Politechnika Warszawska

OKNO

MSI

2. Logika klasyczna - rachunek zdań

Włodzimierz Kasprzak

Wersja 2023

Układ

1. Język logiki
2. Rachunek zdań
3. System wnioskowania
4. Postacie normalne
5. Wnioskowanie przez rezolucję
6. Wnioskowanie „w przód” i „wstecz”
7. Jak reprezentować własności zmienne w czasie?

1. Składnia rachunku zdań

Alfabet:

stałe logiczne *True* i *False*, symbole zdaniowe (P, Q, R,...), spójniki logiczne (koniunkcja, alternatywa, implikacja, równoważność, negacja) $\wedge$, $\vee$, $\Rightarrow$, $\Leftrightarrow$, $\neg$, nawiasy okrągłe **(**, **)**.

Zdania:

- *True* , *False* i symbole zdaniowe są zdaniami atomowymi.
- Jeśli A i B są zdaniami to $(A \wedge B)$, $(A \vee B)$, $(A \Rightarrow B)$ i $(A \Leftrightarrow B)$ są zdaniami.
- Jeśli A jest zdaniem to $\neg A$ jest zdaniem.

Literał to zdanie atomowe lub negacja zdania atomowego.

Własności rachunku zdań

+ Rachunek zdań jest deklaratywny.
+ Rachunek zdań reprezentuje informację: częściową, alternatywną, zanegowaną:
  – inaczej niż większość struktur danych i baz danych.
+ Rachunek zdań jest kompozycyjny:
  – znaczenie $B_{11} \wedge P_{12}$ jest wyprowadzane ze znaczeń B_{11} i P_{12}
+ Znaczenie w rachunku zdań jest niezależne od kontekstu
  – inaczej niż w języku naturalnym, gdzie znaczenie zależy od kontekstu.

- Rachunek zdań ma ograniczone możliwości wyrazu:
  – inaczej niż język naturalny; np. nie można powiedzieć generalnie, że „jamy powodują wiatr w sąsiednich kwadratach“ tylko trzeba stworzyć po jednym zdaniu tego typu dla każdego rzeczywistego kwadratu.

2. Semantyka rachunku zdań

Semantyka rachunku zdań może być formalnie ujęta jako struktura algebraiczna: <***D***, *m* >.

- **Dziedzina** ***D*** **–** zbiór faktów do których odnoszą się zdania.
- **Interpretacja** *m* – funkcja przyporządkowująca symbolom zdaniowym fakty należące do dziedziny i wartościująca zdania jako *True* lub *False*.
- **Model zdania** *A* – interpretacja, w której zdanie *A* jest prawdziwe.
- **Tautologia** – zdanie prawdziwe w każdej interpretacji (modelu świata).
- **Rachunek zdań jest rozstrzygalny**, tzn. istnieje algorytm, który dla dowolnego zdania *A* i interpretacji *m* stwierdza, czy *A* jest prawdziwe.

Tabele prawdy dla spójników

P	Q		$\neg$P	P$\wedge$Q	P$\vee$Q	P$\Rightarrow$Q	P$\Leftrightarrow$Q
0	0		1	0	0	1	1
0	1		1	0	1	1	0
1	0		0	0	1	0	0
1	1		0	1	1	1	1

Interpretacja spójników:

$\neg S$ jest prawdziwe wtw. S jest fałszywe

$S_1 \wedge S_2$ jest prawdziwe wtw. S_1 jest prawdą i S_2 jest prawdą

$S_1 \vee S_2$ jest prawdziwe wtw. S_1 jest prawdą lub S_2 jest prawdą

$S_1 \Rightarrow S_2$ jest prawdziwe wtw. S_1 jest fałszywe lub S_2 jest prawdą, tzn. jest fałszywe wtw. S_1 jest prawdą i S_2 jest fałszywe

$S_1 \Leftrightarrow S_2$ jest prawdziwe wtw. $S_1 \Rightarrow S_2$ jest prawdą i $S_2 \Rightarrow S_1$ jest prawdą

Przykład tautologii

- Dzięki tablicy prawdy pokażemy, że zdanie złożone

$$((P \vee H) \wedge \neg H) \Rightarrow P$$

  jest **tautologią**, tzn. **zawsze prawdziwe** niezależnie od modelu dla P i H:

P	H		P∨H	(P∨H)∧¬H	((P∨H)∧¬H)⇒P
0	0		0	0	1
0	1		1	0	1
1	0		1	1	1
1	1		1	0	1

Ostatnia kolumna reprezentuje nasze zdanie i jest zawsze prawdziwa.

3. Wnioskowanie

- **Procedura sprawdzająca wszystkie modele** jest procedurą wnioskowania poprawną i zupełną. Może ona zostać efektywnie zaimplementowana wtedy, gdy formuły dają się uporządkować (np. odpowiednio do odległości miejsca od pozycji startowej agenta, do którego te formuły się odnoszą).
- Np. funkcja WynikanieZTabPrawdy() (na następnej stronie) jest efektywną implementacją procedury wnioskowania. Korzysta ona z rekursywnej funkcji TabPrawdy(), wywoływanej dla **częściowego** i stopniowo rozszerzanego **modelu**. Podfunkcja CzyModel() sprawdza poprawność zdania (lub bazy wiedzy) w częściowym modelu a podfunkcja Extend() rozszerza model o wartość dla kolejnego symbolu (zdania).
- Dla WynikanieZTabPrawdy() jej spodziewana **złożoność obliczeniowa** również wynosi $O(2^n)$, przy n symbolach.

Sprawdzanie wszystkich modeli

funkcja WynikanieZTabPrawdy(*KB*, α)
zwraca wynik: true lub **false**
{ *symbole* ← *symbole zdaniowe w KB* i α *;*
  return TabPrawdy(*KB*, α , *symbole*, []);
}

funkcja TabPrawdy(*KB*, α, *symbole*, *model*)
zwraca wynik: true lub **false**
{ **if** (*symbole* == $\emptyset$) {
    if CzyModel(*KB*, *model*) **return** CzyModel(α, *model*);
    else return true;
  } **else** {
   P ← Pierwszy(*symbole*); reszta ← Reszta(*symbole*);
   return (TabPrawdy(*KB*, α, *reszta*, Rozszerz(*P*, **true**, *model*)
     && TabPrawdy(*KB*, α, *reszta*, Rozszerz(*P*, **false**,*model*));
} }

Twierdzenia o dedukcji

Wnioskowanie w rachunku zdań

1. **Zdanie jest** **tautologią** wtw. gdy jest prawdziwe we wszystkich modelach. Np.:

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

Tautologia jest powiązana z wnioskowaniem poprzez pierwsze **twierdzenie o dedukcji:**

– $KB \models \alpha$ wtw. gdy **zdanie** $(KB \Rightarrow \alpha)$ jest **tautologią**

2. Zdanie jest **spełnialne** wtw. gdy posiada model. Zdanie jest **niespełnialne** wtw. gdy nie posiada żadnego modelu.

Np.: $A \wedge \neg A$ jest niespełnialne.

Spełnialność jest powiązana z wnioskowaniem poprzez **drugie twierdzenie o dedukcji:**

$KB \models \alpha$ wtw. gdy **zdanie** $(KB \wedge \neg \alpha)$ **jest niespełnialne**.

Ta równoważność prowadzi do **dowodu przez zaprzeczenie**.

Logiczna równoważność

Dwa zdania są **logicznie równoważne** wtw. gdy są poprawne w tych samych modelach:

$\alpha \equiv \beta$ wtw. $(\alpha \models \beta)$ i $(\beta \models \alpha)$

$(\alpha \wedge \beta) \equiv (\beta \wedge \alpha)$ – przemienność $\wedge$

$(\alpha \vee \beta) \equiv (\beta \vee \alpha)$ – przemienność $\vee$

$((\alpha \wedge \beta) \wedge \lambda) \equiv (\alpha \wedge (\beta \wedge \lambda))$– łączność $\wedge$

$((\alpha \vee \beta) \vee \lambda) \equiv (\alpha \vee (\beta \vee \lambda))$– łączność $\wedge$

$\neg(\neg \alpha) \equiv \alpha$ – eliminacja podwójnej negacji

$(\alpha \Rightarrow \beta) \equiv (\neg \beta \Rightarrow \neg \alpha)$ – kontrapozycja

$(\alpha \Rightarrow \beta) \equiv (\neg \alpha \vee \beta)$ – eliminacja implikacji

$(\alpha \Leftrightarrow \beta) \equiv ((\alpha \Rightarrow \beta) \wedge (\beta \Rightarrow \alpha)$ – eliminacja równoważności

$\neg(\alpha \wedge \beta) \equiv (\neg \alpha \vee \neg \beta)$ – prawo de Morgana

$\neg(\alpha \vee \beta) \equiv (\neg \alpha \wedge \neg \beta)$ – prawo de Morgana

$(\alpha \wedge (\beta \vee \lambda)) \equiv ((\alpha \wedge \beta) \vee (\alpha \wedge \lambda))$ – rozdzielczość $\wedge$ względem $\vee$

$(\alpha \vee (\beta \wedge \lambda)) \equiv ((\alpha \vee \beta) \wedge (\alpha \vee \lambda))$ – rozdzielczość $\vee$ względem $\wedge$

Reguła wnioskowania

Dla rachunku zdań istnieją zupełne i poprawne **procedury wnioskowania** (dowodzenia zdań).

Stosują one **reguły wnioskowania** o ogólnej postaci:

$$\frac{poprzednik}{nastepnik}$$

Jeśli w bazie wiedzy spełniony jest warunek dany poprzednikiem reguły to wnioskowana jest poprawność następnika.

Reguła wnioskowania $\dfrac{\alpha_1, \alpha_2, \cdots, \alpha_n}{\beta}$ jest **poprawna**, jeśli zdanie $\boldsymbol{\beta}$ jest prawdziwe w każdej interpretacji, w której prawdziwe są zdania: $\boldsymbol{\alpha_1, \alpha_2, \ldots, \alpha_n}$.

Typowe reguły wnioskowania

Przykłady poprawnych reguł wnioskowania:

- Reguła **odrywania** (*modus ponens*): $$\frac{\alpha, \quad \alpha \Rightarrow \beta}{\beta}$$
- Reguła **eliminacji koniunkcji**: $$\frac{\alpha_1 \wedge \cdots \wedge \alpha_n}{\alpha_i}$$
- Reguła **wprowadzania koniunkcji**: $$\frac{\alpha_1, \cdots, \alpha_n}{\alpha_1 \wedge \cdots \wedge \alpha_n}$$
- Reguła **rezolucji**: $$\frac{\alpha \vee \beta, \quad \neg\beta \vee \gamma}{\alpha \vee \gamma}$$

Przykład: poprawność reguły rezolucji

- Sprawdzamy za pomocą tablicy prawdy poprawność reguły rezolucji:

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

- Wyróżniono wiersze dla których spełniony jest poprzednik tej reguły:

α	β	γ		$\alpha \vee \beta$	$\neg\beta \vee \gamma$		$\alpha \vee \gamma$
0	0	0		0	1		0
0	0	1		0	1		1
0	1	0		1	0		0
0	1	1		1	1		1
1	0	0		1	1		1
1	0	1		1	1		1
1	1	0		1	0		1
1	1	1		1	1		1

- Następnik reguły też jest wtedy zawsze spełniony → poprawna reguła.

Złożoność wnioskowania

- Problem pokazania (dowiedzenia), że określone **zdanie jest tautologią** jest ***NP*-zupełny**, czyli nie oczekujemy aby istniał algorytm wnioskowania w rachunku zdań mający złożoność wielomianową.
- Jednak w wielu przypadkach dowód może zostać przeprowadzony **efektywnie**, ponieważ wynikowe zdanie zwykle nie zależy od wszystkich zdań w *KB* i takie nadmiarowe zdania mogą zostać zignorowane. Ta własność wnioskowania jest rezultatem **monotoniczności**.
- **Monotoniczność**: jeśli formuła A wynika ze zbioru formuł $\mathbf{X}$ to A wynika również z każdego nadzbioru zbioru $\mathbf{X}$:

$$\text{Jeśli } \mathbf{X} \models \mathrm{A} \text{ to } (\mathbf{X}, \mathrm{B}) \models \mathrm{A}$$

Oznacza to, że można zastosować regułę wnioskowania, jeśli tylko poprzednik jest spełniony przez (część) formuł w bazie danych, bez względu na to, co jeszcze zawiera *KB*.

4. Postacie normalne

- **Postać kanoniczna Horna**: pojedynczy literał; lub

  (koniunkcja pozytywnych literałów) $\Rightarrow$ pozytywny literał .

  „Pozytywny” oznacza „nie zanegowany”. Np.:

$$KB = \{\ C\ , (B \Rightarrow A)\ , (C \wedge D \Rightarrow B)\ \}$$

- **Klauzula Horna** jest to równoważna postać alternatyw literałów, w z których jeden jest pozytywny a pozostałe – zanegowane. Np. $(\neg C \vee \neg D \vee B)$.
- Reguła odrywania (*Modus Ponens*) stosowana jest dla KB wyrażonej w postaci kanonicznej Horna :

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

- Z reguły odrywania korzystają procedury wnioskowania: progresywna (wprzód) lub regresywna (wstecz).

Postać normalna CNF i reguła rezolucji

- Conjunctive Normal Form (CNF): koniunkcja alternatyw literałów. Np.: $(A \vee \neg B) \wedge (B \vee \neg C \vee \neg D)$

- Reguła rezolucji (dla CNF):

$$\frac{l_1 \vee \ldots \vee l_k, \qquad m_1 \vee \ldots \vee m_n}{l_1 \vee \ldots \vee l_{i-1} \vee l_{i+1} \vee \ldots \vee l_k \vee m_1 \vee \ldots \vee m_{j-1} \vee m_{j+1} \vee \ldots \vee m_n}$$

gdzie l_i i m_j są komplementarnymi literałami.

Np.: $$\frac{P_{13} \vee P_{22}, \qquad \neg P_{22}}{P_{13}}$$

Poprawność reguły rezolucji

1) Zakładamy, że zachodzi poprzednik:

Równoważność implikacji: $\alpha \Rightarrow \beta \equiv \neg\alpha \vee \beta$

$$\neg(l_1 \vee \ldots \vee l_{i-1} \vee l_{i+1} \vee \ldots \vee l_k) \Rightarrow l_i$$

$$\neg m_j \Rightarrow (m_1 \vee \ldots \vee m_{j-1} \vee m_{j+1} \vee \ldots \vee m_n)$$

2) $l_i = \neg m_j$ i z przechodniości implikacji $\alpha \Rightarrow \beta \Rightarrow \gamma$ wynika:

$$\neg(l_1 \vee \ldots \vee l_{i-1} \vee l_{i+1} \vee \ldots \vee l_k) \Rightarrow (m_1 \vee \ldots \vee m_{j-1} \vee m_{j+1} \vee \ldots \vee m_n)$$

3) Ponownie z równoważności implikacji otrzymujemy:

$$l_1 \vee \ldots \vee l_{i-1} \vee l_{i+1} \vee \ldots \vee l_k \vee m_1 \vee \ldots \vee m_{j-1} \vee m_{j+1} \vee \ldots \vee m_n \,.$$

Konwersja zdania do CNF

Przykład. Dane jest zdanie: $B_{11} \Leftrightarrow (P_{12} \vee P_{21})$

1. Usuwamy $\Leftrightarrow$, zamieniając $\alpha \Leftrightarrow \beta$ 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. Usuwamy $\Rightarrow$, zamieniając $\alpha \Rightarrow \beta$ 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. Wprowadzamy $\neg$ do środka nawiasów stosując reguły de Morgana i ewent. eliminujemy podwójną negację:

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

4. Stosujemy prawo rozdzielczości ($\wedge$ nad $\vee$) i rozpisujemy:

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

Uwaga: Przed dodaniem zdania do KB stosujemy jeszcze regułę eliminacji koniunkcji (jeśli iloczyn klauzul jest spełniony to spełniona jest każda klauzula z osobna) i dodajemy zbiór klauzul.

5. Wnioskowanie przez rezolucję

Procedury wnioskowania (dowodzenia) formuł stosujące regułę rezolucji prowadzą **dowód przez zaprzeczenie**, tzn. aby dowieść, że

$$KB \models \alpha$$

pokazują one, że zdanie

$$(KB \wedge \neg \alpha)$$

jest niespełnialne.

Rezolucja w rachunku zdań

```
funkcja Rezolucja(KB, α)
zwraca wynik: true lub false
{ formuły ← zbiór podformuł (KB ∧ ¬α)  alternatyw literałów;
  nowe ← formuły;
  while ( true )  { aktualne ← ∅ ;
      for each (Ci ∈ nowe, Cj ∈ formuly) {
          rezolwenty ← KrokRezolucji(Ci, Cj);
          if (rezolwenty zawierają zdanie puste) return true;
          aktualne ← aktualne ∪ rezolwenty;
      }
      if (aktualne ⊆ formuły) return false;
      formuły ← formuły ∪ aktualne;  nowe ← aktualne;
  }
}
```

Funkcja Rezolucja() - komentarz

1) Zdania CNF w bazie wiedzy rozszerzonej o negację zapytania (KB $\wedge\neg\alpha$) rozdzielane są na zbiory **klauzul** – zdań o postaci alternatyw literałów.
2) Następnie w pętli, dla każdej odpowiedniej pary klauzul, funkcja KrokRezolucji() generuje *rezolwentę* swoich 2 argumentów, tzn. połączone zdania poprzednika reguły rezolucji pozbawione par komplementarnych symboli.
3) Są możliwe dwa sposoby zakończenia procedury:
   1. nie można dodać nowych zdań → α jest fałszywe w modelu M(KB).
   2. w wyniku rezolucji powstaje puste zdanie, co jest sprzecznością uzyskaną w warunkach zanegowanego zapytania, stąd → α jest prawdziwe w modelu M(KB).

Przykład wnioskowania metodą rezolucji

Niech fragment bazy wiedzy agenta to:

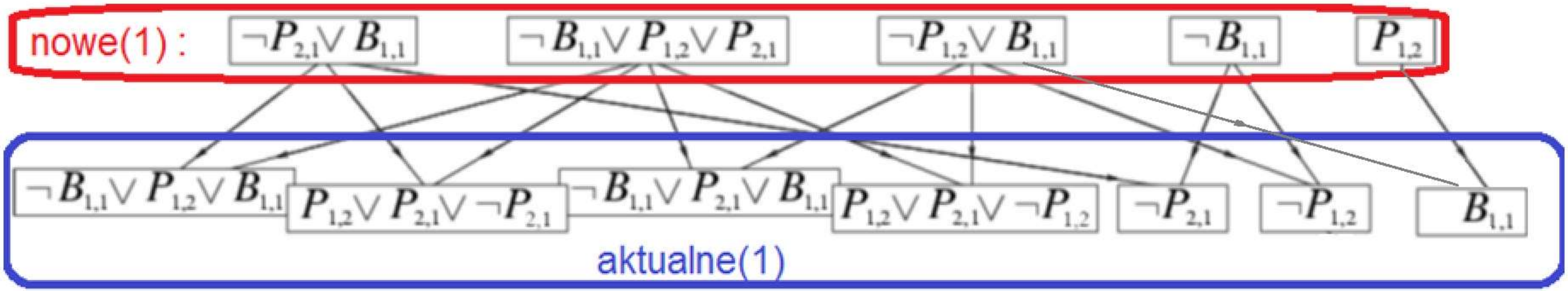
(KB zawiera m.in.) $(B_{11} \Leftrightarrow (P_{1,2} \vee P_{2,1})), \neg B_{11}$

Chcemy dowieść, że zachodzi: $\alpha = \neg P_{12}$.

1. Zamieniamy zdania na postać CNF (jeśli nie są jeszcze w tej postaci) a następnie zastępujemy je zbiorem klauzul o postaci alternatyw literałów. Zbiór „*formuły*” jest identyczny z „nowe(1)”:

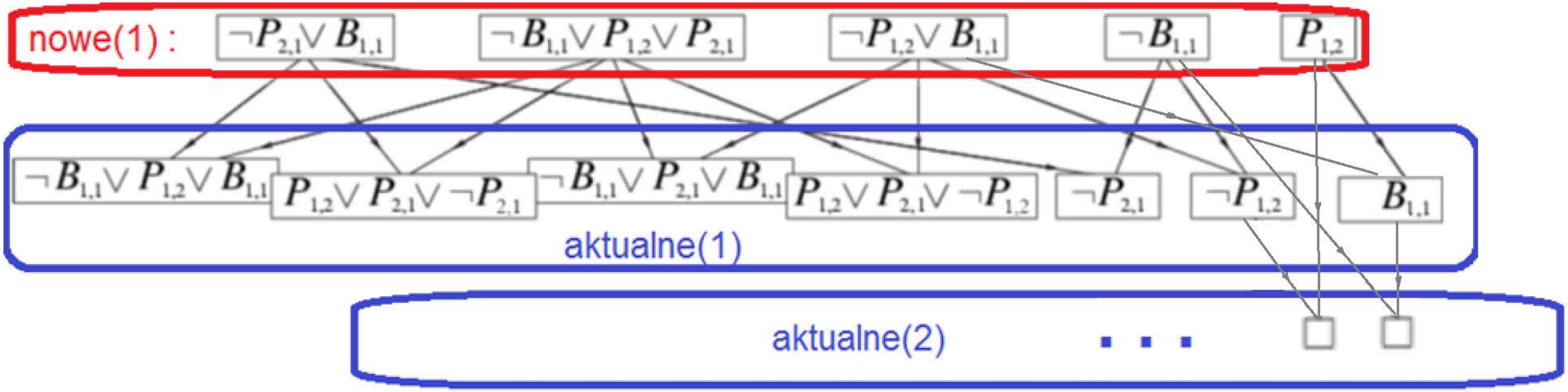
$\boxed{\neg P_{2,1} \vee B_{1,1}}$ $\boxed{\neg B_{1,1} \vee P_{1,2} \vee P_{2,1}}$ $\boxed{\neg P_{1,2} \vee B_{1,1}}$ $\boxed{\neg B_{1,1}}$ $\boxed{P_{1,2}}$

2. Wynik po pierwszej iteracji generowania rezolwent:



Przykład wnioskowania metodą rezolucji (2)

3. Wynik po drugiej iteracji:



- Aktualne(1): *rezolwenty* par zdań ze zbioru nowe(1) zawierających komplementarne literały.
- Kolejny wynik „aktualne(2)", m.in. zawiera dwukrotnie zdanie puste powstałe z rezolucji par P_{12} i $\neg P_{12}$ oraz B_{11} i $\neg B_{11}$ (w rachunku zdań wystarczy wygenerowanie pierwszego zdania pustego). Dowodzi to, że zapytanie wynika z bazy wiedzy: $KB \models \alpha$.

6. Wnioskowanie „w przód” i „wstecz”

- Procedury wnioskowania stosujące **regułę odrywania** (*Modus Ponens*) korzystają z ograniczonej postaci zdań, tzw. postaci kanonicznej (klauzul) Horna.
- Reguła odrywania (Modus Ponens) w rachunku zdań:

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

- Z reguły odrywania korzystają procedury wnioskowania:
  - progresywna (**w przód**) – generowanie zdań - i
  - regresywna (**wstecz**) – dowód wprost.

Wnioskowanie „w przód”

Idea procedury wnioskowania progresywnego („w przód”):

1. wykonaj każdą regułę, której warunek (poprzednik) jest spełniony w KB,
2. dodaj wynik wyprowadzenia (następnik reguły) do KB,
3. kontynuuj kroki 1-2 aż do znalezienia zdania zapytania lub niemożliwości wygenerowania nowych zdań.

Progresywna procedura wnioskowania jest poprawna i zupełna dla KB o postaci klauzul Horna.

Przykład wnioskowania „w przód” (1)

Dane są formuły w KB w postaci klauzul Horna:

$$p \Rightarrow q$$

$$m \wedge n \Rightarrow p$$

$$Ewa \wedge m \Rightarrow n$$

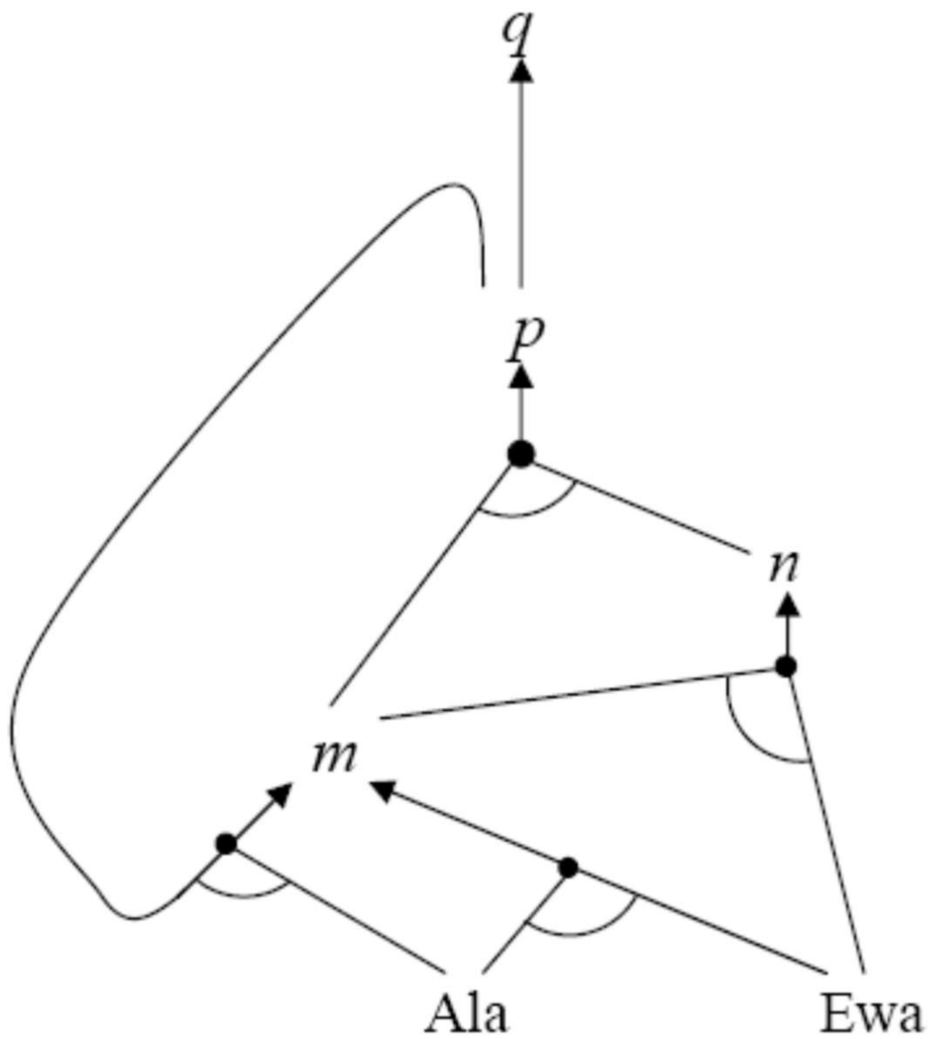
$$Ala \wedge p \Rightarrow m$$

$$Ala \wedge Ewa \Rightarrow m$$

$$Ala$$

$$Ewa$$

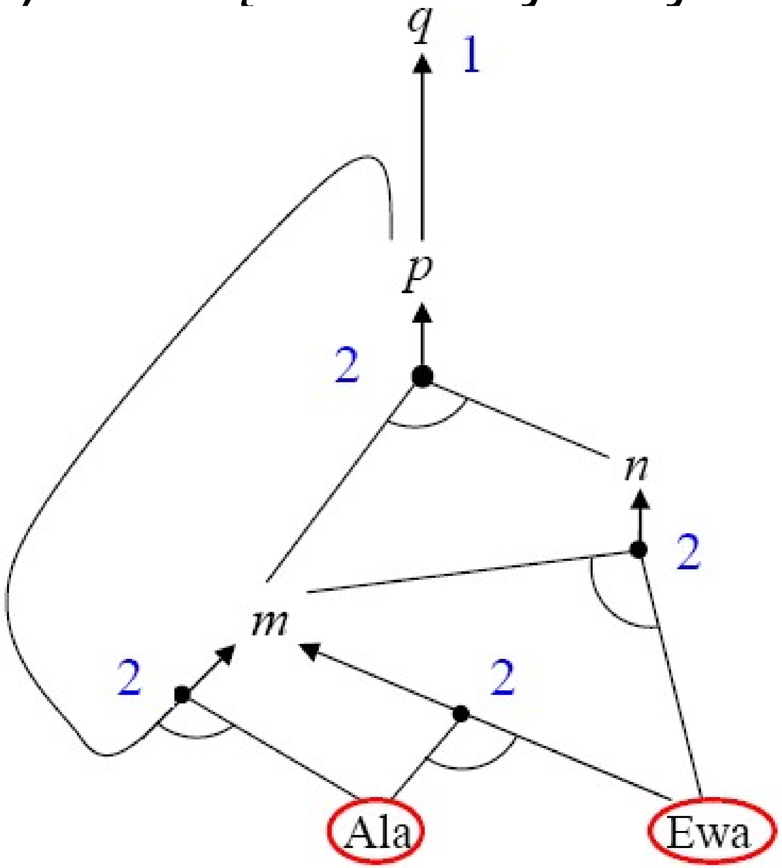
Odpowiada im graf I-LUB:



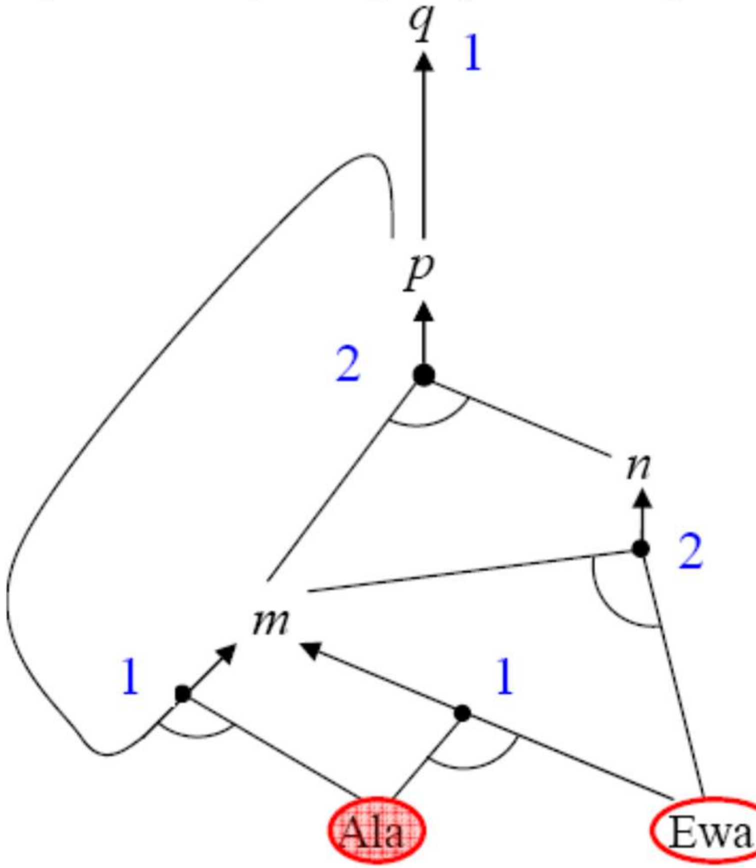
Przykład wnioskowania „w przód” (2)

Każdy węzeł typu I posiada etykietę – odpowiada ona liczbie warunków w poprzedniku reguły pozostających jeszcze do spełnienia.

(1) Początkowe etykiety

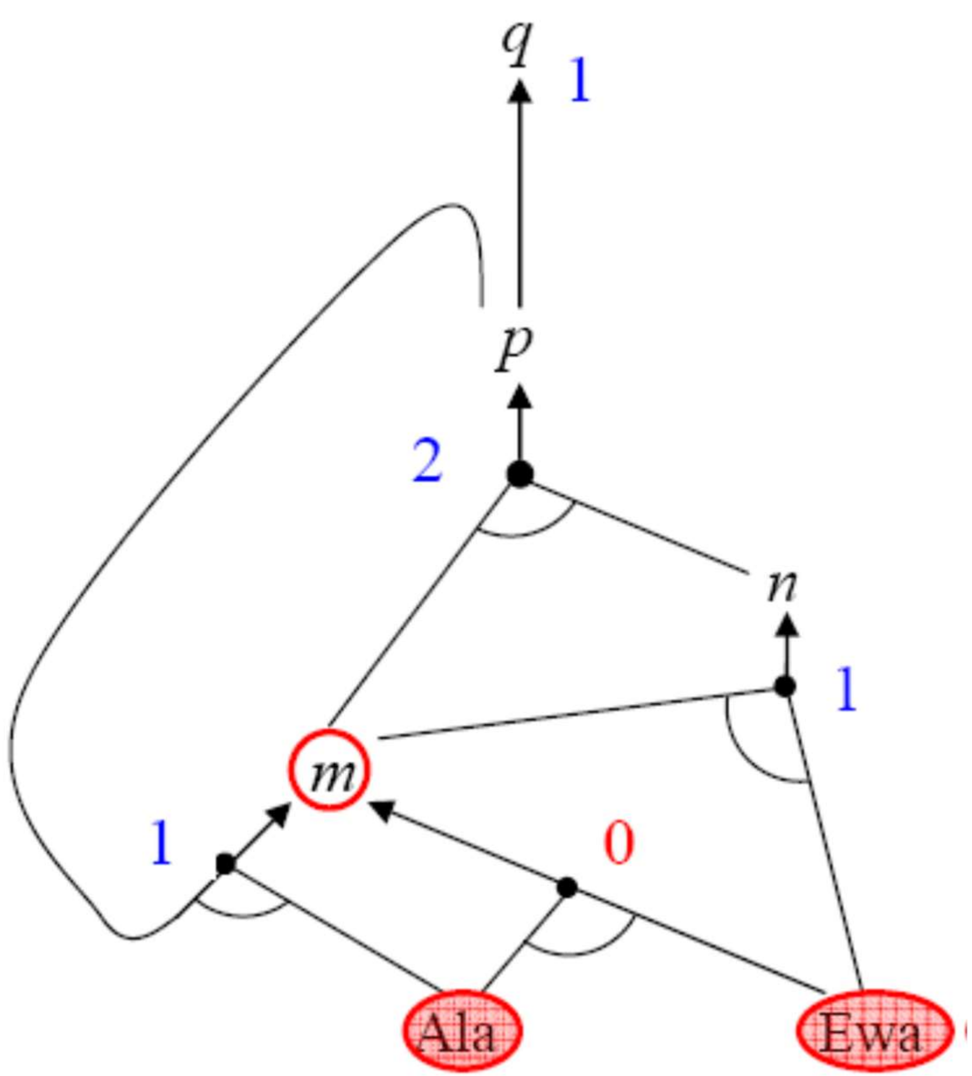


(2) Dodajemy symbol „Ala”

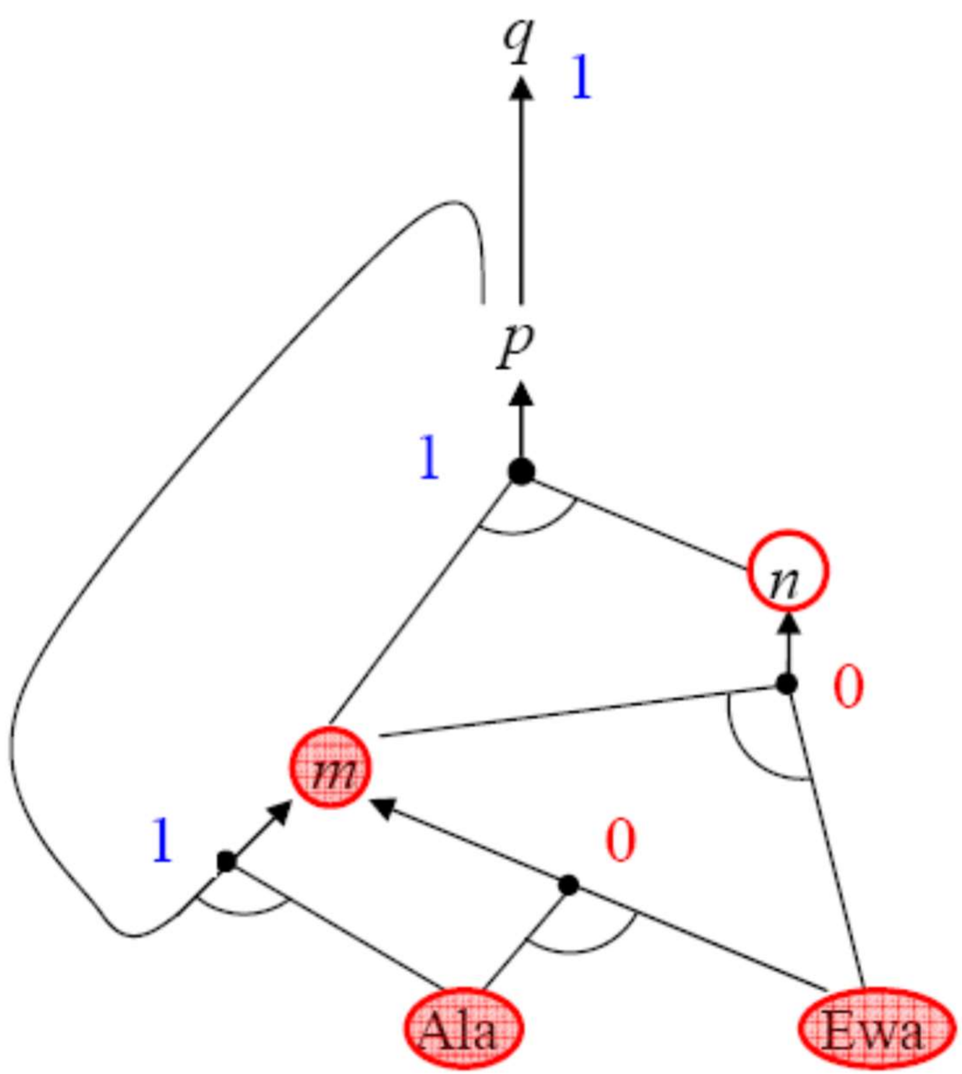


Przykład wnioskowania „w przód” (3)

(3) Po dodaniu „Ewa”

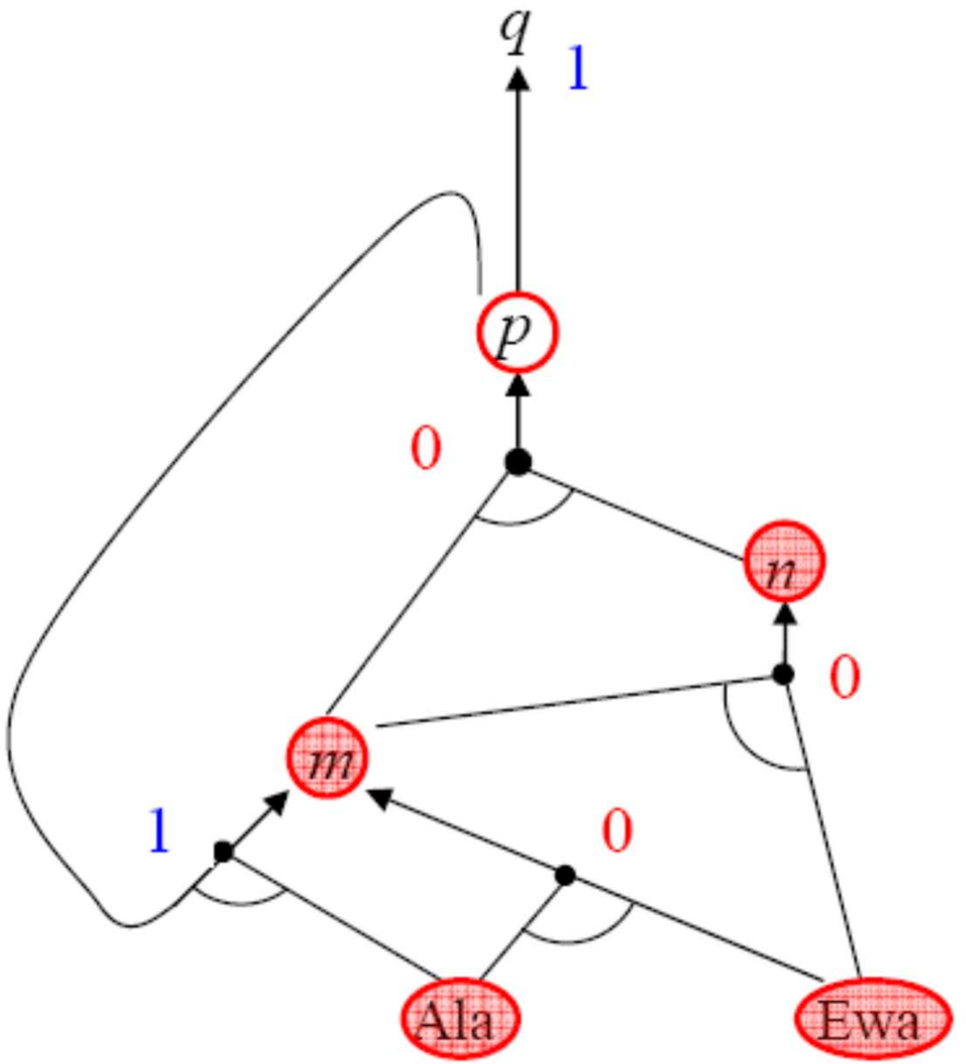


(4) Dodaj *m*

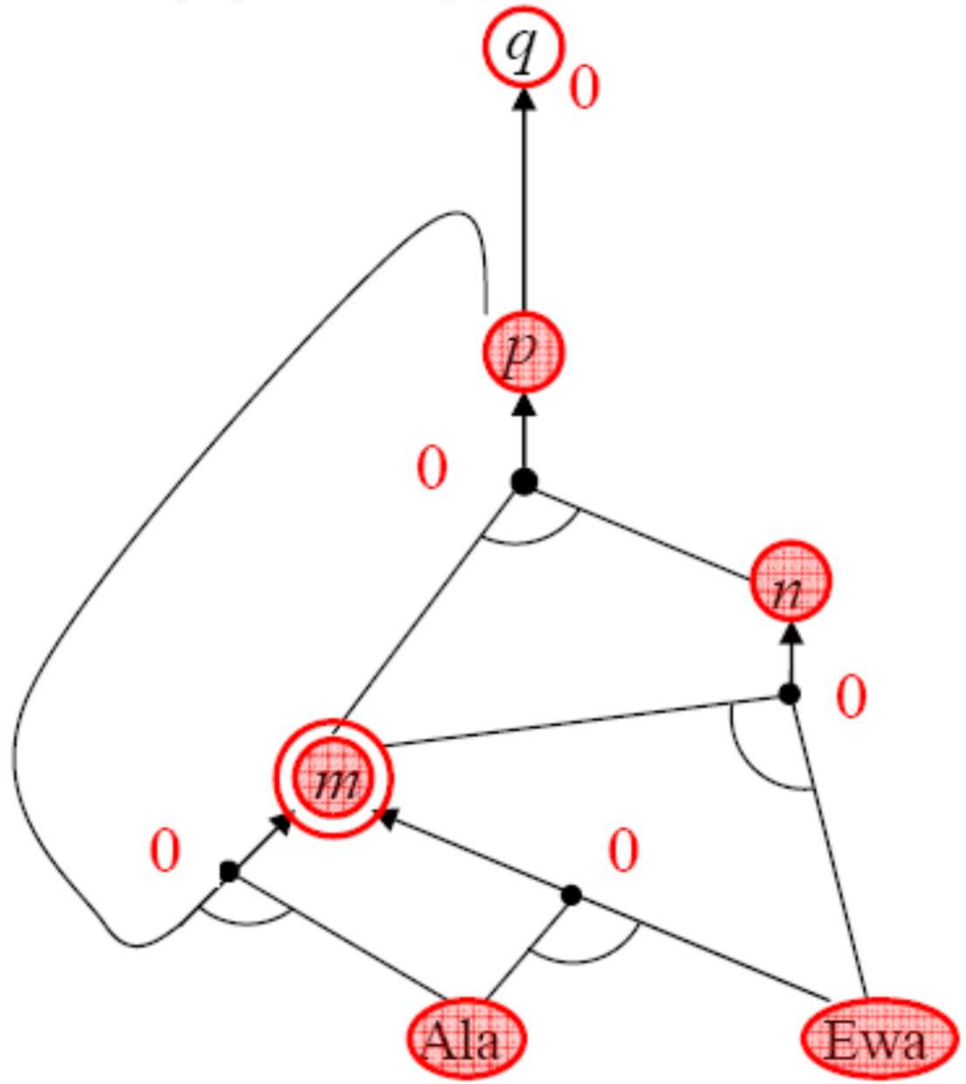


Przykład wnioskowania „w przód” (4)

(5) Dodaj *n*

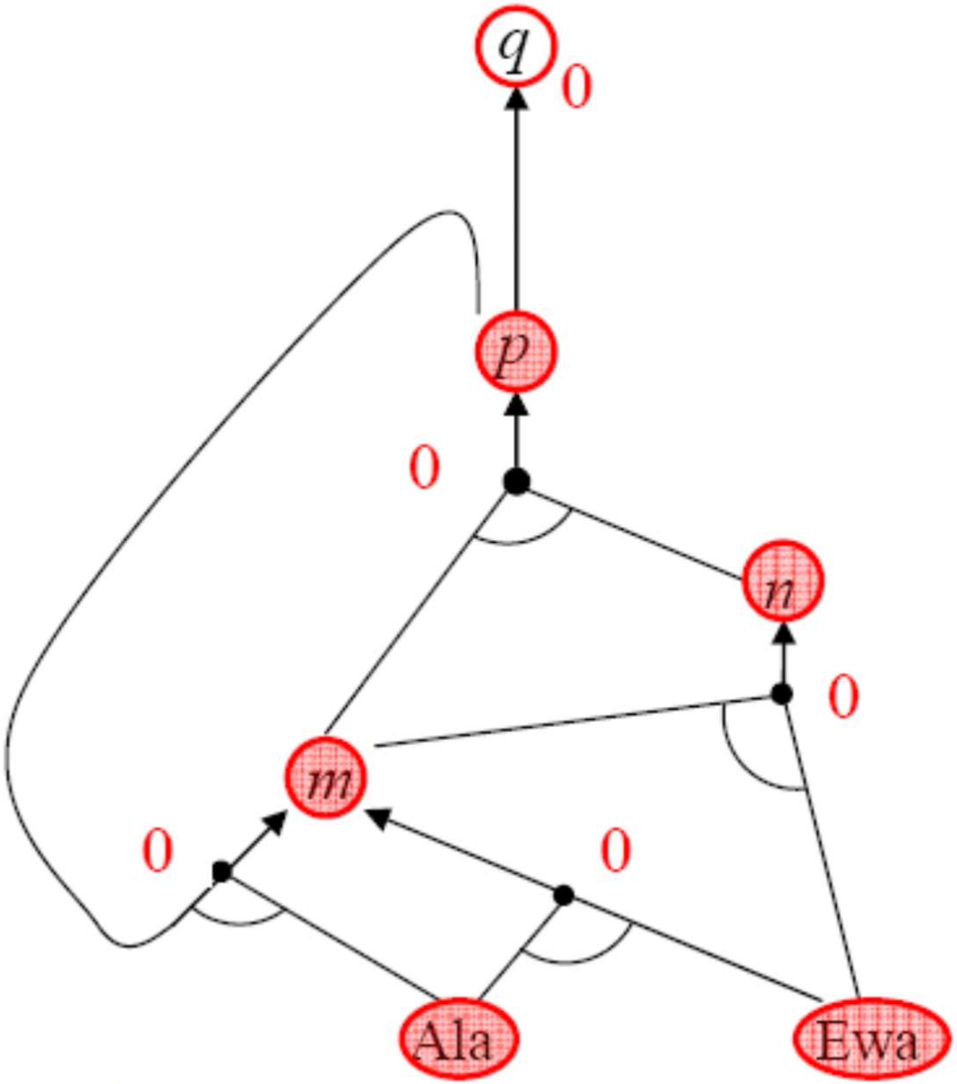


(6) Dodaj *p*

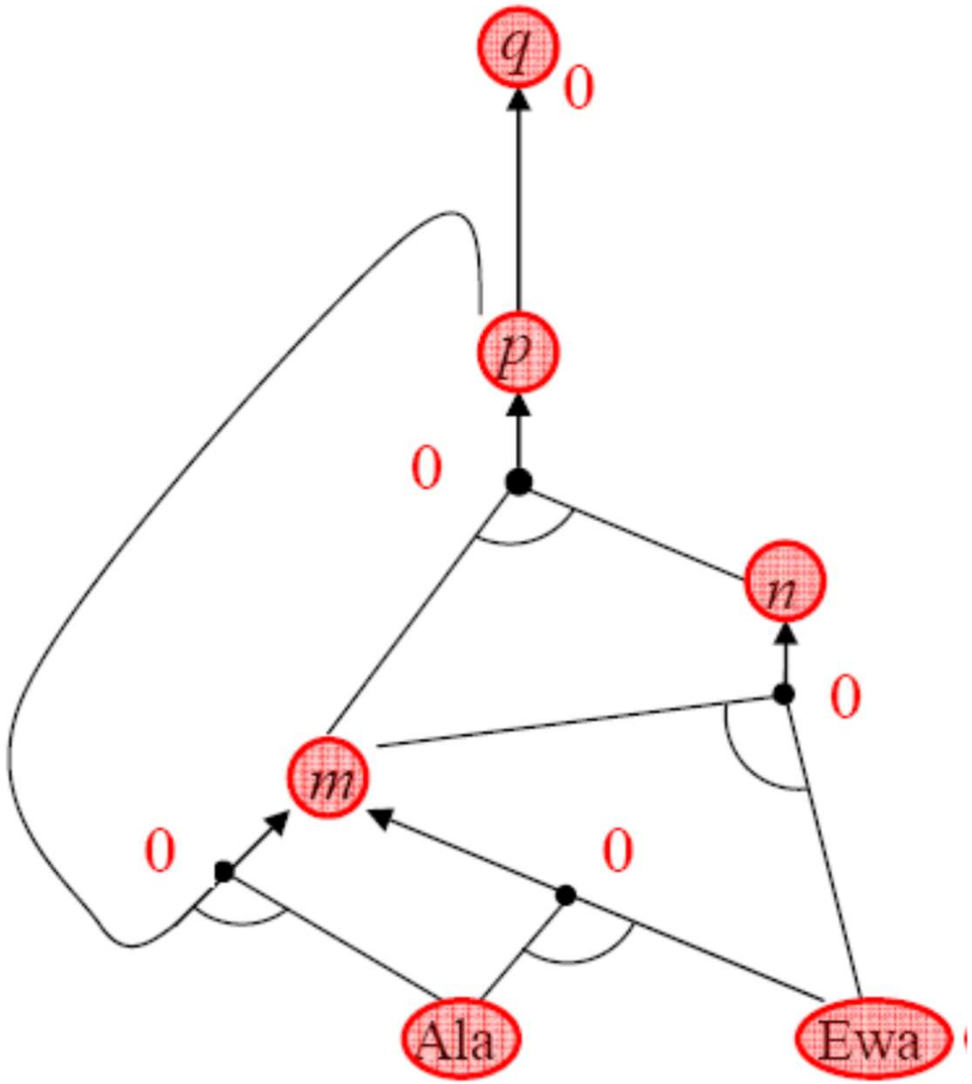


Przykład wnioskowania „w przód” (5)

(7) Próbuje ponownie dodać *m*



(8) Dodaj *q*

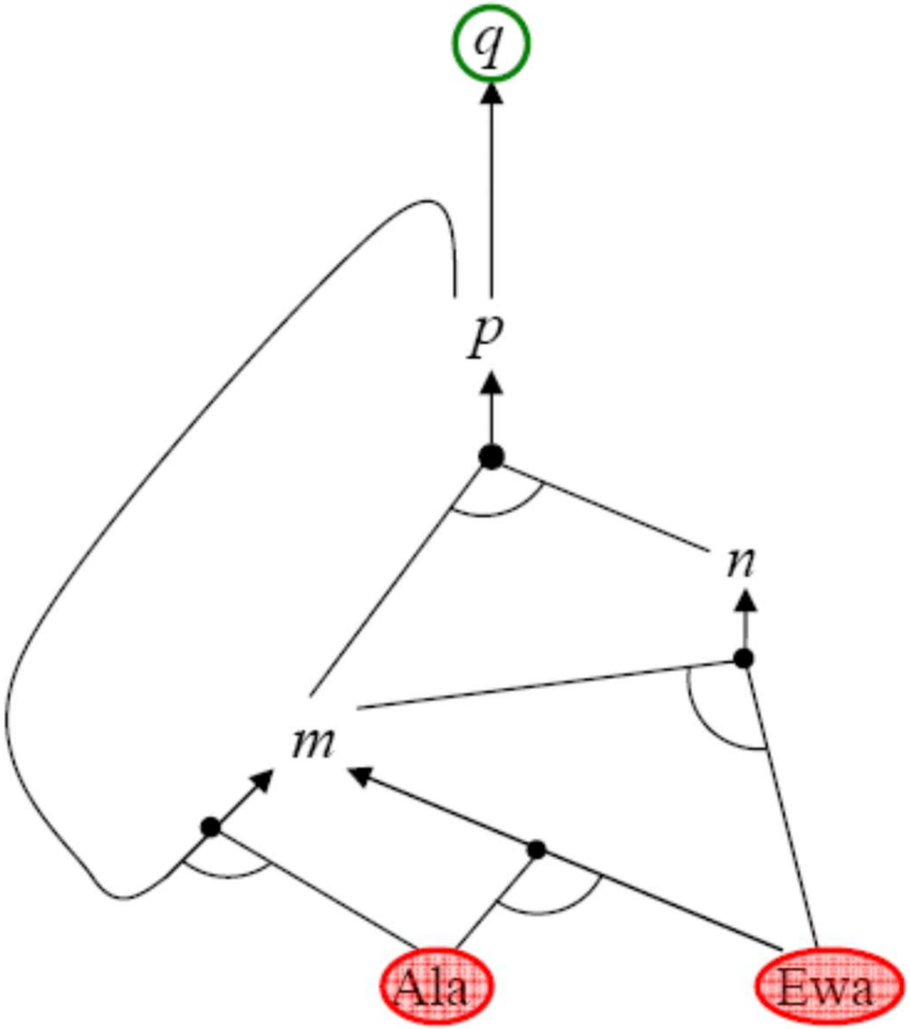


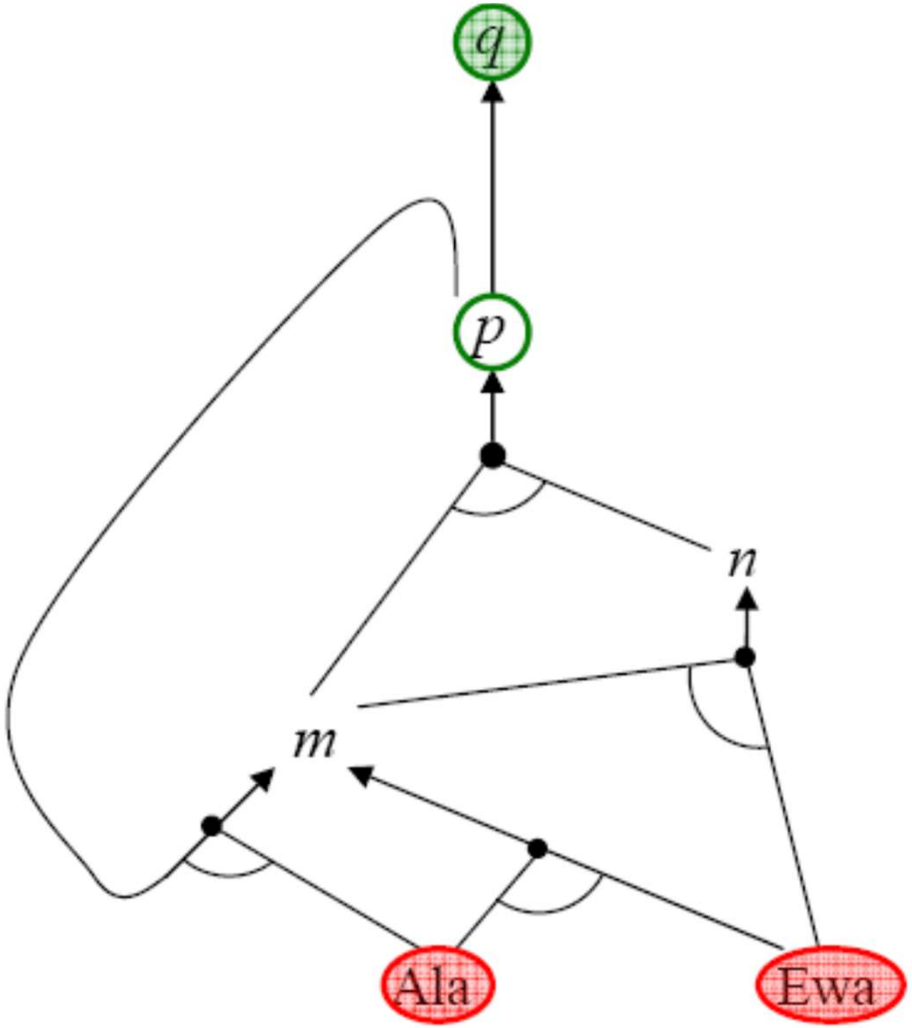
Wnioskowanie „wstecz”

- Funkcja rozpoczyna pracę od zdania zapytania (celu) q.
- Aby sprawdzić prawdziwość q procedura sprawdza, czy q już występuje a jeśli nie, to sprawdza czy istnieje przynajmniej jedna implikacja wyprowadzająca zdanie q. Jeśli tak, to literały stanowiące warunek tej implikacji stają się „pod-celami” i ich prawdziwość z punktu widzenia KB badana jest rekurencyjnie tak, jak poprzednio główny cel.
- Unikanie zapętleń: procedura sprawdza, czy aktualny „pod-cel” nie znajduje się już na stosie wygenerowanych „pod-celów”.
- Unikanie powielania przejść: sprawdza, czy nowy „pod-cel” został już sprawdzony i dowiedziono, że jest prawdziwy lub fałszywy.

Przykład wnioskowania „wstecz” (1)

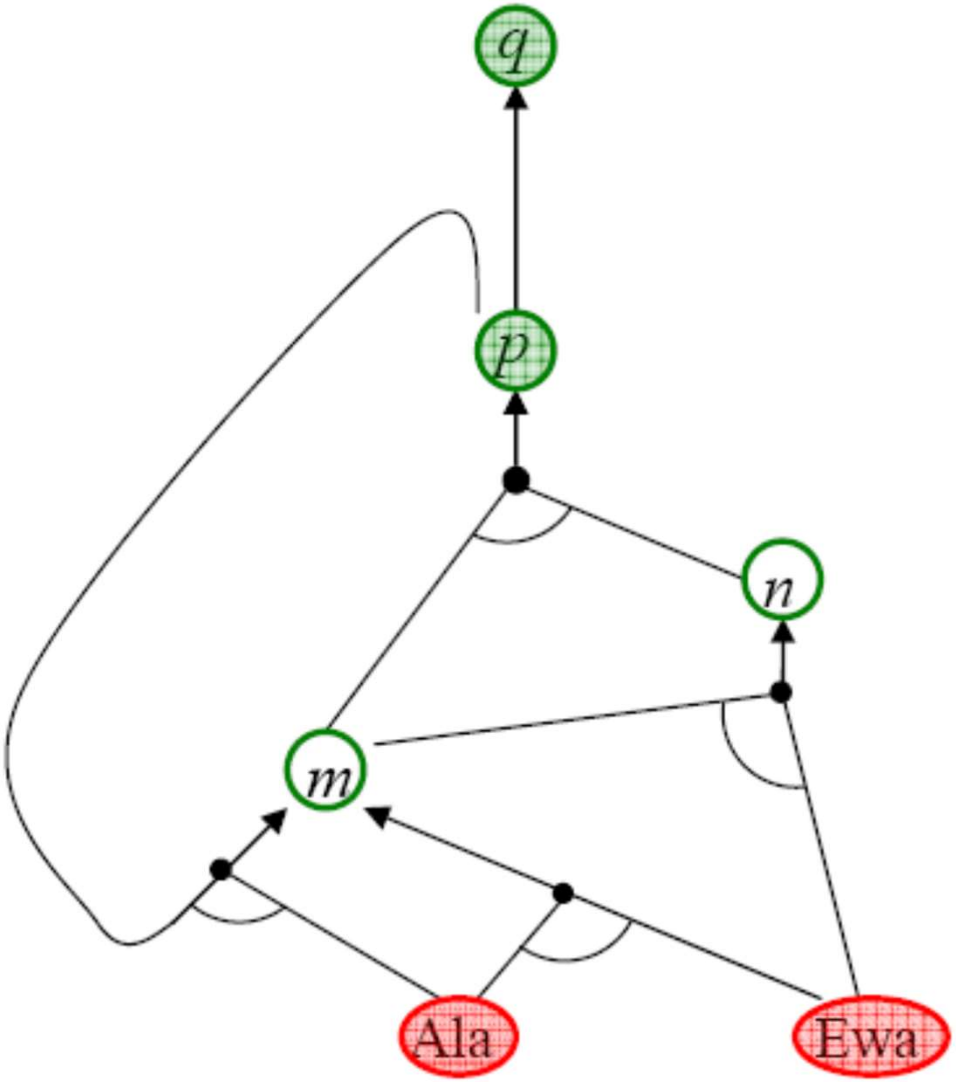
Baza danych zawiera 2 fakty: „Ala” i „Ewa”. Formuła zapytania to „q”. (1) Cel: q (2) Sprawdź: q



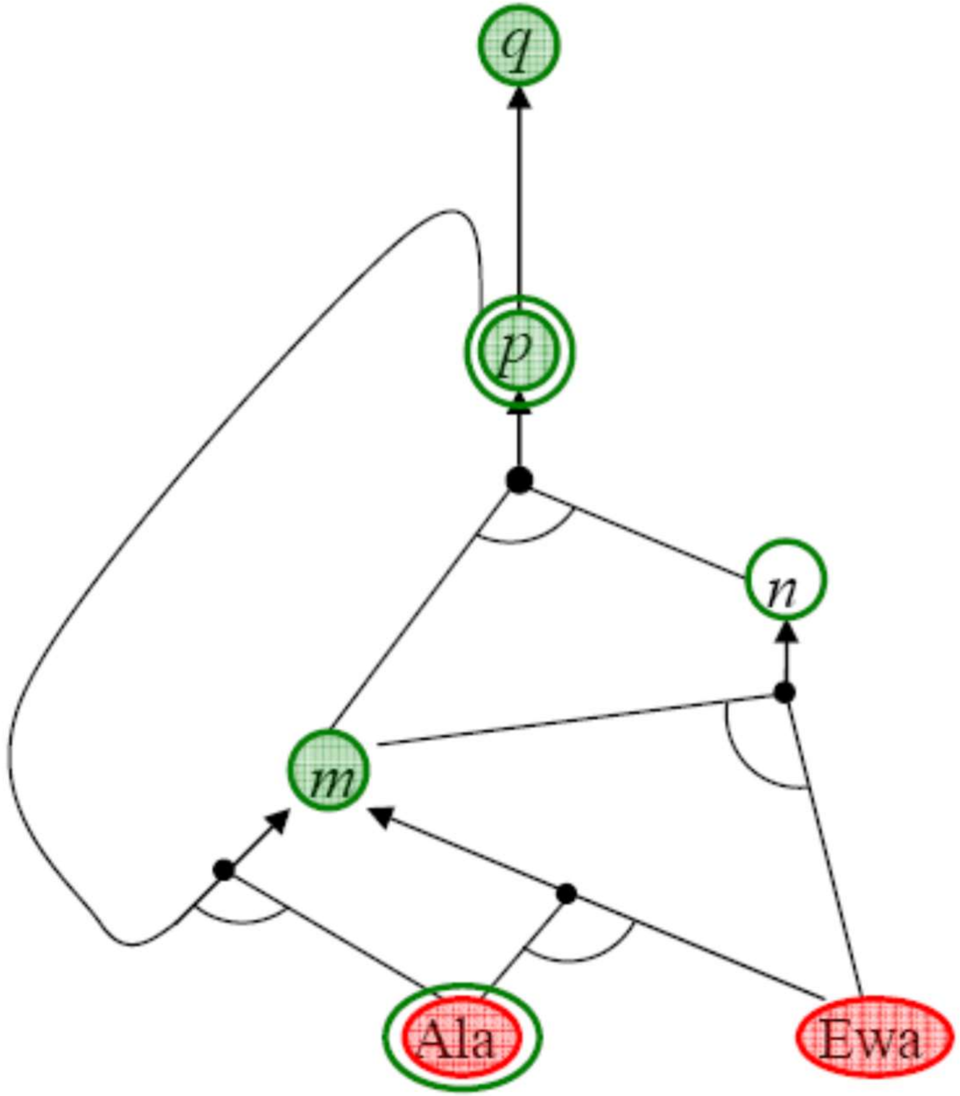


Przykład wnioskowania „wstecz” (2)

(3) Sprawdź p

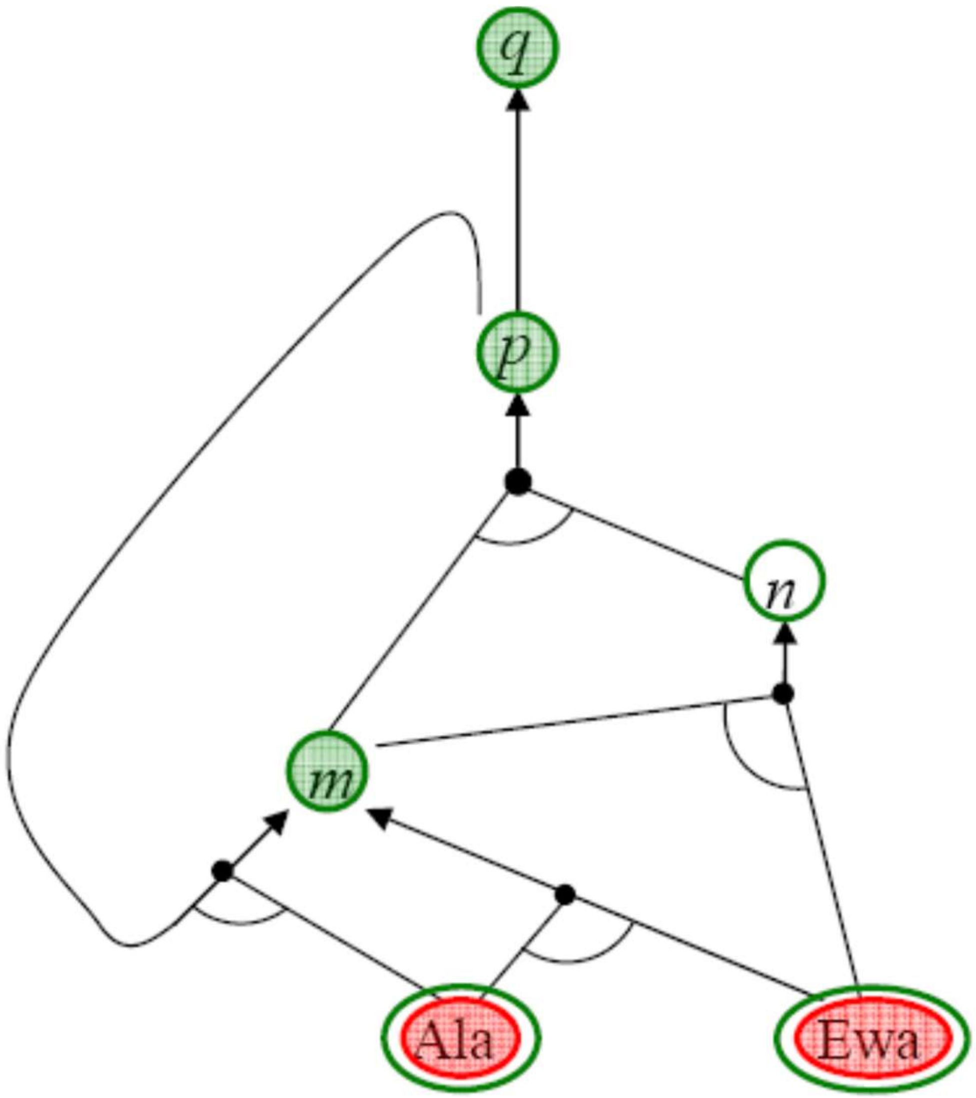


(4) Sprawdź m (1)

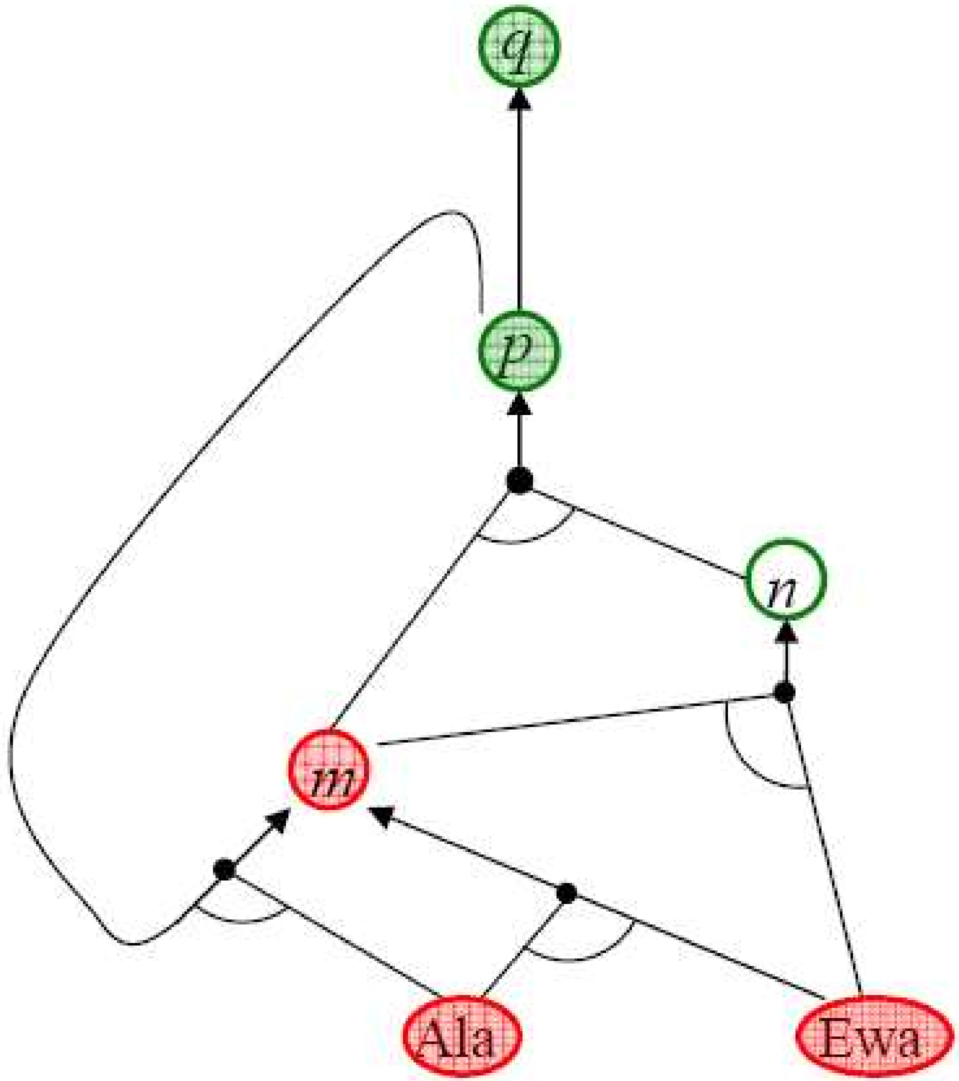


Przykład wnioskowania „wstecz” (3)

(5) Sprawdź m (2)

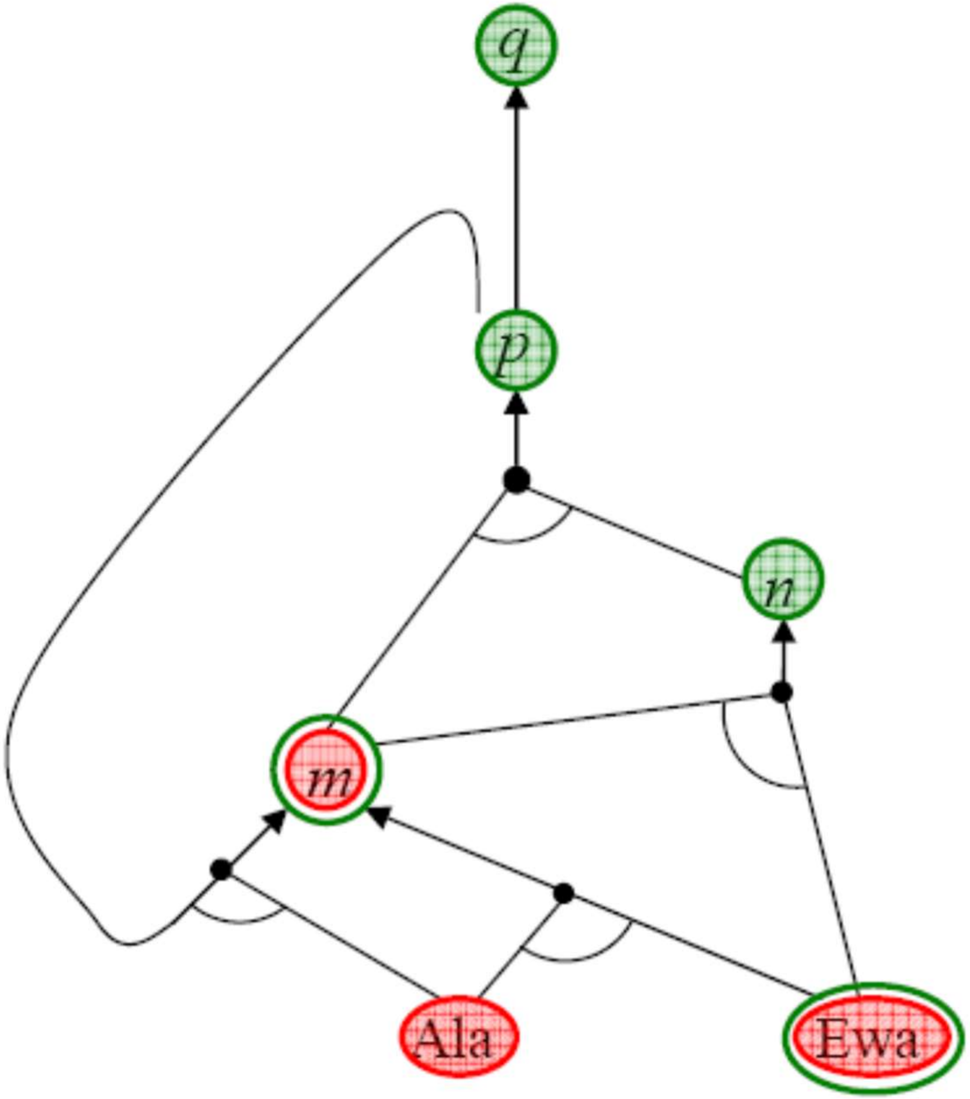


(6) Generuj m

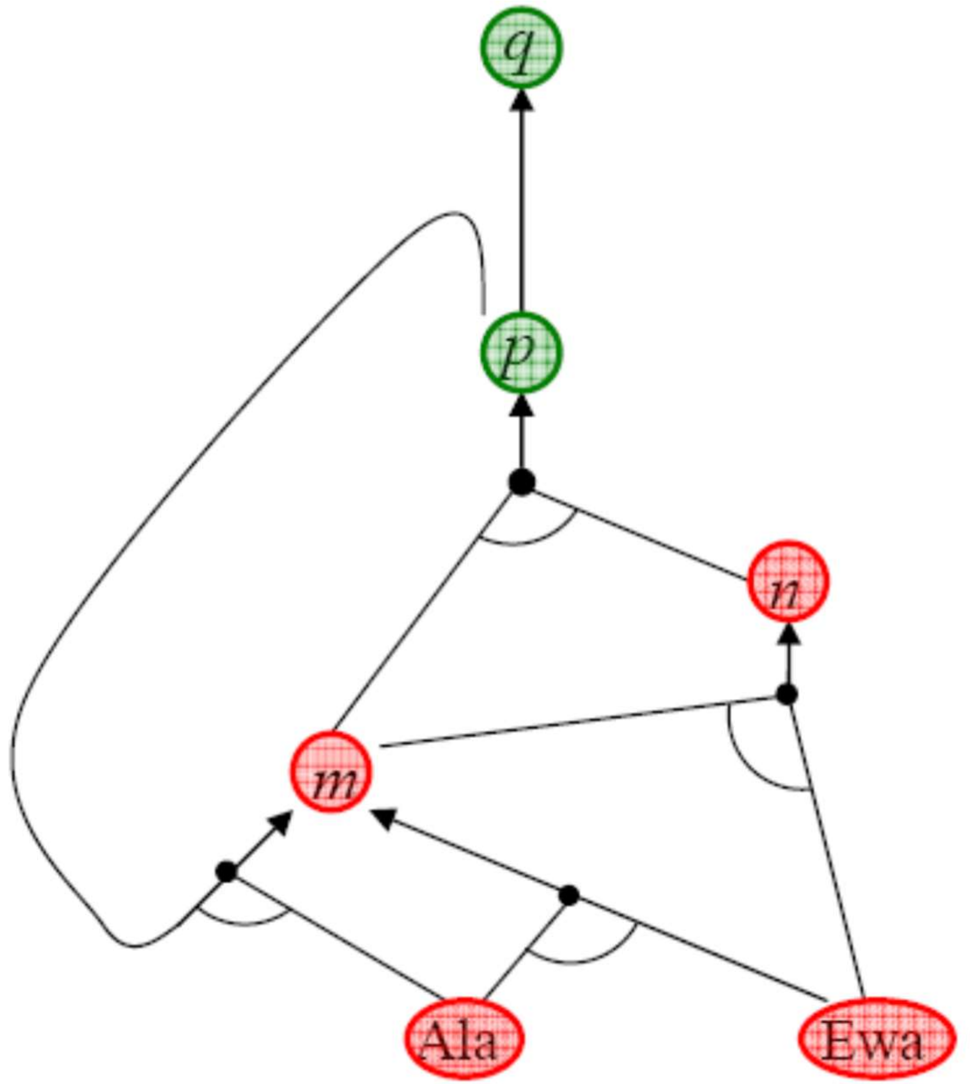


Przykład wnioskowania „wstecz” (4)

(7) Sprawdź *n*

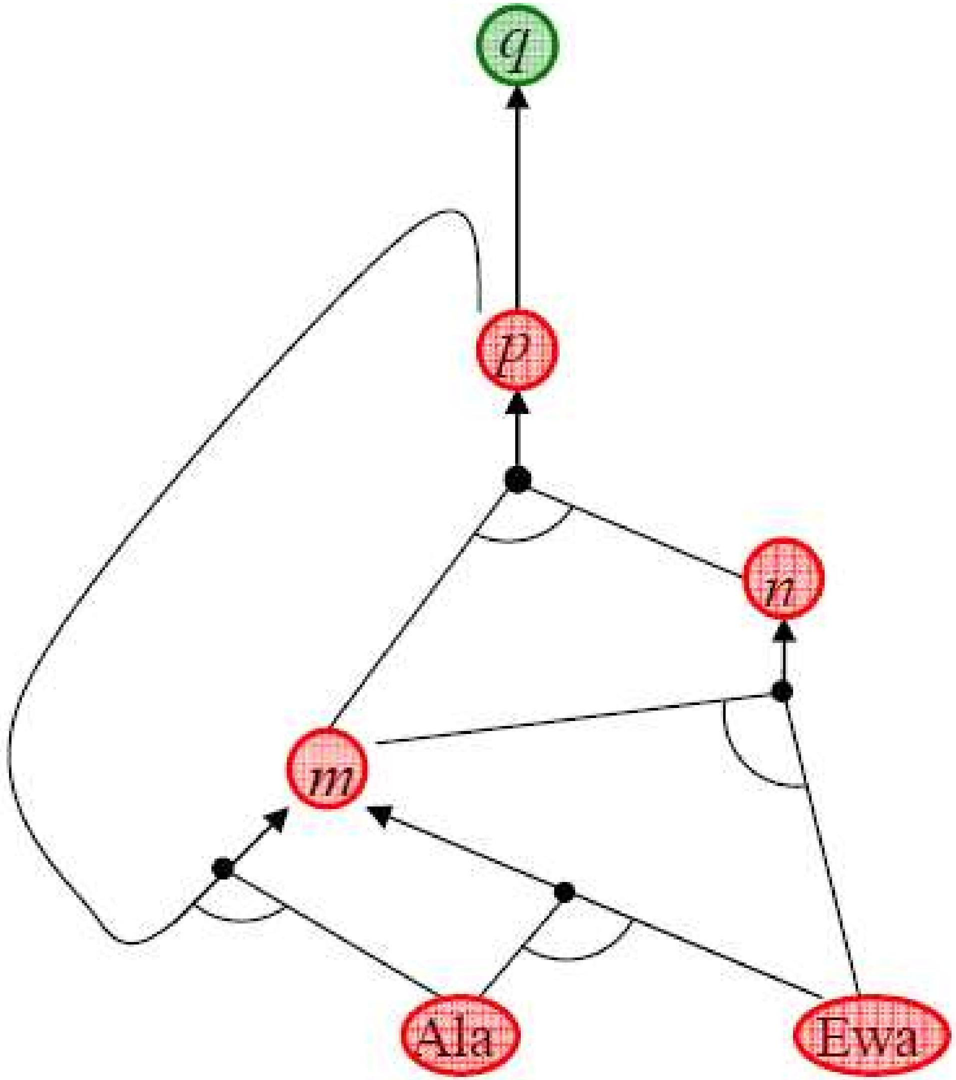


(8) Generuj *n*

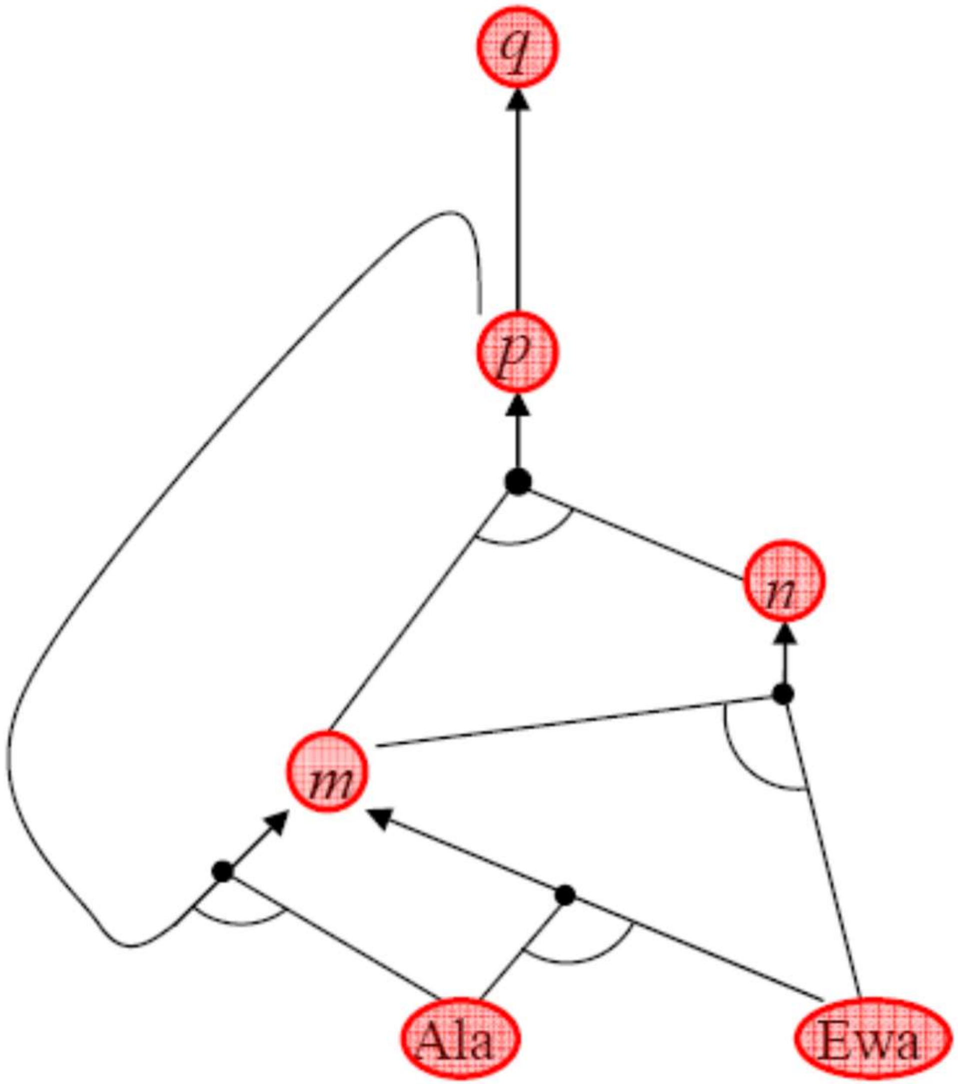


Przykład wnioskowania „wstecz” (5)

(9) Generuj p



(10) Generuj q



„W przód" a „wstecz"

- Procedura progresywna jest sterowana danymi – jest to automatyczne, „nieświadome" przetwarzanie.
  - Np. rozpoznawanie obiektów, rutynowe decyzje.
- Może wykonywać „nadmiarową" pracę, która nie zmierza bezpośrednio do celu.
- Procedura regresywna jest sterowana celem – jest to odpowiednie dla rozwiązywania zadanego problemu.
  - Np. dostarczenie odpowiedzi na pytanie: „Gdzie są moje klucze?"
- Złożoność procedury regresywnej może w praktyce być poniżej liniowej względem rozmiaru KB.

7. Jak reprezentować własności zmienne w czasie?

Wprowadzamy symbole $L_{i,j}$ dla oznaczenia, że agent w 2-wymiarowym *„świecie Wumpusa”* znajduje się w kratce o współrzędnych $[i,j]$. Wtedy możliwym akcjom odpowiadałyby zdania typu: $L_{1,1} \wedge ZwróconyWPrawo \wedge RuchWPrzód \Rightarrow L_{2,1}$

Jednak to nie prowadzi do prawidłowego wnioskowania. Po wykonaniu akcji oba zdania $L_{1,1}$ i $L_{2,1}$ będą w bazie danych uważane za prawidłowe, tymczasem już tak nie jest, gdyż świat zmienia się wraz z upływem czasu.

Jak reprezentować te zmiany w rachunku zdań? Jedynym sposobem jest odpowiednie **indeksowanie symboli**. Np.:

$$L^{\mathbf{1}}_{1,1} \wedge ZwróconyWPrawo^{\mathbf{1}} \wedge RuchWPrzód^{\mathbf{1}} \Rightarrow L^{\mathbf{2}}_{2,1}$$

$$ZwróconyWPrawo^{\mathbf{1}} \wedge ObrótWLewo^{\mathbf{1}} \Rightarrow ZwróconyWGórę^{\mathbf{2}}$$

Ograniczenia rachunku zdań

- Baza wiedzy (KB) zawiera zdania związane z „fizycznym" miejscem świata (np. kratką w „świecie Wumpusa"). Nie ma możliwości wyrażenia w zwarty sposób **wspólnej** własności wszystkich „fizycznych" miejsc.
- W celu reprezentacji akcji musimy wprowadzić osobne zdania dla każdej chwili czasu t i każdego miejsca $[x,y]$. Np. w „świecie Wumpusa" dla każdego kierunku, każdej kratki i czasu zawsze musiałoby istnieć zdanie postaci

$$L^{t}_{x,y} \wedge ZwróconyWPrawo^{t} \wedge RuchWPrzód^{t} \Rightarrow L^{t+1}_{x+1,y}$$

- Efektem jest „eksplozja" liczby zdań w bazie wiedzy przy rosnącym rozmiarze świata i liczbie wykonanych akcji.

Pytania

1. Omówić **elementy składni** rachunku zdań.
2. Wyjaśnić **semantykę** rachunku zdań.
3. Przedstawić dwa twierdzenia o **dedukcji** (poprawnym wnioskowaniu)
4. Przedstawić typowe **reguły** wnioskowania.
5. Jakie są postacie **normalne** zdań ?
6. Omówić wnioskowanie przez **rezolucję**.
7. Omówić wnioskowanie „**w przód**” i „**wstecz**”.
8. Przedstawić problem reprezentacji **własności zmiennych** w czasie i **wspólnych własności** miejsc.