STUDIA INFORMATICA 2012
Volume 33 Number 2A (105)

Wojciech WALOSZEK
Politechnika Gdańska, Katedra Inżynierii Oprogramowania

S-PELLET JAKO IMPLEMENTACJA ZDANIOWEJ REPREZENTACJI MODUŁÓW ALGEBRY KONGLOMERATÓW[1]

Streszczenie. W niniejszym artykule opisano zdaniową reprezentację konglomeratów (semantycznych modułów ontologicznych). Pokazano, w jaki sposób reprezentacja ta może być przydatna do wnioskowania z konglomeratów. Zaprezentowano też prototypowy system o nazwie S-Pellet, realizujący działania algebry konglomeratów za pomocą reprezentacji zdaniowej. System ten wykorzystuje do wnioskowania program Pellet, realizujący algorytm tableau.

Słowa kluczowe: ontologie, modularyzacja, bazy wiedzy, reprezentacja wiedzy, algebra konglomeratów

IMPLEMENTING SENTENTIAL REPRESENTATION OF S-MODULES IN S-PELLET

Summary. In the article we introduce a prototypical system, named S-Pellet, which reasons over semantic modules with use of a standard reasoning engine, Pellet. The architecture, interfaces and a scenario of use of S-Pellet are presented. For the reasoning S-Pellet exploits the theory of sentential representation of s-modules; the theory is described and discussed in the paper.

Keywords: ontologies, knowledge bases, modularization, s-modules

1. Wprowadzenie

W niniejszym artykule podjęto zagadnienie zdaniowej reprezentacji semantycznych modułów ontologicznych – konglomeratów. Idea konglomeratów oraz towarzysząca im algebra zostały przedstawione we wcześniejszych pracach ([1]; w dalszej części artykułu przypo-

[1] Praca częściowo finansowana z funduszy NCBiR, grant nr SP/I/1/77065/10, projekt SYNAT.

mniane zostaną najważniejsze pojęcia związane z tą ideą). Choć konglomeraty w założeniu reprezentują semantykę modułu ontologicznego (przestrzeń dopuszczalnych modeli), reprezentacja zdaniowa okazuje się być przydatnym narzędziem, m.in. dla dowodzenia rozstrzygalności niektórych rodzajów wnioskowania z użyciem konglomeratów.

W artykule zaprezentowano prototypowy system S-Pellet, służący do wnioskowania z modułów semantycznych (konglomeratów) oraz do przeprowadzania na nich działań algebry konglomeratów. S-Pellet opiera się na założeniach reprezentacji zdaniowej i wykorzystuje do wnioskowania biblioteki udostępnione w ramach znanego systemu Pellet [2].

Wykorzystanie modułu wnioskującego Pellet pociąga za sobą znaczące konsekwencje. Wnioskowanie przeprowadzane jest w pamięci operacyjnej komputera. Rozmiar praktycznych ontologii ograniczony jest do kilku tysięcy osobników. Niemniej jednak głównymi celami utworzenia i utrzymywania prototypu są weryfikacja i rozpowszechnianie metod wykorzystania idei konglomeratów w inżynierii ontologii. Przy tworzeniu systemu S-Pellet kierowano się też następującymi wymaganiami:

- możliwie najpełniejszą pełną obsługa algebry konglomeratów: istniejący system RKASEA ze względu na stosowane w nim pewne ograniczenia metody reprezentacji ([3]) w obecnej wersji nie jest w stanie obsłużyć unii i różnicy konglomeratów;
- brakiem konieczności instalowania dodatkowych systemów: wnioskowanie poza pamięcią operacyjną najczęściej wymaga instalowania osobnego systemu zarządzania bazą danych. Dla systemu RKaSeA jest to Oracle, co stanowi barierę dla rozpowszechniania systemu;
- zapewnieniem możliwości szybkiej aktualizacji prototypu wraz z rozwojem algebry: jednym z zadań prototypu jest ułatwienie weryfikacji nowych idei i pomysłów związanych z algebrą konglomeratów. Utworzenie systemu, dla którego nakład związany z modyfikacjami jest stosunkowo nieduży, ułatwi przeprowadzanie takiego procesu.

Jako że jednym z zadań systemu jest oferowanie inżynierom wiedzy możliwości zapoznania się z algebrą konglomeratów, S-Pellet wyposażony został w dwa interfejsy: programistyczny oraz konsolę użytkownika. Interfejs programistyczny oparto na OWL API [4]. Pozwala on na przeprowadzenie różnych rodzajów wnioskowania z poziomu innych programów napisanych w języku Java. Interfejs konsolowy oferuje użytkownikowi możliwość formułowania zapytań *ad hoc* w uproszczonej (m.in. nie opartej na XML-u) wersji języka KQL [5]. Interfejs konsolowy jest dostępny przez WWW pod adresem: http://knot260.eti.pg.gda.pl:3343/SPelletInterface/.

Dalsza część niniejszego artykułu zorganizowana jest w następujący sposób. Rozdział drugi przypomina najważniejsze pojęcia metody konglomeratowej oraz opisuje zdaniową reprezentację konglomeratów. W rozdziale trzecim opisano system S-Pellet: jego architekturę, interfejsy i scenariusz wykorzystania. Rozdział czwarty zawiera dyskusję na temat powią-

zanych prac. W rozdziale piątym zarysowano dalsze kierunki rozwoju prototypu oraz przedstawiono podsumowanie.

2. Zdaniowa reprezentacja konglomeratów

W niniejszym rozdziale przytoczone są najważniejsze zasady podejścia opartego na konglomeratach wraz z wyjaśnieniem problemów reprezentacji zdaniowej. Tytułem wstępu w podrozdziale 2.1 przedstawiono także główne założenia logiki opisowej.

Podejście konglomeratowe zostało zaproponowane jako alternatywa dla istniejących metod modularyzacji ontologii. Autorom metody przyświecał cel udostępnienia użytkownikowi możliwości znacznie bardziej elastycznego manipulowania zawartością modułów ontologicznych [1] oraz tworzenia modułów „na żądanie", podobnie jak dzieje się to z relacjami w relacyjnej bazie danych. W toku dalszych prac okazało się także, że podejście konglomeratowe wykazuje również znaczny potencjał unifikujący jako metoda opisu innych podejść modularnych.

2.1. Podstawowe pojęcia logiki opisowej

Algebra konglomeratów zdefiniowana jest przy wykorzystaniu założeń płynących z logiki opisowej (lub deskrypcyjnej; ang. *description logics*). Logika opisowa definiuje rodzinę formalizmów (*dialektów*), które różnią się co do ekspresywności, ale wykazują wiele wspólnych cech, m.in. wszystkie stanowią pewne podzbiory logiki pierwszego rzędu, wybrane głównie ze względu na właściwości obliczeniowe algorytmów wnioskujących.

W logice opisowej przyjmuje się uproszczenie polegające na tym, że rezygnuje się z definiowania predykatów dowolnej krotności, w zamian wyróżniając trzy typy nazw: osobników, konceptów i ról. Przyjmuje się przy tym najczęściej, że nazwy te pochodzą z ustalonych zbiorów odpowiednio $\mathbf{N_I}$, $\mathbf{N_C}$ i $\mathbf{N_R}$, składających się na tzw. *sygnaturę pełną* (lub *słownik pełny*) $\mathfrak{S} = \mathbf{N_C} \uplus \mathbf{N_R} \uplus \mathbf{N_I}$. Nazwy te są interpretowane, każda zaś *interpretacja bazowa* $\mathcal{I} = (\Delta^{\mathcal{I}}, \cdot^{\mathcal{I}})$ określa dziedzinę, czyli zbiór $\Delta^{\mathcal{I}}$, i funkcję interpretacyjną $\cdot^{\mathcal{I}}$ przypisującą nazwom z $\mathbf{N_I}$, $\mathbf{N_C}$ i $\mathbf{N_R}$ odpowiednio elementy, podzbiory i zbiory par elementów dziedziny.

Oprócz nazw prostych można także posługiwać się konstrukcjami bardziej złożonymi. Zakres tych konstrukcji określony jest przez wybór dialektu $\mathcal{L}$ logiki opisowej. Przykładowo dialekt $\mathcal{ALC}$ pozwala na wykorzystanie konceptów złożonych postaci $\neg C$, $C \sqcap D$, $C \sqcup D$, $\exists R.C$, $\forall R.C$ (gdzie C i D, również w dalszej części artykułu, oznaczają dowolny prosty lub złożony koncept). Koncepty takie mają ściśle określoną interpretację, którą można wywieść

z interpretacji bazowej, przykładowo $(C \sqcap D)^{\mathcal{I}} = C^{\mathcal{I}} \cap D^{\mathcal{I}}$, a $(\exists R.C)^{\mathcal{I}} = \{e \in \Delta^{\mathcal{I}}: \exists f \in C^{\mathcal{I}}: (e,f) \in R^{\mathcal{I}}\}$.

Sama ontologia w logice opisowej składa się ze zdań. Dopuszczalne zdania to aksjomaty podrzędności konceptów i ról – postaci odpowiednio $C \sqsubseteq D$ i $R \sqsubseteq S$, gdzie R i S to dowolne role – oraz asercje unarne i binarne – postaci odpowiednio $C(a)$ i $R(a, b)$, a i b to dowolne osobniki. Interpretacja $\mathcal{I}$ *spełnia* zdanie α (jest jego modelem, co zapisujemy $\mathcal{I} \models \alpha$), jeśli:

dla $\alpha = C \sqsubseteq D$ zachodzi $C^{\mathcal{I}} \subseteq D^{\mathcal{I}}$,

dla $\alpha = R \sqsubseteq S$ zachodzi $R^{\mathcal{I}} \subseteq S^{\mathcal{I}}$,

dla $\alpha = C(a)$ zachodzi $a^{\mathcal{I}} \in C^{\mathcal{I}}$,

dla $\alpha = R(a, b)$ zachodzi $(a^{\mathcal{I}}, b^{\mathcal{I}}) \subseteq R^{\mathcal{I}}$.

Ontologia $\mathcal{O}$ zapisana w dialekcie $\mathcal{L}$ to pewien zbiór zdań wyrażonych w $\mathcal{L}$ (czyli podzbiór $\mathcal{L}$, jako że dialekt utożsamiamy ze zbiorem wszystkich zdań, które można w nim wyrazić). Interpretacja $\mathcal{I}$ spełnia $\mathcal{O}$ wtedy i tylko wtedy, gdy spełnia każde zdanie z $\mathcal{O}$. Przez sygnaturę ontologii $Sig(\mathcal{O})$ rozumiemy zbiór $\mathbf{S}$ wszystkich nazw prostych użytych w $\mathcal{O}$ (przy czym $\mathbf{S} \subseteq \mathfrak{S}$).

2.2. Podstawowe pojęcia algebry konglomeratów

Centralnym pojęciem dla prezentowanego podejścia jest konglomerat, czyli semantyczny moduł ontologiczny. Jego definicja jest niezależna od języka i ściśle bazuje na pojęciu sygnatury (słownika) i interpretacji.

> **Definicja 1** (konglomerat): *Konglomeratem* M nazywamy parę $(\mathbf{S}, \mathbf{W})$, gdzie: $\mathbf{S}$ jest sygnaturą (skończonym podzbiorem $\mathfrak{S}$), a $\mathbf{W}$ zbiorem interpretacji bazowych. Każdą interpretację z $\mathbf{W}$ nazywamy *modelem konglomeratu* M. Zapisy $\mathbf{S}(M)$ i $\mathbf{W}(M)$ oznaczają odpowiednio sygnaturę i zbiór modeli danego konglomeratu M. ■

Dla konglomeratów zdefiniowano algebrę, na którą składa się zestaw podstawowych działań, zdefiniowanych następująco (M i L to dowolne konglomeraty, $\mathbf{S}$ oznacza dowolną sygnaturę):

$$M \cap L = (\mathbf{S}(M) \cup \mathbf{S}(L), \mathbf{W}(M) \cap \mathbf{W}(L)),$$

$$M \cup L = (\mathbf{S}(M) \cup \mathbf{S}(L), \mathbf{W}(M) \cup \mathbf{W}(L)),$$

$$M - L = (\mathbf{S}(M) \cup \mathbf{S}(L), \mathbf{W}(M) - \mathbf{W}(L)),$$

$$\pi_{\mathbf{S}}(M) = (\mathbf{S}, \mathbf{W}(M)|\mathbf{S}).$$

Operacje przecięcia ($\cap$), unii ($\cup$) i różnicy ($-$) konglomeratów są w istocie typowymi operacjami boolowskimi dla zbioru modeli. Operacja projekcji ($\pi_{\mathbf{S}}$) ogranicza natomiast słownik konglomeratu. W jej definicji zastosowano operację *projekcji interpretacji* do sygna-

tury **S**, przy czym wynikiem projekcji pewnej interpretacji $\mathcal{I}$ jest zbiór interpretacji, przypisujący terminom z **S** dokładnie takie same zakresy, pozostałym zaś terminom dowolny zakres: $\mathcal{I}|\mathbf{S} = \{\mathcal{J}: \Delta^{\mathcal{J}} = \Delta^{\mathcal{I}} \wedge \forall X \in \mathbf{S}: X^{\mathcal{J}} = X^{\mathcal{I}}\}$. Podobnie rozumiemy projekcję konglomeratu: konglomerat ten zachowuje zależności pomiędzy terminami z **S**, ale pozwala na dowolną interpretację terminów spoza **S**.

Dla konglomeratów zdefiniowano także dodatkowe działania, przemianowanie i wchłonięcie, które można włączyć do algebry (γ to dowolna funkcja $\mathcal{P}(\mathfrak{S}) \longrightarrow \mathcal{P}(\mathfrak{S})$, C to dowolny koncept z $\mathbf{S}(M)$):

$$\rho_\gamma(M) = (\gamma(\mathbf{S}(M)), \{\mathcal{J} = (\Delta^{\mathcal{J}}, \cdot^{\mathcal{J}}): \exists \mathcal{I} \in \mathbf{W}(M): \Delta^{\mathcal{I}} = \Delta^{\mathcal{J}} \wedge \forall X \in \mathfrak{S}: \gamma(X)^{\mathcal{J}} = X^{\mathcal{I}}\}),$$

$$L \upsilon_C M = (\mathbf{S}(L) \cup \mathbf{S}(M), \{\mathcal{I} \in \mathbf{W}(M): \mathcal{I} \cap C^{\mathcal{I}} \in \mathbf{W}(L)\}).$$

Przemianowanie (ρ_γ) pozwala na zmianę nazw terminów z sygnatury, operacja wchłonięcia (υ_C) natomiast pozwala na zastosowanie zależności pomiędzy terminami z źródłowego konglomeratu L, ale tylko w podzbiorze (zakresie wskazanego konceptu C) dziedziny docelowego konglomeratu M.

Za pomocą wymienionych, podstawowych operacji można zdefiniować algebrę konglomeratów jako *algebrę cylindryczną* [6]. Algebra cylindryczna ([7]) stanowi rozszerzenie algebry Boole'a o specjalny operator cylindryfikacji, odpowiadający właśnie swoistemu rzutowaniu. Jednak aby dopełnić definicję algebry, trzeba zdefiniować uniwersum konglomeratów, na których będzie mógł operować użytkownik.

Stosunkowo naturalny wybór takiej przestrzeni konglomeratów może przebiegać przez wskazanie pewnego dialektu logiki opisowej. Konglomeraty odpowiadające zdaniom w tym dialekcie mogłyby być uznane za bazowe.

Definicja 2 (bazowa przestrzeń konglomeratów): *Bazową przestrzenią konglomeratów dla dialektu* $\mathcal{L}$ nazywamy zbiór $M(\mathcal{L}) = \{M(\alpha)\}_{\alpha \in \mathcal{L}}$, w którym $M(\alpha) = (Sig(\{\alpha\}), \{\mathcal{I}: \mathcal{I} \models \alpha\})$. ■

Tak zdefiniowaną przestrzeń bazową możemy rozszerzyć przez domknięcie względem wybranych działań algebraicznych.

Definicja 3 (przestrzeń konglomeratów): *Przestrzenią konglomeratów dla dialektu* $\mathcal{L}$ *oraz zbioru działań algebry konglomeratów* A nazywamy zbiór $M_A(\mathcal{L})$ stanowiący domknięcie zbioru $M(\mathcal{L})$ względem działań z A. ■

Przykładowo w przestrzeni $M_{\{\cap\}}(\mathcal{L})$ znajdą się konglomeraty reprezentujące wszystkie ontologie wyrażone w $\mathcal{L}$. W takiej przestrzeni dla danej ontologii $\mathcal{O}$ przez $M(\mathcal{O})$ oznaczamy $\cap_{\alpha \in \mathcal{O}} M(\alpha)$; łatwo pokazać, że $\mathcal{I} \models \mathcal{O} \Leftrightarrow \mathcal{I} \in \mathbf{W}(M(\mathcal{O}))$.

2.3. Reprezentowalność zdaniowa konglomeratów

Ze względu na potencjalnie nieskończoną liczbę modeli konglomeraty nie mogą być reprezentowane bezpośrednio w pamięci komputera. Potrzebne jest zatem opracowanie pewnego sposobu opisu przestrzeni dopuszczalnych modeli konglomeratu. Jeden z możliwych sposobów zaproponowany został w [2]: konglomeraty w systemie RKASEA reprezentowane są za pomocą specjalnych struktur grafowych.

Można jednak zadać istotne pytanie o możliwość zastąpienia konglomeratu z wybranej przestrzeni równoważnym zbiorem zdań, tj. takim, który ma takie same modele. Wyznaczenie takiego zbioru zdań pozwoliłoby na wykorzystanie tradycyjnych mechanizmów wnioskujących do odpowiedzi na pytania dotyczące zawartości konglomeratów.

Problem ten był częściowo rozważany w [1] na przykładzie tzw. konglomeratów nielingwistycznych (tj. niemożliwych do reprezentacji za pomocą zbioru zdań). W niniejszym artykule zostanie wprowadzone bardziej precyzyjne pojęcia reprezentowalności [8].

Definicja 4 ($\mathcal{L}$-reprezentowalność): Konglomerat M jest $\mathcal{L}$-reprezentowalny, jeżeli istnieje zbiór zdań $\mathcal{O} \subseteq \mathcal{L}$, który go $\mathcal{L}$-reprezentuje (co zapisujemy $M \sim_{\mathcal{L}} \mathcal{O}$), czyli taki, że $Sig(\mathcal{O}) \subseteq \mathbf{S}(M)$ oraz $\mathbf{W}(M) = \{\mathcal{I}: \mathcal{I} \models \mathcal{O}\}$. ■

Wszystkie konglomeraty z przestrzeni $M_{\{\cap\}}(\mathcal{L})$ są $\mathcal{L}$-reprezentowalne, co można łatwo dowieść przez indukcję (jeśli $M_1 \sim_{\mathcal{L}} \mathcal{O}_1$ oraz $M_2 \sim_{\mathcal{L}} \mathcal{O}_2$, to $M_1 \cap M_2 \sim_{\mathcal{L}} \mathcal{O}_1 \cup \mathcal{O}_2$). Nie jest to już prawdą dla przestrzeni $M_{\{\cap,\ \cup\}}(\mathcal{L})$, czego dowodzi następujący prosty przykład, zaczerpnięty z [1].

Przykład 1: Niech $M = M(A \sqsubseteq B) \cup M(B \sqsubseteq A)$. Oczywiście M należy do $M_{\{\cap,\ \cup\}}(\mathcal{ALC})$. W ramach sygnatury $\mathbf{S}(M)$ nie istnieje jednak w $\mathcal{ALC}$ zbiór zdań, który miałby dokładnie takie same modele jak M. Zauważmy, że nie jest prawdą, że $\mathbf{W}(M) \models A \sqsubseteq B$, ani że $\mathbf{W}(M) \models B \sqsubseteq A$. Jednakże jeśli wykonamy działanie: $M \cap M(\{A \sqcap \neg B(a),\ \neg A \sqcap B(b)\})$, w jego wyniku otrzymamy konglomerat o pustym zbiorze modeli. Zatem choć konglomerat M nie spełniał żadnego ze zdań $A \sqsubseteq B$, $B \sqsubseteq A$, to wprowadzenie do niego asercji przeczących obydwu tym zdaniom spowoduje sprzeczność. ■

Przykład 1 jest ilustracją tego, że niektóre operacje algebry konglomeratów mogą powodować wykroczenie poza ekspresywność języka wykorzystanego do budowy przestrzeni bazowej. Zdaniem, które reprezentowałoby konglomerat M z przykładu 1, byłaby alternatywa „$(A \sqsubseteq B) \vee (B \sqsubseteq A)$”, której jednak nie można sformułować w $\mathcal{ALC}$ ani w żadnym z popularnych dialektów logiki opisowej.

Ten brak ekspresywności w pewnym stopniu można przezwyciężyć, stosując nieco inny rodzaj reprezentacji zdaniowej: nie za pomocą pojedynczego zbioru zdań, ale za pomocą zbioru zbiorów zdań. Ten rodzaj reprezentacji zdefiniowano w dalszej części.

Definicja 5 ($\mathcal{L}$-(2)-reprezentowalność): Konglomerat M jest $\mathcal{L}$-(2)-reprezentowalny, jeżeli istnieje skończony zbiór zbiorów zdań $\mathcal{S} = \{\mathcal{O}_i\}_{i \in [1..k]}$, $k \in \mathbf{N}$, który go $\mathcal{L}$-(2)-reprezentuje (co zapisujemy jako $M \sim_{\mathcal{L}\text{-}(2)} \mathcal{S}$), czyli taki, że $\bigcup_{i \in [1..k]} Sig(\mathcal{O}_i) \subseteq \mathbf{S}(M)$ oraz $\mathbf{W}(M) = \bigcup_{i \in [1..k]} \{\mathcal{I}: \mathcal{I} \models \mathcal{O}_i\}$. ■

Wszystkie konglomeraty z przestrzeni $M_{\{\cap, \cup\}}(\mathcal{L})$ są $\mathcal{L}$-(2)-reprezentowalne, co można dowieść w następujący sposób:

1. Elementy przestrzeni bazowej $M(\mathcal{L})$ są $\mathcal{L}$-(2)-reprezentowalne, gdyż $\mathrm{M}(\alpha) \sim_{\mathcal{L}\text{-}(2)} \{\{\alpha\}\}$.
2. Jeśli $M_1 \sim_{\mathcal{L}\text{-}(2)} \mathcal{S}_1$ oraz $M_2 \sim_{\mathcal{L}\text{-}(2)} \mathcal{S}_2$, to $M_1 \cup M_2 \sim_{\mathcal{L}\text{-}(2)} \mathcal{S}_1 \cup \mathcal{S}_2$.
3. Jeśli $M_1 \sim_{\mathcal{L}\text{-}(2)} \mathcal{S}_1$, $\mathcal{S}_1 = \{\mathcal{O}_{1:i}\}_{i \in [1..k]}$ oraz $M_2 \sim_{\mathcal{L}\text{-}(2)} \mathcal{S}_2$, $\mathcal{S}_2 = \{\mathcal{O}_{2:j}\}_{j \in [1..m]}$, to $M_1 \cap M_2 \sim_{\mathcal{L}\text{-}(2)} \{\mathcal{O}_{1:i} \cup \mathcal{O}_{2:j}: i \in [1..k], j \in [1..m]\}$.

Podobnie jak w poprzednim przypadku, istnieją jednak ograniczenia, które w ogólności nie pozwalają na znalezienie zbioru zbiorów zdań $\mathcal{L}$-(2)-reprezentującego wynik projekcji. Efekt ten można zilustrować następującym przykładem.

Przykład 2: Niech $M = \pi_{\{A, B\}}(M(\{A \sqsubseteq B, A(a)\}))$. $M \in M_{\{\cap, \cup\}}(\mathcal{ALC})$, jednak nie istnieje zbiór zbiorów zdań, których sygnatura ograniczona byłaby do $\{A, B\}$, a który miałby równoważne modele, gdyż $\mathbf{W}(M)$ zawiera wyłącznie interpretacje, w których zakresy zarówno konceptu A, jak i B są niepuste. ■

Przykład 2 wskazuje na niemożność wyrażenia niektórych zależności bez zastosowania dodatkowych terminów. Głównym problemem, wskazanym w przykładzie 2, powodującym, że $\mathcal{L}$-(2)-reprezentacja nie wystarcza, jest konieczność ograniczenia się do słownika reprezentowanego konglomeratu. Kolejny proponowany rodzaj reprezentacji przełamuje to ograniczenie.

Zastosowanym tutaj pomysłem jest wydzielenie pewnej (nieskończonej) puli nazw $\mathbf{D}$ z pełnej sygnatury $\mathfrak{S}$. Nazwy te używane są wyłącznie jako zamienniki terminów usuwanych z sygnatury przez operację projekcji. Ideę tę sformalizowano w następującej definicji.

Definicja 6 ($\mathcal{L}$-(2)-**D**-reprezentowalność): Konglomerat M jest $\mathcal{L}$-(2)-**D**-reprezentowalny, jeżeli istnieje skończony zbiór zbiorów zdań $\mathcal{S} = \{\mathcal{O}_i\}_{i \in [1..k]}$, $k \in \mathbf{N}$, który go $\mathcal{L}$-(2)-**D**-reprezentuje (co zapisujemy jako $M \sim_{\mathcal{L}\text{-}(2)\text{-}\mathbf{D}} \mathcal{S}$), czyli taki, że $\bigcup_{i \in [1..k]} Sig(\mathcal{O}_i) \subseteq \mathbf{S}(M) \cup \mathbf{D}$ oraz $\mathbf{W}(M) = (\bigcup_{i \in [1..k]} \{\mathcal{I}: \mathcal{I} \models \mathcal{O}_i\})|\mathbf{S}(M)$. ■

Przez zastosowanie podobnych kroków jak w poprzednim dowodzie możliwe jest pokazanie, że wszystkie konglomeraty z przestrzeni $M_{\{\cap, \cup, \pi\}}(\mathcal{L})$ są $\mathcal{L}$-(2)-**D**-reprezentowalne. Pierwsze trzy kroki dowodu są analogiczne, należy go natomiast rozszerzyć o kolejny krok:

4. Jeśli $M \sim_{\mathcal{L}\text{-}(2)\text{-}\mathbf{D}} \mathcal{S}$, $\mathcal{S} = \{\mathcal{O}_i\}_{i \in [1..k]}$, to $\pi_{\mathbf{S}}(M) \sim_{\mathcal{L}\text{-}(2)\text{-}\mathbf{D}} \{\gamma_{\mathbf{S}(M)\text{-}\mathbf{S} \to \mathbf{D}}(\mathcal{O}_i): i \in [1..k]\}$.

Przez funkcję $\gamma_{\mathbf{S}(M)\text{-}\mathbf{S} \to \mathbf{D}}$ rozumiemy specjalny rodzaj funkcji przemianowującej, który zamienia terminy spoza $\mathbf{S}$ na unikalne, jeszcze nie użyte w $\mathcal{S}$ terminy z $\mathbf{D}$, przez zapis $\gamma_{\mathbf{S}(M)\text{-}\mathbf{S} \to \mathbf{D}}(\mathcal{O})$ rozumiemy zaś zbiór zdań, który powstał przez systematyczną zamianę

wszystkich terminów atomowych we wszystkich zdaniach z $\mathcal{O}$ zgodnie z działaniem funkcji $\gamma_{\mathbf{S}(M)\text{ - }\mathbf{S} \to \mathbf{D}}$.

Zbiór reprezentowanych działań można rozszerzyć także o operację przemianowania i (częściowo) wchłonięcia. Ilustrują to następujące kroki.

5. Jeśli $M \sim_{\mathcal{L}\text{-}(2)\text{-}\mathbf{D}} \mathcal{S}$, $\mathcal{S} = \{\mathcal{O}_i\}_{i \in [1..k]}$, to $\rho_\gamma(M) \sim_{\mathcal{L}\text{-}(2)\text{-}\mathbf{D}} \{\gamma(\mathcal{O}_i): i \in [1..k]\}$.
6. Jeśli $M_1 \sim_{\mathcal{L}\text{-}(2)\text{-}\mathbf{D}} \mathcal{S}_1$, $\mathcal{S}_1 = \{\mathcal{O}_{1:i}\}_{i \in [1..k]}$ oraz $M_2 \sim_{\mathcal{L}\text{-}(2)\text{-}\mathbf{D}} \mathcal{S}_2$, $\mathcal{S}_2 = \{\mathcal{O}_{2:j}\}_{j \in [1..m]}$, to $M_1 \upsilon_C M_2 \sim_{\mathcal{L}\text{-}(2)} \{\upsilon_C(\mathcal{O}_{1:i}) \cup \mathcal{O}_{2:j}: i \in [1..k], j \in [1..m]\}$.

W kroku 6 zastosowano specjalną funkcję υ_C, „tłumaczącą” zdania reprezentujące moduł M_1, tak aby ograniczyć ich wpływ tylko do zakresu konceptu C. O ile dla aksjomatów podrzędności konceptów i asercji jest to wykonalne i stosunkowo proste (np. zdanie $D \sqsubseteq E$ zostanie zastąpione przez $C \sqcap D \sqsubseteq C \sqcap E$), o tyle aksjomaty podrzędności i przechodniości ról nie zawsze mogą być w ten sposób przetłumaczone. Dla uproszczenia przyjmujemy więc, że krok 6 wykonujemy jedynie wtedy, gdy $\mathcal{S}_1$ nie zawiera takich aksjomatów.

Opisany algorytm (obejmujący kroki 1-6) pozwala zatem na konstruktywne wyznaczenie $\mathcal{L}$-(2)-reprezentacji każdego konglomeratu zbudowanego z konglomeratów przestrzeni bazowej za pomocą działań $\{\cap, \cup, \pi, \rho\}$ (oraz częściowo υ). Jak już wcześniej wspomniano, jest to kluczowe z punktu widzenia systemu S-Pellet. Specyfika zastosowanego sposobu reprezentacji sprawia, że w sposób stosunkowo prosty i bezpośredni możliwe jest tu zastosowanie standardowego mechanizmu wnioskującego do rozwiązania typowych problemów wnioskowania. Przykładowo, przy założeniu że $M \sim_{\mathcal{L}\text{-}(2)\text{-}\mathbf{D}} \mathcal{S}$, $\mathcal{S} = \{\mathcal{O}_i\}_{i \in [1..k]}$:

- *sprawdzenie spójności konglomeratu M* sprowadza się do sprawdzenia spójności ontologii $\{\mathcal{O}_i\}_{i \in [1..k]}$ i, jeśli którakolwiek z nich jest spójna, konglomerat jest również spójny (tj. ma przynajmniej jeden model);
- *sprawdzenie podrzędności konceptu C względem D* sprowadza się do sprawdzenia, czy dla każdej ontologii $\{\mathcal{O}_i\}_{i \in [1..k]}$ zachodzi $\mathcal{O}_i \models C \sqsubseteq D$. Analogiczny sposób postępowania można zastosować dla wszystkich podobnych problemów, tj. do rozłączności, równoważności i spełnialności konceptów;
- *wyznaczenie wystąpień konceptu C* sprowadza się do wyznaczenia zbiorów wystąpień tego konceptu dla każdej ontologii $\{\mathcal{O}_i\}_{i \in [1..k]}$ i wyznaczenia wspólnej części tych zbiorów. Analogiczny sposób postępowania można zastosować dla wszystkich podobnych problemów, tj. do wyznaczenia typów osobnika *a*, wyznaczenia konceptów atomowych dziedziczących od *C* itp.

Na koniec warto podkreślić, że przy zastosowanej metodzie konglomeraty mogą być przechowywane przez cały czas w postaci zbiorów zdań. Jest to wygodne, jako że pozwala użytkownikowi dostarczać inicjalnych postaci modułów w postaci istniejących ontologii. Przeprowadzenie wnioskowania z modułu wymaga natomiast dodatkowych nakładów.

3. System S-Pellet

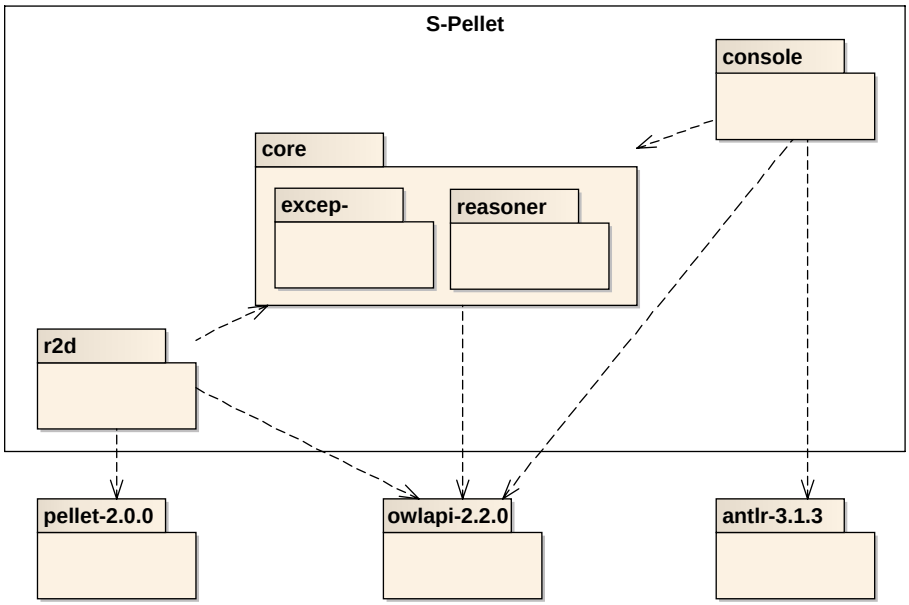
Niniejszy rozdział prezentuje implementacyjne i projektowe aspekty systemu S-Pellet, wykorzystującego opisaną wcześniej zdaniową reprezentację konglomeratów. Kolejne podrozdziały zawierają opis architektury systemu, jego interfejsów, ich implementacji oraz opis scenariusza zastosowania.

3.1. Architektura systemu S-Pellet

System S-Pellet zaimplementowany został w języku Java w wersji 6. Do działania potrzebuje dołączenia standardowego silnika wnioskującego obsługującego interfejs OWL API [4] w wersji przynajmniej 2.2.0. Obecnie w tej roli wykorzystywany jest system Pellet [2] (wersja 2.0.0).

System S-Pellet podzielony jest na kilka podsystemów (rys. 1):

- Podsystem *core* dostarcza definicji interfejsów potrzebnych do opisania konglomeratów oraz wykonywanych na nich operacji. Podsystem ten zawiera dwa dodatkowe moduły: *exceptions* – wprowadzający definicje typowych błędów, które mogą wystąpić podczas przetwarzania konglomeratów – oraz *reasoner* – wprowadzający interfejsy służące do wywołania funkcji implementujących różne rodzaje wnioskowania. Elementy tego podsystemu są dokładniej opisane w podrozdziale 3.2.
- Podsystem *r2d* zawiera klasy implementujące interfejsy zdefiniowane w *core*. Podsystem ten wykorzystuje w swojej pracy algorytm opisany w podrozdziale 2.3, jego elementy zaś opisane są dokładniej w podrozdziale 3.3.



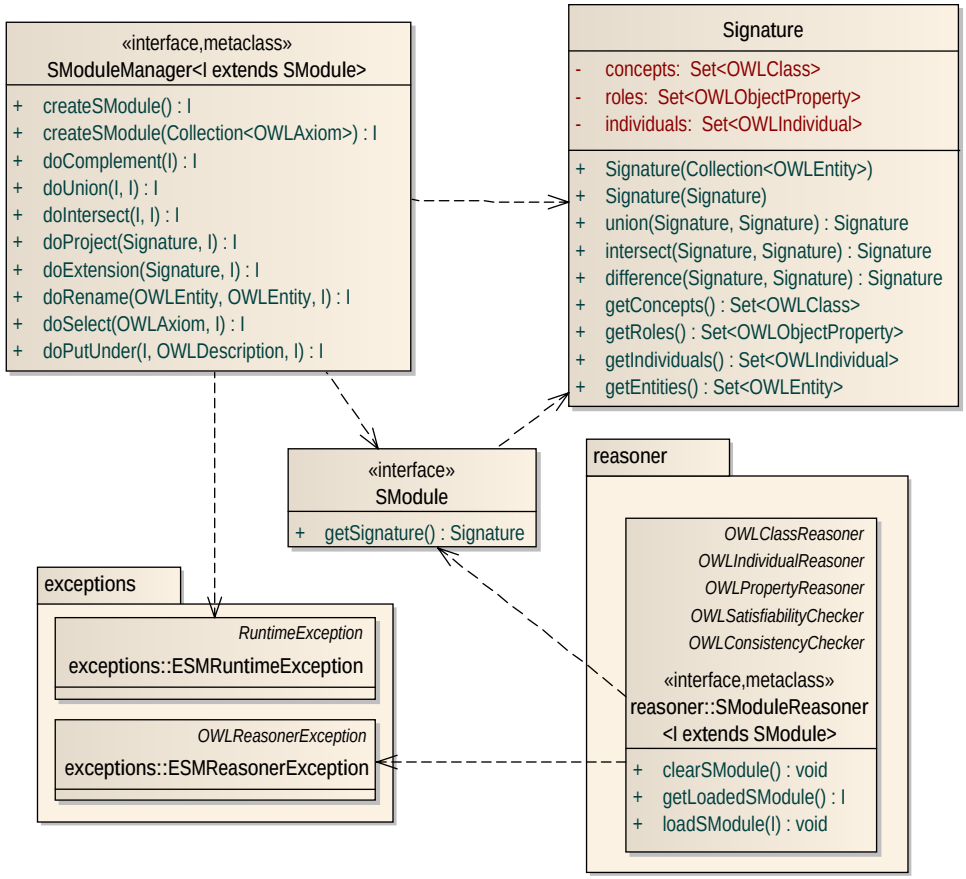
Rys. 1. Architektura systemu S-Pellet
Fig. 1. S-Pellet system architecture

- Podsystem *console* stanowi rodzaj nakładki na podsystem *core* i oferuje możliwość wykonywania (sformułowanych tekstowo) zapytań w uproszczonej odmianie języka KQL. Elementy tego podsystemu są dokładniej opisane w podrozdziale 3.4.

3.2. Interfejs programistyczny systemu S-Pellet

Podsystem *core* definiuje interfejs programistyczny do systemu S-Pellet (i ewentualnie innych, przyszłych systemów wykorzystujących konglomeraty). Interfejs ten opiera się w znacznym stopniu na znanym interfejsie OWL API. Najważniejsze elementy podsystemu przedstawiono na diagramie klas na rys. 2. Wszystkie klasy poprzedzone przedrostkiem OWL pochodzą z OWL API i wykorzystywane są do reprezentowania konceptów, ról i osobników.

Szablon interfejsu *SModuleManager* definiuje metody służące do tworzenia konglomeratu (rodzina metod *createSModule*) oraz wykonywania operacji algebraicznych (metody *doUnion*, *doIntersection*, *doProject* itp.). Zastosowanie szablonu pozwala tu na wykorzystanie jako konglomeratu dowolnej klasy zdefiniowanej przez użytkownika, implementującej interfejs *SModule*, który stanowi najprostszą abstrakcję konglomeratu i daje dostęp do jego sygnatury.



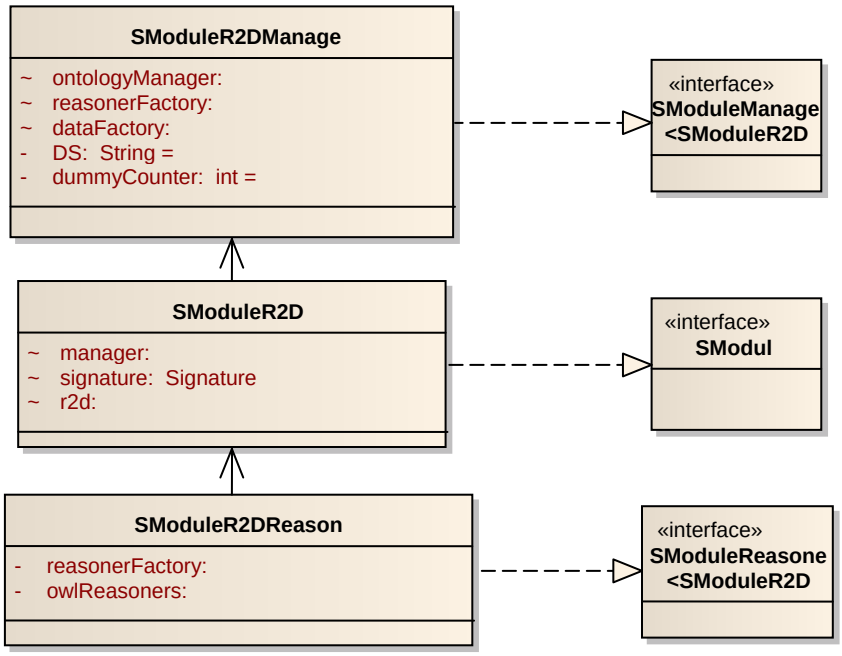
Rys. 2. Główne elementy podsystemu *core*
Fig. 2. Basic elements of *core* subsystem

Podsystem zawiera także klasę *Signature*, która przechowuje słownik konglomeratu (lub ontologii). Klasa ta dostarcza wielu metod pozwalających na wykonywanie podstawowych operacji na sygnaturach, takich jak: sumowanie, odejmowanie i wyznaczanie części wspólnej.

W dodatkowym module *reasoner* zdefiniowano interfejs dla mechanizmu wnioskującego z konglomeratów *SModuleReasoner*. Interfejs ten dziedziczy od wielu interfejsów z OWL API, definiujących metody dla podstawowych problemów wnioskowania. Moduł *exceptions* zawiera natomiast podstawowe definicje błędów, które mogą pojawić się podczas wykonywania operacji algebry konglomeratów lub podczas wnioskowania.

3.3. Realizacja wnioskowania w systemie S-Pellet

Faktyczna realizacja operacji algebraicznych oraz wnioskowania z konglomeratów zachodzi w podsystemie *r2d*. Podsystem ten zawiera zestaw klas realizujących interfejsy zdefiniowane w *core* (rys. 3).



Rys. 3. Główne elementy podsystemu *r2d*
Fig. 3. Basic elements of *r2d* subsystem

Każdy obiekt klasy *SModuleR2D* reprezentuje konglomerat i zawiera zbiór zbiorów zdań, stanowiący jego $\mathcal{L}$-(2)-**D**-reprezentację. Jednak za wyznaczanie poprawnej reprezentacji odpowiedzialne są w istocie obiekty klasy *SModuleR2DManager*. Realizują one operacje algebraiczne, jednocześnie aktualizując reprezentację konglomeratów zgodnie z algorytmem przedstawionym w podrozdziale 2.4. Algorytm ten zaimplementowano według przedstawio-

nych tam założeń. Obiekt *SModuleR2DManager* zachowuje też kontrolę nad pulą nazw pomocniczych **D** (przydzielane są one jako kolejno numerowane koncepty *X_nnn* ze specjalnej przestrzeni nazw *DS*) oraz sprawdza warunki konieczne do przeprowadzenia operacji wchłonięcia.

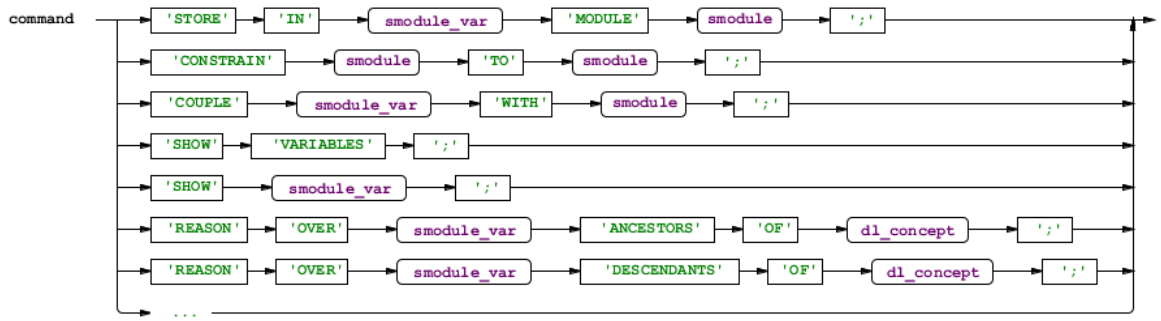
W podsystemie zaimplementowana jest także klasa *SModuleR2DManager*, która odpowiada za przeprowadzanie wnioskowania z konglomeratu. Wnioskowanie realizowane jest zgodnie z naszkicowanym w podrozdziale 2.4 schematem: dla każdego zbioru zdań z $\mathcal{L}$-(2)-**D**-reprezentacji tworzony jest obiekt *OWLReasoner* (za pośrednictwem fabryki *OWLReasonerFactory*, obecnie dostarczanej przez system Pellet), a odpowiedź na podstawowe problemy wnioskowania formułowana jest jako odpowiednia kombinacja odpowiedzi z poszczególnych mechanizmów wnioskujących.

3.4. Interfejs użytkowy systemu S-Pellet

Aby spełnić sformułowane wymaganie, dotyczące umożliwienia zainteresowanym inżynierom zapoznania się z właściwościami algebry konglomeratów, system S-Pellet został wyposażony w specjalną nakładkę na zasadniczy interfejs, umożliwiającą interpretację i przetwarzanie zapytań sformułowanych w uproszczonej, tekstowej wersji języka KQL. Funkcjonalność ta została zaimplementowana w podsystemie *console*.

Do analizy składniowej treści zapytań wykorzystano zewnętrzną bibliotekę *Antlr* w wersji 3.1.3. Biblioteka ta daje możliwości definiowania gramatyk klasy LL(*) [9]. Przez autora biblioteki dostarczane są także dodatkowe narzędzia, pozwalające na generowanie diagramów przedstawiających struktury gramatyczne. Przykładowy diagram opisujący komendę dla systemu S-Pellet przedstawiono na rys. 4. Więcej diagramów stanowiących pełną specyfikację uproszczonego języka KQL zainteresowany czytelnik odnajdzie na stronach http://knot260.eti.pg.gda.pl:3343/SPelletInterface/.

Podsystem *console* zawiera także klasę realizującą prostą funkcjonalność konsoli, do której można kierować różnego rodzaju polecenia i która zapamiętuje stan przetwarzanych konglomeratów. Przez odpowiednie wykorzystanie tej klasy można w prosty sposób tworzyć różne rodzaje interfejsów użytkownika, np. okienkowy lub sieciowy, takie jak wykorzystano na stronach systemu.



Rys. 4. Fragment diagramu składniowego dla elementu *command* gramatyki uproszczonego KQL
Fig. 4. A fragment of syntax diagram for *command* in simplified KQL grammar

3.5. Scenariusz użycia systemu S-Pellet

Ze względu na zaimplementowanie różnych rodzajów interfejsów można łatwo wyobrazić sobie wykorzystanie systemu S-Pellet w różnych zastosowaniach. Z punktu widzenia technicznego może on być wykorzystany jako dołączana biblioteka lub jako serwer sieciowy, wyposażony w interfejs do wykonywania zapytań.

Popularyzatorska rola systemu polega na oferowaniu ogólnodostępnej konsoli do wykonywania różnego rodzaju działań związanych z przetwarzaniem konglomeratów. Za jej pomocą można m.in. prześledzić działanie przykładów zastosowania algebry z [1].

W niniejszym podrozdziale krótko opisano interakcje między użytkownikiem a systemem (przez interfejs konsolowy) podczas realizacji przykładu szukania mordercy z [1]:

1. W pierwszym kroku w systemie należy utworzyć konglomerat zawierający informacje ogólne, mówiące o tym, że istnieje tylko jeden morderca, że jest to osoba, którą wskazuje zaufany świadek, i że zaufany świadek musiał być obecny na miejscu zbrodni:

```
STORE IN m1 MODULE M(
    {victim} ISSUB ATMOST 1 INV murdered,
    EXISTS INV accuses.TrustedWitness ISSUB EXISTS murdered.{victim},
    TrustedWitness ISSUB EXISTS presentAt.CrimeScene );
```

2. Następnie należy utworzyć dwa moduły reprezentujące wersje zebrane przez detektywów (pierwszą, że *johnShady* jest zaufanym świadkiem i oskarża *tedaInnocent* i drugą, mówiącą, że *henryBrillant* jest zaufanym świadkiem wskazującym *markaGuilty*):

```
STORE IN m2 MODULE m1 INTERSECT M(
    TrustedWitness( johnShady ),
    accuses( johnShady, tedInnocent ) );
STORE IN m3 MODULE m1 INTERSECT M(
    TrustedWitness( henryBrillant ),
    accuses( henryBrillant, markGuilty ) );
```

3. Zakładamy, że jedna z wersji zebranych przez detektywów jest prawdziwa, nie wiemy jednak która. Wykonujemy zatem unię utworzonych w pkt. 2 konglomeratów:

```
STORE IN m4 MODULE m2 UNION m3;
```

4. Z utworzonego konglomeratu nie wynika, kto jest mordercą czy też zaufanym świadkiem, o czym można się przekonać, przeprowadzając różne rodzaje wnioskowania:

```
REASON OVER m4 IS tedInnocent RELATED WITH victim BY murdered;
REASON OVER m4 IS markGuilty RELATED WITH victim BY murdered;
REASON OVER m4 IS johnShady OF TYPE TrustedWitness;
REASON OVER m4 IS henryBrillant OF TYPE TrustedWitness;
```

5. Możemy teraz wzbogacić konglomerat o nowo zdobytą, niepodważalną informację na temat morderstwa, mówiącą, że *johnShady* nie był obecny na miejscu zbrodni:

```
STORE IN m5 MODULE SELECT m4 WHERE [NOT EXISTS presentAt.CrimeScene(johnShady)];
```

6. Z nowego konglomeratu możemy już dowiedzieć się, kto jest mordercą:

```
REASON OVER m5 IS johnShady OF TYPE NOT TrustedWitness;
REASON OVER m5 IS markGuilty RELATED WITH victim BY murdered;
```

4. Powiązane prace

S-Pellet jest narzędziem wnioskującym dla ontologii modularnej, zapisanej za pomocą konglomeratów. Ze względu na fakt, że wykorzystuje on nową metodę reprezentowania wiedzy, jedynym systemem, z którym można porównać go bezpośrednio, jest RKASEA, opisana m.in. w [3]. W stosunku do RKasei S-Pellet odznacza się nieco inną filozofią działania. Jak zaznaczono w rozdziale 1, S-Pellet jest systemem znacznie mniej skomplikowanym, niemającym w zamyśle spełniać zadań pełnego systemu zarządzania bazami wiedzy. W zamian oferuje on nieco szersze niż RKASEA pokrycie operacji algebry konglomeratów i jest znacznie prostszy w instalacji.

Nie istnieje zbyt wiele uznanych systemów zapewniających możliwość wnioskowania z ontologii modularnych (bardzo ważnym powodem tego stanu jest zapewne fakt, że żadna z metod modularyzacji nie została przyjęta jako standard). Jednym z najlepiej rozwiniętych pod tym względem mechanizmów wnioskujących jest KaOn [10]. Jego możliwości przetwarzania ontologii modularnych zostały znacznie wzbogacone w ramach projektu NeOn [11]. Opracowane w ten sposób rozszerzenia pozwalają również na wykorzystanie pewnego zestawu działań algebraicznych do operowania na modułach, jednakże są to operacje ściśle syntaktyczne. KaOn wraz z rozszerzeniami dostępny jest jako część systemu zarządzania ontologiami NeOn.

Jeden z nowszych systemów stosujących podejście modularne (kontekstowe) to CKR [12]. Zastosowano tam autorską metodę definiowania ontologii kontekstowych, przy czym zależności między kontekstami opisywane są także w postaci ontologii. System ten nie oferuje jednak możliwości algebraicznego operowania zawartością modułów.

Należy także wspomnieć o tym, że sam Pellet [2] oferuje możliwości przetwarzania ontologii modularnych. Wcześniejsze wersje narzędzia dawały możliwość łączenia modułów ontologicznych według metody $\mathcal{E}$-Connections [13]. Ta możliwość nie występuje już jednak w nowszych wersjach narzędzia. Także dłużej nie jest już wspierany system Drago [14], który pozwala na odwzorowywanie terminologii modułów ontologicznych za pomocą DDL (rozproszonej logiki opisowej; ang. *Distributed Description Logics*) [15]. W dostępnej wersji system oferuje jedynie możliwość odwzorowywania terminologii i nie pozwala na przeprowadzanie algebraicznych operacji na modułach.

5. Kierunki rozwoju i podsumowanie

W artykule przedstawiono koncepcję i konstrukcję systemu S-Pellet. System ten służy do wnioskowania z semantycznych modułów ontologicznych – konglomeratów – i do przeprowadzania na nich operacji algebraicznych. Wykorzystuje przy tym specjalny rodzaj reprezentacji zdaniowej, co pozwala na użycie tradycyjnych mechanizmów wnioskowania.

Dalsze możliwości rozwoju systemu obejmują prace nad zdaniową reprezentacją większej liczby operacji, przede wszystkim różnicy konglomeratów. Pewne wstępne działania w tej kwestii są już wdrażane w życie. Ciekawym pomysłem wydaje się także rozbudowywanie uproszczonej wersji języka KQL w kontakcie z jego potencjalnymi użytkownikami, co pozwoli na usprawnienie rozwoju pełnej wersji języka.

Jak wspomniano we wprowadzeniu, system służy również jako prototyp do testowania nowych elementów podejścia konglomeratowego. To zadanie jest szczególnie ważne w kontekście działań prowadzonych przez grupę KMG w ramach projektu SYNAT (System Nauki i Techniki; http://www.synat.pl). Projekt SYNAT ma za zadanie utworzenie systemu zarządzania wiedzą, wspierającego gromadzenie zasobów ułatwiających pracę naukowców. Konglomeraty wykorzystywane są w nurcie badawczym tego projektu, związanym z wyszukiwaniem i usprawnianiem dostępu do potrzebnych publikacji elektronicznych. Choć docelowo w projekcie planowane jest wykorzystanie RKasei, użycie systemu S-Pellet pozwala na przetestowanie wielu pomysłów i przeprowadzenie ich pilotażu stosunkowo niedużym kosztem.

BIBLIOGRAFIA

1. Goczyła K., Waloszek A., Waloszek W.: Algebra konglomeratów jako narzędzie opisu problemów przetwarzania ontologii. Studia Informatica, Vol. 30, No. 2A(83), Wydawnictwo Politechniki Śląskiej, Gliwice 2009, s. 141÷156.

2. Sirin E., Parsia B., Grau B. C., Kalyanpur A., Katz Y.: Pellet: A Practical OWL-DL Reasoner. Journal of Web Semantics, Vol. 5(2), 2007, s. 51÷53.
3. Goczyła K., Waloszek A., Waloszek W., Zawadzka T.: Wewnętrzna reprezentacja konglomeratowej bazy wiedzy w systemie RKaSeA. Studia Informatica, Vol. 31, No. 2A(89), Wydawnictwo Politechniki Śląskiej, Gliwice 2010, s. 105÷120.
4. Horridge M., Bechhofer S.: The OWL API: A Java API for Working with OWL 2 Ontologies. OWLED 2009, 6th OWL Experienced and Directions Workshop, 2009.
5. Goczyła K., Piotrowski P., Waloszek A., Waloszek W., Zawadzka T.: Język KQL jako realizacja idei języka SQL dla bazy wiedzy. Studia Informatica, Vol. 31, No.2A(89), Wydawnictwo Politechniki Śląskiej, Gliwice 2010, s. 47÷61.
6. Goczyła K., Waloszek A., Waloszek W.: Algebra of ontology modules for semantic agent. Computational collective intelligence: semantic web, social networks and multiagent systems. LNCS, Vol. 5796, Springer, 2009, s. 492÷503.
7. Henkin L., Monk J. D., Tarski A.: Cylindric Algebras pt. 1. Studies in Logic and the Foundations of Mathematics, Vol. 64, North-Holland, Amsterdam 1971.
8. Goczyła K., Waloszek A., Waloszek W.: A Semantic Algebra for Modularized Description Logics Knowledge Bases. Proc. of DL Workshop 2009, CEUR-WS, Vol. 477, 2009.
9. Parr T.: The Definitive ANTLR Reference: Building Domain-Specific Languages. Pragmatic Bookshelf, 2007.
10. Volz R., Oberle D., Staab S., Motik B.: KAON SERVER – A Semantic Web Management System. AT Proc. of the 12th World Wide Web Conference, ACM, Budapeszt 2003.
11. Gomez-Perez A., Suarez-Figueroa M. C.: Scenarios for Building Ontology Networks within the NeOn Methodology. Proc. of the K-CAP 2009, ACM, 2009.
12. Homola M., Tamilin A., Serafini L.: Modeling Contextualized Knowledge. Procs. of the 2nd Workshop on Context, Information and Ontologies, CEUR-WS, Vol. 626, 2010.
13. Grau B. C., Parsia B., Sirin E.: Working with Multiple Ontologies on the Semantic Web. The Semantic Web – ISWC 2004, LNCS, Vol. 3298, Springer, Berlin 2004, s. 620÷634.
14. Serafini L., Tamilin A.: DRAGO: Distributed Reasoning Architecture for the Semantic Web. Proc. of the ESWC'05, LNCS, Vol. 3532, Springer-Verlag, 2005.
15. Borgida A., Serafini L.: Distributed Description Logics: Assimilating Information from Peer Sources. Journal of Data Semantics, No. 1, 2003, s. 153÷184.

Wpłynęło do Redakcji 14 stycznia 2012 r.

Abstract

In the paper we present a prototypical reasoning system S-Pellet. S-Pellet reasons over modular knowledge bases organized in accordance with the s-modular approach [1]. In the approach all the modules are treated strictly semantically, as sets of valid Tarski-style models.

The S-Pellet system allows for performing wide range of algebraic operations on the modules. In order to perform this task it exploits a special kind of sentential (or syntactic) representation for s-modules. The paper introduces the reader to the theory of sentential representability of semantic modules and proves that construction of proper representation can be accomplished for a wide range of modules in finite time by executing relatively simple steps.

Sentential representation enables us to use standard reasoning engines for performing inference tasks. In S-Pellet system Pellet has been utilized in that role [2]. The range of offered inferences embraces answering all the queries specified in OWL API interface [4].

The S-Pellet system is equipped with several interfaces, which eases the process of integrating it into larger systems. One of the interfaces is API (based on OWL API) for s-modular knowledge bases and performing operations on s-modules. S-Pellet also includes a simple console which accepts commands in simplified KQL language [5]. The presence of the console makes the system self-contained and ready to use by any user.

A web version of the console interface is available at: http://knot260.eti.pg.gda.pl:3343/SPelletInterface/. A scenario of use of the console is presented in the paper, but it is also available (along with the specification of the grammar of simplified KQL language) at the additional pages at the above given address.

Adres

Wojciech WALOSZEK: Politechnika Gdańska, Wydział Elektroniki, Telekomunikacji i Informatyki, ul. Narutowicza 11/13, 80-233 Gdańsk, Polska, wowal@eti.pg.gda.pl.