Metody Inżynierii Wiedzy

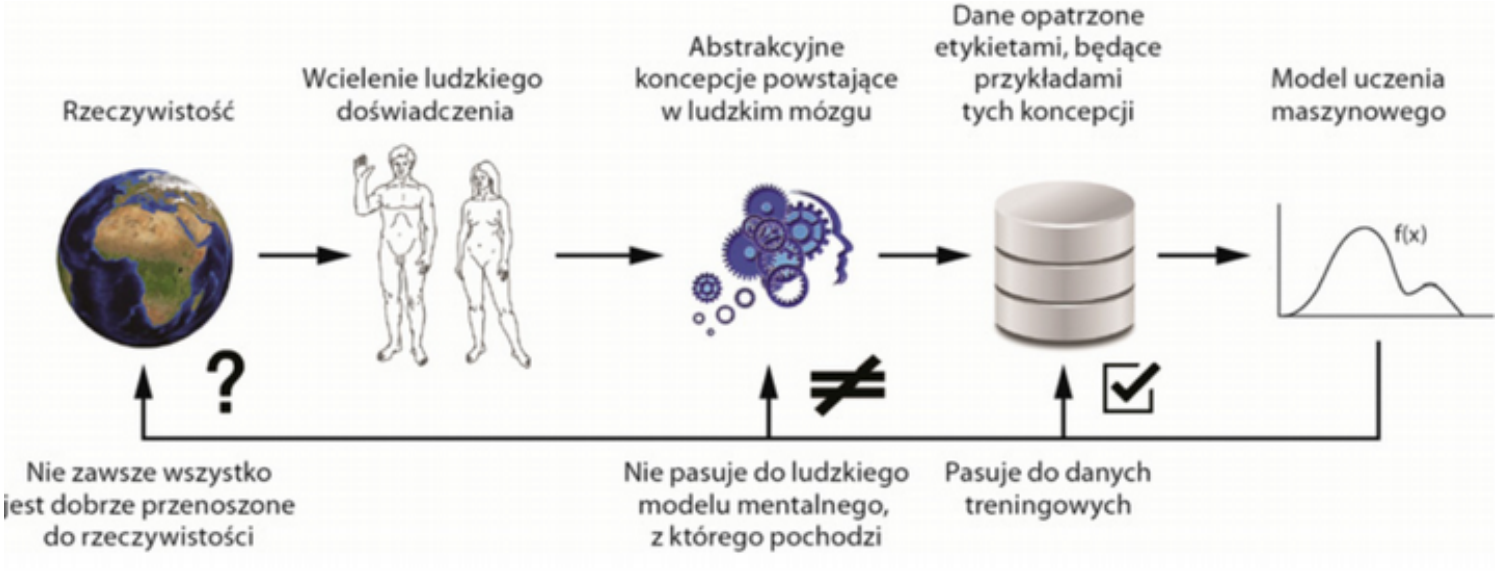
Systemy wnioskujące - wykład 11

Adam Szmigielski
aszmigie@pjwstk.edu.pl
materiały: $ftp(public) : //aszmigie/MIW$

Ograniczenia uczenia głębokiego

- Trudno jest wytrenować model uczenia głębokiego analizującego wymagania dotyczące programu i tworzącego jego kod.
- Wszystko, co wymaga logicznego myślenia, np. programowanie lub stosowanie metodologii naukowej, długoterminowe planowanie i algorytmiczne przetwarzanie danych, jest poza zasięgiem możliwości modeli uczenia głębokiego.
- Sieci neuronowe mają ogromne trudności z uczeniem się algorytmu sortowania.
- Model uczenia głębokiego jest tylko łańcuchem prostych, ciągłych przekształceń geometrycznych mapujących wektory jednej przestrzeni X na wektory innej przestrzeni Y.
- Działania większości programów nie da się sprowadzić do formy ciągłego przekształcenia geometrycznego przypisującego do siebie elementy zbiorów.

Modele uczenia maszynowego a rzeczywistość



- Sieci neuronowe nie rozumieją zadanie, które wykonują — nie rozumieją tego jak ludzie.
- Wykonują o wiele węższe zadanie, polegające na mapowaniu danych zbioru treningowego na ich etykiety punkt po punkcie.

Lokalne uogólnianie a ekstremalne uogólnianie

- Tworzenie prostych przekształceń geometrycznych mających na celu połączenie punktów znacznie odbiega od sposobu myślenia i uczenia się ludzi.
- Ludzie uczą się samodzielnie poprzez doświadczenie — nie pokazuje się im w sposób jawny przykładów treningowych.
- Ludzi i maszyny różni nie tylko przebieg procesu uczenia, ale również natura wygenerowanych reprezentacji danych.
- Umiejętności człowieka wykraczają znacznie poza bezpośrednie łączenie obserwacji z jej etykietą. W naszych umysłach powstają złożone abstrakcyjne modele bieżącej sytuacji, a także nas samych i innych ludzi.

Sztuczna inteligencja

- W celu uzyskania sztucznej inteligencji mogącej konkurować z ludzkim umysłem dysponującym możliwością logicznego myślenia i tworzenia abstrakcji musimy wymyślić rozwiązania pomagające obejść ograniczenia wynikające z prostego mapowania danych wejściowych na dane wyjściowe.
- Substytutem możliwości abstrakcyjnego modelowania różnych sytuacji i koncepcji mogą stać się programy komputerowe.

Kierunki rozwoju uczenia głębokiego

- Modele zaczną przypominać bardziej zwykłe programy komputerowe - w ten sposób uzyskamy mechanizmy logicznego myślenia i generowania abstrakcji — mechanizmy, których brak jest główną słabością bieżących modeli.
- Powstaną nowe formy uczenia maszynowego, które umożliwią uzyskanie założeń określonych w poprzednim punkcie — powstaną modele umożliwiające odejście od różniczkowalnych przekształceń.
- Modele będą wymagały mniejszego zaangażowania ze strony obsługujących je osób,
- Nastąpi zwiększenie możliwości ponownego korzystania z wcześniej wyuczonych cech i architektur,

Modele jako programy, systemy wnioskujące

- Rozwój modeli będących w stanie uzyskać uogólnienia na poziomie nie tylko lokalnym, lecz dążyć do uzyskania modeli zdolnych do tworzenia abstrakcji i logicznego myślenia,
- Obecnie wszystkie programy sztucznej inteligencji przejawiające podstawy logicznego myślenia są kodowane na sztywno przez programistów — oprogramowanie to jest oparte na algorytmach wyszukiwania i formalnej logice,
- Większość inteligencji przejawianej przez programy została zaprojektowana i ustalona na sztywno przez programistów,
- Pożąda się, aby systemy sztucznej inteligencji mogły być w pełni uczone bez żadnej ingerencji ze strony programistów.

Automatyzowane uczenie maszynowe

- Obecnie często stosuje się podstawowe systemy automatyzacji uczenia maszynowego które zajmują się dostrajaniem większości hiperparametrów modelu.
- Architektury modeli będą trenowane — nie będą definiowane ręcznie przez programistów.

Rodzaje wnioskowań

- *Dedukcja* - wniosek wynika logiczne z przesłanek,
- *Indukcja* – Wnioskowanie, polegające na wyprowadzeniu ogólnych wniosków z przesłanek, które są poszczególnymi przypadkami tych wniosków

Indukcja – nauki empiryczne

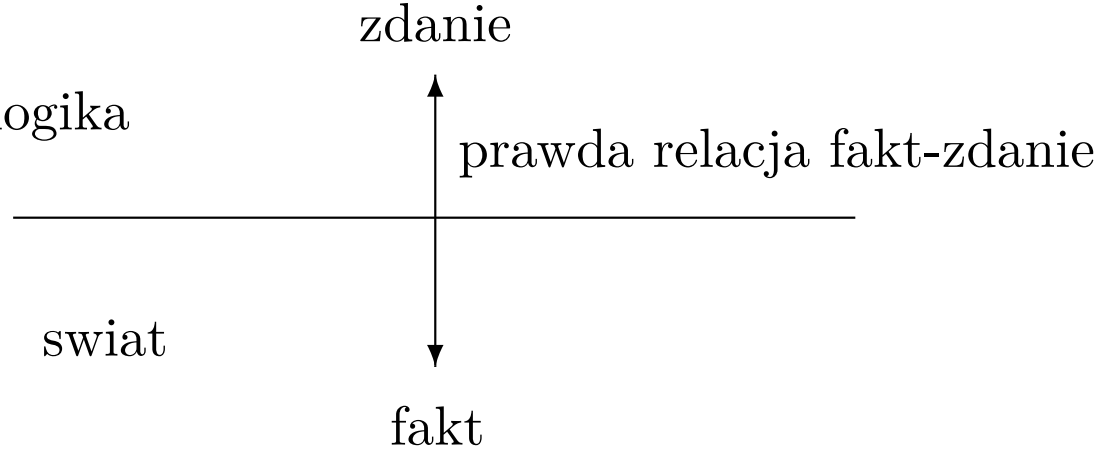
- Metoda polegająca na wprowadzeniu uogólnień na podstawie eksperymentów i obserwacji faktów, formułowaniu i weryfikacji hipotez,
- Fransis Bacon - indukcja i eksperyment to dwie skuteczne metody ustalania prawdy.

Semantyka, syntaktyka

- *Semantyka* - to dyscyplina badająca relacje pomiędzy znakami a przedmiotami, do których się one odnoszą.Semantyka zajmuje się badaniem znaczenia słów,
- *Syntaktyka* – zajmuje się składnią.

Semantyczna definicja prawdy

1. Zdanie "*a*" jest prawdziwe jeśli a jest prawdą:
2. np. Zdanie "*Jabłko jest czerwone*" jest prawdziwe gdy jabłko jest czerwone.



Tautologie

- Każde zdanie języka naturalnego oparte na schemacie tautologicznym jest zdaniem prawdziwym - nie niesie natomiast żadnej informacji:

  Jeśli "Deszcz pada lub nie pada." to nie wiadomo jaka jest pogoda.

- Oprócz prawdy faktycznej (zdania prawdziwe z uwagi na zaistniały stan rzeczy - semantyczna definicja prawdy) istnieje prawda językowa (zdania oparte na tautologicznym schemacie.)

Reguły wynikania

- Regułą wnioskowania jest dowolny zbiór par o ustalonym porządku elementów
$$< \Phi, \{\alpha\} > .$$
- Skończony i niepusty zbiór formuł Φ jest zwany *zbiorem przesłanek*,
- Jednoelementowy zbiór $\{\alpha\}$ jet zbiorem wniosków, a α jest *wnioskiem*.
- zamiast $< \Phi, \{\alpha\} >$ będziemy pisać $< \Phi, \alpha >$.

Rodzaje reguł

- Regułę, która zawiera tylko i wyłącznie wszystkie uszczegółowienia schematu podstawowego nazywać będziemy *regułą elementarną*,
- Reguły niezawodne ("zachowują tautologiczność") - ilekroć wszystkie przesłanki są tautologiami to wniosek jest tautologią.

Opis Syntaktyczny

- Metody konstrukcji rachunku logicznego, które nie odwołują się do żadnych pojęć semantycznych są metodami *syntaktycznymi*.
- Syntaktyczny opis rachunków logicznych jest zazwyczaj *systemem aksjomatycznym*.

Podejście semantyczne

- Klasyczne podejście tabelkowe (stosowane na kursach logiki w szkole średniej i później)
- np. Koniunkcja:

a	b	$a \wedge b$
0	0	0
0	1	0
1	0	0
1	1	1

Podejście syntaktyczne

- *Zbiór aksjomatów*
- *reguły wnioskowania*

Reguły wnioskowania

- Reguła podstawiania
- Reguła odrywania: (modus ponens)

$$\frac{\alpha \ , \quad \alpha \longrightarrow \beta}{\beta}$$

- np.: $\dfrac{\text{Pada deszcz. } , \quad \text{Jeśli pada deszcz to jest mokro.}}{\text{Jest mokro.}}$

Bazy wiedzy

- Zbiory faktów i zbiory reguł zapisanych w formie implikacji
- Bazy wiedzy mogą powstawać:
  - automatycznie, w oparciu o inteligentną analizę danych eksperymentalnych,
  - w wyniku interaktywnego procesu komunikowania się eksperta z interfejsem "podszytym" sztuczną inteligencją.

Wnioskowanie

- Wnioskowanie jest procesem tworzenia nowych konfiguracji symboli (reprezentujących fakty ze świata) ze starych.
- Polega na wielokrotnym stosowaniu podanych reguł do zdań zawartych w bazie wiedzy KB - ang. *knowledge base* (i do niej dodanych w trakcie wnioskowania) i kończy się gdy pożądane zdanie zostanie dowiedzione.
- *Wnioskując operuje się na reprezentacji zdarzeń a nie na samych zdarzeniach.*

Reguły wnioskowania

- Prawdziwe przesłanki prowadzą zawsze do prawdziwych wniosków,
- Proces wnioskowania polega na wielokrotnym stosowaniu podanych reguł do zdań zawartych w KB (i do niej dodanych w trakcie wnioskowania) i kończy się gdy pożądane zdanie zostanie dowiedzione.

Poprawność wnioskowania

- Powiemy, że reguła wnioskowania o postaci

$$\frac{\alpha_1, \alpha_2, \ldots, \alpha_n}{\beta}$$

  jest poprawna (niezawodna), jeśli zdanie β jest prawdziwe, wtedy gdy prawdziwe są zdania: $\alpha_1, \alpha_2, \ldots, \alpha_n$.

- Wówczas formuła

$$(\alpha_1 \wedge \alpha_2 \wedge, \ldots, \wedge \alpha_n) \Rightarrow \beta$$

  jest tautologią

- Poprawna (niezawodna) reguła wnioskowania wyklucza sytuację, gdy z prawdziwych przesłanek wynika fałszywy wniosek.

Przykłady reguł niezawodnych

- modus ponens:

$$\frac{\alpha, \alpha \Rightarrow \beta}{\beta}$$

- wprowadzania koniunkcji:

$$\frac{\alpha_1, \alpha_2, \ldots, \alpha_n}{\alpha_1 \wedge \alpha_2 \wedge, \ldots, \wedge \alpha_n}$$

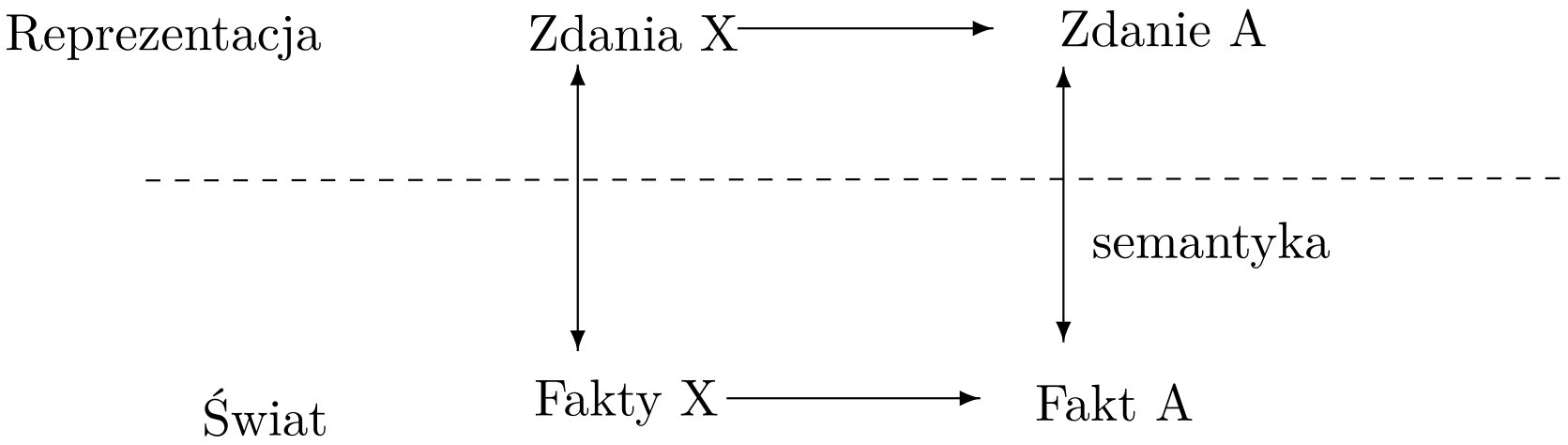
- sylogizm warunkowy:

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

- rezolucja:

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

Wynikanie



- **Wynikanie** jest związkiem pomiędzy zdaniami.
- Ze zbioru zdań X wynika zdanie A (co oznaczamy to jako $X \models A$) , wtedy gdy odzwierciedla to następstwo odpowiadających tym symbolom faktów w modelowanym świecie.

Reprezentacja informacji w systemach logicznych

- System logiczny udostępnia język formalny do reprezentacji informacji, z której można wyciągać wnioski.
- Obok składni i semantyki języka logicznego dołączamy do niego mechanizmy wynikania i wnioskowania.
- Składnia języka logiki L podaje reguły tworzenia poprawnych zdań języka
- Semantyka języka określa znaczenie zdań (formuł):
  1. Podaje znaczenie wszystkich symboli X języka L (czyli zawiera pewne odwzorowanie:
     $X \rightarrow$ elementy modelowanego świata)
  2. sposób, w jaki zdaniom można przypisać znaczenie, co z kolei pozwala określić ich wartość logiczną.

Programowanie a inżynieria wiedzy

- Programista:
  - Wybór języka programowania,
  - Programowanie,
  - Wybór kompilatora,
  - Korzystanie z programu.
- Inżynier wiedzy:
  - Wybór logiki,
  - Budowanie bazy wiedzy,
  - Wybór reguł wnioskowania,
  - Wnioskowanie.
- $\Longrightarrow$ Budowanie bazy wiedzy,
- $\Longrightarrow$ Wybór reguł wnioskowania.

Budowanie bazy wiedzy

- Ontologia dziedziny:
  - do jakich faktów i obiektów ze świata powinny odnosić się zdania w KB,
  - jakie powinny być między nimi relacje.
- Podział na obiekty i relacje między nimi ustala obserwator - inżynier wiedzy.

Budowanie bazy wiedzy

- Rozpozanie dziedziny.
- Baza wiedzy ma dwóch potencjalnych użytkowników:
  1. ludzi oraz,
  2. reguły wnioskowania.
- Nazwy predykatów (obiektów i relacji między obiektami) powinny być „zrozumiałe” dla reguł wnioskowania.
- Wiedza powinna być reprezentowana na możliwie ogólnym poziomie.
- Nazwy powinny być zrozumiałe dla procedur wnioskujących.

Modele w logice

- Modele w logice to formalnie zdefiniowane światy, względem których można określać to co jest prawdziwe a co nie.
- Model dla zbioru zdań X to każdy świat, w którym prawdziwe są wszystkie zdania ze zboru X.
- Zdanie A wynika ze zbioru zdań X, co zaznaczamy $X \models A$, jeśli A jest prawdziwe w każdym modelu dla X.
- Odnosząc to stwierdzenie do bazy wiedzy, która zawiera jedynie zdania prawdziwe w obserwowanym świecie mówimy, że z bazy wiedzy KB wynika zdanie α wtedy i tylko wtedy, gdy α jest prawdziwe dla wszystkich modeli zdań zawartych w KB (oznaczamy, $KB \models \alpha$).

Klauzula Horna

- Klauzula Horna jest wyrażeniem postaci:

$$\alpha_1 \wedge \alpha_2 \wedge, \ldots, \wedge \alpha_n \rightarrow \beta$$

- lub równoważna:

$$\neg\alpha_1 \vee \neg\alpha_2 \vee, \ldots, \vee \neg\alpha_n \vee \beta$$

Uniwersalny kwantyfikator

jest ogólnej postaci:

- $$\forall < zmienne >< formuły >$$

  np. Zdanie „Każdy człowiek jest śmiertelna” zapiszemy z użyciem kwantyfikatora jako: $\forall_x(czlowiek(x) \wedge smiertelny(x))$.

- Wartościowanie formuły z kwantyfikatorem ogólnym: formuła $\forall_x P$ jest prawdziwa w modelu M wtedy i tylko wtedy gdy P jest prawdziwe dla x wartościowanego dowolnym obiektem w tym modelu.

Egzystencjalny kwantyfikator

jest ogólnej postaci:

- $$\exists < zmienne >< formuły >$$

  np. Zdanie "Jakiś człowiek jest filozofem" zapiszemy jako:

  $\exists_x(czlowiek(x) \wedge jestfilozofem(x))$.

  Formuła $\exists_x P$ jest prawdziwa w modelu M wtedy i tylko wtedy gdy P jest prawdziwe dla x wartościowanego jakimś obiektem modelu.

Własności kwantyfikatorów:

- Przemienność kwantyfikatorów ogólnych:
  $\forall_x \forall_y$ jest taka sama jak $\forall_y \forall_x$,
- Przemienność kwantyfikatorów egzystencjalnych:
  $\exists_x \exists_y$ jest taka sama jak $\exists_y \exists_x$,
- $\forall_x \exists_y$ nie jest taka sama jak $\exists_y \forall_x$

Eliminacja kwantyfikatora uniwersalnego

- Każde wartościowanie formuły związanej uniwersalnym kwantyfikatorem wynika z tej formuły. Możemy to zapisać w postaci reguły:

$$\frac{\forall_v \alpha}{SUBST(\{v \backslash g\}, \alpha)}$$

  dla każdej zmiennej v i termu g.

- Tym samym kwantyfikator uniwersalny nie nakłada ograniczeń na wartościowanie związanej nim zmiennej.

- Po uprzednim unikalnym przemianowaniu zmiennych i po eliminacji ewentualnych kwantyfikatorów egzystencjalnych, opuszczamy kwantyfikatory uniwersalne.

Skolemizacja - eliminacja kwantyfikatorów

Eliminacja egzystencjalnego kwantyfikatora nosi nazwę skolemizacji formuły. Rozróżnimy tu dwa przypadki skolemizacji:

1. W formule występuje kwantyfikator uniwersalny poprzedzający kwantyfikator egzystencjalny,
2. W formule nie występuje kwantyfikator uniwersalny poprzedzający kwantyfikator egzystencjalny

Eliminacja kwantyfikatora egzystencjalnego nie poprzedzonego żadnym kwantyfikatorem uniwersalnym

Polega na zastosowaniu następującej reguły wnioskowania:

- Dla każdej formuły α, zmiennej v i pewnego symbolu stałej K, który nie występuje nigdzie indziej w bazie wiedzy, zachodzi:

$$\frac{\exists_v \alpha}{SUBST(\{v \backslash K\}, \alpha)}$$

- Eliminacja kwantyfikatora wiąże się z jednoczesnym przemianowaniem zmiennej v w nim związanej na pewną unikalną stałą K.

Przykład eliminacja kwantyfikatora egzystencjalnego nie poprzedzonego żadnym kwantyfikatorem uniwersalnym

- Dana jest formuła:

$$\exists_x mezczyzna(x) \wedge przodek(x, tomek)$$

- Formuła po eliminacji kwantyfikatora:

$$mezczyzna(K) \wedge przodek(K, tomek)$$

gdzie K jest nowym symbolem stałej zwanej stałą Skolema - nie występowało wcześniej w KB

Eliminacja kwantyfikatora egzystencjalnego poprzedzonego kwantyfikatorem uniwersalnym

- Jeśli kwantyfikator egzystencjalny formuły poprzedzony jest kwantyfikatorem uniwersalnym dla pewnej zmiennej x to w miejsce zmiennej za v podstawiamy unikalny symbol funkcji zwanej **funkcją Skolema o parametrze** x.
- Wyrażamy to w postaci reguły wnioskowania jako:

$$\frac{\forall_x \exists_v \alpha}{SUBST(\{v \backslash F(x)\}, \alpha)}$$

gdzie $F(x)$ jest funkcją skolema,

Przykład eliminacja kwantyfikatora egzystencjalnego poprzedzonego kwantyfikatorem uniwersalnym

- Każdy ma serce:
  $\forall_x Osoba(x) \Longrightarrow \exists_y Serce(y) \wedge Posiada(x, y)$
- Jeżeli zamienimy y na H, to otrzymamy:
  $\forall_x Osoba(x) \Longrightarrow Serce(H) \wedge Posiada(x, H)$, czyli, że każdy ma to samo serce.
- Wprowadzamy funkcję $F(x)$ (f. Skolema), "która zwraca serce danej osoby":
  $\forall_x Osoba(x) \Longrightarrow Serce(F(x)) \wedge Posiada(x, F(x))$

Kwantyfikatory - eliminacja

- Z $\forall_x$ $Lubi(x, Lody)$ za pomocą podstawienia $\{x/Ola\}$ można wywnioskować, że $Lubi(Ola, Lody)$.
- Z $\exists_x$ $Zabić(x, Ofiara)$ za pomocą podstawienia $\{x/Morderca\}$ można wywnioskować, że $Zabić(Morderca, Ofiara)$, jeżeli $Morderca$ nie występuje nigdzie indziej w KB
- $\exists_x$ $Ojciec(x, Jan)$ (tzn. „Jan ma ojca”)
- kwantyfikator $\exists$ można wyeliminować poprzez zamianę x jedynie na zmienną nie występującą jeszcze w KB
- zamiana x na Jan powoduje, że w KB pojawia się stwierdzenie, że Jan jest swoim ojcem ($Ojciec(Jan, Jan)$)

Metody wnioskowania automatycznego

- Wnioskowanie progresywne wprzód,
- Wnioskowanie wstecz,
- Algorytm rezolucji.

Procedura wnioskowanie progresywne wprzód

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.

Przykład wnioskowania wprzód

- Dane są zdania w KB

  $p \Rightarrow q$

  $m \wedge n \Rightarrow p$

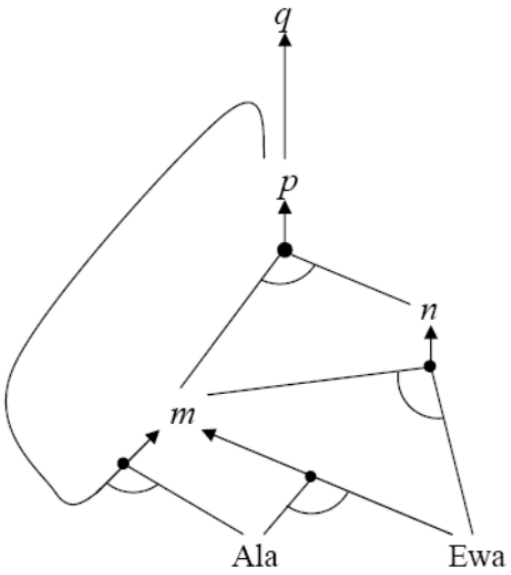
  $Ewa \wedge m \Rightarrow n$

  $Ala \wedge p \Rightarrow m$

  $Ala \wedge Ewa \Rightarrow m$

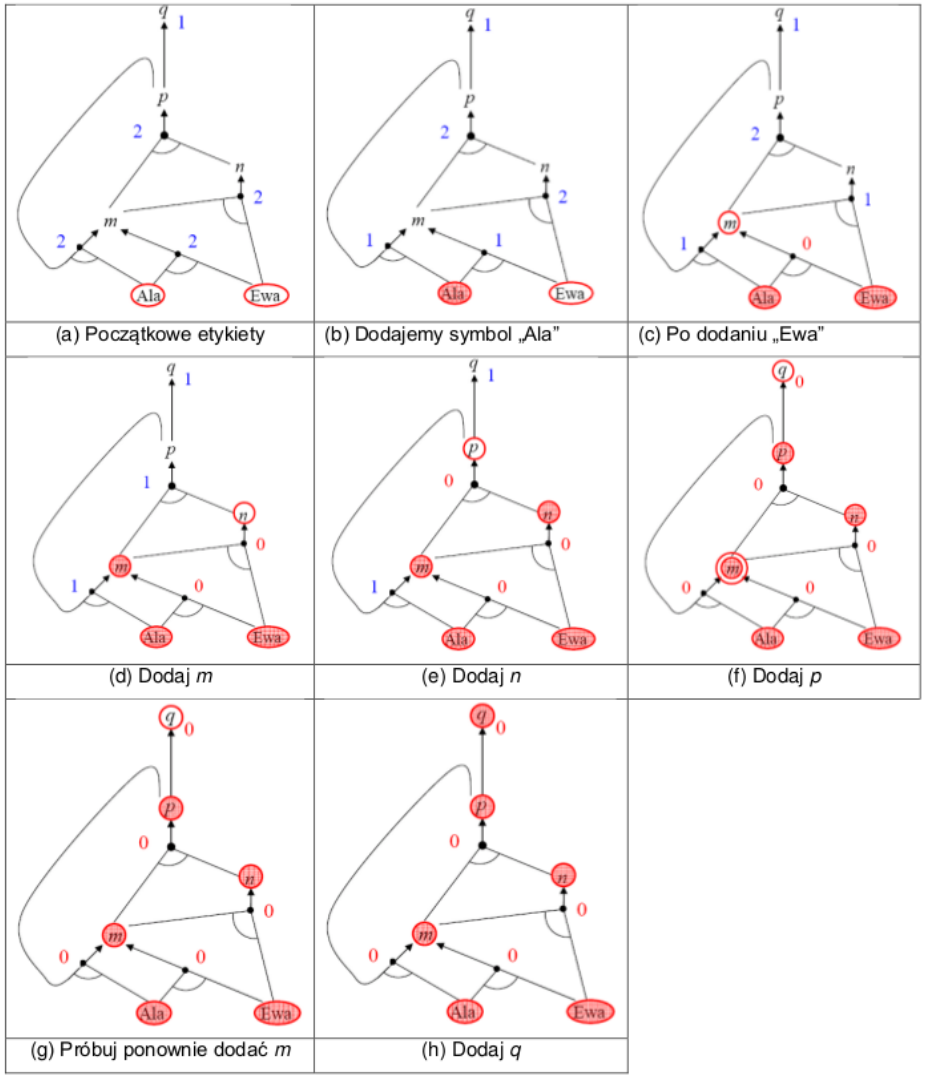
  Ala

  Ewa



- Każdy węzeł typu „I” posiada etykietę – odpowiada ona liczbie warunków w poprzedniku reguły pozostałych jeszcze do spełnienia.

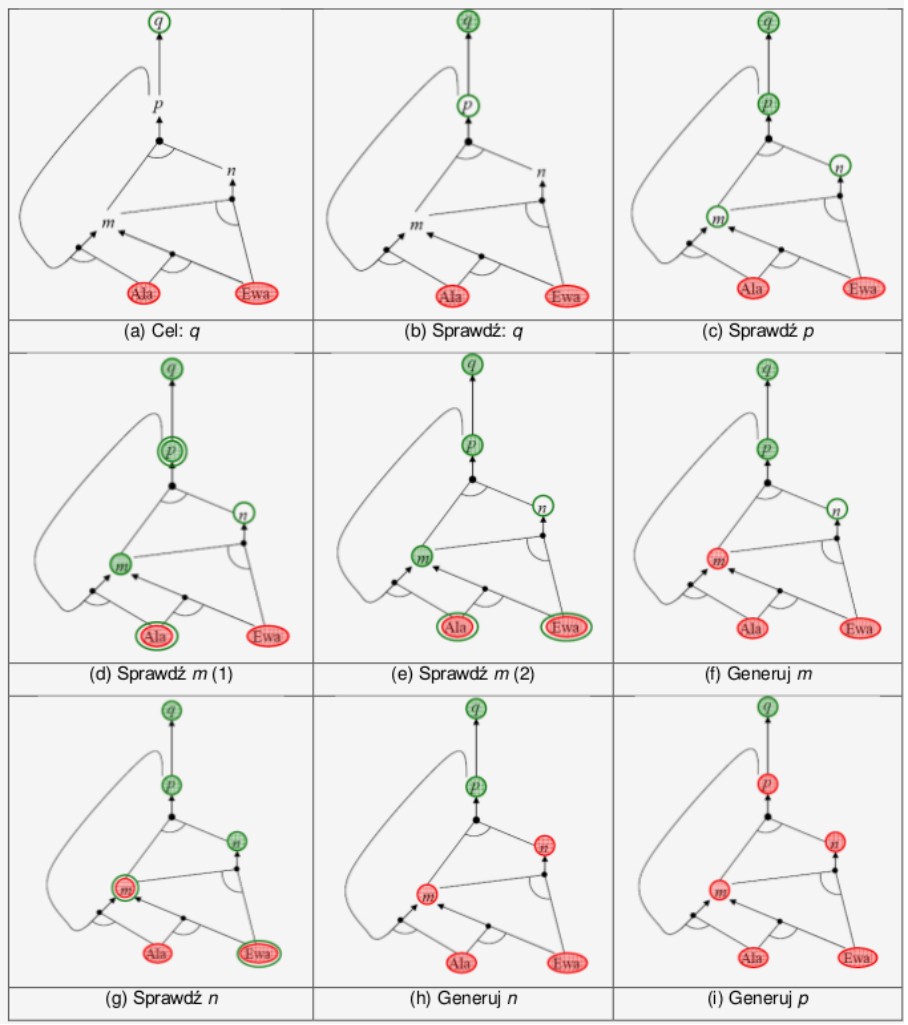
Przykład wnioskowania wprzód cd.

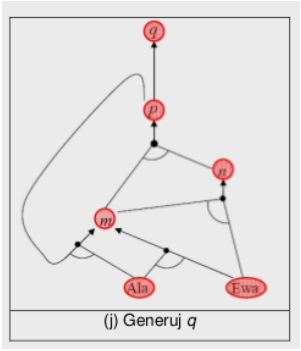


Procedura wnioskowania wstecz

1. Funkcja rozpoczyna od zdania zapytania (celu) q.
2. Aby sprawdzić prawdziwość q procedura sprawdza, czy q już występuje w KB a jeśli nie, to sprawdza czy istnieje przynajmniej jedna implikacja wyprowadzająca zdanie q. Jeśli tak, to literały stanowiące warunek tej implikacji stają się "podcelami" i rekurencyjne badana będzie ich prawdziwość z punktu widzenia aktualnego modelu KB,
3. Unikanie zapętleń: procedura sprawdza, czy aktualny "podcel" nie znajduje się już na stosie wygenerowanych "podcelów".
4. Unikanie powielania przejść: sprawdza, czy nowy "podcel" został już sprawdzony i pokazano to, czy jest prawdziwy lub fałszywy.

Przykład wnioskowania wstecz





Rezolucje

- Dla INF: $\neg A \Longrightarrow B, B \Longrightarrow C \vdash \neg A \Longrightarrow C$
- Dla CNF: $A \vee \neg B, C \vee B \vdash A \vee C$

Rezolucja - procedura

1. Doprowadzamy bazę wiedzy do formuły normalnej,
2. Stosujemy rezolucję.

Doprowadzenie do formuły normalnej

1. eliminacja implikacji:
   $p \Longrightarrow q$ zastępujemy $\neg p \vee q$
2. przesunięcie negacji:
   $\neg(p \vee q)$ zastępujemy $(\neg p \wedge \neg q)$
   $\neg(p \wedge q)$ zastępujemy $(\neg p \vee \neg q)$
3. $(\neg \forall_x p)$ zastępujemy $(\exists_x \neg p)$
   $(\neg \exists_x p)$ zastępujemy $(\forall_x \neg p)$
4. $\neg\neg p$ zastępujemy p
5. $\neg A \vee \neg A \vee B$ zastępujemy $\neg A \vee B$
6. eliminacja kwantyfikatorów szczegółowych (Skolemizacja)

MIW PROLOG projekt

1. Sporządź mapę mieszkania lub jego części (np. pokoju, pokoju wraz z korytarzem). Na mapę nanieś istniejące meble, sprzęty itp.
2. Zaproponuj język opisujący zaistniałą na planie sytuację (wybór faktów).
3. Zaproponuj definicję pojęć przestrzennych opisujących związki przestrzenne, np. pomiędzy, obok, na itp.
4. Wprowadź elementy dynamiczne umożliwiające dodawanie nowych lub zmianę istniejących faktów.
5. Zaproponuj zasady nawigacji, umożliwiające poruszanie się w opisywanej przestrzeni. Zrealizuj konkretne zadanie nawigacji - np. przemieszczenie się blisko telewizora.

6. Elementy, które powinny znaleźć się w projekcie:
   - Termy, termy złożone (fakty),
   - Klauzule (definicje nowych pojęć),
   - Elementy dynamiczne (dodawanie i usuwanie termów w trakcie działania programu),
   - Rekurencja,
   - Unifikacja, cięcie,
   - Listy (operacje na listach).

Elementy dodatkowe

1. Integracja z innymi językami programowania (C, C++, Java).
2. Istnieje możliwość wyboru indywidualnych projektów (np. implementacja do gier) po wcześniejszym uzgodnieniu z osobą prowadzącą ćwiczenia.