Grzegorz J. Nalepa

Sztuczne systemy oparte na wiedzy

1. Wstęp

Sztuczna inteligencja jest dziedziną obfitującą w rozmaite koncepcje naśladowania inteligentnych aspektów działania lub rozumowania, które najczęściej przybierają postać różnorodnych modeli matematycznych. W niniejszym rozdziale będą nas interesowały przede wszystkim modele oparte na reprezentacjach symbolicznych, wykorzystujące głównie wybrane rachunki logiczne. Podejście takie wpisuje się w długą tradycję klasyfikowania modeli sztucznej inteligencji jako modeli symbolicznych i modeli inteligencji obliczeniowej. W tych drugich wykorzystywane są przede wszystkim reprezentacje numeryczne, czego najlepszy przykład mogą stanowić sieci neuronowe i zbiory rozmyte.

2. Symboliczne modele reprezentacji wiedzy

Reprezentacje wiedzy i mechanizmy wnioskowania

W centrum zainteresowania symbolicznej sztucznej inteligencji znajdują się metody reprezentacji wiedzy (ang. *knowledge representation*). Przez reprezentację wiedzy rozumie się najczęściej pewien symboliczny model pozwalający na jej uporządkowanie, przechowanie i przetwarzanie w sztucznym systemie inteligentnym. Z reprezentacjami nierozerwalnie związane są mechanizmy wnioskowania (ang. *reasoning*). W pewnym uproszczeniu można powiedzieć, że o ile same metody reprezentacji dostarczają języka budowy modeli (często mających postać tzw. bazy wiedzy – ang. *knowledge base*) na potrzeby systemów inteligentnych,

o tyle metody wnioskowania określają, do czego można tych modeli użyć[1], jakie problemy da się z ich pomocą rozwiązać, czy też na jakie pytania „odpowie" sztuczny system inteligentny. Rzecz jasna konkretne metody wnioskowania są ściśle powiązane z odpowiednimi metodami reprezentacji wiedzy.

ztuczny język jako metoda reprezentacji wiedzy

Traktowanie metody reprezentacji wiedzy jako sztucznego sformalizowanego języka wskazuje na szereg typowych kwestii, które należy określić podczas jego projektowania. Zaliczają się do nich między innymi określenie słownika i zdefiniowanie składni. Niezwykle istotne jest również przedstawienie właściwej semantyki. Opisanie metod reprezentacji wiedzy w sposób precyzyjny i sformalizowany możliwe jest wskutek oparcia ich na logice matematycznej. Dzięki temu można stwierdzić, co – tzn. jaką grupę problemów – da się opisać za pomocą takiego języka.

Procesy inteligentne jako procesy obliczeniowe

Jakkolwiek większość badaczy z obszaru sztucznej inteligencji nie nawiązuje wprost do tej kwestii, można przyjąć, że u podstaw takiego podejścia leży założenie, iż inteligentne procesy myślowe w sztucznym systemie inteligentnym da się efektywnie modelować za pomocą pewnego maszynowego procesu wnioskowania, który z kolei może być przełożony na proces *stricte* obliczeniowy. Należy jednak zaznaczyć, że jest to założenie silnie redukcjonistyczne.

yrazu języka użyteczność

Z praktycznego punktu widzenia siła wyrazu metod reprezentacji wiedzy odgrywa istotną rolę. Możliwość tworzenia bogatych (w sensie znaczeniowym) opisów rodzi jednak problem wiążący się z ich interpretacją i użyciem, czyli wnioskowaniem. Daje się tu zaobserwować typową zależność, gdy wraz ze wzrostem siły wyrazu języka narastają problemy z efektywnym wnioskowaniem i interpretowaniem utworzonych w nim opisów. W przypadku języka zbyt złożonego najczęściej nie jest możliwe skonstruowanie algorytmów cechujących się akceptowalną złożonością obliczeniową (ang. *tractable*)[2].

ia inżynierii wiedzy

W obrębie sztucznej inteligencji inżynieria wiedzy zajmuje się przede wszystkim modelowaniem i konstruowaniem systemów inteligentnych opartych na reprezentacjach symbolicznych. Typowe procesy inżynierii wiedzy to między innymi: pozyskiwanie (ang. *acquisition*) wiedzy,

1 Omawiane tu metody reprezentacji opierają się na podejściu deklaratywnym, a co za tym idzie – odpowiedniejsze byłoby sformułowanie „jak ich używać w celu rozwiązania konkretnych zadań".

2 W praktyce nie jest możliwe bezpośrednie użycie języka naturalnego, którym posługuje się człowiek, gdyż jest to język zbyt niejednoznaczny. Co za tym idzie, konieczne jest zastosowanie języków sformalizowanych, a przy tym bardziej

modelowanie baz wiedzy i analiza wiedzy (niejednokrotnie wyróżnia się weryfikację, walidację i ewaluację). Podstawowym zadaniem inżynierii wiedzy jest dostarczanie efektywnych metod budowy symbolicznych systemów inteligentnych.

W kolejnych częściach tego rozdziału zostaną wskazane podstawowe rachunki logiczne wykorzystywane do formalizowania metod reprezentacji. Jedną z klasycznych metod reprezentacji jest programowanie w logice, czyli bezpośrednie wykorzystanie podzbioru rachunku predykatów jako efektywnego języka opisu wiedzy symbolicznej. Podejście to można również z powodzeniem wykorzystać do rozwiązywania problemów opisywanych za pomocą ograniczeń. Jedną z podstawowych metod opisu wiedzy są z kolei reguły decyzyjne, wykorzystywane między innymi do budowy systemów ekspertowych. Wczesną metodą reprezentacji były również ramy (szablony semantyczne), w informatyce stanowiące antycypację metod obiektowych. Kolejną istotną grupą metod są sieci semantyczne i oparte na nich ontologie formalne. Na zakończenie rozdziału wskazane zostaną wybrane metody uczenia maszynowego wykorzystywane do reprezentacji symbolicznych.

3. Rachunki logiczne

Klasyczny rachunek zdań

Klasyczny rachunek zdań jest jednym z najprostszych formalizmów używanych do opisu sztucznych systemów inteligentnych. Udostępnia język o prostej składni, obejmującej symbole używane do oznaczenia zmiennych zdaniowych (np. p, q, r), które za pomocą spójników logicznych mogą być łączone w wyrażenia. Zdanie w sensie logicznym najczęściej powinno mieć postać prostego stwierdzenia, na przykład „pada deszcz". Podstawowe spójniki logiczne to koniunkcja ($\wedge$), alternatywa ($\vee$) oraz negacja ($\neg$). Za pomocą alternatywy i negacji definiuje się również implikację ($\rightarrow$) materialną („$\neg p \vee q$"), natomiast prawdziwość lub fałszywość każdej formuły rachunku zdań możemy określić za pomocą funkcji interpretacji.

Reguły wnioskowania

Wnioskowanie w rachunku zdań jest opisywane przy użyciu szeregu reguł (schematów) wnioskowania. Ich przykładami są *modus ponens*: „1) $p \rightarrow q$, 2) p, 3) q", czy też *modus tollens*: „1) $p \rightarrow q$, 2) $\neg q$, 3) $\neg p$". Odczytujemy je jako: „jeżeli z p wynika q, oraz p, wtedy q" i analogicznie: „jeżeli z p wynika q, oraz nie q, wtedy nie p". Używając reguł wnioskowania, można przeprowadzać dowodzenie twierdzeń opisywanych za

Prostota składni i semantyki oraz metod wnioskowania sprawiły, że rachunek zdań jest częstokroć wykorzystywany jako narzędzie logiczne opisu prostych systemów wnioskujących. Niemniej jako język opisu jest to rozwiązanie o ograniczonej sile wyrazu. Dlatego wykorzystuje się rachunek predykatów pierwszego rzędu mający większą siłę wyrazu.

Rachunek predykatów wszego rzędu

W logice pierwszego rzędu język jest wzbogacony o symbole funkcyjne, pozwalające między innymi na strukturalizację baz wiedzy, na przykład „książka(autor, tytuł)", i o symbole relacyjne, czyli predykaty, na przykład „Kolor(zielony)" czy „Nauczyciel(Kowalski, Nowak)". Ponadto użycie zmiennych logicznych i kwantyfikatorów (uniwersalnego i szczegółowego) pozwala na zapis złożonych relacji o charakterze warunków czy reguł wnioskowania.

Duża siła wyrazu rachunku predykatów pierwszego rzędu sprawia, że jest on wykorzystywany do opisu systemów inteligentnych. Jednakże konsekwencją takiej siły wyrazu są między innymi problemy z wnioskowaniem, w tym jego nierozstrzygalność (w ogólnym przypadku), tzn. możliwość skonstruowania formuł (wyrażeń poprawnych składniowo), których spełnialności nie można efektywnie zweryfikować. Aby uporać się z tym problemem, stosuje się specjalnie skonstruowane podzbiory rachunku predykatów, zakładające szereg ograniczeń składniowych. Przykładowo w programowaniu w logice ogranicza się składnię dopuszczalnych formuł do tzw. klauzul Horna[3], a w logikach opisowych (ang. *description logics*) ogranicza się liczbę argumentów predykatów do maksymalnie dwóch.

Ograniczenia składniowe

Poza rachunkiem zdań i predykatów niejednokrotnie stosuje się rachunki logiczne właściwiej uchwytujące intuicje potrzebne do modelowania wiedzy o budowie systemu, czy też lepiej oddające charakter jego działania. Przykładem takiego rachunku jest logika modalna, wprowadzająca operatory konieczności i możliwości, której semantyka wykorzystuje notację światów możliwych (ang. *possible world semantics*) wprowadzoną przez Saula Kripkego. W systemach inteligentnych rachunki logiczne są stosowane między innymi do modelowania przekonań agentów. Na potrzeby systemów współbieżnych, w tym wieloagentowych, zaproponowano wyspecjalizowane logiki modalne modelujące ich współpracę, na przykład ATL (ang. *Alternating-time Temporal Logic*). Z kolei do modelowania pojęć nieostrych (lub niejednoznacznych) użyteczne okazały się między innymi logika rozmyta (ang. *fuzzy logic*) oraz teoria zbiorów przybliżonych (ang. *rough sets*).

ika modalna

ika rozmyta i zbiory przybliżone

3 Czyli formuł w postaci koniunkcji, w której co najmniej jeden literał nie jest zanegowany.

4. Programowanie w logice

Koncepcja programowania w logice zakłada bezpośrednie użycie formuł logicznych rachunku predykatów pierwszego rzędu, ograniczonych do postaci klauzul Horna, jako metody reprezentacji wiedzy dla systemu inteligentnego. Ze względu na konieczność zapewnienia efektywnych algorytmów wnioskowania opis składa się wyłącznie ze wspominanych już klauzul Horna. Są to formuły w postaci:

Klauzule Horna

$$p \vee p \vee p \vee p \vee \neg h$$

Co za tym idzie, mogą być one traktowane jako równoważne implikacji:

$$p \wedge p \wedge p \wedge p \rightarrow h$$

Podstawowe zadanie wnioskowania to sprawdzenie spełnialności celu (ang. *goal*), co oznacza przede wszystkim konieczność ustalenia wartości zmiennych logicznych występujących w formule opisującej cel. Na znalezienie przez system stosownego rozwiązania można spojrzeć jako na szczególny przypadek automatycznego dowodzenia spełnialności formuły logicznej.

Najczęściej wykorzystywaną realizacją tej koncepcji jest język Prolog[4]. W praktyce system prologowy obejmuje kompilator języka oraz powłokę pozwalającą na uruchamianie mechanizmu wnioskującego. Jako technikę dowodzenia stosuje się w nim rezolucję SLD (ang. *Selective Linear Definite clause resolution*) wraz z mechanizmem nawrotów przy poszukiwaniu rozwiązań. System Prolog przeszukuje przestrzeń rozwiązań za pomocą techniki przeszukiwania w głąb. Znajduje jedno poprawne rozwiązanie, a jeżeli konieczne jest odszukanie wszystkich, można to osiągnąć przez wymuszenie nawrotu i kontynuację poszukiwań. Sam język programowania służy do opisania bazy wiedzy systemu i określenia zadania wnioskowania – celu.

Język Prolog

Baza wiedzy systemu składa się z dwóch rodzajów klauzul: faktów (klauzule proste) i reguł (klauzule złożone). Prolog wprowadza przejrzystą notację tekstową dla programów w logice. Analogiczne do zdań funkcjonujących w języku naturalnym klauzule kończy się kropką. Fakty odpowiadają stwierdzeniom, zdaniom oznajmującym, reguły zaś opisują pewne prawidłowości czy relacje zachodzące pomiędzy symbolami

4 Prolog jest uniwersalnym językiem programowania opartym na koncepcji programowania w logice.

oznaczonymi za pomocą zmiennych logicznych. Nazwy zmiennych logicznych (niewiadomych) pisane są wielką literą, podczas gdy nazwy stałych, podobnie jak i predykatów, zaczynają się od małej litery. Konwencję tę można prześledzić na poniższym przykładzie bazy wiedzy:

```
doktorant(kasia).
doktorant(jarek).
doktorant(olek).
srednia(kasia,4.6).
srednia(jarek,4.4).
srednia(olek,4.7).
kobieta(kasia).
mezczyzna(jarek).
mezczyzna(olek).
mezczyzna(jozef).
promotor(jozef,kasia).
promotor(jozef,adrian).
promotor(stefan,olek).
promotor(piotr,jarek).
student(adrian).
student(S) :- doktorant(S).
dba(P,S) :- promotor(P,S),student(S).
stypendium(Student) :- promotor(_,Student),srednia(Student,S),S > 4.5.
```

Powyższa baza wiedzy składa się z 18 klauzul, w tym 15 faktów i 3 reguł. Opisują one 8 predykatów.

Dla powyższej bazy wiedzy można podać przykładowe pytania (cele), na które system wnioskujący Prologu automatycznie znajdzie odpowiedzi:

```
? - student(S).

S = adrian ;
S = kasia ;
S = jarek ;
S = olek.

? - stypendium(Kto).

Kto = kasia ;
Kto = olek ;

? - dba(P,S).
P = jozef,
S = kasia ;
```



```
P = jozef,
S = adrian ;

P = stefan, S = olek ;

P = piotr,
S = jarek.
```

Mocne i słabe strony Prologu

Jako narzędzie programistyczne Prolog wprowadza szereg przydatnych rozwiązań technicznych na poziomie języka, na przykład obsługę list (czyli struktur składających się z dowolnej liczby elementów), oraz na poziomie wyszukiwania rozwiązań czy sterowania wnioskowaniem, na przykład mechanizm odcięcia (ang. *cut*).

Do niewątpliwych zalet Prologu należą jego prostota, siła wyrazu i uniwersalność. Narzędzie to ma jednak swoje ograniczenia, spośród których należy wymienić możliwości podstawowego mechanizmu wyszukiwania rozwiązań (choć jego modyfikacja jest stosunkowo prosta) oraz sam język opisu, nie zawsze pozwalający na efektywne i jednoznaczne zdefiniowanie analizowanego problemu. Nie zmienia to faktu, że jako język opisu i prototypowania systemów inteligentnych Prolog cieszy się popularnością, szczególnie w Europie, gdzie powstał. Przykładem uniwersalnego i rozpowszechnionego w środowisku akademickim systemu prologowego jest SWI Prolog[5]. W USA bardzo popularny jest natomiast język LISP.

Mechanizmy opisu i rozwiązywania problemów rozważane w paradygmacie programowania w logice niejednokrotnie są wykorzystywane w innych językach programowania wysokiego poziomu. Ponadto wiele koncepcji stało się inspiracją do tworzenia innych metod symbolicznego opisu problemów dla systemów inteligentnych, w tym dla programowania z ograniczeniami, systemów regułowych czy ontologii formalnych.

5. Programowanie z ograniczeniami

Modelowanie symboliczne problemów za pomocą reguł określających ograniczenie możliwych rozwiązań jest intuicyjne i często wykorzystywane w praktyce. Dysponując takim modelem, można użyć specjalizowanych mechanizmów wnioskowania do automatycznego wyszukania dopuszczalnych rozwiązań. Mogą to być zarówno mechanizmy związane

5 Zob. <www.swi-prolog.org> [dostęp: 05.08.2015].

z konkretnym obszarem zastosowań, jak i te mające charakter uniwersalny. Co za tym idzie, w podejściu opierającym się na CSP (ang. *Constraint Satisfaction Problems*) wyróżnia się fazy modelowania i automatycznego odnajdywania rozwiązań.

Modelowanie problemów graniczeniami

W fazie modelowania problem z ograniczeniami (CSP) opisuje się, wykorzystując zmienne, z których każda może przyjąć wartość z określonej dziedziny. Przy ich pomocy formułuje się ograniczenia, najczęściej mające charakter równań lub wykluczeń. Przykłady takich modeli można odnaleźć w szeregu łamigłówek, na przykład w popularnym sudoku. Proste modele CSP występują również w tzw. łamigłówkach kryptoarytmetycznych, na przykład:

```
  TWO
+ TWO
--------
= FOUR
```

W powyższym przypadku występuje sześć zmiennych odpowiadających literom T, W, O, F, U, R, oraz trzy pomocnicze odpowiadające przeniesieniu w dodawaniu, na przykład X1, X2 i X3. Każda z tych zmiennych może przyjmować wartość z dziedziny cyfr arabskich 0-9. Można też sformułować pięć następujących ograniczeń:

1) O+O=R+10*X1, 2) X1+W+W=U+10*X2, 3) X2+T+T=O+10*X3, 4) X3=F, 5) alldifferent(T, W, O, F, U, R).

Sens ostatniego z nich jest taki, że każda ze zmiennych musi mieć inną wartość. Przykładem rozwiązania tego problemu jest następujący zbiór wartości zmiennych:

{T = 1, W = 3, O = 2, F = 0, U = 6, R = 4}.

Problemy z ograniczeniami mogą mieć różny stopień skomplikowania, a co za tym idzie, wymagać bogatszego języka opisu i bardziej złożonego mechanizmu wnioskowania. Ilustruje to poniższy przykład obejmujący wnioskowanie temporalne:

1) SPOTKANIE trwało cały dzień.
2) Każda osoba uczestniczyła w nim przez określony czas.
3) SPOTKANIE zaczęło się, gdy ANNA była obecna, a zakończyło się, gdy BARBARA była obecna.

4) BARBARA przybyła po rozpoczęciu spotkania.
5) CELINA była obecna, gdy przybyła BARBARA, lecz było to już po wyjściu ANNY.
6) DOROTA rozmawiała z BARBARĄ w obecności CELINY. Pytanie: czy ANNA spotkała BARBARĘ?

Ze względu na swoją uniwersalność podejście oparte na CSP znalazło wiele zastosowań. Z perspektywy kognitywistyki warto w tym miejscu wspomnieć klasyczne prace Paula Thagarda i Kersten Verbeurgt, w których autorzy dyskutują możliwą interpretację spójności pojęciowej jako problem z ograniczeniami.

Do rozwiązywania złożonych problemów z ograniczeniami najczęściej stosuje się specjalizowane pakiety udostępniające język modelowania ograniczeń i mechanizmy wnioskujące (ang. *solvers*). Przykładem takiego narzędzia jest otwarty pakiet ECLiPSe[6].

6. Reguły jako model wiedzy

Systemy regułowe

Termin „systemy regułowe" (ang. *rule-based systems*) można odnieść do szeregu rozwiązań wykorzystujących reguły jako mechanizm symbolicznego modelowania wiedzy. Największą popularność zdobyły one dzięki systemom ekspertowym (ang. *expert systems*), z których wiele (przynajmniej na początku) w celu opisu bazy wiedzy posługiwało się właśnie regułami. Systemy ekspertowe przez wiele lat były uznawane za jedną z najważniejszych technologii sztucznej inteligencji. Znalazły liczne zastosowania praktyczne jako główny komponent systemów decyzyjnych i diagnostycznych w medycynie, ekonomii czy przemyśle. Sukces ten zawdzięczają pragmatycznym celom, jakie stawiali sobie ich twórcy. W przeciwieństwie do wczesnych systemów sztucznej inteligencji pragnęli oni dostarczyć rozwiązań technicznych, które będą naśladowały relatywnie wąski fragment inteligentnego działania człowieka, jakim jest podejmowanie decyzji w jednej dobrze określonej dziedzinie, na przykład diagnostyce medycznej pewnej grupy chorób. Współcześnie systemy ekspertowe najczęściej są osadzane jako komponenty większych systemów informatycznych i poza klasyczną reprezentacją regułową mogą wykorzystywać inne metody reprezentacji i wnioskowania.

6 Więcej o pakiecie na < www.eclipseclp.org> [dostęp: 05.08.2015].

Rodzaje reguł

Ogólnie w systemach regułowych można wyróżnić kilka postaci reguł. Podstawową jest reguła decyzyjna, związana z wnioskowaniem wprzód (od przesłanek do konkluzji):

JEŻELI są spełnione pewne warunki WTEDY podejmij określoną decyzję

Z logicznego punktu widzenia taka reguła może odpowiadać implikacji. Niejednokrotnie wykorzystuje się reguły w postaci derywacyjnej, związanej z wnioskowaniem wstecz (od celu do przyczyn/warunków):

dana sytuacja zachodzi WTEDY GDY są spełnione pewne warunki

Przykładowo:

JEŻELI pada deszcz WTEDY zabieram parasol
zdam egzamin WTEDY GDY będę pilnie studiował przez cały semestr

Można też wskazać inne typy reguł, w tym omawiane już wcześniej ograniczenia, stanowiące ich specyficzny przypadek. Należy także rozważyć reguły decyzyjne bez przesłanek. W takiej sytuacji można ich operacyjnie używać jako mechanizmów definiowania faktów. Nietrudno zauważyć, że baza wiedzy systemu regułowego, składająca się z reguł i faktów, jest do pewnego stopnia analogiczna do bazy wiedzy w języku Prolog. Ponadto, w zależności od przyjętej siły wyrazu reguł, rozważa się bardziej uporządkowaną notację opartą na koncepcji atrybutów (cech).

Przykładowo:

JEŻELI Pogoda(deszczowa) WTEDY Ubranie(peleryna)

Szkieletowe
emy regułowe

W typowych narzędziach do budowy systemów regułowych baza wiedzy może być opisywana za pomocą odpowiedniego języka regułowego, będącego specjalnym mechanizmem programowania. Następnie zawartość takiej bazy jest interpretowana za pomocą mechanizmu wnioskującego (ang. *inference engine*). Takie narzędzia określa się jako szkieletowe systemy ekspertowe (ang. *expert system shells*). Najbardziej rozpowszechnione otwarte implementacje takich systemów to CLIPS czy Jess.

nalizacja bazy
wiedzy

Można również spotkać się z podejściem, w którym reguły są precyzyjnie opisywane za pomocą formalizmu logicznego. Formalizacja bazy wiedzy pozwala na ujednoznacznienie jej semantyki oraz, co nie mniej istotne, na wyeliminowanie szeregu anomalii logicznych, jakie mogą się pojawić podczas konstruowania regułowych baz wiedzy. Do typowych anomalii należy zaliczyć niezupełność (ang. *incompleteness*), rozumianą między innymi jako brak pewnych informacji, niespójność (ang. *inconsistency*), przykładowo występowanie reguł ze sobą sprzecznych, czy też nadmiarowość (ang. *redundancy*), oznaczającą na przykład występowanie reguł zduplikowanych lub równoważnych logicznie. Usunięcie tych i innych defektów pozwala na zwiększenie wydajności systemu regułowego, a często – co nawet istotniejsze – bezpieczeństwa jego działania.

Reguły biznesowe

Do najnowszych zastosowań systemów regułowych można zaliczyć obszar reguł biznesowych (ang. *business rules*), które stanowią fundament aplikacji wspomagających działanie dużych przedsiębiorstw. Często są one wykorzystywane w instytucjach finansowych czy przy zarządzaniu kontaktami z klientami (ang. *customer relationship management*), między innymi w automatycznym profilowaniu klientów. Co za tym idzie, nowa generacja systemów regułowych najczęściej jest utożsamiania z systemami zarządzania regułami biznesowymi (ang. *business rules management engine*). Klasycznym przykładem otwartej implementacji takiego narzędzia jest system Drools[7].

7. Drzewa i tablice decyzyjne

Wizualizacja informacji

Jedną z ważnych – z praktycznego punku widzenia – cech metody reprezentacji wiedzy jest możliwość wizualizacji informacji. Prezentowane dotychczas metody w znacznej mierze opierały się na tekstowej notacji pewnych formuł matematycznych. Tymczasem w projekcie projektowania i prototypownia rzeczywistych systemów znacznie przydatniejsze okazują się metody wizualne. Cechują się one najczęściej większą siłą wyrazu i pozwalają nie tylko na łatwiejsze konstruowanie modelu, lecz również na lepszą komunikację projektantów systemu. Jednymi z podstawowych metod reprezentacji wizualnej są drzewa i tablice decyzyjne.

Tablice decyzyjne

Tablica decyzyjna składa się z szeregu komórek uporządkowanych w wiersze i kolumny. W zależności od przyjętej konwencji jedne lub drugie odpowiadają semantycznie regułom decyzyjnym. W nagłówkach tablicy zapisywane są przeważnie nazwy atrybutów. W poniższym przykładzie przyjęto, że regułom odpowiadają kolejne wiersze tablicy, a kolumny interpretujemy od lewej do prawej strony.

7 Zob. <drools.org> [dostęp: 05.08.2015].

PoraRoku	Opady	Ubranie
lato	nie	*t-shirt*
jesień	tak	*palto*
zima	tak	kurtka_z_kapturem

W przykładowej tablicy decyzyjnej zapisane są trzy reguły, w których zostały użyte trzy atrybuty, z czego dwa w częściach decyzyjnych reguł, a jeden w części warunkowej. Jesteśmy w stanie wskazać dziedziny tych atrybutów (czyli zbiór ich dopuszczalnych wartości), w szczególności możemy zauważyć, że atrybut „Opady" przyjmuje tylko dwie wartości: tak/nie, prawda/fałsz, 0/1 i możemy go nazwać binarnym lub w sensie prawdziwościowym – boolowskim[8].

pełność zbioru reguł

Ponadto, biorąc pod uwagę semantykę atrybutu „PoraRoku", można postawić pytanie, czy taki trzyelementowy zbiór reguł jest zupełny, tzn. czy dla każdej możliwej kombinacji wartości atrybutów warunkowych jest określona decyzja (innymi słowy, czy zbiór reguł pokrywa iloczyn kartezjański dziedzin atrybutów warunkowych). Nasuwającym się uzupełnieniem byłoby dodanie reguły opisującej sposób ubrania się podczas wiosny. Co więcej, w celu zapewnienia zupełności ubiór powinien być uzupełniony o sytuacje, w których pada i nie pada. Co za tym idzie, w systemie zupełnym powinno być określone osiem reguł.

Tablice decyzyjne ułatwiają projektowanie zbiorów reguł, a także niejednokrotnie ich analizę. W przypadku dużych zbiorów reguł (liczących niejednokrotnie tysiące reguł i dziesiątki atrybutów) bardziej użyteczna staje się notacja drzew decyzyjnych odzwierciedlająca hierarchię atrybutów i proces podejmowania decyzji.

ewa decyzyjne

Z formalnego punktu widzenia drzewo (w tym drzewo decyzyjne) możemy traktować jako graf acykliczny (zbiór wierzchołków/węzłów, połączonych krawędziami). Dokładniej drzewo definiuje się najczęściej jako nieskierowany (przy określaniu krawędzi kolejność węzłów nie ma znaczenia) graf spójny (pomiędzy każdymi dwoma węzłami istnieje łącząca je ścieżka, ciąg krawędzi) i acykliczny (węzły w tej ścieżce się nie powtarzają).

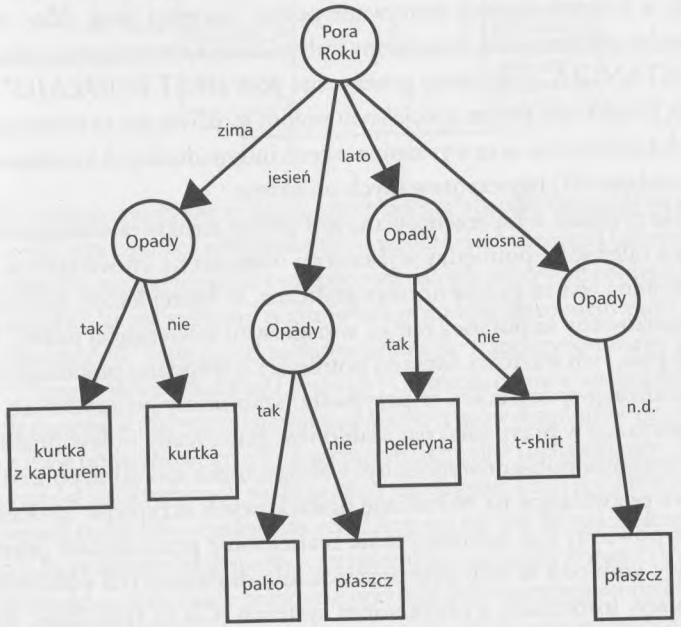
W drzewie decyzyjnym w węzłach zapisuje się nazwy atrybutów, a na krawędziach zaznacza się ich wartości (formalnie: etykietuje się je). Dla zaprezentowanej poprzednio tablicy decyzyjnej można skonstruować drzewo zilustrowane niżej na rysunku 1. Dodano w nim brakujące węzły,

8 Boolowski, czyli przyjmujący jedną z dwóch wartości prawdziwości, najczęściej oznaczanych 0 i 1.

przy czym w przypadku „wiosny" wprowadzono wartość „n.d." oznaczającą, że „płaszcz" wkłada się bez względu na opady. Ponadto przyjęto notację grafu dwudzielnego, gdzie w liściach wprowadzono inny typ wierzchołków dla określenia wartości atrybutu decyzyjnego.

W ogólnym przypadku można przyjąć, że z deklaratywnego punktu widzenia drzewa i tablice decyzyjne odpowiadają zbiorom reguł. Z operacyjnego punktu widzenia, w tym ze względu na sposób przetwarzania, obie reprezentacje pozwalają na określenie kolejności przetwarzania informacji. W reprezentacjach strukturalnych ważną rolę odgrywa również wizualny aspekt reprezentacji, podobnie ma się rzecz z modelem grafowym w sieciach semantycznych.

Rysunek 1. Drzewo decyzyjne



8. Reprezentacje strukturalne

Porządkowanie i hierarchizacja informacji

Reprezentacje strukturalne są używane głównie do porządkowania i hierarchizacji informacji przechowywanych w bazie wiedzy systemu. Za klasyczną reprezentację strukturalną wprowadzoną w sztucznej inteligencji uznaje się szablony semantyczne zastosowane przez Marvina

Minskiego (nazywane czasem ramami, ang. *frames*). Początkowo zostały one zaproponowane w celu ułatwienia reprezentowania i interpretowania danych w systemie wizyjnym, na przykład rozpoznawania obiektów i ich zależności przestrzennych w obrazie z kamery. Z czasem stały się jednym z uniwersalnych narzędzi strukturalizowania informacji, pozwalającym również na wizualizację. Szablony semantyczne nie doczekały się pojedynczej standaryzowanej notacji czy formalizacji i – podobnie jak omawiane sieci semantyczne – stały się pewnego rodzaju konwencją.

Szablony indywidualne i generyczne

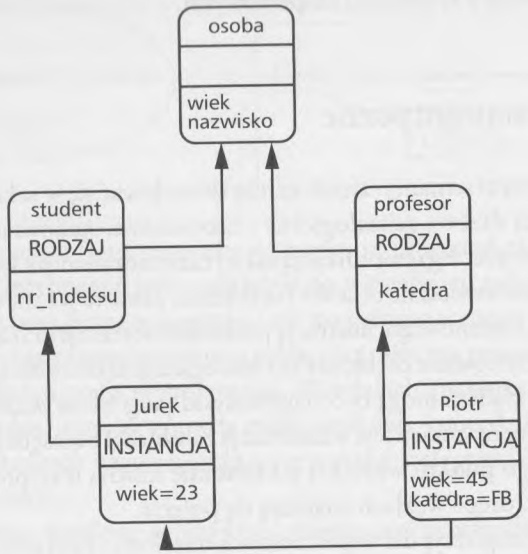
Zazwyczaj przyjmuje się, że pojedynczy szablon reprezentuje pewien obiekt lub klasę (rodzaj) obiektów w obserwowanym i opisywanym świecie. W pierwszym przypadku można mówić o szablonie indywidualnym, w drugim o szablonie generycznym. Szablony składają się z pól (ang. *slot*), w których są przechowywane pewne wartości (ang. *filler, value*). Ponadto przyjmuje się, że szablony indywidualne zawierają specjalne pole „INSTANCJA", a szablony generyczne pole „JEST RODZAJU" (ang. *is-a*). Dzięki tym dwóm specjalnym polom możliwe jest utworzenie hierarchii szablonów oraz wyróżnienie cech indywidualnych i rodzajowych (gatunkowych) reprezentowanych obiektów.

Na rysunku 2 zaprezentowana jest prosta struktura szablonów opisująca zależności pomiędzy wybranymi osobami na uniwersytecie. Wykorzystana jest tu prosta notacja graficzna, w której każdy szablon jest wizualizowany za pomocą ramki, w nagłówku zawierającej nazwę, a poniżej pola i ich wartości. Strzałki pomiędzy szablonami pokazują relację generalizacji-specyfikacji w przypadku szablonów generycznych oraz instantyzacji w przypadku par szablonów generyczny-indywidualny.

Struktura szablonów

Struktura szablonów może być wzbogacona o specjalne pola zdarzeniowe pozwalające na określenie dodatkowych skryptów (procedur), które powinny być automatycznie realizowane przez system przetwarzający szablony w celu przeprowadzania dodatkowych obliczeń czy wymiany informacji z otoczeniem systemu. Co za tym idzie, strukturę szablonów można potraktować jako definicję pewnego modelu informacji, zaś skrypty jako mechanizm określający dynamikę ich przetwarzania. W takim ujęciu rozbudowany system przetwarzania szablonów i uruchamiania skryptów staje się kompletnym środowiskiem programowania.

Rysunek 2. Szablony semantyczne



Koncepcja szablonów semantycznych odegrała bardzo dużą rolę we wprowadzeniu reprezentacji obiektowych do sztucznej inteligencji i informatyki. Inspirowany nią paradygmat programowania i projektowania obiektowego od ponad 20 lat jest jednym z podstawowym podejść do modelowania oprogramowania. W fazie projektowania używa się standaryzowanego języka UML (ang. *Unified Modeling Language*)[9]. Pozwala on na wizualne modelowanie systemów z użyciem kilkunastu różnych typów diagramów dzielących się na dwie podstawowe grupy: diagramy strukturalne i behawioralne. Wśród diagramów strukturalnych najważniejszy jest diagram klas i obiektów, oparty na notacji szablonów semantycznych. Diagram ten jest jednak znacznie bogatszy. Pozwala na określanie szeregu zależności pomiędzy klasami i wzbogacenie modelu o informacje przydatne przy jego implementacji w obiektowym języku programowania. Jednym z pierwszych takich języków był Smalltalk, ważną rolę odegrał też późniejszy Eiffel. Współcześnie najczęściej wykorzystuje się C++, Javę, ObjectiveC, choć elementy programowania obiektowego są też dostępne w wielu innych językach.

Język UML

Warto w tym miejscu zauważyć, że reprezentacje strukturalne, takie jak szablony semantyczne, można z powodzeniem opisać z wykorzystaniem wcześniej zaprezentowanego paradygmatu programowania w logice.

9 Zob. www.omg.org/spec/UML.

Same szablony semantyczne były często łączone z innymi rozwiązaniami, w tym również z systemami ekspertowymi.

9. Sieci semantyczne

Wizualizacja zależności pomiędzy pojęciami

Początków sieci semantycznych należy doszukiwać się w szkicach przedstawiających drzewa genealogiczne i taksonomie, stanowiące uporządkowane słowniki. Egzemplifikacją takiej taksonomii może być klasyczne wczesnośredniowieczne drzewo Porfiriusza, zawarte we wstępie do prac Arystotelesa i stanowiące ilustrację podziału substancji przez odpowiednio zhierarchizowane cechy, czy też biologiczna taksonomia Linneusza. Innym przykładem mogą być mapy umysłu (ang. *mind maps*). Wspólną cechą tych reprezentacji jest wizualizacja zależności pomiędzy pojęciami. Z formalnego punktu widzenia taksonomię można interpretować jako drzewo, w którego węzłach znajdują się pojęcia.

Wizualizacja uł logicznych

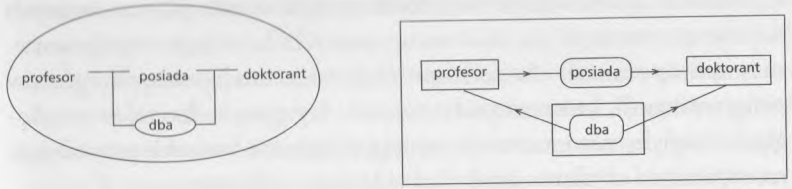
Kolejną inspirację dla współczesnych ontologii i sieci semantycznych można dostrzec w metodach wizualizacji formuł logicznych. Charles S. Peirce zaproponował notację grafów egzystencjalnych, która w jego zamierzeniu miała pozwalać na wizualizację formuł rachunku predykatów sformułowanego przez Gottloba Fregego. Notacja pozwalała nie tylko na zapis formuł, lecz również na ich przekształcenia, a co za tym idzie – na wnioskowanie. Wychodząc od tej koncepcji, John Sowa sformułował na potrzeby sztucznej inteligencji i lingwistyki grafy konceptualne. Spośród ich licznych zalet warto wspomnieć o dostosowaniu do mechanizmów programowania logicznego. Opracowano między innymi implementację systemu przetwarzającego grafy konceptualne w języku Prolog[10]. Sowa klasyfikuje te notacje jako sieci asercyjne, podczas gdy taksonomie określa sieciami definicyjnymi. Wyróżnia również sieci wykonawcze, do których zalicza między innymi sieci bayesowskie[11].

Na rysunku 3 zaprezentowano przykład obu typów grafów dla użytej przez Sowę formuły „Każdy profesor, który ma doktoranta, dba o niego".

10 Zob. <prologpluscg.sourceforge.net> i <amine-platform.sourceforge.net> [dostęp: 05.08.2015].

11 Sieci bayesowskie są formalizmem pozwalającym na graficzne modelowanie zależności probabilistycznych pomiędzy zdarzeniami.

Rysunek 3. Przykład grafu konceptualnego (lewy) i egzystencjalnego (prawy)



Budowa sieci semantycznych

Sieci semantyczne (ang. *semantic networks*) są powszechnie używane w sztucznej inteligencji jako narzędzie do wizualizacji zależności pomiędzy pojęciami. Sama notacja sieci nie ma jednoznacznego standardu, czyli jest jedynie pewną konwencją notacyjną. Sieć ma postać etykietowanego spójnego grafu skierowanego. W węzłach znajdują się pojęcia (najczęściej opisywane za pomocą rzeczowników), krawędzie reprezentują relacje, których nazwy znajdują się w etykietach grafu (najczęściej są to czasowniki).

Wprawdzie określone w powyższy sposób sieci są wygodnym narzędziem do konceptualizacji i prototypownia systemów inteligentnych, jednak brak ich jednoznacznego standardu i jednoznacznej formalizacji uniemożliwiał ich szersze użycie i wprowadzenie procedur automatycznego wnioskowania. Istotne wysiłki na gruncie formalizacji przeprowadzono w projekcie KL-One, łączącym szablony i sieci semantyczne oraz wprowadzającym formalizm, który został później rozwinięty do współcześnie używanych logiki opisowych. Dynamiczny rozwój oraz formalizacja metod reprezentacji wiedzy rozszerzających sieci semantyczne nastąpiły wraz z powołaniem projektu sieci semantycznej.

KL-One

Projekt sieci semantycznej

Za oficjalny początek projektu sieci semantycznej (ang. *the Semantic Web*, nie należy mylić z sieciami semantycznymi, ang. *semantic networks*) uznaje się artykuł *The Semantic Web* autorstwa pomysłodawcy World Wide Web Tima B. Lee i współpracowników, który ukazał się w „Scientific American" (maj 2001). Autorzy proponują w nim nową warstwową architekturę dla sieci komputerowej następnej generacji. Dokumenty byłyby przechowywane w sposób uporządkowany i strukturalizowany za pomocą języka XML (ang. *eXtensible Markup Language*), który mógłby być automatycznie interpretowany przez komputery. Ponadto dokumentom towarzyszyłyby adnotacje semantyczne w języku RDF (ang. *Resource Description Framework*) pozwalające na formułowanie stwierdzeń-faktów o strukturze „podmiot-orzeczenie-dopełnienie". Informacje o pojęciach pojawiających się w adnotacjach RDF byłyby natomiast porządkowane i hierarchizowane w ontologiach formalnych zapisywanych w języku

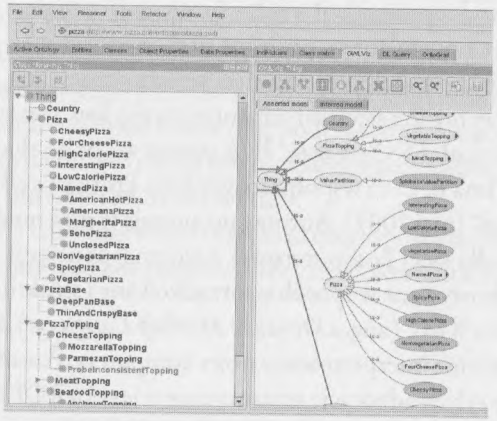
Język OWL

OWL (ang. *Web Ontology Language*). To właśnie OWL jest głównym narzędziem pozwalającym na całościowy opis ontologii, stanowiących współczesne rozwinięcie sieci semantycznych. Lee i jego współpracownicy zaproponowali również konstruowanie inteligentnych agentów programowych, które na podstawie wiedzy zgromadzonej w ontologiach mogłyby automatycznie wyciągać logiczne wnioski, pozwalające na wspieranie użytkowników sieci w szeregu czynności.

Obecnie OWL stanowi całą rodzinę języków różniących się oferowanymi mechanizmami składniowymi i semantycznymi, a co za tym idzie – siłą wyrazu. W pierwszej wersji standardu OWL wyróżnione trzy podzbiory języka: OWL Lite, DL i Full, a w drugiej – coraz powszechniej używanej – trzy profile EL, QL i RL. Podczas gdy w pierwszej generacji podjęzyków (OWL1) skupiano się na ich sile wyrazu i możliwości opisu w logice opisowej, w drugiej (OWL2), poza formalizacją, położono nacisk na obszary zastosowań i typ projektowanych ontologii. Profil EL ma charakter uniwersalny, QL upraszcza łączenie ontologii z relacyjnymi bazami danych, zaś RL upraszcza integrację z systemami regułowymi. Poza standaryzacją opisu ontologii podstawową zaletą OWL (DL i OWL2) jest pełna formalizacja tego opisu za pomocą logiki matematycznej.

Na rysunku 4 zaprezentowana została wizualizacja przykładowej ontologii[12] wykonana z użyciem narzędzia Protege[13], stanowiącego jeden z najpopularniejszych edytorów ontologii.

Rysunek 4. Wizualizacja ontologii formalnej



12 Zob. <owl.cs.manchester.ac.uk/co-ode-files/ontologies/pizza.owl> [dostęp: 05.08.2015].

13 Zob. <www. protege.stanford.edu> [dostęp: 05.08.2015]

Logiki opisowe

Logiki opisowe (ang. *description logics*, DL) to grupa formalizmów opracowana na potrzeby pełnego opisu reprezentacji wiedzy i zadań wnioskowania w ontologiach. Poszczególne warianty logik różnią się składnią (używanymi operatorami), siłą wyrazu i złożonością obliczeniową algorytmów rozwiązujących zadania wnioskowania. Klasyfikację tych języków można znaleźć między innymi na stronie internetowej *Complexity of Reasoning in Description Logics*[14]. Baza wiedzy w logice opisowej obejmuje część asercyjną (ABox) i część terminologiczną (TBox). Logiki opisowe wprowadzają własną terminologię, lecz są podzbiorem rachunku predykatów pierwszego rzędu. Najistotniejsze pojęcia to: koncept (klasa), rola (własność/predykat) oraz indywiduum (obiekt).

Podstawowa logika opisowa, ALC, pozwala między innymi na definiowanie pojęć poprzez ich teoriomnogościowe przecięcie, sumę i dopełnienie, a także kwantyfikację uniwersalną oraz egzystencjalną ze względu na rolę. ABox zwiera asercje co do pojęć i ról, a TBox definiuje zawieranie pojęć. Różne warianty OWL używają innych logik opisowych. W szczególności OWL Full używa logiki *SHOIN(D)* a OWL2 *SROIQ(D)*. Dzięki użyciu logiki opisowej baza wiedzy w OWL ma odpowiadającą jej postać logiczną, a wszystkie zadania wnioskowania są formalnie zdefiniowane i sprowadzane do sprawdzania spełnialności odpowiednich formuł logicznych. Do tego celu służą specjalizowane narzędzia wnioskujące (ang. *DL reasoners*), takie jak Pellet[15] czy Hermit[16].

Sieci semantyczne stanowią przystępną metodę modelowania zależności pomiędzy pojęciami. Opracowanie języka OWL pozwoliło na wprowadzenie standaryzowanych narzędzi pozwalających na budowanie ontologii dużej grupie osób. Ontologie formalne, a szerzej metody semantycznego opisu i zarządzania wiedzą, stały się istotnymi i powszechnie używanymi narzędziami opisu wiedzy symbolicznej w sztucznej inteligencji, informatyce i kognitywistyce.

10. Podsumowanie

Celem niniejszego rozdziału jest zaprezentowanie wybranych istotnych metod opisu baz wiedzy w sztucznych systemach inteligentnych. Większość z przywołanych tu metod posiada odpowiadające im formalizacje

14 Zob. <www.cs.man.ac.uk/~ezolin/dl> [dostęp: 05.08.2015].

15 Zob. <clarkparsia.com/pellet> [dostęp: 05.08.2015].

16 Zob. <hermit-reasoner.com> [dostęp: 05.08.2015].

w wybranym rachunku logicznym. Niektóre oferują możliwość wizualizacji bazy wiedzy, co z punku widzenia użytkowników i projektantów ułatwia proces jej budowania. Z praktycznego punktu widzenia znaczącą rolę odgrywa to, czy dla metody danej reprezentacji powstały narzędzia wspierające ten proces. Przykład rozwoju ontologii formalnych pokazuje, że czasami dopiero połączenie kilku elementów udostępnia daną metodę szerszemu gronu użytkowników i projektantów.

Uczenie maszynowe

Omawiane tu metody są używane przez inżynierów wiedzy do manualnego budowania baz wiedzy w procesie inżynierii wiedzy. Patrząc na tworzenie baz wiedzy z szerszej perspektywy, należałoby również wskazać podejścia pozwalające na automatyczne budowanie baz wiedzy. W sztucznej inteligencji rozpatruje się wiele podejść do uczenia maszynowego (ang. *machine learning*), które zdaniem wielu stało się niemal oddzielną dziedziną nauki. Istnieje szereg dojrzałych metod uczenia stosowanych na potrzeby reprezentacji symbolicznych i logicznych. Warto tu wspomnieć choćby o algorytmach uczenia drzew decyzyjnych, budowania zbiorów reguł czy indukcyjnego programowania logicznego. Cechą wspólną tych podejść jest najczęściej indukcja przez algorytm uczący modeli logicznych w postaci reguł, drzew czy formuł rachunku zdań z obszernej bazy faktów stanowiących przykłady uczące. Niejednokrotnie metody te pozwalają na bardzo skuteczne wsparcie projektanta bazy wiedzy, a w prostszych przypadkach na jej w pełni automatyczne skonstruowanie.

Przewodnik bibliograficzny

Sztuczna inteligencja i systemy inteligentne

Flasiński M., (2011) – *Wstęp do sztucznej inteligencji*, Warszawa: WN PWN.

Kisielewicz A., (2011) – *Sztuczna inteligencja. Podsumowanie przedsięwzięcia naukowego*, Warszawa: WNT.

Levesque H., (2012) – *Thinking as Computation*, Cambrigde, MA: MIT Press.

Poole D., Mackworth A., (2010) – *Artificial Intelligence: Foundations of Computational Agents*, Cambridge, Mass.: CUP.

Russell S., Norvig P., (2010) – *Artificial Intelligence: A Modern Approach*, Pearson.

Rutkowski L., (2012) – *Metody i techniki sztucznej inteligencji. Inteligencja obliczeniowa*, Warszawa: WN PWN.

Wooldridge M., (2009) – *An Introduction to MultiAgent Systems*, Hoboken, NJ: John Wiley & Sons.

Podstawy logiczne, programowanie w logice i z ograniczeniami

Apt K.R., (2003) – *Principles of Constraint Programming*, Cambridge, Mass.: Cambridge Univeristy Press.

Apt K.R., Wallace M.G., (2007) – *Constraint Logic Programming Using ECLiPSe*, Cambridge, Mass.: Cambridge Univeristy Press.

Bratko I., (2011) – *Prolog Programming for Artificial Intelligence*, Boston: Addison-Wesley.

Dechter R., (2003) – *Constraint processing*, Burlington, MA: Morgan Kaufman.

Genesereth M.R., Nilsson N.J., (1987) – *Logical Foundations of Artificial Intelligence*, Burlington, MA: Morgan Kaufmann.

Kripke S., (1963) – *Semantical Analysis of Modal Logic*, „Zeitschrift für Mathematische Logik und Grundlagen der Mathematik" 9, s. 67-96.

Ligęza A., (2006) – *Logical Foundations for Rule-Based Systems*, Heidelberg: Springer.

Newell A., (1982) – *The Knowledge Level*, „Artificial Intelligence" 18(1), s. 87-127.

Nilsson U., Małuszyński J., (1995) – *Logic, Programming and Prolog*, Hoboken, NJ: John Wiley & Sons.

Pawlak Z., (1991) – *Rough Sets – Theoretical Aspects of Reasoning about Data*, Boston-London-Dordrecht: Kluwer Academic Publishers.

Thagard P., Verbeurgt K., (1998) – *Coherence as Constraint Satisfaction*, „Cognitive Science" 22, s. 1-24.

Zadeh L.A. i in. (red.), (1996) – *Fuzzy Sets, Fuzzy Logic and Fuzzy Systems*, Singapore: World Scientific Publishing.

Reprezentacja wiedzy i sieć semantyczna

Allemang D., Hendler J., (2008) – *Semantic Web for the Working Ontologist, Effective Modelling in RDFS and OWL*, Burlington, MA: Morgan Kaufmann.

Antoniou G., Van Harmelen F., (2004) – *A Semantic Web Primer*, Cambridge, Mass.: MIT Press.

Baader F. i in., (2008) – *The Description Logic Handbook*, Cambridge, Mass.: CUP.

Berners-Lee T., Hendler J., Lassila Or., (2001) – *The Semantic Web*, „Scientific American" 284 (5), s. 34-43.