Ćwiczenie 8

Logika i wnioskowanie

Adam Nowakowski (300428)

1. Zadanie

Zadaniem była implementacja algorytmu *wnioskowania przez rezolucję i zaprzeczenie*. Algorytm miał przyjmować zbiór klauzul, zbiór faktów obserwowalnych (literałów) oraz fakt nieobserwowalny (literału) do udowodnienia. Należało także zbadać prawdziwość faktów dla przykładowych problemów.

2. Wprowadzenie teoretyczne

Wnioskowanie przez rezolucję i zaprzeczenie jest narzędziem do analizy **zbioru klauzul.** Aby w pełni zrozumieć z czym mamy do czynienia, trzeba objaśnić kilka terminów związanych z rachunkiem predykatów.

- **Rachunek predykatów** to naturalne rozszerzenie logiki zdań. Ponieważ do dobrego opisu zależności logicznych w świecie nie wystarczają zdania i spójniki logiczne, wymyślono, że stwierdzenia będą parametryzowane za pomocą obiektów. Przykład (definicja granicy ciągu):

  $$\forall_{\varepsilon>0}\exists_{N>0}\forall_{i\geq N}\|x_i - x^*\| < \varepsilon$$

  Gdzie:
  - x^*- granica ciągu
  - $\|x_i - x^*\| < \varepsilon$ – predykat
  - x_i, x^*, ε – obiekty od których zależy prawdziwość

- Dla **obiektów** można formułować stwierdzenia dotyczące ich cech i relacji między nimi. Można je podzielić na:
  - Stałe – obiekty z pewnego uniwersum zapisane jako liczby bądź słowa z wielkiej litery (np. Jan lub 5)
  - Zmienne – pewne uogólnienie obiektów, zapisywane z małej litery (np. x, y, nazwisko)
  - Funkcje – mapują obiekty na inne obiekty (np. Plus(2,3), Następnik(x), Ojciec(Jan))
  - Termy – ogólne pojęcie obejmujące stałe zmienne oraz funkcje – czyli wszystko to co można nazwać obiektem

- **Predykat** to wyrażenie postaci PRED($t_1, \dots, t_n$). Przypisuje termom $t_1, \dots, t_n$ prawdę lub fałsz. (np. DODATNIA(x), OJCEC(Jan), PADA

- **Atom** to predykat P(…) oraz T i F.

- **Literałem** jest formuła postaci P(…), bądź ¬P(…), gdzie P to predykat.

- **Formuła poprawnie zbudowana rachunku predykatów I rzędu** (PFZ) jest jednolitą formą zapisu w rachunku predykatów, która ułatwia późniejszą jej analizę.

- Za pomocą pewnych tożsamości dotyczących FPZ, każdą taką formułę można sprowadzić do formy klauzuli. **Klauzula** to alternatywa literałów, czyli formuła postaci $L_1 \vee ... \vee L_n$.

Reguła rezolucji ma formę:

$$\frac{p \vee q, \neg q \vee r}{p \vee r}$$

Oznacza to tyle, że gdy w zbiorze klauzul znajdują się klauzule $p \vee q$ oraz $\neg q \vee r$, to ich prawdziwość jest spełniona, jeżeli jest spełnione $p \vee r$. Jest to pewien sposób na analizę zbioru klauzul i wnioskowanie prawdziwości.

Chcąc dowiedzieć się czegoś na temat jakiegoś faktu, którego nie jesteśmy w stanie zaobserwować i mając pewną wiedzę o uniwersum z którego ten fakt pochodzi możemy zastosować wnioskowanie przez rezolucję i zaprzeczenie. Działa ono w następujący sposób:

- Mamy pewien zbiór klauzul, który opisuje wiedzę nt. uniwersum, w którym przeprowadzamy wnioskowanie oraz fakty, które możemy zaobserwować i jesteśmy pewni co do ich prawdziwości. Oba te zbiory są naszą **bazą wiedzy**.
- Mamy też fakt, o którym chcemy się czegoś dowiedzieć. Zakładamy jego prawdziwość lub fałszywość i chcemy tego dowieść. W tym celu robimy dowód nie wprost: do naszej bazy wiedzy dodajemy negację naszego faktu nieobserwowalnego.
- Dobieramy kolejno klauzule z bazy wiedzy w pary i przeprowadzamy wnioskowanie za pomocą reguły rezolucji, w wyniku czego powstają coraz to kolejne – z reguły prostsze klauzule, które także dodajemy do bazy wiedzy.
- Baza nam rośnie do momentu, gdy nie zauważymy w niej klauzul postaci q oraz $\neg q$. Oznacza to, że w bazie pojawia się sprzeczność, co za pomocą dowodu nie wprost, wyjaśnia, że postawiona przez nas teza była prawdziwa. Jeżeli jednak, pomimo sprawdzenia każdej możliwości za pomocą reguły rezolucji, nie jesteśmy znaleźć takiej pary – nasza teza zostaje obalona.

3. Implementacja

W katalogu `/implementacja` znajdują się dwa skrypty w kodzie Pythona:

- *`reasoning.py`* – zawiera implementację algorytmu wnioskowania przez rezolucję oraz zaprzeczenie oraz klasy, które pomagają w jej implementacji.
- *`test.py`* – zawiera wnioskowanie ostatniego przykładu z tabeli

Klasa `Fact` zawiera implementację reprezentacji faktu, która ma pola odpowiedzialne za nazwę oraz za jego prawdziwość (czy jest zanegowany, czy nie). Jego metody ułatwiają dalszą pracę z obiektem tej klasy. Zaimplementowane są: test równości dwóch faktów oraz metoda, która zwraca nowy – zanegowany fakt.

Klasa `Clause` zawiera implementację reprezentacji klauzuli. Jest przedstawiona za pomocą listy faktów, które są w alternatywie. Jej metody odpowiedzialne są za: dodawanie dwóch klauzul (alternatywa dwóch klauzul), odejmowanie z klauzuli faktu oraz jego usuwania (odejmowanie zwraca nową klauzulę, usuwanie operuje na danym obiekcie), test równości dwóch klauzul oraz test czy dana klauzula jest atomem.

Funkcja `resolution()` zwraca listę możliwych wyników działania rezolucji na dwóch klauzulach.

Funkcja `reasoning()` implementuje algorytm wnioskowania przez rezolucję opisany w punkcie 2, z dokładnością do faktu, że zwraca nam tylko wiadomość typu: „C is False”, bądź „C is True”, a nie „not C is True”, czy „not C is False”,

4. Rozwiązywanie przykładowych problemów wnioskowania

Do zbadania były problemy przedstawione w tabeli:

	Klauzule	Fakty obserwowalne	Fakt nieobserwowalny
1.	$\neg A \vee B$	A	B
2.	$\neg A \vee B$	$\neg A$	B
3.	$\neg A \vee C$ $\neg B \vee C$ $A \vee B$	$\emptyset$	C
4.	$\neg A \vee B \vee C$ $\neg B \vee D$ $\neg C \vee \neg D$ $C \vee D$ $\neg C \vee E$ $\neg D \vee \neg E$	E	$\neg B$
5.	$A \vee B \vee C$ $\neg A \vee D$ $\neg B \vee D$ $\neg C \vee \neg E \vee F$ $D \vee E \vee F$ $A \vee \neg F$ $\neg D \vee \neg E$	$\neg A, B$	C

Wyniki są następujące:

1. Prawda
2. Fałsz
3. Prawda
4. Prawda
5. Fałsz