Zeszyty Naukowe Warszawskiej Wyższej Szkoły Informatyki Nr 9, Rok 7, 2013, s. 7-14

Negatywna informacja w języku regułowym 4QL

Jan Małuszyński[*], Andrzej Szałas[**]

Streszczenie

Problematyka negatywnej informacji w językach regułowych jest zasadnicza z punktu widzenia dużej liczby aplikacji. Była ona rozważana w rozszerzeniach języków zapytań w dedukcyjnych bazach danych, opartych na wnioskowaniach niemonotonicznych, początkowo wynikających z założenia o zamkniętym świecie CWA (Closed World Assumption). W wielu zastosowaniach, w tym związanych z Semantycznym Internetem i robotyką, założenie CWA nie jest właściwe i zwykle przyjmuje się w nich założenie o świecie otwartym OWA (Open World Assumption).

W niniejszym artykule omawiamy nowe podejście do tego problemu, przedstawione w [2] [3] [4], gdzie zaproponowaliśmy język regułowy 4QL w stylu Datalogu, jednak bez ograniczeń na wystąpienie negacji. Język ma architekturę warstwową. Najniższe warstwy 4QL, oparte na OWA, są w pełni monotoniczne. W celu zmniejszenia stref niewiedzy/sprzeczności w [2] [3] wprowadzono proste konstrukcje pozwalające na wyrażanie mechanizmów wnioskowań niemonotonicznych, w tym umożliwiających rozwiązywanie sprzeczności, użycie lokalnych domknięć świata (a więc także CWA) oraz różnych form wnioskowań przez domniemania.

Obliczanie zapytań w 4QL ma złożoność wielomianową ze względu na rozmiar bazy danych.

Słowa kluczowe: *regułowe języki zapytań, wnioskowanie niemonotoniczne, informacja negatywna, informacja sprzeczna, informacja niepełna*

[*] Uniwersytet w Linkopingu.

[**] Uniwersytet w Linkopingu; Uniwersytet Warszawski.

1 Wprowadzenie

Języki zapytań w dedukcyjnych bazach danych mają ponad trzydziestoletnią historię [1]. Tradycyjnie przyjmuje się w nich, że baza danych dzieli się na fakty oraz reguły pozwalające na wnioskowanie nowych faktów na podstawie faktów zapamiętanych w bazie lub wcześniej z niej wywnioskowanych. Fakty są wyrażane poprzez $R(\mathbf{a})$, gdzie R jest pewnym symbolem relacyjnym, zaś **a** – wektorem stałych. Reguły mają zwykle postać:

$$\pm R(\mathbf{x}) \text{ :- } \pm R_1(\mathbf{x}_1),...,\pm R_n(\mathbf{x}_n). \quad (1)$$

W powyższej regule $R, R_1,...,R_n$ są symbolami relacyjnymi, $\mathbf{x}, \mathbf{x}_1,..., \mathbf{x}_n$ – wektorami zmiennych lub stałych, zaś $\pm$ jest znakiem negacji ($\neg$) lub znakiem pustym. Wyrażenie $\pm R(\mathbf{x})$ nazywamy *konkluzją* reguły, zaś $\pm R_1(\mathbf{x}_1),...,\pm R_n(\mathbf{x}_n)$ – jej *przesłankami*.

Intuicyjnym znaczeniem reguły (1) jest:

„jeśli wszystkie przesłanki $\pm R_1(\mathbf{x}_1),...,\pm R_n(\mathbf{x}_n)$ są spełnione, to wyprowadź również $\pm R(\mathbf{x})$".

Negacja, jako operator niemonotoniczny, zwykle powoduje poważne problemy złożonościowe i/lub semantyczne. O ile negacja w przesłankach reguł była intensywnie badana, o tyle dopuszczenie negacji w konkluzjach reguł przynosi dodatkowe problemy, w szczególności może okazać się, że różne reguły prowadzą do sprzecznego wniosku. W takim przypadku użycie logiki klasycznej skutkuje trywializacją wnioskowania (można udowodnić każdą tezę). Tymczasem w warunkach sprzecznej informacji systemy inteligentne powinny nadal funkcjonować bez posługiwania się absurdalnymi konkluzjami. Ponadto sprzeczność często daje się rozwiązać. Na przykład sprzeczna informacja dotycząca temperatury może wymagać dokonania ponownego jej pomiaru lub sprawdzenia poprawności działania czujnika temperatury.

Innym problemem jest niepełna informacja. W bazach danych przyjmuje się zwykle założenie o zamkniętym świecie (CWA), w którym zakłada się, że jeśli nie da się wykazać faktu na podstawie posiadanej wiedzy, to przyjmujemy, że jest on fałszywy. Tego typu podejście jest nie do przyjęcia w systemach działających w rzeczywistym świecie. Na przykład z faktu, że nie mamy wiedzy o zbyt wysokiej temperaturze nie powinniśmy automatycznie wnioskować, że temperatura nie jest zbyt wysoka (jak miałoby to miejsce przy założeniu CWA).

W niniejszym artykule podejmujemy problematykę wiedzy negatywnej, a w konsekwencji również potencjalnie niepełnej i sprzecznej, jako efekt rozważań semantycznych dotyczących reguł z negacją w konkluzjach i przesłankach. Zaproponowane w [2] [3] [4] rozwiązanie przyjmuje cztery wartości logiczne dla reprezentacji powstających zjawisk:

- T – reprezentującą prawdę (ang. *True*);
- F – reprezentującą fałsz (ang. *False*);
- U – reprezentującą niewiedzę (ang. *Unknown*);
- I – reprezentującą sprzeczność (ang. *Inconsistent*).

Język 4QL został zaimplementowany w postaci eksperymentalnych interpreterów (dostępnych na stronie 4ql.org). Obecnie jest prowadzony projekt jego implementacji w Naukowej i Akademickiej Sieci Komputerowej NASK.

Artykuł ma charakter popularyzatorski. Zaczynamy go od zaprezentowania składni języka 4QL. Następnie omawiamy jego semantykę i przedstawiamy przykład ilustrujący podstawowe idee. Dokładna analiza dyskutowanych rozwiązań została przeprowadzona w artykułach [2] [3] [4] [6], gdzie czytelnik znajdzie także omówienie literatury przedmiotu.

2 Składnia języka 4QL

Literałem nazywamy wyrażenie postaci $\pm R(\mathbf{x})$,
gdzie:

- $\pm$ jest znakiem negacji ($\neg$) lub znakiem pustym;
- R jest symbolem relacyjnym;
- $\mathbf{x}$ jest wektorem argumentów będących zmiennymi lub stałymi.

Literał postaci $\neg R(\mathbf{x})$ nazywamy *negatywnym*, a postaci $R(\mathbf{x})$ − *pozytywnym*. Literałem *ugruntowanym* nazywamy literał bez zmiennych.

Dalej $\neg\neg R(\mathbf{x})$ utożsamiamy z $R(\mathbf{x})$.

Przez *regułę 4QL* rozumieć będziemy wyrażenie postaci:

$$\pm R(\mathbf{x}) :- \pm R_{11}(\mathbf{x}_{11}),...,\pm R_{1n}(\mathbf{x}_{1n}) \text{ V } ... \text{ V } \pm R_{m1}(\mathbf{x}_{m1}),...,\pm R_{ms}(\mathbf{x}_{ms}). \quad (2)$$

W pracach [2], [3] zakładaliśmy, że dla dany literał ugruntowany może być konkluzją co najwyżej jednej reguły. W pracach [4, 6] to ograniczenie zostało usunięte.

W tradycyjnych językach regułowych dysjunkcja nie musi występować w przesłankach reguł. Można ją uzyskać za pomocą większej liczby reguł. Mianowicie:

$$(A \text{ V } B) \rightarrow C \text{ jest równoważne } (A \rightarrow C) \Lambda (B \rightarrow C).$$

W przypadku omawianego przez nas języka analogiczna równoważność nie jest prawdziwa.

Reguły z pustą przesłanką nazywamy *faktami*. Pusta przesłanka jest interpretowana jako prawda (wartość logiczna T).

W dalszej części artykułu dla prostoty zakładamy, że mamy do czynienia z regułami bez zmiennych. Sprowadzanie reguł do tej postaci może być przeprowadzone w sposób standardowo przyjęty w regułowych językach bazodanowych.

3 Semantyka języka 4QL

Dla zdefiniowania semantyki użyjemy czterowartościowej logiki zaproponowanej w pracy [5]. Matryce logiczne dla spójników koniunkcji, dysjunkcji, implikacji i negacji są przedstawione na następnej stronie.

Koniunkcja i dysjunkcja odzwierciedla przyjęty porządek na wartościach logicznych, w którym:

$$F < U < I < T. \tag{3}$$

Definicja implikacji wynika z przyjętych założeń:

- nie wnioskujemy z fałszywych przesłanek i niewiedzy; dlatego przyjmujemy, że implikacja z przesłankami T, U ma wartość T (jest więc spełniona niezależnie od wartości konkluzji);
- ze sprzeczności może wynikać jedynie sprzeczność;
- prawdziwe przesłanki mogą prowadzić do wniosków prawdziwych lub sprzecznych (bowiem inna reguła może nadać wartość T konkluzji przeczącej wywnioskowanemu wcześniej literałowi).

$\wedge$	F	U	I	T
F	F	F	F	F
U	F	U	U	U
I	F	U	I	I
T	F	U	I	T

$\vee$	F	U	I	T
F	F	U	I	T
U	U	U	I	T
I	I	I	I	T
T	T	T	T	T

$\rightarrow$	F	U	I	T
F	T	T	T	T
U	T	T	T	T
I	F	F	T	F
T	F	F	T	T

	$\neg$
F	T
U	U
I	I
T	F

Rysunek 1. Matryce logiczne dla spójników koniunkcji, dysjunkcji, implikacji i negacji

Przez *interpretację* rozumieć będziemy skończony zbiór literałów ugruntowanych. *Wartość literału* l w interpretacji M, oznaczaną przez $M(l)$, definiujemy jako:

- $M(l) = T$ gdy $l \in M$ oraz $\neg l \notin M$;
- $M(l) = I$ gdy $l \in M$ oraz $\neg l \in M$;
- $M(l) = U$ gdy $l \notin M$ oraz $\neg l \notin M$;
- $M(l) = F$ gdy $l \notin M$ oraz $\neg l \in M$.

Modelem zbioru reguł nazywamy interpretację, w której prawdziwa jest każda reguła, rozumiana jako implikacja:

$$[(\pm R_{11}(\mathbf{x}_{11}) \wedge \ldots \wedge \pm R_{1n}(\mathbf{x}_{1n})) \vee \ldots \vee (\pm R_{m1}(\mathbf{x}_{m1}) \wedge \ldots \wedge \pm R_{mr}(\mathbf{x}_{mr}))] \rightarrow \pm R(\mathbf{x})$$

Warto zauważyć, że tradycyjna semantyka, oparta na modelach minimalnych nie jest adekwatna dla języka 4QL. Rozważmy bowiem zbiór reguł S [2]:

wait :– overloaded V rest time.
rest time :– wait.
¬overloaded :– rest time.
overloaded.

Minimalnym modelem dla S jest:

{overloaded, ¬overloaded, wait, rest_time}.

Jednak jedynym faktem jest tu overloaded. Prawdziwość wait oraz rest_time nie jest więc uzasadniona żadnym faktem. Intuicyjnie poprawnym modelem jest:

{overloaded, ¬overloaded, wait, ¬wait, rest_time, ¬rest_time},

w którym wszystkie literały są sprzeczne (mają wartość I).

Nie wchodząc w formalne definicje (podane w pracach [2] [3] [4]), przyjmiemy, że właściwym modelem jest taki, w którym wartości logiczne literałów są uzasadnione wnioskowaniem ugruntowanym w faktach. Takie modele będziemy nazywali *dobrze wspieranymi* (ang. *well-supported*).

Dokładniej literał *l* przyjmuje w takim modelu wartość:

- T, gdy istnieje wnioskowanie wychodzące z faktów, prowadzące do prawdziwości *l* i nie istnieje wnioskowanie temu przeczące (tj. prowadzące w ten czy inny sposób do sprzeczności);
- I, gdy istnieje wnioskowanie wychodzące z faktów, prowadzące do sprzeczności dotyczącej *l*;
- U, gdy nie istnieje wnioskowanie wychodzące z faktów, prowadzące do *l* lub ¬*l*;
- F, gdy istnieje wnioskowanie wychodzące z faktów, prowadzące do fałszywości *l* i nie istnieje wnioskowanie temu przeczące.

Okazuje się (por. [2] [3] [4]), że dla każdego zbioru reguł istnieje dokładnie jeden dobrze wspierany model. Można go obliczyć w czasie wielomianowym ze względu na liczbę reguł [2] [3] [4] (zakładamy tu, że fakty tworzą bazę danych). Co więcej, każde wielomianowo obliczalne zapytanie da się wyrazić w (pełnym) języku 4QL [3].

4 Moduły i literały zewnętrzne

Dla obsługi sprzeczności i braków wiedzy w 4QL wprowadzone zostały moduły i literały zewnętrzne, pozwalające na zadawanie zapytań modułom przez inne moduły. Pojęcie modułu jest dobrze znane z wielu języków programowania. *Literały zewnętrzne* mają postać:

m.literał(parametry)

lub

m.literał(parametry) IN S,

gdzie m jest nazwą modułu, a S jest podzbiorem zbioru wartości logicznych {F, U, I, T}. Intuicyjnie:

- m.literał(parametry) oznacza zapytanie modułu m o wartości literału literał(parametry) rozumianego jako zapytanie;
- m.literał(parametry) IN S oznacza zapytanie, czy m.literał(parametry) jest w zbiorze wartości logicznych S.

Przyjmuje się założenie o niecykliczności zapytań. Mianowicie – jeśli moduł m zadaje pytanie modułowi n, to moduł n nie może zadać pytania modułowi m bezpośrednio ani pośrednio. Gdybyśmy dopuścili zapytania cykliczne – mogłoby to prowadzić do problemów semantycznych. Dla ilustracji problemu rozważmy dwa moduły z zapytaniami tworzącymi cykl:

- moduł m, który zawiera jedynie regułę:

p:- n.q IN {U}. (4)

- moduł n, który zawiera regułę:

q:- m.p IN {U}. (5)

W takim przypadku wyniki zależą od kolejności wykonywania obliczeń. Jeśli najpierw wyliczymy konkluzję reguły (4), p przyjmie wartość T (gdyż wówczas q w module n ma wartość U, a więc literał n.q IN {U} ma wartość logiczną T), zaś q będzie miało wartość U (skoro p w module m ma teraz wartość T – literał m.p IN {U} ma wartość logiczną F). Podobnie – jeśli najpierw wyliczymy konkluzję reguły (5), q przyjmie wartość T, zaś p będzie miało wartość U.

Inny problem pojawia się, gdy w module n zmienimy regułę na:

q:- m.p IN {T}. (6)

Wówczas p oraz q przyjmą wartość T (na początku q ma wartość U, więc reguła (4) powoduje, że p staje się T). Teraz m.p IN {T} ma wartość T, więc na podstawie reguły (6) również q przyjmuje wartość T. Skoro tak, to wniosek, że p ma wartość T, uzyskany na podstawie reguły (4), przestaje mieć uzasadnienie. Gdyby go aktualizować, p przyjęłoby wartość U, w konsekwencji q przyjmuje tę samą wartość, bowiem p IN {T} staje się fałszywe. Na mocy reguły (4), p znów przyjęłoby wartość T i proces obliczeniowy zapętla się.

Przykład

Rozważmy wariant przykładu z pracy [4], ilustrujący omówione wcześniej zagadnienia oraz pokazujący sposób podejścia do wnioskowań niemonotonicznych (rozstrzygających sprzeczności i uzupełniających braki wiedzy). W tym celu rozważmy scenariusz:

- człowiek staje się podejrzany, jeśli są świadkowie zeznający, że popełnił dane przestępstwo;
- człowiek nie jest podejrzany, jeśli są świadkowie dający mu alibi;
- w czasie śledztwa zbierane są zeznania świadków;
- w czasie procesu za winnych są uznawani ci podejrzani, na których winę wskazują zeznania, a którzy zarazem nie mają alibi.

W powyższym scenariuszu mamy do czynienia z negatywną konkluzją (człowiek **nie** jest podejrzany jeśli ma alibi), negatywną przesłanką (**nie** ma alibi), domniemanie niewinności prowadzi do wnioskowania z niewiedzy (przy braku jakichkolwiek zeznań przyjmiemy, że człowiek jest niewinny), zaś rozstrzyganie wątpliwości na korzyść podejrzanego prowadzi do odpowiedniego rozstrzygania sprzecznych zeznań.

Możemy zdefiniować dwa moduły: 'śledztwo' oraz 'rozstrzyganie':

- moduł 'śledztwo' może zawierać fakty dotyczące zeznań o popełnieniu przestępstwa i alibi, a także reguły:

podejrzany(X):- zeznanie_przeciw(X).

¬ podejrzany(X):- alibi(X).

- moduł 'rozstrzyganie' może zawierać reguły:

  ¬winny(X):- śledztwo.podejrzany(X) IN {F, U, I}.

  winny(X):- śledztwo.podejrzany(X) IN {T}, śledztwo.alibi(X) IN {F}.

Zauważmy, że przesłanki mogą mieć również wartości U oraz I, reprezentujące brak odpowiednich zeznań lub ich sprzeczność. Dlatego reguła z konkluzją ¬winny(X) bierze w przesłankach pod uwagę również sytuacje, w których moduł 'śledztwo' stwierdza, że wartością podejrzany(X) jest U lub I.

5 Podsumowanie

W artykule przedstawiliśmy zasadnicze koncepcje leżące u podstaw monotonicznych warstw regułowego języka zapytań 4QL [2] [3] [4], pozwalającego na użycie negacji w konkluzjach i przesłankach reguł. Semantyka tego języka jest czterowartościowa, uwzględniająca prawdziwość, fałszywość, sprzeczność oraz brak wiedzy. Podstawowy mechanizm obliczeniowy został oparty na pojęciu dobrze wspieranego modelu. Zaproponowany algorytm obliczania takich modeli ma złożoność wielomianową, może być więc praktycznie stosowany.

Wyższe warstwy 4QL pozwalają na wyrażanie mechanizmów wnioskowania niemonotonicznego.

Do interesujących, otwartych jeszcze problemów zaliczamy m.in.:

- dla uzyskania odpowiedzi na zapytania, w pracach [2] [3] [4] konstruuje się cały dobrze wspierany model; interesującym i ważnym zagadnieniem jest podanie technik obliczania zapytań niewymagających skonstruowania całego dobrze wspieranego modelu;
- optymalizację algorytmu wyliczania dobrze wspieranego modelu z pracy [4];
- podanie technik pozwalających na możliwie efektywne obliczanie modelu dobrze wspieranego po zmianie części faktów, na podstawie wcześniej obliczonego modelu.

Interesujące są także praktyczne zastosowania języka 4QL.

Bibliografia

[1] Abiteboul S., Hull R., Vianu V., *Foundations of Databases*, Addison-Wesley Pub. Co., Reading 1996

[2] Małuszyński J., Szałas A., *Living with inconsistency and taming nonmonotonicity*, Datalog Reloaded, LNCS 6702, Springer-Verlag, 2011, s. 384-398, http://dx.doi.org/10.1007/978-3-642-24206-9_22

[3] Małuszyński J., Szałas A., *Logical foundations and complexity of 4QL, a query language with unrestricted negation*, "Journal of Applied Non-Classical Logics" 2011, Vol. 21 (2), s. 211-232, http://dx.doi.org/10.3166/jancl.21.211-232

[4] Małuszyński J., Szałas A., *Partiality and Inconsistency in Agents' Belief Bases,* Proceedings of KES-AMSTA, Frontiers of Artificial Intelligence and Applications, IOS Press, 2013
[5] Małuszyński J., Szałas A., Vitória A., *Paraconsistent logic programs with four-valued rough sets*, w: *Rough Sets and Current Trends in Computing*, (red.) Chan C.-C., Grzymala-Busse J., Ziarko, Proceedings of 6th International Conference, RSCTC 2008, LNAI 5306, Springer-Verlag, Berlin-Heidelberg 2008, s. 41-51, http://dx.doi.org/10.1007/978-3-540-88425-5_5
[6] Szałas A., *How an Agent Might Think*, "Logic Journal of IGPL" 2013, http://dx.doi.org/10.1093/jigpal/jzs05

Negative information in a rule-based language 4QL

Abstract

The problem of negative information in rule languages is crucial in many applications. It has been addressed in extensions of query languages in deductive databases, based on nonmonotonic logics initially derived from the Closed World Assumption (CWA). In many applications, including Semantic Web technologies and robotics systems, CWA is not necessarily applicable and developments in these fields usually follow the Open World Assumption (OWA).

In this paper we summarize a novel approach to the problem reported in [2] [3] [4], where we proposed a DATALOG-like language 4QL with unrestricted negation. The language supports a layered architecture. The monotonic layer of 4QL, based on OWA, is fully monotonic. To reduce the unknown/inconsistent zones, in [2] [3] we have introduced simple constructs which allow one to express various mechanisms of nonmonotonic reasoning. In particular, this provides means for application-specific disambiguation of inconsistent information, the use of Local CWA (thus also CWA, if needed), and various forms of default reasoning.

Query evaluation in 4QL is still tractable as regards its data complexity.

Keywords: *rule-based query languages, nonmonotonic reasoning, negative information, inconsistent information, incomplete information*