ANALIZATOR WNIOSKOWANIA W ROZMYTYM JĘZYKU ZAPYTAŃ

MAGDALENA KRAKOWIAK
Zachodniopomorski Uniwersytet Technologiczny

Streszczenie

W artykule zaprezentowano rozwiązanie dotyczące zastosowania wnioskowania przybliżonego w systemach wspomagania decyzji. W ramach przeprowadzonych badań na podstawie własnych definicji zapytania rozmytego i definicji reguły logicznej, opracowano algorytm analizatora wnioskowania. Stanowi on część analizatora zapytania w zaprojektowanym modelu interaktywnego rozmytego języka zapytań MELSQL.

Słowa kluczowe: wnioskowanie przybliżone, systemy wspomagania decyzji, rozmyty język zapytań, rozmyte reguły logiczne, schematy wnioskowania MP, MT, UMP i UMT

1. Wprowadzenie

Współczesne systemy baz danych ukierunkowane są głównie na wspomaganie podejmowania decyzji, co jest istotnym czynnikiem zarządzania. Oczekiwane wsparcie ze strony systemu informatycznego możemy rozpatrywać w dwóch aspektach. Po pierwsze system powinien pozwolić na sprawne wyszukiwanie informacji wg wszystkich kombinacji kryteriów i umożliwić znalezienie każdej danej. Jest to realizowane poprzez pełność języka zapytań. Drugi aspekt wspomagania decyzji to wnioskowanie na podstawie określonych reguł logicznych. Wsparcie na tym poziomie związane jest ze złożonym procesem odkrywania wiedzy, a reguły logiczne pozwalają na efektywną i silną jej reprezentację.

Powyższe dwa poziomy wsparcia systemów informatycznych możemy rozpatrywać w kategorii zapytań precyzyjnych i rozmytych.[1] W pierwszym przypadku wyszukiwanie informacji w bazie danych (zapytania twarde) realizowane są przy użyciu języka SQL, a do komunikacji użytkownika z systemem odkrywania wiedzy służy generacja regułowych języków zapytań (np. MineSQL). Stanowią one rozszerzenie języka SQL o nowe typy danych, polecenia i operatory, co ma swoje uzasadnienie w jednoczesnej obsłudze dwóch typów zapytań. Użytkownik może budować klasyczne zapytania w celu wyszukiwania informacji oraz specyfikować poszukiwane reguły i dane, które muszą być eksplorowane w celu odkrycia tych reguł.

W przypadku obsługi zapytań rozmytych wyszukiwanie informacji realizują rozmyte języki zapytań. Przykładem takiego rozwiązania bez obsługi złożonych reguł jest system FQUERY for Access [7]. Natomiast rozmyty język MELSQL stanowi autorskie rozwiązanie mające m.in. na celu implementację reguł rozmytych czyli użycie informacji nieprecyzyjnych w przesłankach i konkluzjach. Opracowany model łączy w sobie funkcjonalność systemów regułowych z obsługą zapytań rozmytych.

2. Wnioskowanie przybliżone

W szeroko stosowanych w praktyce systemach rozmytych, zawierające się w poprzednikach i następnikach reguł zmienne lingwistyczne (reprezentowane przez zbiory rozmyte) odpowiadają wartościom wejściowym i wyjściowym systemu. Tego typu rozwiązania mają zastosowanie głównie w modelowaniu i sterowaniu rozmytym, związanych z wnioskowaniem przybliżonym (ang. approximate reasoning) [5]. Jest ono oparte na sprawdzonej i powszechnie stosowanej w praktyce regule Uogólniony Modus Ponens (UMP).

Uogólniony Modus Ponens stanowi rozmyty odpowiednik klasycznego sposobu wnioskowania Modus Ponens[1], którego podstawowe założenia dotyczą przyjmowania tylko wartości logicznej TRUE (1) lub FALSE (0) przez przesłankę (x=A) i konkluzję (y=B) oraz całkowitej zgodności stwierdzonego faktu z użytą do wnioskowania przesłanką implikacji:

JEŚLI (x=A) TO (y=B).

(2.1)

Zastosowanie reguły Uogólniony Modus Ponens daje możliwość użycie rozmytego sformułowania zaobserwowanego faktu:

$x=A^*$,

(2.2)

gdzie A^* może być zbiorem rozmytym „częściowo A" z przestrzeni lingwistycznej lub numerycznej zmiennej x, co pozwala na wyciągnięcie wniosku na podstawie implikacji (2.1), że:

$y=B^*$,

(2.3)

gdzie B^* może oznaczać zbiór rozmyty „mniej więcej B" z przestrzeni lingwistycznej lub numerycznej zmiennej y.

Poza klasyczną regułą wnioskowania Modus Ponens, w logice dwuwartościowej stosuje się także m.in. tryb Modus Tollens[2]. Ten schemat wnioskowania logicznego polega na zaprzeczeniu faktu przy założeniu, że konkluzja jest nieprawdziwa; „tryb obalający [...] przez obalenie" [3]. Zatem przy uznaniu implikacji (2.1) i założeniu, że $y \neq B$ wnioskujemy, że $x \neq A$.

Reguła Modus Tollens, tak jak tryb Modus Ponens, ma również swoją rozmytą wersję – Uogólniony Modus Tollens (UMT). Jej przesłankę stanowi wniosek wyciągnięty zgodnie z regułą UMP (2.3), a z kolei wniosek odpowiada zaobserwowanemu faktowi (2.2).

W wyniku analizy poszczególnych schematów wnioskowania wyraźnie zauważa się zasadniczą różnicę pomiędzy klasycznymi i rozmytymi regułami dowodzenia. W logice klasycznej, aby móc przeprowadzić wnioskowanie (zastosować regułę), fakt (x=A) musi całkowicie odpowiadać przesłance w implikacji i możliwy jest tylko wówczas wniosek zgodny z konkluzją implikacji (y=B). W przypadku rozmytych reguł dowodzenia do przeprowadzenia wnioskowania nie jest wymagana całkowita zgodność przesłanki i wniosku z implikacją, gdyż możemy operować na różnych

[1] *Sposób potwierdzający przez potwierdzenie (łac. Modus ponendo ponens)*
[2] *Sposób zaprzeczający przy pomocy zaprzeczenia (łac. Modus tollendo tollens)*

zbiorach. Zbiory te muszą jednak pochodzić z tej samej przestrzeni zmiennych użytych w przesłance i konkluzji.

Z różnicy wymagań co do zgodności przesłanki i wniosku z implikacją pomiędzy przedstawionymi powyżej typami reguł, można wywnioskować, że zastosowanie rozmytych reguł dowodzenia w zapytaniach do baz danych wyspecjalizowanych systemów wspomagania decyzji znacznie wpłynie na zwiększenie ich gotowości informacyjnej. Niejednokrotne użycie w zapytaniu nieznanych sformułowań, których brak także w bazie reguł pozwoli na wyciągnięcie wniosków, co zdecydowanie zwiększy efektywność odpowiedzi.

Przykładowo szukając bardzo dobrych klientów i odnotowując bardzo duże zakupy klienta X, przy braku w bazie reguł implikacji z dokładnie takim poprzednikiem (przesłanką), możemy zastosować regułę:

JEŚLI (klient ma duże zakupy) TO (klient jest dobry), (2.4)

zgodnie z którą wyciągniemy wniosek, że klient X jest bardzo dobry.

W rozważanym przypadku mamy następujące parametry podstawowe reguły:

x - zmienna lingwistyczna „zakupy klienta”,
y - zmienna lingwistyczna „ocena klienta”,
A - zbiór rozmyty „duże zakupy klienta”,
A^* - zbiór rozmyty „bardzo duże zakupy klienta”,
B - zbiór rozmyty „dobry klient”,
B* - zbiór rozmyty „bardzo dobry klient”,

które uzupełniamy o zbiory rozmyte wszystkich wartości zmiennych (przestrzenie rozważań zmiennych):

T_x - zbiór wartości zmiennej x = („bardzo małe”, „małe”, „średnie”, „duże”, „bardzo duże”),
T_y - zbiór wartości zmiennej y = („bardzo słaby”, „słaby”, „średni”, „dobry”, „bardzo dobry”).

Wyciągnięty wniosek, że klient X jest bardzo dobry to wynik wnioskowania zgodnie z regułą Uogólniony Modus Ponens. Należy zauważyć, że nie zawsze otrzymane w ten sposób wnioski są prawdziwe. Wynika to z faktu, że nie wszystkie reguły posiadają własności ekstrapolacyjne, które są warunkiem stosowania trybu UMP [2].

Uznając implikację opisaną wzorem (2.4) przy stwierdzeniu, że klient X nie jest dobry, musimy uznać także fakt, że klient X nie ma u nas dużych zakupów. Wynika to ze schematu wnioskowania klasycznej reguły Modus Tollens. Chcąc zobrazować działanie jej rozmytej wersji (UMT) zakładamy stwierdzenie, że klient X jest słaby. Przebieg wnioskowania może wyglądać następująco:

(klient X jest słaby) -> (klient X nie jest dobry) -> (klient X nie ma dużych zakupów) -> (klient X ma małe zakupy).

Wnioskowanie (inferencja) stanowi główną część modelu rozmytego systemu (rys.1). Poprzedza go blok rozmywania (fuzyfikacji), w którym dokonywane jest obliczanie stopni przynależności

do poszczególnych zbiorów rozmytych wejść. Na ich podstawie w bloku wnioskowania, w trzech kolejno występujących po sobie etapach, obliczana jest tzw. wynikowa funkcja przynależności wyjścia modelu. Pierwszy z etapów wymaga dokonania oceny przesłanek poszczególnych reguł czyli określenia stopnia ich spełnienia wartością z przedziału [0,1] w przeciwieństwie do reguł logiki klasycznej, których stopień mógł przyjmować tylko wartość 0 lub 1.

Sposób realizacji tego zadania zależny jest od postaci przesłanki, wśród których wyróżnia się następujące typy:

- przesłanki proste
  JEŚLI (x=A)

  (2.5)
- koniunkcyjne przesłanki złożone
  JEŚLI $(x_1=A_1)$ I $(x_2=B_2)$

  (2.6)
- alternatywne przesłanki złożone
  JEŚLI $(x_1=A_1)$ LUB $(x_2=B_2)$.

  (2.7)

Alternatywne przesłanki złożone (2.7) cechują m.in. reguły powstałe w wyniku agregacji wielu reguł o takiej samej konkluzji (następniku).

Stopień spełnienie przesłanek prostych (2.5) odpowiada stopniowi przynależności wartości x do zbioru A. Natomiast dla przesłanek złożonych koniunkcyjnych (2.6) i alternatywnych (2.7) należy obliczyć odpowiednio funkcję przynależności iloczynu (ang. intersection) i sumy (ang. union) zbioru. Do realizacji tych operacji zaproponowano po raz pierwszy operatory MIN i MAX [6].

Operator MIN uwzględnia tylko fakt mniejszości jednego ze stopni przynależności od drugiego, z czego wynika jego prostota i szybkość obliczeń. Nie branie pod uwagę wartości różnicy pomiędzy poszczególnymi stopniami przynależności powoduje jednak dużą utratę informacji. Ogranicza to znacznie zakres stosowania operatora MIN i coraz częściej zastępuje się go operatorem iloczynu PROD [5], który zależy ilościowo od wartości obydwu składowych funkcji przynależności. Operatory stosowane do realizacji operacji przecięcia zbioru nazywane są operatorami t-normy [2] i stanowią funkcję modelującą operację iloczynu (I/AND) dwóch zbiorów rozmytych.

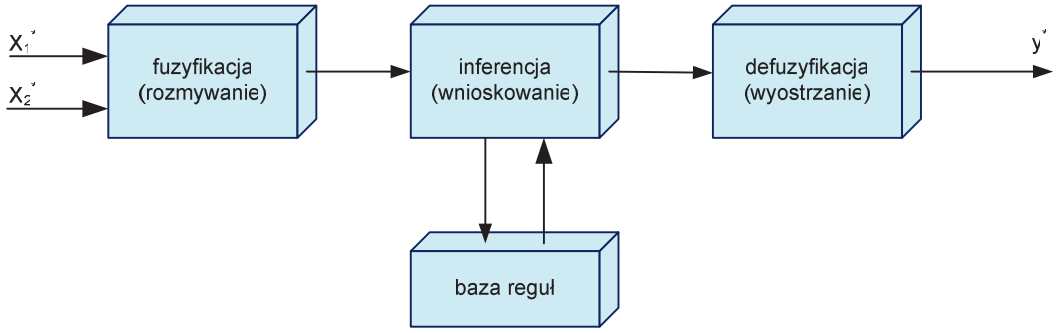
Operatory komplementarne do operatorów t-normy to operatory s-normy modelujące operację sumy (LUB/OR) dwóch zbiorów rozmytych. Jednym z nich jest operator MAX (komplementarny do operatora MIN), uwzględniający tylko większość jednego ze stopni przynależności od drugiego.

Poza przedstawionymi przesłankami złożonymi (2.6 i 2.7) istnieje również możliwość wystąpienia bardziej złożonej formy przesłanki:

JEŚLI $(x_1=A_1)$ I $(x_2=B_2)$ LUB $(x_1=A_2)$ I $(x_2=B_1)$, (2.8)

której stopień spełnienia oblicza się wykonując najpierw operację I (stosując operator t-normy), a potem LUB (stosując operator s-normy).

W drugim etapie działania bloku wnioskowania określa się funkcję przynależności konkluzji poszczególnych reguł dla danych wartości wejść modelu [5]. Stopień ich aktywizacji pozwala w trzecim etapie określić postać wynikową funkcji przynależności zwana akumulacją.



Rys. 1. Struktura modelu rozmytego

Źródło: opracowanie własne na podstawie [5]

Ostatni blok modelu rozmytego to defuzyfikacja (wyostrzanie) polegająca na określania ostrej wartości wyjścia y^*. Do przeprowadzenia tej operacji stosowana jest, zależnie od specyfiki modelu i tym samym wyboru kryterium, jedna z następujących metod: środka maksimum SM (ang. Middle of Maxima), pierwszego maksimum PM (ang. First of Maxima), ostatniego maksimum OM (ang. Last of Maxima), środka ciężkości SC (ang. Center of Gravity), środka sum SS (ang. Center of Sum) i metoda wysokości W (ang. Height Method).

3. Definicja reguł rozmytych

Podstawowe założenia opracowanego modelu języka zapytań MELSQL, związane z wnioskowaniem, dotyczą przede wszystkim możliwości tworzenia i przetwarzania reguł logicznych składających się ze złożonych przesłanek alternatywnych oraz koniunkcyjnych, a także ich bardziej złożonych form (2.8). Zakłada się, że poszczególnym przesłankom prostym odpowiadają predykaty proste. Stanowią one także główny składnik definicji zapytania rozmytego, czego konsekwencją jest ich złożona postać uwzględniająca cechy decydujące o charakterze rozmytym zapytania.

Z założenia zapytanie rozmyte charakteryzuje się wystąpieniem w nim przynajmniej jednego z elementów [4]:

- wartości rozmyte atrybutów (wartości lingwistyczne, liczby przybliżone)
- rozmyte operatory arytmetyczne
- rozmyte kwantyfikatory
- operatory kompensacyjne.

Pierwsze trzy cechy muszą mieć odzwierciedlenie w definicji samego predykatu, który w wyniku tego przyjmuje następującą postać [4]:

$$P = X_k \, KA \, X_o \, O \, X_w \, W, \tag{3.1}$$

gdzie:

X_k - wartość podobieństwa dla kwantyfikatora K

K - kwantyfikator twardy (istnieje, wszyscy, żaden) lub rozmyty (prawie wszyscy, prawie żaden)

A - atrybut

X_o - wartość podobieństwa dla operatora arytmetycznego O

O - operator arytmetyczny twardy (=, ≠, <, >, =>, =<) lub rozmyty (prawie =, prawie ≠, prawie <, prawie >)

X_w - wartość wg rozkładu możliwości/podobieństw dla warunku

W - wartość atrybutu precyzyjna lub rozmyta (wartość lingwistyczna, liczba przybliżona).

W związku z powyższym w języku MELSQL proponuje się następującą postać reguły rozmytej:

$$\text{JEŚLI} \quad (P_1\ X_{ok}\ O_l\ P_2...) \quad \text{TO} \quad A\ X_o\ O\ X_w\ W, \tag{3.2}$$

gdzie:

P - predykat prosty

X_{ok} - współczynnik operatora kompensacyjnego (waga, próg reakcji)

O_l - operator logiczny (AND, OR)

A - atrybut lub funkcja agregująca (count(), min(), max(), avg(), sum())

X_o - wartość podobieństwa dla operatora arytmetycznego O

O - operator arytmetyczny twardy (=, ≠, <, >, =>, =<) lub rozmyty (prawie =, prawie ≠, prawie <, prawie >)

X_w - wartość wg rozkładu możliwości/podobieństw dla warunku

W - wartość atrybutu precyzyjna lub rozmyta (wartość lingwistyczna, liczba przybliżona).

Przedstawiona definicja reguły odpowiada założeniom dotyczącym budowy jej poprzednika i następnika. Z określonej wzorem 3.2 implikacji wynika, że do tworzenia poprzednika można stosować przesłanki złożone, ale następnik stanowi przesłanka prosta.

Występujące w przesłankach złożonych operatory logiczne AND i OR mogą podlegać modyfikacjom zgodnie z zasadami kompensacji, jakie naturalnie stosują ludzie w zależności od nastroju czy sytuacji. Użytkownik może zatem określić stopień kompensacji współczynnikiem zmieniającym swoją wartość w zakresie od 0 do 1 zadając akceptowalną granicę wartości spełnienia przesłanki złożonej.

Operatory t-normy wyliczające przecięcie (iloczyn logiczny) i operatory s-normy wyliczające sumę logiczną zbiorów rozmytych są tzw. operatorami domniemanymi dającymi cały wachlarz możliwości otrzymanych wyników (przypuszczeń co do sposobu realizacji tych operacji).

4. Analizator wnioskowania

Analizator wnioskowania stanowi jeden z głównych bloków operacji w procesie analizy zapytań opracowanego języka MELSQL. Jego podstawowa funkcja związana jest z realizacją obsługi zapytania w sytuacji, gdy użytkownik użył nieznanego dla systemu sformułowania. W analizatorze predykatu, będącego częścią analizatora zapytania, w przypadku identyfikacji nieznanej wartości

lingwistycznej (dla której brak określonych wartości funkcji przynależności) następuje przejście do analizatora wnioskowania. Wówczas, jako pierwszy etap, przeszukiwana jest baza reguł w celu sprawdzenia czy analizowany predykat nie stanowi prawej strony reguły użytkownika. Gdy tak jest i nie zmieniły się preferencje oraz ocena sytuacji pytającego, predykat prosty zostaje zamieniony w lewą stronę potwierdzonej reguły i następuje powrót do analizatora predykatu. Odnalezienie wielu reguł o pożądanej konkluzji nie jest możliwe, gdyż system w celu redukcji liczby rejestrowanych implikacji dokonuje automatycznej ich agregacji.

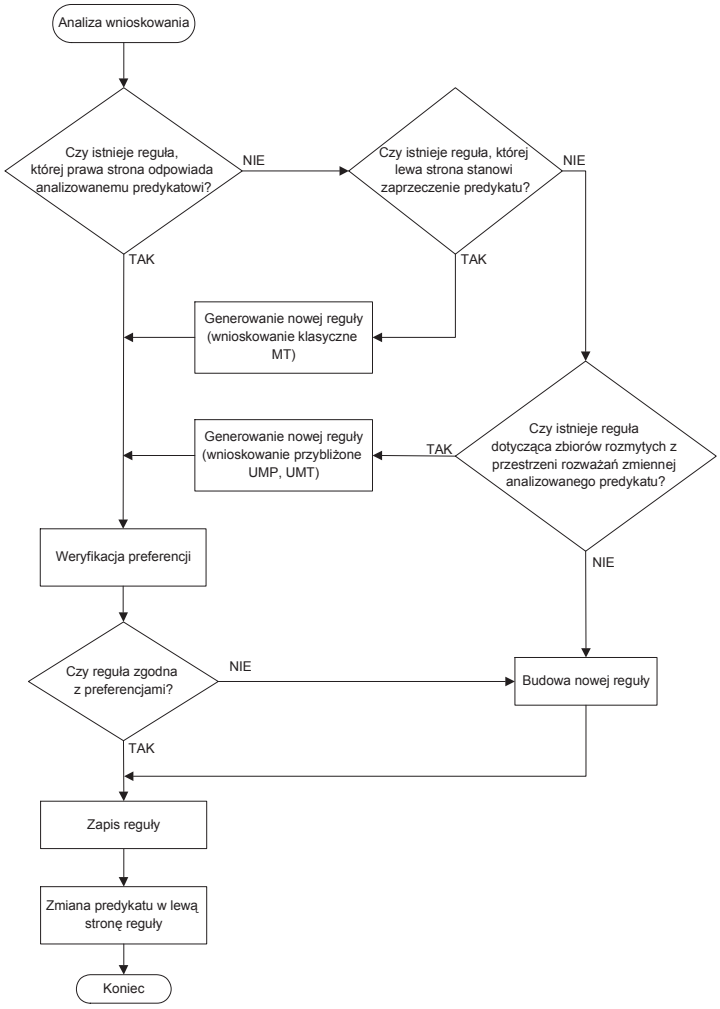
Przykładowo, gdy w zadanym przez użytkownika zapytaniu wystąpi nieznane pojęcie dobry klient (w systemie nie zdefiniowano funkcji przynależności klienta do zbioru dobrych klientów), wówczas nie ma konieczności budowy nowej funkcji, ale istnieje możliwość skorzystania z reguły zgodnej z implikacją jeśli klient dużo kupuje to klient jest dobry (2.4), czego następstwem będzie zamiana predykatu klient jest dobry na klient dużo kupuje. Analizator predykatu wykorzysta wówczas zdefiniowaną funkcję przynależności zakupów klienta do zbioru zakupów dużych. Powyższy proces jest zgodny z klasyczną regułą wnioskowania MP, w której przesłanka i wniosek muszą być zgodne z implikacją.

Gdy w systemie nie ma zarejestrowanej reguły, której następnik byłby zgodny z analizowanym predykatem, należy szukać reguły, której lewa strona stanowi zaprzeczenie predykatu. Jeżeli zostanie taka znaleziona to na jej podstawie generowana jest nowa implikacja i poddana weryfikacji przez użytkownika, gdyż system uwzględnia ewentualną zmianę w czasie jego preferencji. Akceptacja nowej reguły pozwala na jej zapis i zamianę predykatu w jej lewą stronę.

Kontynuując zapytanie o dobrego klienta i odnalezienie w bazie reguł na przykład implikacji jeśli klient nie jest dobry to klient nie kupuje dużo powoduje utworzenie przez system nowej reguły zgodnie ze schematem wnioskowania MT. Wówczas, korzystając z prawa podwójnego przeczenia (nie nie kupuje dużo <=> kupuje dużo, nie nie jest dobry <=> jest dobry) nowopowstała reguła przyjmie postać implikacji (2.4).

Brak w bazie reguł spełniających jeden z dwóch opisanych warunków (analizowany predykat stanowi następnik reguły lub jego zaprzeczenie jest poprzednikiem reguły) powoduje ustalenie nowego kryterium wyszukiwania – zbiorów rozmytych z przestrzeni rozważań zmiennej analizowanego predykatu. Odnalezienie reguły, której lewa lub prawa strona operuje na takim zbiorze pozwala na wygenerowanie na jej podstawie nowej reguły. Należy pamiętać, że warunkiem otrzymania prawdziwych wyników w tym przypadku są własności ekstrapolacyjne reguł i dlatego szczególną rolę odgrywa tu weryfikacja użytkownika. Po jego akceptacji następuje zapis nowej implikacji i zamiana predykatu w jej lewą stronę.

Analizując przykładowy predykat dotyczący dobrych klientów możemy skorzystać z dowolnej reguły, w następniku lub poprzedniku której występuje zbiór rozmyty wartości zmiennej klient ("bardzo słaby", "słaby", "średni", "dobry" i "bardzo dobry"). Zarejestrowana w bazie reguła jeśli klient bardzo dużo kupuje to klient jest bardzo dobry pozwala zgodnie ze schematem UMP wygenerować implikację (2.4). Tę samą implikację otrzymamy korzystając z trybu UMT na podstawie reguły jeśli klient jest bardzo dobry to klient bardzo dużo kupuje. Zbiory rozmyte “bardzo duże zakupy” i “duże zakupy” oraz “bardzo dobry klient” i “dobry klient” znajdują się w tych samych przestrzeniach rozważań zmiennych “zakupy” i “klient”, zatem spełniają podstawowy warunek generowania nowych reguł.



Rys.2. Algorytm działania analizatora wnioskowania

Źródło: opracowanie własne

W przypadku braku odpowiedniej reguły, zmiany preferencji czy też odrzucenia przez użytkownika wygenerowanych rozwiązań, moduł MELSQL umożliwia budowę nowej reguły. Prawą jej stronę stanowi analizowany predykat prosty. Zgodnie z definicją (wzór 3.2) lewą stronę reguły może stanowić przesłanka prosta lub złożona, składająca się z predykatów prostych. Do konstrukcji predykatów prostych i łączących ich operatorów w fazie tworzenia poprzednika nowej reguły wykorzystywany jest, opracowany w modelu języka MELSQL, blok selekcji kreatora zapytań.

Akceptacja nowoutworzonej reguły pozwala na jej zapis i użycie, w wyniku czego analizowany predykat prosty przyjmuje postać jej lewej strony i w większości przypadków staje się predykatem złożonym.

5. Podsumowanie

We współcześnie projektowanych systemach informatycznych język zapytań odgrywa kluczową rolę. Stanowi on podstawowy element komunikacji pomiędzy człowiekiem a komputerem. Sprawna i efektywna obsługa zapytań użytkownika zwiększa niewątpliwie gotowość informacyjną systemu, co daje przewagę nad konkurencją. Stąd dążenia do takiego zamodelowania systemu do gromadzenia, przetwarzania i udostępniania danych, aby w komunikacji człowiek-komputer zbliżyć się do możliwe naturalnego dialogu.

Stosowane głównie w modelowaniu i sterowaniu rozmytym, wnioskowanie przybliżone znacznie zwiększa efektywność działania analizy zapytań opracowanego modelu języka MELSQL. Przede wszystkim umożliwia uzyskanie większej ilości pełnej informacji oraz pozwala na wyciągnięcie wniosków nawet w przypadku użycia sformułowań, których brak w bazie reguł. Z pewnością zwiększona efektywność odpowiedzi wpływa na podniesienie gotowości informacyjnej systemu wspomagania decyzji.

Bibliografia

1. Budziński R., Krakowiak M.: Modelowanie zapytań i bazy reguł w regułowym języku zapytań z wykorzystaniem logiki rozmytej. W: Studia i Materiały Polskiego Stowarzyszenia Zarządzania Wiedzą, nr 13, Bydgoszcz 2008, s.5-15.
2. Knappe H.: Nichtlineare Regelungstechnik und Fuzzy-Control.Renningen-Malmscheim, BRD, Export Verlag 1994.
3. Kotarbiński T.: Elementy teorii poznania, logiki formalnej i metodologii nauk, PWN, Warszawa 1986.
4. Lipińska M.[3]: Metoda transformacji zapytań na zapytania w standardzie SQL w bazach danych. W: Materiały VIII Sesji Naukowej Informatyki., Szczecin 2003.
5. Piegat A.: Modelowanie i sterowanie rozmyte, Akademicka Oficyna Wydawnicza Exit, Warszawa 1999.
6. Zadeh L.A.:Fuzzy sets. W: Infomation and Control, vol. 8, 1965. Pp.338-353.
7. Zadrożny S.: Zapytania nieprecyzyjne i lingwistyczne podsumowanie baz danych, Akademicka Oficyna Wydawnicza Exit, Warszawa 2006.

[3] *Publikacja własna pod poprzednim nazwiskiem*

ANALYSER OF INFERENCE IN FUZZY QUERY LANGUAGE

Summary

The paper presents a solution concerning of using approximate reasoning in Decision Support System (DSS). Making in the work analysis, upon individual fuzzy query definition and fuzzy logic rules definition, was presented algorithm of analyser of inference. It is a part of query's analyser in designed model of interactive query language using fuzzy logic MELSQL.

Keywords: approximate reasoning, computer decisions making systems, fuzzy query language, fuzzy logic rules, reasoning patterns MP, MT, UMP and UMT

Magdalena Krakowiak
Wydział Informatyki
Zachodniopomorski Uniwersytet Technologiczny
e-mail: makrakowiak@wi.ps.pl