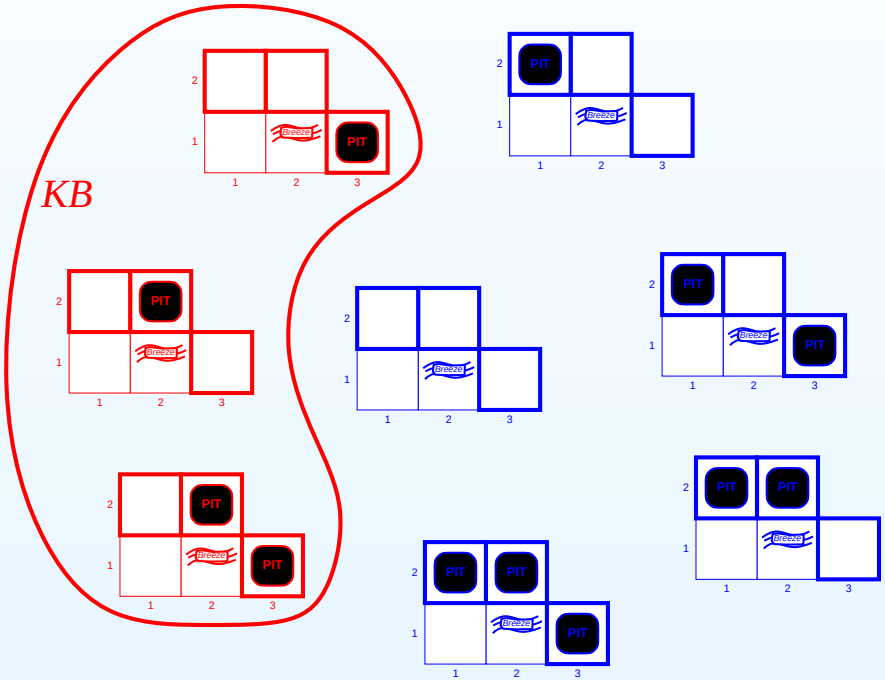


Modele dla świata wumpusa



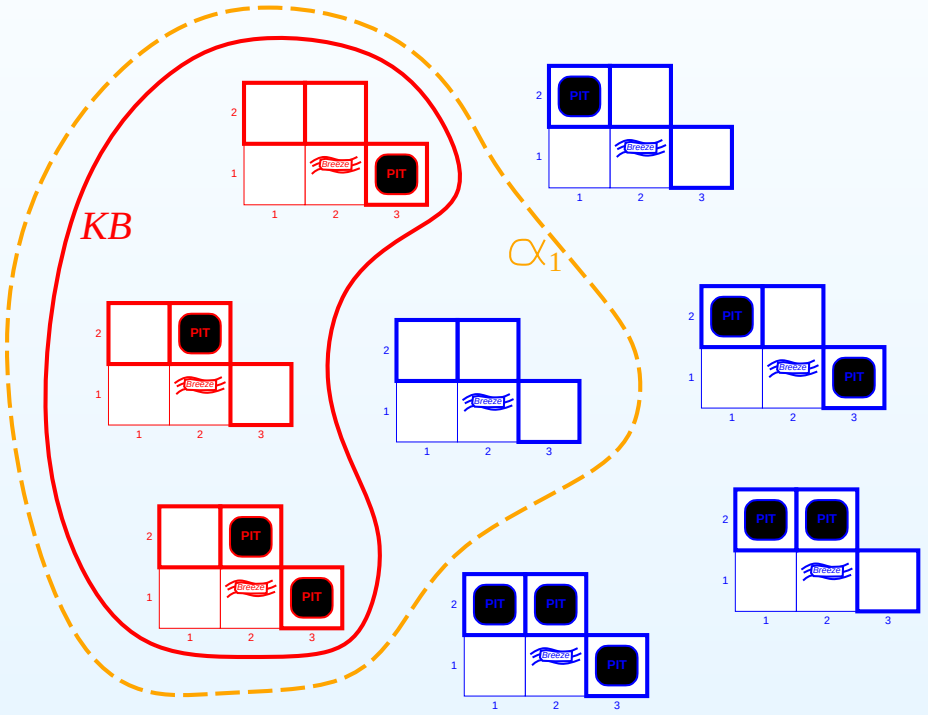
Wszystkie możliwe modele bez wiedzy.

Modele dla świata wumpusa



KB = reguły świata wumpusa. Tylko w trzech modelach KB jest prawdziwe.

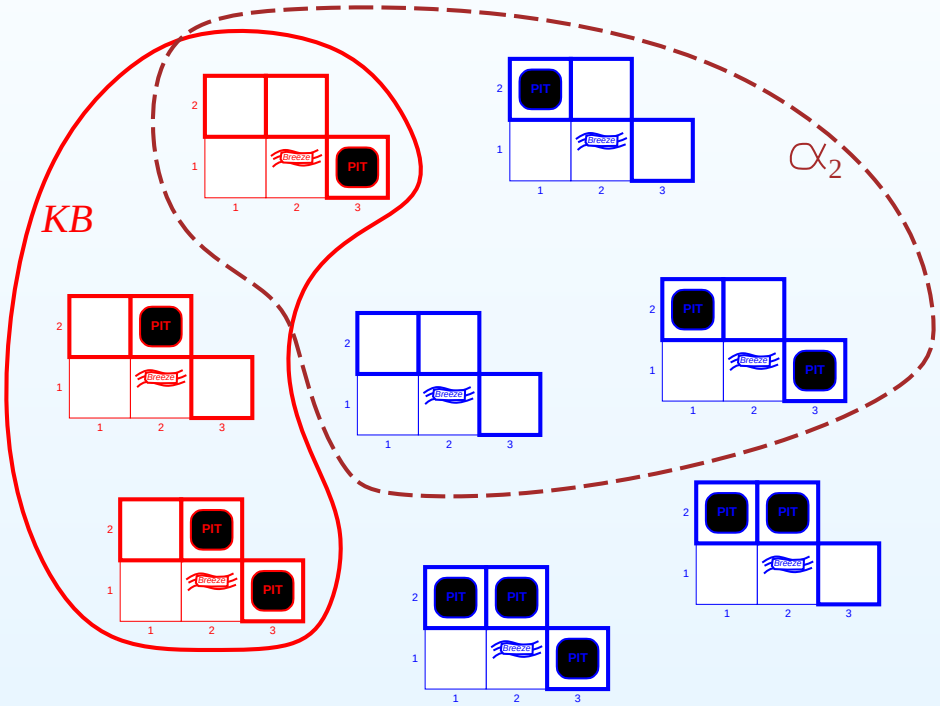
Modele dla świata wumpusa



KB = reguły świata wumpusa. Tylko w trzech modelach KB jest prawdziwe.

Konkluzja α_1 = „$[1,2]$ bezpieczne”. Dla każdego modelu, dla którego KB jest prawdziwe α_1 też jest prawdziwe. Stąd $KB \models \alpha_1$: $[1,2]$ bezpieczne.

Modele dla świata wumpusa



KB = reguły świata wumpusa. Tylko w trzech modelach KB jest prawdziwe.

Konkluzja α_2 = „$[2,2]$ bezpieczne”.Dla niektórych modeli, dla których KB jest TRUE, α_2 jest FALSE. Stąd $KB \not\models \alpha_2$. Zatem nie można wywnioskować, czy w $[2.2]$ jest bezpiecznie, ani też czy jest dołek.

Wnioskowanie w logice

Przykład pokazał jak zastosować konsekwencję, by uzyskać konkluzję, czyli jak przeprowadzić wnioskowanie. Przedstawiony algorytm postępowania jest nazywany sprawdzaniem modelu, bo sprawdza, czy we wszystkich modelach, które zachowują KB α też jest zachowane (TRUE).

Zbiór wszystkich konsekwencji z KB można potraktować jako stóg siana. Konsekwencja α jako igłę.

Zatem jeżeli logiczna konsekwencja jest igłą w stogu siana, to wnioskowanie jest procedurą odszukania igły.

Algorytm wnioskowania w logice

i oznacza algorytm wyprowadzający α z KB, co zapisujemy:

$$KB \vdash_i \alpha$$

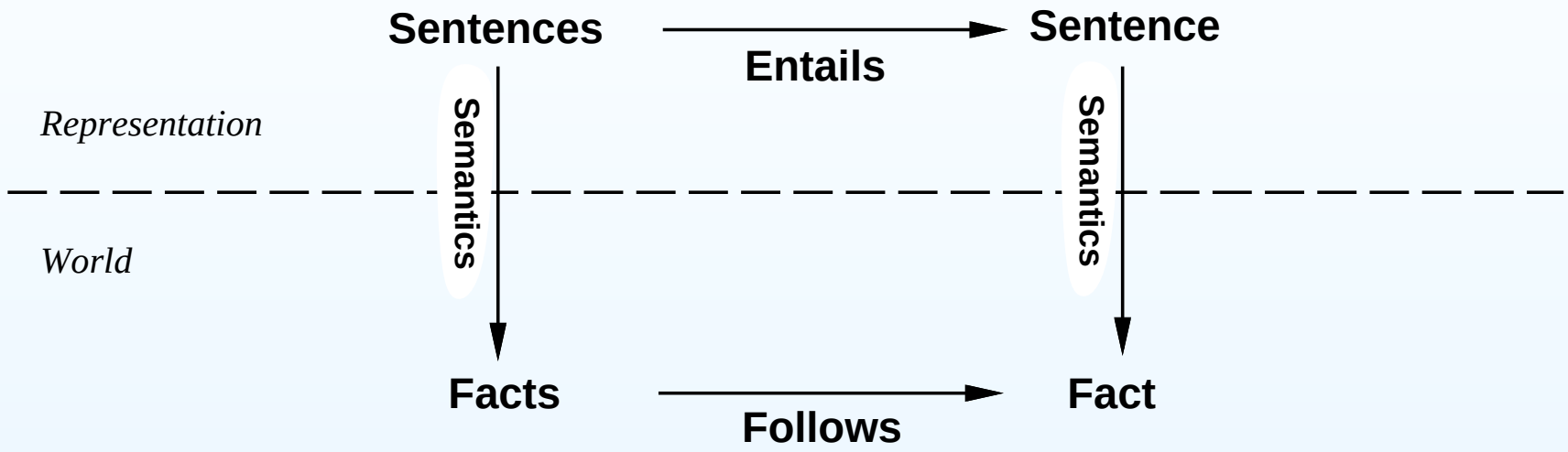
i czytamy: α jest wnioskowane z KB przez algorytm i.

Poprawność algorytmu (algorytm zachowujący prawdę) (Soundness): i jest poprawnym algorytmem wnioskowania, jeżeli dla każdego $KB \vdash_i \alpha$ prawdziwe jest też, że $KB \models \alpha$. Algorytm, który jest „unsound” będzie znajdował nieistniejące igły.

Zupełność algorytmu (Completeness): i jest zupełnym algorytmem wnioskowania, jeżeli dla każdego $KB \models \alpha$ prawdziwe jest też, że $KB \vdash_i \alpha$. Innymi słowy może wywnioskować każde zdanie, które jest logiczną konsekwencją.

Jeżeli przesłanki (KB) są prawdziwe w świecie rzeczywistym, to każde zdanie (formuła) α wywnioskowane z KB przez poprawny algorytm wnioskowania jest też prawdziwe w świecie rzeczywistym.

Zależność pomiędzy reprezentacją i rzeczywistością



Zdania są fizyczną konfiguracją agenta, a wnioskowanie jest procesem kostruowania nowych konfiguracji. Wnioskowanie w logice powinno zapewnić, że nowe konfiguracje reprezentują aspekty rzeczywistości, które pochodzą od starego stanu (konfiguracji).

Rachunek zdań—składnia (syntaksa)

Rachunek zdań jest najprostszą z logik.

Składnia definiuje dopuszczalne zdania (formuły).

Symbole takie jak np. P_1, Q, $R_{1,1}$ itp. są zdaniami atomowymi (formułami).

Każda formuła może być TRUE lub FALSE.

Istnieje pięć różnych operatorów, za pomocą których można tworzyć złożone formuły.

1. Negacja—Jeżeli S (positive literal) jest formułą $\neg S$ (negative literal) jest formułą.
2. Koniunkcja—Jeżeli S_1 i S_2 to formuły, $S_1 \wedge S_2$ jest formułą.
3. Dysjunkcja—Jeżeli S_1 i S_2 to formuły, $S_1 \vee S_2$ jest formułą.
4. Implikacja (warunek)—Jeżeli S_1 i S_2 to formuły, $S_1 \rightarrow S_2$ jest formułą. Implikacja znana jest też jako reguła czyli zdania typu IF-THEN
5. Równoważność—Jeżeli S_1 i S_2 to formuły, $S_1 \leftrightarrow S_2$ jest formułą. Czytane: „wtedy i tylko wtedy”.

Rachunek zdań—semantyka

Semantyka określa zasady uzyskiwania prawdy dla zdania zgodnie z pewnym modelem. Model w rachunku zdań ustala wartość TRUE lub FALSE dla każdego symbolu.

np. model m_1

$P_{1,2}$	$P_{2,2}$	$P_{3,1}$
$FALSE$	$FALSE$	$TRUE$

(Dla podanych symboli, 8 możliwych modeli można podać „od ręki”.)

Nie znamy znaczenia symbolu $P_{1,2}$. Może to być np. zdanie „W polu $[1, 2]$ jest dołek”.

Rachunek zdań—semantyka

Semantyka ma określić jak obliczyć wartość TRUE dla każdego zdania, gdy dany jest model. Wykonuje się tę procedurę rekurencyjnie. Wszystkie zdania zbudowane są z formuł atomowych i pięciu operatorów.
Reguły:

- TRUE jest TRUE w każdym modelu, a FLASE jest FALSE w każdym modelu.
- Wartość TRUE symbolu (formuły atomowej) wynika z modelu.
- Dla zdań złożonych w modelu m

$\neg S$	TRUE w m	witw	S	FALSE w m			
$S_1 \wedge S_2$	TRUE w m	witw	S_1	TRUE w m *and*	S_2	TRUE w m
$S_1 \vee S_2$	TRUE w m	witw	S_1	TRUE w m *or*	S_2	TRUE w m
$S_1 \rightarrow S_2$	TRUE w m	witw	S_1	FALSE w m *or*	S_2	TRUE w m
jw.	FALSE w m	witw	S_1	TRUE w m *and*	S_2	FALSE w m
$S_1 \leftrightarrow S_2$	TRUE w m	witw	$S_1 \rightarrow S_2$	TRUE w m *and*	$S_2 \rightarrow S_1$	TRUE w m

Tablica prawdy dla operatorów

P	Q	$\neg P$	$P \wedge Q$	$P \vee Q$	$P \rightarrow Q$	$P \leftrightarrow Q$
$false$	$false$	$true$	$false$	$false$	$true$	$true$
$false$	$true$	$true$	$false$	$true$	$true$	$false$
$true$	$false$	$false$	$false$	$true$	$false$	$false$
$true$	$true$	$false$	$true$	$true$	$true$	$true$

Prosty proces rekurencyjny może ocenić dowolne zdanie np. dla jednego z modeli świata wumpusa:,
$\neg P_{1,2} \wedge (P_{2,2} \vee P_{3,1})$ = $true \wedge (false \vee true) = true \wedge true = true$

Świat wumpusa—prosta KB

KB jest koniunkcją zdań w niej zawartych: $KB = S_1 \wedge \ldots \wedge S_n$.

Niech $P_{i,j}$ jest $true$, jeżeli jest dołek w $[i, j]$.
Niech $B_{i,j}$ jest $true$, jeżeli jest powiew w $[i, j]$.

KB zawiera reguły:

- Nie ma dołka w $[1, 1]$:
  $R_1 : \neg P_{1,1}$.
- Pole ma podmuch *witw*, gdy sąsiaduje z dołkiem. (Tymczasem dla każdego dołka).
  $R_2 : B_{1,1} \leftrightarrow (P_{1,2} \vee P_{2,1})$.
  $R_3 : B_{2,1} \leftrightarrow (P_{1,1} \vee P_{2,2} \vee P_{3,1})$.
- Poprzednie zdania są zawsze prawdziwe w rzeczywistości wumpusa. Dodajemy odczyt z sensorów z dwóch odwiedzonych pól $[1, 1]$ i $[2, 1]$
  $R_4 : \neg B_{1,1}$.
  $R_5 : B_{2,1}$.

$KB = R_1 \wedge R_2 \wedge R_3 \wedge R_4 \wedge R_5$

Wnioskowanie

Celem wnioskowania w logice jest ustalenie czy $KB \models \alpha$ dla pewnego zdania α.

Przedstawiony poniżej pierwszy algorytm wnioskowania jest bezpośrednią implementacją definicji:

1. Utwórz wszystkie modele (przypisując TRUE i FALSE dla każdego symbolu występującego w KB)
2. Sprawdź czy α jest TRUE w każdym modelu, w którym KB jest TRUE.

Tablica prawdy — wszystkie modele

$B_{1,1}$	$B_{2,1}$	$P_{1,1}$	$P_{1,2}$	$P_{2,1}$	$P_{2,2}$	$P_{3,1}$	R_1	R_2	R_3	R_4	R_5	KB
f	f	f	f	f	f	f	t	t	t	t	f	f
f	f	f	f	f	f	t	t	t	f	t	f	f
$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$
f	t	f	f	f	f	f	t	t	f	t	t	f
f	t	f	f	f	f	t	t	t	t	t	t	t
f	t	f	f	f	t	f	t	t	t	t	t	t
f	t	f	f	f	t	t	t	t	t	t	t	t
f	t	f	f	t	f	f	t	f	f	t	t	f
$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$
t	t	t	t	t	t	t	f	t	t	f	t	f

$KB = R_1 \wedge R_2 \wedge R_3 \wedge R_4 \wedge R_5 = TRUE$ tylko w 3 przypadkach na $2^7 = 128$.

Wnioskowanie przez wartościowanie

Wartościowanie wszystkich modeli metodą w głąb jest poprawne i kompletne. $O(2^n)$ dla n symboli; problem jest *co-NP-trudny*.

```
1  function TT-Entails?(KB, alpha) return true or false
2     inputs: KB- baza wiedzy, zdania w notacji rachunku zdań
3           alpha - zapytanie, zdanie w notacji rachunku zdań
4
5     symbols = lista symboli zdań z KB i alpha
6     return TT-Check-All(KB, alpha, symbols, [])
7
8  function TT-Check-All(KB, alpha, symbols, model)
9                           return true or false
10    if Empty(symbols) then
11        if PL-True(KB, model) then return PL-True(alpha, model)
12        else return true
13    else do
14        P = First(symbols); rest = Rest(symbols)
15        return TT-Check-All(KB, alpha, rest, Extend(P, true, model))
16            and
17             TT-Check-All(KB, alpha, rest, Extend(P, false, model))
```

Logiczna równoważność

Dwa zdania są równoważne, jeżeli są TRUE w tym samym zbiorze modeli, co zapisujemy $\alpha \equiv \beta$ witw $\alpha \models \beta$ i $\beta \models \alpha$

$$
\begin{aligned}
(\alpha \wedge \beta) &\equiv (\beta \wedge \alpha) \quad przemiennosc \wedge \\
(\alpha \vee \beta) &\equiv (\beta \vee \alpha) \quad przemiennosc \vee \\
((\alpha \wedge \beta) \wedge \gamma) &\equiv (\alpha \wedge (\beta \wedge \gamma)) \quad lacznosc \wedge \\
((\alpha \vee \beta) \vee \gamma) &\equiv (\alpha \vee (\beta \vee \gamma)) \quad lacznosc \vee \\
\neg(\neg\alpha) &\equiv \alpha \quad eliminacja\ podwojnej\ negacji \\
(\alpha \rightarrow \beta) &\equiv (\neg\beta \rightarrow \neg\alpha) \\
(\alpha \rightarrow \beta) &\equiv (\neg\alpha \vee \beta) \quad eliminacja\ implikacji \\
(\alpha \equiv \beta) &\equiv ((\alpha \rightarrow \beta) \wedge (\beta \rightarrow \alpha)) \quad eliminacja\ rownowaznosci \\
\neg(\alpha \wedge \beta) &\equiv (\neg\alpha \vee \neg\beta) \quad De\ Morgan \\
\neg(\alpha \vee \beta) &\equiv (\neg\alpha \wedge \neg\beta) \quad De\ Morgan \\
(\alpha \wedge (\beta \vee \gamma)) &\equiv ((\alpha \wedge \beta) \vee (\alpha \wedge \gamma)) \quad rozdzielnosc \wedge\ wzgledem \vee \\
(\alpha \vee (\beta \wedge \gamma)) &\equiv ((\alpha \vee \beta) \wedge (\alpha \vee \gamma)) \quad rozdzielnosc \vee\ wzgledem \wedge
\end{aligned}
$$

Prawdziwość i spełnialność

Zdanie (formuła) jest prawdziwa lub inaczej jest tautologią, jeżeli jest TRUE we *wszystkich modelach*.
np. $True$, $A \vee \neg A$, $A \rightarrow A$, $(A \wedge (A \rightarrow B)) \rightarrow B$

Twierdzenie o dowodzeniu dedukcją:
Dla każdej formuły α i β:
$\alpha \models \beta$ wtedy i tylko wtedy, gdy $(\alpha \rightarrow \beta)$ jest tautologią.

Zdanie (formuła) jest spełnialne, jeżeli jest TRUE dla *niektórych modeli*.
np. $KB = R_1 \wedge R_2 \wedge R_3 \wedge R_4 \wedge R_5$ jest spełnialne tylko w 3 modelach. Jeżeli zdanie α jest TRUE w modelu m, to mówi się, że m spełnia α lub m jest modelem α.

Zdanie (formuła) jest niespełnialna jeżeli w *żadnym modelu* nie jest TRUE.
np. $A \wedge \neg A$

Twierdzenie o dowodzeniu reductio ad absurdum.
$\alpha \models \beta$ wtedy i tylko wtedy zdanie $(\alpha \wedge \neg \beta)$ jest niespełnialne
Inaczej nazywa się tą metodę przez obalanie lub sprzeczność.

Reguły (wzorce) wnioskowania

- Modus Ponens

$$\frac{\alpha \rightarrow \beta, \qquad \alpha}{\beta}$$

  Jeżeli dane jest α i $\alpha \rightarrow \beta$, to możemy wnioskować β.

- And-Elimination—z iloczynu można wnioskować każdy czynnik

$$\frac{\alpha \wedge \beta}{\alpha}$$

- Równoważność

$$\frac{\alpha \leftrightarrow \beta}{(\alpha \rightarrow \beta) \wedge (\beta \rightarrow \alpha)} \qquad and \qquad \frac{(\alpha \rightarrow \beta) \wedge (\beta \rightarrow \alpha)}{\alpha \leftrightarrow \beta}$$

Reguły są poprawne (sound).

Rezolucja

Rezolucja jest regułą wnioskowania, która jest poprawna (sound) i będzie zupełna, jeżeli zastosujemy w niej dowolny algorytm przeszukiwania, który jest zupełny, np. iteracyjne zagłębianie.

Reguła rezolucji

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

gdzie ℓ_i and m_j są literałami komplementarnymi ($m_j = \neg \ell_i$)
np.,

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

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

Koniunkcyjna postać normalna (CNF)

Regułę rezolucji stosuje się tylko dla dysjunkcji literałów. Zatem konieczne będzie posiadanie bazy wiedzy w postaci takich właśnie sum logicznych.

Każde zdanie w rachunku zdań jest logicznym równoważnikiem koniunkcji dysjunkcji literałów.

Koniunkcyjna forma normalna (Conjunctive Normal Form) (CNF) to koniunkcja dysjunkcji literałów.
np., $(A \vee \neg B) \wedge (B \vee \neg C \vee \neg D)$

Klauzula to suma literałów. Powyższy przykład zawiera dwie klauzule.

Konwersja do koniunkcyjnej postaci normalnej

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

1. Eliminacja $\leftrightarrow$, zamień $\alpha \leftrightarrow \beta$ na $(\alpha \rightarrow \beta) \wedge (\beta \rightarrow \alpha)$.

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

2. Eliminacja $\rightarrow$, zamień $\alpha \rightarrow \beta$ na $\neg\alpha \vee \beta$.

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

3. Przesuń $\neg$ do nawiasów stosując prawa de Morgana i podwójną negacje:

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

4. Zastosuj prawa rozdzielności $\vee$ względem $\wedge$:

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

Algorytm rezolucji

Dowodzenia przez sprzeczność (reducio ad absurdum) czyli, aby wykazać $KB \models \alpha$ wykazane zostanie, że $KB \wedge \neg\alpha$ jest niespełnialne. Wykonuje się to przez wykazanie sprzeczności.

```
1  function PL-Resolution(KB,alpha) return true or false
2     inputs: KB---baza wiedzy formuły w rachunku zdań
3           alpha---zapytanie, formuła w rachunku zdań
4
5     clauses = zbiór klauzul w postaci CNF w tym KB i ~alpha
6     new = {}
7     loop do
8         for each Ci, Cj in clauses do
9             resolvents = PL-Resolve(Ci,Cj)
10            if resolvents contains the empty clause then return true
11            new = new sum resolvents
12        if new subset clauses then return false
13        clauses = clauses sum new
```

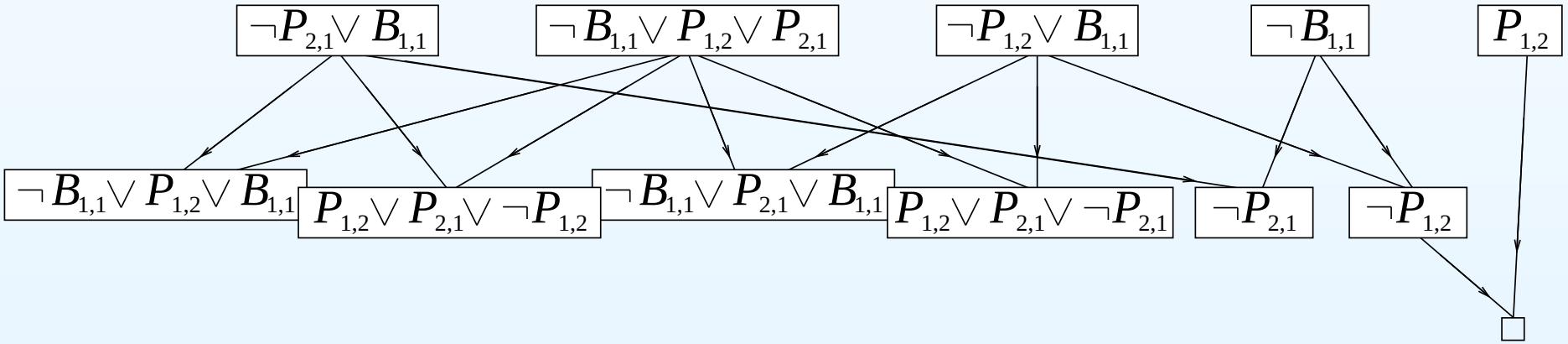
Procedura PL-Resolve daje w wyniku zbiór wszystkich możliwych rezolwent z zadanej pary wejściowej.

Przykład dowodzenia z użyciem rezolucji

$KB = R_2 \wedge R_4 = (B_{1,1} \leftrightarrow (P_{1,2} \vee P_{2,1})) \wedge \neg B_{1,1}$

Dowodzimy:
$\alpha = \neg P_{1,2}$

Po konwersji $KB \wedge \neg\alpha$ na CNF otrzymujemy pierwszy wiersz na schemacie.



Drugi wiersz powstaje jako rezolwenty z połączeń wszystkich klauzul z pierwszego wiersza. Ostatecznie dwie klauzule zostają połączone prowadząc do klauzuli pustej (sprzeczności) oznaczonej jako mały kwadrat. Zatem dowiedliśmy, że α.

Dowodzenie w przód i wstecz

Wnioskowanie w kierunku celu (forward chaining)—dowodzenie w przód.
Wnioskowanie w kierunku przesłanek (backward chaining)—dowodzenie wstecz.

Algorytmy zostaną przedstawione dla zdań w postaci klauzul Horna.

Klauzula Horna jest to dysjunkcja literałów, z których co najwyżej jeden jest pozytywny (niezanegowany) np. $(B \vee \neg A \vee \neg G)$.
Inaczej można podać, że klauzle Horna to $(S_1 \wedge \ldots \wedge S_n) \rightarrow R$

Zatem baza wiedzy, to KB = *koniunkcja klauzul Horna*
np. $C \wedge (B \rightarrow A) \wedge (C \wedge D \rightarrow B)$
(lub klauzul Horna w postaci normalnej)

Zalety stosowania klauzul Horna

1. Łatwa interpretowalność w zapisie z koniunkcją (definite clauses).
2. Wnioskowanie z klauzul Horna może być wnioskowaniem w przód jak i wstecz.
3. Potwierdzenie logicznej konsekwencji w klauzulach Horna ma liniową złożoność czasową od wielkości KB.

Alforytm wnioskowania w przód

```
1  function PL-FC-Entails(KB, q) return true or false
2    inputs: KB - baza wiedzy, zbiór klauzul Horna
3          q zapytanie (formuła)
4    local variables:
5        count - tablica indeksowana klauzulami, początkowo liczba przesłanek
6        inferred - tablica indeksowana literałami początkowo
7        wszystkie pola false
8        agenda - lista literałów, początkowo literały z KB
9
10   while agenda is not empty do
11       p = Pop(agenda)
12       if p = q then return true
13       unless inferred[p] do
14            inferred[p] = true
15            for each Horn clause c in whose premise p appears do
16                decrement count[c]
17                if count[c] = 0 then do
18                    Push(Head[c],agenda)
19    return false
```

Wnioskowanie w przód (kierunku celu)

Idea: odpal dowolną regułę, której przesłanki są spełnialne w KB i dodaj konkluzje do KB. Wykonuj dopóki nie znajdziesz $zapytania$.

Przykład

$P \rightarrow Q$

$L \wedge M \rightarrow P$

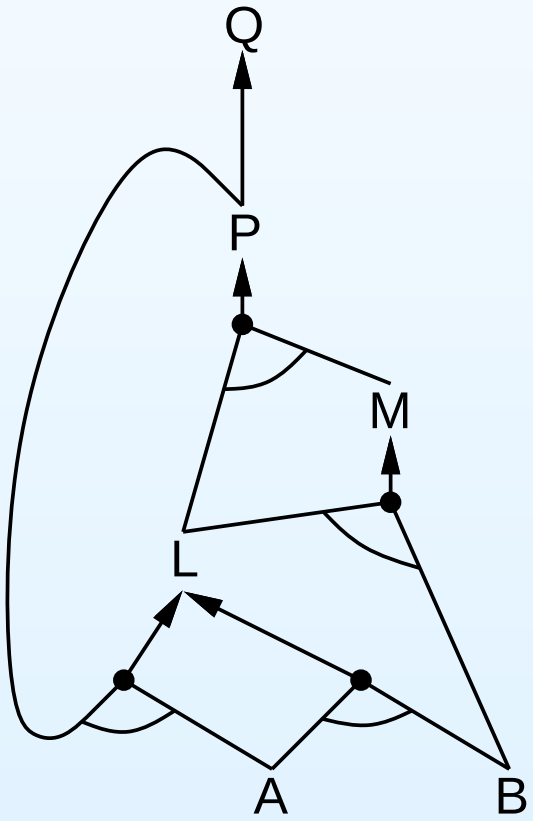
$B \wedge L \rightarrow M$

$A \wedge P \rightarrow L$

$A \wedge B \rightarrow L$

A

B



Przykład wnioskowania w przód

Dowiedź, że Q.

$P \rightarrow Q$

$L \wedge M \rightarrow P$

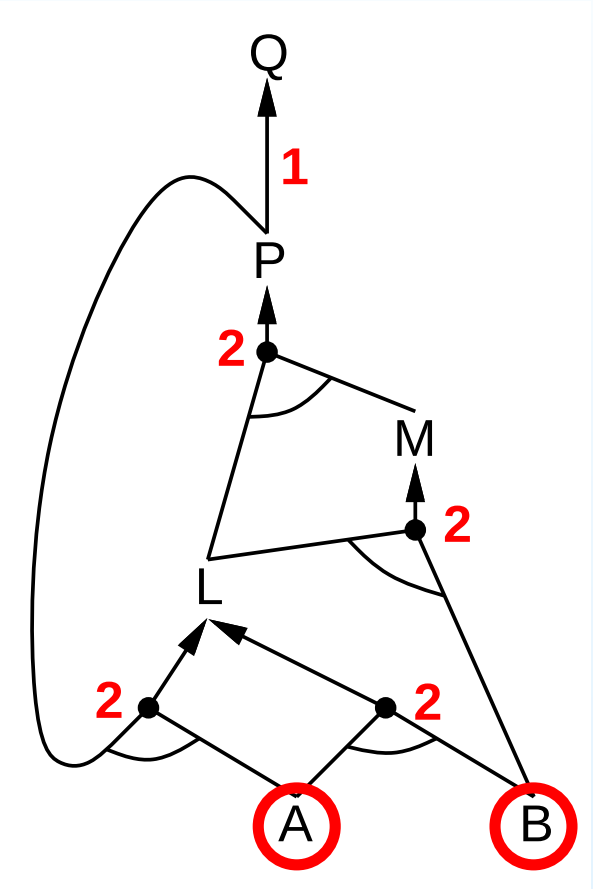
$B \wedge L \rightarrow M$

$A \wedge P \rightarrow L$

$A \wedge B \rightarrow L$

A

B



Przykład wnioskowania w przód

Dowiedź, że Q.

$P \rightarrow Q$

$L \wedge M \rightarrow P$

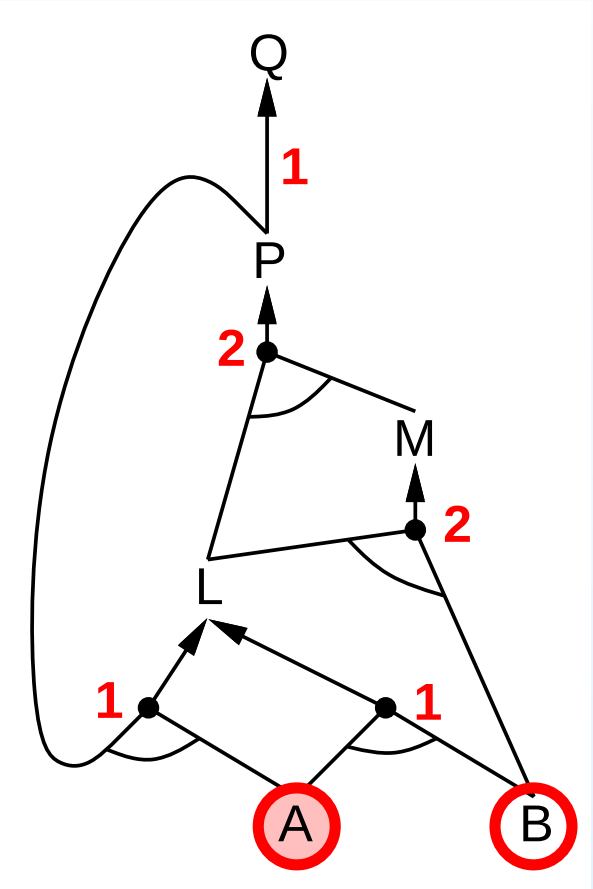
$B \wedge L \rightarrow M$

$A \wedge P \rightarrow L$

$A \wedge B \rightarrow L$

A

B



Przykład wnioskowania w przód

Dowiedź, że Q.

$P \rightarrow Q$

$L \wedge M \rightarrow P$

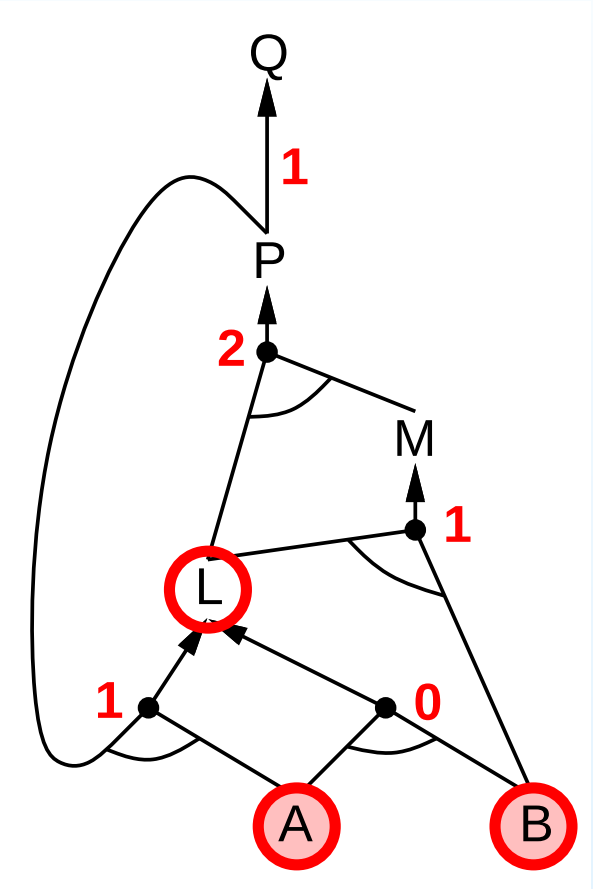
$B \wedge L \rightarrow M$

$A \wedge P \rightarrow L$

$A \wedge B \rightarrow L$

A

B



Przykład wnioskowania w przód

Dowiedź, że Q.

$P \rightarrow Q$

$L \wedge M \rightarrow P$

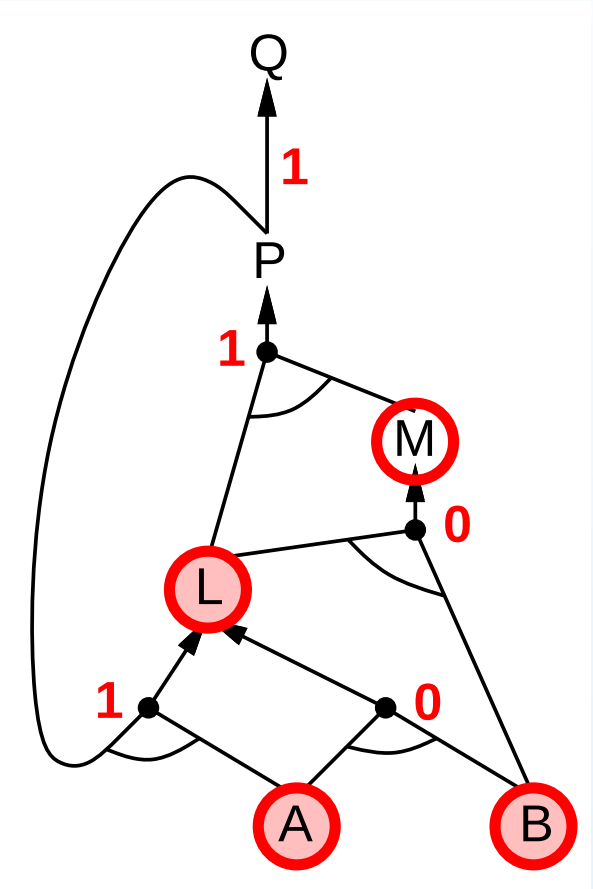
$B \wedge L \rightarrow M$

$A \wedge P \rightarrow L$

$A \wedge B \rightarrow L$

A

B



Przykład wnioskowania w przód

Dowiedź, że Q.

$P \rightarrow Q$

$L \wedge M \rightarrow P$

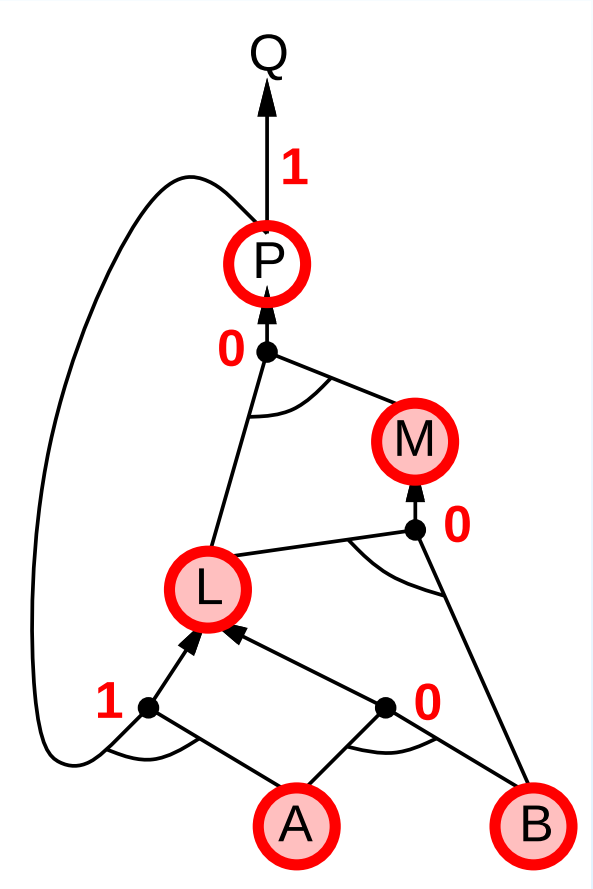
$B \wedge L \rightarrow M$

$A \wedge P \rightarrow L$

$A \wedge B \rightarrow L$

A

B



Przykład wnioskowania w przód

Dowiedź, że Q.

$P \rightarrow Q$

$L \wedge M \rightarrow P$

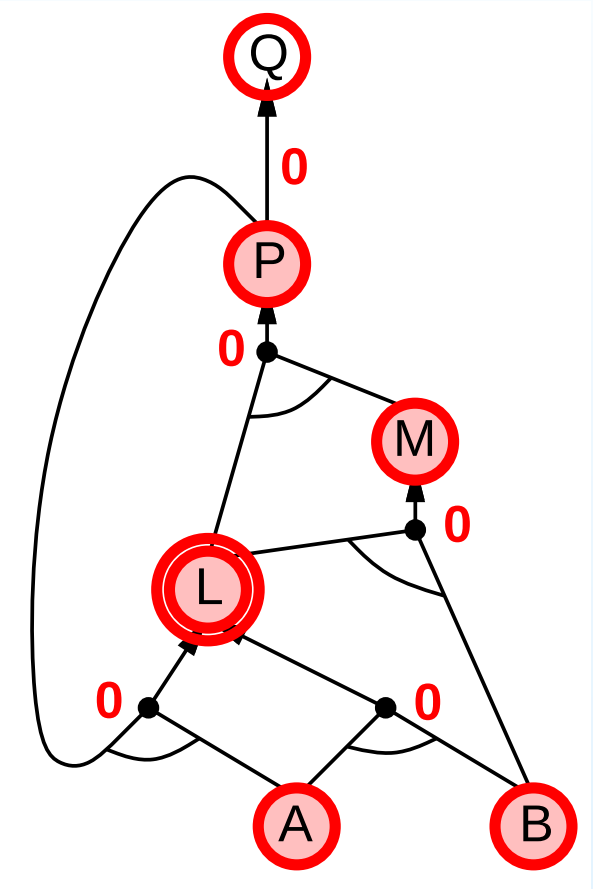
$B \wedge L \rightarrow M$

$A \wedge P \rightarrow L$

$A \wedge B \rightarrow L$

A

B



Przykład wnioskowania w przód

Dowiedź, że Q.

$P \rightarrow Q$

$L \wedge M \rightarrow P$

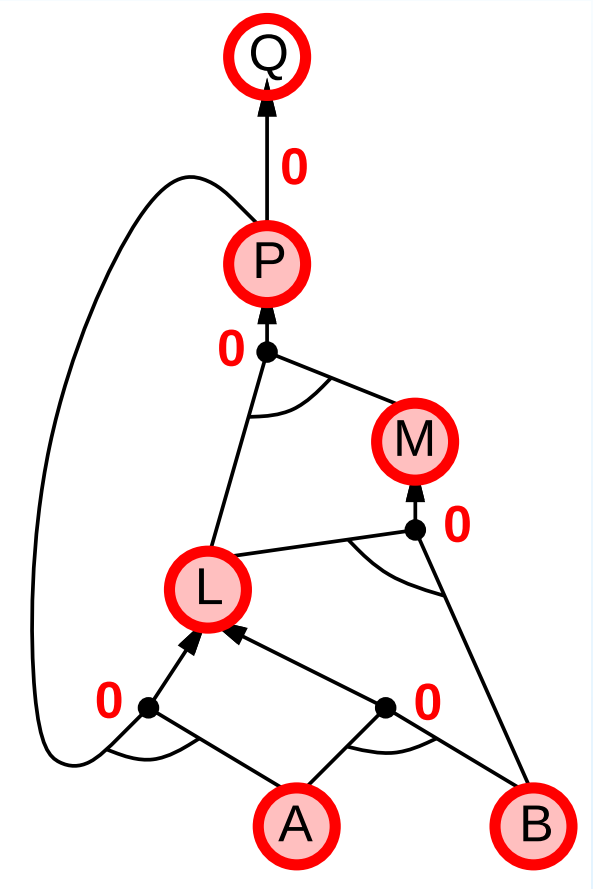
$B \wedge L \rightarrow M$

$A \wedge P \rightarrow L$

$A \wedge B \rightarrow L$

A

B



Przykład wnioskowania w przód

Dowiedź, że Q.

$P \rightarrow Q$

$L \wedge M \rightarrow P$

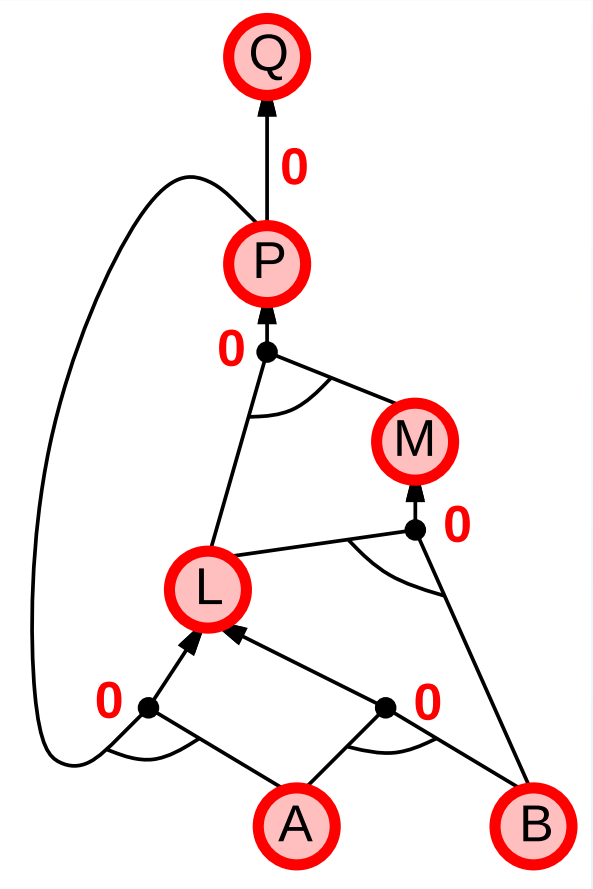
$B \wedge L \rightarrow M$

$A \wedge P \rightarrow L$

$A \wedge B \rightarrow L$

A

B



Wnioskowanie wstecz

Idea: skanuj zależności wstecz zaczynając do zapytania q.
Aby dowieść q przez wnioskowanie wstecz:
sprawdź czy q jest już znane
lub
dowiedź przez wnioskowanie wstecz wszystkie przesłanki reguł mających q w konkluzji.

Unikaj pętli: sprawdzaj czy nowy podcel nie był już sprawdzany.

Unikaj powtarzania pracy: sprawdź czy nowy podcel

1. nie osiągnął TRUE,
2. już przypadkiem nie zawiódł (FALSE).

Przykład wnioskowania wstecz

Dowiedź, że Q.

$P \rightarrow Q$

$L \wedge M \rightarrow P$

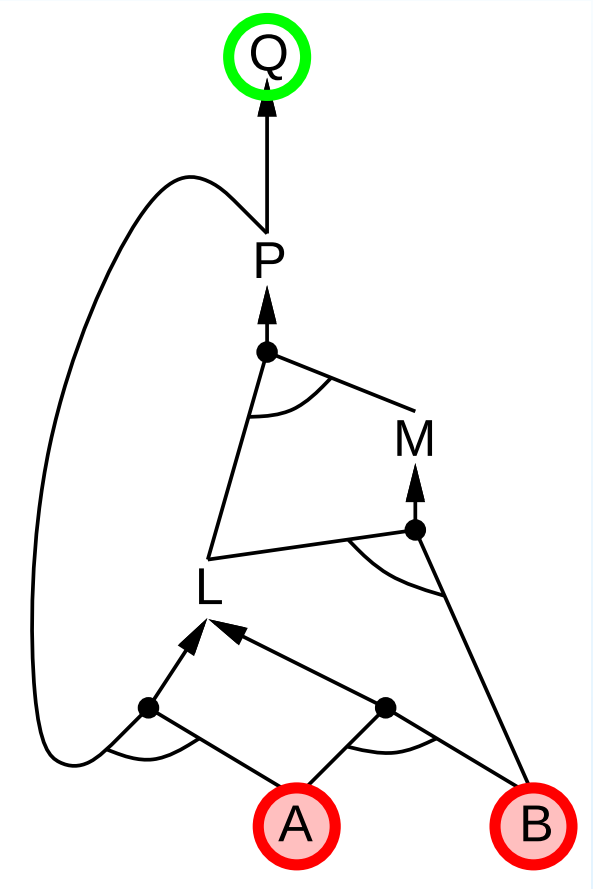
$B \wedge L \rightarrow M$

$A \wedge P \rightarrow L$

$A \wedge B \rightarrow L$

A

B



Przykład wnioskowania wstecz

Dowiedź, że Q.

$P \rightarrow Q$

$L \wedge M \rightarrow P$

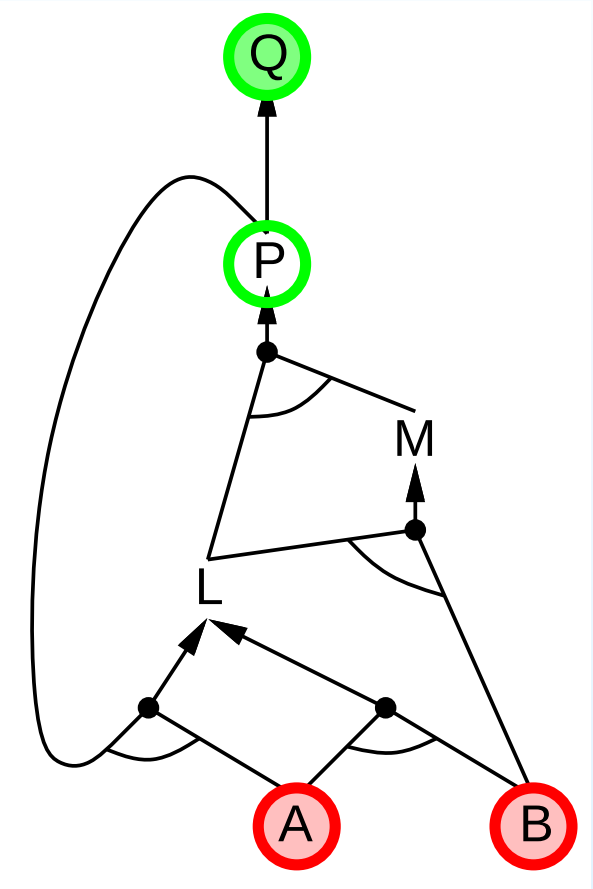
$B \wedge L \rightarrow M$

$A \wedge P \rightarrow L$

$A \wedge B \rightarrow L$

A

B



Przykład wnioskowania wstecz

Dowiedź, że Q.

$P \rightarrow Q$

$L \wedge M \rightarrow P$

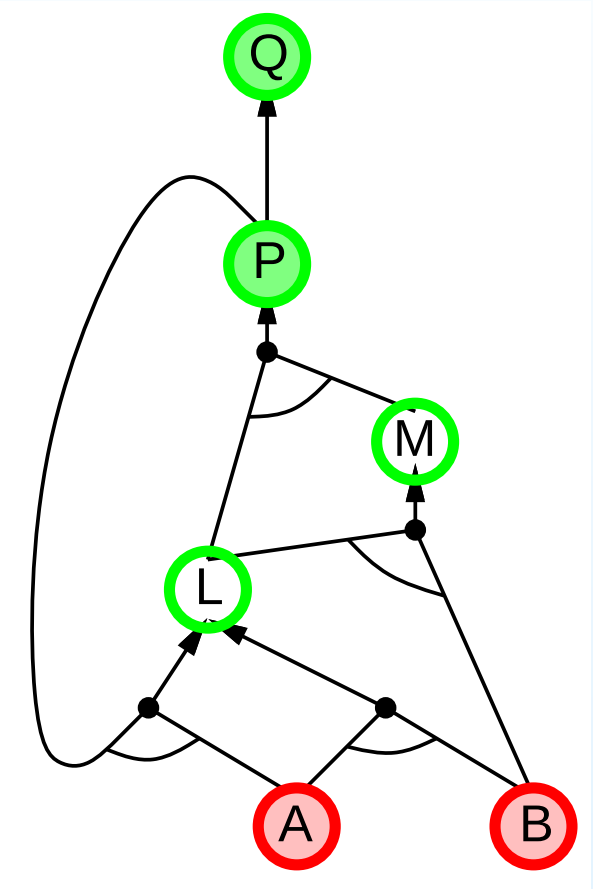
$B \wedge L \rightarrow M$

$A \wedge P \rightarrow L$

$A \wedge B \rightarrow L$

A

B



Przykład wnioskowania wstecz

Dowiedź, że Q.

$P \rightarrow Q$

$L \wedge M \rightarrow P$

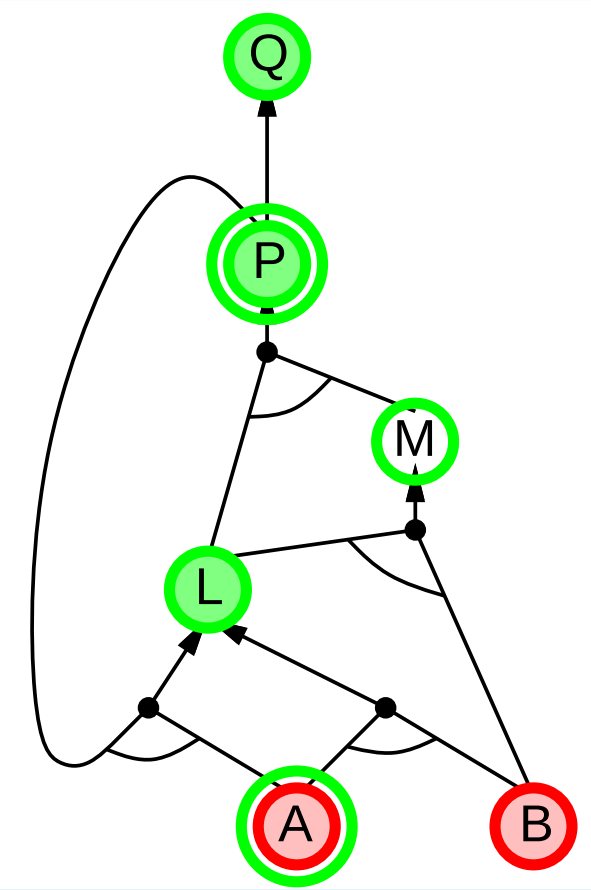
$B \wedge L \rightarrow M$

$A \wedge P \rightarrow L$

$A \wedge B \rightarrow L$

A

B



Przykład wnioskowania wstecz

Dowiedź, że Q.

$P \rightarrow Q$

$L \wedge M \rightarrow P$

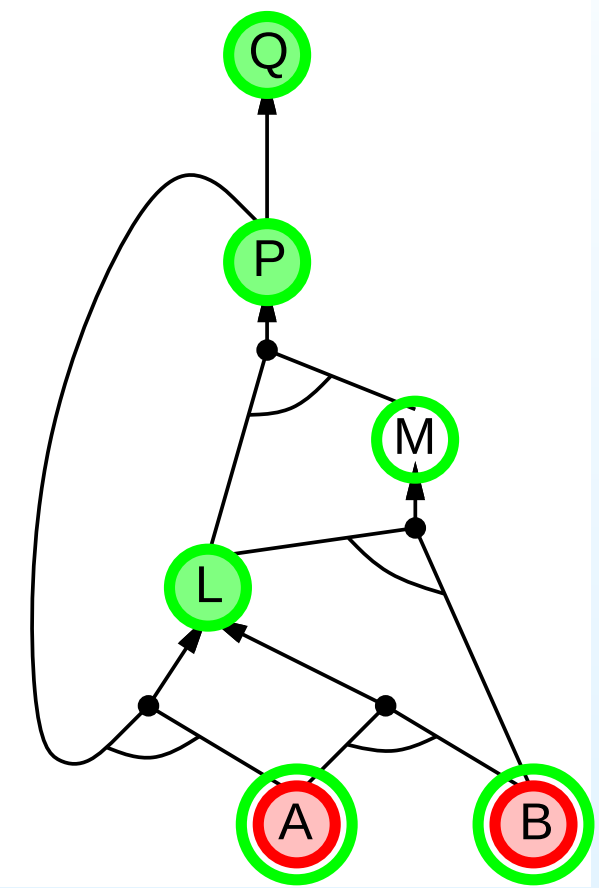
$B \wedge L \rightarrow M$

$A \wedge P \rightarrow L$

$A \wedge B \rightarrow L$

A

B



Przykład wnioskowania wstecz

Dowiedź, że Q.

$P \rightarrow Q$

$L \wedge M \rightarrow P$

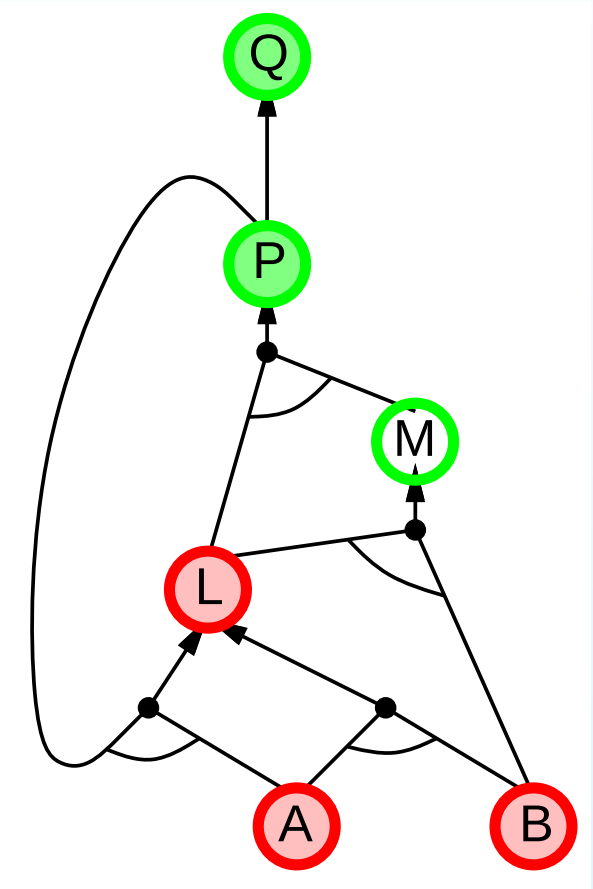
$B \wedge L \rightarrow M$

$A \wedge P \rightarrow L$

$A \wedge B \rightarrow L$

A

B



Przykład wnioskowania wstecz

Dowiedź, że Q.

$P \rightarrow Q$

$L \wedge M \rightarrow P$

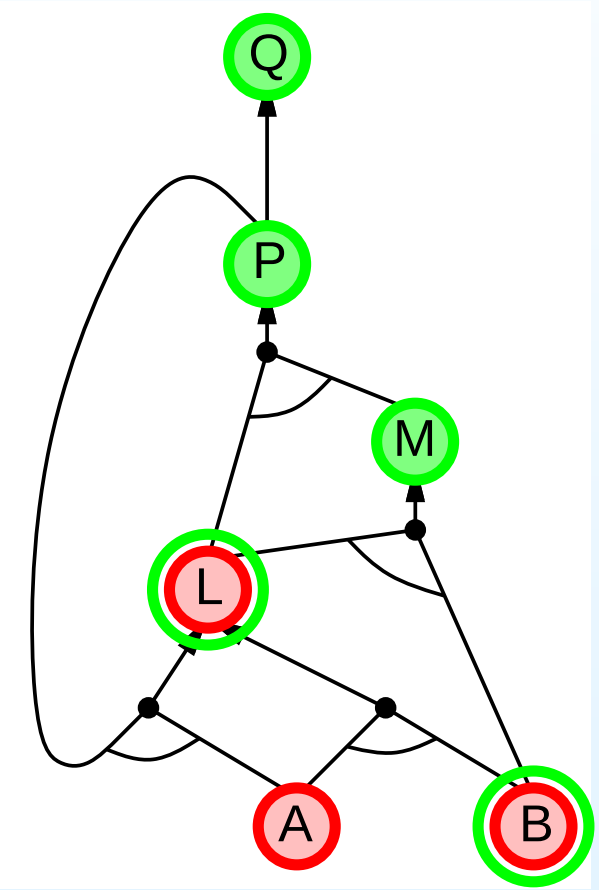
$B \wedge L \rightarrow M$

$A \wedge P \rightarrow L$

$A \wedge B \rightarrow L$

A

B



Przykład wnioskowania wstecz

Dowiedź, że Q.

$P \rightarrow Q$

$L \wedge M \rightarrow P$

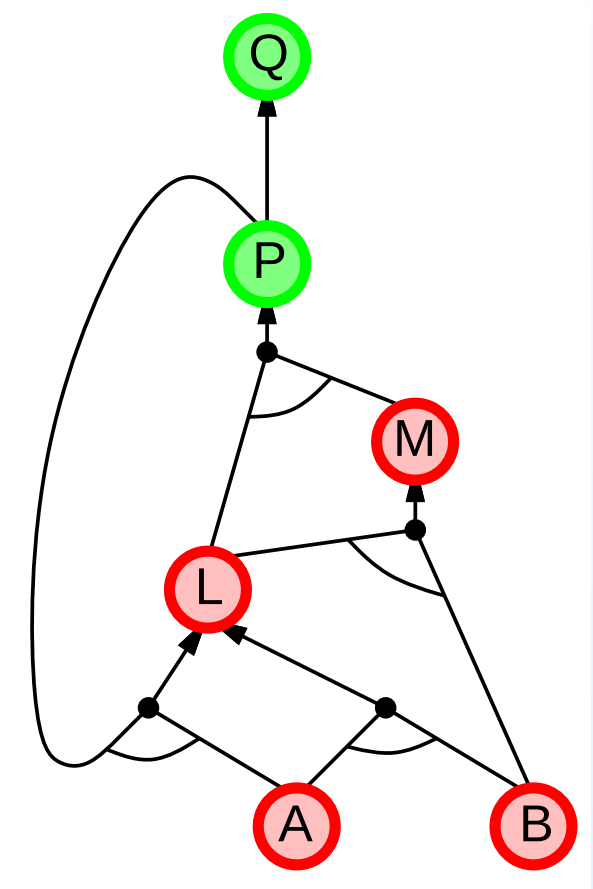
$B \wedge L \rightarrow M$

$A \wedge P \rightarrow L$

$A \wedge B \rightarrow L$

A

B



Przykład wnioskowania wstecz

Dowiedź, że Q.

$P \rightarrow Q$

$L \wedge M \rightarrow P$

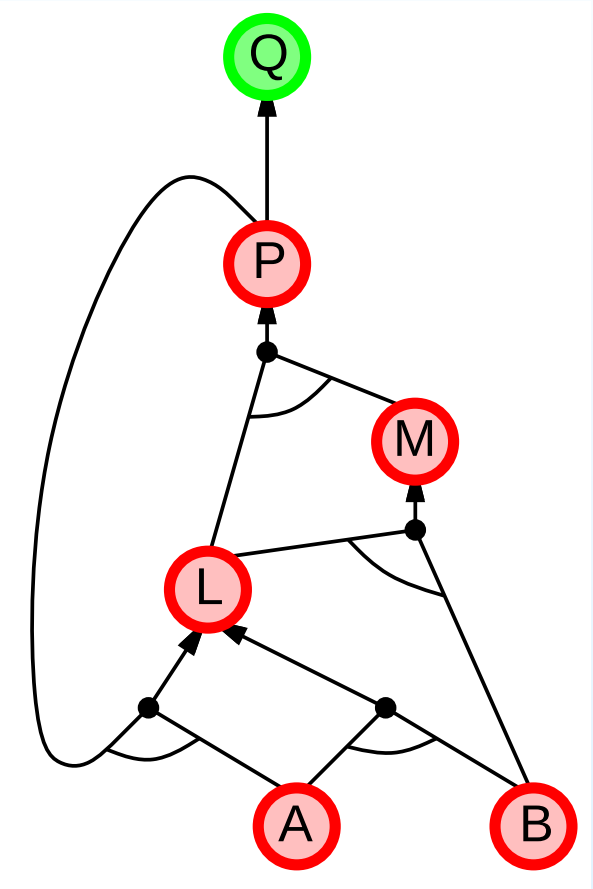
$B \wedge L \rightarrow M$

$A \wedge P \rightarrow L$

$A \wedge B \rightarrow L$

A

B



Przykład wnioskowania wstecz

Dowiedź, że Q.

$P \rightarrow Q$

$L \wedge M \rightarrow P$

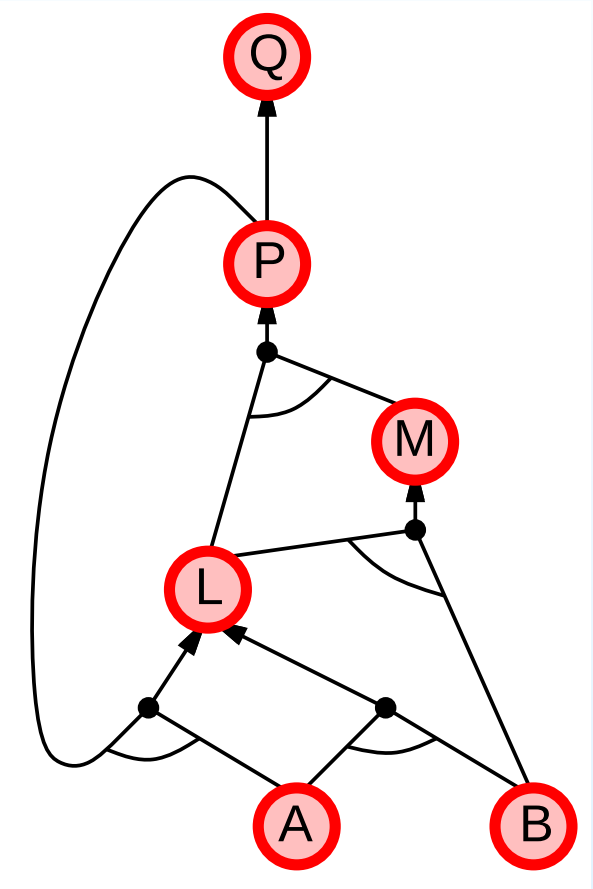
$B \wedge L \rightarrow M$

$A \wedge P \rightarrow L$

$A \wedge B \rightarrow L$

A

B



Porównanie wnioskowania wstecz i w przód

Wnioskowanie w kierunku celu zaczyna od danych. Symuluje automatyczne, nieświadome procesy np. rozpoznawanie obrazów.

Może wykonać wiele niepotrzebnych kroków, które nie prowadzą do celu.

Wnioskowanie wstecz w kierunku przesłanek zaczyna się od celu. Odpowiedni dla zadań typu „problem solving” np. Gdzie są moje klucze? Jak mogę otrzymać tytuł mgr?

Złożoność wnioskowania wstecz może być dużo mniejsza niż liniowa (od KB).