STUDIA INFORMATICA 2015
Volume 36 Number 1 (119)

Agnieszka NOWAK-BRZEZIŃSKA
Uniwersytet Śląski, Instytut Informatyki

KBEXPLORATOR A INNE NARZĘDZIA EKSPLORACJI REGUŁOWYCH BAZ WIEDZY

Streszczenie. Celem artykułu jest przedstawienie funkcjonalności systemu kbExplorator na tle dostępnych narzędzi, pozwalających na tworzenie i edycję regułowych baz wiedzy wraz z algorytmami wnioskowania. Scharakteryzowano, a następnie porównano trzy z dostępnych narzędzi: Jess, Drools oraz PC-Shell, a także opisano podstawowe cechy współtworzonego przez autorkę narzędzia kbExplorator. Prócz formalnych definicji przedstawiono także przykład regułowej bazy wiedzy z podziałami reguł i omówiono proces wnioskowania realizowany dla takiej bazy wiedzy.

Słowa kluczowe: regułowe bazy wiedzy, procesy wnioskowania, podziały reguł, podobieństwo reguł, systemy ekspertowe

KBEXPLORATOR VS. RULE-BASED KNOWLEDGE BASES

Summary. The aim of the study was to describe the functionality of the kbExplorator system which allows to create and use of rule-based knowledge bases with an inference algorithms. Three available tools: Jess, Drools and PC-Shell were characterized and compared to the kbExplorator co-created by the author. Apart from formal definition, an example of the knowledge base with rules partition was shown, and application process implemented for such a knowledge base was discussed.

Keywords: rule-based knowledge base, inference process, rules partition, similarity of rules, expert systems

1. Wprowadzenie

Przez eksplorację regułowej bazy wiedzy (RBW) należy rozumieć zarówno proces jej tworzenia, jak i – w późniejszym etapie – eksploracji. Popularność systemów wspomagania decyzji (SWD), które najczęściej przechowują wiedzę dziedzinową w formie RBW, sprawia, że warto poddać analizie użyteczność dostępnych narzędzi do tworzenia tego typu systemów.

Warte porównania są czynniki, takie jak przyjazny interfejs użytkownika, format bazy wiedzy (BW), realizowane algorytmy wnioskowania, a także dostępność modułu objaśniającego czy modułu akwizycji wiedzy. Z tego względu rozdział drugi niniejszego artykułu przedstawia charakterystykę SWD, wyróżniając niezbędne elementy składowe tego typu systemów oraz ich znaczenie. Dostępne na rynku narzędzia, tj. JESS [1], Drools [2] oraz PC-Shell [3], pełnią podobną rolę: dla zapisanej w pliku (o ustalonym formacie) BW z regułami prowadzą wnioskowanie (jedną bądź obiema metodami[1]) i przedstawiają użytkownikowi proponowaną decyzję. Nie są one jednak w pełni wystarczające dla każdego typu użytkownika, bez względu na jego umiejętności i wiedzę z zakresu tworzenia i/lub użytkowania SWD. Dogłębna analiza działania tych narzędzi (przedstawiona w skrócie w rozdziale 3) pozwoliła dostrzec sporo ograniczeń, i tym samym stała się argumentem do zaprojektowania oraz stworzenia nowego (alternatywnego) narzędzia, które będzie miało na celu tworzenie, edycję, a następnie wnioskowanie dla RBW. Dla tej samej BW działanie systemów może być różne, gdyż kwestią kluczową pozostaje zawsze algorytm wnioskowania, na podstawie którego dany system realizuje proces wspomagania decyzji. Autorka w pracy [5] wiele uwagi poświęciła efektywności SWD. Wnioski z tamtych badań stały się argumentem za opracowaniem nowego podejścia do optymalizacji RBW, poprzez zmianę struktury BW oraz optymalizację procesów wnioskowania [6, 7]. Zmiana struktury BW opierać się ma na tworzeniu *podziałów reguł*[2], zaś zwiększenie efektywności procesów wnioskowania ma być realizowane w odniesieniu do, określanego w literaturze jako najszybszy, algorytmu wnioskowania – RETE [4]. Algorytm okazał się bardzo efektywny, gdyż pozwalał na realizację procesów wnioskowania (nawet dla bardzo licznych baz wiedzy) w bardzo krótkim czasie. Rozdział czwarty artykułu zawiera opis systemu kbExplorator, przedstawiając architekturę systemu, analizę jego wymagań, jak i podstawowe funkcjonalności. W rozdziale ujęto zarówno opis proponowanego podejścia do modularyzacji dużych regułowych baz wiedzy, algorytmów tworzenia podziałów reguł oraz algorytmów wnioskowania, a także porównano tworzony system z narzędziami opisanymi wcześniej w rozdziale trzecim.

2. Charakterystyka systemów z regułową bazą wiedzy

Definicja systemu ekspertowego określa, że jest to program komputerowy wykorzystujący wcześniej zgromadzoną wiedzę oraz określone procedury rozumowania do wspomagania podejmowania decyzji i rozstrzygania problemów o wysokim stopniu złożoności, których rozwiązanie wymaga specjalistycznej wiedzy eksperta. Jako taki, system ten musi się zatem

[1] Wyróżniamy wnioskowanie w przód (sterowane danymi) i wstecz (sterowane celem).
[2] Podział reguł (ang. *rules partition*) ma charakter grup reguł o podobnych (bądź tych samych) własnościach.

składać ze zgromadzonej dziedzinowej wiedzy oraz procedur wnioskowania. Nie są to jednak elementy wystarczające, by sprawnie obsługiwać taki system. Typowa budowa systemu ekspertowego obejmuje cztery podstawowe elementy: bazę wiedzy[3], mechanizm wnioskujący[4], interfejs użytkownika[5] oraz moduł objaśniający[6]. Dla podkreślenia roli dwóch modułów: bazy wiedzy oraz wnioskowania, można posłużyć się symbolicznym zapisem, zgodnie z którym *system ekspertowy = baza wiedzy + moduł wnioskowania*. Ze względu na wygodę użytkowania danego systemu idealnym rozwiązaniem będzie narzędzie łączące oba elementy w jednym miejscu. Istotne znaczenie ma zarówno przyjęta reprezentacja wiedzy, jak i realizowane metody wnioskowania. Najbardziej intuicyjną formą reprezentacji wiedzy z dziedziny jest reprezentacja regułowa w formie odzwierciedlającej łańcuchy przyczynowo-skutkowe typu: *Jeżeli warunek_1 & warunek_2 & …to konkluzja* np. w plikach tekstowych i/lub XML. Niestety, tylko niektóre z najbardziej popularnych narzędzi na to pozwalają. Równie istotna jest realizowana metoda wnioskowania. Nie wszystkie narzędzia pozwalają na użycie obu możliwych metod wnioskowania, tj. w przód i wstecz. Algorytmy te różnią się znacząco pod kątem wiedzy, jakiej wymagają na wejściu, jak i wiedzy dostarczanej na wyjście. Stąd realizacja tylko jednej metody jest sporym ograniczeniem dla użytkownika. Istnieje wiele wersji algorytmów wnioskowania, które prześcigają się m.in. pod kątem niższej złożoności obliczeniowej. Najbardziej popularnym algorytmem wnioskowania jest algorytm RETE, nazywany często algorytmem wyznaczania tzw. zbioru konfliktowego[7]. Reguły, które mogą być uaktywnione, dopisujemy do zbioru konfliktowego i ostatecznie to wybrana strategia (LIFO[8] oraz FIFO[9]) decyduje o tym, w jakiej kolejności reguły z tego zbioru będą uaktywnione, ge-

[3] Baza wiedzy zawiera wiedzę ekspertów z danej dziedziny zapisaną np. w postaci reguł, ram, sieci semantycznych i zorganizowaną np. hierarchicznie, tj. podzieloną na kilka poziomów (dotyczy to szczególnie rozległych baz wiedzy potrzebnych dla dużych i skomplikowanych systemów ekspertowych), gdzie zawartość wyższych poziomów określa się jako metawiedzę, czyli „wiedzę o wiedzy".

[4] Mechanizm wnioskujący to serce systemu ekspertowego odpowiedzialne za poprawne zastosowanie wiedzy dziedzinowej.

[5] Interfejs użytkownika umożliwia komunikację użytkownika z systemem i pozwala m.in. na wprowadzanie danych do systemu, prezentowanie objaśnień w trakcie realizowanego procesu wnioskowania.

[6] Moduł objaśnień, który najczęściej jest elementem interfejsu użytkownika, dostarcza dwojakiego rodzaju wyjaśnień. Po pierwsze, na każdym etapie procesu wnioskowania powinien wyjaśniać użytkownikowi celowość zadawania konkretnych pytań, zaś na końcu, podczas prezentacji decyzji systemu, drogę prowadzonego rozumowania. Bez modułu objaśnień system ekspertowy może poprawnie działać, jednak dla celów kontroli i weryfikacji poprawności funkcjonowania umieszczenie tego modułu jest celowe.

[7] Zbiór konfliktowy (ang. conflict set) to zbiór reguł kandydujących do uaktywnienia w procesie wnioskowania. Gdy w danym momencie więcej niż jedną regułę można uaktywnić, wybrana strategia wyboru reguł decyduje ostatecznie o tym, w jakiej kolejności będą one analizowane.

[8] LIFO: w procesie wnioskowania kolejność uaktywniania reguł jest przeciwna w stosunku do kolejności ich zapisu (reguły najwcześniej dodane będą uaktywnione najpóźniej). Według tej strategii kolejność uaktywnienia reguł będzie następująca:r9, r1, r7, czyli nowe fakty zostaną wygenerowane w następującej kolejności: d=4, c=1, f=1.

[9] FIFO jako przeciwieństwo strategii LIFO uaktywni reguły w kolejności zgodniej z czasem ich dodania do zbioru konfliktowego, a więc kolejność uaktywnionych reguł będzie następująca: r7, r1, r9, zaś zbiór faktów zostanie zasilany o nowe fakty w kolejności: f=1, c=1, d=4.

nerując nowe fakty. Dokładnie algorytm opisano już w wielu pracach, m.in. [4]. Dla przykładowej bazy wiedzy zawierającej 9 reguł:

```
r1: a=1 & b=1 => c=1; r2: a=1 & b=2 => c=2 ; r3: a=1 & b=3 => c=1
r4: b=3 & d=3 => e=1; r5: b=3 & d=2 => e=1; r6: b=3 => e=2
r7: d=4 => f=1 ; r8: d=4 & g=1 => f=1 ; r9: a=1 => d=4
```

wnioskowanie dla trzech faktów: *a=1, b=1 oraz d=4* będzie przebiegało następująco. Po pojawieniu się faktu *a=1* w pamięci podręcznej jedynie reguła r9 może być uaktywniona, więc dodajemy ją do zbioru konfliktowego. Pojawienie się faktu *b=1* spowoduje dopisanie reguły r1 do zbioru konfliktowego. Ostatecznie, po załadowaniu do pamięci faktu *d=4*, zbiór ten zostanie zasilony o regułę r7. Zatem, wnioskowanie zostanie przeprowadzone dla 3 reguł: r9, r1 oraz r7. W zależności od wybranej strategii sterowania wnioskowaniem (LIFO/FIFO) w różnej kolejności będą uaktywnione te 3 reguły kandydujące. Nie wszystkie analizowane systemy bazują na tym algorytmie wnioskowania. Elementem ważnym przy użytkowaniu SWD, zwłaszcza w dzisiejszych czasach, jest także wieloplatformowość i dostępność narzędzi na różnego typu urządzeniach. Niewątpliwie dużo większa jest użyteczność systemu, który pozwala na wykorzystanie jego możliwości, bez ograniczeń dotyczących systemu operacyjnego czy sprzętu, na którym są użytkowane. Pod tym względem analizowane narzędzia są także ograniczone.

Dobrym rozwiązaniem dla dużych zbiorów reguł jest modularyzacja bazy wiedzy, tj. podział reguł na mniejsze podzbiory. Proces ten jest realizowany zarówno w celu lepszej organizacji reguł, jak i skrócenia czasu przeglądu bazy wiedzy. Jest to odrębne zagadnienie i nie będzie szerzej omawiane w niniejszym artykule. Warto jedynie nadmienić, że we współtworzonym (przez autorkę) systemie kbExplorator modularyzacja realizowana jest na podstawie koncepcji *podziałów reguł* (opisanej w rozdziale 4.2.).

3. Przegląd istniejących narzędzi

Prezentowane w niniejszym rozdziale narzędzia należą do najczęściej opisywanych w literaturze, jak i wykorzystywanych w praktyce systemów do tworzenia RBW oraz realizowania procesów wnioskowania dla takich BW. Jess oraz Drools są przykładami współczesnych rozwiązań, czyli tzw. silników regułowych[10], zaimplementowanych w językach typu Java, C++ itp. Należy wspomnieć, że Jess miał być alternatywą dla CLIPSa[11] i jego klonem, jednak w obecnej wersji istnieje wiele różnic między tymi narzędziami[12]. Spośród wielu czynni-

[10] systemów zarządzania regułami biznesowymi (ang. business rule management system – BRMS).
[11] *C Language Integrated Production System.*
[12] Narzędzie znalazło wiele zastosowań w innych dziedzinach: w rozpoznawaniu obrazów, rozumieniu scen itp. Prócz regułowej reprezentacji wiedzy pozwala na reprezentację zorientowaną obiektowo oraz proceduralnie.

ków, które warto było porównać, wybrano m.in. to, czy narzędzie jest darmowe, czy dostępny jest edytor tworzenia BW oraz jakie metody wnioskowania są realizowane. Innym czynnikiem, o którym należy wspomnieć porównując oba narzędzia, jest format BW. Środowisko Jess bazę wiedzy przechowuje w plikach z rozszerzeniem *.cls*. Format zapisu reguły w Jess wygląda następująco: `(defrule RuleName "comment" (fact_1) ... (fact_N) => (action_1) ... (action_M))`[13]

W zależności od tego, jaka strategia wyboru reguł była użyta, reguły te będą uaktywnione w różnej kolejności[14]. Przykładowa BW z 9 regułami (z rozdziału 2) zapisana w Jess będzie miała następującą składnię:

```
(reset)
(assert (a=1))
…
(defrule r1 (and (a=1) (b=1)) => (assert (c=1))(printout t "c=1" crlf))
(defrule r2 (and (a=1) (b=2)) => (assert (c=2))(printout t "c=2" crlf))
….
 (defrule r9 (a=1) => (assert (d=4))(printout t "d=4" crlf))
(run)
(facts)
```

Z kolei Drools dostarcza reguły w plikach z rozszerzeniem *.drl.* Ogólnie, format BW wyglądać może następująco:

```
package package-name
imports
globals
functions
queries
rules
zaś przykładowa reguła może mieć postać:
rule "GoodBye"
    when Message( status == Message.GOODBYE, myMessage : message )
    then System.out.println( myMessage );
end
```

PC-Shell jest pierwszym polskim (dostępnym komercyjnie) hybrydowym systemem ekspertowym. Sama aplikacja jest dziedzinowo niezależnym narzędziem do tworzenia systemów ekspertowych. Hybrydowość systemu polega na tym, że łączy on w sobie wiele różnych metod rozwiązywania problemów. Architektura pozwala na podział dużej bazy wiedzy na wiele mniejszych części. Należy podkreślić, że maszyna wnioskująca używa mechanizmu *realiza-*

Napisany w języku C posiada implementacje dla systemów Windows, Mac OS oraz Unix. Wyższość Jess nad jego odpowiednikiem CLIPSem polega na wieloplatformowości realizowanej dzięki implementacji w Javie.

[13] Lewa strona odpowiada definiowaniu faktów, zaś prawa strona definiowaniu decyzji (akcji). Prócz możliwości definiowania faktów/przesłanek/konkluzji reguł o różnych typach danych można także definiować funkcje dzięki konstrukcji `(deffunction <name> (<parameter>*) [<comment>] <expression>*.` Po otwarciu pliku z regułami należy je wczytać do pamięci podręcznej przez użycie np. funkcji *reset*, a następnie wywołać procedurę wnioskowania przez funkcję *run*. Dostępne są specjalne funkcje do wyświetlenia faktów (*facts*), dodawania nowych faktów (*assert*), do usuwania wybranych faktów (*retract*) oraz wiele innych.

[14] Jeśli użyjemy strategii LIFO[14], reguły r1,…,r9 będą uaktywniane w kolejności: r7, r1, r9, a więc fakty będą generowane zgodnie z kolejnością: f=1, c=1, d=4, zaś gdybyśmy wybrali strategię FIFO[14], inna byłaby kolejność uaktywniania reguł oraz faktów: d=4, c=1, f=1.

cji nawrotów (podobnie jak to rozwiązano w Prologu). Rozważana powyżej przykładowa baza wiedzy w języku PC-Shell miałaby następującą postać:

```
knowledge base baza
facets
single yes;
    a:  query "Wartość atrybutu a ?"
        val oneof { "1", "2" };
….
end;
rules
c="1" if a="1",b="1";
…
d="4" if a="1";
end;
facts
a ="1";b ="1";d ="4";
end;
control
    run;
    ...
    goal("f=1");    delNewFacts;
    end;
end;
end;
```

Baza wiedzy składa się z dwóch części: bloku deklaracji *faset*, reguł oraz faktów, a następnie bloku kontrolnego. Dla tak skonstruowanych faktów wnioskowanie w przód podobnie jak w przypadku narzędzia JESS wyprowadzi na wyjście trzy nowe fakty: f=1,d=4,c=1. Wszystkie opisane narzędzia dobrze sprawdzają się, gdy są obsługiwane przez programistów. Wymagają one bowiem umiejętności konfiguracji środowisk, w których narzędzie pracuje, czy umiejętności programowania pewnych fragmentów dotyczących działania systemu. Ich użytkowanie będzie się wówczas kojarzyło ze stałymi kłopotami podczas kompilacji baz wiedzy i realizacji procesów wnioskowania. Wspomniane narzędzia cechuje wiele podobieństw: realizują podobne metody wnioskowania, a także udostępniają podobne strategie doboru reguł itp. Wśród różnic można wymienić to, że jedynie PC-Shell dostarcza tzw. *modułu objaśniającego*, pozwalającego użytkownikowi śledzić tok rozumowania prowadzonego przez system. Wynikiem takiego porównania z uwzględnieniem systemu PC-Shell jest tabela 1.

Tabela 1

Porównanie własności narzędzi tworzenia systemów ekspertowych

Własność	Jess	Drools	PC-Shell

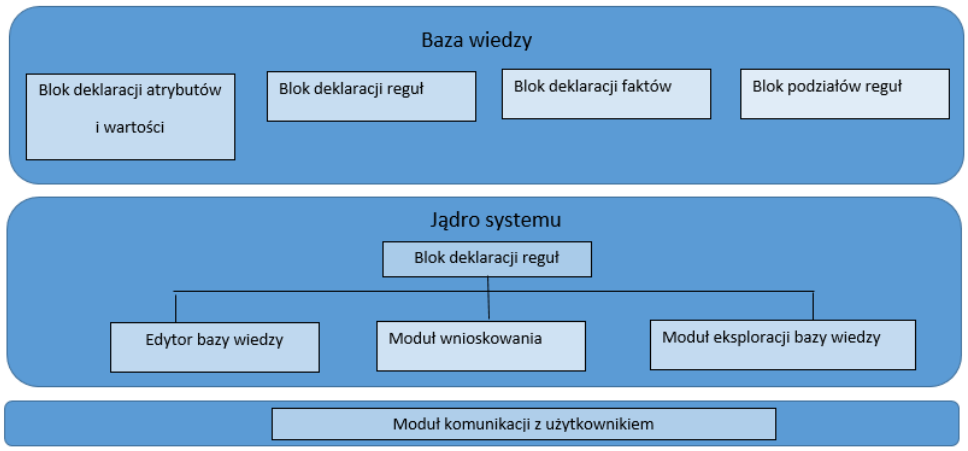
Język implementacji	Java	Java	C++
kreator BW	Nie	Nie	Tak
modularyzacja BW	Nie	Tak	Tak
Strategie wnioskowania i metody wnioskowania	Salience[15], LIFO Wnioskowanie w przód i wstecz[16] Maszyna wnioskująca: pattern matcher[17] oraz agenda[18]	w przód i wstecz	w przód i wstecz oraz mieszane
Dostępność	komercyjne/niekomercyjne	Niekomercyjne	komercyjne
wieloformatowość BW	Tak	Tak	Nie
format BW	*.cpl / *.cls	*.drl	*.bw
moduł objaśniający	Nie	Nie	Tak[19]
weryfikacja poprawności BW	Nie	Nie	Tak
Typ danych	jak integer, float oraz string	jak integer, float oraz string	jak integer, float oraz string
Definiowanie funkcji/procedur	Tak	Tak	Tak
Wady	Kłopot dla użytkowników niepotrafiących programować, możliwa redundancja faktów[20]	Konieczność znajomości jęz. programowania	Zależny platformowo

4. kbExplorator – przedstawienie idei systemu

Analiza narzędzi do tworzenia systemów ekspertowych (rozdział 3) pozwoliła określić oczekiwane funkcjonalności takich narzędzi. Nie powinno być ograniczeń typu: tworzenie BW w edytorze tekstu bądź w konsoli, składnia powinna być możliwie najprostsza, funkcje/procedury wbudowane powinny mieć nazwy bezpośrednio kojarzone z ich znaczeniem. Nie powinno być ograniczeń co do liczby reguł, ich długości oraz typów danych w regułach i faktach. System powinien pozwalać na wnioskowanie przy użyciu obu metod wnioskowania: wstecz oraz w przód. Oczywiście mile widziane są narzędzia wewnętrzne, pozwalające na walidację poprawności kodu, czy narzędzia objaśniające sposób prowadzonego rozumo-

[15] Domyślna strategia *salience* oznacza, że to użytkownik decyduje, które reguły będą uaktywnianie przed innymi (i nada im w tym celu np. wyższy priorytet).
[16] Specjalna funkcja *do-backward-chaining name* (przed definiowaniem reguł) narzuca silnikowi regułowemu wnioskowanie wstecz.
[17] Dana reguła staje się uaktywniona w momencie, gdy *pattern matcher* potwierdzi, że lewa strona reguły ma pokrycie w bazie faktów.
[18] Agenda zawiera uporządkowaną listę reguł do uaktywnienia.
[19] PC-Shell odpowiada na pytanie typu "jak?" – gdy po zakończonym wnioskowaniu i przedstawieniu użytkownikowi decyzji system tłumaczy, jak doszedł do takiego wniosku oraz "dlaczego?", które służy wyjaśnieniu potrzeby zadawania konkretnego pytania użytkownikowi. Ma on wtedy świadomość tego, jak ważna jest w danym momencie odpowiedź na zadane przez system pytanie.
[20] Jeśli kilka reguł ma tę samą konkluzję, będą mogły być w danym momencie uaktywnione i w pamięci podręcznej mamy do czynienia z redundancją faktów.

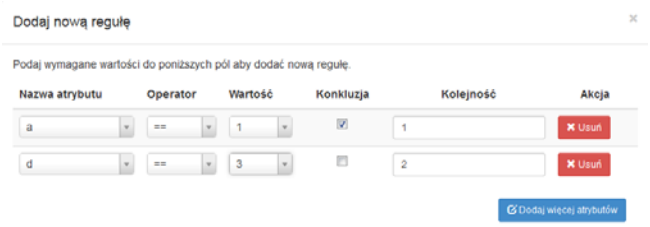
wania. Wszystkie te cechy stały się bezpośrednio argumentami przemawiającymi za potrzebą stworzenia alternatywnego narzędzia, które umożliwiłoby tworzenie RBW w wygodny (przyjazny dla użytkownika) sposób, a następnie eksplorację takich baz wiedzy. Część tych założeń została zrealizowana w systemie *kbExplorator*[21]. System ma pozwalać na tworzenie i eksplorację BW zarówno dzięki procesom wnioskowania z wszelkimi potrzebnymi objaśnieniami, jak i dzięki zaproponowanej strukturze *podziałów reguł*. System zaimplementowano przy użyciu takich środowisk programistycznych, jak PHP czy C+ oraz Java i JS. Architektura systemu przedstawiona została na rysunku 1[22].



Rys. 1. Architektura systemu kbExplorator
Fig. 1. kbExplorator's architecture

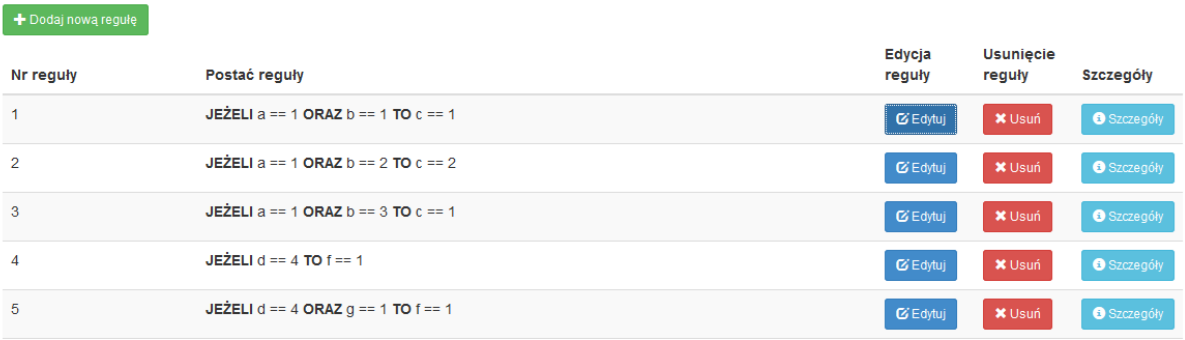
System jest wieloplatformowy i realizuje funkcję serwera z kontami użytkowników, którzy tworzą swoje dziedzinowe bazy na serwerze i tam korzystają z jego funkcjonalności. Konto użytkownika pozwala na użycie wygodnego kreatora BW do definicji atrybutów oraz reguł[23]. System kbExplorator składa się z kilku modułów: kreatora bazy wiedzy, modułu wnioskowania oraz eksploratora[24] BW. Użytkownik ma możliwość tworzenia BW z definicją dowolnej liczby atrybutów (i ich typów), dowolnej liczby wartości dla każdego atrybutu oraz dowolnej liczby reguł o dowolnej długości. Okno dodania nowej reguły przedstawia rys. 2.

[21] Udostępnienie prototypowej wersji systemu kbExplorator planowane jest na miesiąc lipiec br. Dostępny będzie pod adresem http://kbExplorer.ii.us.edu.pl.
[22] Na chwilę obecną zaimplementowano edytor bazy wiedzy (opisany w rozdziale 4 niniejszej pracy) z obsługą użytkowników i ich kont oraz część algorytmów wnioskowania. Do zrealizowania został moduł eksploracji bazy wiedzy.
[23] Nie ma ograniczeń co do liczby atrybutów i ich wartości, jak i liczby reguł. Rzeczywiste bazy wiedzy tworzone przy użyciu systemu zawierały nawet kilka tysięcy reguł.
[24] Eksplorator BW pozwoli inżynierowi wiedzy wychwytywać reguły, które prowadzą do niespójnych decyzji, jest narzędziem pozwalającym panować nad poprawnym działaniem takiego systemu. Usuwanie redundancji wśród reguł pozwoli minimalizować zużycie pamięci komputera do zapisu wiedzy nadmiarowej oraz redukować czas potrzebny na analizę tej samej informacji (wiele razy niepotrzebnie).



Rys. 2. Okno dodania nowej reguły do bazy wiedzy
Fig. 2. The new rule window

Podobną łatwością odznacza się opcja dopisywania/edycji atrybutów i ich wartości. Okno edycji reguł pozwala przeglądać wszystkie reguły i dowolnie je edytować (rys. 3).



Rys. 3. Okno zarządzania listą reguł
Fig. 3. Rules list window

Użytkownik dla tak stworzonej bazy wiedzy korzysta z możliwości wnioskowania w przód i wstecz. Wnioskowanie wyposażone jest w moduł objaśniający, pozwalający użytkownikowi śledzić i rozumieć drogę prowadzonego przez system rozumowania[25].

4.1. Modularyzacja bazy wiedzy – koncepcja *podziałów reguł*

System kbExplorator zakłada możliwość tworzenia RBW o dowolnej liczbie i długości reguł. Duża liczba reguł w BW nakłada na inżyniera wiedzy konieczność efektywnego zarządzania dużym zbiorem danych. Z tego względu, w systemie kbExplorator zaproponowano koncepcję tzw. *podziałów reguł*. Zakłada się, że baza wiedzy może być prosta i złożona[26] i że dla każdej bazy reguł *R*, liczącej n-reguł, istnieje skończony zbiór potęgowy 2^R o liczebności 2^n. Każdy, dowolnie utworzony podzbiór reguł $R \in 2^R$ nazywany będzie *grupą reguł*, zaś uzyskany podział reguł na grupy, zgodnie z pewnym przyjętym kryterium podziału, nazywać

[25] Eksploracja BW to bardzo ważna (kolejna po wnioskowaniu) funkcja systemu kbExplorator. Znajduje ona (o ile takowe istnieją) ukryte zależności między atrybutami bądź regułami czy też weryfikuje poprawność wiedzy tak zakodowanej. Wychwytuje także reguły redundantne, sprzeczne czy nietypowe. Pozwala to udoskonalać i poszerzać wiedzę zawartą w danej BW. Dla każdej reguły i faktu użytkownik ma do dyspozycji proste formularze, dzięki którym może definiować przesłanki i konkluzje, wybierając z listy wcześniej zdefiniowanych atrybutów i ich wartości. Szczegółowy opis eksploracji regułowych BW w kbExploratorze będzie treścią kolejnych prac autorów.

[26] Złożona BW to BW, w której konkluzja danej reguły jest przesłanką innej reguły w tej samej BW.

będziemy *podziałem reguł* PR⊆2^{R} [27]. Oczekiwanymi będą podziały tworzone przez konkretną tzw. *strategię podziału* (partition strategy)(PS) generującą dla bazy reguł podział PR = {R_1, R_2, …, R_k}, gdzie *k* określa liczbę grup reguł wchodzących w skład podziału PR, R_i to i-ta grupa reguł, oraz *i = 1, ..., k*. Koncepcja podziałów reguł bazuje na dwóch typach strategii podziału:

- Strategie proste: tworzą docelowy podział przez jednokrotny przegląd zbioru reguł *R* i przydział każdej reguły *r* do odpowiedniej grupy R_i, zgodnie z wartością funkcji mc(r,R_i)[28], np. strategia tworzenia podziału reguł o tej samej decyzji. Zwykle strategie proste osiągają docelowy podział przy złożoności nie większej niż O(nk), gdzie n=|*R*| i k=|PR|.
- Strategie złożone: najczęściej nie generują docelowego podziału w jednym kroku. Strategia złożona[29] zdefiniowana może być przez ciąg strategii prostych lub ich złożenie, także przez iterację pojedynczej strategii prostej, w której kryterium przynależności określane funkcją mc(r, R_i) zmienia się zgodnie ze specyfiką rozważanej strategii złożonej. Przykładem strategii złożonej jest strategia tworzenia podziału z reprezentantami grup, bazująca na analizie skupień.

4.2. Algorytmy tworzenia *podziałów reguł*

Algorytm tworzenia podziału bazujący na strategii prostej przedstawia się następująco. Dla każdej reguły *r*∈*R* sprawdza się, czy w poszukiwanym podziale PR istnieje grupa reguł R_i, taka że reguła *r* może być zaliczona do tej grupy (wartość funkcji mc(r,R_i)≤T). Jeżeli taka grupa nie istnieje, analizowana reguła *r* staje się zaczątkiem nowej, jednoelementowej grupy, która dołączana jest do tworzonego podziału PR. Złożoność obliczeniowa algorytmu to O(nd), gdzie n=|*R*|, d=|PR|. Szczególnym przypadkiem strategii prostej jest specyficzny podział nazywany dalej *selekcją*[30]. Koncepcja podziałów RBW na pewną liczbę grup (bądź wręcz na hierarchiczny podział bazy wiedzy na strukturę grup podziałów reguł) ma pozwolić na optymalizację procesów wnioskowania, poprzez (i) skrócenie czasu potrzebnego na prze-

[27] Zakładamy, że dla podstawowych strategii podział PR będzie rozłączny i kompletny, jednak dopuszczamy także możliwość użycia strategii podziału, które naruszą te założenia.

[28] O przynależności reguły r do pewnej grupy R_i⊆PR decyduje kryterium przynależności, określone przez funkcję *mc*, będącą liczbą z przedziału [0 .. 1]. Wartość 0 oznacza brak przynależności reguły r do grupy R_i, wartość 1 pełną przynależność, wartość z przedziału 0 < mc < 1 oznacza przynależność częściową.

[29] Podział początkowy poddawany jest później modyfikacjom, zgodnie ze specyfiką metody stanowiącej podstawę algorytmu budowania podziału.

[30] Selekcja dzieli zbiór reguł *R* na dwa podzbiory R_i oraz $R - R_i$ w taki sposób, że wszystkie reguły zawarte w *R* spełniają kryterium przynależności pewnej strategii podziału PS, a wszystkie pozostałe owego kryterium nie spełniają. Selekcja jest bardzo użyteczną strategią podziału i odgrywa kluczową rolę w analizie złożonych baz regułowych. W sensie praktycznym, wykonanie selekcji będzie operacją o złożoności liniowej O(n) względem liczby reguł *n*.

gląd wszystkich reguł w klasycznych bazach w celu znalezienia reguł do uaktywnienia, oraz (ii) wykorzystanie własności reprezentantów grup (tzw. *profili*) do szybkiego znajdowania odpowiednich grup reguł jak i efektywnej eksploracji reguł[31]. Ponadto, w idei przedstawionej metody zawarto założenie o własności podobieństwa mierzonej na różnych płaszczyznach i różnymi metodami. Podobieństwo może dotyczyć samych reguł, ale i grup reguł, a także podobieństwo reguły do danej grupy. W procesie wnioskowania podobieństwo będzie wykorzystywane przy wyszukiwaniu grupy/reguły do uaktywnienia. Dwie reguły mogą być przykładowo traktowane jako podobne, jeśli posiadają wspólne/podobne przesłanki i/lub gdy posiadają takie same/podobne konkluzje. Podobieństwo *s* części warunkowych reguł definiujemy za pomocą równania (1).

$$s(r_i, r_j) = \frac{|\, cond(r_i) \cap cond(r_j) \,|}{|\, cond(r_i) \cup cond(r_j) \,|} \tag{1}$$

gdzie licznik reprezentuje liczbę wspólnych przesłanek reguł r_i i r_j, zaś mianownik sumę różnych przesłanek obu tych reguł. Im wartość *s* będzie bliższa 0, tym mniej wspólnych przesłanek mają porównywane reguły, i odwrotnie: wartość *s* bliska 1 odpowiada sytuacji, gdy reguły porównywane mają niemal wszystkie przesłanki identyczne.

Dla każdej grupy reguł R_i można określić reprezentanta grupy. Reprezentant ów powinien charakteryzować dobrze grupę, do której jest przypisany, tj. określać obiekty ujęte w tej grupie. Podziały reguł, w zależności od wybranej strategii podziału, wymagają algorytmu prostego podziału (tzw. *selekcji*) bądź bardziej złożonych podziałów. Algorytm tworzenia podziału prostego będzie algorytmem sparametryzowanym, jednym z parametrów będzie właśnie funkcja *mc* oraz wartość progowa *T*[32].

4.3. Algorytmy wnioskowania dla regułowych baz wiedzy z podziałami reguł

Wśród strategii prostych i złożonych zakłada się strategie bazujące na idei podobieństwa reguł do innych reguł/grup. Jeśli reguły podobne do siebie połączymy w grupy, to w procesie wnioskowania w przód będzie nas interesowało znalezienie grupy/grup reguł pokrywającej/pokrywających zbiór faktów i/lub ewentualnie cel wnioskowania. Wystarczy zatem przejrzeć otrzymane w wyniku podziału *k* grupy (reprezentantów tych grup) i wybrać grupę najbardziej odpowiednią. Wnioskowanie może być wtedy realizowane tylko w obrębie wybranej grupy, co niewątpliwie skróci czas wnioskowania. Jednocześnie miara podobieństwa pozwoli

[31] Reprezentant grupy reguł może być traktowany jako tzw. metareguła dostarczająca wiedzy o wiedzy ujętej w regułach tej grupy. Nie ma potrzeby wówczas przeglądać każdej reguły w ramach grupy, wystarczy przejrzeć jej reprezentanta. Ponadto, reprezentant grupy informuje tzw. inżyniera wiedzy o tym, jak wiele reguł zawiera dana grupa i jaką wiedzę zawiera. To może pozwalać na dążenie do wzbogacania bazy wiedzy w kwestiach niedostatecznie dotąd zdiagnozowanych.

[32] Algorytm będzie kwalifikował rozważaną regułę r do grupy R_i, jeżeli wartość funkcji $mc(r, R_i) \leq T$.

znajdować reguły najbardziej podobne, a więc niekoniecznie pokrywające w pełni zbiór faktów. Klasyczne wnioskowanie wybiera jedynie *reguły pewne*, zaś proponowana tu koncepcja pozwoli także uaktywniać tzw. *reguły prawdopodobne*[33].

Wśród strategii prostych jest strategia grupowania reguł o tych samych/podobnych konkluzjach [8]. Dzieląc całą bazę wiedzy na grupy reguł o identycznych bądź podobnych konkluzjach, pozwalamy na optymalizację procesów wnioskowania, bowiem wystarczy znaleźć grupę reguł z konkluzją taką jak cel wnioskowania i przeprowadzić proces dowodzenia jedynie na regułach z danej grupy. Niewątpliwie wpłynie to na skrócenie czasu wnioskowania.

4.4. kbExplorator a inne narzędzia – porównanie

Tabela 2

Porównanie systemu kbExplorator do innych narzędzi

Własność	kbExplorator
Język implementacji	Java/C++/PHP/JS
kreator BW	Tak
modularyzacja BW	Tak
Strategie wnioskowania i metody wnioskowania	w przód i wstecz
Dostępność	Darmowe
wieloformatowość BW	Tak
format BW	TXT/XML
moduł objaśniający	Tak[34]
weryfikacja poprawności BW	Tak
Typ danych	jak integer, float oraz string
Definiowanie funkcji/procedur	Nie
Wady	brak

System kbExplorator powstał w odpowiedzi na ograniczenia dostępnych narzędzi, mających na celu tworzenie baz wiedzy i realizację procesów wnioskowania. Dodatkowym zadaniem stała się eksploracja baz wiedzy, rozumiana nie tylko jako wydobywanie nowej wiedzy w procesie wnioskowania, ale i identyfikacja niewidocznych na pierwszy rzut oka powiązań między regułami, mających wpływ zarówno na szybkość działania systemu, jak i jakość dostarczanej przez niego wiedzy. Podstawowymi założeniami systemu kbExplorator stały się: wieloplatformowość, intuicyjny interfejs użytkownika, pozwalający w możliwie najprostszy sposób tworzyć i edytować dziedzinowe BW, modularyzacja BW czy realizacja modułu ob-

[33] Reguły *pewne* pokrywają w pełni wszystkie przesłanki, zaś reguły *prawdopodobne* nie pokrywają w pełni przesłanek, a jedynie wykazują wystarczająco wysokie podobieństwo do podanych faktów.

[34] Jednym z głównych założeń systemu kbExplorator ma być możliwość eksploracji RBW rozumiana m.in. jako wydobywanie wiedzy z danych na etapie wnioskowania, jak i identyfikację ukrytych (nieoczywistych) zależności w danych. Moduł objaśniający będzie miał zatem nie tylko możliwość przedstawienia użytkownikowi procesu swojego rozumowania, ale i prezentację odkrytych zależności w danych. Docelowo, eksploracja ta ma być wyposażona nie tylko w objaśnienia opisowe (tekstowe), ale i graficzne (autorzy systemu świadomi są bowiem tego, jak wiele możliwości daje wizualizacja dużego zbioru danych, zwłaszcza takich specyficznych danych, jakimi są reguły, tj. łańcuchy przyczynowo-skutkowe).

jaśnień. Tabela 2 przedstawia ocenę parametrów systemu kbExplorator w porównaniu do innych narzędzi opisanych w tabeli 1. Wszystkie najbardziej kluczowe elementy udało się zrealizować w tworzonym systemie kbExplorator.

5. Podsumowanie

W pracy przedstawiono ideę działania systemu kbExplorator, mającego na celu tworzenie RBW opartych na idei *podziałów reguł* oraz realizację procesów wnioskowania dla takich BW. By uzasadnić potrzebę tworzenia nowego narzędzia do realizacji tzw. dziedzinowych baz wiedzy, dokonano przeglądu dostępnych narzędzi tego typu (tabele 1 i 2). Przedstawiono podstawowe definicje i pojęcia dotyczące grupy reguł, podziału i strategii podziałów. Omówiono ideę wnioskowania w przód i wstecz dla tak utworzonych baz wiedzy z *podziałami reguł* oraz krótko przedstawiono podstawowe funkcje systemu. Szczegółowe informacje dotyczące realizowanych algorytmów przedstawiać będą kolejne prace autorki.

6. Podziękowania

Niniejsza praca jest częścią projektu „Eksploracja regułowych baz wiedzy” sfinansowanego ze środków Narodowego Centrum Nauki (NCN: 2011/03/D/ST6/03027).

BIBLIOGRAFIA

1. Friedman-Hill E.: Jess in Action: Rule Based Systems in Java. Manning Publications. 2012.
2. http://docs.jboss.org/drools/release/5.2.0.Final/drools-expert-docs/html/ch01.html
3. https://krzysztof-michalik.com/my-ai-systems/pc-shell-hybrid-expert-system/
4. Forgy C.: A network match routine for production systems. Working Paper, 1974.
5. Nowak-Brzezińska A.: Optymalizacja systemów wspomagania decyzji. Studia Informatica, Vol. 32, No. 2A(96), Gliwice 2011, s. 403÷416.
6. Nowak-Brzezińska A., Simiński R.: ‚New inference algorithms based on rules partition. CS&P 2014, Informatik-Berichte, Vol. 245, Humboldt-University, Chemnitz, Germany 2014.

7. Nowak-Brzezińska A., Simiński R.: Knowledge mining approach for optimization of inference processes in rule knowledge bases. Lecture Notes in Computer Science, Vol. 7567, Springer, Berlin, Heidelberg 2012, s. 534÷537.
8. Simiński R.: Extraction of Rules Dependencies for Optimization of Backward Inference Algorithm. Communications in Computer and Information Science, Vol. 424, Springer, 2014, s. 191÷200.

Abstract

The article presents the idea of rules partition for rule-based knowledge bases' exploration. It also presents the concept of the system kbExplorator for the creation and exploration of knowledge bases and the implementation of inference processes. To justify the need of creating a new tool three available tools: Jess, Drools and PC-Shell were characterized and compared to the kbExplorator co-created by the author. Apart from formal definition, an example of the knowledge base with rules partition was shown, and application process implemented for such a knowledge base was discussed.

Adres

Agnieszka NOWAK-BRZEZIŃSKA: Uniwersytet Śląski, Instytut Informatyki,
ul. Będzińska 39, 41-200 Sosnowiec, Polska, agnieszka.nowak@us.edu.pl.