

XII Konferencja PLOUG  
Zakopane  
Październik 2006

# Aspekty narzędziowe w projekcie systemu do analizowania naruszeń prawa

Czesław Jędrzejek, Jarosław Bąk, Jolanta Cybulka, Jacek Martinek

*Centrum Doskonałości w dziedzinie Telematyki, Instytut Automatyki i Inżynierii Informatycznej,  
Politechnika Poznańska, Pl. M. Skłodowskiej-Curie 5, 60-965 Poznań  
e-mail: jkkbak@poczta.wp.pl*

## **Streszczenie**

Przedstawia się założenia i cele projektowe Systemu Analizatora Faktów i Związków (AFIZ), który jest pomyślany jako wsparcie w dochodzeniach kryminalnych. Projektowany system będzie wspomagać wykrywanie naruszeń prawa oraz nadzorować i ewidencjonować gromadzenie materiałów dowodowych. Omawia się standardy, metody i narzędzia przydatne do zastosowania w systemie, w szczególności standardy języków reprezentowania metadanych i wiedzy w systemie wraz z przetwarzającymi je narzędziami.



# 1. Założenia projektowe systemu do analizowania naruszeń prawa

## 1.1. Cel przedsięwzięcia i architektura systemu

Celem projektu jest wykonanie prototypu Systemu Analizatora Faktów i Związków (AFIZ), który będzie wspomagał przeprowadzanie dochodzeń kryminalnych umożliwiając automatyczne analizowanie faktów i odkrywanie nowych powiązań zachodzących pomiędzy nimi. Realizowany system będzie wspomagać wykrywanie przestępstw gospodarczych związanych z obrotem pieniężnym, realizowanie czynności śledczych i gromadzenie dowodów przestępstw.

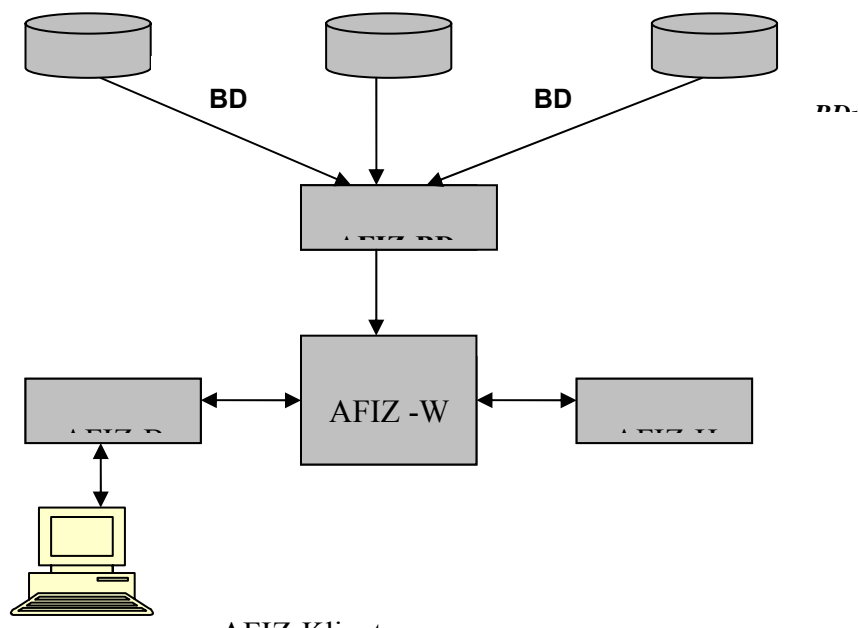
Wspomagana analiza materiału dowodowego ma szczególne znaczenie przy postępowaniach wykorzystujących dane informatyczne. Ze względu na ich ogromną ilość oraz często trudno czytelną formę, wspomagana komputerowo analiza danych jest niezbędna do efektywnego wykorzystania tego typu danych w postępowaniu dowodowym. Projektowany system będzie wspomagać wykrywanie naruszeń prawa oraz nadzorować i ewidencjonować gromadzenie materiałów dowodowych. System będzie uwzględniał struktury danych i formy reprezentacji wykorzystywane przez systemy stosowane aktualnie w polskiej policji, przy znacznym rozszerzeniu i częściowym zautomatyzowaniu procesów analizy informacji pod kątem wykrywania naruszeń prawa. System AFIZ będzie importował informacje z różnych baz danych, zawierających dane o osobach (w szczególności o osobach, które naruszyły prawo), o grupach przestępczych, instytucjach, pojazdach lub innych przedmiotach, a także o zdarzeniach lub przebiegach zdarzeń, wiążących się z naruszeniami prawa. Opracujemy jednolitą charakterystykę semantyki informacji zawartych w tych bazach danych, a także szablony importu danych do projektowanego systemu.

Wiedzę wykorzystywaną w rozważanym systemie można podzielić na wiedzę potrzebną do zaprojektowania systemu oraz na dane robocze używane w trakcie jego działania. W skład wiedzy potrzebnej do zaprojektowania systemu wejść:

- przepisy prawa związane z przestępstwami wchodzącymi w zakres zastosowań systemu,
- przepisy prawa dotyczące metod ścigania przestępstw,
- wiedza o przestępstwach i metodach ich wykrywania stosowana przez funkcjonariuszy policji,
- informacja o rodzajach i schematach baz danych używanych przez policję,
- opisy metod analizy danych, dotyczących przestępstw, także te znane z publikacji.

Dane robocze, wykorzystywane w trakcie działania systemu, będą pobierane z policyjnych baz danych (z uwzględnieniem ustawy o ochronie informacji niejawnych), a także uzyskiwane bezpośrednio z Internetu oraz od użytkowników systemu. Pozyskane dane będą reprezentowane zarówno syntaktycznie, jak i semantycznie. Ponadto, autorzy zamierzają zastosować opracowane dotychczas własne metody w zakresie reprezentowania przepisów prawa ([MaCy05]) oraz budowy i stosowania metadanych semantycznych w postaci ontologii ([CyMa05, CMP06]).

Ogólną architekturę systemu AFIZ złożonego z czterech głównych podsystemów zobrazowano na rys. 1:



Rys. 1. Struktura modułarna systemu AFIZ

1. AFIZ-W – podsystem wnioskujący, służący do analizowania zebranych danych; w jego ramach funkcjonują działania odpowiedzialne za zebranie i przygotowanie danych z istniejących repozytoriów;
2. AFIZ-BD – adapter integrujący dane pochodzące z istniejących repozytoriów informatycznych (policyjnych i będących własnością organów śledczych); utworzony bank (baza) danych służy do przechowywania zebranych danych w celu dokonywania ich analizy oraz do przechowywania rezultatów tej analizy;
3. AFIZ-H – maszyna wnioskująca do analizowania i weryfikowania hipotez śledczych;
4. AFIZ-B – podsystem zapewniania bezpieczeństwa.

## 1.2. Charakterystyka podsystemu wnioskującego

Jak już wspomniano, system AFIZ-W będzie przetwarzał dane pochodzące z różnorodnych źródeł zewnętrznych, co wiąże się z potrzebą ich integrowania, zwłaszcza na poziomie semantycznym. W tym celu zostaną zbudowane semantyczne bazy danych (zawierające metadane) oraz odpowiednie szablony do importowania danych zewnętrznych do systemu.

Wiedza wykorzystywana w trakcie działania systemu będzie miała postać specjalnie opracowanych reguł deklaratywnych, specyfikujących procesy analizy danych, pozyskanych ze wspomnianych źródeł. Jako przykład rozpatrzmy regułę dotyczącą wyludzenia ubezpieczeń, która może przyjąć postać:

osoba A przypuszczalnie wyludziła odszkodowanie za utratę przedmiotu B  
od ubezpieczyciela C w dniu D **jeśli**

(osoba A uzyskała odszkodowanie za utratę przedmiotu B od ubezpieczyciela C w dniu D i

osoba A1 sprzedała przedmiot B1 w dniu D1 i

przypuszczalnie A = A1 i

przypuszczalnie  $B = B1$  i  
dzień D nie jest odległy od dnia D1).

Jest to reguła warunkowa, odpowiadająca klauzuli języka programowania Prolog. Spełnienie warunków ujętych w nawiasy () pozwala wysnuć wniosek o przypuszczalnym wyłudzeniu odszkodowania. Wielkimi literami oznaczono zmienne, za które podstawia się konkretne dane. Dane mogą być reprezentowane za pomocą struktur cech. Powiedzmy, że od ubezpieczyciela uzyskano następujące dane o utraconym telefonie komórkowym zawarte w strukturze B, zaś obserwacje dotyczące przedmiotu sprzedawanego (struktura B1) nie są dokładne i mają postać:

B = [ 'telefon komórkowy',  
producent, Nokia,  
seria, 7360,  
kolor, brązowy ]

B1 = [ 'telefon komórkowy',  
kolor, ciemny ].

Informacja zawarta w B1 nie jest bezpośrednio zgodna z informacją w B. Jeśli jednak system dysponuje semantycznymi metadanymi (np. w formie ontologii), z której można odczytać, że:

ciemny 'jest podobne do' brązowy,

to B 'jest zgodne z' B1 i warunek  $B = B1$  przypuszczalnie jest spełniony.

Rozważaną ontologię zbudujemy zgodnie z metodologią DOLCE wzbogaconą o D&S [OL-G+05] (deskrypcje i sytuacje) do zbudowania modelu konceptualnego obejmującego przedstawioną sytuację dotyczącą wyłudzenia ubezpieczeń. W metodologii zakłada się, że część bytów „wprost” konstytuuje rzeczywistość, a część stanowi zreifikowane ontologicznie konteksty, wprowadzane przez dziedziny istniejące jedynie umownie (tzn. zdefiniowane przez człowieka), jak prawo, socjologia, psychologia itp. Część „rzeczywista” składa się na koncept Sytuacja, a część „umowna” – na pojęcie *Deskrypcja Sytuacji*. Obydwie „części” są powiązane relacją *spełniania* (*Deskrypcja Sytuacji spełnia Sytuację*), czyli opis sytuacji ma model w postaci sytuacji rzeczywistej.

Analizując frazy rzeczownikowe i czasownikowe użyte w zaprezentowanej regule i strukturach cech, zauważamy, że do świata rzeczywistego należą pojęcia<sup>1</sup>:

- 1) *Osoba* jako specjalizacja pojęcia Racjonalny Obiekt Fizyczny,
- 2) *Instytucja Ubezpieczająca* jako specjalizacja Instytucji,
- 3) *Telefon Komórkowy* specjalizujący Materialny Artefakt,
- 4) *Kolor* specjalizujący Jakość Fizyczną (fizyczny atrybut „przylegający do bytu”),
- 5) *RGB* specjalizujący Jednostkę Miary (określa paletę kolorów w formacie RGB, które służą do wartościowania jakości typu Kolor).

Pojawiają się także pewne rzeczywiste *Zdarzenia*, opisane za pomocą Przebiegów Zdarzeń. Natomiast deskrypcje będą dotyczyły sytuacji: „sprzedaży towaru”, „uzyskania odszkodowania” i „wyłudzenia odszkodowania”, będących w istocie agregatami deskrypcji: przebiegów zdarzeń, ról uczestników zdarzeń oraz parametrów, czyli okoliczności towarzyszących uczestnictwu ról w przebiegach zdarzeń. W tym kontekście wyróżnimy:

- 1) specjalizacje konceptu Przebieg Zdarzeń: *Sprzedaż*, *Uzyskanie Odszkodowania* i *Wyłudzenie Odszkodowania* (wszystkie przebiegi zdarzeń są Zadaniem Złożonymi),
- 2) Role Komercyjne: *Towar*, *Odszkodowanie*, *Ubezpieczyciel*, *Ubezpieczony*, *Sprzedający*,

<sup>1</sup> Czcionką Arial zaznaczono koncepty DOLCE+D&S.

- 3) Role Socjalne: *Podjejrany o Wyludzenie Odszkodowania, Utracone Dobro*,
- 4) Parametry: *Dzień Sprzedaży, Dzień Uzyskania Odszkodowania*, wartościowane w Regionie Czasowym,
- 5) Parametr: *Skala Odcieni*, specjalizowany przez *Ciemny* i *Jasny*, wartościowany w przestrzeni *RGB*,
- 6) Parametr: *Nazwa Koloru*, specjalizowany przez m.in. *Brązowy*, wartościowany także w przestrzeni *RGB*.

W ontologii odniesienia zdefiniowano zestaw relacji formalnych wiążących pojęcia „rzeczywiste” z ich „deskrypcjami”. Podamy kilka przykładów. Role są odgrywane przez byty „rzeczywiste”, np. *Telefon Komórkowy* odgrywa rolę *Towaru* i *Utraconego Dobra*. *Osoba* gra rolę *Ubezpieczonego*, *Sprzedającego*, *Podjezranego o Wyludzenie Odszkodowania*, *Institucja Ubezpieczająca* gra rolę *Ubezpieczyciela* i in. Tak rozumiana konceptualizacja wspiera rozumowanie, że osoba sprzedająca, ubezpieczona i wyludzająca odszkodowanie może być tą samą osobą. Podobnie rzecz się ma z *Telefonem Komórkowym*, w odniesieniu do którego z analizy wartości atrybutu jakościowego *Kolor* wynika, że:

ciemny 'jest podobne do' brązowy

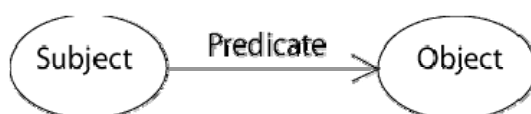
(przedział wartości RGB dla odcienia ciemnego zawiera w sobie wartości RGB dla koloru brązowego). Także analiza wartości przypisanych dniom pozwala wnioskować o ich nieodległej czasowej lokalizacji.

## 2. Standardy do wykorzystania przy realizacji systemu

Istotną właściwością projektowanego systemu jest możliwość reprezentowania semantyki przetwarzanych danych. W tym celu można wykorzystać kilka istniejących standardów w zakresie tzw. *metadanych*, czyli danych opisujących inne dane. Idea metadanych pojawiła się około 10 lat temu towarzysząc pomysłowi zbudowania semantycznej sieci WWW (*semantic Web* [SBW06, TR06]). Współcześnie, do najbardziej znanych modeli reprezentacji metadanych należą *RDF* (*Resource Description Framework*, [LaSw]) wraz z językiem schematów *RDFS* (*RDF Schema*, [BrGu]) oraz ontologie wyrażane w języku *OWL* (*Web Ontology Language*, [OWL]). Zarówno *RDF(S)*, jak i *OWL* mogą być przechowywane w plikach tekstowych, jak i w wysoce ustrukturalizowanych magazynach, jakimi są relacyjne bazy danych, wyposażone w specjalizowane języki zapytań, np. *SPARQL* [SPARQL]. Pojawiły się także próby łączenia języków do reprezentowania metadanych z językami reprezentacji wiedzy w postaci regułowej i w postaci programów w logice.

### 2.1. Języki RDF i RDFS do reprezentowania metadanych

*RDF* jest modelem służącym do opisywania zasobów sieciowych identyfikowanych za pomocą *URI*. Każdy opis zasobu (*resource*, *subject*) jest zbiorem stwierdzeń (*statements*), z których każde specyfikuje właściwość (*predicate*) zasobu oraz wartość tej właściwości (*object*). Pojedyncze stwierdzenie ma postać trójki: (*subject*, *predicate*, *object*), reprezentowanej na różne sposoby, np. w postaci elementarnego grafu przedstawionego na rys. 2, w zapisie wyrażonym w języku *XML*, w języku trójkowym *N3* i innych. Stwierdzenia mogą podlegać reifikacji stając się zasobami, które mogą być również opisywane poprzez stwierdzenia wyższych rzędów.



Rys. 2. Stwierdzenie RDF w postaci diagramu (grafu)

Zasoby mogą być grupowane w złożone struktury zwane *kontenerami* (*bag*, *alternative* lub *sequence*), a także typowane za pomocą konceptów (*class*) definiowanych w języku RDFS. Język RDFS umożliwia tworzenie hierarchii taksonomicznej klas za pomocą subsumpcji (właściwość *rdfs:subClassOf*) oraz takichże dla relacji (właściwość *rdfs:subPropertyOf*), a także określanie powiązań pomiędzy klasami za pomocą relacji (właściwości *rdfs:domain* i *rdfs:range*). Możliwe jest również posługiwanie się obiektami klas (typowanie zasobów za pomocą właściwości *rdf:type*). Pewne właściwości języka, jak reifikacja oraz dopuszczenie możliwości, że klasa może być swoją instancją powodują, że nie jest on wygodny do stosowania w praktyce.

Pojawiają się pierwsze rezultaty dotyczące praktyki i wydajności, dotyczące wdrożeń baz danych RDF. Istnieje co najmniej kilka magazynów przekraczających rozmiary 25 mln trójek RDF. Taki np. rozmiar osiągnął serwis Kongresu Stanów Zjednoczonych GovTrack.us [KoUSA], który publikuje informację na temat procesu legislacyjnego. Serwis ten udostępnia interfejs w języku SPARQL. Okazuje się, że przy tych wielkościach baz większość implementacji ma problemy ze skalowalnością systemów.

Największym magazynem danych RDF jest obecnie Ingenta MetaStore [PoPa06]. Projekt ten dotyczy budowy elastycznego i skalowanego repozytorium przechowującego metadane w zakresie 17 milionów artykułów i 20.000 publikacji o tematyce naukowej, technicznej i medycznej. Repozytorium obecnie zawiera około 200 milionów trójek o różnorodnej semantyce, poprzez FOAF, Dublin Core, PRISM. Średnie obciążenie strony WWW IngentaConnect [PrIn] wynosi około 2 milionów sesji miesięcznie. Choć system Ingenta MetaStore spełnił założenia wydajnościowe, twórcy systemu zmuszeni byli do kilku istotnych modyfikacji standaryzowanych rozwiązań W3C.

## 2.2. Język zapytań SPARQL

Stosowanie RDF w praktyce wymaga efektywnych narzędzi przetwarzających, a te zapewniają relacyjne bazy danych. Zdefiniowano SQL-owy język do manipulowania zapisami RDF o nazwie SPARQL. Zachodzą następujące kluczowe pytania:

- na ile język zapytań SPARQL różni się od języka SQL?
- jaka jest relacja zapytań SPARQL do operacji wnioskowania oraz
- jak wydajna jest technologia bazodanowa oparta na RDF?

Ponieważ trójki RDF mogą być reprezentowane w tabeli relacyjnej o trzech kolumnach, wydaje się, że wystarczyłoby zastosowanie klasycznego języka SQL. Jednakże, dane RDF są często przechowywane w plikach tekstowych formatu XML, co utrudnia bezpośrednie stosowanie SQL i SPARQL może być w takim wypadku przydatny.

Obydwie formy reprezentacji (plik tekstowy, baza danych) mają zarówno zalety, jak i wady. Model relacyjny wraz z SQL został zaprojektowany do przechowywania ogromnych ilości danych strukturalnych (regularnych), które zazwyczaj zawierają wartość w każdej kolumnie każdej tabeli. SQL definiuje specjalną wartość zwaną *null value*, która umożliwia reprezentowanie informacji nieznannej lub danej nieodpowiedniego typu. Występowanie takich wartości komplikuje jednak definicję modelu oraz sprawia, że formułowanie i realizowanie zapytań jest dość trudne w przypadku danych o mniej widocznej strukturze. Dla danych RDF taka sytuacja, np. powstała w wyniku wykonania operacji UNION prowadzi do odpowiedzi, dla której jedna ze zmiennych nie ma rozwiązania (przypadek „*unbound*”).

Przy porównaniu schematów SQL i schematów RDF (i tak samo schematów XML) należy wziąć pod uwagę kwestię normalizacji. Relacyjne bazy danych po normalizacji nie posiadają pełnych informacji o strukturze. W wielu przypadkach format RDF odpowiada modelowi związków encji. Tabele SQL z kluczem głównym o *n* kolumnach mogą zostać zdekomponowane do *n-1* związków encji dla każdego wiersza. Każda tego typu asercja ma postać: *wartość\_klucza*, *na-*

*zwa\_kolumny*, *wartość\_kolumny* (na przykład: *Kowalski zarabia 6000.00*). Takie proste podejście pozwala na transformację tabel SQL w grafy RDF. Podobnie instancje modelu danych XPath mogą być dekomponowane w asercje związków relacji postaci: *id\_węzła\_rodzicielskiego*, *potomek\_lub\_atrybut*, *element\_potomka\_lub\_wartość\_atrybutu\_lub\_id\_węzła* (na przykład: *węzeł\_Kowalski\_atrybut\_data\_urodzin\_21.01.1970*).

Dodatek do standardu SQL, czyli SQL/XML pozwala reprezentować dane relacyjne w postaci XML (dzięki instancjom języka ścieżek XPath). Standard ten umożliwia odwzorowywanie całych tabel w strukturę XML, nie pozwala to jednak uchwycić znaczenia (semantyki) samych danych. Oczywiście, w każdym przypadku odwzorowywanie może być zaprojektowane w sposób, który umożliwia uchwycenie większej części semantyki. Jednak możliwość transformacji całej semantyki pozostaje obecnie pytaniem otwartym. W podobny sposób do transformacji danych pomiędzy modelami, można również tłumaczyć języki zapytań powiązane z każdym modelem. Wiadomo, że istnieje implementacja języka XQuery umożliwiająca transformację zapytania XQuery w zapytanie SQL. Istnieje również możliwość translacji większości (o ile nie wszystkich) zapytań języka SPARQL do postaci SQL i podobnie zapytania SPARQL-a można przekształcić na zapis w XQuery.

Porównamy języki SQL i SPARQL przedstawiając te „same” dane w dwóch formach odpowiadających formatom poszczególnych modeli danych oraz formułując odpowiednie zapytania. Pierwszy format ma postać tabel relacyjnej bazy danych, w którym zgromadzono dane o przestępcach (tab.1) i o skradzionych przedmiotach (tab.2). Drugim formatem są trójki RDF (tab. 3). Następnie przedstawimy zapytanie wyrażone w językach SQL i SPARQL, które w języku naturalnym brzmi: „Który przestępca jest oskarżony o kradzież?”

Tabela 1. Tabela Przestępcy

ID	Nr_sprawy	Nazwisko	Imię
10	3212/2005	Kowalski	Janusz
...	...	...	...
95	1432/2006	Nowak	Aleksander

Tabela 2. Tabela Przedmioty

ID	Przedmiot_kradzieży	Przestępca
12	Telefon komórkowy	95
...	...	...
76	Samochód	95

Zapytanie w języku SQL odpowiadające rozważanemu pytaniu przyjmuje postać:

```
SELECT L.Nazwisko
FROM Przestępcy AS L JOIN Przedmioty AS P
USING L.ID = P.Przestępca
```

W tab. 3. użyto następujących XML-owych przestrzeni nazw:

```
przes: http://przes.example.org/przestepcy#
przed: http://przed.example.org/przedmioty#
rdb: http://databases.example.org/
```

Tabela 3. Reprezentacja danych z dla tabel Przestępcy i Przedmioty w formacie RDF

Zasób	Właściwość	Wartość właściwości
przes:id10	rdb:przestepcy/column#nr_sprawy	3212/2005
przes:id10	rdb:przestepcy/column#nazwisko	Kowalski
przes:id10	rdb:przestepcy/column#imie	Janusz

przes:id95	rdb:przestepcy/column#nr_sprawy	1432/2006
przes:id95	rdb:przestepcy/column#nazwisko	Nowak
przes:id95	rdb:przestepcy/column#imię	Aleksander
przes:id...	...	...
przed:id12	rdb:przedmioty/column#przedmiot_kradziezy	Telefon komórkowy
przed:id12	rdb:przedmioty/column#przestepca	95
przed:id76	rdb:przedmioty/column#przedmiot_kradziezy	Samochód
przed:id76	rdb:przedmioty/column#przestepca	95
przed:id...	...	...

Zapytanie wyrażone w składni SPARQL wygląda następująco:

```
SELECT ?Nazwisko
WHERE
{
  ?l rdb:przestepcy/column#nazwisko ?Nazwisko
  ?p rdb:przedmioty/column#przestepca ?l
}
```

Na podstawie powyższego przykładu można by wywnioskować, że pisanie zapytań w obydwu językach charakteryzuje się podobnym stopniem skomplikowania i są one wyrażalne za pomocą formuł o zbliżonej wielkości. Nie jest to jednak prawdą, jak pokazano w [M06].

Reprezentowane danych w postaci RDF niesie ze sobą dużo korzyści na poziomie koncepcyjnym. Nie jest to jednak gwarancją sukcesu komercyjnego tego formatu. Do osiągnięcia tego celu kolekcje RDF mogłyby być składowane w relacyjnej bazie danych a następnie odpytywane za pomocą SQL. Skoro tak, to dlaczego SQL nie może być językiem, który będzie działał przed i po wnioskowaniu? Można bowiem podać przykłady zapytań, dla których nie można w ramach modelu relacyjnego podać odpowiedzi, nawet dla pewnych instancji komponentów zbioru danych opisujących relacje między komponentami urządzenia [RaGe02].

Języki SQL i SPARQL różnią się wystarczająco np. w odniesieniu do wyniku operacji UNION, co również usprawiedliwia powstanie języka zapytań do kolekcji RDF. Opracowanie translacji wyrażeń SPARQL w wyrażenia SQL pozwoli użytkownikom na przechowywanie danych RDF w relacyjnych bazach danych, a zapytania będą mogły być zapisywane zarówno w składni SQL, jak i SPARQL w zależności od preferencji użytkownika. Ułatwiłoby to bardzo integrację obecnie istniejących systemów z nowymi technologiami takimi jak RDF i SPARQL. Trwają prace nad umożliwieniem serializacji kolekcji RDF w dokumenty XML, oraz translacji zapytań SPARQL na zapytania XQuery.

### 2.3. Ontologie i język OWL

Nowoczesnym modelem metadanych są struktury konceptualne zwane *ontologiami*. Pozwalają definiować semantykę poprzez specyfikowanie zamierzonego znaczenia (*intended meaning*) pojęć (klas) powiązanych za pomocą binarnych relacji. Pojęcia są zazwyczaj właściwościami spełnianymi przez przykłady pojęć (ich instancje), czyli są zbiorami takich przykładów. Pomędzy pojęciami (i relacjami) może zachodzić subsumowanie (specjalizowanie), a przykłady różnych pojęć mogą być powiązane za pomocą zdefiniowanych relacji. Relacje mogą mieć pewne właściwości algebraiczne, jak: symetria, przechodniość, bycie funkcją, czy posiadanie relacji odwrotnej. W odniesieniu do pojęć (traktowanych jak zbiory przykładów) można definiować ich synonimie, rozłączność, sumę, przecięcie i dopełnienie mnogościowe. Można także rozróżniać (lub utożsamiać) przykłady pojęć, a także definiować powiązania pomiędzy wskazanymi parami przykładów pojęć.

Tak rozumiane struktury ontologiczne są wyrażalne w języku OWL, mającym strukturę warstwową o rosnącym stopniu skomplikowania (i malejącej jednocześnie efektywności narzędzi przetwarzających zapisy ontologiczne). W języku OWL pojęcia można definiować jawnie albo poprzez wyrażenia nakładające pewne rodzaje więzów na relacje wiążące pojęcia. Warstwy OWL to:

- *OWL Lite* umożliwiająca głównie tworzenie taksonomii pojęć; definiowanie pojęć poprzez nakładanie więzów na relacje jest ograniczone do prostych więzów dotyczących liczności 0 lub 1 (tzn. można zadeklarować, że pewne pojęcie jest zbiorem przykładów powiązanych z przynajmniej jednym albo co najwyżej jednym, albo dokładnie jednym przykładem innego pojęcia); w warstwie *OWL Lite* nie można formułować ekstensjonalnych definicji pojęć;
- *OWL DL*, odpowiadająca semantycznie logikom deskrypcyjnym, umożliwia tworzenie złożonych struktur pojęciowych poprzez nakładanie kilku rodzajów więzów na relacje; nie można wszakże definiować dowolnych relacji zachodzących pomiędzy pojęciami, a jedynie pomiędzy ich przykładami, co jest konsekwencją założenia o separacji typów, tzn. pojęcie nie może być traktowane jak indywiduum albo jak relacja, podobnie, relacja nie może być traktowana ani jak indywiduum, ani jak klasa; efektem wprowadzenia tych ograniczeń jest obliczeniowa efektywność i rozstrzygalność systemów realizujących tę warstwę języka;
- warstwa *OWL Full* nie zawiera omówionych ograniczeń, za to nie ma zagwarantowanej efektywności i rozstrzygalności.

Zapis w języku OWL składa się z definicji ontologicznych metadanych (*container*), definicji relacji binarnych, pojęć, przykładów pojęć oraz aksjomatów definiujących omówione dodatkowe właściwości pojęć i relacji. Język OWL DL (odpowiadający formalnie logikom deskrypcyjnym) jest coraz częściej stosowany w praktyce, np. w systemach KAON, Protégé i Jena2.

## 2.4. Język SWRL do reprezentowania ontologii i reguł

Istnieje propozycja języka SWRL (*Semantic Web Rule Language*) opartego na połączeniu podjęzyków OWL DL i OWL Lite języka OWL z fragmentami języka RuleML (*Rule Markup Language*). Propozycja ta rozszerza zbiór aksjomatów zapisanych w języku OWL tak, aby obejmował reguły mające postać klauzul. Pozwala to łączyć takie reguły z bazą wiedzy zapisaną w języku OWL. Proponowane reguły mają postać implikacji pomiędzy poprzednikiem (ciałem) i następnikiem (nagłówkiem). Poprzednik i następnik mogą być zbudowane z koniunkcji atomów, pojedynczego atomu albo być puste. (Wiadomo, że regułę zawierającą koniunkcję w następniku można przekształcić w zestaw reguł zawierających tylko jeden atom w następniku). Reguła ma następujące znaczenie: jeśli warunki podane w poprzedniku zachodzą, to warunki podane w następniku także muszą zachodzić. Pusty poprzednik jest traktowany jako trywialnie prawdziwy, a pusty następnik jako trywialnie fałszywy.

Atomy w regułach mogą być postaci  $C(x)$ ,  $P(x,y)$ ,  $sameAs(x,y)$  lub  $differentFrom(x,y)$ , gdzie  $C$  jest deskrypcją pojęcia języka OWL,  $P$  jest własnością, a  $x$ ,  $y$  są albo zmiennymi, albo indywiduami, albo wartościami danych. Język SWRL ma większą siłę wyrazu niż OWL DL, ale jego zdania są tylko częściowo rozstrzygalne.

Narzędzia realizujące wnioskowanie dla języka SWRL mają charakter eksperymentalny.

## 2.5. Inne języki do reprezentowania wiedzy

Przy projektowaniu systemu do analizowania naruszeń prawa należy wziąć pod uwagę możliwość zastosowania języka Prolog, lub innych środków programowania w logice. Prolog posiada szereg zalet, do których należą zwiezłość i czytelność programów, oraz łatwość programowania i uruchamiania programów. W języku tym zaimplementowano wiele różnych maszyn wnioskujących, co pozwala traktować go jak język „niskiego poziomu” dla programowania w logice. Istnieją

systemy Prologu [SWI] obudowane narzędziami do tworzenia interfejsu z użytkownikiem oraz modułami do współpracy z bazą danych zapisaną w języku RDF. Trzeba jednak pamiętać, że strategia wnioskowania, stosowana w większości systemów Prologu (strategia SLDNF), może być źródłem niepoprawnego działania systemu dla niektórych programów. Na przykład, jeśli napiszemy program, definiujący relację *x jest przodkiem y*,

```
przodek(X, Y) :- przodek(X, Z), rodzic(Z, Y).  
przodek(X, Y) :- rodzic(X, Y).
```

to przy stosowaniu wspomnianej strategii program zapętlę się.

Istnieją już systemy programowania w logice, na przykład system XSB [XSB], które stosują bardziej wymyślną strategię SLG, pozwalającą uniknąć wspomnianych problemów. Strategia ta posługuje się tablicowaniem pośrednich rezultatów obliczeń. We wspomnianym systemie rozbudowano także techniki indeksowania danych, co pozwala poprawiać skuteczność wyszukiwania danych i polepszać skalowalność systemu realizowanego za pomocą programowania w logice.

### 3. Narzędzia i środowiska do wykorzystania przy realizacji systemu

#### 3.1. Serwer SPASQL – SPARQL dostarczony wraz z MySQL

W pracy [P06] przedstawiono SPASQL – zmodyfikowany serwer MySQL, który umożliwia przetwarzanie zarówno zapytań SQL, jak i SPARQL. Język SPARQL jest interpretowany bezpośrednio na serwerze bazy danych. Podejście to zwiększa wydajność zadawania zapytań oraz umożliwia aplikacjom łączenie się z bazą danych poprzez konwencjonalny protokół MySQL. Zatem, dane mogą być przetwarzane przez aplikacje tak, jak zwykle dane relacyjne. W praktyce oznacza to zamianę zapytania ze skryptu PHP na zapytanie SPARQL, zamiast na zapytanie SQL. Poniżej znajduje się przykład użycia zapytania SPARQL w skrypcie PHP w miejscu, w którym zazwyczaj znajduje się zapytanie w postaci SQL:

```
<? mysql_connect (localhost, $username, $password);  
  @mysql_select_db($database) or die( "Nie można znaleźć bazy danych");  
  mysql_query("SPARQL:  
  SELECT ?ulica ?lokal  
  WHERE  
  {  
    ?o <Zamowienia.AdresDostawy> ?ulica  
    ?ulica <Adresy.lokal> ?lokal  
  };");  
?>
```

Schematy relacyjne charakteryzują się sztywną strukturą, która pociąga za sobą konieczność dokonywania znaczących zmian w schemacie, np. w przypadku potrzeby dodania nowego atrybutu. Przechowywanie dodatkowych atrybutów w postaci trójek pozwala bazie danych na jednoczesne radzenie sobie z danymi w postaci stałej (tabeli), jak również danych nieregularnych. Takie podejście sprawia, iż zapytania postaci relacyjnej mogą zostać połączone z zapytaniami operującymi na trójkach danych, co pozwala na połączenie wydajności z elastycznością. Jest to niewątpliwie kompromis pomiędzy wydajnością operowania na danych relacyjnych a możliwościami, jakie niesie ze sobą format RDF, co uosabia ideę heterogenicznych baz danych.

Aby przekonwertować pytanie SPARQL do SQL należałoby najpierw przekształcić dane w formacie relacyjnym na dane w formacie RDF. Poprawnie przekształcone zapytanie SQL na przekształconych danych ma miejsce wtedy, gdy jego rezultat jest taki sam, jak w przypadku oryginalnego zapytania SPARQL na oryginalnych danych RDF.

### 3.2. Model danych RDF w Oracle Spatial 10g Release 2

Oracle wspomaga tworzenie sieci semantycznej jako ewolucji do następnej fazy zarządzania danymi tworząc centrum technologii semantycznych (*Semantic Technologies Center*). Produkt Oracle Spatial 10g Release 2 jest pierwszą komercyjną platformą do zarządzania aplikacjami opartymi na RDF [ORACLE]. Wprowadzono nowe typy danych do zarządzania strukturami RDF. Trójki RDF można ładować do magazynu danych, indeksować i budować do nich zapytania. W SQL stosuje się funkcje RDF\_MATCH, która działa podobnie jak SPARQL. Ponad 100 firm partnerskich testowało i wspiera Oracle Spatial 10g Release 2.

### 3.3. System Jena

Jena [JENA] jest systemem do tworzenia aplikacji dla sieci semantycznej. Udostępnia szereg maszyn wnioskujących między innymi dla języków RDFS, OWL, SPARQL oraz maszynę wnioskującą dla reguł, mającą przeznaczenie ogólne. Pozwala to wywodzić dodatkowe fakty z danych modelu i deskrypcji klas oraz programować różne zadania przetwarzania lub transformowania danych zapisanych w języku RDF. Maszyna wnioskująca ogólnego przeznaczenia może działać w trybach progresywnym, regresywnym (połączonym z tablicowaniem) lub hybrydowym.

W trybie progresywnym każda reguła, która zostanie uruchomiona, może generować dodatkowe trójki, umieszczając je w grafie dedukcji, może usuwać je oraz może włączać dalsze reguły. Taka kaskada uruchomień reguł jest kontynuowana tak długo, dopóki nie można już uruchomić żadnej z nich.

W trybie regresywnym stosuje się strategię wnioskowania podobną do strategii Prologu. Pytanie przekształca się w cel i próbuje się go rozwiązać przez uzgadnianie z zapamiętanymi trójkami i przez stosowanie reguł regresywnych. Język reguł jest ograniczony, bowiem, podobnie jak w Datalogu [Datalog], nie można stosować termów złożonych. Ogranicza to typy danych, nie pozwalając przetwarzać struktur danych o charakterze rekurencyjnym. W trybie regresywnym umożliwiono stosowanie tablicowania pośrednich rezultatów obliczeń. Jeśli cel jest tablicowany, wtedy zapamiętywane są wszystkie wyliczone uzgodnienia dla tego celu i w razie, jeśli odpowiadają przyszłym celom, to można je później wykorzystać. Gdy w taki sposób zostaną skonsumowane wszystkie znane odpowiedzi, to cel zawiesza się, dopóki pewna inna gałąź obliczeń nie wygeneruje nowych rezultatów i wtedy następuje wznowienie obliczania celu. Pozwala to wykonywać sukcesywnie takie reguły rekurencyjne (jak, na przykład, wyrażające przechodnie domknięcie), które wywołują nieskończone pętle w normalnej strategii obliczeń Prologu. Cele, które mają być tablicowane są identyfikowane odpowiednim polem własności trójki.

W trybie hybrydowym można włączyć oba tryby maszyny wnioskującej równocześnie i wtedy tryb progresywny może być użyty do informowania trybu regresywnego, który z kolei zajmuje się uzyskaniem odpowiedzi na pytanie. System Jena jest zaimplementowany w Javie.

### 3.4. Systemy programowania w logice

Jak już wspomniano w punkcie 2.5, nasza uwaga koncentruje się na dwóch systemach programowania w logice, a mianowicie są to:

- System SWI-Prolog [SWI] z obiektowym rozszerzeniem XPCE służącym do programowania interfejsu z użytkownikiem, oraz wyposażony w bibliotekę modułów współpracy z siecią semantyczną,
- System XSB [XSB] oparty na tzw. dobrze ufundowanej semantyce programów w logice, stosujący strategię SLG z tablicowaniem pośrednich rezultatów obliczeń, pracę wielowątkową oraz rozbudowane techniki indeksowania danych.

## 4. Podsumowanie

Budowa systemu wnioskującego, działającego na dużej bazie wiedzy, zmusza do wprowadzenia wielu nowych technologii wspierających, począwszy od narzędzi do budowy struktur ontologicznych, a skończywszy na repozytoriach. Jednym z najważniejszych zagadnień jest kwestia operacji na schematach baz danych [Pan05] i transformacji pomiędzy różnymi schematami. Gdyby udało się zrealizować wnioskowanie dla magazynów RDF o dużej skali, byłby to duży postęp w rozwoju systemów wykorzystujących elementy sztucznej inteligencji [RaQ], po wielu latach praktycznych niepowodzeń w dziedzinie zastosowania obiektowych i dedukcyjnych baz danych (bazy RDF zawierają pewne cechy tych rodzajów baz danych). Duża aktywność wielu projektów typu Ingenta oraz aktywność nowych firm wnoszą wiele cennych informacji, które pozwalają na coraz lepsze projektowanie i tworzenie systemów bazodanowych wspieranych semantycznie przez ontologie i metadane RDF. Pojawiają się głosy, że do roku 2009 nowe repozytoria firm globalnych będą masowo budowane w technologii baz danych RDF, stanowiąc istotną konkurencję dla dotychczasowych baz SQL. Niestety, choć prowadzone są zaawansowane prace dotyczące przeniesienia wyników projektu WordNet (leksykon semantyczny), dotyczącego języka angielskiego, na inne języki [WN], brak na razie podobnych prac dla języka polskiego, co będzie stanowić istotne ograniczenie projektu AFIZ.

## Bibliografia

- [MaCy05] Martinek J., Cybulka J., Dynamics of Legal Provisions and its Representation, 10th International Conference on Artificial Intelligence and Law, ICAIL'05, June 6-11, 2005, Bologna Italy, The ACM Press, New York, pp. 20-24
- [CyMa05] Cybulka J., Martinek J., Application Ontology Development - A Case Study, 2nd Language and Technology Conference, L&TC'05, April 21-23, 2005, Poznań, pp.240-244
- [CMP06] Cybulka J., Meissner A., Pankowski T.: Schema- and Ontology-Based XML Data Exchange in Semantic E-Business Applications. 9th International Conference on Business Information Systems, BIS 2006, May 31-June 2, 2006, Klagenfurt Austria, GI-Edition, pp. 429-441
- [OLG+05] Oberle D., Lamparter S., Grimm S., Vrandečić D., Staab S., Gangemi A. Towards Ontologies for Formalizing Modularization and Communication in Large Software Systems, Technical Report, University of Karlsruhe, AIFB, 2005
- [SBW06] Shadbolt N., Berners-Lee T., Hall W., The Semantic Web Revisited, IEEE Intelligent Systems 21(3) pp. 96-101, 2006
- [TR06] Taniar D., Rahayu J.W., (eds), Web semantics and ontology, Idea Group Inc., 2006
- [LaSw] Lassila, O., Swick, R. R., Resource Description Framework (RDF): Model and Syntax Specification, rekomendacja W3C, <http://www.w3.org/TR/REC-rdf-syntax/>
- [SPARQL] SPARQL - Query Language for RDF, [www.w3.org/TR/2004/WD-rdf-sparql-query-20041012/](http://www.w3.org/TR/2004/WD-rdf-sparql-query-20041012/)
- [M06] Melton J., SQL, XQuery, and SPARQL: What's Wrong With This Picture? Materiały konferencji "Building Web 2.0", Amsterdam, Holandia, Maj, 2006.
- [Datalog] <http://en.wikipedia.org/wiki/Datalog>
- [RaGe02] Ramakrishnan R., Gehrke J., Database Management Systems, 3rd Ed., McGraw-Hill, 2002
- [SWI] SWI-Prolog's Semantic Web library, <http://www.w3.org/2001/sw/BestPractices/WNET/wconversion.html#swiprolog#swiprolog>; <http://www.swi-prolog.org/>
- [P06] Prud'hommeaux E., SPASQL: SPARQL Support In MySQL, Materiały konferencji "Building Web 2.0", Amsterdam, Holandia, Maj, 2006.
- [ORACLE] [http://www.oracle.com/technology/tech/semantic\\_technologies/htdocs/what\\_oracle\\_brings.html](http://www.oracle.com/technology/tech/semantic_technologies/htdocs/what_oracle_brings.html)
- [PoPa06] Portwin K., Parvatikar P., Building and Managing a Massive Triple Store: An Experience, Materiały konferencji "Building Web 2.0", Amsterdam, Holandia, Maj, 2006

- [Pan05] Pankowski T, Specifying Schema Mappings for Query Reformulation in Data Integration Systems, W: Advances in Web Intelligence, LNCS 3528, Springer Verlag, 2005, pp. 361-366
- [RaQ] Reasoning-aware Querying, <http://www.pms.ifi.lmu.de/reverse-wgi4/facts>
- [WN] [http://www.globalwordnet.org/gwa/wordnet\\_table.htm](http://www.globalwordnet.org/gwa/wordnet_table.htm)
- [KoUSA] Serwis Kongresu Stanów Zjednoczonych, <http://www.govtrack.us/>
- [PrIn] Projekt Ingenta, <http://www.ingentaconnect.com/>
- [JENA] Jena – A Semantic Web Framework for Java, <http://jena.sourceforge.net/>
- [XSB] The XSB System Version 3.0, <http://xsb.sourceforge.net/>
- [BrGu] Brickley, D., Guha, R. V., Resource Description Framework (RDF) Schema Specification 1.0, <http://www.w3.org/TR/2000/CR-rdf-schema-20000327>
- [OWL] Web Ontology Language (OWL) Guide Version 1.0, <http://www.w3.org/TR/owl-guide>