Sztuczna inteligencja

Systemy oparte na wiedzy

Joanna Kołodziejczyk

2024

Plan wykładu

Wprowadzenie do Systemów Opartych na Wiedzy

Definicja i Początki:

- Sztuczna inteligencja od lat 50-tych XX wieku.
- Systemy oparte na wiedzy (KBS) jako metoda naśladowania ludzkiej inteligencji i rozumowania.

Zastosowania:

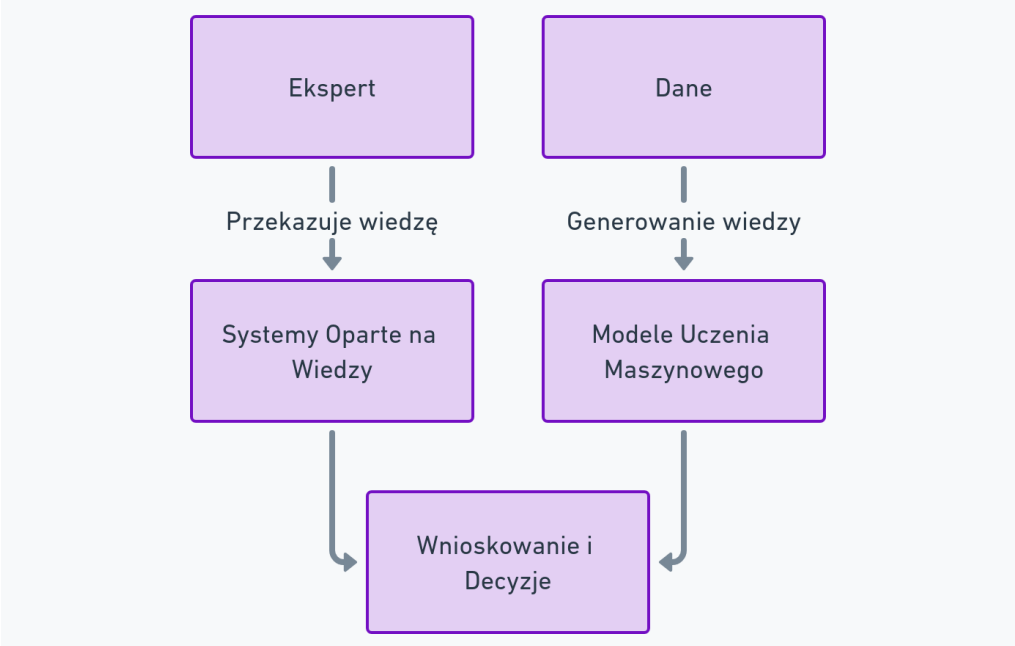
- Przetwarzanie i wykorzystanie wiedzy do podejmowania decyzji.
- Rozwiązywanie złożonych zadań: wnioskowanie, uczenie się, planowanie, „rozumienie”, język naturalny.

Kluczowe Cechy Systemów Opartych na Wiedzy

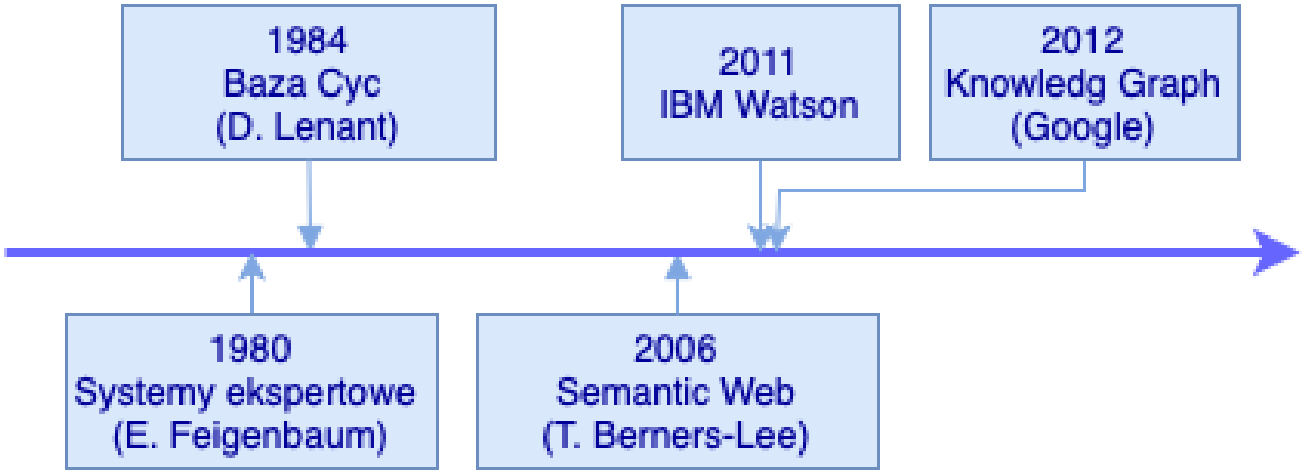
Struktura Wiedzy:

- Sformalizowana postać wiedzy.
- Wiedza pochodząca od specjalistów w danej dziedzinie.

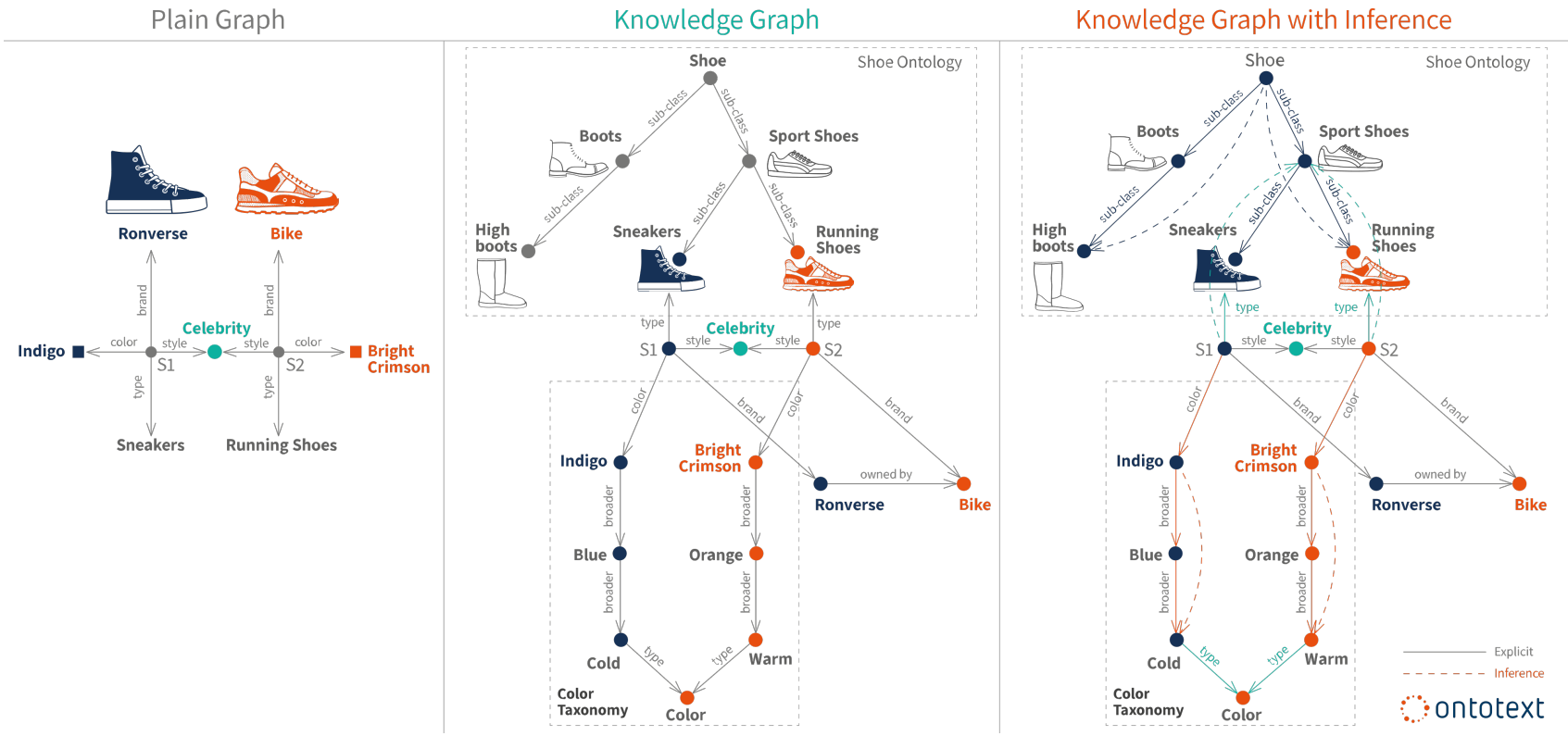
Porównanie z Modelami Uczenia Maszynowego:



Rozwój systemów z wiedzą na przestrzeni lat



Grafy wiedzy



Reprezentacji Wiedzy

Kluczowe zagadnienia badawcze:

- Reprezentacja wiedzy jako główne wyzwanie w systemach z wiedzą.
- Efektywne przetwarzanie wiedzy do symulacji rozumowania.

Logika Predykatów Pierwszego Rzędu w reprezentacji

- Podstawa dla programowania logicznego.
- Przykład: Język Prolog - definiowanie faktów i reguł.

Remedium na ograniczenia logiki dwuwartościowej

Logika Rozmyta

- Obsługa częściowej prawdy i niuansów języka naturalnego.

Systemy Probabilistyczne

- Sieci Bayesowskie jako przykład modelowania probabilistycznego.

Definicja Wiedzy

Wiedza: Zbiór informacji, umiejętności i przekonań nabytych przez doświadczenie, edukację lub odkrywanie.

Zastosowania:

- Rozumienie, przewidywanie, wyjaśnianie różnych zjawisk.
- Różne konteksty i definicje w zależności od dziedziny.

Perspektywy na Wiedzę

Filozofia:

- Sokrates: Wiedza jako pojęcia odzwierciedlające realny świat.
- Platon: Wiedza jako uzasadnione przekonanie zgodne z rzeczywistością.

Psychologia Poznawcza:

- Wiedza jako informacje w sieciach semantycznych.

Zarządzanie Wiedzą:

- Nonaka i Takeuchi: Wiedza jako proces tworzenia znaczeń przez interakcję społeczną.

Perspektywy na Wiedzę w Informatyce i Systemach Ekspertowych

Informatyka:

- Wiedza jako dane przetworzone do formy decyzyjnej.

Systemy Ekspertowe:

- Wiedza jako zbiór reguł i heurystyk symulujących rozumowanie eksperta.

Jak różne mogą być definicje?

„Czym jest Wiedźmin?”

Jak różne mogą być definicje?

„Czym jest Wiedźmin?”

1. to cykl powieści Andrzeja Sapkowskiego

Jak różne mogą być definicje?

„Czym jest Wiedźmin?”

1. to cykl powieści Andrzeja Sapkowskiego
2. gra komputerowa

Jak różne mogą być definicje?

„Czym jest Wiedźmin?”

1. to cykl powieści Andrzeja Sapkowskiego
2. gra komputerowa
3. bohater książki, mutant wyszkolony by walczyć z potworami

Wprowadzenie do Reprezentacji Wiedzy

Reprezentacja wiedzy

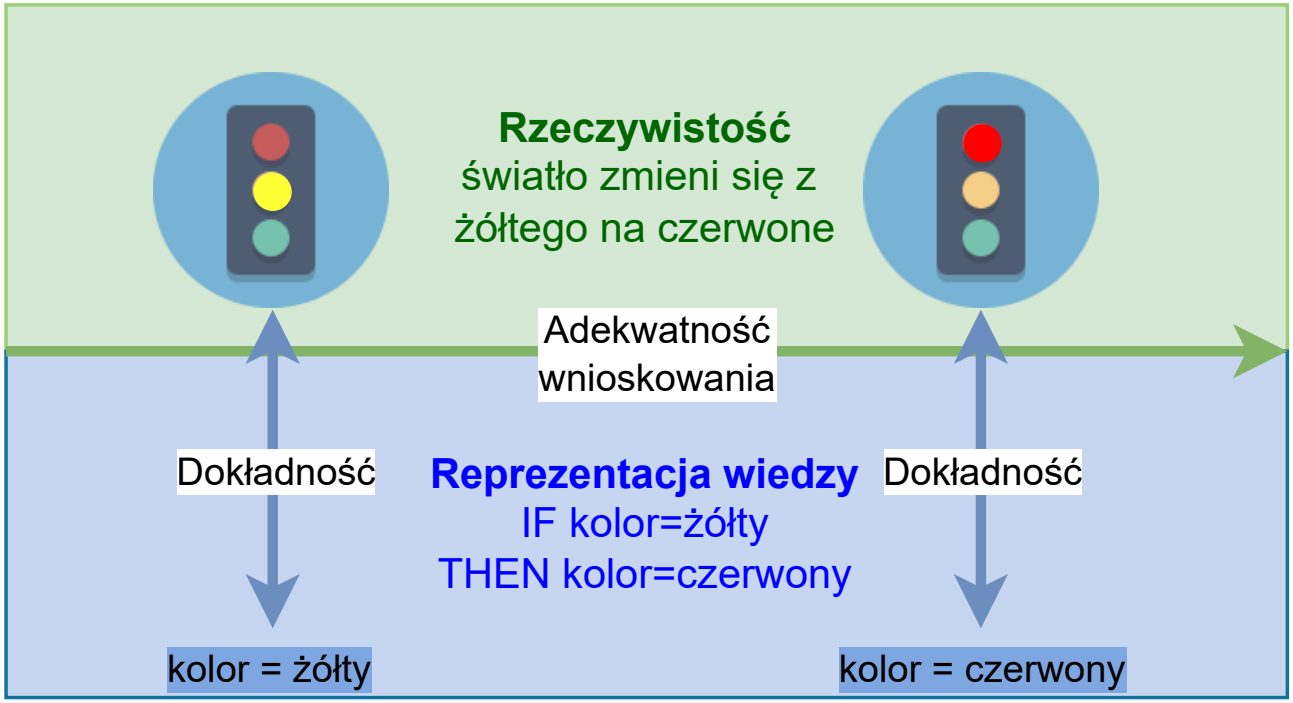
stanowi fundamentalne pojęcie w sztucznej inteligencji, ponieważ umożliwia systemom AI rozumienie i przetwarzanie informacji. Jej celem jest, aby wiedzę ludzką przekształcić w formę, którą komputery mogą przetwarzać, do symulowania inteligentnego zachowania.

Właściwości Dobrego Systemu Reprezentacji Wiedzy

Dobry system reprezentacji wiedzy musi posiadać takie właściwości jak:

1. **Dokładność reprezentacji**: Powinien pozwalać reprezentować dowolny koncept, relację i obserwację ze świata rzeczywistego.
2. **Adekwatność wnioskowania**: Powinien być w stanie manipulować strukturami reprezentacyjnymi w celu wytworzenia nowej wiedzy w taki sposób, by powstały wynik odpowiadał temu co wydedukuje człowiek.

Przykład Reprezentacji Wiedzy

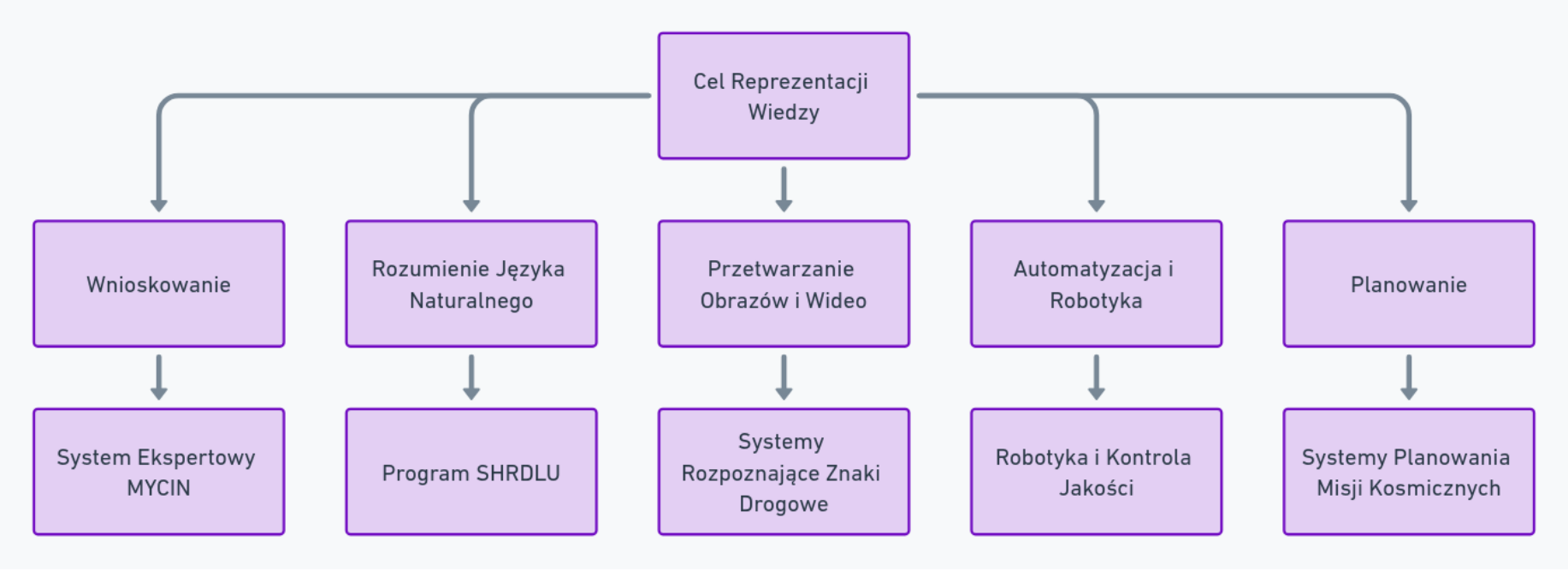


Cel Reprezentacji Wiedzy

Katalog zadań, które można realizować z wykorzystaniem systemów opartych na wiedzy, które wykorzystują pewną formę reprezentacji można ująć w postaci następującej listy:

1. **Wnioskowanie**
2. **Rozumienie języka naturalnego**
3. **Przetwarzanie obrazów i wideo**
4. **Automatyzacja i robotyka**
5. **Planowanie**

Przykłady Zastosowań



Podsumowanie

Systemy oparte na wiedzy rozwijają się od ponad 70 lat i w chwili obecnej w niewielkim stopniu spotyka się „czyste" systemy przetwarzania symbolicznego, natomiast często łączy się je z innymi podejściami takimi jak modelowanie danych, optymalizacja czy inne podejścia do podejmowania decyzji.

Plan wykładu

Wprowadzenie do Logiki Predykatów

W sztucznej inteligencji od dawna stosowano operacje logiczne do przetwarzania informacji zawartych w deklaratywnych bazach wiedzy. Początki tego podejścia sięgają 1958 roku, kiedy John McCarthy zalecał je do prezentowania informacji o specyficznej dziedzinie problemu.

Charakterystyka:

- Wykorzystanie języka deklaratywnego, często analogicznego do języka logiki predykatów pierwszego rzędu.
- Użycie reguł logiki do wyciągania wniosków z zapisanej wiedzy.

Zastosowanie Logiki Predykatów

Jeżeli logika predykatów pierwszego rzędu jest używana do rozwiązania „rzeczywistych" problemów, to wymaga eksperckiej znajomości danej dziedziny.

Wymagania:

- Ekspercka wiedza w danej dziedzinie.
- Zastosowanie w rozwiązywaniu konkretnych problemów.

1. Consistency

Spójność

System logiczny jest spójny, jeśli nie prowadzi do sprzecznych wniosków.

W logice predykatów pierwszego rzędu możliwe sprzeczności wynikają tylko z niespójnych aksjomatów; system jest spójny, jeśli jego aksjomaty są spójne.

2. Completeness

Zupełność

System jest zupełny, jeśli każde prawdziwe zdanie można w nim udowodnić.

System oparty o logikę predykatów jest zupełna w sensie teoretycznym, zgodnie z twierdzeniem Gödla o kompletności.

3. Soundness

Poprawność

System jest poprawny, jeśli każde udowodnione w nim zdanie jest prawdziwe.

Każde zdanie udowodnione w logice predykatów jest prawdziwe w każdym jej modelu.

4. Decidability

Jednoznaczność

Możliwość ustalenia prawdziwości każdego zdania za pomocą skończonego procesu.

W logice predykatów nie wszystkie zdania są możliwe do udowodnienia ze względu na nieskończoną liczbę potencjalnych interpretacji.

5. Expressiveness

Ekspresyjność

Zdolność wyrażania bogatego zestawu koncepcji.

Logika predykatów pozwala na wyrażanie szerokiego zakresu zdań dotyczących obiektów i ich relacji.

6. Deductibility

Możliwości dedukcyjne

Zdolność do wyprowadzania wniosków na podstawie dostępnych informacji i reguł.

Logika predykatów umożliwia dedukcję bardziej złożonych struktur dzięki kwantyfikatorom i zmiennym.

Symbole w Logice Predykatów

- **Zmienne:** *x, y, z* - zmienne zaczynają się od małej litery.
- **Stałe:** *Geralt, Proba_Traw, Nilfgard, 15* - stałe zaczynają się od wielkiej litery lub cyfry.
- **Funkcje:** *opiekun[2], znak[2], ryzyko[1]* - funkcje zwracają wartość, zaczynają się od małej litery.
- **Predykaty:** *Kocha[2], Sasiad[2], Driada[1]* - predykaty zaczynają się wielką literą.
- **Kwantyfikatory:** ogólny $\forall$ i szczególny $\exists$.
- **Łączniki logiczne:** koniunkcja $\wedge$, dysjunkcja $\vee$, negacja $\neg$, implikacja $\rightarrow$, równoważność $\leftrightarrow$.
- **Symbol równości termów:** $=$.
- **Pomocnicze:** nawiasy okrągłe () i przecinki ,.

Formuła Atomowa

Formuła atomowa: Driada(x), gdzie Driada jest symbolem predykatu, a x jest zmienną.

Negacja

Negacja: $\neg \text{Driada}(x)$;

Koniunkcja

Koniunkcja: $\mathrm{Driada}(x) \wedge \mathrm{Elf}(x)$;

Dysjunkcja

Dysjunkcja: $\text{Driada}(x) \vee \text{Elf}(x)$;

Implikacja

Implikacja: $\text{Driada}(x) \rightarrow \text{Elf}(x)$;

Równoważność

Równoważność: $\text{Driada}(x) \leftrightarrow \text{Elf}(x)$;

Kwantyfikacja Uniwersalna

Kwantyfikacja uniwersalna: $\forall x(\text{Driada}(x))$;

Kwantyfikacja Egzystencjalna

Kwantyfikacja egzystencjalna: $\exists x(\mathsf{Elf}(x))$;

Równość Termów

Równość termów: Protektor(Ciri) = Geralt

Przykład wyrażenia

Przykładowa formuła wykorzystująca powyższe elementy składni:

$$\forall x(\text{Elf}(x) \rightarrow \exists y(\text{Czlowiek}(y) \wedge \text{Nienawidzi}(x, y)))$$

Przykład wyrażenia

Przykładowa formuła wykorzystująca powyższe elementy składni:

$$\forall x(\text{Elf}(x) \rightarrow \exists y(\text{Czlowiek}(y) \wedge \text{Nienawidzi}(x, y)))$$

Interpretacja

Ta formuła oznacza, że dla każdego x, jeśli Elf(x) jest prawdziwe, to istnieje taki y, że Czlowiek(y) jest prawdziwe i Nienawidzi jest relacją między x i y. Interpretując zdanie potocznie można je sformułować jako: „Każdy elf nienawidzi jakiegoś człowieka”.

Wprowadzenie do Semantyki

Interpretacja, czyli przypisanie znaczenia, naturalnie przenosi nas w obszar semantyki. Semantyka opisuje jakie wartości, czyli znaczenie, przypisuje się symbolom w formułach składniowych, określając, kiedy formuła jest prawdziwa.

Interpretacja Stałych

1. **Obiekty do stałych:** Każdej stałej w formule przypisuje się konkretny obiekt z dziedziny (domeny).

Interpretacja Symboli Funkcyjnych

2. **Funkcje do symboli funkcyjnych:** Symbole funkcyjne są interpretowane jako konkretne funkcje, które przypisują obiekty do obiektów w dziedzinie.

Interpretacja Symboli Predykatów

3. **Relacje do symboli predykatów:** Symbole predykatów są interpretowane jako specyficzne relacje między obiektami.

Interpretacja Kwantyfikatorów

Kwantyfikatory są interpretowane jako odnoszące się do wszystkich obiektów w opisywanej dziedzinie (domenie) (dla kwantyfikatora uniwersalnego) lub do istnienia przynajmniej jednego obiektu (dla kwantyfikatora egzystencjalnego), który spełnia daną formułę.

Kwantyfikator Uniwersalny

Rozważmy zdanie z kwantyfikatorem uniwersalnym: „Każdy wiedźmin jest mutantem", zapisane w logice predykatów jako:

$$\forall x(\text{Wiedzmin}(x) \rightarrow \text{Mutant}(x)).$$

Przykłady: $x = \text{Vesemir}$, $x = \text{Ciri}$, $x = \text{Bazyliszek}$.

Kwantyfikator Egzystencjalny

Rozważmy zdanie z kwantyfikatorem egzystencjalnym: „Geralt jest kochany przez czarodziejkę", zapisane w logice predykatów jako:

$$\exists x(\text{Czarodziejka}(x) \wedge \text{Kocha}(x, \text{Geralt})).$$

Przykłady: $x = \text{Yennefer}$, $x = \text{Ciri}$, $x = \text{Bazyliszek}$.

Uwaga o Kwantyfikatorach

Uwaga

Kwantyfikator uniwersalny łączy się tylko z implikacją, a kwantyfikator egzystencjalny tylko z koniunkcją.

Semantyka i Model

Semantyka określa pojęcie modelu dla formuł. Model to interpretacja, w której formuła jest prawdziwa. Mówi się, że formuła jest prawdziwa w modelu, lub że model spełnia formułę.
Niech $\mathcal{F}$ będzie formułą, a $\mathcal{M}$ modelem. Wówczas:

$$\text{Jeżeli } \mathcal{M} \text{ jest modelem dla } \mathcal{F}, \text{ to } \mathcal{M} \models \mathcal{F}$$

gdzie $\models$ oznacza, że model spełnia formułę.

Wprowadzenie do Aksjomatów Logiki Predykatów

Rozważanie przykładów i omówienie schematów wnioskowania wymaga znajomości własności wyrażeń w logice predykatów. Wprowadźmy pojęcie tautologii. Tautologie logiki predykatów są formułami prawdziwymi w dowolnym zbiorze/modelu.

Definicje

Definition

Mówimy, że dwa predykaty/funkcje $A(x)$ i $B(x)$ są równoważne, gdy mają ten sam zakres/wykres.

Definition

Mówimy, że dwie formuły domknięte A i B są równoważne, gdy formuła $A \Leftrightarrow B$ jest tautologią.

Definition

Dwa predykaty $A(x)$ i $B(x)$, $x \in X$ są równoważne wtedy i tylko wtedy, gdy zdanie $\forall x(A(x) \Leftrightarrow B(x))$ jest prawdziwe.

Tautologie

Tautologie, przydatne w rozumieniu przykładów i dalszych algorytmów. Dla uproszczenia pojedyncze symbole A i B prezentują zdanie z rachunku zdań, ale mogą być też uogólnione do formuł atomowych w logice predykatów pierwszego rzędu.

Tautologie - wywodzące się z rachunku zdań

1. Prawo wyłączonego środka: $A \vee (\neg A)$
2. Prawo sprzeczności (niemożliwe jest by formuła była jednocześnie prawdziwa i fałszywa): $\neg(A \wedge (\neg A))$
3. Prawo podwójnej negacji: $A \leftrightarrow \neg(\neg A)$
4. I prawo de Morgana: $\neg(A \wedge B) \leftrightarrow ((\neg A) \vee (\neg B))$
5. II prawo de Morgana: $\neg(A \vee B) \leftrightarrow ((\neg A) \wedge (\neg B))$
6. Prawo przemienności koniunkcji: $(A \wedge B) \leftrightarrow (B \wedge A)$
7. Prawo przemienności dysjunkcji:$(A \vee B) \leftrightarrow (B \vee A)$
8. Prawo łączności koniunkcji: $(A \wedge (B \wedge C)) \leftrightarrow ((A \wedge B) \wedge C))$
9. Prawo łączności dysjunkcji: $(A \vee (B \vee C)) \leftrightarrow ((A \vee B) \vee C))$
10. Prawo eliminacji równoważności: $(A \leftrightarrow B) \leftrightarrow (A \rightarrow B) \wedge (B \rightarrow A)$
11. Prawo eliminacji implikacji: $(A \rightarrow B) \leftrightarrow (\neg A \vee B)$
12. Rozdzielność koniunkcji względem alternatywy $(A \wedge (B \vee C)) \leftrightarrow ((A \wedge B) \vee (A \wedge C))$
13. Rozdzielność alternatywy względem koniunkcji $(A \vee (B \wedge C)) \leftrightarrow ((A \vee B) \wedge (A \vee C))$

Tautologie - Właściwe dla rachunku predykatów (kwantyfikatorów)

1. Prawa de Morgana rachunku kwantyfikatorów:
   1. $\neg\forall x A(x) \leftrightarrow \exists x \neg A(x)$
   2. $\neg\exists x A(x) \leftrightarrow \forall x \neg A(x)$
2. Przemienność kwantyfikatorów uniwersalnych:
   $\forall x \forall y A(x, y) \leftrightarrow \forall y \forall x A(x, y)$
3. Przemienność kwantyfikatorów egzystencjalnych:
   $\exists x \exists y A(x, y) \leftrightarrow \exists y \exists x A(x, y)$

Tautologie - Właściwe dla rachunku predykatów (kwantyfikatorów)

Różne kwantyfikatory nie są przemienne.

Rozważmy znaczenie zapisów, które na przykładzie pokażą, że interpretacja jest inna, gdy zmienimy kolejnością kwantyfikator uniwersalny z egzystencjalnym:

1. $\forall x(\exists y(Kocha(x, y)))$
2. $\exists y(\forall x(Kocha(x, y)))$

Pierwsze zdanie opisuje przypadek, gdy stwierdzany, że każdy kogoś kocha, natomiast drugi mówi, że istnieje ktoś (y), który jest przez wszystkich kochany.

Założenie o Unikalności Nazw i Zamkniętej Dziedzinie

Aby wyrazić w logice predykatów, że Geralt, Vesemir i Eskel są wiedźminami, wprowadzamy stałe Geralt, Vesemir i Eskel oraz predykat Wiedzmin, a następnie używamy zdań atomowych:

$$\text{Wiedzmin}(\text{Geralt}),$$
$$\text{Wiedzmin}(\text{Vesemir}),$$
$$\text{Wiedzmin}(\text{Eskel}).$$

Wszystkie stałe reprezentujące imiona powinny zatem spełniać warunek: $\text{Geralt} \neq \text{Vesemir}, \text{Geralt} \neq \text{Eskel}, \text{Vesemir} \neq \text{Eskel}$.

Założenie o Unikalności Nazw

Założenie o unikalności nazw

dla predykatu lub funkcji jest wyrażone przez formuły:

$$\forall x_1 \dots x_m y_1 \dots y_n (f(x_1, \dots, x_m) \neq g(y_1, \dots, y_n))$$

dla wszystkich par funkcji f, g, oraz

$$\forall x_1 \dots x_n y_1 \dots y_n (f(x_1, \dots, x_n) = f(y_1, \dots, y_n) \rightarrow (x_1 = y_1 \wedge \dots \wedge x_n = y_n))$$

dla wszystkich stałych funkcji f o liczbie argumentów większej niż 0. Te formuły pociągają za sobą $t_1 = t_2$ dla jakichkolwiek różnych termów t_1, t_2.

Założenie o Zamkniętej Dziedzinie

Własność wprowadzona przez Keitha Clarka w 1978 jest często używana w teorii programowania logicznego. Używa się w niej jeszcze założenia o zamkniętej dziedzinie, które zakłada, że żaden model nie zawiera elementów domenowych innych niż te wykorzystane w postaci symboli stałych.

Warunek zamkniętej dziedziny (Domain Closure Assumption)

opisuje wzór:

$$\forall x(x = C_1 \vee \cdots \vee C_n) \tag{1}$$

Założenie o Zamkniętym Świecie

Warunek umożliwia zastąpienie wszystkich kwantyfikatorów w dowolnej formule wielokrotnymi koniunkcjami (np. $\forall x(F(x) \leftrightarrow F(C_1) \wedge \cdots \wedge F(C_n))$) i dysjunkcjami (np. $\exists x(F(x) \leftrightarrow F(C_1) \vee \cdots \vee F(C_n))$).

W dziedzinie programowania logicznego wykorzystuje się założenie o zamkniętym świecie (Closed World Assumption), które zakłada, że zdania atomowe, o których nie wiemy, że są prawdziwe, uważa się za fałszywe.

Przykład bazy wiedzy

W dalszej części wykorzystany zostanie przykład pokazujący wykorzystanie logiki predykatów do opisu wiedzy czyli pewnych faktów i relacji oraz w dalszej części wnioskowania. Spełnia on założenia o unikalności nazw, zamkniętej dziedzinie i zamkniętym świecie.

Historyjna z Wiedźmina

1. Geralt jest wojownikiem.
2. Geralt mieszka w Kaer Morhen.
3. Każdy mieszkaniec Kaer Morhen jest wiedźminem.
4. Bazyliszek jest potworem.
5. Wiedźmini albo zabijają Bazyliszka dla zarobku lub jest im obojętny.
6. Każdy bywa obojętny wobec czegoś.
7. Wojownicy walczą z potworem, który nie jest im obojętny.
8. Geralt walczył z Bazyliszkiem.

Historyjka w formie logiki predykatów

1. Geralt jest wojownikiem.

$$\text{Wojownik}(\text{Geralt})$$

Historyjka w formie logiki predykatów

1. Geralt jest wojownikiem.

$$\text{Wojownik}(\text{Geralt})$$

2. Geralt mieszka w Kaer Morhen.

$$\text{Mieszka}(\text{Geralt}, \text{Kaer Morhen})$$

Historyjka w formie logiki predykatów

1. Geralt jest wojownikiem.

$$\text{Wojownik}(\text{Geralt})$$

2. Geralt mieszka w Kaer Morhen.

$$\text{Mieszka}(\text{Geralt}, \text{Kaer Morhen})$$

3. Każdy mieszkaniec Kaer Morhen jest wiedźminem.

$$\forall x(\text{Mieszka}(x, \text{Kaer Morhen}) \rightarrow \text{Wiedzmin}(x))$$

Historyjka w formie logiki predykatów

1. Geralt jest wojownikiem.

$$\text{Wojownik}(\text{Geralt})$$

2. Geralt mieszka w Kaer Morhen.

$$\text{Mieszka}(\text{Geralt}, \text{Kaer Morhen})$$

3. Każdy mieszkaniec Kaer Morhen jest wiedźminem.

$$\forall x(\text{Mieszka}(x, \text{Kaer Morhen}) \rightarrow \text{Wiedzmin}(x))$$

4. Bazyliszek jest potworem.

$$\text{Potwor}(\text{Bazyliszek})$$

Historyjka w formie logiki predykatów

1. Geralt jest wojownikiem.

$$\text{Wojownik}(\text{Geralt})$$

2. Geralt mieszka w Kaer Morhen.

$$\text{Mieszka}(\text{Geralt}, \text{Kaer Morhen})$$

3. Każdy mieszkaniec Kaer Morhen jest wiedźminem.

$$\forall x(\text{Mieszka}(x, \text{Kaer Morhen}) \rightarrow \text{Wiedzmin}(x))$$

4. Bazyliszek jest potworem.

$$\text{Potwor}(\text{Bazyliszek})$$

5. Wiedźmini albo zabijają Bazyliszka dla zarobku lub jest im obojętny.

$$\forall x(\text{Wiedzmin}(x) \rightarrow \text{Mord\_Na\_Zlecenie}(x, \text{Bazyliszek}) \vee \text{Obojetny}(x, \text{Bazyliszek}))$$

Historyjka w formie logiki predykatów

1. Geralt jest wojownikiem.

$$\text{Wojownik}(\text{Geralt})$$

2. Geralt mieszka w Kaer Morhen.

$$\text{Mieszka}(\text{Geralt}, \text{Kaer Morhen})$$

3. Każdy mieszkaniec Kaer Morhen jest wiedźminem.

$$\forall x(\text{Mieszka}(x, \text{Kaer Morhen}) \rightarrow \text{Wiedzmin}(x))$$

4. Bazyliszek jest potworem.

$$\text{Potwor}(\text{Bazyliszek})$$

5. Wiedźmini albo zabijają Bazyliszka dla zarobku lub jest im obojętny.

$$\forall x(\text{Wiedzmin}(x) \rightarrow \text{Mord\_Na\_Zlecenie}(x, \text{Bazyliszek}) \vee \text{Obojetny}(x, \text{Bazyliszek}))$$

6. Każdy bywa obojętny wobec czegoś.

$$\forall x(\exists y(\text{Obojetny}(x, y))$$

Historyjka w formie logiki predykatów

1. Geralt jest wojownikiem.

$$\text{Wojownik}(\text{Geralt})$$

2. Geralt mieszka w Kaer Morhen.

$$\text{Mieszka}(\text{Geralt}, \text{Kaer Morhen})$$

3. Każdy mieszkaniec Kaer Morhen jest wiedźminem.

$$\forall x(\text{Mieszka}(x, \text{Kaer Morhen}) \rightarrow \text{Wiedzmin}(x))$$

4. Bazyliszek jest potworem.

$$\text{Potwor}(\text{Bazyliszek})$$

5. Wiedźmini albo zabijają Bazyliszka dla zarobku lub jest im obojętny.

$$\forall x(\text{Wiedzmin}(x) \rightarrow \text{Mord\_Na\_Zlecenie}(x, \text{Bazyliszek}) \vee \text{Obojetny}(x, \text{Bazyliszek}))$$

6. Każdy bywa obojętny wobec czegoś.

$$\forall x(\exists y(\text{Obojetny}(x, y))$$

7. Wojownicy walczą z potworem, który nie jest im obojętny.

$$\forall x \forall y(\text{Wojownik}(x) \wedge \text{Potwor}(y) \wedge \text{Walka}(x, y) \rightarrow \neg\text{Obojetny}(x, y))$$

Historyjka w formie logiki predykatów

1. Geralt jest wojownikiem.

$$\text{Wojownik}(\text{Geralt})$$

2. Geralt mieszka w Kaer Morhen.

$$\text{Mieszka}(\text{Geralt}, \text{Kaer Morhen})$$

3. Każdy mieszkaniec Kaer Morhen jest wiedźminem.

$$\forall x(\text{Mieszka}(x, \text{Kaer Morhen}) \rightarrow \text{Wiedzmin}(x))$$

4. Bazyliszek jest potworem.

$$\text{Potwor}(\text{Bazyliszek})$$

5. Wiedźmini albo zabijają Bazyliszka dla zarobku lub jest im obojętny.

$$\forall x(\text{Wiedzmin}(x) \rightarrow \text{Mord\_Na\_Zlecenie}(x, \text{Bazyliszek}) \vee \text{Obojetny}(x, \text{Bazyliszek}))$$

6. Każdy bywa obojętny wobec czegoś.

$$\forall x(\exists y(\text{Obojetny}(x, y))$$

7. Wojownicy walczą z potworem, który nie jest im obojętny.

$$\forall x \forall y(\text{Wojownik}(x) \wedge \text{Potwor}(y) \wedge \text{Walka}(x, y) \rightarrow \neg\text{Obojetny}(x, y))$$

8. Geralt walczył z Bazyliszkiem.

Analiza przykładu

- pomija następstwo czasu, kolejność zdarzeń;
- nie wskazuje wyraźnie, że jest to opis nierzeczywistego świata
- brak jest określenia zasięgu w zdaniu 6. Czy dla każdego istnieje ktoś dla kogo jest się obojętnym? Czy są to różne osoby? Czy istnieje ktoś, wobec kogo wszyscy są obojętni?
- Zdanie 7 można też przedstawić jako
  $\forall x, y(\text{Wojownik}(x) \wedge \text{Potwor}(y) \wedge \neg\text{Obojetny}(x, y) \rightarrow \text{Walka}(x, y))$
  gdyż nie jest jednoznaczne, co jest przyczyną a co skutkiem.

Inżynieria wiedzy

To kroki procesu modelowania problemu za pomocą formalnych metod, takich jak na przykład logika predykatów, oraz do dalszego wykorzystania tego modelu do wnioskowania i rozwiązywania konkretnych problemów.

Krok 1 - Zidentyfikuj zadanie.

Pierwszym krokiem jest określenie celu lub problemu, który ma być rozwiązany. To wymaga zrozumienia kluczowych aspektów zadania oraz jego kontekstu. Wiąże się też z określeniem przebiegu dialogu, odkrycia jakie będą możliwe pytania do systemu.

Krok 2 - Zgromadź wymaganą wiedzę.

Następnie zbierz wszystkie niezbędne informacje i dane, które są istotne dla zadania. Ta wiedza może pochodzić z różnych źródeł, w tym z literatury, ekspertów w danej dziedzinie, lub poprzez obserwacje.

Inżynieria wiedzy

Krok 3 - Wybierz słownik predykatów, funkcji i stałych.

Określ i zdefiniuj predykaty, funkcje i stałe, które będą używane do reprezentowania wiedzy w ramach zadania. To pomoże w formalizacji problemu.

Krok 4 - Zakoduj ogólną wiedzę dotyczącą zadania.

Zapisz wszystkie aksjomaty dotyczące termów ze słownika. Na tym etapie wykrywa się luki i błędy koncepcyjne.

Krok 5 - Zakoduj specyficzną wiedzę dotyczącą zadania.

Oprócz ogólnej wiedzy, ważne jest także zakodowanie informacji szczegółowych i specyficznych, wejść do systemu lub informacji znanych podczas uruchomienia systemu (fakty).

Inżynieria wiedzy

Krok 6 -Postaw pytania do procedury wnioskowania i otrzymaj odpowiedzi.

Użyj systemu wnioskowania do zadawania pytań związanych z zadaniem i odbieraj odpowiedzi, które pomogą w podejmowaniu decyzji lub dalszej analizie.

Krok 7 - Usuń błędy z bazy wiedzy.

Na koniec, ważne jest, aby przeanalizować wszystkie odpowiedzi systemy i poprawić wszelkie błędy lub nieścisłości w bazie wiedzy, aby zapewnić jej zgodność z oczekiwaniami projektanta.

Unifikacja

Unifikacja

w logice pierwszego rzędu jest procesem mającym kluczowe znaczenie i polega na znalezieniu substytucji, która sprawia, że dwie formuły atomowe stają się identyczne.
Wyrażenie $x_1/t_1, x_2/t_2, ..., x_n/t_n$ oznacza substytucję mapującą zmienną x_i na term t_i, dla $1 \leq i \leq n$.

Ogólna reguła mówi, że

- stała unifikuje się ze stałą,
- zmienna unifikuje się ze stałymi i funkcjami,
- formuły złożone muszą ulec dekompozycji i każdy z argumentów jest sprawdzany pod względem możliwości substytucji

Algorytm Unifikacji

Algorytm 1 Algorytm Unifikacji

procedura Unifikacja(*Exp*) ▷ Zbiór par wyrażeń do unifikacji. *Expr*
  dopóki *Expr* $\neq \{\}$ **wykonaj** ▷ istnieją pary do unifikacji
    Wybierz parę wyrażeń $(expr_1, expr_2)$.
    jeżeli $expr_1$ i $expr_2$ są identyczne **to**
      Nie dodawaj nic do listy substytucji
    jeżeli $expr_1$ jest zmienną **to**
      jeżeli $expr_1$ nie występuje w $expr_2$ **to**
        Dodaj do listy substytucji $expr_1 / expr_2$
      w przeciwnym razie
        break ▷ (niepowodzenie)
    jeżeli $expr_2$ jest zmienną **to**
      jeżeli $expr_2$ nie występuje w $expr_1$ **to**
        Dodaj do listy substytucji $expr_2 / expr_1$
      w przeciwnym razie
        break ▷ (niepowodzenie)
    jeżeli $expr_1 = f(r_1, \ldots, r_n)$ i $expr_2 = f(s_1, \ldots, s_n)$ **to** ▷ oba wyrażenia są złożone i mają taką samą liczbę argumentów
      Dodaj do listy substytucji wynik Unifikuj ($[r_1, \ldots, r_n]$ i $[s_1, \ldots, s_n]$) ▷ składniki wyrażeń
    w przeciwnym razie ▷ oba wyrażenia są złożone i mają inną liczbę argumentów
      break ▷ (niepowodzenie)
  jeżeli wszystkie pary unifikowalne **to**
    zwróć Lista Substytucji
  w przeciwnym razie
    zwróć Niepowodzenie

Przykłady możliwej unifikacji

Wyrażenia do Unifikacji:
(1) Zabija(w, x)
(2) Zabija(Geralt, Strzyga)
Proces Unifikacji:

- W pierwszej formule x, w są zmiennymi.
- W drugiej formule mamy konkretne byty: Geralt i Strzyga.
- Unifikacja poprzez zastąpienie x do Strzyga i zastąpienie w do Geralt.

Wynik Unifikacji: Substytucja: $\{x/\text{Strzyga}, w/\text{Geralt}\}$

Przykłady niemożliwej unifikacji

Wyrażenia do Unifikacji:
(1) Przyjaciel(Geralt, x)
(2) Przyjaciel(Yennefer, Triss)
Proces Unifikacji:

- W pierwszym wyrażeniu x jest zmienną, a Geralt odnosi się do postaci z książki.
- W drugim wyrażeniu Yennefer i Triss to konkretne postaci.
- Próba unifikacji napotyka na problem, ponieważ *Geralt* i *Yennefer* są różnymi postaciami/stałymi.

Wynik Unifikacji: Niepowodzenie

Reguły wnioskowania

Oprócz unifikacji do wnioskowania potrzebne są zasady, które w danym języku formalnym określą jak poprawnie wyprowadzać wnioski z jednego lub więcej zdań/wyrażeń, które są uznawane za prawdziwe (TRUE), oczywiście gdy mowa o logice dwuwartościowej. Reguły wnioskowania w logice umożliwiają konstrukcję poprawnych argumentów i dowodów.

Modus Ponens

$$\frac{P(a_1, \ldots, a_n), P(a_1, \ldots, a_n) \rightarrow Q(b_1, \ldots, b_m)}{Q(b_1, \ldots, b_m)}\theta \qquad (2)$$

Example

$$\frac{\text{Wiedzmin}(\text{Geralt}), \forall x(\text{Wiedzmin}(x) \rightarrow \text{Mutant}(x))}{\text{Mutant}(\text{Geralt})}(x/\text{Geralt})$$

Modus Tollens

$$\frac{\neg Q(b_1, \ldots, b_m), P(a_1, \ldots, a_n) \rightarrow Q(b_1, \ldots, b_m)}{\neg P(a_1, \ldots, a_n)}\theta \qquad (3)$$

Example

$$\frac{\neg \text{Mutant}(\text{Yennefer}), \forall x(\text{Wiedzmin}(x) \rightarrow \text{Mutant}(x))}{\neg \text{Wiedzmin}(\text{Yennefer})}(x/\text{Yennefer})$$

Modus Ponendo Tollens

$$\frac{\neg(P(a_1,\ldots,a_n) \wedge Q(b_1,\ldots,b_m)), P(a_1,\ldots,a_n)}{\neg Q(b_1,\ldots,b_m)}\theta \qquad (4)$$

Example

$$\frac{\neg(\exists x(\text{Wiedzmin(x)} \wedge \text{Czarodziej}(x))), \text{Wiedzmin}(\mathit{Geralt})}{\neg\text{Czarodziej(Geralt)}}(x/\text{Geralt})$$

Sylogizm Dysjunkcyjny

$$\frac{P(a_1, \ldots, a_n) \vee Q(b_1, \ldots, b_m), \neg P(a_1, \ldots, a_n)}{Q(b_1, \ldots, b_m)}\theta \qquad (5)$$

Example

$$\frac{\forall x(\text{Wiedzmin(x)} \vee \text{Czarodziej}(x)), \neg\text{Wiedzmin(Yennefer)}}{\text{Czarodziej(Yennefer)}}(x/\text{Yennefer})$$

Reguła Rezolucji

$$\frac{P_1(a_1,\ldots,a_n) \vee Q_1(b_1,\ldots,b_m), \neg P_1(c_1,\ldots,c_n) \vee Q_2(d_1,\ldots,d_m)}{Q_1(b_1,\ldots,b_m) \vee Q_2(d_1,\ldots,d_m)}\theta \tag{6}$$

Example

$$\frac{\text{Wiedzmin(Geralt)} \vee \neg\text{Potwor}(x), \neg\text{Wiedzmin(y)} \vee \text{Mutant(y)}}{\neg\text{Potwor}(x) \vee \text{Mutant(Geralt)}}(y/\text{Geralt})$$

Łańcuchowanie Progresywne

Łańcuchowanie progresywne można wyobrazić sobie jako przechodzenie po wszystkich regułach bazy wiedzy dla wszystkich znanych, zadeklarowanych faktów. Proces ten jest kontynuowany tak długo, aż wyczerpią się fakty i reguły do dowodzenia.

Algorytm i Optymalizacje Łańcuchowania Progresywnego

Algorytm łańcuchowania progresywnego jest mało wydajny dla dużych baz wiedzy ze względu na zagnieżdżenie trzech pętli. Stosuje się różne zabiegi zmniejszające złożoność, takie jak wybieranie tylko tych reguł, które mają w przesłance fakty dostępne z listy faktów, oraz unifikacja.

Łańcuchowanie Regresywne

Łańcuchowanie regresywne rozpoczyna się od celu wnioskowania (hipotezy) i kontynuowane jest przez poszukiwanie reguł, których konkluzje są predykatami celu wnioskowania. Z przesłanki tej reguły powstaje nowa hipoteza do udowodnienia.

Algorytm i Charakterystyka Łańcuchowania Regresywnego

Algorytm łańcuchowania regresywnego wykorzystuje rekurencję do przetwarzania celów. Jest to algorytm przeszukiwania w głąb, którego złożoność pamięciowa jest liniowa względem wielkości dowodu. Pomimo ryzyka nieskończonej rekurencji, jest to główne podejście stosowane w językach programowania opartych na logice, np. w Prologu.

Przykład

Rozważmy przykład prostej bazy wiedzy, by pokazać jak działa łańcuchowanie progresywne. Rozważmy następującą historyjkę o wiedźminie:

Geralt mieszka w zamku Kaer Morhen. Mieszkańcy tego zamku uczą się skutecznej walki, by zostać wojownikiem. Ponadto w pewnym momencie przechodzą oni też próbę traw. Po próbie traw, stają się wiedźminami.

Cel wnioskowania: Chcemy się dowiedzieć, czy Geralt jest wiedźminem.

Przykład - cd

Zapiszmy historię w logice predykatów.

Fakt F1 $\text{Mieszka}(\text{Geralt}, \text{Kaer\_Morhen})$

Reguła R1 $\forall x(\text{Mieszka}(x, \text{Kaer\_Morhen}) \rightarrow \text{Wojownik}(x))$

Reguła R2 $\forall x(\text{Mieszka}(x, \text{Kaer\_Morhen}) \rightarrow \text{Proba}(x, \text{traw}))$

Reguła R3 $\forall x(\text{Proba}(x, \text{traw}) \rightarrow \text{Wiedzmin}(x))$

Cel G1 $\text{Wiedzmin}(\text{Geralt}))$

Przykład – łańcuchowanie progresywne

1. Zaczynamy od znanego faktu **F1** i stosujemy go z reguła **R1**. Zastosowanie dedukcji i unifikacji z substytucją (x/Geralt) wprowadza nowy fakt:

   F2: Wojownik(Geralt))

2. Kontynuujemy dopasowanie **F1** do kolejnej reguły **R2**. Zgodnie z regułą modus ponens i unifikacją z substytucją (x/Geralt) uzyskujemy fakt:

   F3: Proba(Geralt, traw))

3. Próba zastosowania faktu **F1** z regułą **R3** zawodzi, gdyż przesłanka tego faktu nie pasuje do predykatu z faktu.

4. Posługujemy się teraz faktem **F2**, który nie pasuje do przesłanki żadnej z trzech reguł.

5. Używamy faktu **F3**, który nie pasuje do predykatów w przesłankach reguł **R1** i **R2**, ale pasuje do **R3**. Spełnialność przesłanki oznacza prawdziwość konkluzji, zatem powstaje nowy fakt, przy liście substytucji (x/Geralt):

   F4: Wiedzmin(Geralt))

Przykład – łańcuchowanie progresywne

1. Zaczynamy od hipotezy **G1** i szukamy najpierw faktu, a potem reguły, która w konkluzji ma formułę zgodną z formułą celu. Sprawdzamy kolejno: **F1, R1, R2 i R3**, jest to reguła **R3**. Zastosowanie unifikacji z substytucją (x/Geralt) wprowadza nową hipotezę $G2$:

   G2: Proba(Geralt, traw)

2. Kontynuujemy z nowym celem **G2** i szukamy faktu lub reguły, która w konkluzji ma formułę zgodną z formułą celu. Sprawdzamy kolejno: **F1, R1, R2**, jest to **R2**. Uzyskujemy unifikację z substytucją (x/Geralt) nową hipotez **G3**:

   G3: Mieszka(Geralt, Kaer_Morhen)

3. Kontynuujemy z nowym celem **G3** i szukamy faktu lub reguły, która w konkluzji ma formułę zgodną z formułą celu. Sprawdzamy kolejno: **F1**. W tym kroku nie generuje się nowego celu, oznacza to, że potwierdziliśmy spełnialność celu **G3**. Co oznacza, że cel **G2** ma status TRUE i **G1** też.

Wnioskowanie z użyciem rezolucji

Dowodzenie przez zastosowanie reguły rezolucji, prowadzi do zupełnego algorytmu wnioskowania z wykorzystaniem dowolnego zupełnego algorytmu wyszukiwania.

Wiele systemów dowodzenia twierdzeń w tym reguła rezolucji w logice pierwszego rzędu konwertuje formułę pierwszego rzędu do formy klauzulowej przed podjęciem próby jej udowodnienia.

1. Eliminacja Równoważności

$$A \leftrightarrow B \equiv (A \rightarrow B) \wedge (B \rightarrow A) \tag{7}$$

2. Eliminacja Implikacji

$$A \rightarrow B \equiv \neg A \vee B \tag{8}$$

3. Wyprowadzenie Negacji

- Podwójne zaprzeczenie: $\neg\neg A \equiv A$
- I prawo de Morgana: $\neg(A \wedge B) \equiv \neg A \vee \neg B$
- II prawo de Morgana: $\neg(A \vee B) \equiv \neg A \wedge \neg B$
- Prawo de Morgana dla kwantyfikatora ogólnego: $\neg\forall x A(x) \equiv \exists x \neg A(x)$
- Prawo de Morgana dla kwantyfikatora egzystencjalnego: $\neg\exists x A(x) \equiv \forall x \neg A(x)$

4. Standaryzacja Zmiennych

Zmienne związane kwantyfikatorami powinny być unikalne dla każdego kwantyfikatora, aby uniknąć dwuznaczności.

5. Skolemizacja

$$\exists x A[x] \equiv A[f(x_1, \ldots, x_n)] \tag{9}$$

gdzie $x_1, \ldots, x_n$ to zmienne kwantyfikowane uniwersalnie, a f to nowa funkcja Skolema.

6. Przeniesienie Kwantyfikatorów Uniwersalnych

Kwantyfikatory uniwersalne są przenoszone na początek wyrażenia.

7. Usunięcie Kwantyfikatorów Uniwersalnych

Kwantyfikatory uniwersalne są usuwane z wyrażenia.

8. Rozdzielenie iloczynu sumą

$$(A \vee (B \wedge C)) \equiv ((A \vee B) \wedge (A \vee C)) \tag{10}$$

9. Wyodrębnienie Sum - Klauzul

Wyrażenia są rozbijane na klauzule przy użyciu łącznika $\wedge$, a każda klauzula jest umieszczana w nawiasach klamrowych.

Postać klauzulowa- przykład

Rozważmy następujące zdanie: Każda czarodziejka, która zna Geralta jest w nim zakochana, albo uważa, że inne kobiety, które kochają się w jakimś wiedźminie są naiwne. Możliwa forma tego zdania w logice predykatów:

$$\forall x(\text{Czarodziej}(x) \wedge \text{Kobieta}(x) \wedge \text{Zna}(x, \text{Geralt}) \rightarrow (\text{Kocha}(x, \text{Geralt}) \vee (\forall y(\exists z(\text{Kobieta}(y) \wedge \text{Wiedzmin}(z) \wedge \text{Kocha}(y, z))) \rightarrow \text{Ocenia}(x, y, \text{naiwna}))))$$

1. Eliminacja Równoważności

Brak — krok pomijany.

2. Eliminacja Implikacji

$$\forall x(\neg(\text{Czarodziej}(x) \wedge \text{Kobieta}(x) \wedge \text{Zna}(x, \text{Geralt}))\vee$$
$$(\text{Kocha}(x, \text{Geralt})\vee$$
$$(\forall y \neg(\exists z \text{Kobieta}(y) \wedge \text{Wiedzmin}(z) \wedge \text{Kocha}(y, z))\vee$$
$$\text{Ocenia}(x, y, \text{naiwna})))$$

3. Wyprowadzenie Negacji

$$\forall x(\neg \text{Czarodziej}(x) \vee \neg \text{Kobieta}(x) \vee \neg \text{Zna}(x, \text{Geralt}) \vee$$
$$(\text{Kocha}(x, \text{Geralt}) \vee$$
$$(\forall y(\forall z(\neg \text{Kobieta}(y) \vee \neg \text{Wiedzmin}(z) \vee \neg \text{Kocha}(y, z)))) \vee$$
$$\text{Ocenia}(x, y, \text{naiwna})))$$

4. Standaryzacja Zmiennych

Każda zmienna związana kwantyfikatorem ma inny identyfikator — krok pomijamy.

5. Skolemizacja

Brak kwantyfikatora egzystencjalnego — krok pomijamy.

6. Przeniesienie Kwantyfikatorów Uniwersalnych

$$\forall x \forall y \forall z (\neg \text{Czarodziej}(x) \vee \neg \text{Kobieta}(x) \vee \neg \text{Zna}(x, \text{Geralt}) \vee \text{Kocha}(x, \text{Geralt}) \vee \neg \text{Kobieta}(y) \vee \neg \text{Wiedzmin}(z) \vee \neg \text{Kocha}(y, z) \vee \text{Ocenia}(x, y, \text{naiwna}))$$

7. Usunięcie Kwantyfikatorów Uniwersalnych

$$\neg\text{Czarodziej}(x) \vee \neg\text{Kobieta}(x) \vee \neg\text{Zna}(x, \text{Geralt}) \vee \text{Kocha}(x, \text{Geralt}) \vee$$
$$\neg\text{Kobieta}(y) \vee \neg\text{Wiedzmin}(z) \vee \neg\text{Kocha}(y, z) \vee \text{Ocenia}(x, y, \text{naiwna})$$

8. Rozdzielność sum względem iloczynów

Brak — krok pomijamy.

9. Wyodrębnienie Sum - Klauzul

$$\{\neg\text{Czarodziej}(x) \vee \neg\text{Kobieta}(x) \vee \neg\text{Zna}(x, \text{Geralt}) \vee \text{Kocha}(x, \text{Geralt}) \vee$$
$$\neg\text{Kobieta}(y) \vee \neg\text{Wiedzmin}(z) \vee \neg\text{Kocha}(y, z) \vee \text{Ocenia}(x, y, \text{naiwna})\}$$

Przykład KB po przekształceniu na CNF

Lp.	Wyrażenie logiczne	Postać klauzulowa
1	Wojownik(Geralt)	{Wojownik(Geralt)}
2	Mieszka(Geralt, Kaer Morhen)	{Mieszka(Geralt, Kaer Morhen)}
3	$\forall x$(Mieszka(x, Kaer Morhen) $\rightarrow$ Wiedzmin(x))	{$\neg$Mieszka(x, Kaer Morhen) $\vee$ Wiedzmin(x)}
4	Potwor(Bazyliszek)	{Potwor(Bazyliszek)}
5	$\forall x$(Wiedzmin(x) $\rightarrow$ Mord_Na_Zlecenie(x, Bazyliszek)$\vee$ Obojetny(x, Bazyliszek))	{$\neg$Wiedzmin(x) $\vee$ Mord_Na_Zlecenie(x, Bazyliszek)$\vee$ Obojetny(x, Bazyliszek)}
6	$\forall x(\exists y$(Obojetny(x, y)))	{Obojetny($x, S(x)$)} *($S(x)$ funkcja skolemowa, która jako argument przyjmuje zmienną spod kwantyfikatora uniwersalnego)*
7	$\forall x \forall y$(Wojownik(x) $\wedge$ Potwor(y) $\wedge$ Walka(x, y) $\rightarrow \neg$Obojetny(x, y))	{$\neg$Wojownik(x) $\vee$ $\neg$Potwor(y) $\vee$ $\neg$Walka(x, y) $\vee \neg$Obojetny(x, y)}
8	Walka(Geralt, Bazyliszek)	{Walka(Geralt, Bazyliszek)}

Reductio ad absurdum

W procesie dowodzenia często, istnieje ogólny zbiór A aksjomatów oraz szczególna formuła F, którą chce się udowodnić. Chodzi więc o wykazanie, że formuła $A \rightarrow F$ jest ważna. W podejściu dowodzenia przez zaprzeczenie, wykazuje się, że $\neg(A \rightarrow F)$ jest niespełnialna. Zatem, $\neg(A \rightarrow F)$ jest przekształcana na $A \wedge \neg F$ w formie klauzulowej. Następnie uzyskuje się zbiór SA klauzul z A oraz zbiór SF klauzul z $\neg F$. Zbiór $SA \cup SF$ jest niespełnialny wtedy i tylko wtedy, gdy $A \rightarrow F$ jest ważne/prawdziwe.

Przykład

Wracając do przykładu z książek o wiedźminie. Użyjemy zapisanych tam klauzul do udowodnienia, że Geralt zamordował na zlecenie Bazyliszka. Do bazy klauzul dodamy zanegowaną hipotezę do udowodnienia czyli:
$\{\neg \text{Mord\_Na\_Zlecenie}(\text{Geralt}, \text{Bazyliszek})\}$.

Wnioskowanie rezolucją

K	Klauzula	Komentarz
1	$\{\neg \text{Mord\_Na\_Zlecenie}(\text{Geralt}, \text{Bazyliszek})\}$	hipoteza
2	$\{\neg \text{Wiedzmin}(x) \vee \text{Mord\_Na\_Zlecenie}(x, \text{Bazyliszek}) \vee \overline{\text{Obojetny}}(x, \text{Bazyliszek})\}$	klauzula 5
3	$\{\neg \text{Wiedzmin}(\text{Geralt}) \vee \text{Obojetny}(\text{Geralt}, \text{Bazyliszek})\}$	substytucja x/Geralt i rezolwenta z rezolucji 1, 2
4	$\{\neg \text{Mieszka}(x, \text{Kaer Morhen}) \vee \text{Wiedzmin}(x)\}$	klauzula 3
5	$\{\neg \text{Mieszka}(\text{Geralt}, \text{Kaer Morhen}) \vee \text{Obojetny}(\text{Geralt}, \text{Bazyliszek})\}$	substytucja x/Geralt i resolwenta z rezolucji dla 3 i 4
6	$\{\text{Mieszka}(\text{Geralt}, \text{Kaer Morhen})\}$	klauzula 2
7	$\{\text{Obojetny}(\text{Geralt}, \text{Bazyliszek})\}$	resolwenta z rezolucji dla 5 i 6
8	$\{\neg \text{Wojownik}(x) \vee \neg \text{Potwor}(y) \vee \neg \text{Walka}(x, y) \vee \neg \text{Obojetny}(x, y)\}$	klauzula 7
9	$\{\neg \text{Wojownik}(\text{Geralt}) \vee \neg \text{Potwor}(\text{Bazyliszek}) \vee \neg \text{Walka}(\text{Geralt}, \text{Bazyliszek})\}$	substytucja x/Geralt i y/Bayliszek i resolwenta z rezolucji dla 7 i 8
10	$\{\text{Wojownik}(\text{Geralt})\}$	klauzula 1
11	$\{\neg \text{Potwor}(\text{Bazyliszek}) \vee \neg \text{Walka}(\text{Geralt}, \text{Bazyliszek})\}$	resolwenta z rezolucji dla 9 i 10
12	$\{\text{Potwor}(\text{Bazyliszek})\}$	klauzula 4
13	$\{\neg \text{Walka}(\textit{Geralt}, \textit{Bazyliszek})\}$	resolwenta z rezolucji dla 11 i 12
14	$\{\text{Walka}(\textit{Geralt}, \textit{Bazyliszek})\}$	klauzula 8
15	$\{\}$	rezolucja 13 i 14 - sprzeczność - FALSE

Jak można usprawnić algorytm rezolucji

Jest wiele technik, dzięki którym można zmniejszyć złożoność obliczeniową algorytmu.

Klauzule Jednostkowe

Klauzule jednostkowe to technika, gdzie wybierane są zdania proste składające się z pojedynczych termów. Rezolucja jednostkowa wymaga zastosowania tylko pojedynczego wyrażenia, co sprawia, że ten algorytm samodzielnie nie jest zupełny, ale jest skuteczny w zbiorze klauzul Horna.

Dedukcja P1

Dedukcja P1 opiera się na założeniu, że rezolucja pozytywna jest kompletna, to znaczy, że jeśli zbiór S jest niespełnialny, to istnieje sprzeczność z S, w której wszystkie rezolucje są pozytywne.

Hiperrezolucja

Hiperrezolucja to modyfikacja rezolucji pozytywnej, w której seria pozytywnych resolwent jest wykonywana naraz. Jest czasami użyteczna, ponieważ redukuje liczbę pośrednich wyników, które muszą być przechowywane w weryfikatorze.

Rezolucja na Wejściu

Rezolucja na wejściu próbuje wykazać niespełnialność, gdy do zbioru klauzul aksjomatów A dodamy zanegowane wyrażenie do udowodnienia F i uruchamiamy rezolucję. Rezolucje obejmujące klauzule z F są bardziej przydatne, ponieważ rezolucje obejmujące dwie klauzule z A są połączeniem ogólnych aksjomatów.

Zbiory Wsparcia

Strategia zbioru wsparcia pozwala na wykonywanie tylko rezolucji obejmujących klauzule F lub klauzule z nich pochodzące. Można to osiągnąć za pomocą strategii zbioru wsparcia, jeśli zbiór z F jest odpowiednio dobrany.

Wprowadzenie do Prologu

Prolog (programming in logic) jest najpopularniejszym językiem do programowania w logice. Efektywność języka Prolog wynika z połączenia ograniczonej formy logiki pierwszego rzędu oraz specyficznych strategii rezolucji i wyszukiwania.

Składnia Prologu

Przedstawimy składnię języka Prolog wychodząc od postaci klauzulowej w logice predykatów pierwszego rzędu. Klauzule pozytywne wyprzedzają klauzule negatywne:

$$(\{P_1(x) \vee P_2(x) \vee \cdots \vee P_n(x)) \vee (\neg N_1(x) \vee \neg N_2(x) \vee \cdots \vee \neg N_m(x))\}$$

Przekształcenie Klauzul

Po zastosowaniu I prawa de Morgana uzyskujemy postać zdania:

$$\{(P_1(x) \vee P_2(x) \vee \cdots \vee P_n(x)) \vee (\neg(N_1(x) \wedge N_2(x) \wedge \cdots \wedge N_m(x))\}$$

Forma Implikacji

Przechodząc z formy sumy na implikację otrzymujemy:

$$\{(P_1(x) \vee P_2(x) \vee \cdots \vee P_n(x)) \leftarrow (N_1(x) \wedge N_2(x) \wedge \cdots \wedge N_m(x))\}$$

Notacja Semiprologowa

Wprowadzając oznaczenia:

- $\wedge$ zamienimy na przecinek (,)
- $\vee$ zamienimy na średnik (;)
- $\leftarrow$ zamienimy na dwukropek-myślnik (: $-$)

otrzymujemy klauzulę postaci:
Nazwijmy tą notację semiprologową.

Definicja Klauzul Horna

Definition

Klauzula Horna to klauzula, która ma co najwyżej jeden pozytywny predykat. Dzieli się je na:

1. Klauzule z głową posiadające jeden pozytywny predykat.
2. Klauzule bez głowy bez pozytywnego predykatu.

Notacja Semiprologowa

Stosując notację semiprologową, klauzula Horna z jednym wyrażeniem pozytywnym P_i można zapisać jako:

$$\{(P(x) : -(N_1(x), N_2(x), \ldots, N_m(x))\}$$

Ograniczenia Klauzul Horna

Implikacja, której konkluzja jest dysjunkcją, nie jest wyrażalna w formie klauzul Horna. Na przykład, reguła „Jeśli zdiagnozowano u Ciebie wysokie ciśnienie krwi, musisz albo zmniejszyć spożycie soli, albo przyjmować leki” nie jest reprezentowalna jako klauzula Horna.

Zastosowanie Klauzul Horna w Prologu

Klauzule Horna znalazły ogromne zastosowanie w językach programowania logicznego, takich jak Prolog. W Prologu program składa się ze zbioru klauzul Horna, które opisują fakty i reguły:

- **Fakty**: klauzule Horna z samą głową bez treści.
- **Reguły**: klauzule Horna z głową i z treścią.

Podejście Deklaratywne w Prologu

W Prologu stosuje się podejście deklaratywne, które opisuje logikę obliczeń bez ich bezpośredniej implementacji. Kod Prologu opisuje, co ma być osiągnięte, a nie jak to osiągnąć.

Strategia Rezolucji w Prologu

Stosowana w Prologu strategia rezolucji to liniowa rezolucja wejściowa, a strategia wyszukiwania to łańcuchowanie regresywne, często realizowane przez algorytm przeszukiwania strategią w głąb (Depth First Search).

Komponenty Prologu