ROCZNIKI FILOZOFICZNE
Tom XLVI-XLVII, zeszyt 1 – 1998-1999

PIOTR KULICKI
Lublin

REPREZENTACJA WIEDZY NEGATYWNEJ PRZY OPRACOWANIU KOMPUTEROWYM FRAGMENTU PISMA ŚWIĘTEGO[1]

W ostatnich latach zasięg stosowania metod komputerowych rozszerzył się i obejmuje oprócz zagadnień tradycyjnie związanych z komputerami także automatyczne przetwarzanie tekstów w języku naturalnym. W całości tej problematyki można wyróżnić trzy podstawowe zagadnienia: pozyskiwanie informacji zawartej w tekście, reprezentacja wiedzy w ten sposób uzyskanej oraz udostępnianie jej użytkownikowi. W niniejszej pracy zajmiemy się problemem należącym do drugiej z wymienionych grup – reprezentacją wiedzy negatywnej zawartej w tekście. Przez wiedzę negatywną rozumiemy tu wiedzę o tym, że jakiś stan rzeczy *nie* ma miejsca, że jakaś relacja *nie* zachodzi.

Problem reprezentacji wiedzy negatywnej pojawił się przy próbie komputerowej reprezentacji treści fragmentu Pisma św. (Rdz 11, 10-25, 19). Do opracowania wybrano tekst biblijny, ponieważ z jednej strony jest treściowo bogaty i różnorodny, informacja w nim zawarta jest niepełna i przedstawiona często w formie nie uporządkowanej, z drugiej zaś strony jest on dobrze znany i wielostronnie opracowany. Pozwala to na szerokie zastosowanie wypracowanych metod komputerowego opracowania oraz korzystanie z dorobku różnorodnych dyscyplin związanych z analizą tekstu.

[1] Artykuł ten jest kontynuacją badań przedstawionych w pracy magisterskiej autora, w której w nieco innej wersji przedstawiony jest zasadniczy rezultat niniejszej pracy. Tam też znajduje się pełny tekst programu implementującego ten rezultat w języku Turbo Prolog 2.0.

W wybranym fragmencie tekstu dużo miejsca poświęca się genealogii. Jej reprezentacja stanowi punkt wyjścia rozważań w tej pracy. Genealogia osób biblijnych jest stosunkowo mało skomplikowana i łatwa do wyodrębnienia z całości tekstu, jednocześnie jednak jej komputerowa reprezentacja dostarcza ciekawych problemów. Rozważając podstawowe relacje rodzinne bycia ojcem, matką, przodkiem czy krewnym zachodzące pomiędzy osobami występującymi w tekście, dochodzimy do tego, że z tekstu wynika, iż relacje te pomiędzy niektórymi z wymienionych osób zachodzą, pomiędzy innymi z pewnością nie zachodzą, a w jeszcze innych wypadkach tekst nie pozwala na rozstrzygnięcie tej kwestii. I tak na przykład, korzystając z tego, co jest napisane w tekście, oraz minimalnej wiedzy bazowej dotyczącej relacji rodzinnych, możemy stwierdzić, że Noe jest przodkiem Abrahama (w tekście podany jest łańcuch genealogiczny, prowadzący od Noego do Abrahama), Abraham jest krewnym Lota (mają wspólnego przodka) oraz że Izaak nie jest przodkiem Abrahama (ponieważ jest jego synem), a Abraham nie jest matką Ismaela (bo nie jest kobietą). Tekst pozostawia także niektóre pytania bez odpowiedzi, np. jaka jest relacja pomiędzy Abrahamem a Melhizedekiem. Celem niniejszej pracy jest znalezienie takiej reprezentacji komputerowej dla genealogii, która odpowiadałaby zawartości tekstu, w szczególności pozwalała na rozróżnienie pomiędzy wiedzą negatywną a niewiedzą. Proponowane rozwiązanie opiera się na metodzie aksjomatycznego odrzucania przedstawionej przez J. Łukasiewicza (por. [5]).

W punkcie pierwszym zajmiemy się źródłami informacji negatywnej. W punkcie drugim przedstawimy proponowaną reprezentację dla informacji negatywnej i system wnioskowania z niej korzystający. Punkt trzeci poświęcony jest związkom z innymi pracami i alternatywnym rozwiązaniom. Punkt czwarty zawiera konkluzje oraz uwagi na temat perspektyw dalszych prac.

I. ŹRÓDŁA INFORMACJI NEGATYWNEJ

Aby rozważać problem reprezentacji wiedzy negatywnej na temat relacji rodzinnych, prześledzimy najpierw możliwe źródła takiej wiedzy. Znaleziono trzy rodzaje informacji pozwalające w przypadku genealogii na stwierdzenie, że relacja nie zachodzi:

- własności zbioru relacji rodzinnych,
- pełność informacji w pewnych fragmentach,
- związki z chronologią.

W pierwszej grupie wiedzę negatywną czerpiemy z analizy własności rozpatrywanych relacji rodzinnych. Prowadzi ona do stwierdzenia, że pewne stany rzeczy wykluczają się. Jeśli więc jedna z takich wykluczających się relacji zachodzi, druga zachodzić nie może. Odnaleziono następujące własności relacji rodzinnych mogące być podstawą tego typu zależności:

- asymetryczność relacji bycia przodkiem,
- jedyność ojca i matki,
- przynależność członów relacji do konkretnej płci.

Własności te mają charakter abstrakcyjny i rozwiązanie problemu reprezentacji wiedzy negatywnej wyprowadzonej przy ich wykorzystaniu będzie mogło być przeniesione do innych dziedzin przedmiotowych o podobnych własnościach formalnych.

Druga grupa czynników pozwalających na wyprowadzenie informacji negatywnej jest także ogólna i niezależna od dziedziny przedmiotowej. W tekstach z niepełną informacją mogą występować fragmenty, w których informacja jest pełną. W tych fragmentach, tak jak w pełnych bazach danych, brak pozytywnej odpowiedzi na pytanie równoważny jest z właściwą odpowiedzią negatywną. W rozpatrywanym przypadku genealogii osób biblijnych możemy uznać, że sytuacja taka ma miejsce, gdy wymienione są wszystkie dzieci jakiejś osoby.

Relacje rodzinne, choć dają się wyodrębnić spośród pozostałych informacji w tekście, nie są od nich całkowicie niezależne. Dlatego też można pewne informacje o genealogii uzyskać z fragmentów tekstu nie dotyczących jej bezpośrednio. Najprostszym przykładem tego typu zależności są związki genealogii z chronologią wydarzeń. Bezpośrednio z genealogią związane są zwłaszcza dwa rodzaje wydarzeń: urodzenia i śmierci. Wiadomo, że osoba urodzona później nie może być przodkiem osoby urodzonej wcześniej, matka musi żyć w momencie urodzenia się dziecka, a ojciec na jakiś czas przed nim.

Wszystkie wymienione źródła informacji negatywnej wyprowadzają ją z pozytywnych informacji zawartych w tekście. Sytuacja taka wydaje się najbardziej interesująca z punktu widzenia komputerowej reprezentacji. Mogą oczywiście pojawić się *explicite* w tekście informacje negatywne, komputerowa reprezentacja w tym przypadku wydaje się jednak osobnym problemem.

Spośród trzech wymienionych grup źródeł wiedzy negatywnej zajmiemy się w dalszym ciągu reprezentacją pierwszej. Reprezentacja drugiej grupy nie stwarza istotnych trudności, sprowadza się do reprezentacji wiedzy w pełnej bazie danych. Trzecia grupa dodaje nieco złożoności zagadnieniu, wychodząc

poza ściśle określone fragmenty tekstu, jednak zasadniczo można zastosować tu rozwiązania dotyczące pierwszej grupy.

II. REPREZENTACJA WIEDZY NEGATYWNEJ

Proponowana reprezentacja wiedzy negatywnej zainspirowana jest metodą aksjomatycznego odrzucania dla systemów dedukcyjnych. Metoda ta została stworzona przez Arystotelesa, który budując teorię sylogizmów asertorycznych, użył jej do wykazania pełności swojej sylogistyki (por. [1]). Metoda aksjomatycznego odrzucania została później opisana i rozbudowana przez J. Łukasiewicza i w jego pracach znaleźć można szczegóły (zob. np. [5]). Przypomnimy pokrótce, na czym ona polega, odwołując się do przykładu sylogistyki.

Arystoteles pokazując, że sylogizmy nie należące do jego systemu są nieprawidłowe, nie pokazuje kontrprzykładów dla nich wszystkich, ale dla wybranych; nazwijmy je aksjomatami odrzuconymi. Następnie sprowadza do aksjomatów odrzuconych pozostałe nieprawidłowe sylogizmy. Sprowadzanie to opiera się na wyprowadzalności w systemie. Jeżeli z jakiejś formuły wyprowadzalny jest aksjomat odrzucony, to również ona jest odrzucona.

Dokładniej można to przedstawić następująco. Niech T będzie zbiorem tez systemu, a T^{-1} zbiorem formuł nie będących tezami. Zbiór aksjomatów odrzuconych $Ax^{-1} \subseteq T^{-1}$ jest dobrany w taki sposób, aby dla każdej nie-tezy $\alpha \in T^{-1}$ istniał taki aksjomat odrzucony $\beta \in Ax^{-1}$, że:

$$\beta \in Cn\ (T \cup \alpha).$$

W ten sposób każda formuła α jest tezą lub nie-tezą, tzn. $\alpha \in T \cup T^{-1}$, czyli teoria jest pełna.

Analogiczną technikę możemy zastosować przy naszym problemie z dziedziny reprezentacji wiedzy. Punktem wyjścia będzie dedukcyjna baza danych wyrażona w języku Prolog. Występująca w Prologu negacja zdefiniowana jako niepowodzenie (zob. [4]) nie spełnia wymagań stawianych przed odpowiedzią negatywną przy rozpatrywanym tu problemie, wprowadzamy więc odmienne rozwiązanie. Do bazy danych, jako osobny element, dołączamy nową grupę wyrażeń: wyrażenia odrzucone. Reprezentować one mają wiedzę o wykluczaniu się pewnych stanów rzeczy. I tak, jeśli z jakiegoś zdania, co do którego chcemy się dowiedzieć, czy jest fałszywe, oraz z informacji za-

wartych w bazie danych można wyprowadzić wyrażenie odrzucone, należy zdanie to także odrzucić. Zakładamy przy tym, że sama baza danych jest niesprzeczna i nie da się z niej wyprowadzić wyrażenia odrzuconego. Skoro więc formuły odrzucone są fałszywe, a informacje w bazie danych prawdziwe, zdanie będące przedmiotem zapytania jest fałszywe.

Procedura uzyskiwania odpowiedzi negatywnej na kwerendę przedstawia się następująco:

1. Dołącz do programu formułę będącą przedmiotem kwerendy.
2. Spróbuj udowodnić w tak zmodyfikowanym programie wyrażenie odrzucone:
   (a) Jeżeli uda się udowodnić wyrażenie odrzucone, udziel odpowiedzi *nie*.
   (b) W przeciwnym wypadku nie udzielaj odpowiedzi *nie*.
3. Usuń dołączoną formułę z programu, powracając do jego pierwotnego stanu.

Rozwiązanie takie pozwala na osiągnięcie naszego celu – odróżnienie wiedzy negatywnej od niewiedzy. Cel osiągnięty jest stosunkowo niewielkim kosztem: zachowany jest aparat wnioskujący Prologu, a w programie obsługi bazy danych dokonana jest niewielka modyfikacja dotycząca odpowiedzi negatywnej. Problemem pozostaje znalezienie wyrażeń odrzuconych stanowiących optymalną reprezentację negatywnej wiedzy w konkretnym przypadku oraz efektywność otrzymanej procedury.

Wyrażenia odrzucone muszą być dobrane tak, aby można je było wyprowadzić z wszelkich innych fałszywych wyrażeń. Z drugiej strony reprezentują one wiedzę o wykluczających się stanach rzeczy w reprezentowanej rzeczywistości. W pewnych sytuacjach ich dobór jest oczywisty. W naszym przypadku związków rodzinnych jest tak przy reprezentacji jedyności ojca i matki oraz przynależności członów relacji do płci. Otrzymujemy następujące wyrażenia odrzucone:

1) jedyność ojca i matki:
   (a) rodzic(A,C) i płeć(A,P) i rodzic(B,C) i płeć(B,P) i różne(A,B)[2],
2) przynależność członów pewnych relacji do konkretnej płci:
   (a) ojciec(A,B) i płeć(A,żeńska),
   (b) matka(A,B) i płeć(A,męska),

[2] Przyjmujemy dla formalnego zapisu relacji rodzinnych notację prefiksową, rozumiejąc ją w naturalny sposób, np. rodzic(A,B) oznacza, że osoba A jest rodzicem osoby B. Wielkie litery w pozycji argumentów relacji stanowią zmienne.

(c) syn(A,B) i płeć(A,żeńska),

(d) córka(A,B) i płeć(A,męska).

W przypadku asymetryczności relacji bycia przodkiem reprezentacja jest mniej oczywista, ale za to nieco prostsza. Wystarczy bowiem odrzucić wyrażenie:

przodek(A,A).

Wyrażenie to jest dedukcyjnie najsłabszym rozszerzeniem przyjętej pozytywnej teorii związków rodzinnych. Obserwacja ta może stanowić ogólniejszą wskazówkę dla reprezentacji wiedzy negatywnej przy użyciu proponowanej tu metody. Zawsze szukać należy wyrażenia najsłabszego dedukcyjnie spośród wyrażeń, które chcemy odrzucić, aby można było je wyprowadzić z każdego innego wyrażenia odrzuconego.

Bezpośrednia implementacja przedstawionej metody byłaby nieekonomiczna. Po dołączeniu do bazy danych zdania podlegającego negowaniu w procedurze dla negatywnej odpowiedzi sprawdzana jest niesprzeczność całej zmodyfikowanej bazy danych. Nie jest to konieczne. Proponujemy więc powiązanie predykatu *formuła_odrzucona* z formułami, które podlegać mają odrzucaniu, i sprawdzanie tylko tego fragmentu bazy danych, który powiązany jest z dołączonym w trakcie procedury negowania zdaniem. Powiązanie to w przypadku naszej reprezentacji genealogii opiera się na specyfice rozpatrywanego zagadnienia i nie ma ogólnej wartości, natomiast zrealizowane jest niewielkimi środkami, poprzez modyfikacje definicji predykatu *formuła_odrzucona*, bez zmian w aparacie wnioskującym Prologu.

W rezultacie otrzymujemy program obsługujący bazę danych, odpowiadający na pytania użytkownika: tak, nie lub brak_danych zgodnie z informacją zawartą w tej bazie. Przedstawiony program korzysta z informacji o płci osób oraz o relacji bycia rodzicem zawartych *explicite* w bazie danych. Aby uprościć implementację przedstawionej procedury w Turbo Prologu, dokonamy pewnych modyfikacji w reprezentacji wiedzy w programie. Zamiast używania predykatów reprezentujących poszczególne relacje rodzinne wprowadzimy predykaty *zachodzi* oraz *nie_zachodzi*, poprzez które reprezentować będziemy informację o wszystkich tych relacjach. Argumentami tych predykatów są nazwa relacji rodzinnej oraz jej dwa argumenty. Dodatkowo wprowadzimy predykat *czy_zachodzi*, zawierający trzy argumenty pojawiające się w poprzednich predykatach oraz czwarty argument, stanowiący odpowiedź na kwerendę. Tak określony predykat ma charakter metasystemowy w stosunku do właściwego programu opisującego związki rodzinne. Jest on odpowie-

dzialny za obsługę bazy danych i w bardziej zaawansowanej implementacji mógłby stanowić raczej część kompilatora Prologu niż samego programu napisanego w tym języku.

Predykat ten, dający odpowiedzi na kwerendy, zdefiniowany jest następująco[3]:

```
czy_zachodzi(Relacja,Argument1,Argument2,tak) :-
    zachodzi(Relacja,Argument1,Argument2), !.
czy_zachodzi(Relacja,Argument1,Argument2,nie) :-
    nie_zachodzi(Relacja,Argument1,Argument2), !.
czy_zachodzi(Relacja,Argument1,Argument2,brak_danych).
```

Pierwsza klauzula tej definicji odnosi do pozytywnych definicji relacji rodzinnych. Odwołuje się ona do predykatu *zachodzi:*

```
zachodzi(ojciec,Ojciec,Dziecko) :-
    rodzic(Ojciec,Dziecko), plec(Ojciec,meska).
zachodzi(matka,Matka,Dziecko) :-
    rodzic(Matka,Dziecko), plec(Matka,meska).
zachodzi(syn,Syn,Rodzic) :-
    rodzic(Rodzic,Syn), plec(Syn,meska).
zachodzi(corka,Corka,Rodzic) :-
    rodzic(Rodzic,Corka), plec(Corka,meska).

zachodzi(przodek,Przodek,Potomek) :-
    rodzic(Przodek,Potomek).
zachodzi(przodek,Przodek,Potomek) :-
    rodzic(Przodek,Osoba),
    zachodzi(przodek,Osoba,Potomek).
```

[3] Wszystkie te definicje zapisane są w języku Turbo Prolog 2.0 i stanowią fragmenty działającego programu komputerowego. Ponieważ w niektórych wersjach Prologu pojawiają się kłopoty z niełacińskimi literami (np. w języku polskim z literami *ą*, *ć*, *ę* itd.), specyficznie polskie znaki zostały zastąpione ich łacińskimi odpowiednikami, np. zamiast słowa „płeć" występuje napis *plec*. Aby odczytać deklaratywny sens definicji, należy rozumieć symbol *:-* jako implikację skierowaną od strony prawej do lewej, a przecinki po jego prawej stronie jako koniunkcje. Predykaty *assert*, *retract*, *fail* oraz symbol *!* mają charakter czysto operacyjny. Szczegóły dotyczące używanego w pracy języka można znaleźć np. w pracy [8].

```
zachodzi(potomek,Potomek,Przodek) :-
    zachodzi(przodek,Przodek,Potomek).

zachodzi(krewny,Osoba1,Osoba2) :- zachodzi(przodek,Osoba1,Osoba2).
zachodzi(krewny,Osoba1,Osoba2) :- zachodzi(przodek,Osoba2,Osoba1).
zachodzi(krewny,Osoba1,Osoba2) :-
    -zachodzi(przodek,Osoba3,Osoba1),
    zachodzi(przodek,Osoba3,Osoba2),
    rozne(Osoba1,Osoba2).
```

Kluczowa dla naszego problemu jest klauzula druga, odpowiedzialna za odpowiedź „nie", odwołująca się do predykatu *nie_zachodzi*:

```
nie_zachodzi(Relacja,Argument1,Argument2) :-
    assert(przyjete_roboczo(Relacja,Argument1,Argument2)),
    formula_odrzucona(Relacja,Argument1,Argument2),
    retractall(przyjete_roboczo(_,_,_)), !.
nie_zachodzi(_,_,_) :- retractall(przyjete_roboczo(_,_,_)), fail.
```

Dla realizacji opisanej wyżej procedury potrzebna jest modyfikacja predykatu *zachodzi*, wzbogacająca jego definicję o następujące klauzule:

```
zachodzi(Relacja,Argument1,Argument2) :-
    przyjete_roboczo(Relacja,Argument1,Argument2).
zachodzi(przodek,Przodek,Potomek) :-
    przyjete_roboczo(potomek,Potomek,Przodek).
```

Przyjęte są następujące formuły odrzucone (są to te same formuły, które wymienione zostały wcześniej w formie dostosowanej do struktury programu):

```
% monotonicznosc relacji „przodek":
formula_odrzucona(_,A,_) :- zachodzi(przodek,A,A), !.
formula_odrzucona(_,_, A) :- zachodzi(przodek,A,A), !.

% przyporzadkowanie członom relacji określonej płci:
formula_odrzucona(ojciec,A,_) :- plec(A,meska).
formula_odrzucona(matka,A,_) :- plec(A,meska).
formula_odrzucona(syn,A,_) :- plec(A,meska).
```

```
formula_odrzucona(corka,A,_) :- plec(A,meska).

% jedyność członu relacji „rodzic":
formula_odrzucona(_,A,C) :- rodzic(A,C), plec(A,P),
    rodzic(B,C), plec(B,P), rozne(A,B).
formula_odrzucona(_,C,A) :- rodzic(A,C), plec(A,P),
    rodzic(B,C), plec(B,P), rozne(A,B).
```

Przy tak zdefiniowanym predykacie *formula_odrzucona* procedura dla odpowiedzi „nie" nie wymaga przeszukiwania całości bazy danych, a jedynie fragmentów związanych z dołączonym w trakcie wykonywania procedury zdaniem.

Ostatnia klauzula definicji predykatu *czy_zachodzi* daje odpowiedź „brak_danych" na kwerendę w wypadku, gdy nie da się uzyskać ani odpowiedzi „tak" ani „nie".

III. PRZEDSTAWIONE ROZWIĄZANIE A INNE PRACE Z TEJ DZIEDZINY

Na koniec odniesiemy się do alternatywnych rozwiązań rozważanego problemu i wskażemy na inne prace związane z proponowanym tu rozwiązaniem.

Konkurencyjny sposób reprezentacji wiedzy negatywnej stosowany jest w tzw. rozszerzonym programowaniu w logice (*extended logic programming*, zob. np. [6]). Język Prologu rozszerza się o nowy spójnik negacji (¬), zwany twardą (*hard*) bądź klasyczną negacją. Predykat poprzedzony tym funktorem operacyjnie traktowany jest jak nowy predykat, niezależny od tego samego predykatu bez negacji. Aparat dedukcyjny Prologu pozostaje bez żadnych zmian. Nowy funktor może pojawiać się zarówno w ciele klauzuli, jak i w głowie, zanegowany predykat nie różni się więc, w aspekcie operacyjnym, niczym od innych, zwykłych predykatów. Reprezentuje on natomiast wiedzę o tym, że relacja nie zachodzi.

W reprezentacji genealogii moglibyśmy, na przykład, przyjąć następującą definicję wyrażającą wiedzę o tym, że relacja „przodek" nie zachodzi:

```
¬ przodek(Potomek,Przodek) :-
        przodek(Przodek,Potomek).
¬ przodek(Osoba,Osoba).
```

Nie będziemy tu omawiać szczegółowo relacji pomiędzy rozszerzonym programowaniem w logice a rozwiązaniem proponowanym w tej pracy – stanowią one odrębne zagadnienie; ograniczymy się jedynie do kilku uwag. Przyjęte w tej pracy rozwiązanie posługujące się formułami odrzuconymi wydaje się bliższe naturalnej specyfikacji problemu. Dodaje się do programu ogólne obserwacje dotyczące negatywnych aspektów opisywanego fragmentu rzeczywistości, tego, co w nim nie jest możliwe. Przy budowaniu rozszerzonego programu w logice z obserwacji tych programista musi dopiero wyprowadzić bardziej szczegółowe reguły składające się na negatywne definicje. W rezultacie program taki, przynajmniej w rozpatrywanym w tej pracy zagadnieniu, jest także dłuższy niż przy zastosowaniu formuł odrzuconych. Z drugiej strony wnioskowanie przy użyciu formuł odrzuconych może być mniej ekonomiczne w odpowiadaniu na kwerendy. Konkludując możemy stwierdzić, że oba podejścia mają swoje wady i zalety, a wybór zależy od zastosowania.

Koncepcję negacji w programie w logice bardzo zbliżoną do wykorzystanej w tej pracy przedstawili D. M. Gabbay i M. J. Sergot, nazywając ją „negacja jako sprzeczność" (*negation as inconsistency*, zob. [3]). Autorzy ci wprowadzają do Prologu funktor negacji, definiowany operacyjnie w podobny sposób, jak w niniejszej pracy odpowiedź negatywna na kwerendę. Rolę formuł odrzuconych odgrywają u tych autorów więzy integralności bazy danych. W naszym rozwiązaniu wiedzę negatywną ujmujemy raczej poprzez poznawczy stosunek do zdania, negatywną odpowiedź na kwerendę niż jako funktor negacji w języku. Jest to bliższe fundamentalnemu charakterowi wiedzy negatywnej i pozwala oprzeć się na klasycznych pojęciach mających szeroką podbudowę teoretyczną. Unikamy także bezpośredniego odniesienia do więzów integralności bazy danych, mówiąc raczej o wyrażeniach odrzuconych, ze względu na to, że istota więzów integralności jest wciąż w teorii programowania niejasna i dyskutowana. Zawężenie użycia dyskutowanej tu procedury zastosowane w naszym podejściu pozwala na implementację wystarczająco efektywną dla celów praktycznych.

W proponowanym rozwiązaniu problemu reprezentacji wiedzy negatywnej korzystamy z wyprowadzalności w teorii pierwszego rzędu. Tego typu wnioskowaniami zajmuje się programowanie w logice z użyciem hipotez, zwane też intuicjonistycznym programowaniem w logice (*intuitionistic logic programming*, zob. np. [2]). Prace w tej dziedzinie przynoszą rezultaty zarówno teoretyczne, jak i praktyczne w postaci implementacji rozważanych idei.

Dla podniesienia efektywności przedstawionego rozwiązania zastosować można techniki sprawdzania integralności bazy danych przedstawione w pracach [4] i [7]. Rezultaty tych prac pozwalają na ogólne rozwiązanie tego problemu, my zastosowaliśmy rozwiązanie szczegółowe związane ze specyficznymi własnościami teorii, wystarczające dla postawionych celów.

Technika odrzucania zastosowana w tej pracy jest uogólnieniem techniki budowania dowodów nie wprost. W klasycznym ujęciu dowodu nie wprost dołącza się do dowodu negację dowodzonej formuły i wyprowadza się w dowodzie sprzeczność. W prezentowanej w tej pracy procedurze odrzucania także dołącza się do rozumowania negację dowodzonej formuły: szukając argumentu za negacją formuły, dołącza się tę formułę do programu. Różnica polega na rozszerzeniu pojęcia sprzeczności. W ujęciu klasycznym sprzeczność interpretowana jest jako sprzeczność logiczna, w naszym podejściu może to być także niezgodność z posiadaną wiedzą.

IV. KONKLUZJE I DALSZE PRACE

Związki rodzinne, o których mowa w tekście Pisma św. są skomplikowane, informacja o nich jest niepełna i rozsiana po różnych fragmentach tekstu. W tej sytuacji trudno jest rozróżnić, jakie informacje dają się z tekstu wydobyć, jakie relacje zachodzą, jakie z pewnością nie zachodzą, a o jakich informacji nie ma. Przedstawione rozwiązanie pozwala sytuację tę uporządkować. Na pytania rozstrzygnięcia można uzyskać odpowiedź: „tak", „nie" lub „brak_danych". Kluczowym problemem jest tu rozróżnienie pomiędzy odpowiedzią „nie" a „brak_danych". Przedstawione rozwiązanie odwołuje się do metody aksjomatycznego odrzucania, swymi korzeniami sięgającej Arystotelesa. Jest uogólnieniem techniki dowodu nie wprost. Zastosowana metoda jest ogólna, niezależna od dziedziny przedmiotowej i można ją stosować do reprezentacji w różnych sytuacjach, gdy mamy do czynienia z niepełną informacją.

Praca ma charakter praktyczny – rozwiązania problemu reprezentacji wiedzy negatywnej zawartej w tekście. W związku z tym nie zbudowaliśmy ogólnej teorii dla zastosowanego rozwiązania, a odniesienia do innych prac i konkurencyjnych rozwiązań są raczej powierzchowne. W dalszych pracach należałoby wskazać ogólną teoretyczną podbudowę zastosowanej techniki, dokładniej zbadać jej wady i zalety oraz możliwości aplikacji. Pojawia się także szereg innych problemów, takich jak automatyczne znajdowanie aksjomatów odrzuconych oraz ogólna i efektywna implementacja przedstawionych idei.

BIBLIOGRAFIA

1. A r y s t o t e l e s, Analityki I.
2. B o n n e r A. J., M c C a r t y L. T., V a d a p a r t y K., Expressing Database Queries with Intuitionistic Logic, [w:] Proceedings of the North American Conference on Logic Programming, Cleveland 1989, s. 831-850.
3. G a b b a y D. M., S e r g o t M. J., Negation as Inconsistency, „Journal of Logic Programming", 1986, 1, 1-35.
4. L l o y d W. J., Foundations of Logic Programming, Berlin 1987[2].
5. Ł u k a s i e w i c z J., O sylogistyce Arystotelesa, (Sprawozdania Polskiej Akademii Umiejętności, 44, nr 6), Kraków 1939, s. 220-227.
6. P e a r c e D., W a g n e r G., Reasoning with Negative Information, I: Strong Negation in Logic Programs, [w:] Language, Knowledge and Intentionality, (Acta Philosophica Fennica, 49), ed. L. Haaparanta, M. Kusch, I. Niiniluoto, Helsinki 1990, s. 430-453.
7. S a d r i F., K o w a l s k i R., A Theorem-Proving Approach to Database Integrity, [w:] Foundations of Deductive Databases and Logic Programming, ed. J. Minker, Los Altos, Cal. 1988.
8. S z a j n a J., A d a m s k i M., K o z ł o w s k i T., Turbo Prolog. Programowanie w języku logiki, Warszawa 1991.

REPRESENTATION OF NEGATIVE KNOWLEDGE IN COMPUTER STUDY OF THE BIBLE

S u m m a r y

The paper concerns the problem of representation of negative knowledge (i.e. knowledge that some facts do not hold) obtained from a text in a natural language. The text from the Bible was chosen, because it is rich and diverse in its contents and formally disordered. Thus, methods developed for that text can be broadly applied to other domains. More specifically we consider the genealogy of people occurring in the chosen fragment in the text. We notice that the family relations between them hold in some situations, in other situations do not hold and in still others the text contains no information about them. The distinction between the last two cases is particularly interesting.

The presented method of knowledge representation, which is the main result of the paper, is based on the technique of axiomatic rejection created by Aristotle. A program is extended by a set of rejected formulae. Then, during a query answering procedure, the query is added to the database and we try to derive a rejected formula in the database modified in such a way. If we succeed, the negative answer for the original query is given. The details of the implementation of this procedure in Prolog are presented. Furthermore some related works, other solutions to the problem and open questions are mentioned.

Summarized by Piotr Kulicki