Od redaktora i wydawcy

Czym jest wiedza?

Czynnym pytaniem zastanawiali się już starożytni, głowili się filozofowie, lamią słowa psychologowie, próbowali na nie odpowiedzieć praktycznie. Choć wiedza powierzchownie kojarzona jest z informacjami, nauką, doświadczeniem, zbiorem faktów, nie posiada ona jednej, uniwersalnej definicji. W zależności od kontekstu odpowiedź na to pytanie może przybrać inną postać i formę.

Czym jest przetwarzanie wiedzy?

To kolejna niewiadoma, sięgająca w swej maturze do sposobów reprezentacji wiedzy, jej interpretacji i wykorzystania, włączając w to metody wnioskowania i podejmowania decyzji. Podobnie jak w pytaniu o wiedzę, mnogość możliwych odpowiedzi może tu być ogromna.

Czym jest komputerowe przetwarzanie wiedzy?

Odpowiedź na to pytanie jest projekcją sumy odpowiedzi na powyższe dwa pytania na płaszczyznę zdefiniowaną przemianami „komputerowe”. Mówiąc prosto, jest to dziedzina, w której wykorzystuje się komputery do rozwiązywania złożonych problemów zdefiniowanych na różnych poziomach abstrakcji. Wykracza ona poza samą implementację algorytmów skierowanych wartości parametrów opisujących obiekt na świat, istnieje na pograniczu sztucznej inteligencji i inteligencji instytucji. W sąsiadach pomiędzy czynią, czym możemy w stanie sami przeanalizować, a czym, co unika naszym zmysłem z powodu wielkości ilościowych i szczegółowych.

W niniejszej książce zebrano opracowania wykonane przez studentów V roku Automatyki i robotyki w ramach kursu „Komputerowe przetwarzanie wiedzy” prowadzonego przez mnie na Politechnice Wrocławskiej w semestrze zimowym 2009/2010. Zgodnie z zarysowanym kształtem odpowiedzi na ostatecznie powyższych pytań, zadanie studentów polegało na tym, aby zaimplementować jakiekolwiek publikowane bądź autorskie algorytmy, co na posłużeniu się zdobywaną dzięki temu wiedzą do rozwiązywania jakiegoś problemu na wyższym poziomie abstrakcji.

Zakres tematyczny opracowań można zauważyć w następującym zestawieniu:
- Przetwarzania dokumentów
- Portale społecznościowe
- Moduły informacyjne
- Ochrona informacji w sieci
- Reprzäsentacja wiedzy
- Operacje na danych
- Logika, programowanie i wnioskowanie
- Ontologie informatyczne

Mam nadzieję, że lektura tych opracowań okaza się interesująca dla czytelnika.

Tomasz Kubik
Wrocław, wrzesień 2011
KOMPUTEROWE PRZETWARZANIE WIEDZY

Kolekcja prac 2009/2010
pod redakcją Tomasza Kubika
Skład komputerowy, projekt okładki

Tomasz Kubik

Książka udostępniana na licencji Creative Commons: Uznanie autorstwa-Użycie niekomercyjne-Na tych samych warunkach 3.0, Wrocław 2011. Pewne prawa zastrzeżone na rzecz Autorów i Wydawcy. Zezwala się na niekomercyjne wykorzystanie treści pod warunkiem wskazania Autorów i Wydawcy jako właścicieli praw do tekstu oraz zachowania niniejszej informacji licencyjnej tak długo, jak tylko na utwory zależne będzie udzielana taka sama licencja. Tekst licencji dostępny na stronie: http://creativecommons.org/licenses/by-nc-sa/3.0/pl/


Wydawca
Tomasz Kubik

Druk i oprawa
I-BiS sc., ul. Lelewela 4, 53-505 Wrocław

Skład niniejszej książki wykonano w systemie LaTeX
Słowo wstępné

1 Indeksowanie w systemach zarządzania dokumentami
   1.1. Wprowadzenie
   1.2. Szczegóły DMS
   1.3. Indeksowanie
   1.4. Indeksacja czasowo-przestrzenna w repozytoriach danych
       1.4.1. Cechy przestrzennych baz danych
   1.5. Projekt przykładowej aplikacji
       1.5.1. Wykorzystane narzędzia
       1.5.2. GoogleMaps API
   1.6. Opis wykonanej implementacji
       1.6.1. Zasada działania programu
       1.6.2. Dodawanie plików do bazy

2 Elektroniczne systemy podawcze do obsługi konferencji i wydawnictw
   2.1. Funkcje i zastosowania
       2.1.1. Przykłady
   2.2. Podstawy implementacji
       2.2.1. Moodle
       2.2.2. Django
       2.2.3. MySQL
       2.2.4. Apache
   2.3. Prototyp systemu
       2.3.1. Instalacja w Ubuntu 9.10
       2.3.2. Część niefunkcjonalna
       2.3.3. Część funkcjonalna
   2.4. Uwagi końcowe

3 Portale społecznościowe
   3.1. Wstęp
       3.1.1. Istota portali społecznościowych
       3.1.2. Etapy rozwoju Internetu
Spis treści

3.2. Narzędzia programowe do budowy portalu ........................................ 36
   3.2.1. LAMP ......................................................... 36
   3.2.2. Narzędzia zarządzania bazami danych ......................................... 37
   3.2.3. CMS .................................................................. 37
   3.2.4. Joomla! .................................................................. 38
   3.2.5. Community Builder .............................................................. 39
   3.2.6. Instalacja .................................................................. 40

3.3. Charakterystyka wybranych portalu społecznościowych .................................... 40
   3.3.1. demotywatory.pl ............................................................... 40
   3.3.2. digart.pl .................................................................... 41
   3.3.3. dioda.com.pl ................................................................. 41
   3.3.4. wrzuta.pl .................................................................. 42

3.4. Metodologia tworzenia własnego portalu ..................................................... 42
   3.4.1. Koncepcja .................................................................. 42
   3.4.2. Domena .................................................................... 42
   3.4.3. Narzędzia .................................................................. 43
   3.4.4. Zabezpieczenia .................................................................. 45

3.5. Założenia projektowe .................................................................................. 47
   3.5.1. Własności niefunkcjonalne ......................................................... 48
   3.5.2. Własności funkcjonalne ....................................................... 49

3.6. Implementacja ................................................................................. 50
   3.6.1. Od strony gościa ................................................................. 50
   3.6.2. Od strony użytkownika ........................................................... 51
   3.6.3. Od strony administratora .......................................................... 51
   3.6.4. Moduły i dodatkowe elementy ......................................................... 54
   3.6.5. Zabezpieczenia .................................................................. 55
   3.6.6. RSS ........................................................................... 55

4 Metody obrony przed robotami sieciowymi ...................................................... 56
   4.1. Wstęp ........................................................................... 56
   4.2. Roboty sieciowe .................................................................. 57
      4.2.1. Czym są roboty sieciowe? ....................................................... 57
      4.2.2. Jak działa robot indeksujący? ............................................. 57
   4.3. Zastosowanie robotów sieciowych ...................................................... 58
      4.3.1. Roboty neutralne ................................................................. 58
      4.3.2. Roboty złośliwe ................................................................. 61
   4.4. Obrona przed robotami sieciowymi ...................................................... 62
      4.4.1. Sprecyzowanie zasad użytkowania strony, Robots.txt .................. 62
      4.4.2. Morale na straży bezpieczeństwa WWW ................................. 62
      4.4.3. Walka z atakami pochodzącymi z botnetów .......................... 63
      4.4.4. Rejestracja jako ochrona przed botami ....................................... 65
      4.4.5. CAPTCHA ................................................................. 65
      4.4.6. Antyspam ................................................................. 67
      4.4.7. Obrona przed atakiem typu DoS i DDoS ................................... 68
      4.4.8. Podsumowanie .................................................................. 69
4.5. Projekt gotowych rozwiązań ochronnych przed netbotami 69
  4.5.1. Założenia projektowe 69
  4.5.2. Wynik projektu 69
  4.5.3. Dodanie toolbox’a 70
  4.5.4. Akceptacja regulaminu 71
  4.5.5. Projekt - uwagi końcowe 72

5 Programowanie agentowe w eksploracji danych 74
  5.1. Wstęp 74
    5.1.1. Wprowadzenie do programowania agentowego i eksploracji danych 74
  5.2. Systemy wieloagentowe 76
    5.2.1. Podstawowe własności 76
    5.2.2. Opis komunikacji w systemie wieloagentowym 77
    5.2.3. Koordynacja współpracy, rozwiązywanie konfliktów i planowanie 78
  5.3. Koncepcja systemu 80
    5.3.1. Cele systemu i funkcjonalności 80
    5.3.2. Opis działania systemu 82
    5.3.3. Opis Agentów 83
  5.4. Realizacja systemu 85
    5.4.1. Dobór narzędzi, urządzeń i protokołów 85
    5.4.2. Przykłady implementacji 86
    5.4.3. Opis instalacji 87

6 Metody drażenie danych i ich zastosowania 89
  6.1. Wstęp 89
  6.2. Metody drażenie danych 90
    6.2.1. Klasyfikacja 90
    6.2.2. Grupowanie 91
    6.2.3. Odkrywanie asocjacji 91
    6.2.4. Odkrywanie sekwencji 92
  6.3. Opis problemu i jego rozwiązania 92
    6.3.1. Deskryptory wielokątowe 93
    6.3.2. Deformacje wielokątów 93
    6.3.3. Szacowanie dystansu deformacji 94
    6.3.4. Źródło danych 94

7 Transformacja danych za pomocą szablonów 95
  7.1. Wstęp 95
  7.2. XSLT 96
    7.2.1. Algorytm transformacji 97
    7.2.2. Podsumowanie 97
  7.3. GAWK 98
    7.3.1. Wzorce 98
    7.3.2. Akcje 98
Spis treści

7.3.3. Zmienne .................................................. 99
7.3.4. Podsumowanie ........................................... 99
7.4. Systemy produkcyjne ....................................... 99
  7.4.1. Budowa systemu ......................................... 99
  7.4.2. Działanie systemu ....................................... 100
  7.4.3. Podsumowanie .......................................... 101
7.5. Systemy przepisywania termów .......................... 101
7.6. Projekt ...................................................... 102
  7.6.1. Transformacja BLPX do XML .......................... 102
  7.6.2. Transformacja XML do BLPX .......................... 107

8 Budowa modeli informacyjnych i ich profilowanie 109
  8.1. Modele informacyjne ....................................... 109
  8.1.1. Profilowanie modeli informacyjnych .................. 110
  8.2. Języki modelowania ....................................... 110
    8.2.1. XML ................................................ 110
    8.2.2. RDF ............................................... 110
    8.2.3. OWL ............................................... 112
    8.2.4. UML ............................................... 113
    8.2.5. Różnice pomiędzy językami UML i OWL ............ 113
  8.3. Wybrane języki opisu geometrii ........................ 114
    8.3.1. GML ............................................... 114
    8.3.2. X3D .............................................. 115
  8.4. Realizacja profilu X3D ................................... 116
  8.5. Realizacja parsera X3D ................................... 117
    8.5.1. Opis programu ...................................... 120
    8.5.2. Wnioski .......................................... 120

9 Reprezentacja informacji niepewnej i niepełnej 121
  9.1. Wprowadzenie ............................................. 121
  9.2. Reprezentacja wiedzy .................................... 122
    9.2.1. Reprezentacja wiedzy niepewnej .................... 122
    9.2.2. Reprezentacja informacji niepełnej ................ 128
  9.3. Posługiwanie się wiedzą niepewną i niepełną w praktyce 130
    9.3.1. Realizacja projektu ................................ 130
    9.3.2. Przeprowadzone eksperymenty ....................... 133
    9.3.3. Wnioski .......................................... 134

10 Reprezentacja wiedzy zmieniającej się w czasie 136
  10.1. Temporalne bazy danych ................................ 136
    10.1.1. Wprowadzenie ..................................... 136
    10.1.2. Modelowanie czasu w bazach danych ................ 136
    10.1.3. Rodzaje temporalnych baz danych ................... 137
    10.1.4. Znaczniki czasowe ................................ 138
    10.1.5. Przykłady tempoporanych baz danych ............... 140
  10.2. Temporalne rozszerzenie języka XML ................... 140
Spis treści

10.2.1. Język XML ................................................................. 140
10.2.2. Temporalny dokument XML ...................................... 142
10.2.3. Język zapytań dla temporalnego XML ......................... 143
10.3. Przykład zastosowanie temporalnej bazy danych ................. 145
  10.3.1. Ogólna charakterystyka rozwiązania ........................... 145
  10.3.2. Dane temporalne i baza danych ................................ 145
  10.3.3. Dwuczęściowa struktura rozwiązania ......................... 146
  10.3.4. Przykładowe zapytania i odpowiedzi ......................... 148

11 Logika temporalna .......................................................... 150
  11.1. Wprowadzenie ........................................................... 150
  11.2. Podstawowe operatory logik temporalnych ....................... 150
      11.2.1. Operatory czasu liniowego .................................... 150
      11.2.2. Operatory poprzedzania ....................................... 152
      11.2.3. Operatory przeszłości ......................................... 152
      11.2.4. Operatory czasu rozgałęzionego ............................... 152
  11.3. Liniowa logika temporalna .......................................... 153
      11.3.1. Definicja ......................................................... 153
      11.3.2. Składnia ....................................................... 153
      11.3.3. Ważniejsze twierdzenia ...................................... 153
      11.3.4. Zastosowania ................................................. 154
  11.4. Logika CTL .............................................................. 154
  11.5. Logika CTL .............................................................. 155
      11.5.1. Definicja ....................................................... 155
      11.5.2. Składnia ....................................................... 155
      11.5.3. Semantyka ..................................................... 155
      11.5.4. Przykładowe formuły ......................................... 156
      11.5.5. Zastosowania ................................................. 156
  11.6. Zastosowanie LTL w praktyce ...................................... 157
      11.6.1. Opis narzędzi ................................................. 157
      11.6.2. Weryfikacja systemu stworzonego na sterownikach PLC ... 159
  11.7. Podsumowanie ......................................................... 163

12 Programowanie deklaratywne i logika obliczeniowa ................. 164
  12.1. Logika ................................................................. 164
      12.1.1. Krótka historia ............................................... 164
      12.1.2. Metody wnioskowania ........................................ 165
      12.1.3. Logika w matematyce ........................................ 166
      12.1.4. Podstawy logiki matematycznej .............................. 166
      12.1.5. Logika pierwszego rzędu .................................... 169
      12.1.6. Logika drugiego rzędu ....................................... 170
      12.1.7. Logika obliczeniowa ......................................... 171
  12.2. Algorytmy wnioskowania ............................................. 171
      12.2.1. Metoda tabel semantycznych ................................ 171
      12.2.2. Rezolucja ..................................................... 172
  12.3. Programowanie deklaratywne ...................................... 173

7
12.3.1. Programowanie funkcyjne ........................................ 173
12.3.2. Programowanie logiczne ........................................ 174
12.3.3. Programowanie z ograniczeniami .............................. 175
12.4. Przykład zastosowania .............................................. 176
  12.4.1. Prolog .......................................................... 177
  12.4.2. Otter - system automatycznego wnioskowania ............. 178
  12.4.3. Program symulujący świat klocków ......................... 179
13 Miary podobieństwa w ontologiach .................................. 181
  13.1. Ontologia .......................................................... 181
  13.2. Sposoby zapisu ontologii ........................................ 183
  13.3. Miary podobieństwa .............................................. 184
    13.3.1. Podobieństwo wewnętrz ontologii ......................... 185
    13.3.2. Podobieństwo pomiędzy ontologiami ..................... 187
  13.4. Zastosowania pomiaru podobieństwa w ontologiach ........... 190
  13.5. Zrealizowany projekt ............................................ 191
SŁOWO WSTĘPNE


1. Przetwarzanie dokumentów
2. Portale społecznościowe
3. Modele informacyjne
4. Ochrona informacji w sieci
5. Reprezentacja wiedzy
6. Operacje na danych
7. Logika, programowanie i wnioskowanie
8.Ontologie informatyczne

Do ilustracji zawartości książki mogą posłużyć dwa przykładowe projekty:

• K. Turczyn i D. Urban, Reprezentacja wiedzy zmieniającej się w czasie: Autorzy projektu omówili podstawowe zagadnienia związane z modelowaniem czasu w bazach danych. Pokazali też, w jaki sposób można wykorzystać temporalne bazy danych do rozwiązywania problemów praktycznych. Przedstawili przykład narzędzia programowego, które pozwala na przechowywanie informacji
Spis treści

na temat użytkowników komputera działającego w systemie Linux, a także uruchamianych przez nich procesów. Narzędzie to może wspomagać administratorów systemu w realizacji takich zadań, jak monitoring i konserwacja systemu.

- L. Alrae i Ł. Chodorski, Metody obrony przed robotami sieciowymi: Ten projekt, podobnie jak projekt opisany powyżej, ma charakter praktyczny. Po dokonaniu przeglądu metod zabezpieczeń stron internetowych przed atakami automatów (aplikacji potrafiących przeglądać treści stron internetowych) autorzy projektu zaimplementowali zestaw narzędzi programowych dostępnych z poziomu programu Bluefish, pozwalających zabezpieczać strony internetowe. Opisali też przykładową stronę internetową, do której ochrony te narzędzia zostały wykorzystane.

Opisy projektów posiadają spójną koncepcję i jednolitą formę, które nadają im charakter zamkniętej całości. Każdy z nich zaczyna się od wprowadzenia i wyjaśnienia podstawowych pojęć, często poprzez odwołanie się do przykładów. Następnie, formuluje się cel projektu, przedstawia możliwe sposoby jego rozwiązania, a wybrane rozwiązanie omawia w sposób szczegółowy lub ilustruje aplikacją programową. Opis zamyka podsumowanie zawierające ocenę zastosowanych algorytmów i uzyskanych wyników oraz spis wykorzystanej literatury.

Spodziewam się, że ze względu na swoją treść i bardzo staranne opracowanie redakcyjne książka będzie pożądanym źródłem informacji dla studentów i doktorantów pragnących zdobyć podstawowe rozumowanie w zagadnieniach i metodach komputerowego przetwarzania wiedzy, i stosować je w swojej pracy.

Obie kolekcje Komputerowe przetwarzanie wiedzy powstały dzięki inicjatywie programowej i pracy redakcyjnej dra inż. Tomasza Kubika. Chciałbym Mu w tym miejscu wyrazić za to najwyższe uznanie.

Prof. Krzysztof Tchoń,
opiekun specjalności Robotyka,
Wrocław, wrzesień 2011
1.1. Wprowadzenie

Od początku lat 80-tych ubiegłego wieku wielu dostawców oprogramowania zaczęło tworzyć systemy wspierające składowania dokumentów papierowych. Systemy te zarządzali nie tylko papierowymi formami dokumentów, ale także np. zdjęciami. W miarę upływu czasu systemy te zaczęły ewoluować. Wkrótce okazało się, że istniejące potrzeby znacznie wykraczają poza zarządzanie dokumentami fizycznymi i koniecznym jest wprowadzenie nowych rozwiązań, obsługujących dokumenty elektroniczne i inne pliki tworzone za pomocą komputerów.


1. Indeksowanie w systemach zarządzania dokumentami

odpowiedzialnych za nadzorowanie przepływu dokumentów między pracownikami w firmie.

Nie ma jednoznacnej i ścisłej definicji określającej, co ma zawierać system do zarządzania dokumentami. Funkcje tego systemu są zdeterminowane polityką firmy, w której są wdrażane. Podstawowe zagadnienia oraz pytania, które są rozważane przy omawianiu systemów DMS, to:

• Przechowywanie dokumentów - gdzie dokumenty mają być przechowywane? Inaczej mówiąc, określenie miejsca fizycznego dostępu do dokumentów dla użytkowników. Odpowiadając na to pytanie należy określić, czy dokumenty będą dostępne w sieci intranet?

• Wypełnianie dokumentów - w jaki sposób dokumenty mają być wypełnianie treścią? Jaka zostanie wybrana metoda do zindeksowania nowych dokumentów w systemie? Czy dane będą trzymane w bazie danych, czy w postaci systemu plików?

• Wyszukiwanie danych - w jaki sposób zrealizowane ma być wyszukiwanie/przeszukiwanie dokumentów? Zazwyczaj odnajdywanie dokumentów oznacza przeglądanie listy plików zorganizowanej wg jakichś kryteriów (w katalogach o znaczącej nazwie, po nazwie pełnej rolę sygnatury). Często wyszukiwanie polega na szybkim przeglądaniu zawartości plików w poszukiwaniu konkretnych treści. Wykorzystuje się wtedy słowa kluczowe, frazy bądź szablony. Możliwa jest też realizacja wyszukiwania kontekstowego, jest to jednak stosunkowo nowa podejście, a stosowane w nim mechanizmy wciąż są usprawniane aby spełnić wymagania odbiorcy/uzużytkownika końcowego. Najczęściej budowanie dobrego mechanizmu wyszukiwania wymaga przeprowadzenia analizy pozwalającej określić zakres indeksowanej informacji dla poszczególnych dokumentów.

• Bezpieczeństwo danych - w jaki sposób zabezpieczyć dokumenty przed niepożądonym dostępem i zmianą? Należy określić sposób realizacji zabezpieczeń przy dostępie do czytania, usuwania oraz modyfikowania dokumentów, osoby do tego uprawnione, a co za tym idzie grupy osób posiadające dane uprawnienia.

• Kopie, nagłe przypadki utraty danych - czy możliwe są przypadki utraty dokumentów i czy istnieją mechanizmy przywracania utraconych danych? Czy istnieje dobrze zorganizowana polityka kopii bezpieczeństwa? Czy repozytoria dokumentów są replikowane, czy archiwa są fizycznie rozdzielone?

• Retencja dokumentów - jak długo dokumenty mają być trzymane w tak zwanym ‘zachowku”? Czy ustalono politykę, która sprawi, że dokumenty zostaną obsłużone zgodnie z wynikającą z ich ważności hierarchią?

• Archiwizowanie - w jaki sposób jest zorganizowana polityka archiwizacji?

• Dystrybucja dokumentów - w jaki sposób osoby uprawnione uzyskują dostęp do dokumentów?

• Przepływ dokumentów - jeżeli jest potrzeba przepływu dokumentu między pracownikami, to jak ma on przebiegać?
1.2. Szczegóły DMS

- Utworzenie nowego dokumentu - w jaki sposób dokumenty są tworzone? Jest to pytanie zasadnicze w przypadku tworzenia dokumentów ścisłego zarządzania, gdy są one obsługuwane przez wiele osób.
- Autentyfikacja dokumentu - czy jest sposób na potwierdzenie autentywności dokumentu lub sprawdzenie, ze weryfikacja ta już została przeprowadzona (ang. proofread)?
- Śledzenie zmian - kto, kiedy i jaką czynność wykonał na dokumencie?

Nakreślone zagadnienia obrazują, jak wiele problemów związanych jest z tworzeniem i użyciem systemów DMS i jak bardzo mogą być one złożone. Systemy do zarządzania dokumentami nie są nowym pomysłem. Często od nich zależy poprawne funkcjonowanie firm i biznesu, gdzie tradycyjny obieg dokumentów i ich poszukiwanie nie są możliwe.

1.2. Szczegóły DMS

Systemy typu DMS najczęściej łączą w sobie mechanizmy: magazynowania, wersjonowania, opisu (z wykorzystaniem metadanych, ang. metadata), zabezpieczeń, indeksacji oraz dostępu do danych.

- Metadane - jest to, najprościej mówiąc, opis dokumentu, coś, co pozwala użytkownikowi szybko zorientować się z czym ma do czynienia. Metadane mogą zawierać datę utworzenia dokumentu albo użytkownika, który go stworzył, czy też inne ważne informacje istotne dla danego profilu działalności, który DMS ma wspomagać. System może być tak zbudowany, aby sam selekcjonował potrzebne informacje z pliku (tryb pracy automatyczny), albo aby współpracował z użytkownikiem wprowadzającym pliki do systemu (tryb półautomatyczny).
- Integracja - wiele systemów dostarcza integracji z zewnętrznymi aplikacjami. Użytkownicy mogą wtedy mieć dostęp bezpośredni do danych z repozytorium, pobierać pliki i wprowadzać w nich zmiany, a następnie zapisywać je do repozytorium w nowych wersjach bez opuszczania aplikacji. Taką integrację oferują narzędzia typu Office Suite - czyli zestawy programów biurowych (jak Sun Open Office lub MS Office). Integracje takie najczęściej są dokonywane przez otwarte standardy takie jak ODMA, LDAP, WebDAV, SOAP.
- Przechwytywanie obrazu dokumentu - dokumenty, które zostały zeskanowane powinny zostać poddane obróbce narzędzi optycznego rozpoznawania znaków OCR (ang. Optical Character Recognition). Często spotykane są rozwiązania implementujące system OCR w warstwie sprzętowej (hardware), albo wykorzystujące osobną aplikację komputerową w celu przekształcenia obrazu czytelnego dla człowieka w reprezentację zrozumiałą dla komputera (tekst w postaci cyfrowej).
- Indeksowanie - pozwala przyspieszyć wyszukiwanie dokumentów a tym samym szybszy dostęp do informacji. Indeksowanie można zrealizować wykorzystując prosty mechanizm przechowywania tzw. ścieżek unikalnych identyfikatorów dla grupy dokumentów. Jednak w praktyce można spotkać bardziej złożone formy indeksowania, jak np. klasyfikowanie dokumentów po ich me-
tadanych. Spotykane jest też klasyfikowanie po słowach kluczowych, które zostały wyselekcjonowane z treści dokumentu.

- Przechowywanie informacji - jest sednem systemów do zarządzania dokumentami. Odpowiada za to, gdzie dokumenty są składowane oraz przenoszenie dokumentu do innej części hierarchii. Istnieje w hierarchicznych systemach przechowywania danych (ang. Hierarchical Storage Management, HSM) gdzie dokumenty są dzielone na wysoko oraz nisko kosztowe, w zależności od kosztu przechowywania. Systemy takie istnieją, ponieważ przechowywanie danych na dyskach twardych jest kosztowniejsze niż przechowywanie tej samej informacji na dyskach optycznych, czy taśmach magnetycznych. Często ostateczny sposób przechowywania informacji jest to wynik kompromisu, wyborem/integracją sposobów przechowywania danych na droższych twardych dyskach z szybkim czasem dostępu lub tańszych, albo wolniejszych dysków optycznych, czy też taśm magnetycznych. Systemy HSM kopują dane firmy na wolniejsze urządzenia oraz przekopowują je z powrotem na urządzenia droższe i szybsze, kiedy zachodzi potrzeba dostępu do takiej danej. HMS system „domyśla się”, które dane mają być uznane za bardziej lub mniej znaczące.

- Dystrybucja - opublikowany dokument musi być w formacie, który może być łatwo zmieniany. Powszechną regulowaną prawnie praktyką jest przechowywanie oryginalnej formy dokumentu jako pliku tylko do archiwizacji, który nigdy nie jest udostępniany.

- Bezpieczeństwo - jest kluczowym elementem systemu, powiązany z dystrybucją plików. Oczywiście, nie ma ścisłej określonej definicji, co oznacza bezpieczeństwo informacji. Każda firma ma obowiązek wprowadzać wewnętrzne procedury dotyczące bezpieczeństwa informacji, a systemy DMS mają im podlegać. Niektóre systemy DMS mają wprowadzone systemy uprawnień dla poszczególnych pracowników, albo grup pracowników.

- Obieg dokumentów - jego realizacja jest złożonym problem. Tak zwany WorkFlow wygląda tak, że pracownik po obejrzeniu dokumentu decyduje komu przekazać dalej dany dokument. Systemy regulowe pozwalają administratorowi napisać regułę przejścia danego dokumentu w firmie. Do czynienia z takim obiegiem możemy mieć np. w księgowości, gdzie zanim fakta dotrze do księgowego, musi zostać zatwierdzona przez inną osobę.

- Współpraca - w uproszczeniu chodzi tu o to, aby system pozwalał na dostęp do dokumentu osobie autoryzowanej. Dostęp do takiego dokumentu powinien być zablokowany dla innych pracowników. Innym przykładem współpracy jest współdzielenie danego dokumentu przez grupę osób, które mogą razem dokonywać na nim zmiany.


- Wyszukiwanie - wyszukiwanie plików lub folderów może odbywać się za pomocą wzorców (ang. template attributes), przeglądu zupełnego (ang. full text search) czy też innych mechanizmów.
1.3. Indeksowanie

Indeksowanie to sposób na przyspieszenie wyszukiwania. Indeksowanie od lat z powodzeniem stosowane jest w publikacjach naukowych, książkach biograficznych i innych publikacjach. Indeksy stanowią osobny rozdział w książce, który czytelnik przegląda w pierwszej kolejności, gdy chce znaleźć jakąś konkretną informację, np. definicję niezrozumiałego pojęcia. Lokalizacja tej informacji podana jest w indeksie z dokładnością do strony. Dzięki niej czytelnik może dotrzeć do miejsca z poszukiwanym opisem bez konieczności przeczytania całej książki.

W szczególności indeksować można dokumenty czy też pliki. Indeksowanie plików jest niezbędne, gdy jest ich zbyt dużo, czyli w sytuacji, kiedy ich wyszukiwanie na poziomie percepacji człowieka jest niewykonalne. Wykorzystując pojemność komputerów można indeksować dokumenty według różnych kryteriów. Odpowiedni dobór tych kryteriów jest niezmiernie ważny. Indeksowanie samochodów jedynie wg koloru nadwozia, gdy są one postrzegane w kontekście ogółu pojazdów poruszających się po drogach, mija się z celem. Z kolei usystematyzowane pojazdy zgodnie z szablonem marka/model/rok produkcji/typ paliwa... itd. pozwoliłoby uzyskać dobrą, z punktu widzenia przeszukiwania, bazę danych.

Nie ma jednego dobrego sposobu tworzenia indeksów. Zazwyczaj wszystko zależy od potrzeb i sposobu wykorzystania danych. Podany wcześniej przykład indeksowania samochodów jest słuszny z punktu widzenia pasjonata motoryzacji i pracownika serwisu, ale w ograniczonym zakresie byłby przydatny w policji, gdzie istotny jest numer rejestracyjny pojazdów, numer seryjny nadwozia czy też silnika. Za dobrą praktykę można uznać sytuację, w której określenia profilu indeksu oraz głównych jego założeń podejmuje się użytkownik (czy też grupa użytkowników). Jeśli nie jest on na tyle kompetentny i rozumny w problematyce można próbować ustalić te założenia poprzez wywiad, lub zapoznanie się ze środowiskiem pracy, w którym ma zostać zaimplementowane indeksowanie.

Mówiąc o indeksowaniu danych i systemów temu służących nie sposób ominąć korzyści, jakie dzięki indeksowaniu można osiągnąć. Dzięki indeksowaniu znalezienie informacji, zwłaszcza przy wykorzystaniu komputerów, jest szybkie, precyzyjne i użytkowe. Poprzez udostępnienie mechanizmów wyszukiwania wiele osób może dotrzeć do tej samej informacji, a to pozwala na lepszy ich przepływ w ramach firmy, projektu, społeczności, kraju, czy świata. Mechanizmy te pozwalają uniezależnić pracowników od siebie w kontekście równoległego wykonywania zadań związanych z obiektem informacji/dokumentów/plików.

Aby zrozumieć to zrównoleglenie należy wyobrazić sobie tradycyjny system przechowywania dokumentów, opierający się o składowanie dokumentów na półkach układane wg dat, nazwisk czy też tematów (system biblioteczny). Nowy pracownik, czy też pracownik innego działu poszukując jakiegoś dokumentu
1. Indeksowanie w systemach zarządzania dokumentami

musi być obsłużony przez specjalistę zorientowanego w sposobie organizacji całego zasobu. Po pierwsze dlatego, że poszukujący informacji nie musi wiedzieć o organizacji przechowywania dokumentów, a po drugie by odkładając coś w nieodpowiednie miejsce nie popsuł tym samym misternie tworzonego systemu. Tworzy to tzw. „wąskie gardła” w systemie. Wprowadzenie analogowych indeksów (numerów inwentarzowych) i sortowanie według nich pozwala usunąć te trudności.

Należy zwrócić uwagę, że system indeksowania wspierany komputerowo pozwala wymieniać dokumenty pomiędzy pracownikami bez fizycznego ich przejmieszczania. Publikujący plik dba o to, aby plik został prawidłowo zindeksowany (tą czynność można też prawie całkowicie zautomatyzować), a odnalezienie go przez osobę, która potrzebuje informacji w nim zawartych sprowadza się do wpisania odpowiednich słów kluczowych i/lub parametrów wyszukiwania.

Można zauważyć jak złożonym systemem są systemy do zarządzania dokumentami. Systemy takie są niezwykle interesujące np. w Multiagencjach Ubezpieczeniowych, gdzie najczęściej przechowywane są skany dokumentów i równolegle odpowiadające im dokumenty elektroniczne, z podziałem na druki ścisłego zarządzania, bądź nie. Bez systemów DMS niemożliwe było by prowadzenie działalności większości korporacji.

1.4. Indeksacja czasowo-przestrzenna w repozytoriach danych


1.4.1. Cechy przestrzennych baz danych

Bazy danych używających specjalnych indeksów w celu przyspieszenia wyszukiwania. W tradycyjnych bazach danych indeksy zakładane są na wartościach wybranych, najczęściej wykorzystywanych atrybutów. W przypadku danych przestrzennych indeks jest czymś więcej niż prostą wartością atrybutu, dającą się łatwo uszeregować. Indeksy przestrzenne ujmować muszą wzajemne topologiczne i geometryczne relacje pomiędzy przestrzennymi obiektami. Istnieją różne rodzaje indeksów. Do najbardziej popularnego należy tak zwany R-tree. Obiekty takie jak kształty, linie i punkty są grupowane używając tak zwanego najmniejszego granicznego prostokąta MBR (ang. *minimum bounding rectangle*). Obiekty
W celu zbadania możliwości indeksowania czasowo-przestrzennego dokumentów zaimplementowano prosta aplikację webową, z interaktywnym interfejsem opartym o GoogleMaps API, którą można byłoby zastosować jako rozszerzenie istniejącego już systemu zarządzania dokumentami. Dostarczony mechanizm wyszukiwania pozwalać miał na przeszukiwaniu dokumentów z całego systemu, w poszczególnych regionach, ograniczając ilość plików do pewnego okresu czasu.

1.5.1. Wykorzystane narzędzia

Aplikacja zbudowana została przy wykorzystaniu następujących narzędzi i technologii:

- GoogleMaps API - framework wykorzystany w implementacji interaktywnego komponentu, pozwalającego na definiowanie obszaru wyszukiwania oraz wyświetlanie wyników wyszukiwania na podkładzie mapy.
- PHP5 - język skryptowy wykorzystywany przy tworzeniu logiki aplikacji webowej,
- PostgreSQL - relacyjna baza danych, wykorzystana do implementacji warstwy danych tworzonej aplikacji
- PostGIS - roszszerzenie relacyjno-obiektowej bazy danych PostgreSQL, dodające możliwość zapisywania danych geograficznych wprost do bazy danych i ich przetwarzania
- Pear (ang. PHP Extension and Application Repository) - framework i systemem dystrybucji roszszerzeń do języka PHP.

1.5.2. GoogleMaps API

Google Maps (znane przez pewien czas jako Google Local) to jeden z serwisów wyszukiwarki internetowej Google. Umożliwia wyświetlanie szczegółowych
1. Indeksowanie w systemach zarządzania dokumentami


API (ang. Application Programming Interface) to interfejs programowania aplikacji – specyfikacja procedur lub funkcji umożliwiających komunikację z systemem zewnętrznym w stosunku do aplikacji korzystającej z API. Google Maps API umożliwia korzystanie z mapy takiej jak ta na stronie maps.google.com i większości oferowanych przez nią funkcji na dowolnej stronie internetowej. GoogleMaps API oferuje następujące (podstawowe) funkcje: dodawanie znaczników na mapę, zmiana domyślnych ikon znacznika, wyświetlanie informacji na mapie, obsługa okienek informacyjnych InfoWindow w celu prezentacji treści na mapie, dodawanie wbudowanych kontrolek i definiowanie sterowania mapą, zawarcia - dzięki nim można lepiej kontrolować działanie mapy, wstawianie wieloboków i linii do narysowania granic czy tras dojazdu, obliczenia geograficzne za pomocą wbudowanych metod, wczytywanie danych z pliku XML, geokodowanie adresów.

Najczęściej spotykane rozwiązania, oparte o funkcjonalności GoogleMaps API to:

• zastosowania domowe
  – wyświetlanie odwiedzonych miejsc, tras wycieczek
  – tworzenie galerii zdjęć ze wskazaniem lokalizacji ujęcia
  – prezentowanie ciekawych miejsc geograficznych
  – interaktywne dodatki do blogów

• zastosowania profesjonalne i komercyjne
  – wyświetlanie map dojazdu do firmy
  – śledzenie pozycji obiektów
  – serwisy informacyjne i lokalizatory internetowe
  – bazy obiektów przyrodniczych, geograficznych
  – serwisy typu mash-up

1.6. Opis wykonanej implementacji

1.6.1. Zasada działania programu

Program został zaopatrzony w interaktywny interfejs pokazywany poprzez przeglądarkę internetową. Użytkownik może obserwować, z jakim obszarem czy regionem związane są przechowywane w systemie pliki. Wszystkie pliki które zostały wgrane do systemu mają swoje odpowiedniki w postaci znaczników umieszczonych na mapie dostarczonej przez GoogleMaps API. Pliki te można filtrować za pomocą ustawień wyszukiwarki. Wyszukiwarka pozwala ponadto na zdefiniowanie przedziałów czasowych ważności prezentowanych informacji. Ponieważ każdy plik może być skojarzony z dwoma datami (początkiem i końcem obowiązywania na danym obszarze), uzyskuje się dzięki temu filtrację czasową. Pliki
1.6. Opis wykonanej implementacji

można też wyszukiwać zadając prostokątem ograniczającym obszar obowiązywania, wskazując na konkretne miejsce lub określając promieniowanie bufora wokół wskazanego miejsca (wyrażony w kilometrach). Okno główne programu przedstawiono na rys. 1.1.

1.6.2. Dodawanie plików do bazy

System zasilany jest poprzez wprowadzenie zestawu metadanych dla każdego pliku. Aby dodać plik do systemu należy: określić nazwę pliku i ścieżkę dostępu do niego, wskazać miejsce na mapie odpowiadające zawartości pliku, dołączyć opis oraz dodatkowe informacje, które mają stanowić metadane pliku.

Następnie należy podać okres ważności odpowiadający zawartości pliku. Okno służące do wprowadzania danych do systemu przedstawiono na rys. 1.2. Pliki zapisywane są w systemie plików, a w bazie danych - informacje o oryginalnej nazwie pliku. Należy wspomnieć, że może to być tabela obcego DMS, który mógłby korzystać z opisanego w niniejszej pracy modułu do wyszukiwania czasowo-przestrzennego.

Na rys. 1.3 przedstawiono budowę bazy SQL. Pobieranie lub wyświetlanie plików jest przejrzyste, ponieważ znając ID pliku można pobrać go z dysku za pomocą oryginalnej nazwy.

1. Indeksowanie w systemach zarządzania dokumentami

Do przechowywania danych geograficznych jest używany moduł PostgreSQL. PostGIS zapewnia szereg udogodnień w wyszukiwaniu, czy składowaniu danych takich jak w naszym projekcie. Możliwe jest obliczanie odległości na sferze jaką jest planeta.

Stworzony projekt może być używany z innymi programami takimi jak systemu DMS. W założeniu projekt, może być skorelowany z zewnętrznym systemem w warstwie aplikacyjnej. Korelacja taka musi być np. przy usuwaniu pliku. Należy wtedy usunąć informacje o tym pliku zarówno z bazy danych projekto-wego programu, jak i z zewnętrznego systemu DMS.
1.6. Opis wykonanej implementacji

Rys. 1.3: Schemat bazy SQL.

*dms_domain_links*
+id: integer not null default nextval('dms_domain_links_id_seq': regclass)
+dms_semantic_id: integer
+dms_space_id: integer
+domain_start_at: timestamp
+domain_end_at: timestamp
+created_at: timestamp
+updated_at: timestamp

*Tabele POSTGIS*

*geometry_columns*
+f_table_catalog: character varying(256)
+f_table_schema: character varying(256)
+f_table_name: character varying(256)
+C_geometry_column: character varying(256)
+coord_dimension: integer
+sr_id: integer
+type: character varying(30)

*spatial_ref_sys*
+sr_id: integer
+auth_name: character varying(256)
+auth_srid: integer
+srtext: character varying(256)
+proj4text: character varying(256)

*dms_files*
+id: integer not null default nextval('dms_files_id_seq': regclass)
+file_name: character varying(512)
+created_at: timestamp

*dms_semantics*
+id: integer not null default nextval('dms_semantics_id_seq': regclass)
+file_id: integer
+name: character varying(512)
+description: character varying(2048)
+additional_info: character varying(2048)
+created_at: timestamp
+updated_at: timestamp

*dms_spaces*
+id: integer not null default nextval('dms_spaces_id_seq': regclass)
+name: character varying(30)
+created_at: timestamp
+file_point: geometry
+file_polygon: geometry
+file_id: integer
Elektroniczne systemy podawcze do obsługi konferencji i wydawnictw

M. Mielnicki, K. Bubiński

W dzisiejszym świecie systemy komputerowe pełnią rolę narzędzi wspierających wykonanie różnych prac przez ludzi. Dzięki nim oraz zautomatyzowanym metodom przetwarzania danych poprawiła się znacznie sprawność i szybkość z jaką można realizować nawet skomplikowane zadania.

Elektroniczne systemy podawcze do obsługi konferencji i wydawnictw są przykładami praktycznych przypadków użycia systemów komputerowych. Dzięki nim autorzy mogą zdalnie zgłaszać swoje artykuły oraz obserwować status obsługi własnych zgłoszeń, zaś organizatorzy konferencji bądź wydawcy mogą przenieść część wykonywania wszelkich dających się zautomatyzować na barki komputerów. Niniejszy rozdział jest poświęcony na przedstawienie zagadnień związanych z budową takich systemów.

2.1. Funkcje i zastosowania

Głównym zadaniem omawianych systemów jest wspieranie wymiany danych, komunikacji pomiędzy użytkownikami systemu oraz przechowywania informacji. Są one stosowane przez wydawnictwa internetowe, portale e-learningowe i organizatorów konferencji [3,1]. Mogą także znaleźć zastosowanie m.in. przy budowie portali społecznościowych, zarządzaniu projektami wieloosobowymi, prowadzeniu szkół internetowych, czyli wszędzie tam, gdzie wymagana jest zdalna wymiana informacji, uporządkowanie danych, dzielenie ich na kategorie. Przykłady działających systemów wymieniono poniżej.

2.1.1. Przykłady

Jednym z przykładów sprawnie działającego systemu podawczego jest system o interfejsie użytkownika udostępnionym pod adresem [http://unibook.com](http://unibook.com). Interfejs ten to interaktywna strona sieci Web z ofertą wydawnictwa Unibook. Za
2.1. Funkcje i zastosowania

pośrednictwem udostępnionego na niej kreatora każdy użytkownik może spróbować opublikować własną pracę.

Publikacji własnej pracy odbywa się zaledwie w kilku krokach. Na początku należy podać tytuł publikacji oraz dane o autorze i jego biografii (wszystko z kilkukuzdaniowym opisem). Ważne jest także, aby wybrać kategorię, język oraz poziom dostępności publikacji (dla wszystkich, lub tylko dla osób posiadających odpowiednie hasło). Następnie należy wybrać plik do wysłania oraz ustawić kilka opcji związanych z formą wydruku publikacji (format, czcionka, okładka, papier itp.). Gdy użytkownik dopełni wszystkich formalności musi zalogować się w celu weryfikacji autentyczności wysłanej publikacji. Następnie, po upływie około tygodnia, przychodzi odpowiedź od recenzenta. Określa on autentyczność pracy oraz sensowność (także opłacalność) inwestycji w wysłane pracę.

Wiele wydawnictw określa także specyficzny format, w którym powinny być przygotowane dostarczane dokumenty. Jest to zwykle plik pdf lub doc. Układ pliku często można spotkać na stronie internetowej wydawnictwa skąd można go pobierać i wykorzystać do napisania własnej publikacji.

Elektroniczne systemy do obsługi konferencji mają wspierać organizatorów w zadaniach związanych z obsługą konferencji. Systemy te są podobne w budowie. Zwykle wyróżnia się kilka rodzajów ich użytkowników: administratorzy, prelegenci, recenzenci, słuchacze.

• Administratorzy - mają za zadanie zarządzanie działaniem systemu. Nadzorują poprawne działanie systemu, zarządzają kontami użytkowników, działami, sekcjami w systemie.
• Prelegenci - nadsyłają swoje artykuły przed końcowowym terminem nadsyłania prac.
• Recenzenci - dokonują oceny przesłanego artykułu oraz podejmują decyzje o akceptacji bądź odrzuceniu. W przypadku negatywnej opinii prelegent jest zobowiązany do poprawienia dokumentu oraz ponownego wprowadzenia do systemu. Pozytywna opinia pracy wiąże się z jednoczesnym dopuszczeniem jej do wygłoszenia.
• Słuchacze - są zainteresowani wysłuchaniem referatów oraz zapoznaniem się z treścią artykułów.

2. Elektroniczne systemy podawcze do obsługi konferencji i wydawnictw

słuchacze mogą wpisać swoją opinię na temat prezentacji, czy np. zagłosować na najlepszą prezentację.


Jako kolejny przykład można przestawić Elektroniczny Urząd Podawczy ZUS. Jest to urząd udostępniający publiczne środki komunikacji elektronicznej, służące do przekazywania informacji w formie elektronicznej do podmiotu publicznego przy wykorzystaniu powszechnie dostępnej sieci teleinformatycznej. Głownym celem działania urzędu jest przekazywanie dokumentów ubezpieczeniowych. Przy pomocy tego systemu można złożyć poniższe dokumenty:

1. **ZUS - EWN** - wniosek płatnika składek o wydanie zaświadczenia o niezaleganiu w opłacaniu składek,
2. **ZUS - EZS** - wniosek płatnika składek o zwrot nadpłaconych składek,
3. **ZUS - ERU** - zgłoszenie reklamacji do informacji o stanie konta osoby ubezpieczonej,
4. **ZUS - EWZ** - wniosek ubezpieczonego o wydanie zaświadczenia o zgłoszeniu do ubezpieczenia zdrowotnego,
5. **ZUS - EPW** - wniosek o ustalenie przekroczenia rocznej granicy podstawy wymiaru składek (30-krotność),
6. **ZUS - EWP** - wniosek płatnika składek o udostępnienie programu Płatnik.

2.2. Podstawy implementacji

dowód osobisty lub paszport. Trzeba także podać dane kontaktowe urzędu odbiorczego oraz podać cel składania podania. Po wprowadzeniu wszystkich danych należy poddać je weryfikacji klikając na odpowiedni przycisk.

2.2. Podstawy implementacji


2.2.1. Moodle

Moodle (ang. Modular Object-Oriented Dynamic Learning Environment, [www.moodle.org]) jest to platforma e-learningowa o otwartym kodzie (dostępna na licencji GNU GPL), przygotowana do współpracy z większością systemów operacyjnych (Linux, MS Windows, Mac OS X, NetWare 6). Platforma ta jest napisana w języku PHP, może korzystać z jednego z dwóch serwerów baz danych (MySQL lub PostgreSQL), oraz wymaga do działania serwer HTTP Apache.

System Moodle najczęściej jest wykorzystywany do nauki przez internet lub do wspomagania tradycyjnych metod nauczania [2]. Po zainstalowaniu i uruchomieniu w systemie znajduje się 7 różnych typów użytkowników:

• administrator - osoba odpowiedzialna za system, mająca dostęp do wszystkich możliwych opcji
• course creator - osoba tworząca kursy i prowadząca je
• teacher - osoba prowadząca kurs, mogąca nadawać kursowi daną formę oraz oceniać uczestników kursu
• non-editing teacher - osoba prowadząca kursy według wcześniej przygotowanej formy, mogąca jedynie oceniać swoich studentów
• student - osoba mogąca zapisywać się na kursy i brać udział w związanych z nimi dyskusjach
• guest - osoba odwiedzająca system, mogąca zobaczyć główną stronę systemu, ale nie mogąca zapisywać się na kursy ani ich komentować.

System Moodle jest bardzo elastyczny - pozwala na zmianę uprawnień każdego z typów użytkowników, dodatkowo można w nim na bieżąco wprowadzać własne typy użytkowników w zależności od występujących potrzeb.

Platforma Moodle pozwala tworzyć różnego rodzaju kursy, które mogą być podzielone na kategorie oraz posiadać swoje "podkursy". Tworząc kurs można określić:

• numer ID kursu,
2. Elektroniczne systemy podawcze do obsługi konferencji i wydawnictw

- format kursu (czy jest to kurs składający się z regularnie odbywających się spotkań, czy może ma charakter bardziej towarzyski, gdzie czas spotkań nie jest ściśle określony),
- długość trwania kursu,
- datę rozpoczęcia kursu,
- datę początkową i końcową określającą przedział czasu, w którym można się na dany kurs zapisywać,
- podział uczestników na grupy,
- czy dane grupy widzą na wzajem swoje fora,
- klucz dostępu do kursu, gdy porządkowane jest, aby tylko niektórzy użytkownicy mogli się na niego zapisać (klucz jest wysyłany tym osobom na maila, podanego przy rejestracji do systemu),
- język, w którym prowadzony będzie kurs
- nauczycieli prowadzących oraz osoby opiekujące się kursem.

2.2.2. Django


- automatycznie generowany i kompletny panel administracyjny, z możliwością dalszego dostosowywania,
- przyjazne adresy dokumentów z możliwością dowolnego ich kształtowania,
- prosty lecz funkcjonalny system szablonów czytelny zarówno dla grafików jak i dla programistów,
- oddzielenie logiki aplikacji (widok) logiki biznesowej (model) wyglądu (szablony) oraz baz danych,
- wsparcie dla wielojęzycznych aplikacji,
- bardzo duża skalowalność i wydajność pod obciążeniem,
- wydajne systemy cache’owania, obsługa memcached,
- własne, prosty serwer do testowania aplikacji,
- współpraca z Apache poprzez WSGI (domyślnie) i mod_python oraz z innymi serwerami poprzez protokoły FastCGI i SCGI,
- DRY czyli zasada „nie powtarzaj się” w odniesieniu do tworzenia aplikacji, (np. strukturę bazy danych Django generuje ze zwykłych klas Pythona),
- posiada ORM wysokiego poziomu pozwalający na łatwe i bezpieczne operowania na bazach danych bez użycia SQL,
- obsługuje następujące bazy danych: PostgreSQL, MySQL, SQLite oraz Oracle,
- rozpowszechniony jest na liberalnej licencji BSD.
2.2.3. MySQL

MySQL jest wolnodostępnym systemem zarządzania relacyjnymi bazami danych stworzonym przez szwedzką firmę MySQL AB\(^1\). MySQL obsługuje obecnie większość standardu ANSI/ISO SQL (tj. SQL:2003). Wprowadza także swoje rozszerzenia i nowe elementy języka, którymi są m.in. procedury składowane, wizualizacje, perspektywy, kursory, harmonogram zadań, partykjonowanie tabel. MySQL cieszy się opinią jednego z szybszych serwerów bazodanowych, dzięki czemu nadaje się jako serwer dla często odwiedzanych witryn WWW. MySQL oferuje także różne typy mechanizmów bazodanowych, z których każdy typ przeznaczony jest do innego zastosowania. Są to m.in.:

- **MyISAM** - domyślny mechanizm, nie obsługuje transakcji ani kluczy, umożliwia natomiast wyszukiwanie pełnotekstowe,
- **MEMORY** - najszybszy, gdyż wszystko jest przechowywane wyłącznie w pamięci RAM. Ma jednak kilka ograniczeń, między innymi nie przechowuje danych po wyłączeniu serwera MySQL,
- **CSV** - przechowuje dane w standardowych plikach CSV.
- **ARCHIVE** - przechowuje dane w spakowanych archiwach. Umożliwia wyłącznie dodawanie i pobieranie rekordów.

2.2.4. Apache

Apache jest obecnie najczęściej wykorzystywanym serwerem WWW. Współpracuje on m.in. z interpreterem języka PHP oraz serwerem bazy danych MySQL. Cechuje się on wysokim poziomem bezpieczeństwa działania poprzez obsługę protokołów transmisyjnych http, https, ftp, ftps oraz ssl.

2.3. Prototyp systemu

W celu przetestowania technik i metod wykorzystywanych w systemach podawczych stworzono prototyp systemu służącego do wysyłania prac przez studentów. System ten korzysta z serwera bazy danych, którego zadaniem jest przechowywanie wysłanych plików. Użytkownicy mogą wysłać do systemu sprawdzać terminy nadsyłania prac oraz ich ocenę. System został zaimplementowany jako aplikacja webowa, dlatego wymaga on od użytkownika dostępu do internetu i zainstalowanej przeglądarki internetowej. Dodatkowo, aby móc korzystać z systemu, każdy użytkownik powinien być w nim zarejestrowany (tj. posiadać założone konto).

2.3.1. Instalacja w Ubuntu 9.10

Poniżej zamieszczono opis instalacji zaimplementowanego prototypu na systemie Ubuntu 9.10. Ponieważ prototyp korzysta z zewnętrznych narzędzi (baz

\(^1\)MySQL AB została kupiona 16 stycznia 2008 roku przez Sun Microsystems, a ten 27 stycznia 2010 roku przez Oracle, przyp. red.
2. Elektroniczne systemy podawcze do obsługi konferencji i wydawnictw
danych), jego instalacja powinna rozpocząć się od instalacji tych narzędzi. Naj-pierw powinien zostać zainstalowany serwer bazy danych MySQL. W tym celu wy-starczy wydać polecenie:

```
sudo apt-get install mysql-server php5-mysql
```

Po zainstalowaniu serwera bazy danych należy dokonać pewnych zabiegów konfiguracyjnych. Należy zalogować się na konto administratora serwera bazy da-
nych

```
mysql -u root password NewRootDatabasePassword
```
a następnie utworzyć tabele, które wykorzystane będą przez Moodle:

```
CREATE DATABASE moodle
DEFAULT CHARACTER SET utf8 COLLATE utf8_unicode_ci;
GRANT ALL PRIVILEGES ON moodle.* TO moodleuser@localhost IDENTIFIED BY 'NewMoodleDatabasePassword';
GRANT SELECT,LOCK TABLES on moodle.* TO moodlebackup@localhost IDENTIFIED BY 'MoodleBackupPassword';
FLUSH PRIVILEGES;
QUIT
```

Kolejnym krokiem jest instalacja serwera Apache wykorzystując polecenie apt-get:

```
sudo apt-get install apache2 libapache2-mod-php5 php5-gd
sudo apt-get install libapache2-mod-security php5-ldap php5-odbc
```

Po zainstalowaniu należy zrestartować serwer poleceniem:

```
sudo /etc/init.d/apache2 restart
```

Potem powinno zostać zainstalowane dodatkowe oprogramowanie, takie jak pro-
gram antywirusowy, tłumacz, zip, itp.:

```
sudo apt-get install openssh-server unattended-upgrades
sudo apt-get install unzip zip aspell-en aspell-fr aspell-de
sudo apt-get install curl php5-curl php5-xmlrpc
sudo apt-get install clamav-base clamav-freshclam clamav
```

Kolejnym krokiem jest ściągnięcie pakietu Moodle i rozpakowaniu go:

```
cd /var/www
sudo tar -zxf moodle-latest-19.tgz
sudo mkdir /var/moodledata
sudo chown -R www-data:www-data /var/moodledata
```
2.3. Prototyp systemu

Następnie należy zmienić domyślny katalog serwera www. W tym celu należy edytować plik konfiguracyjny:

```
sudo vim /etc/apache2/sites-available/default
```
i zmienić linię:

```
DocumentRoot /var/www
```
na

```
DocumentRoot /var/www/moodle/
```
oraz zrestartować serwer apache.

Mając tak przygotowane środowisko można przystąpić do pracy, wpisując w polu wprowadzania adresu przeglądarki internetowej adres `localhost`. W wyniku przejścia do tego adresu w oknie przeglądarki pojawi się strona www, za pomocą której w szybki sposób będzie można stworzyć własny portal wymiany plików.

2.3.2. Część niefunkcjonalna

Założenia niefunkcjonalne:

- interfejs użytkownika systemu może być uruchomiony przy dowolnej rozdzielczości ekranu,
- wszystkie strony serwisu cechują się prawidłowym wyświetlanie w najbardziej znanych przeglądarkach internetowych, takich jak: Firefox, Opera, Internet Explorer,
- warstwa danych realizowana jest za pomocą serwera baz danych MySQL,
- aplikacja wdrażana jest na serwerze HTTP Apache,
- zarządzanie treścią serwisu jest możliwe po zalogowaniu się do Panelu Administracyjnego,
- serwis jest napisany w taki sposób, aby spełniać wszystkie wymagania wyznaczone przez World Wide Web Consortium (W3C),
- maksymalny rozmiar przesyłanych w systemie pojedynczych plików to 2MB.

Makietę wyglądu okna po zalogowaniu się do systemu przedstawiono na rys. 2.1

2.3.3. Część funkcjonalna

Typy użytkowników

Założono, że prototyp obsługiwał będzie użytkowników czterech typów:

- Administrator - osoba, która sprawuje nadzór nad działaniem całego portalu. Ma ona możliwość zmiany wszystkich ustawień.
- Nauczyciel - osoba, która może tworzyć kursy oraz dodawać materiały i listy zadań do nich. Ma także możliwość wyznaczania terminów końcowych na dostarczenie zadań przez studentów. Do jej uprawnień należy także ustalenie sposobu oceniania studentów na danym kursie, zasady wyliczania średniej, jak
2. Elektroniczne systemy podawcze do obsługi konferencji i wydawnictw

Powyżej przedstawiono makiety okien po załogowaniu się do systemu.

- Nagłówek
- Menu lewe
- Menu prawe
- Zasoby użytkownika

Również samo wystawianie ocen. Nauczyciel tworząc kursy może ograniczyć część Studentów możliwość zapisywania się na nie.

- Student - osoba, która ma możliwość zapisywania się na kursy, przeglądania materiałów i zadań dostarczonych przez prowadzącego. Może załączać pliki z rozwiązaniami zadań oraz widzi zbliżający się deadline danego zadania. System umożliwia też Studentom wgląd do własnych ocen, oraz komunikację z prowadzącym.

- Gość - nowo zarejestrowana osoba, która czeka na przyjęcie do grona Studentów przez Nauczyciela bądź Administratorkę.

Funkcjonowanie systemu od strony nowego użytkownika

Po wpisaniu adresu portalu w przeglądarce internetowej i jego wybraniu otwiera się strona startowa portalu. Jest ona dostępna w dwóch językach: polskim i angielskim (rys. 2.2).

Rys. 2.1: Makieta wyglądu okna po załogowaniu się do systemu: 1) nagłówek, 2) część merytoryczna, 3) lewe menu, 4) prawe menu.

Rys. 2.2: Strona startowa portalu.
2.3. Prototyp systemu

Po lewej stronie znajdują się kontrolki umożliwiające zalogowanie się do systemu z nazwą użytkownika oraz hasłem. Poniżej mieści się opcja pomocy w logowaniu, użyteczna dla osób które zapomniały swojego hasła bądź nazwy użytkownika. Po wybraniu tej opcji następuje przejście do strony, gdzie podając adres e-mail użyty podczas rejestracji użytkownik ma możliwość uzyskania informacji o przypisanej mu nazwie oraz hasle (zostaną one wysłane na zarejestrowany adres e-mail).

Po prawej stronie znajduje się informacja dla osób, które po raz pierwszy korzystają z tej aplikacji. Dzięki nim można przejść do strony rejestracji w portalu. Proces rejestracji wymaga wpisania kilku podstawowych informacji o użytkowniku. Tworzone konto na początku jest nieaktywne. Aby aktywować konto wymagane jest odebranie wiadomości e-mail wysłanej automatycznie przez portal i kliknięcie na link tam zawarty. Po wykonaniu tej czynności konto użytkownika zostanie aktywowane.

Portal jest skonstruowany w ten sposób, iż konto po aktywacji przyjmuje status Gość, który nie ma dostępu do żadnych informacji zawartych w portalu. Czeka on na nadanie mu statusu Studenta przez Nauczyciela bądź Administratora.


Rys. 2.3: Strona główna.
2. Elektroniczne systemy podawcze do obsługi konferencji i wydawnictw

W ten sposób uzyskuje się szybki dostęp do materiałów i miejsc, w których możliwe jest dostarczenie sprawozdań. Jeżeli wybrany zostanie któryś z kursów widniejących na środku ekranu, na ekranie zostaje przedstawiony opis tego kursu oraz jego podkursów. Klikając dalej można dojść do strony poświęconej danemu kursowi/podkursowi (rys. 2.4). Jak widać, Student ma wgląd do wszystkich zagadnień związanych z danym kursem (takich jak np. lista uczestników danego kursu, forum dyskusyjne, zadania, zasoby materiałów do kursu, oceny, możliwość zadawania pytań prowadzącemu). Po prawej stronie widoczna jest tabela z najbliższymi terminami związanymi z danym kursem oraz raporty ostatnich aktywności.

Podstawową funkcją systemu jest zbieranie nadsyłanych prac. Każdy zalogowany użytkownik ma możliwość wysłania na serwer swojego pliku będącego sprawozdaniem, raportem, czy inną formą wymaganą przez prowadzącego zajęcia. Aby wysłać plik uprzednio prowadzący zajęcia musi dodać do rubryki z terminami nadsyłanych prac przez użytkownika nową pozycję oraz podać ostateczny termin jej nadesłania. Dopiero wtedy użytkownik ma możliwość wysłania nowego pliku. Nadesłane prace można aktualizować do czasu aż minie ostateczny termin. Ogólny schemat przepływu wiadomości oraz przesyłania plików przedstawiono na rys. 2.5.

2.4. Uwagi końcowe

Zrealizowany system, oparty o platformę e-learningową moodle, działa w pełni poprawnie. Realizuje on wszystkie zamierzone w celach projektowych założenia, czyli:

- bezpieczne logowanie (poprzez HTTP lub HTTPS),
- możliwość przesyłania plików między prowadzącym a studentami z określeniem terminów (końcowych).
2.4. Uwagi końcowe

Rys. 2.5: Podstawowy schemat przepływu informacji.

Ilość możliwości, które stwarza moodle jest tak duża, że aby skonfigurować system według własnych upodobań trzeba na to poświęcić naprawdę sporo czasu. Plusem jest fakt, że system jest domyślnie skonfigurowany i jeżeli godzimy się na standardowe ustawienia, to postawienie tego typu platformy edukacyjnej nie powinno stanowić problemu.

Spoglądając na stworzoną aplikację od strony użytkownika, trzeba przyznać, że do szybkiej jej obsługi wymagana jest odrobnina doświadczenia, przyzwyczajenia. Według autorów opracowania nie jest to wada aplikacji, gdyż każdy tego typu portal posiadający wiele funkcjonalności, wiele zakładek, na pierwszy rzut oka może trochę przytłaczać. Zapewne po kilkukrotnym skorzystaniu z portalu stworzonego przy pomocy moodle użytkownik poczuje się w nim „jak w domu” i doceni wszystkie jego dodatkowe możliwości.

Literatura

3.1. Wstęp

W dobie powszechnej komputeryzacji oraz szerokim dostępie do internetu bardzo popularne stały się portale społecznościowe. W niniejszym rozdziale zostaną omówione zagadnienia związane z budową takich portali. Wyjaśnione zostanie, czym jest portal społecznościowy, czym się charakteryzuje, jak się go tworzy. W końcowej części rozdziału przedstawiony zostanie krótki opis budowy portalu w środowisku Joomla!

3.1.1. Istota portali społecznościowych

By właściwie poruszać się po tematyce portali społecznościowych należy zdefiniować kilka pojęć. W pierwszej kolejności należy zdefiniować pojęcie samego Internetu, a następnie społeczności internetowej oraz serwisu społecznościowego.

Definicja 3.1.1 Internet - połączone ze sobą sieci oparte na protokole TCP/IP, które używa i rozwija społeczność oraz zbiór zasobów, które znajdują się w sieci.

Definicja 3.1.2 Społeczność internetowa - zbiorowość ludzka, w której interakcje odbywają się za pośrednictwem Internetu.

Definicja 3.1.3 Serwisy społecznościowe - rodzaj społeczności internetowej zgrupowanych w konkretnym serwisie internetowym, którego użytkownicy zaspokajają swoją potrzebę kontaktów z innymi ludźmi poprzez wymianę informacji, doświadczeń i zainteresowań.

Człowiek jest istotą społeczną. Potrzebuje społeczeństwa aby poprawnie się rozwijać. W pierwszym etapie życia osobami, które uczą, są rodzice, później koledzy czy koleżanki w szkole, nauczyciele. W kolejnych etapach życia człowiek buduje swój światopogląd opierając się o opinie innych. Potrzebuje społeczeństwa, żeby określić swoje w nim miejsce. Nawet samotnik potrzebuje mieć kogoś,
3.1. Wstęp

od kogo może się odizolować i dzięki temu nazwać samotnikiem. Tak jak na danego człowieka wpływają inni ludzie, tak on sam wpływa na innych. Wszystko wiąże się z pewnymi potrzebami, które dzięki technologii Web 2.0, mogą być realizowane w Internecie:

1. potrzeba przynależności do grupy;
2. potrzeba samorealizacji;
3. potrzeba kontaktu z innymi ludźmi;
4. potrzeba bycia rozpoznawanym, wyróżnionym;
5. potrzeba bycia pięknym;
6. wspólne zainteresowania lub hobby;
7. występowanie wspólnych cech demograficznych;
8. istnienie elementu łączącego, np: konkretny produkt;


3.1.2. Etapy rozwoju Internetu

W historii rozwoju Internetu można wyróżnić trzy etapy, które często identyfikowane są z technologiami (a może raczej paradygmatami) tworzenia stron internetowych. Pierwszy z nich (Definicja 3.1.4) dotyczy klasycznych stron, gdzie istniał jeden lub kilku administratorów, a użytkownicy mogli jedynie im zgłaszać wszelkie uwagi dotyczące treści strony.

**Definicja 3.1.4** Web 1.0 - technologia tworzenia stron internetowych, w której można odróżnić twórcę (twórców) strony od użytkowników.

Drugi, obecnie istniejący (Definicja 3.1.5), bazuje na technologii umożliwiających tworzenie interaktywnych stron z forami dyskusyjnymi, blogami, multimediami itp.

**Definicja 3.1.5** Web 2.0 - technologia tworzenia serwisów internetowych, gdzie każdy użytkownik ma wpływ na treści w nim zamieszczane.
3. Portale społecznościowe

Praktycznie każdy może stworzyć sobie witrynę internetową w technologii Web 2.0. Jednak aby ta strona nosiła miano zgodnej z Web 2.0 musi jeszcze wystąpić kilka elementów zupełnie niezależnych od jej twórców. Cała idea wymienionej technologii polega na tym, że najistotniejszy głos w tworzeniu danej witryny mają użytkownicy. Większa część informacji umieszczona na stronie musi pochodzić właśnie od nich. To oni swoimi wpisami, komentarzami czy zamieszczanymi treścią kształtują wizerunek witryny. Możliwości właściciela strony ograniczone są do moderacji i administracji. Dlatego też, kiedy jest mowa o portalu stworzonym w technologii Web 2.0, pojawiają się takie elementy, jak:

1. ocena portalu;
2. komentarze go dotyczące;
3. rekomendacje na jego temat przekazane znajomym;
4. opisy współdzielone przez serwisy typu: Wykop, Gwar;
5. otagowanie (na innych forach, stronach, portalach muszą zostać umieszczone odnośniki do tego portalu).

Od jakiegoś już czasu w środowisku informatycznym pojawia się termin: technologia Web 3.0. Każda zmiana numeru technologii wiązała się z konkretną zmianą podejścia w tworzeniu witryn. W tym przypadku ma być to przeskok w sferze interpretacji zgromadzonych i udostępnionych danych. Mówi się, że w niedalekiej przyszłości Internet będzie umożliwiał nie tylko wyszukiwanie informacji po odrębnych wyrazach, ale po pełnych zdaniach napisanych w języku naturalnym. Wyjaśnia to następujący przykład poszukiwania odpowiedzi na pytanie o ilość posłów w sejmie. W obecnych wyszukiwarkach prawdopodobnie wiemy o tym co wpisałby: „sejm ilość posłów”. Przy nowym podejściu ma być możliwe zadanie pytania: „ile posłów pracuje w sejmie?”

Definicja 3.1.6 Web 3.0 - technologia przechowywania i współdzielenie informacji w sposób umożliwiający ich interpretację logiczną z wykorzystaniem sztucznej inteligencji.

3.2. Narzędzia programowe do budowy portali

3.2.1. LAMP

Akronim LAMP został ukuty przez Michael'a Kunze'a w 1998 roku i odnosił się do metody tworzenia portali, wykorzystującej następujące narzędzia:

• Linux – rodzina unixopodobnych systemów operacyjnych, opartych o jądro Linux,
• Apache – serwer http, pozwalający na wyświetlanie stron w internecie,
• MySQL – system zarządzania relacyjnymi bazami danych,
• PHP – skryptowy, obiektowy język programowania, zaprojektowany do tworzenia stron internetowych w czasie rzeczywistym.

Środowiska te są programami typu open-source, dzięki czemu nie należy martwić się o legalność oprogramowania oraz system aktualizacji, który w świecie Open
3.2. Narzędzia programowe do budowy portali

Source działa w sposób sprawny. Dodatkowym atutem jest częstość wydawania nowych wersji programów.


3.2.2. Narzędzia zarządzania bazami danych

Po wprowadzeniu LAMP, pojawienie się aplikacji internetowych służących do tworzenia forów dyskusyjnych, takich jak phpBB lub phpMyAdmin, było tylko kwestią czasu. Od momentu powstania phpBB i phpMyAdmin dosłownie każdy, mający dostęp do miejsca na serwerze, mógł stworzyć własne forum dyskusyjne. Internet zaroślił się od takich zbiorowości, które częstokroćcecechowały się wąską tematyką prowadzonych dyskusji, bądź materiałów.

Przez długi okres czasu, phpBB i phpMyAdmin były wystarczającymi narzędziami dla odbiorców. Doczekały się różnych wersji i odmian, przy czym cały czas były nastawione na proste konfigurację i administrację. Jednakże często forum dyskusyjne nie wystarczało, dlatego też zaczęto poszukiwać nowych rozwiązań. Do budowania takich aplikacji wykorzystywano język skryptowy PHP. Jest to specyficzny język, gdyż program nie jest kompilowany do kodu maszynowego, a wykonywany przez interpretera PHP. Dzięki temu uzyskano przenosność kodu i jednocześnie odciążono procesor klienta, gdyż sam PHP pracuje po stronie serwera.

3.2.3. CMS

CMS to aplikacja internetowa, która pozwala na łatwe tworzenie stron WWW oraz późniejsze, łatwe ich administrowanie, nawet przez osoby nie będące biegłymi w technikach komputerowych. Główna idea CMS to oddzielenie zawartości (treści) strony od jej szaty graficznej. Do określania wyglądu strony korzysta się z szablonów takich jak np. CSS.

Definicja 3.2.1 Kaskadowe Arkusze Stylów (ang. Cascading Style Sheets, CSS) to język opisu wizualnego aspektu strony internetowej. Opis taki to lista dyrektyw, ustalających jak strona ma być wyświetlana w przeglądarce. Reguły te pozwalają na wybór rodziny czcionek, koloru tekstu, wielkości marginesów itp.

W strukturze samego systemu CMS można wyróżnić kilka elementów:

1. **Front i Back End** – jest to, odpowiednio, część widoczne dla gości i zalogowanych użytkowników oraz warstwa administracyjna strony (konfiguracja, konserwacja, czyszczenie, tworzenie statystyk oraz przygotowanie zawartości).
3. Portale społecznościowe

2. **Ustawienia Konfiguracyjne** – ustawienia dla całej strony www, t.j. tytułowy tekst okna przeglądarki, hasła-klucze dla przeglądarek, przełączniki wyłączające i włączające stronę itp.,

3. **Prawa Dostępu** – czyli takie zarządzanie systemem CMS, gdzie użytkownicy mają własne konta i różne prawa dostępu do treści,

4. **Zawartość** – wszelkie dane (tekst, muzyka, film), które są zawarte na stronie. Najczęściej umieszcza się takie dane w różnych strukturach, aby łatwiej nimi zarządzać,

5. **Szablon** – zbiór reguł mówiących o wyglądzie strony,

6. **Rozszerzenia** – możliwość rozbudowy strony o dodatkowe funkcjonalności,

7. **Przepływ Pracy** – metody pracy, które określają zakres działalności i przepływ informacji pomiędzy zarządzającymi stroną.

3.2.4. **Joomla!**

Wracając do LAMPa i skryptów phpMyAdmin i phpBB, narzędzia te szybko przestały wystarczać. Dało się wyczuć pewien niedosyt związany z możliwościami tworzonych serwisów i ich administrowania. phpBB pomagał tworzyć fora dyskusyjne, jednak często brakowało takim witrynom funkcjonalności znanych z normalnych stron internetowych. Właściwe powody popchnęły programistów w kierunku poszukiwania nowej jakości.


Mambo Foundation zdało sobie sprawę z faktu, że utraciło zespół twórców oraz przychylność i zainteresowanie świata Open Source. Sytuacja stała się tak napięta, że fora dyskusyjne zapełniły się wyzwiskami pod kierunkiem wszystkich stron konfliktu: Miro, Fundacji Mambo i zwykłych użytkowników. 26 sierpnia Mambo Foundation wypuściło kolejną wersję Mambo, która okazała się gwoździem do trumny i spotkała się z nieprzychylną reakcją świata Open Source.

Szybko naprawiono błędy i 1 września ten sam zespół wypuścił nowy system CMS - Joomla!. Sama nazwa w języku suahili oznacza Żazemi!”. System spotkał
3.2. Narzędzia programowe do budowy portali

się z dużym uznaniem użytkowników i prawie natychmiast na forum Joomla! zarejestrowało się 8 tys. użytkowników.

Cały spór jest nieco niejasny i zawiły. Wiadomo natomiast, że dzięki tej sytuacji narodziła się nowa jakość.

Budowa

Joomla! charakteryzuje się tym, że większość informacji przekazywanych na stronie, tworzy się jako artykuły. Artykuł można być opublikowany (są wtedy widoczne na stronie) lub nieopublikowane (wtedy są widoczne tylko z panelu administracyjnego). Artykuły pisze się za pomocą narzędzia, przypominającego edytor tekstów z pakietów biurowych. Do spójnego zarządzania artykułami stosuje się sekcje, które są jakby zbiorami do których należą artykuły. Podstrony serwisu korzystają z sekcji, a później z samych artykułów.

Wykorzystanie wbudowanych narzędzi do tworzenia artykułów pozwala na szybkie i efektywne tworzenie zawartości, która poza tekstem może również zawierać zdjęcia, filmiki bądź nawet dźwięki. Pliki multimedialne zbierane są w bibliotece mediów, gdzie można je przeglądać bądź zmieniać.

Joomla! jako typowy CMS składa się z:

• Front End – przegląd serwisu z punktu widzenia zwykłego użytkownika (jak w przykładzie przedstawiony jest na rys. 3.3.)
• Back End – przegląd serwisu „od kuchni”, np. z punktu widzenia jego administratora. Administrator otrzymujemy możliwość korzystania z wielu ciekawych narzędzi. Jednym z potrzebniejszych narzędzi w serwisie jest... statystyka odwiedzin i najczęściej wyszukiwane słowa. Na rys. 3.1 przedstawiony jest widok aplikacji od strony administratora.

Tak naprawdę potęga Joomla! zawiera się w elastyczności, którą zawdzięcza się komponentom i modułom. Są to rozszerzenia, pozwalające na portalu stworzyć chociażby sklep, bądź forum dyskusyjne. Sieć jest bogata w różne komponenty i moduły.

3.2.5. Community Builder

Jednym z komponentów jest Community Builder. Jest to narzędzie do (dosłownie) budowania społeczności. Pozwala ono użytkownikom na zakładanie profili na takim serwisie, dzięki czemu jest możliwość integracji pomiędzy użytkownikami. Ta funkcyjność znana jest już z forów dyskusyjnych.

Community Builder wprowadza tutaj nową jakość. Możliwość zakładania profili, bezpośrednia wymiana informacji pomiędzy użytkownikami czy uczestnictwo w dyskusji na forum jest niczym w porównaniu z możliwością budowania samego portalu przez użytkownika. Osoba taka może pisać artykuły, newsy czy informacje, nawet na stronę tytułową. Tak naprawdę każdy użytkownik portalu opartego na CB może go również tworzyć.
3. Portale społecznościowe

3.2.6. Instalacja


3.3. Charakterystyka wybranych portali społecznościowych

Ze względu na różnice w budowie, zasięgu, jak również funkcjonalności, można wyróżnić dwa rodzaje serwisów społecznościowych: wewnętrzne i zewnętrzne. Główne różnice pomiędzy tymi dwoma rodzajami zostały wymienione w tab. 3.3.

3.3.1. demotywatory.pl

Opis: Jest to portal przeznaczony do umieszczania w nim zdjęć z dodatkowym opisem. Najczęściej jest to śmieszny komentarz dotyczący otaczającej nas rzeczywistości. Wydzielone są w nim dwie zasadnicze strony: pierwsza - główna i druga - poczekalnia. Do poczekalni trafiają wszystkie demotywatory 1 stworzone przez...
3.3. Charakterystyka wybranych portali społecznościowych

<table>
<thead>
<tr>
<th>cecha</th>
<th>serwis wewnętrzny</th>
<th>serwis zewnętrzny</th>
</tr>
</thead>
<tbody>
<tr>
<td>charakter</td>
<td>zamknięty/prywatny</td>
<td>otwarty</td>
</tr>
<tr>
<td>interakcje między użytkownikami</td>
<td>łatwe między użytkownikami tej samej grupy, w innym wypadku ograniczone</td>
<td>pełna możliwość interakcji</td>
</tr>
<tr>
<td>udostępnianie danych między użytkownikami</td>
<td>łatwe między użytkownikami tej samej grupy, w innym wypadku ograniczone</td>
<td>łatwe</td>
</tr>
</tbody>
</table>

Tab. 3.1: Różnice pomiędzy portalami wewnętrznymi, a zewnętrznymi.

użytkowników portalu. Następnie z nich wybierane są przez moderatorów co zabawniejsze i umieszczane na głównej. Każde zdjęcie może być skomentowane i ocenione. Oceny te są pomocą przy podejmowaniu decyzji przez moderatora, to on ma ostateczne zdanie który demotywator trafi na główną stronę. Wszystkie zdjęcia z opisem przysłane przez użytkowników pozostają w odpowiednim archiwum strony (poczekalnia ma swoje archiwum, główna swoje).

Użyte technologie: RSS, potwierdzenie rejestracji kodem z e-maila, strona napisana w PHP z elementami Java script.

3.3.2. digart.pl


Użyte technologie: RSS, potwierdzenie aktywacji kodem z e-maila, podczas rejestracji trzeba potwierdzić osobno zgodę na przetwarzanie danych oraz zobowiązanie do przestrzegania regulaminu (technologia ang. go through click), strona napisana w PHP z elementami Java script.

3.3.3. dioda.com.pl

Opis: Aktualnie największe forum robotyki amatorskiej w kraju. Miejsce wymiany doświadczeń adeptów elektroniki, mikrosystemów, programowania. Jeżeli ktokolwiek chce zacząć przygodę z robotyką, powinien zająć się właśnie na to forum. Na jego łamach bardziej doświadczeni użytkownicy chwalą się swoimi rozwiązaniami, podpowiadają rozwiązania problemów początkującym, a nawet
3. Portale społecznościowe

piszą swoiste artykułu (w formie postu na forum), gdzie każdy może zadać pytanie autorowi.

Użyte technologie: Serwis stworzony na CMSie phpBB, strona napisana w PHP z elementami Java script, RSS, potwierdzenie rejestracji linkiem wysłanym na e-maila.

3.3.4. wrzuta.pl


Użyte technologie: Podczas rejestracji należy potwierdzić znajomość regulaminu (technologia go through click), strona napisana w PHP z elementami Java script, potwierdzenie rejestracji linkiem wysłanym na e-maila.

3.4. Metodologia tworzenia własnego portalu

3.4.1. Koncepcja

Jeżeli użytkownik chce założyć portal społecznościowy, to musi zastanowić się nad tematyką portalu. Sukces naszej-klasy (http://wroclaw.gazeta.pl/wroclaw/1,35751,4502920.html) pokazuje, że by stworzyć popularny serwis, należy mieć dobrze opracowany i sprecyzowany pomysł.


3.4.2. Domena


Nie bez znaczenia dla właściciela jest również rodzaj serwera, na którym będzie działać portal. Własny serwer pozwoli zaoszczędzić pieniądze, które wydano by na profesjonalny hosting. Jednakże z drugiej strony, oznacza to zakupienie sprzętu, podłączenie sieci i serwisowanie serwera, co obciąża właściciela strony. Samo kupno bądź podłączenie nie jest tak zajmujące jak serwisowanie i administrowanie takiego serwisu. Często wymierne finansowo jest jednak wykupienie
3.4. Metodologia tworzenia własnego portalu

hostingu, gdyż wtedy otrzymuje się gotowy serwer, bez konieczności dodatkowej opieki nad sprzętem.

**Definicja 3.4.1** Hosting - usługa polegająca na udostępnianiu przez dostawców Internetu miejsca na swoich serwerach dla różnych usług np. serwisy WWW, konta pocztowe, radio internetowe itp.

Należy wspomnieć, że ceny hostingu w Polsce często są duże i dobrą alternatywą, jest korzystanie z hostingu, np. w Stanach Zjednoczonych. Ze względu na duży rynek i potrzeby, konkurencja jest tam duża, co z kolei przekłada się na wysoki standard usług proponowanych klientowi. Ceny serwisów hostingowych w porównaniu do polskich, są często niższe. Problem stanowi rozliczenie - należy posiadać kartę kredytową umożliwiającą transakcje w internecie (nie każda ma taka funkcjonalność) lub konto w PayPal.

### 3.4.3. Narzędzia

Podczas budowania portalu, wykorzystuje się ogromną liczbę różnorakich narzędzi, których lista wraz z przykładami i charakterystyką programów, zostanie przedstawiona poniżej.

**Klient ftp**

FTP (File Transfer Protocol) to protokół transmisji typu klient-serwer, umożliwiający przesyłanie plików z i na serwer za pośrednictwem sieci TCP/IP. Klient ftp to aplikacja umożliwiająca połączenie z serwerem ftp i pozwalająca na np. skopiowanie plików. Często klenci FTP wbudowani są już w przeglądarki internetowe, jednakże nie mają oni pełnej funkcjonalności. Dlatego też do przesyłania warto używać prawdziwych klientów FTP. Przykłady takich programów znajdują się poniżej.

- **FileZilla** – jest to międzyplatformowy, całkowicie darmowy (Open Source) klient FTP. Ma wiele interesujących opcji i intuicyjne interfejsy, choć nie jest polecony dla nowych użytkowników. Potęga FileZilla jest możliwość przesyłania plików o rozmiarze do 4GB i możliwość kontrolowania wysyłania plików, przez co podczas kopowania na/z serwera, możemy komfortowo pracować z innym programem wykorzystującym TCP/IP.

- **WinScp** – jest to darmowy klient ftp z interfejsem graficznym, tylko dla Windows. Cechuje go prostota, intuicyjna obsługa oraz możliwość fuzji z programem Putty, co wzbogaca program o funkcjonalność konsoli. Dodatkowo umożliwia połączenie SSH. SSH to protokół transmisji typu klient-serwer, służący do terminalowego łączenia się ze zdalnymi komputerami. Jest następcą Telnetu; transfer danych w SSH jest zaszyfrowany.

- **TotalComannder** – jest to shareware’owy menadżer plików dla Windows. Jeśli chodzi o funkcje, to śmiało można określić go mianem kombajnu, gdyż pozwala na obsługę archiwów, transfer plików ipf. Jego zaletą jest wszechstronność i bardzo dobra ergonomia interfejsu.
3. Portale społecznościowe


**Narzędzia graficzne**

Aspekt graficzny strony jest bardzo ważny. Zakłada się, że potrzeba niecałej sekundy oglądania, aby użytkownik strony wyrobił sobie pierwszą opinię na jej temat. Dlatego ważną jest ergonomia samej strony, a przede wszystkim kolory i różnego rodzaju multimedia. Warto skupić się na obrazkach, gdyż często jeden, ale dobry, obrazek zastępuje pół strony tekstu.

- **AcdSee** – jest uważany za jedną z najlepszych przeglądarek plików graficznych na świecie. Program nie jest darmowy i działa tylko pod kontrolą Windowsa, jednakże oferuje możliwość otwierania i edycji ogromnej ilości plików. Posiada wbudowane funkcje obróbki grafiki i charakteryzuje się bardzo przyjaznym interfejsem. Istnieje wersja shareware oprogramowania, pozobawiona części funkcjonalności.

- **IrfanView** – jest to bardzo szybka i ciesząca się dużą popularnością przeglądarka graficzna, konkurencyjna do IrfanView. Zawiera podobnie dużo funkcji, pozwalających na szybką retusz zdjęć, ale przede wszystkim jest darmowa. Do- stępna jest zarówno na system Windows jak i Linux.


**Edytory tekstu**

Narzędzia należące do tej grupy, służą do tworzenia zarówno tekstów (chociaż systemy budowania portali społecznościowych zawierają wbudowane narzędzia do edycji i tworzenia tekstów) jak i do pisania samych stron. W drugim przypadku warto wykorzystywać edytory, które posiadają mechanizmy wspomagania programowania, choćby podświetlanie składni PHP. Przedstawione poniżej programy można wykorzystywać w obu funkcjach. Dodatkowo wymieniono narzędzia wspomagające kreowanie szablonów CSS.


- **Kate** – można go nazwać kombajnem programistycznym. Pozwala na pisanie kodu w wielu językach. Jego głównymi atutami są podkreślanie składni oraz możliwość pracy w wielu zakładkach. Dodatkowo oferuje funkcję automatycz-
3.4. Metodologia tworzenia własnego portalu

negó dopełniania składni poleceń i generowania wcięć charakterystycznych dla różnych języków programowania.

- **Emacs** – jest to program, który dla wielu stał się kultowym narzędziem. Jest to właściwie środowisko, które umożliwia tworzenie, pisanie i edytowanie najprzeróżniejszych programów. Ma wbudowany system wspomagania pisania programów w chyba wszystkich językach programowania. Dodatkowo zaimplementowano w nim nawet proste gry. Jak przystało na wszechstronne narzędzie, można znaleźć wersje dla wszystkich systemów operacyjnych.

- **Free CSS Toolbox** – jest to narzędzie służące do generowania, sprawdzania poprawności i poprawiania szablonów CSS. Walidator (moduł do sprawdzania poprawności) pochodzi z konsorcjum W3C. Przestrzeń robocza podzielona jest na dwa obszary: **Input**, gdzie jest widoczny plik wejściowy oraz **Output**, gdzie wyświetlane jest plik po wprowadzonych zmianach. Funkcjonalność, która wyróżnia te program to opcja kompresji kodu. Pozwala ona usuwać zbędne znaki, puste linie i pomaga zachować czytelność kodu. Dodatkowo program sam wyszukuje błędy.

**Słowniki**

Podczas pracy nad portalem należy używać słowników i jeśli jest to możliwe, włączać funkcje automatycznego sprawdzania pisowni. Działania takie pozwala unikać sytuacji, gdy na głównej stronie "uje się kreskuje". Strona, na której roii się od błędów jest trudna do czytania i taki portal traci na wizerunku.

3.4.4. Zabezpieczenia

Podczas budowy portalu, należy zastanowić się nad zabezpieczeniami. Mają one chronić sam portal jak i użytkowników przed różnymi, niepożadanymi sytuacjami. Autorem takich sytuacji może być zarówno człowiek jak i bot lub robak komputerowy.

**Definicja 3.4.2** *Bot to program komputerowy, który wykonuje pewne czynności. Często ma za zadanie udawać człowieka. Boty mają charakter pozytywny (np. w grach komputerowych) jak i negatywny (podsuwanie się pod osoby).*

**Definicja 3.4.3** *Robak komputerowy to samo replikujący się program, przypominający wirusa komputerowego. Robak nie potrzebuje nośnika do propagacji - wykorzystując luki w oprogramowaniu, przenosi się między komputerami. Robaki mogą być wykorzystywane do tworzenia Botnetów - sieci składających się nawet z milionów zarażonych komputerów, które wykorzystuje się później do przeprowadzania ataków na serwisy internetowe.*

Zabezpieczenia można podzielić na parę grup. Poniżej przedstawiona jest propozycja podziału.
3. Portale społecznościowe

Zabezpieczenia przy rejestracji


Dodatkowym zabezpieczeniem podczas rejestracji są checkboxy, czyli miejsca które, by stać się użytkownikami portalu, należy zaznaczyć. Szczególnym, a zarazem głównym zabezpieczeniem strony jest checkbox dotyczący potwierdzenia przeczytania regulaminu. W regulaminie zawarte są prawa i obowiązki użytkownika portalu. Zasady są jasno określone i dobrze sprecyzowane, dzięki czemu w razie problemów, możliwe jest szybkie wyjaśnienie sprawy. Dzięki wprowadzeniu takiej opcji portal także zabezpiecza się przeciw prawnym skutkom prowadzenia działalności.

Zabezpieczenia podczas logowania


Kolejną modyfikacją jest stosowanie bezpiecznego logowania, czyli logowania z użyciem szyfrowania hasła lub wykorzystanie do logowania strony z certyfikałem bezpieczeństwa. Należy wspomnieć o dodatkowym zabezpieczeniu, może nieco trybliwym, aczkolwiek ciekawym. Jeśli podczas logowania podamy np. złe hasło, to warto by portal nie zwracał błędu o tym, że podaliśmy złe hasło. W przeciwnym wypadku potencjalny przestępca będzie miał wskazówkę, iż powinnym łamać hasło.
3.5. Założenia projektowe

Ostatnim już wyraźnie oznaczonym mechanizmem bezpieczeństwa, jest podawanie podczas logowania oprócz loginu i hasła, jeszcze dodatkowej informacji. Może to być wcześniej wspomniany napis z obrazka, bądź podane podczas rejestracji imię psa.

Warto również wspomnieć o samym hasle. Metody łamania hasła zmieniały się wraz z przebiegiem programistów. Obecnie najczęściej bazują one na metodzie wyrazów słownikowych (metoda siłowa, ang. brute force, w której sprawdzane są wszystkie możliwości, wymaga dużej mocy obliczeniowej by skończyć zadanie w przyzwoitym czasie), dlatego też nie zaleca się stosowania hasel opartych na takich wyrażeniach, nawet wzbogaconych liczbami, np. heniek85. Część serwisów podczas zakładania konta, informuje o sile hasła, czyli trudności złamania. Najbardziej odporne hasła składają się z ciągu liter i cyfr w pseudolosowej kolejności. Poleca się, aby hasło było zlepkiem pierwszych znaków w zdaniu (dodatkowo ze względu na budowę modułu logowania, można używać małych i dużych liter, co daje 2 razy więcej możliwości), np. Litwo! Ojczyzno Moja, tyś jest jak zdrowie, ile Cię trzeba cenić..., wygeneruje następujące hasło: LomTjZiCTC.

Zabezpieczenia podczas użytkowania

Podstawowym zabezpieczeniem podczas użytkowania są prawa dostępu użytkownika. Podczas budowy portalu należy wyznaczyć parę stopni praw dostępu. Standardowe definiuje się je dla: zwykłego użytkownika, moderatora, administratora i czasem superadministratora. Zwykły użytkownik nie może ingerować w wewnętrzne budowę portalu, co do pewnego stopnia jest możliwe dla moderatora, a całkowicie dostępne dla administratora. Podczas projektowania zabezpieczeń należy ponadto zastanowić się, czy nie istnieje potrzeba stworzenia stopni pośrednich. Istnieje również mechanizm, który pozwala nawet zwykłemu użytkownikowi ingerować we wnętrzu portalu. Jest to tzw. ACL.

Definicja 3.4.4 ACL (ang. Access Control List) - to lista kontroli dostępu. W systemach uniksowych pozwala ona na bardzo dokładną i szczegółową kontrolę dostępu do plików. Dzięki ACL możemy dla każdego użytkownika, rozpoznawanego przez system, np. po loginie, zdefiniować różne prawa dostępu do jednego pliku.

Utrzymanie systemu w dobrej kondycji wymaga większych uprawnień. Dzięki nim dobry administrator, czyli osoba odpowiedzialna za serwis, jest w stanie szybko wykryć nieprawidłowości w systemie i je naprawić (wykorzystując narzędzia dostarczone z portalu bądź napisane przez niego samego). Należy wspomnieć, że trudno zbudować autonomiczny system administrujący serwisem, ze względu na rozmity i zawyły sposób myślenia i wnioskowania ludzi.

3.5. Założenia projektowe

W związku z trwającymi badaniami oraz toczącymi się pracami magisterskimi odczuwana jest potrzeba stworzenia portalu, który umożliwiłby wymianę danych, doświadczeń i komunikację pomiędzy użytkownikami badającymi biosygnale. W środowisku tym niezwykle ważne są dane pomiarowe, które wygodnie
3. Portale społecznościowe

byłoby magazynować i opisywać. W celu demonstracji metodologii budowy portalu dla społeczności składającej się z tak specyficznych użytkowników postanowiono zbudować takowy w oparciu o system Joomla! i dodatek Community Builder.

3.5.1. Właściwości niefunkcyjne

Ze względu na charakter prac badawczych, portal ma charakter zamknięty i dostęp z zewnątrz, bez posiadanej konta, jest niemożliwy.

Warstwa sprzętowa platformy

Zdecydowano się na wykorzystanie komputera laboratoryjnego jako jednostki serwerowej portalu. Z jednej strony posunięcie to ułatwiło znacznie instalację i rozbudowę systemu, z drugiej jednak strony stworzyło to problemu podczas awarii zasilania, gdy doszło do uszkodzenia systemu plików serwera i konieczności reinstalacji systemu i portalu.

Parametry serwera z zainstalowanym systemem Ubuntu 9.10 LTS były następujące:

• procesor dwurdzeniowy Core 2 Duo E4500 (2,2 GHz),
• 1 GB Ram,
• 45 GB przestrzeni dyskowej przeznaczonej na serwer.


Warstwa programowa platformy

System operacyjny

W wersji finalnej serwer pracował pod kontrolą systemu operacyjnego Ubuntu 9.10 LTS. System został tak skonfigurowany, aby serwer wrócił sam do poprawnej pracy po utracie zasilania. W trakcie konfiguracji niezbędna była instalacja serwera Apache2, narzędzi do PHP i MySQL oraz serwerów SSH i FTP. Wszystkie narzędzia zostały pobrane w najnowszych wersjach. Po konfiguracji wyżej wymienionego oprogramowania, rozpoczęto instalację systemu zarządzania treścią Joomla!.

Joomla!

Przeprowadzono instalację, która odbywała się poprzez przeglądarkę internetową. Podczas samej konfiguracji, należało podać dane serwera MySQL, gdyż


### 3.5.2. Własności funkcjonalne

Grupa tych własności określa zakres funkcji portalu. W przypadku opisywanej portalu najważniejsza jest możliwość szybkiej komunikacji między użytkownikami i możliwość prowadzenia debat. Cel ten osiąga się dwojako:

- wykorzystując forum dyskusyjne,
- za pomocą artykułów, które są widoczne tylko dla zalogowanych użytkowników (pozwalają one w sposób przystępny przedstawiać wyniki prac wraz z metadanymi).


Jak już wcześniej wspomniano, portal jest całkowicie zamknięty i tylko administrator może zapraszać kolejnych użytkowników. Podyktowane jest to koniecznością ochrony wyników prac, które są pracami badawczymi.

Poniżej wymieniono założone poziomy dostępu. Należy wspomnieć, że każdy następny poziom dostępu, niejako dziedziczy cechy po poprzednich.

#### Użytkownik

Prawa:

- może tworzyć artykuły,
- istnieje możliwość wzajemnej komunikacji pomiędzy użytkownikami portalu.

#### Moderator

Moderator ma ograniczone prawa w stosunku do Administratora. Zadania i obowiązki:

- możliwość edycji artykułów tworzonych przez użytkowników,
- monitorowanie portalu,
3. Portale społecznościowe

Administrator

Zadania i obowiązki:

• tylko administrator może zakładać konta użytkownikom, osoba z zewnątrz nie może sama zarejestrować swoje konta; takie obostrzenie wymagane jest ze względu na charakter toczonej prac,
• zajmuje się aktualizacją pakietów serwisu, jak i aktualizacją samego serwera,
• dodaje usuwa opcje i funkcjonalności, jak np. kanał RSS,
• monitorowanie systemu i serwisu,
• rozbudowa serwisu,
• administrator nadaje poziomy dostępu.

3.6. Implementacja

Poniżej zamieszczono etapy budowy opisywanego portalu:

1. Wybranie i przygotowanie komputera, pełniącego rolę serwera (sprzęt i oprogramowanie).
2. Instalacja programów Apache2, MySQL, PHP w najnowszych wersjach oraz ich konfiguracja.
3. Ściąganie najnowszej wersji Joomla!, dodatków i pakietów językowych.
4. Rozpakowanie archiwum z Joomla! i skopiowanie do katalogu, w którym przechowywane są strony internetowe.
5. Wybranie w oknie przeglądarki adresu strony (np. localhost) i postępowanie zgodnie z instrukcjami na ekranie.
6. Instalacja modułów językowych, np. z pomocą wbudowanego instalatora, gdzie podawana jest tylko ścieżka do archiwum zip na komputerze.
8. Instalacja dodatkowych modułów (jak Comunity Builder, Fireboard itp.).
9. Rejestracja użytkowników.

Po wpisaniu adresu serwera w pasku adresowym przeglądarki pojawiła się standardowa strona Joomla!. Ze względu na zamknięty charakter portalu, przeprowadzono szereg modyfikacji mających na celu zminimalizowanie ilości informacji wypływających na zewnątrz.

W trakcie trwania prac zmodyfikowano szablon CSS odpowiedzialny za wygląd strony. Dodano anatomiczny rysunek ręki jako logo strony. Należy wspomnieć, że tak prosta z pozoru czynność, okazała się problematyczna, gdyż należy uważać podczas modyfikacji pliku CSS szablonu strony.

3.6.1. Od strony gościa

Na rys. 3.2 przedstawiono portal od strony gościa, odwiedzającego anonimowo stronę. W lewej kolumnie widać panel logowania się. Obsługuje on możliwość przywracania hasel, wyeliminowano możliwość rejestracji. Z tak zbudowanej strony można dowiedzieć się, co zespół robi, kto do niego należy oraz przeczy-
3.6. Implementacja

tać artykuły nie objęte restykcjami co do odbiorcy. Na stronie głównej pojawiają się również najnowsze informacje dotyczące badań. Gość nie ma możliwości wejścia na forum bądź skorzystania z jakiejkolwiek innej funkcji.

Rys. 3.2: Strona tytułowa portalu niezalogowanego użytkownika.

3.6.2. Od strony użytkownika

Na rys. 3.3 przedstawiono wygląd po zalogowaniu na portal. Różni się on znacząco w stosunku do poprzedniego interfejsu. W górnym i bocznym menu pojawiły się dodatkowe opcje. Boczne menu zostało dodatkowo wzbogacone o menu użytkownika. Dzięki niemu można wejść na forum, utworzyć nowy artykuł (rys. 3.4), bądź edytować profil.


3.6.3. Od strony administratora

Na rys. 3.6 przedstawiono wygląd strony po zalogowaniu jako administrator (robi się to dodając na koniec adresu dopisek \administrator i podając hasło ustalone przy instalacji systemu). Zaplecze podzielone jest na odpowiednie zakładki:
3. Portale społecznościowe

Rys. 3.3: Strona tytułowa portalu zalogowanego użytkownika.

Rys. 3.4: Tworzenie artykułu.

- w Witryna gdzie znaleźć można wszelkie ustawienia naszej strony,
- Menu odpowiada za poszczególne menu jak i ich ogólny układ na stronie,
- Artykuły doty whole artykułów, sekcji,kategorii oraz artykułów na stronie startowej,
- Komponenty zawiera wszelkie zainstalowane moduły komponentów, rozszerzające możliwości strony,
3.6. Implementacja

Rys. 3.5: Artykuł zredagowany przez użytkownika.

Rys. 3.6: Strona tytułowa portalu zalogowanego użytkownika - administratora.

- w **Rozszerzeniach** można znaleźć narzędzia do instalacji i włączania dodatkowych modułów,
- **Narzędzia** umożliwiają uruchomienie poczty na portalu,
- w **Pomoc** poza materiałami pomagającymi w budowie strony, znaleźć można wszystkie informacje o systemie; ta funkcjonalność przydaje się gdy napotyka się błąd trudny do zidentyfikowania, gdyż najczęściej nie stoi on po stronie sa-
3. Portale społecznościowe

...ego portalu. Opisy szczegółowe poszczególnych zakładek, jak również ich wykorzystanie można znaleźć w [1].

Z punktu widzenia administratora portalu, najważniejsza jest możliwość dodawania nowych użytkowników.

3.6.4. Moduły i dodatkowe elementy

Community Builder

CB jest dodatkiem do Joomla!, który ma za zadanie tworzenie społeczności internetowych. Moduł ten może być synchronizowany z innymi modułami, co w efekcie daje dobre narzędzie do tworzenia portali społecznościowych. CB to właściwie system do zarządzania użytkownikami. Pozwala na zbieranie informacji oraz na zachowanie indywidualizmu wśród grupy użytkowników. Roszzerza profil użytkownika o możliwość dodania np. numeru fax, dokładnego adresu czy chociażby dodania zdjęcia do portretu użytkownika.

Forum Fireboard

Jak już wcześnie wspomniano, CB ma możliwość integrowania się wraz z innymi narzędziami. Zdecydowano, że warto rozbudować portal o funkcjonalność forum, które jest jedną z lepszych metod szybkiej wymiany informacji. Wykorzystano do tego dodatek Fireboard, który ma możliwość wykorzystania tablic z danymi, stworzonymi przez CB. Na rys. 3.7 przedstawiono wygląd tego wbudowanego modułu. Użytkownik ma wyświetlane pełne statystyki, informacje o nowych postach i użytkownikach korzystających aktualnie z forum. Dostęp do tej czę-
3.6. Implementacja

ści strony również jest niemożliwy dla osób niebędących zarejestrowanymi użytkownikami. Należy wspomnieć, że Fireboard został napisany dla starszej wersji Joomla! Dlatego też należy włączyć zgodność ze starszą wersją systemu.

3.6.5. Zabezpieczenia

Portal w postaci finalnej nie został wzbogacony o rozszerzone mechanizmy zabezpieczające. Podstawowym zabezpieczeniem są prawa użytkowników i goście, które nie pozwalają danej osobie zrobić więcej, niż wynika to z ustawień. Dodatkowo w każdej chwili, istnieje możliwość takiej konfiguracji serwera ftp, ssh i http, aby przyjmowały połączenia tylko z wymienionych adresów IP. Zabezpiecza to w prosty sposób, przed niepożądanymi próbami logowania się do portalu. Z drugiej jednak strony, uniemożliwia korzystanie z portalu przez uprawnionych użytkowników z adresów innych niż zapisane.

3.6.6. RSS

Został uruchomiony kanał RSS, który przekazuje najnowsze informacje ze strony. Istnieje możliwość zablokowania dostępu do RSS dla gości i wykorzystanie go jako kanału informacyjnego dla współpracowników. Dodatkowo strona wyposażona została w moduł pozwalający na odczytywanie wiadomości wysyłanych przez RSS z innych portali. W ten sposób istnieje możliwość oglądania wiadomości z serwisu www.eka.pwr.wroc.pl.

Literatura

4.1. Wstęp


WWW bazuje na architekturze klient-serwer. Rolę klienta zazwyczaj pełni przeglądarka internetowa (program komputerowy wysyłający żądania protokoły komunikacyjnego HTTP do serwerów WWW w celu pobrania udostępnionych na nich dokumentów i wyświetlenia ich zawartości użytkownikowi), zaś rolę serwera - serwer WWW (program działający na serwerze internetowym, obsługujący żądania protokołu komunikacyjnego HTTP). Sama przeglądarka nie potrafi „myśleć” - to znaczy sama nie potrafi obserwować i interpretować wydarzeń zachodzących w sieci (np. nie jest w stanie poinformować użytkownika, że na jakiejs aukcji został wystawiony przedmiot go interesujący), nie przewiduje ruchów użytkownika, nie wypełnia automatycznie formularzy, nie dokonuje zakupów, nie pobiera automatycznie istotnych zasobów. Wykonać te czynności może tylko jednostka inteligentna, potrafiąca podejmować działania i przetwarzać dane - czyli robot sieciowy.
4.2. Roboty sieciowe

Jak powszechnie wiadomo, potrzeba jest matką wynalazków. Tak też było z powstaniem robotów sieciowych. Ze względu na duży nakład pracy, jaki użytkownik musi poświęcić na ręczne wykonywanie czynności wskazywania adresów stron internetowych, analizowania dostarczonych danych itp. w oknie przeglądarki, zaproponowano, aby czynności te wykonywał ktoś inny - robot sieciowy bądź cała grupa robotów sieciowych. Roboty sieciowe realizują zadania wykraczające daleko poza ograniczenia przeglądarki (czyli potrafią zbierać i filtrować informacje istotne dla użytkownika, mogą interpretować to, co znajdują w sieci, potrafią podejmować decyzje za użytkownika). Warto podkreślić, iż przeglądarki często korzystają z możliwości botów internetowych (więcej o zastosowaniach napisano w podrozdziale 4.3).

4.2.1. Czym są roboty sieciowe?


Istnieje też specjalny mechanizm informowania automatów o tym, czego nie powinny robić na stronie WWW. Dokonuje się to za pomocą protokołu REP (ang. Robots Exclusion Protocol), który pozwala na określenie reguł dla tzw. web spideringu, do których bot powinien się dostosować. Reguły te umieszcza się na serwerze w pliku o nazwie robots.txt albo w znacznikach meta dokumentów HTML.

4.2.2. Jak działa robot indeksujący?

Koncepcja działania robota indeksującego jest dość prosta. Jest to program korzystający z protokołu HTTP, który podobnie jak przeglądarka wysyła żądanie do serwera o adresie podanym przez użytkownika i zapisuje odpowiedzi na dysku. Po pobraniu dokumentów robot analizuje je i wyodrębnia z nich hiperlinie URL, które umieszcza na liście hiperlinków do odwiedzenia w kolejnej iteracji (na rys. 4.1 pokazano schemat działania prostego robota). Istnieje wiele strategii poruszania się robotów po hiperlinkach. Ważne jest, by robot nie odwiedzał tych samych stron, jak również sprawnie zapisywał istotne informacje podczas ich przeglądania. W wyniku działania robota ostatecznie powstaje baza danych, składająca się z tych informacji, które robot zakwalifikował jako istotne.

Ze względu na ogromną ilość danych publikowanych na stronach WWW, żaden robot nie jest w stanie wszystkie je przeszukać i przeanalizować. Z tego względu o efektywności działania robota w dużej mierze decyduje sposób imple-

57
4. Metody obrony przed robotami sieciowymi

Rys. 4.1: Schemat działania prostego robota.


4.3. Zastosowanie robotów sieciowych

W niniejszym podrozdziale zaprezentowane zostaną przykłady robotów sieciowych realizujących różne funkcje. Wymienione zastosowania oczywście nie wyczerpują tematu. Podrozdział ten ma uzmysłowić czytelnikowi, iż roboty sieciowe istnieją i mogą być wykorzystane do realizacji rozmaitych dobrych i, niestety, złych celów.

4.3.1. Roboty neutralne

Rozumiejąc sposób działania robotów sieciowych łatwo sobie wyobrazić ich liczne zastosowania. Właściwie można na ten temat napisać osobną książkę. Najczęściej roboty pełnią rolę szperaczy sieciowych (ang. web crawlers), które prze- mierzają sieć i zbierają z niej określone informacje. Roboty działają mogą na rzecz osób, które je stworzyły bądź uruchomiły. Oczywiście takie działanie może w niektórych sytuacjach nieco przeszkadzać innym użytkownikom internetu, aczkolwiek jest to skutek uboczny, a nie cel ich działania. Dla przykładu, właściciel
4.3. Zastosowanie robotów sieciowych

ogromnej i rozbudowanej strony internetowej może być zainteresowany infor-
macją, czy hiperlinki zamieszczone na jego stronie nie są martwe. Informację tę
może mu dostarczyć bot, który na bieżąco będzie te hiperlinki weryfikował. In-
nym przykładem zastosowania mogą być boty, które służą do pozycjonowania
stron. Wiele potężnych wyszukiwarek posiada takie boty (np. Google, Yahoo!).
Kolejnym przykładem może być bot do wyszukiwania informacji dostępnej w za-
sobach dyskowych, wspomagający użytkownika np. przy kolekcjonowaniu zdjęć.

Robots stworzone w celach komercyjnych

Motywacja do tworzenia robotów internetowych może mieć biznesowe pod-
łoże. Przykładami robotów, które powstały właśnie do spełniania biznesowych
potrzeb użytkowników są roboty kupujące. Roboty te to inteligentni agenci in-
ternetowi, którzy w sposób automatyczny dokonują zakupu w sieci w imieniu
człowieka. Wykorzystywanie tego typu botów jest znacznie lepsze od dokonywa-
nia zakupów w internecie w sposób manualny, ponieważ roboty nie tylko mogą
automatycznie przeprowadzić transakcję, ale również wykryć zdarzenia decydu-
jące o właściwym momencie jej przeprowadzenia (jak np. pojawienie się no-
wego produktu lub obniżenie ceny). Roboty zwalniają użytkownika od śmud-
nego, manualnego poszukiwania ofert, oszczędzają jego czas oraz chronią przed
ludzkimi bladami (np. przypadkowym przeoczeniu). Potrafią też wykorzystać
okazję, która występuje przez krótki okres czasu lub która może zostać odkryta
po wielu godzinach przeszukiwania stron.

Robot kupujący działa w taki sposób, iż posiadając przedmiot „na oku” (po
uprzednim przeanalizowaniu witryny) decyduje czy dokonać zakupu czy nie.
Kryteria zostały wprowadzone przez osobę, która stworzyła robota. Przed podję-
ciem decyzji i realizacją transakcji robot loguje się na konto użytkownika, w imieniu
którego działa (jeśli istnieje taka potrzeba). Przy dokonywaniu zakupu zostaje wy-
pełniony formularz w sposób automatyczny - wprowadza do niego informacje
m.in. o adresie, typie przesyłki itd. Ostatecznym krokiem jest sparsowanie final-
nnej strony, na której znajduje się ostateczny raport mówiący o tym, czy transakcja
przebiegła pomyślnie. Uogólniony schemat działania robota kupującego przed-
stawiono na rys. 4.2.

Omawiane roboty można spotkać na różnego rodzaju aukcjach (np. eBay),
dzieś wszelkie najbardziej interesujące przedmioty są natychmiast wykupywane.
Innymi przypadkami są portale, które sprzedają przez internet bilety na kon-
certy. Na takich portalach tuż po pokazaniu się nowych biletów, roboty wykupują
wszystkie miejsca z pierwszych rzędów, aby właściciel bota mógł je odsprzedać
później po wyższych cenach.

Robots stworzone w celach charytatywnych

Poza robotami, które potrafią przynieść różnego rodzaju korzyści właściciel-
owi, istnieją również takie, które pomagają innym. Istnieje strona internetowa
FreeRice.com na której zamieszczono grę polegającą na wyborze dla zadanego
 pytania jednej z czterech odpowiedzi. Z każdą dobrą odpowiedzią sponsorzy
4. Metody obrony przed robotami sieciowymi

Rys. 4.2: Schemat działania robota kupującego.

przeznaczają 8 ziarenek ryżu dla ludzi potrzebujących. Jest to świetne miejsce na użycie bota internetowego. Stworzono więc roboty grające w taką grę w sposób automatyczny. Są one znacznie szybsze od ludzi i mogą grać 24 godziny na dobę. Istnieją różne strategie wybierania poprawnej odpowiedzi. Można dokonywać wyboru w sposób całkowicie losowy (25% na trafienie prawidłowej), można też zatrudnić robota uczącego się ze słownikiem poprawnych i złych odpowiedzi. Z powodu rosnącej ilości skryptów (botów) używanych na tej stronie, ilość ryżu podarowanego znacznie wzrosła. Zostało oszacowane przy założeniu 3 sekundowej przerwy między iteracjami, iż jeden taki bot może nakarmić 8 ludzi dziennie, jeśli działa 24 godziny na dobę. Idąc dalej można stworzyć wielowątkowego robota sieciowego, który przy 50 wątkach będzie w stanie uzbierać 600 tysięcy ziarenka ryżu na godzinę, co pozwala na nakarmienie około 720 osób na świecie. Do tej pory nie pojawiły się sprzeciwów ze strony sponsorów. Aczkolwiek, tego typu boty są projektowane z rozwagą, aby „nie przesadzić”, to znaczy zniechęcić sponsorów, a co za tym idzie, nie doprowadzić do zamknięcia portalu.

Boty naśladowujące człowieka

Boty naśladowujące człowieka są botami obdarzonymi lepszą lub gorszą sztuczną inteligencją. Działają w taki sposób, aby inni użytkownicy nie mogli rozpoznać ich mask. Przykładem jest robot, którego zadaniem jest wypełnienie jakiegoś formularza, na którym znajduje się CAPTCHA. Wprowadzenie po-

60
4.3. Zastosowanie robotów sieciowych

prawnego ciągu znaków przez bota wymaga prawidłowego odczytania specjalnie spreparowanego obrazka. Tego typu robot posiada mechanizmy przetwarzające obraz, tak aby wydzielić informacje w nim zawarte. Zazwyczaj roboty posiadające tego typu funkcję zalicza się do robotów złośliwych, których celem może być wysyłanie spamu.

Innym przykładem mogą być tzw. Pokerboty. Są to roboty internetowe, które grają w pokera online. Jako, że powstaje coraz więcej portalów, na których można grać za prawdziwe pieniądze, popularność Pokerbotów rośnie. Uczestnikami wielu aktualnie odbywających się partii są boty. Bywa tak, iż użytkownik gra z inną „żywą” osobą, oraz kilkoma innymi botami, które należą do niego, mimo, że grają tak, jakby były osobnymi graczkami. Współpracując z botami oraz znając ich karty posiada się ogromną przewagę nad „ofiarami”. W ten sposób wiele osób „zeruje” na początkujących graczy, którzy nie mają świadomości, iż tak naprawdę przy jednym stole nie gra kilku różnych osób, lecz gra tylko on, zaś po drugiej stołu jest przeciwnik razem z botami z nim współpracującymi.

4.3.2. Roboty złośliwe

Roboty złośliwe są zdolne do zautomatyzowanych ataków na komputery w sieci. Ataki te mogą przyjmować różne formy. Często boty tworzą tzw. botnet - jest to sieć robotów internetowych, które kooperują ze sobą. Dla przykładu przeprowadzenie ataku typu DDoS przeprowadzany jest równocześnie z wielu komputerów, nad którymi kontrolę przejęły boty, trojany - takie komputery nazywane noszą nazwę zombie. Atak DDoS polega na tym, że na dany sygnał komputer równocześnie atakuje system ofiary, masowo wysyłając żądanie o skorzystanie z usługi, jakie komputer oferuje. Na każde wywołanie serwer musi przydzielić pewne zasoby, które są ograniczone, przez co przy dużej ilości wywołań komputer chwilowo przerywa działanie lub się całkowicie zwiesza, gdy zasoby się wyczerpyły.

Rodzaje ataków

Istnieją następujące typy robotów złośliwych:

• Boty ścigające całą zawartość strony a następnie użycie ich bez zgody autora.
• Boty spamujące, które zbierają adresy e-mail ze stron internetowych w celu wysyłania niechcianej przez właściciela adresu informacji (spamu). Innym sposobem jest cykliczne wypełnianie formularzy i wysyłanie bezwartościowych informacji.
• Roboty internetowe, które wchodząc na stronę i ścigając wszystkie pliki lub podążając za każdym linkiem równocześnie powodują zajęcie łącza i spowolnienie pracy serwera.
• Botnet - czyli sieć botów, które są w stanie przeprowadzić zorganizowany atak na jakiś komputer znajdujący się w sieci.
• Boty, które przemierzają się w celu znalezienia komputera, który jest niezabezpieczony, lub posiada luki, które powodują iż jest podatny na zainfekowanie wirusem, tojanem - czyli potencjalny zombie.
4. Metody obrony przed robotami sieciowymi

4.4. Obrona przed robotami sieciowymi


4.4.1. Sprecyzowanie zasad użytkowania strony, Robots.txt


- **User-agent:** - to pole z informacją o tym jakich botów dotyczy dany rekord.
- **User-agent:*** - zaznacza, że tyczy się wszystkich botów.
- **Disallow:** - określa, które zasoby mają być niedostępne.

Alternatywnym rozwiązaniem jest zaznaczenie podobnych informacji w znaczniku META ROBOTS w sekcji HEAD dokumentu HTML. Wypisanie odpowiednich parametrów powoduje ustalenie ograniczeń nałożonych na boty.

4.4.2. Morale na straży bezpieczeństwa WWW

zaakceptowaniu zasad mamy do czynienia z ewidentnym naruszeniem prawa. Wydawać by się mogło, że kliknięcie przycisku, nikogo w niczym nie powstrzymuje. Zgadza się, ale może to być podstawą w procesie karnym - jeśli do takiego dojścia... Oczywiście ochrona przeciw botom kojarzy się z walką z przestępcami, którzy ewidentnie łamią wszelkie przepisy i normy, wyrządzając szkody. Do walki z takimi przeciwnikami trzeba użyć siłowych rozwiązań, a nie słów i wiary w ich nawrócenie. Mimo wszystko jednak informacja o tym, co wolno a co nie, jest pewnego rodzaju obroną którą należy stosować.

4.4.3. Walka z atakami pochodzącymi z botnetów

Co czwarty domowy komputer podłączony do sieci jest zombie. Abstrahując od rodzajów infekcji, ich skutków, oraz przyczyn, można zadać sobie: „jak się przed tym bronić?”

Firewall


Podejmowanie akcji obronnych po zainfekowaniu

4. Metody obrony przed robotami sieciowymi

krokiem jest zdobycie informacji potrzebnych do celnego kontratak. Istnieje kilka sposobów ich pozyskania.

Sniffing

Sniffing to najłatwiejszy i najszybszy ze sposobów polegający na podsłuchaniu sieciowych pakietów wychodzących z komputera. Sniffer to program komputerowy lub urządzenie, którego zadaniem jest przechwytywanie danych przepływających w sieci. Popularnym narzędziem tego typu są: Wireshark (przed 2006 rokiem Ethereal), tcpdump, sniffit, ettercap, dsniff oraz snort. Większość z wymienionych narzędzi jest dostępnych za darmo w sieci. Można również w łatwy sposób dowiedzieć się, jak je wykorzystać.


Stosując podsłuch znajdowane jest w pakietach nawiązanie połączenia TCP pomiędzy botem a serwerem IRC. Stąd uzyskiwany jest adres serwera oraz nr portu, na którym działa usługa IRC. Dzięki tym informacjom można już obserwować botnet, lecz aby go zniszczyć, konieczne jest hasło dostępu do niego.

Czytanie plików binarnych

dane konfiguracyjne bota, takie jak komenda logowania, hasło, nazwa kanału, klucz, adres serwera, wersja bota itp. Czyli wszystkie dane potrzebne do zniszczenia botneta.

4.4.4. Rejestracja jako ochrona przed botami

Rejestracja to dowodzenie, że osoba chcąca korzystać z pewnej strony internetowej jest człowiekiem. Podanie loginu, hasła, maile jest niezłą weryfikacją tego, czy można danego użytkownika dopuścić do danych. Jest to utrudnienie dla botów, które prócz stworzenia sztucznych danych w celu rejestracji muszą zazwyczaj czekać na zgodę. Dodatkowo w razie wykrycia niepożądanych działań osobnika istnieje możliwość banowania go i odsunięcia od zasobu informacji. Ograniczenie polegające na rejestracji staje się również pewną niewygodą dla uczciwych użytkowników, ale zauważysz na to, że ułatwia to kontrolę nad treściami i zachowaniami społeczności można śmiało powiedzieć, że jest to bardzo wskazany proceder. Niestety, boty często nie potrzebują rejestracji, żeby dobrać się do danych, lub potrafią to ominąć.

4.4.5. CAPTCHA


Test Turinga

CAPTCHA ma stanowić zaporę, która selektywnie dopuszcza do pewnych zasobów ludzi a roboty nie. Sprawdza się wówczas, gdy jakość decyzji podejmowanych przez maszynę jest dużo gorsza od człowieka. Przewaga człowieka jest jego intuicją, doświadczением oraz umiejętnością łączenia abstrakcyjnych faktów pochodzących z różnych źródeł, również tych z którymi nigdy wcześniej nie miał styczności - jest po prostu inteligentny. Maszyna posiada możliwości rozpoznawania danych i posiada algorytm, na podstawie którego przetwarza dane. Potęga maszyny jest praktycznie nieograniczony zbiór danych, szybkość działania oraz nieustępliwość. W przypadku rozpoznawania obrazu komputer ma konkretne zadanie do wykonania - określić jaki zbiór znaków znajduje się w okienku wyświetla-
4. Metody obrony przed robotami sieciowymi

nym na monitorze. Czy jest to rzeczywiście zadanie przekraczające możliwości bota? Biorąc pod uwagę zestaw technik OCR (ang. Optical Character Recognition), służących do rozpoznawania znaków i całych tekstów w pliku graficznym, które mogą stosować bota, odpowiedź może być negatywna. Wniosek jest zatem taki, że jeśli już mają być zastosowane takie zabezpieczenie, to powinny one być trudne do złamania dla botów.

Sposoby przedstawienia ciągu znaków, który może rozpoznać tylko człowiek, nie jest łatwe. Zdecydowanie złe pomysły to zastosowanie niskiego kontrastu, zmiana odstępu między literami, zmiana koloru, rozmywanie znaków, tasowanie rozmianami, wpatlanie drobnych szumów do obrazka czy też tworzenie tła w kratkę. Wszystkie takie zabiegi nie sprawiają wielkiej trudności botom. Poniżej zamieszczono kilka pomysłów na przedstawiania obrazków tak, aby były trudno rozpoznawalne przez bota:

• Litery pisane blisko siebie mogą być problemem, gdyż nie można wtedy analizować każdej z osobna i jest szansa, że dwie litery zostaną zinterpretowane jako jedna.
• Zastosowanie „źabiego oka” w pewnych miejscach tekstu jest to dużo lepszy pomysł od obróconego albo falującego napisu, który nie tak trudno automatycznie wyprostować. Wszelkie nielinearne transformacje tekstu, które nie da się rozpoznać na podstawie konturu tekstu są kłopotliwe w rozpoznawaniu.
• Słowa nie znajdujące się w słownikach wykluczają możliwość dopasowywania do takich, które mogą znajdować się w gotowej bazie danych.

Racjonalne sposoby tworzenia CAPTCHA

Maszyna działa w sekwencyjny sposób. Podkreślone to było już wielokrotnie. Twórca botów ma o wiele trudniejsze zadanie od hakerów osobiście fatygujących się o złamanie pewnych zabezpieczeń. Wykopanie rowu przy użyciu szpadla jest ciężką pracą, lecz gdy się zaprzemy i poświęcimy tej czynności sporo czasu, to nam się uda. Zbudowania robota do kopania rowów nie jest już tak prostym zajęciem - trzeba go nauczyć kilku podstawowych czynności, musi on nabrać pewnych intuiacji, przystosować się do różnych możliwości itp. Lecz gdy już jest ukończony, może z powodzeniem kopać i kopać i nigdy się nie mręczy. To może zbyt trywialne i ogólne porównanie, ale nie bez sensu. Boty internetowe nie posiadają takich algorytmów pomagających im rozpoznawać znaki i wpiąć właściwy ich ciąg w pole CAPTCHA. Znając sposoby działania OCRu, możliwości współczesnych algorytmów rozpoznawania obrazu (a przede wszystkim ich braków) można wyciągnąć wnioski co do sposobu tworzenia zabezpieczeń.

Bot atakuje słabe punkty, aby go powstrzymać, dobrze jest znać z kolei jego słabości i ograniczenia. Ciąg znaków nie musi (nawet nie powinien) być trudny do rozpoznania przez człowieka - powinien być nie do rozgryzienia przez pozba- wiony wyobraźni automat.

Historia pokazuje, że każdą twierdzę da się zdobyć. Wszystko to w kwietniu czasu, który jest niezbędny na rozwój broni oraz zdobycie informacji o jej słabych punktach. Bronimy się tak, aby uniemowliwić atak który znamy, jest możliwe w danej
4.4. Obrona przed robotami sieciowymi


Skoro i tak nasz bastion padnie, należy go stawiać w sposób możliwie oszczędny, a przy tym sprawić, żeby napastnik odniósł jak najwięcej ran przy ataku. Do tego przydatna jest ocena tego, co chcemy chronić. Jeśli przejdziemy, że nikt nie będzie chciał nas atakować, to nie ma sensu zajmować się budową defensywy. Jeśli nasze dane mogą stać się celem ataku, to wypadałoby je zabezpieczyć. Wartość zabezpieczenia powinna być proporcjonalna do jakości naszych danych. W przypadku tworzenia stron internetowych, które zazwyczaj nie są narażone na zmasowany atak wybitnych treserów botów, najrozsądniejsze wydaje się stworzenie jak najszybciej, i bez zbędnych wysiłków dość dobrą CAPCHE.

4.4.6. Antyspam

Dwa wcześnie opisane sposoby zabezpieczenia: CAPTCHA i rejestrowanie mają jedną podstawową wadę - są uciążliwe dla uczciwego użytkownika i do tego ich skuteczność nie jest doskonała. Inna metoda, to proste przyblokowanie dokładnie tego, co jest niezbędne dla spamera.

Filtracja wiadomości

Po pierwsze można zablokować to, co przychodzi. Najprostszy filtr to taki, który powoduje niedostawanie wiadomości z zakazanych stron (taka lista może być dynamicznie tworzona w trakcie otrzymywania wiadomości, lub pobierana z sieci na podstawie wniosków innych osób). Nieznane dochodzące wiadomości również można w dość trwawnie sposób przyporządkować do klasy spamów. Zauważmy, że rzadko kiedy ktoś w komentarzach podruża więcej niż kilka linków, więc pierwszą obroną jest odrzucanie postów z dużą ilością, co charakteryzuje spam. Wykrycie linków jest banalnie proste: wystarczy zliczyć ilość http i https.Client wykonania tego jest dostępny na stronie: \texttt{http://pl2.php.net/substr-count}.

Przyglądając się jakie są najczęstsze spamy można dojść do wniosku, że to jest całkiem niezła metoda. Kolejna racjonalna metoda filtracji to nie przyjmowanie wiadomości o dużej liczbie słów kluczowych takich jak np. jakiś produktów.
4. Metody obrony przed robotami sieciowymi

Zablokowanie dostępu


4.4.7. Obrona przed atakiem typu DoS i DDoS

4.5. Projekt gotowych rozwiązań ochronnych przed netbotami

ków Linuxa zapora HLShield. Blokuje on wszystkie znane ataki na serwery, w tym DDoS. Instalacja jest prosta i szybka, korzystanie z narzędzia bardzo intuicyjne.

4.4.8. Podsumowanie


4.5. Projekt gotowych rozwiązań ochronnych przed netbotami

4.5.1. Założenia projektowe

Autorzy niniejszego rozdziału zaplanowali zebrać, zmodyfikować i opracować fragmenty kodów napisanych w języku PHP, tworząc bazę narzędzi do ochrony stron www przed netbotami. W wyniku prac projektowych powstał gotowy toolbox dla edytora Bluefish, stanowiący wygodne wsparcie projektantom stron internetowych. Po wybraniu jednej z opcji użytkownik edytora miał otrzymać kod określonego zabezpieczenia. Dodatkowo wsparciem użytkownika miał być manual z opisem każdego z rozwiązań i podpowiedziami, jak je stosować. O wyborze języka PHP przewyższyły względy jego popularności i prostoty. Edytor Bluefish został wybrany ze względu na to, iż jest to darmowy, jeden z najbardziej popularnych edytorów dostępnych na wiele platform.

4.5.2. Wynik projektu

W wyniku przeprowadzenia prac projektowych powstał zestaw narzędzi do obrony przed atakami robotów (dostępnych przez toolbox programu Bluefish) z przykładową stroną internetową, do ochrony której zostały one wykorzystane. Bazując na kodzie tej strony można w łatwy sposób zbudować zabezpieczenia dla innych stron. Aby to zrobić wystarczy zmodyfikować pliki projektu. Zaimplementowano następujące narzędzia:

- Okienko z przepisami (warunkiem wejścia na stronę jest zaznaczenia 'akceptacji' tych warunków oraz wciśnięcie 'OK')
4. Metody obrony przed robotami sieciowymi

- Link pułapka na bota (gdy bot indeksuje stronę, trafia na link, który jest odsyłaczem do pustej strony, która zawiera kolejny link pułapkę... Dzięki temu zostaje złapany w dynamiczną sieć tworzonych stron bez treści, po których błądzie).
- Blokada dostępu do strony przez użytkownika z pewnej puli IP (może być dopisywane IP ustalonych botów, lub będą automatycznie dopisywane boty, które wpadły w siłę linku pułapki).
- Formularz rejestrujący użytkownika (wzbogacony o formularz pułapkę, posiadający jedno ukryte pole tekstowe do wypełnienia, które jest niewidoczne dla użytkownika).
- Pytanie, na które odpowiedź będzie podstawą dostępu do strony.
- CAPTCHA

Aby skorzystać z jednego z dostarczonych zabezpieczeń należy zastosować kroki opisane w dalszej części tego rozdziału.

4.5.3. Dodanie toolbox’a

W celu skorzystania z zaimplementowanego Toolboxa środowiska programistycznego Bluefish należy najpierw to środowisko zainstalować. Następnie, w celu dodania toolboxa, należy odszukać oryginalny plik `custom_menu` i zastąpić go plikiem `custom_menu`. Przykładowo - w systemie Linux, plik ten znajduje się w `/home/<użytkownik>/bluefish` przy domyślnej instalacji środowiska Bluefish. Wygląd toolboxa po prawidłowym dołączeniu go do środowiska bluefish przedstawiono na rys. 4.3. Na rysunku widać sposób korzystania z przygotowanych rozwiązań. Po kliknięciu w wybrane pole zostaje wygenerowany kod w php.

Rys. 4.3: Przykład dołączonego toolbox’a do środowiska bluefish.
4.5. Projekt gotowych rozwiązań ochronnych przed netbotami

dedykowany dla danego rozwiązania. Schemat struktury strony zabezpieczonej stworzonymi narzędziami przedstawiono na rys. 4.4.

Rys. 4.4: Schemat struktury zabezpieczonej strony.

4.5.4. Akceptacja regulaminu

Aby dodać podstawowe zabezpieczenie strony jakim jest akceptacja regulaminu należy:


CAPTCHA

Kod źródłowy CAPTCHA znajduje się w pliku capcha.php. Jest to ogólnoświetne gotowe rozwiązanie, stworzone przez Jose Rodriguez na licencji GPLv3.
4. *Metody obrony przed robotami sieciowymi*

Aby z niego skorzystać należy skopiować folder `resources` do folderu, w którym znajdują się pliki źródłowe strony. Sposób wykorzystania CAPTCHA został pokazany w kolejnym podpunkcie.

**Zabezpieczony formularz rejestracyjny**

Aby wykonać formularz rejestracyjny należy:

1. W nowym dokumencie z menu podręcznego wybrać PHP+HTML → Obrona → Rejestracja → `rejestrujform.php`. Kod przedstawia pole rejestracyjne, oraz odwołania do bazy danych o użytkownikach i generatora losowych słów oraz pliku CAPTCHA.

**Logowanie**

Aby wprowadzić możliwość logowania się użytkownika na stronę internetową należy:

1. W nowym dokumencie z menu podręcznego wybrać PHP+HTML → Obrona → Logowanie → `logujform.php`.
2. W nowym dokumencie z menu podręcznego wybrać PHP+HTML → Obrona → Logowanie → `loguj.php`.

**Strona główna**

Kod przykładowej strony głównej został stworzony jedynie w celach testowych. Przykład wykonania takiej prostej strony, która została zabezpieczona wcześniej omówionymi zabezpieczeniami został przedstawiony w kodzie: PHP+HTML → Obrona → StronaGłowna

4.5.5. **Projekt - uwagi końcowe**

Przykład zabezpieczonej strony internetowej znajduje się pod adresem `http://chomiczak.ovh.org/waski/start.php`. Pole rejestracji, jako przykład zabezpieczenia zamieszczonego jako wykorzystanie opisywanych metod obrony przedstawiono na rys. 4.5. Przedstawione rozwiązania są przykładami, które
4.5. Projekt gotowych rozwiązań ochronnych przed netbotami

Rys. 4.5: Przykład zabezpieczonej strony internetowej [http://chomiczak.ovh.org/waski/start.php].

Użytkownik powinien wykorzystać zgodnie ze swoimi potrzebami. Warto zaznaczyć, że w różnych przypadkach zastosowanie danego zabezpieczenia może się okazać właściwe.

Literatura

5. Wstęp

W dobie informatyzacji niemal każdej dziedziny życia wraz z powszechnym dostępem do środków pozwalających na gromadzenie informacji, pojawia się problem ekstrakcji wiedzy z danych z uwzględnieniem minimalizacji czasu przeznaczonego na poszukiwanie. Ze względu na ogrom istniejących zasobów oraz tempo pojawiania się nowych niemożliwe jest poszukiwanie w nich wiedzy bez użycia zaawansowanych technik informatycznych.

5.1.1. Wprowadzenie do programowania agentowego i eksploracji danych

Eksploracja danych (ang. Data Mining) jest procesem automatycznego odkrywania wiedzy w bazach danych, hurtowniach danych, repozytoriach danych i innych (www, obiektowe bazy danych itp.) poprzez wykrywanie nietrywialnych związków, zależności, podobieństw, trendów. Eksploracja danych to także proces zmniejszenia ilości danych za pomocą tworzenia wykresów, reguł logicznych, klasyfikatorów (np. drzew decyzyjnych), zbiorów skupień itp.


Zapytania operacyjne są prostymi poleceniami realizowanymi na przykład za pomocą standardu SQL. Przykładem takiego zapytania może być sformułowanie: „Jaka jest średnia semestralna studentów Politechniki Wrocławskiej w semestrze letnim 2009?”.

Bardziej złożonym zapytaniem jest zapytanie analityczne, na przykład: „Jaka była średnia semestralna studentów uczelni wyższych z uwzględnieniem podziału na uczelnie techniczne, akademie medyczne, uniwersytety w ostatnich 3 lata-
5.1. Wstęp

tach?”. Typ ten opiera się na modelu OLAP (ang. *Online Analytical Processing*), który można interpretować jako rozszerzenie standardu SQL o możliwość interpretowania złożonych zapytań zawierających agregaty.

Bardziej ogólne zapytania, czyli zapytania eksploracyjne, wymagają innego rodzaju analizy danych. Ich interpretacja nie jest możliwa ani za pomocą SQL, ani OLAP. Przykładem może być sformułowanie: „Czy można przypuścić, że poziom edukacji w Polsce wzrasta?”. Warto podkreślić, że Data Mining jest jednym z etapów pozyskiwania wiedzy z danych, nie jest natomiast realizacją standardu SQL, analizą OLAP ani systemem ekspertowym. Do metod Eksploracji danych należą:

- grupowanie,
- klasyfikacja,
- odkrywanie sekwencji,
- odkrywanie charakterystyk,
- analiza przebiegów czasowych,
- odkrywanie asocjacji,
- wykrywanie zmian i odchyleń,
- eksploracja www,
- eksploracja tekstów.

Narzędziami wykorzystanymi w Eksploracji danych może być większość dostępnych koncepcji programistycznych realizowanych właściwie w dowolnym języku, od programowania sekwencyjnego w języku maszynowym aż do programowania agentowego.


- obliczeniowa, która jest autonomiczna co oznacza, że agent potrafi działać bez zewnętrznej ingerencji człowieka lub innych agentów,
- osadzona w konkretnym środowisku, czyli przyjmująca bodźce ze środowiska i wykonująca na nim akcje,
- elastyczna.

Elastyczność agenta realizuje się poprzez jego cechy takie jak:

- reaktywność,
- proaktywność,
- socjalność,
- mobilność.

Reaktywność jest zdolnością agenta do odbierania zmian zaistniałych w otoczeniu i natychmiastowej reakcji. Proaktywność oznacza, że agent potrafi podejmować inicjatywę, zmieniać metody postępowania w związku z zaistniałymi zdarzeniami oraz celami. Dodatkowo agent posiada umiejętności formułowania i modifikacji własnych celów i podcelów. Socjalność agenta oznacza jego zdolność do interakcji z innymi agentami i ludźmi za pomocą ustalonego języka komunikacji. Mobilność oznacza, że agent (program agentowy) potrafi przenosić się w inne miejsce (na inną maszynę, do innego środowiska) i kontynuować swoje zadania. Agent jest jednostką inteligentną i współpracującą. Inteligencja wyraża się w działaniu elastycznym i racjonalnym poprzez uczenie się, rozwiązywanie pro-
blemów i podejmowanie decyzji. Umiejętność współpracy agenta z innymi agentami i człowiekiem opiera się na dwóch wzorcach interakcji: koordynacji zorientowanej na cel lub zadanie oraz współzawodnictwo, gdzie w systemie znajdują się agenci posiadający różne cele. Ze względu na sposób postrzegania otoczenia, wpływ na środowisko i podejmowanie decyzji wyróżnić można trzy rodzaje agentów:

- agenci reaktywni,
- agenci intencjonalni,
- agenci socjalni.

Najprostszym rodzajem jest agent reaktywny. Potrafi on reagować na zmiany w środowisku i komunikaty od innych agentów lub użytkownika. Agent reaktywny nie potrafi definiować celów, ani wnioskować o intencjach. Przykładem takich agentów będa Agenci Wyszukujący. Kolejną grupą są agenci intencjonalni. Mogą oni budować własne cele w oparciu o doświadczenie, przekonania i intencje, planować realizację celów, wykrywać konflikty między podcеляmi, oraz wykonywać plany i w razie potrzeby modyfikować je w trakcie wykonywania. Przykładem agenta intencjonalnego w systemie opisanym dalej będzie Agent Personalfi. Trzecią grupę stanowią agenci socjalni, którzy dodatkowo zbierają informacje o innych agentach i na tej podstawie tworzą modele złożone z ich przekonań celów i planów oraz wyciągają wnioski o tych agentach (przewidyją hipotetyczne zachowania, reakcje, intencje i zobowiązań).

Definicja agenta wiąże gościśle z środowiskiem w którym agent działa. Środowiskiem może być niemalże każdy obiekt czy nawet pojęcie, np. świat rzeczywisty, gra planszowa, rodzina procesów związanych z wyborem i zamawianiem wyjazdów, giełdy walutowe itp. Środowiska można opisać za pomocą pięciu cech:

- dostępności,
- epizodyczności,
- dynamiczności,
- ciągłości,
- determinizmu.

Więcej o sposobie klasyfikacji środowiska można znaleźć w pracy [2].

Jakkolwiek koncepcja programowania agentowego wygląda zachęcająco, jednakże nie jest tak, że po włączeniu komputer od razu można skontaktować się z agentem osobistym, który wyświetli wyselekcjonowane wiadomości, przedstawi plan dnia, a na podstawie prognozy pogody, programu dnia i upodobań użytkownika doradzi mu jak się ubrać. Nie jest też niestety tak, że zadanie zarządzania siecią komputerową można przekazać inteligentnemu agentowi, który na przykład uzgodni z agentami lokalnymi dla węzłów sieci, jaki jest najlepszy harmonogram instalacji nowej wersji oprogramowania. W opinii autorów pracy koncepcja ta jest wciąż na etapie rozwoju i wymaga znaczących usprawnień, być może rewolucyjnych, w kwestii szeroko pojętego bezpieczeństwa danych.

5.2. Systemy wieloagentowe

5.2.1. Podstawowe właściwości

System wieloagentowy jest zbiorem powiązanych ze sobą – w pewnym, abstractyjnym sensie – wielu agentów. Trybownym przypadkiem jest system zło-
5.2. Systemy wieloagentowe

żony z jednego agenta. Systemy agentowe okazują się doskonałym modelem do reprezentowania problemów dających się rozwiązać różnymi sposobami. Zaletą ich stosowania jest możliwość działania rozproszonego, współbieżnego i asynchronicznego. Pozwala to na rozwiązywanie problemów, w których dane są rozproszone, z możliwością korzystania z już działających rozwiązań, niekoniecznie w duchu wieloagentowości, we współpracy z innymi systemami. System wieloagentowy jest lokalnie odporny na błędy w funkcjonowaniu w tym sensie, że wykluczenie jednego z szeregowych agentów nie powoduje zaniku funkcjonalności całości. Wadą systemów wieloagentowych jest konieczność specyfikacji wielu wzorców interakcji, sposobów przydzielania zadań, sposobów rozwijywania konfliktów, reguł wnioskowania o innych agentach i użytkownikach systemu. Zasadniczym problemem jest takie zaprojektowanie systemu, aby wykazywał on spójność w działaniu.

5.2.2. Opis komunikacji w systemie wieloagentowym

Komunikacja w systemie wieloagentowym jest synchroniczną lub asynchroniczną wymianą danych pomiędzy agentami. Ponadto współdzielenie wiedzy wymaga komunikacji, która z kolei potrzebuje wspólnego dla niej języka. W ramach języka można wyróżnić takie składowe jak:

- syntaktyka – opisuje relacje, które zachodzą między wyrażeniami (znakami językowymi) wewnątrz języka i które mają charakter formalny,
- semantyka – opisuje relacje między znakami (w tym wyrażeniami) a rzeczywistością, do której znaki się odnoszą,
- pragmatyka – opisuje relacje między znakiem a odbiorcą (interpretatorem).

Badania nad problemem wspólnego języka, służącego do opisu zawartości bazy wiedzy, doprowadziły do powstania języka KIF (ang. Knowledge Interchange Format). Może on być używany jako pośredni (podczas tłumaczenia jednego języka reprezentacji wiedzy na inny) lub jako wspólny język opisu zawartości bazy wiedzy kilku agentów, używających różnych wewnętrznych języków reprezentacji. Ponadto język KIF posiada reprezentację czytelną dla ludzi. Przykładem wyrażenia zapisanego za pomocą formatu KIF może być:

\[
\begin{array}{c}
(> (\text{cena przejazd1}) (\text{cena przejazd2}))
\end{array}
\]

informuje ono, że cena za przejazd przejazd1 jest większa od przejazd2.


Performatywy w języku KQML jest ciągiem znaków ASCII. Każda wiadomość, oprócz swej nazwy identyfikującej rodzaj performatywu, posiada zbiór parame-
5. Programowanie agentowe w eksploracji danych

trów w postaci :nazwa wartość. Do podstawowych parametrów performatywów w języku KQML należą:

- :sender – nadawca wiadomości,
- :receiver – odbiorca wiadomości,
- :reply-with – zawiera identyfikator, na który powoła się odbiorca odpowiadający na tę wiadomość,
- :in-reply-to – identyfikator określający, na jaką wiadomość odpowiada agent,
- :content – zawartość wiadomości,
- :language – język, w którym reprezentowana jest zawartość wiadomości (:content),
- :ontology – nazwa ontologii, do której odnosi się zawartość wiadomości (:content),
- :force – określa czy nadawca w przyszłości może zmienić znaczenie tego performatywu.

Przykłady komunikatów:

- achieve – nadawca chce, aby odbiorca coś spowodował,
- advertise – nadawca ogłasza, że może realizować usługę,
- ask-one – nadawca prosi odbiorcę o odpowiedź na pytanie,
- ask-all – nadawca pyta wszystkich,
- evaluate – nadawca prosi odbiorcę o przeliczenie wyrażenia.

Oto prosty przykład komunikatu w KQML:

{ask-one :sender A :receiver B :language KIF :content (jedzie DWR2043 ?x)}.


5.2.3. Koordynacja współpracy, rozwiązywanie konfliktów i planowanie

W przypadku systemów wieloagentowych, zachodzi zazwyczaj potrzeba koordynacji działań poszczególnych agentów. Zabiegi te nie służą do osiągnięcia celu, jednak znacznie usprawniają jego osiągnięcie.

Jednym ze sposobów koordynowania działań są sieci kontraktowe. Działają one w oparciu o zasady regulujące rzeczywisty rynek przetargów i ofert współpracy. Komunikacja pomiędzy klientem (menadżerem) a dostawcą (kontrahentem) jest realizowana poprzez wymianę komunikatów żądających oraz dostarczających konkretne oferty i ich ewolucję. Podczas żądania wyróżniane są cztery fazy:
5.2. Systemy wieloagentowe

- rozgłoszanie przez menadżera treści zadania do tych kontrahentów, którzy jego zdaniem są zdolni wykonać dane zadanie,
- przesyłanie do menadżera ofert kontrahentów, którzy mają możliwość wyko-
nania zadania,
- analiza przez menadżera otrzymanych ofert i wybór jednego kontrahenta do wspó-
łpracy,
- wysłanie potwierdzenia przez kontrahenta do gotowości wykonania zadań lub
w przeciwnym wypadku powrót do któregoś z wcześniejszych punktów.

Agenci mogą uczestniczyć w wielu, równolegle odbywających się przetargach.
W jednym czasie dany agent może być zatem zarówno kontrahentem jednego
przetargu jak i menadżerem innego. Przedstawione podejście ma zarówno dobre jak i złe strony. Do zalet sieci kontraktowych należą:

- łatwość modelowania i projektowania w systemach wieloagentowych,
- brak konieczności tworzenia dodatkowych mechanizmów sterujących,
- możliwość wyboru kontrahenta po uwzględnieniu wielu różnych czynników, np. czas, koszt realizacji.

Do wad tego podejścia należą:

- duża liczba przesyłanych komunikatów,
- potrzebna struktura obsługująca równolegle realizowane procesy w przypadku
dużej ilości jednocześnie ogłaszanych ofert,
- konieczność precyzyjnego podziału zadania na oferty.

Innym sposobem koordynowania działań są systemy tablicowe. Składają się
one z niezależnych, nie komunikujących się bezpośrednio ze sobą modułów,
ktoż dzielą dane potrzebne do rozwiązania konkretnego problemu przy pomocy
struktury zwanej tablicą. Na ogólny model systemu tablicowego składają się trzy
podsystemy:

- źródło wiedzy – tworzonych przez niezależnych ekspertów,
- tablicy – struktury przechowującej tymczasowe dane i fazy rozwiązywania pro-
błemu,
- sterowania – modułu zarządzającego dostępem do tablicy i jej zawartością.

Do zalet systemu tablicowego należą:

- elastyczność definiowania poszczególnych modułów,
- eksperti mogą reprezentować różne dziedziny wiedzy i posługiwać się różnymi
pojęciami opisującymi tę wiedzę,
- centralizacja sterowania pozwala na efektywne podejście do dynamicznie po-
jawiających się rozwiązań,
- system ten może być traktowany jako bazowy i pozwala na projektować i modelo-
wane różne architektury agentowe.

Główną wadą systemów tablicowych jest ich duży koszt realizacji i dość niska
(w porównaniu do nakładów tworzenia) efektywność tych systemów.
5. Programowanie agentowe w eksploracji danych

Jak już wcześniej wspomniano, współpracę agentów dzieli się na dwa podstawowe typy: kooperację i współzawodnictwo. W skład systemu prezentowanego w pracy wchodzą agenci kooperujący. W związku z tym dzielą się pomiędzy sobą zadania i wykonują je równolegle i rozproszenie tak, aby razem otrzymać żądane wyniki. Nadrzędnym zagadnieniem jest tutaj planowanie działania, którego etapy przedstawiają się następująco:

• dekompozycja zadań na podproblemy,
• przydział podproblemów poszczególnym agentom i grupom agentów,
• tworzenie rozwiązań podproblemów,
• łączenie rozwiązań.

Planowanie w ogólności tworzą dwa etapy: planowanie działania oraz przede wszystkim ustalenie celów bez których działanie jest bezpodstawne. Można wyróżnić cele nadrzędne, zazwyczaj definiowane przez użytkownika systemu oraz cele podrzędne, które wynikają z dekompozycji zadań na podproblemy. System wieloagentowy powinien posiadać zdolność definiowania przynajmniej celów podrzędnych.

5.3. Koncepcja systemu


5.3.1. Cele systemu i funkcjonalności

W definiowaniu celów systemu należy przede wszystkim sformułować pytania lub ich zakres, na które oczekuje się odpowiedzi oraz ustalić źródła danych wykorzystywanych w pozyskiwaniu wiedzy. System ma odpowiedzieć na pytania dotyczące możliwości przejazdu z otoczenia miasta A do otoczenia miasta B w nienieznanej niekoniecznie ścisłe określonym terminie z uwzględnieniem przyjętych kryteriów. Przykładowe pytania i oczekiwane typy odpowiedzi przedstawiono w tab. 5.1. Ze względu na ograniczenie dziedziny, pytania mają charakter zamknięty i składają się z następujących elementów:
5.3. Koncepcja systemu

- miejsce wyjazdu, może być zadane jako otoczenie pewnej miejscowości o dowolnym promieniu lub domyslnie miejscowość, w której znajduje się użytkownik,
- miejsce docelowe, sprecyzowane jak wyżej,
- data (okres) wyjazdu lub przyjazdu, domyslnie będzie to najszybsza możliwa data wyjazdu,
- zakres ceny przejazdu,
- rodzaj samochodu,
- wiek, płeć i staż kierowcy,
- uwzględnienie kryterium minimalizacji kosztów i/lub czasu podróży, i/lub ilości przesiadek.

Odpowiedzi systemu będą listami ofert przejazdów uszeregowane w kolejności od najbardziej do najmniej adekwatnych do indywidualnych wymagań użytkownika. W tym celu wprowadzono współczynnik atrakcyjności, obrazujący stopień w jakim agent ocenia dopasowanie odpowiedzi do profilu pytania użytkownika. Dodatkowo wprowadzono współczynnik zadowolenia, który będzie informacją zwrotną dla agenta o skuteczności jego działania. Źródłami danych, z któ-

**Tab. 5.1: Przykładowe pytania i oczekiwane typy odpowiedzi w systemie.**

<table>
<thead>
<tr>
<th>Nr</th>
<th>Pytanie</th>
<th>Odpowiedź</th>
</tr>
</thead>
<tbody>
<tr>
<td>1.</td>
<td>Jakie są możliwości dojazdu z Wrocławia do Warszawy przed świętami?</td>
<td>Lista ofert przejazdów z Wrocławia do Warszawy w okresie kilku dni przed świętami.</td>
</tr>
<tr>
<td>2.</td>
<td>Jak mogę się dostać dziś wieczorem do Gdańska z okolic Szczecina?</td>
<td>Lista ofert przejazdów z otoczenia Szczecina do Gdańska, z czasem przyjazdu do dziś do północy.</td>
</tr>
<tr>
<td>5.</td>
<td>Jakie są możliwości dojazdu z Wrocławia do Warszawy dziś? Może być z przesiadkami.</td>
<td>Lista ofert przejazdów z Wrocławia do Warszawy w dniu dzisiejszym z uwzględnieniem ewentualnych przesiadek</td>
</tr>
</tbody>
</table>


System wyposażony został w bazę zawierającą adresy takich serwisów. Dodatkowo Agenci Indeksujący na podstawie własnych przekonań (zbioru słów klu-
5. Programowanie agentowe w eksplozacji danych

czowych powiązanych z dziedziną i relacji między nimi) będą aktualizować bazę danych, przeszukiując Internet i dodając nowe strony z danymi.

5.3.2. Opis działania systemu

Pracę systemu można podzielić na dwa niezależne nurty: wyszukiwanie połączeń oraz uaktualnianie bazy serwisów z ofertami przejazdów. Zasadniczą rolą systemu jest udzielanie odpowiedzi na pytania użytkownika. Schemat systemu pokazano na rys. 5.1.

![Rys. 5.1: Schemat systemu.](image)

Na rys. 5.2 przedstawiono fizyczne rozmieszczenie agentów. Agent Koordynujący i indeksujący będzie uruchamiany na serwerze systemu. Razem z nimi automatycznie tworzona jest trójka agentów RMA, ams i df, które odpowiadają za poprawne działanie platformy. Agent Wyszukujący będzie znajdował się na serwerach systemów oferujących ogłoszenia, a agent personalny – na komputerze użytkownika.

Pracę systemu można podzielić na następujące rodzaje działania, w kolejności opisującej proces pozyskiwania odpowiedzi na zadane pytanie:

- zadanie pytania przez użytkownika,
- odbiór i interpretacja pytania przez Agenta Personalnego,
- przekazanie zlecenia Agentowi Koordynującemu,
- rozdzielenie zadań na Agentów Wyszukujących,
5.3. Koncepcja systemu


5.3.3. Opis Agentów

Charakterystyki agentów pracujących w systemie zamieszczono w tab. 5.2.
5. Programowanie agentowe w eksplozacji danych

W systemie, ze względu na jego niską złożoność, zdecydowano się użyć systemu tablicowego. Po otrzymaniu zadania Agent Koordynujący bazujący na liście dostępnych serwisów rozdziela je między Agentami Wyszukującymi. Następnie tworzy listę odpowiedzi usuwając powtarzające się wpisy. Agenci Indeksujący działają w sposób praktycznie niezależny i ciągły, przeszukując cały internet. Taki sposób przydzielania zadań nie zmniejszy ilości występujących w systemie konfliktów. Zasadniczo w systemie przewidziano następujące rodzaje konfliktów:

- jednoczesne żądanie dostępu do listy odpowiedzi i bazy adresów serwisów,
- jednoczesne zlecenie dwóch różnych zadań przez Agentów Personalnych,
- zlecenie zadania przez Agenta Personalnego w chwili, gdy nie ma wolnych Agentów Wyszukujących,
- żądanie obsługi przez użytkownika w przypadku braku wolnych Agentów Personalnych.

Tab. 5.2: Opis agentów

<table>
<thead>
<tr>
<th>Nazwa agenta</th>
<th>Rodzaj</th>
<th>Zadania</th>
</tr>
</thead>
</table>
| Personalny   | instytucjonalny | • odbieranie pytań  
• interpretacja pytań  
• budowanie profilu użytkownika  
• przygotowanie odpowiedzi na podstawie profilu użytkownika |
| Wyszukujący  | reaktywny   | • wyszukiwanie ofert przejazdów na podstawie specyfikacji  
• posługiwanie się mapą odległości i kalendarzem w realizacji zadań |
| Koordynujący | socjalny | • koordynacja pracą agentów w systemie  
• planowanie i realizacja planów  
• specyfikacja zadań  
• generowanie celów działania  
• dokonywanie zmian w przekonaniach agentów na podstawie nowych informacji o świecie |
| Indeksujący  | instytucjonalny | • poszukiwanie serwisów z ofertami przejazdów  
• aktualizowanie bazy danych z adresami serwisów  
• aktualizowanie bazy słów kluczowych |

W przypadku żądania dostępu do listy odpowiedzi i bazy adresów serwisów przez dwóch lub więcej Agentów Wyszukujących czy Indeksujących, Agent Koordynujący przydziela zasoby w kolejności zależnej od ich identyfikatora. Dostęp do bazy będzie trwał przez ustalony czas, a następnie Agent Koordynujący będzie odbierał przydzielone zasoby i przekazywał do następnego agenta szerego-
5.4. Realizacja systemu

5.4.1. Dobór narzędzi, urządzeń i protokołów


Do komunikacji między agentami udostępniony został język ACL (ang. Agent Communication Language). Jest to specjalna klasa, która posiada szereg metod umożliwiających wygodne konstruowanie i przetwarzanie treści komunikatów. Aby wysłać komunikat należy wypełnić odpowiednie atrybuty, a następnie wywołać metodę send(). Agent odbierający daną wiadomość używa metody receive(). Do postawowych atrybutów należą:

- get/setPerformative() – pobiera/ustawia performatywy,
- get/setSender() – pobiera/ustawia nadawcę,
- add/getAllReceiver() – dodaje/pobiera wszystkich odbiorców,
- get/setLanguage() – pobiera/ustawia język komunikatu,
- get/setOntology() – pobiera/ustawia ontologię.
5. Programowanie agentowe w eksploracji danych

- get/setContent() – pobiera/ustawia komunikat.

Każy agent posiada swoją unikalną nazwą postaci:
<nazwa-agenta>@<nazwa-hosta>:<nr-portu>/JADE


5.4.2. Przykłady implementacji

Poniżej zamieszczone zostały przykłady implementacji klas agenta oraz zachowania.

- Klasa przykładowego agenta:

```java
package Agent;

// ładujemy bibliotekę agenta
import jade.core.Agent;

// klasa przykładowego agenta
public class PrzykladowyAgent extends Agent {
    // metoda wywoływana przy inicjalizacji agenta
    protected void init() {
        System.out.println("PrzykladowyAgent zostal uruchomiony.");
        addBehaviour(new PrzykladoweZachowanie());
    }

    // metoda wywoływana przy kończeniu życia agenta (po wykonaniu
    // na niej metody doDelete())
    protected void takeDown() {
        System.out.println("PrzykladowyAgent konczy działanie.");
    }
}
```

- Klasa przykładowego zachowania:

```java
package Agent;

// ładujemy bibliotekę z zachowaniami
import jade.core.behaviours.*;

// klasa przykładowego zachowania
public class PrzykladoweZachowanie extends Behaviour {
    int licznik = 0;

    // metoda opisująca zachowanie agenta
    public void action() {
        // pole myAgent zawiera wskaźnik agenta, do którego odnosi się
        // aktualne zachowanie
```
5.4. Realizacja systemu

```java
System.out.println("Agent\" + myAgent.getName() + "\wykonuje\"+\"twoje\"+\"zachowanie.\")
}

// metoda informująca czy zakończyć wykonywanie danego zachowania
public boolean done()
{
    if (++licznik == 3)
    {
        myAgent.doDelete();
        return true;
    }
    return false;
}
}
```

Po uruchomieniu środowiska agentowego jade.Boot oraz przykładowego agenta poleceniem:

```
java jade.Boot <nazwa-agenta>:<pakiet>.<klasa-agenta>(argumenty),
```

otrzymuje się informacje o uruchomieniu agenta. Następnie zaimplementowane zachowanie jest wywoływane 3 razy, po czym działanie agenta zostaje zakończone.

5.4.3. Opis instalacji

Poniżej opisano sposób instalacji na serwerze systemu oraz na komputerze użytkownika będącym klientem. We wszystkich przypadkach należy w zmiennej systemowej CLASSPATH dodać ścieżkę z bibliotekami Jade, np. ./jade/lib/jade.jar.

Instalacja i uruchomienie środowiska i Agenta Koordynującego na serwerze przebiega w następujących krokach:

- skopiowanie katalogu projekt do dowolnego katalogu,
- wywołanie z linii polecenia
  ```
  java jade.Boot -gui AgentKoordynujacy:Agent.AgentKoordynujacy
  ```

Instalacja i uruchomienie Agenta Personalnego na komputerze użytkownika przebiega w następujących krokach:

- skopiowanie katalogu projekt do dowolnego katalogu,
- wywołanie polecenia z java Gui.Gui linię poleceń,
- ustawienie adresu hosta z Agentem Koordynującym poprzez wybranie z Menu głównego zakładki Opcje.

**Literatura**

5. Programowanie agentowe w eksploracji danych


6.1. Wstęp

Postęp technologiczny w zakresie cyfrowego generowania i gromadzenia informacji doprowadził do przekształcenia się baz danych wielu przedsiębiorstw, urzędów i placówek badawczych w zbiorniki ogromnych ilości danych. Wraz z gwałtownym wzrostem ilości gromadzonych danych coraz trudniej jest je analizować i rozumieć. Koniecznym stało się wprowadzenie metod pozwalających na ich sensowne przetwarzanie. Doprowadziło to do powstania teorii i narzędzi, które zebrać można pod jedynym mianownikiem: eksploracja danych KDD (ang. knowledge discovery in databases) [1].

Podstawowym problemem KDD jest przetwarzanie dużej ilości danych w formy bardziej zwięzłe (np. raport) lub użyteczne (np. estymowanie wartości dla przyszłych przypadków). Eksploracja danych jest procesem zautomatyzowanym, w którym poszukiwane są nietrywialne, dotychczas nieznane, potencjalnie użyteczne reguły, zależności itd. w dużych repozytoriach danych. Jego celem jest analiza procesów w celu lepszego ich rozumienia. W procesie odkrywania wiedzy wyróżnia się następujące etapy (rys. 6.1):

- Selekcja danych (ang. data selection) - wybieranie tych danych z bazy danych, które są istotne dla zadań analizy.
- Przetwarzanie wstępne (ang. data preprocessing) - przygotowanie danych, usuwanie „zanieczyszczeń” i niespójności w danych itp.
- Transformacja danych (ang. data transformation) – przekształcanie i konsolidowanie danych do postaci przydatnej dla eksploracji, na przykład ich sumowanie i/lub agregowanie (np. w hurtowni danych).
- Eksploracja danych (ang. data mining) – stosowanie „inteligentnych” metod w celu odkrycia istotnych zależności zwanych wzorcami (ang. patterns).
- Ocena wzorców (ang. pattern evaluation) – identyfikacja naprawdę interesujących wzorców w oparciu o pewne miary ważności.
6. Metody drążenie danych i ich zastosowania

Rys. 6.1: Kroki w procesie eksploracji danych (wg [1]).

- Reprezentacja wiedzy (ang. knowledge presentation) – przedstawienie odkrytej wiedzy użytkownikowi za pomocą technik wizualizacji i reprezentacji wiedzy.

Eksploracja danych (inaczej też drążenie danych) może być wykonywana za pomocą osobnej aplikacji lub algorytmu wbudowanego w proces KDD. Prowadzi ona, przy określonych ograniczeniach obliczeniowych, do znalezienia wzorców w danych (których może być nieskończoność wiele). Ważnym czynnikiem w drążeniu danych jest określenie celu drążenia. Można wyróżnić dwa typy celów:
- weryfikacja – system jest ograniczony do weryfikacji hipotezy użytkownika,
- odkrycie – system odkrywa nowe wzorce

6.2. Metody drążenie danych

Większość metod używanych w drążeniu danych pochodzi z wypróbowanych i przetestowanych metod takich jak uczenie maszynowe, rozpoznawania wzorców, regresji i statystyki. Należy podkreślić, iż w skład wielu metod drążenia danych spotykanych w literaturze wchodzi tylko parę fundamentalnych technik: klasifikacja/regresja, grupowanie, odkrywanie sekwencji, odkrywanie charakterystyk, analiza przebiegów czasowych, odkrywanie asocjacji.

6.2.1. Klasifikacja

Jedną z najstarszych jak również najważniejszych metod eksplozacji danych o istotnym znaczeniu praktycznym, jest metoda klasifikacji. Polecą ona na znajdywaniu odwzorowań danych w zbior predefiniowanych klas. Na podstawie zawartości bazy danych budowany jest model (np. drzewo decyzyjne, reguły logiczne), który służy do klasifikowania nowych obiektów w bazie danych lub głębszego zrozumienia istniejącego podziału obiektów na predefiniowane klasy (rys. 6.2a). Klasifikacja znalazła szereg zastosowań np.: rozpoznawanie trendów na rynkach finansowych, automatyczne rozpoznawanie obiektów w dużych bazach danych obrazów, wspomaganie decyzji przyznawania kredytów bankowych. Ogromne zastosowanie znalazła w systemach medycznych. Przykładowo, w b-
6.2. Metody drążenie danych

zie danych medycznych znalezione mogą być reguły klasyfikujące poszczególne schorzenia, a następnie przy pomocy znalezionych reguł automatycznie może być przeprowadzone diagnozowanie kolejnych pacjentów.

![Rys. 6.2: Problem a) klasyfikacji, b) grupowania.](image)

Klasyfikacja jest metodą eksploracji danych, która może odbywać się w sposób nadzorowany (z nauczycielem). Proces klasyfikacji składa się z kilku etapów, od budowania modelu, przez fazę testowania aż po predykcję nieznanych wartości. Głównym celem klasyfikacji jest zbudowanie formalnego modelu zwanego klasyfikatorem. Danymi wejściowymi w procesie klasyfikacji są krotki należące do zbioru treningowego (przykłady, obserwacje, próbki). Krotki składają się z listy wartości atrybutów opisowych (tzw. deskryptorów) i wybranego atrybutu decyzyjnego (ang. *class label attribute*). Wynikiem procesu klasyfikacji jest pewien otrzymany model (klasyfikator), który przydziela każdej krotce (przykładowi) wartość atrybutu decyzyjnego w oparciu o wartości pozostałych atrybutów (deskryptorów).

6.2.2. Grupowanie

Kolejną klasyczną metodą jest grupowanie (klastrowanie). Obejmuje ono metody analizy danych i znajdowania skończonych zbiorów klas obiektów posiadających podobne cechy. W przeciwieństwie do metod klasyfikacji i predykcji, klasyfikacja obiektów (podział na klasy) nie jest znana a-priori, lecz jest celem metod grupowania. Metody te grupują obiekty w klasę w taki sposób, aby maksymalizować wewnętrzklasowe podobieństwo obiektów i minimalizować podobieństwo pomiędzy klasami obiektów (rys. 6.2b). Grupowanie znalazło szereg zastosowań w różnych dziedzinach życia, np. grupowanie dokumentów, grupowanie klientów czy określenia segmentacji rynku.

6.2.3. Odkrywanie asocjacji

Odkrywanie asocjacji jest jedną z najciekawszych i najbardziej popularnych technik eksploracji danych. Celem procesu odkrywania asocjacji jest znalezienie
6. Metody drążenie danych i ich zastosowania

interesujących zależności lub korelacji, nazywanych ogólnie asocjacjami, pomiędzy danymi w dużych zbiorach danych. Wynikiem procesu odkrywania asocjacji jest zbiór reguł asocjacyjnych, opisujących znalezione zależności lub korelacje między danymi. Sztandarowym przykładem reguły asocjacyjnej jest reguła wygenerowana w odniesieniu do bazy danych supermarketu: „klienci, którzy kupują pieluszki, kupują również piwo”. Celem tej analizy jest znalezienie naturalnych wzorców zachowań konsumenckich klientów poprzez analizę produktów, które są przez klientów supermarketu kupowane najczęściej wspólnie np.: „klienci, którzy kupują chleb, masło i ser, kupują również wodę mineralną i ketchup”.

6.2.4. Odkrywanie sekwencji

Wzorce sekwencji stanowią ważną klasę wzorców symbolicznych opisujących zależności występujące pomiędzy zdarzeniami zachodzącymi w pewnym przedziale czasu. W przypadku wzorców symbolicznych zdarzenia są opisane wartościami atrybutów kategorycznych. W przypadku, gdy zdarzenia są opisane wartościami numerycznymi mowa jest o przebiegach czasowych lub o analizie trendów. W przypadku analizy trendów, najczęściej stosuje się metody analizy przebiegów czasowych lub metody predykcji.

Problem odkrywania wzorców sekwencji polega, najogólniej mówiąc, na analizie bazy danych zawierającej informacje o zdarzeniach, które wystąpiły w określonym przedziale czasu, w celu znalezienia zależności pomiędzy występowaniem określonych zdarzeń w czasie. Przykładem wzorca sekwencji jest kurs akcji BPH, który podczas ostatnich trzech sesji wzrósł o 0.5%, 0.9%, 0.1%, na następnej sesji spadnie o 0.5%.

6.3. Opis problemu i jego rozwiązania

Studia w dziedzinie inwestowania na rynku akcji należą do jednej z najbardziej rozwijanej dziedziny drążenia danych. Ryzyko które każdy inwestor ponosi każdego dnia może zostać minimalizowane poprzez analizę archiwów notowań w celu znalezienia powtarzających się wzorców, pomocnych w podejmowaniu decyzji. Występowanie cykli ułatwia analizę giełdy przy pomocy różnego rodzaju oscylatorów, sieci neuronowych itd. W celach testowych podjęto budowę narzędzia do analizy jednego z indeksów Giełda Papierów Wartościowych w Warszawie w celu znalezienia prawidłowości i wzorców. W tym celu posłużono się analizą opartą na wielomianach i mierzyć ich podobieństwa [2].

Metoda Mierzenie podobieństwa zachowania giełdy w oparciu o deskryptory wielokątów wykonywana jest zasadniczo w dwóch etapach:

- dane są wczytywane i transformowane (modelowane) do postaci deskryptorów wielokątów,
- wyliczany jest koszt transformacji z jednego wielokąta do drugiego.

Otrzymane rezultaty określają podobieństwo między poszczególnymi danymi. Jest ono podstawą do określenia, w jakim cyklu znajduje się aktualna wycena spółek i ewentualnie podjęcia decyzji o zakupie papierów wartościowych. Me-
6.3. Opis problemu i jego rozwiązania
toda ta, w przeciwieństwie do metod korzystających z zaawansowanych ilościowych wskaźników probabilistycznych, jest stosunkowo prosta. Pozwala ona na przewidywanie cykli biznesowych w długim terminie, a także jest intuicyjna dla przedsiębiorców, gdyż uzyskane wyniki (podobieństwa) mogą oni skonfrontować ze swoimi doświadczeniami i intuicjami.

6.3.1. Deskryptory wielokątowe

Deskryptory wielokątowe zawiera i charakteryzuje zależności pomiędzy danymi, tj. podobieństwo dwóch deskryptorów odzwierciedla zmienne zależności pomiędzy dystrybucjami danych. Ich wybór jest podyktowany tym, iż dane giełdowe ze swej przypominają dane losowe lub szum, stąd trudno jest je zamódlować przy pomocy funkcji liniowej lub nieliniowej.

W celu reprezentowania dystrybucji danych, deskryptor wielokątowy łączy kilka wypukłych wielokątów. Pojedynczy, wypukły wielokąt może być opisany poprzez punkt odniesienia wewnątrz figury oraz 

\[ N \]

osi wskazujące kierunki normalne oraz dystans od punktu odniesienia do krawędzi.

Bazując na statystycznej charakterystyce dystrybucji danych, deskryptor wielokątowy może być uczony iteracyjnie. Punkt odniesienia \( C \) wyliczony jest za pomocą reguły

\[ C = \operatorname{arg\,min}_{P_i \in P} \left( \sum_{P_j \in P} \operatorname{dist}(P_i, P_j) \right), \quad (6.1) \]

gdzie \( P_i \) oraz \( P_j \) są punktami z \( P \) i \( \operatorname{dist}(P_i, P_j) \) jest odległością między \( P_i \) i \( P_j \).

Wykorzystując punkt \( C \) oraz losowo inicjowane osie \( (A_i, i = 1, \ldots, N) \), punkty \( P \) są klaszteryzowaniu zgodnie z regułą:

\[ W = \operatorname{arg\,max}_{W_j} A_j \cdot (P_i - C) \quad \frac{1}{||A_j||^2}, \quad (6.2) \]

gdzie \( A_j \) jest osią klastra \( W_j \), gdzie \( j = 1, \ldots, N \). Następnie każda os \( A_j \) jest doprecyzowywany przez punkty z klastra \( W_j \). Niech \( D \) będzie wymiarem punktów danych i dzieli klastry na \( D \) części za pomocą hiperpłaszczyzny przechodzącą przez \( C \). Następnie liczona jest średnia z punktów każdego podklastra i jest generowana płaszczyzna, która przechodzi przez wszystkie \( D \) średnich punktów. Ustalony jest nowy kierunek osi na ortogonalny do hiperpłaszczyzny, a jej długość – na odległość punktu \( C \) od niej. Proces jest powtarzany dopóki osie się nie zbiegają.

6.3.2. Deformacje wielokątów

W celu opisu różnicy między dwoma deskryptorami wielokątowymi, wprowadzone zostało pojęcie dystansu deformacji, oznaczający minimalny całkowity koszt operatorów transformujący jeden wielokąt w drugi. Zostały zdefiniowane następujące operatorzy:

- spłaszczający - usuwa kąt \( a_i \) z listy,
- dodający - wstawia kąt do listy za kątem \( a_i \),

93
6. Metody drążenie danych i ich zastosowania

- wyostrzający - zwiększa kąt \( a_i \) o \( \delta \) poprzez zmniejszenie \( a_{i-1} \) i zwiększenie \( a_{i+1} \) o \( \delta /2 \),
- obracający - obraca wielokąt o kąt \( \delta \),
- rozszerzający - rozszerza \( i \)-tą oś o \( \delta \).

W celu uproszczenia obliczeń minimalnego dystansu, wielokąty są reprezentowane poprzez sekwencje kątów, oraz nakładane są ograniczenia na kolejność wykorzystywania operatorów w trakcie procesu deformowania.

6.3.3. Szacowanie dystansu deformacji

Problem szukania najkrótszej możliwej drogi od przekształcenia jednego wielokąta do drugiego może zostać sprowadzony do postaci problemu znalezienia odległości między dwoma ciągami znaków. Napisy są dzielone na podnapisy, tak, że najkrótszy możliwy dystans przejścia może zostać wyliczony bezpośrednio. Następnie szukane jest globalne minimum ilości potrzebnych transformacji, w celu przejścia od jednego napisu do drugiego.

6.3.4. Źródło danych

W celu przeprowadzenia analizy, niezbędne jest posiadanie odpowiednich danych. Będą to dane indeksu giełdowego MWIG40, nowotowanego na GPW i obrazujących kondycję oraz aktualny sentyment rynku do spółek średniej wielkości. Dobór wynika z trudności obrazowania średniej oraz długoterminowej tendencji na rynku oraz jest mniej wrażliwy na nastroje panujące na giełdach światowych. Dane te przyjęto do zmodelowania deskryptorów wielokątów i przeprowadzenia obliczeń mających za zadanie zweryfikować skuteczność obranej metody.

Literatura

7.1. Wstęp


Sednem dopasowywania wzorców jest odpowiednie zaprojektowanie szablonów. Jeśli dane mają być sensownie przetwarzane, przy tworzeniu wzorców konieczna jest znajomość formatu pliku z danymi wejściowymi. Format niezgodny z oczekiwaniami może prowadzić do powstania błędnych wyników. Nie wszystkie dostępne silniki (programy, które przetwarzają dane z wykorzystaniem z szablonów) umożliwiają kontrolę poprawności danych wejściowych. Jest to dość oczywiste, ponieważ uniwersalność silnika stoi poniekąd w sprzeczności z kontrolą formatu pliku wejściowego.

Zdefiniowanie formatu pliku wejściowego zazwyczaj nakłada liczne ograniczenia na sposób zapisu danych. Jak się okazuje, pomimo ograniczeń, pliki ze zdefiniowanym formatem mogą być bardzo funkcjonalne. Świadczy o tym choćby popularność i wszechobecnosc formatu XML.

Standardem przetwarzania plików XML jest technologia XSLT. Znajduje on zastosowanie przy tworzeniu stron internetowych, gdzie umożliwia automatyczne formatowanie wyświetlanych danych. Umożliwia również konwersję danych do innych formatów opartych na XML i nie tylko.
7. Transformacja danych za pomocą szablonów

Inną metodę przetwarzania danych dostarcza język AWK. Narzędziem opartym na tym języku jest program AWK, lub jego wersja na licencji GNU - GAWK. Program ten nie nakłada ograniczeń dotyczących formatów plików wejściowych i wyjściowych.

Niniejszy rozdział poświęcono opisowi dwóch narzędzi przeznaczonych to szablonowego przetwarzania danych oraz prezentacji paru ciekawszych możliwości tej metody. Mimo że mechanizm przetwarzania szablonowego wydaje się dość proste, daje duże możliwości, jak choćby automatyczne generowanie kodu. Przykładem, na którym testowano ten typ przetwarzania był program do różniczkowania symbolicznego funkcji. Opracowane rozwiązanie zrealizowano z wykorzystaniem GAWK oraz XSLT. Pozwala ono wczytać dane (tj. wzory) zapisane w formacie \LaTeX{}, oraz produkuje wynik różniczkowania również w formacie \LaTeX{}.

7.2. XSLT

XSLT (ang. XSL Transformations, Extensible Stylesheet Language Transformations), jest opartym na XML językiem przekształceń dla dokumentów XML. Jest on stworzony do szablonowego przetwarzania danych, więc wraz z przetwarzanym dokumentem XML musi być dostarczony szablon. Sam szablon jest poprawnym dokumentem XML. Proces transformacji pokazano na rys. 7.1. Format pliku wejściowego musi być zgodny z XML, natomiast formatem wyjściowym może być: XSL-FO, HTML/XHTML, dowolny XML (w tym zgodny z XSLT), dowolny tekst (np. inny format tekstowy taki jak \LaTeX{}), pliki grafiki wektorowej w formacie SVG, wzory matematyczne MathML, dokumenty PDF, ODF.

Rys. 7.1: Proces transformacji XSLT.

XSLT jest rozwijany przez W3C, jako część rodziny języków XSL, aktualną wersją jest XSLT 2.0. Dzięki łatwości implementacji i powszechnemu stosowaniu XML jako standardu zapisu informacji, XSLT stał się narzędziem stosowanym w wielu rodzajach oprogramowania, najpopularniejsze to generowanie stron WWW w HTML i XHTML oraz konwersja między alternatywnymi formatami.

Najpopularniejsze procesory XSLT to: Saxon, Xalan-Java/C++, xsltproc, XT, sablotron, MS msxml, PHP 5 (funkcje xslt), przeglądarki WWW.
7.2. Algorytm transformacji

- Przygotowanie do transformacji:
  - Parsowanie arkusza XSLT i wejściowego XML, budowanie ich drzew.
  - Usunięcie nadmiarowych białych znaków.
  - Dołączenie standardowych reguł do drzewa XSLT.
- Transformacja:
  - Tworzenie głównego elementu drzewa wyjściowego
  - Przetworzone zostają elementy drzewa wejściowego, zaczynając od elementu głównego.
  - Następuje zwrócenie drzewa wyjściowego.


7.2.2. Podsumowanie

Stosowanie XSLT pozwala na wiele operacji takich jak:

- generacja tekstu statycznego (np. „Rozdział”)
- sortowanie elementów
- obliczenia arytmetyczne i logiczne
- opuszczanie fragmentów dokumentu
- usuwanie białych znaków
- przemieszczanie fragmentów tekstów
- zamiana kolejności
- transformacja drzewa dokumentu
- instrukcje warunkowe i pętle
- powielanie fragmentów

W przeciwieństwie do GAWK, XSLT służy do przetwarzania tylko jednego typu dokumentów - XML. Zaletą rekomponującą tę niedogodność jest możliwość sprawdzenia struktury danych wejściowych. Duże możliwości operowania na przetwarzanych plikach (w tym transformacje drzew dokumentu - których nie wykonamy za pomocą arkuszy CSS), bardzo duża popularność, sprawiają że ten sposób przetwarzania szablonowego jest nadal rozwijany i staje się standardem w zastosowaniach WWW i nie tylko.
7. Transformacja danych za pomocą szablonów

7.3. GAWK


Szablony przetwarzania mogą być zadawane na 3 sposoby: z wiersza poleceń, jako osobny plik, lub, dzięki popularnemu w powłokach linuxowych mechem #!, wywołanie programu może ograniczyć się do uruchomienia odpowiedniego skryptu, w którym podany jest szablon przetwarzania.

Schemat szablonu jest bardzo prosty

wzorzec { akcja }

Każda para wzorzec-akcja musi być zapisana w osobnym wierszu. Akcja podejmowana jest w przypadku dopasowania wzorca do przetwarzanego rekordu.

7.3.1. Wzorce

Język udostępnia 7 rodzajów wzorców:

• BEGIN - jest wzorcem, który nie zostaje dopasowywany do danych wejściowych. Akcja przypisana temu wzorcowi wykonywana jest przed odczytaniem danych wejściowych,
• END - wzorzec podobny do BEGIN, z tym, że przypisana mu akcja zostaje wykonywana po przetworzeniu wszystkich danych wejściowych,
• wyrażenie - akcja jest wykonywana, gdy przetwarzany rekord jest równy podanemu wyrażeniu,
• /wyrażenie regularne/ - akcja jest wykonywana, gdy przetwarzany rekord jest równy podanemu wyrażeniu regularnemu,
• wzorzec złożony - są to wzorce połączone operatorami,
• wzorzec1, wzorzec2 - wzorzec powoduje dopasowanie wszystkich rekordów od rekordu dopasowanego do wzorzec1, aż do rekordu dopasowanego do wzorzec2,
• brak wzorca - powoduje wykonanie akcji niezależnie od rekordu.

7.3.2. Akcje

W języku AWK akcjami są wszystkie instrukcje. Język udostępnia wbudowane funkcje i umożliwia tworzenie własnych. Dostępne akcje to:

• brak akcji,
• operacje arytmetyczne, w tym również zmiennoprzecinkowe,
• operacje na łańcuchach,
• operacje wejścia - wyjścia,
• operacje na czasie,
• operacje na bitach,
• instrukcje warunkowe i pętle,
7.4. Systemy produkcyjne

- instrukcje sterujące przetwarzaniem - system(cmd-line) - umożliwia wyporządkowanie przetwarzania bieżącego rekordu lub pliku,

7.3.3. Zmienne

Język przechowuje wiele parametrów w predefiniowanych zmiennych. Niektóre z nich to:

- FILENAME - nazwa aktualnie przetwarzanego pliku wejściowego,
- FNR - liczba rekordów w bieżącym pliku,
- IGNORECASE - ignorowanie wielkości liter przy dopasowywaniu,
- FIELDWIDTHS - powoduje podział rekordu na pola o szerokościach podanych w zmiennej,
- FS - separator pol wejściowych,
- NF - liczba pól w bieżącym rekordzie,
- NR - liczba odczytanych dotychczas rekordów,
- RS - separator rekordów wejściowych.

7.3.4. Podsumowanie


7.4. Systemy produkcyjne

System produkcyjny jest metoda reprezentacji wiedzy oparta na "regułach produkcji", będącymi parami: warunek-działanie. Mają one postać:

JEŻELI (IF) przesłanka, TO (THEN) konkluzja

Lewa część reguły określa warunki jej stosowalności (przesłanka), natomiast prawa określa jej działanie (konkluzja). Przesłanka może zawierać większą liczbę zdań połączonych ze sobą funktorami logicznymi, czyli klauzule i spójniki. Klauzulą jest każde zdarzenie opisane w regule, używając spójników I oraz LUB, możemy uzależnić ich wykonanie od kilku faktów, lub od alternatywy tych faktów. Można to przedstawić na przykładzie:

Przykład 1  JEŻELI w przestrzeni roboczej znajduje się człowiek LUB niezwiązany z zadaniem przedmiot TO ustrzymaj pracę.

7.4.1. Budowa systemu

Zarys architektury systemu pokazano na rys. 7.2. Tworzą go: pamięć robocza, interpreter, baz reguł.
7. Transformacja danych za pomocą szablonów

Rys. 7.2: Schemat systemu produkcyjnego [1].

Pamięć robocza to tzw. kontekst, wiedza systemu na temat zadanego problemu. Może to być zarówno prosty zbiór faktów określający stan systemu, jak i skomplikowana struktura danych. Zawartość kontekstu na skutek stosowania reguł zmienia się w czasie, nieaktualne dane zostają kasowane i zastępowane bieżącymi faktami, co jest bodźcem do działania dla systemu.

Interpreter interpretuje fakty i wykonuje odpowiednie czynności z tym związaną. Ten proces, polegający na łączeniu faktów z regułami, zwany jest wnioskowaniem. Wyróżniamy jego dwa typy: wnioskowanie wstecz (dla konkretnych reguł szukane są odpowiednie faktów) oraz wnioskowanie w przód (dla faktów dobierane są reguły). Gdy na początku znanych jest wiele faktów przewaga pokazuje wnioskowanie w przód, jest przy tym łatwiejsze do zaimplementowania oraz czulsze na zmiany w kontekście. Wnioskowanie wstecz jest natomiast bardziej efektywne i posiada bardziej przejrzysty zapis.

Baza reguł determinuje zachowanie systemu. Wszystkie reguły są przetrzymywane w specjalnie do tego celu przeznaczonej pamięci. Nie może być ona modyfikowana w trakcie pracy systemu, stąd jest to tzw. pamięć niepracująca. W przeciwieństwie do procedur, reguły nie odwołują się do siebie, minimalizują więc oddziaływanie między sobą.

7.4.2. Działanie systemu

System działa w pętli, której cykl można przedstawić poniższym schematem:

- skojarzenie przesłanek reguł z zawartością pamięci pracującej,
- zakończenie jeśli nie odpowiada żadna reguła,
- rozwiązanie konfliktu jeśli skojarzono wiele reguł,
- wykonywanie czynności określonej przez konkluzje reguł.

Wystąpienie nowego zdarzenia pobudza system do pracy. W pierwszym etapie działania systemu fakty zostają skojarzone z odpowiednimi przesłankami reguł, jeśli może być zastosowana więcej niż jedna reguła dla danego zdarzenia, zostają usunięte wszystkie których wynikiem byłby powtarzające się konkluzje. Następnie zostaje wybrana jedna z nich, stan systemu zostaje sprawdzony i ewentualnie mogą zostać wykonane pozostałe. Jeśli nie można natomiast zastosować żadnej
reguły dla istniejących zdarzeń, praca systemu zostaje przerwana. Po wybraniu reguły, zostaje wykonana jej konkluzja, stan systemu zostaje zapisany do pamięci pracującej - powstaje nowy fakt. Pobudza to system do pracy i przechodzi on do fazy pierwszej cyklu.

7.4.3. Podsumowanie

Systemy produkcyjne zapewniają naturalność i przejrzystość zapisu w wielu zastosowaniach. Dużą ich zaletą jest modularność, poszczególne reguły produkcyjne nie są powiązane ze sobą (mogą oddziaływać na siebie jedynie za pomocą pamięci roboczej), co sprawia że mogą być one dowolnie: modyfikowane, dodawane, usuwane bez konieczności ingerowania w inne reguły. Jest to możliwe, dzięki oddzieleniu reguł od części wykonawczej - interpretera.

Dużym plusem jest fakt, iż poprzez podgląd pamięci pracującej i znajomość reguł możemy dokładnie przesledzić drogę wnioskowania i uzyskać wyjaśnienie zwróconego przez system wyniku. W dużych systemach, dla kilkuset reguł trudno jest przewidzieć działanie systemu. Budowa reguł w oparciu o strukturę JEŻELI-TO pozwala na szerokie spektrum zastosowań w wielu dziedzinach, takich jak tworzenie sztucznej inteligencji lub reprezentacja wiedzy.

Największą wadą systemów produkcyjnych to trudność w znalezieniu formalnego algorytmu ich działania, gdyż poszczególne reguły nie są sekwencyjne. Znajdują one zastosowanie w różnych dziedzinach, gdzie wiedza w nich zapisana symuluje wiedzę wykwalifikowanych pracowników (tzw. systemy eksperckie).

7.5. Systemy przepisywania termów

Ze swojej natury metoda szablonowego przetwarzania danych idealnie nadaje się do realizacji systemu przepisywania termów.

Zdefiniujmy zbiór termów $T$ dla pewnej struktury $\mathcal{A} = \langle A, \Sigma_f, \Sigma_r \rangle$. Zbiór $T$ jest zbiorem napisów (zatem jego elementy pisane będą w cudzysłowach) reprezentujących funkcje struktury $\mathcal{A}$ oraz zmienne. Zmienne są napisami reprezentującymi dowolne z elementów zbioru $A$ - z dziedziny struktury. Ponadto, jeśli "$t_1", \ldots, "t_n" \in T$, to "$f(t_1, \ldots, t_n)" \in T$.

Przykład 2 Rozważmy strukturę liczb rzeczywistych ze standardowymi operacjami algebraicznymi. Niech $x_i \in \mathbb{R}$, dla $i \in \mathbb{N}$ oznaczają zmienne będące elementami zbioru liczb rzeczywistych i niech $x_i \in T$. Wtedy termami są również na przykład "$\sin(x_1)"", "$x_3 + x_8$", "$x_9^2$". Termom tym odpowiadają kolejno funkcje $\sin(x_1)$, $x_3 + x_8$, $x_9^2$.

Stałe również są termami. Są one traktowane jako napisy oznaczające pewne funkcje zero-argumentowe.

W celu przepisywania termów konieczne jest podanie zbioru aksjomatów równościowych $E$. Jest to zbiór równań. Jeśli równania te są prawdziwe w rozpatrywanej algебrze, to można zastępować wyrażenia z lewej strony równości wyrażeniami z prawej i na odwrot.

101
7. Transformacja danych za pomocą szablonów

Przykład 3 Niech \( f(a, b) = b \) \( \in E \) oraz \( g(b, b) = c' \) \( \in E \). Mamy dany term \( g(b, f(a, b))'' \). Można go zredukować w dwóch krokach do termu \( c' \).


Trudniejszym problemem jest konieczność wielokrotnego przekształcania danych. Wyrażenie otrzymane przez przekształcenie może być ponownie poddawane przekształceniu tak, jak w przykładzie [3]. Działanie takie nie jest standardowe dla silników przetwarzających szablonowo i musi być wymuszone.


7.6. Projekt

Celem projektu jest stworzenie aplikacji, która będzie różniczkowała wyrażenia matematyczne wprowadzone w formacie \( \LaTeX \). Wynik będzie otrzymywany w tym samym formacie.

Transformacja danych realizowana jest w dwóch etapach: transformacji z \( \LaTeX \) do XML przy użyciu GAWK, a następnie konwersji z XML do \( \LaTeX \) w oparciu o szablony XSLT.

7.6.1. Transformacja \( \LaTeX \) do XML

Na format danych wejściowych nałożono pewne ograniczenia. Przede wszystkim ograniczony jest zestaw rozumianych przez interpreter operatorów. Po drugie, określono priorytety działań. Informacje te zebrano w tab. [7.1]. Większa liczba priorytetu oznacza, że działanie wykona się wcześniej. Działania o równym priorytecie wykonywane są od strony lewej do prawej. Ponadto wymaga się, aby każde wyrażenie poprzedzone było przez \( \frac{d}{dx} \) (w zapisie \( \text{\LaTeX} \frac{d}{dx} \)), gdzie \( x \) może być dowolną, pojedynczą zmienną, po której odbędzie się różniczkowanie. Wyrażenie podane na wejście aplikacji będzie różniczkowane aż do końca, niezależnie od nawiasów.
Tab. 7.1: Zestaw dostępnych operatorów i ich priorytet

<table>
<thead>
<tr>
<th>Operator</th>
<th>Opis</th>
<th>Priorytet</th>
</tr>
</thead>
<tbody>
<tr>
<td>(, ) \left (, \right )</td>
<td>nawiasy</td>
<td>5</td>
</tr>
<tr>
<td>\sin, \cos, \tan, \ctg, \ctg, \sqrt ( )</td>
<td>obsługiwane tylko domyślne pierwiastki 2 stopnia</td>
<td>4</td>
</tr>
<tr>
<td>^</td>
<td>potęgowanie</td>
<td>3</td>
</tr>
<tr>
<td>/, :, \frac ( )</td>
<td>niejawnie - gdy brak operatora między zmiennymi</td>
<td>2</td>
</tr>
<tr>
<td>-, +</td>
<td>operatory binarne</td>
<td>1</td>
</tr>
</tbody>
</table>

GAWK - konwersja B\LaTeX{} → XML


```gawk
#!/usr/bin/gawk -f
# Inicjalizacja zmiennych
BEGIN{
    # Nazwa pliku wyjściowego
    plikWy="wyjscie.xml"
    # Zmienne indeksowe
    indx=0;
    indxzm=0;
}

Dalej dokonuje się wstępne przetworzenie danych. Na przykładzie ujednoczania jednego rodzaju nawiasów można zobaczyć, jak dokonuje się podmian różnych ciągów znaków.

```gawk
# Ujednolicanie nawiasów. Zamiana nawiasy dopasowujące się # do rozmiaru wyrażeń na zwykle
while(match($0,"\\\left\(")){
    gsub("\\\left\(\", "\(\", $0)
}
```gawk

Dopóki w rekordzie $0 zachodzi dopasowanie wyrażenia "\\\left\(\" podmieniane jest ono na "\". W wyrażeniach regularnych zarówno znak \, jak i \{ jest znakiem specjalnym. Anulowanie szczególnego znaczenia znaku specjalnego
7. Transformacja danych za pomocą szablonów

odbywa się przez wstawienie znaku \ przed znakiem specjalnym. Zatem, aby
anulować znaczenie nawiasu, trzeba poprzedzić go ukośnikiem, którego znacze-
nie specjalne trzeba anulować poprzedzając go ukośnikiem. Stąd 2 ukośniki. W
zwykłych ciągach znaków wystarczy pojedynczy ukośnik do anulowania znacze-
nia.

Poniżej przedstawiono pokrótce zamianę ułamków zwykłych na dzielenie
infiksowe. Idea algorytmu polega na ustaleniu pozycji odpowiednich miejsc
w ciągu wejściowym. W ogólności ułamki mają postać

\[ \frac{\text{a}}{\text{b}} \]

Ustalana jest teraz długość odpowiednich łańcuchów. Niech la będzie ilością
znaków przed a, lb oraz lc ilością znaków odpowiednio od b oraz c włącznie
do końca. Niech l będzie długością rekordu. Wyrażenie można zamienić do po-
staci infiksowej przez wybranie z początkowego rekordu pierwszych la znaków.
Za nimi wstawiane są w nawiasie L-la-6-lb-1 znaki od la+7. Wstawiany jest
operator dzielenia / i dalej w nawiasie lb-lc-2 znaki od lb+1. Następnie dopi-
sywane są znaki od lc do końca. W ten sposób otrzymuje się zamianę ułamków
zwykłych na dzielenie infiksowe.

Przygotowane dane (bez spacji, z ujednoliconym zapisem operatorów, z infik-
sowym dzieleniem w miejsce ułamków) znajdują się w zmiennej

$0$

sprawdzane jest czy na początku rekordu znajduje się pojedyncza litera mała
lub duża z opcjonalnym nawiasem klamrowym, oddzielonym od litery znakiem

```bash
# Wyszukuje symbole jednoliterowe z ewentualnymi indeksami
if(match($0, "^[a-zA-Z](_{[0123456789]+})?" , L)){
  gsub("^[a-zA-Z](_{[0123456789]+})?" , "" , $0);
  if(length(WYNIK)==0)
  WYNIK=L[0];
  else
  WYNIK=WYNIK" "$L[0];
}

$0=WYNIK;
```

Sprawdzane jest, czy na początku rekordu znajduje się pojedyncza litera mała
lub duża z opcjonalnym nawiasem klamrowym, oddzielonym od litery znakiem
Podkreślenia, w którym to nawiasie może wystąpić dowolna ilość cyfr. Jeśli wystąpi takie dopasowanie, dopasowane wyrażenie zapisane zostanie do tablicy \( L \). Następnie z rekordu usuwany jest dopasowany ciąg, a sama wartość dopasowania dołączana jest za pośrednictwem pojedynczej spacji do zawartości zmiennej \( \text{WYNIK} \). Jeśli zmieniona ta jest pusta, to dopasowanie zostanie po prostu do niej wpisane. W ten sposób przetwarzane są wszystkie obsługiwane wyrażenia. Na koniec zmieniona \( \text{WYNIK} \) przepisywana jest do rekordu \( \$0 \).

Po rozdzieleniu wszystkich symboli, wykonać można pętlę poszukującą niejawnych operatorów mnożenia, to znaczy liczb lub zmiennych nie rozdzielonych żadnym operatorem (dostatecznie sprawdzić, czy dwa kolejne pola nie są operatarami ani nawiasami). W przypadku znalezienia takich, wstawiany jest między nie znak mnożenia.

```c
# Wyszukuje niejawne mnożenie i zamienia na jawne
for(i=1;i<NF;i++){
  if(match($i, "\\\\[a-zA-Z]+\|([1234567890]+(\.[1234567890]+)?)\|([a-zA-Z])")
      &&
    match($i+1, "\\\\[a-zA-Z]+\|([1234567890]+(\.[1234567890]+)?)\|([a-zA-Z])")
  {
    $i=$i" \times ";
  }
}
```

W przedstawionym kodzie linia z warunkiem \( \text{if} \) została złamana, aby mogła zmienić się na stronie. Nie jest to jednak poprawne pod względem składni.

Skomentowania wymaga dostawianie spacji, czyli znaku separamującego do pola rekordu. Wewnątrz powyższej pętli nie powoduje to zmiany ilości pól w rekordzie. Jest tak, ponieważ, zgodnie z dokumentacją GAWK, po zapisaniu nowej wartości do pola, rekord zostanie ponownie przetworzony dopiero przy odwołaniu do \( \$0 \). Natychmiastowe przeliczenie rekordu następuje przy bezpośredniej (a nie przez pola rekordu) modyfikacji \( \$0 \).

Dalsza część kodu odpowiada za ustalanie kolejności działań. Metoda działania tej części kodu może wymagać wyjaśnienia. Całe wyrażenie jest tu zamieniane fragmentami na łańcuchy znaków reprezentujące te fragmenty. Wyrażenie jest w ten sposób „zwijane” aż do jednego łańcucha.

Zasadę działania dobrze obrazuje przykład. Rozważmy wyrażenie

\[
\sin ( 2 + x ) ^ 2 - \left( \frac{a}{2} \right) \times \pi
\]

Pierwszym etapem jest wyodrębnienie nawiasów. Ich zawartość jest zastępowana przez równoważny łańcuch znaków. Można to porównać do przepisywania termów. Zbiór równań tworzony jest w tablicy, co pozwala odtworzyć później pierwotną postać wyrażenia. Nawiasy nie będą już potrzebne, ponieważ nowa reprezentacja będzie jednoznaczna nawet bez nich. Zatem w pierwszym kroku dokonywane są dwa podstawienia
7. Transformacja danych za pomocą szablonów

NAWIAS[1] <- 2 + x
NAWIAS[2] <- a / 2

Ponadto nie zaszkodzi potraktować całego wyrażenia, jako nawiasu. Podstawić więc można


Teraz w każdym z nawiasów wyszukiwany jest operator o najwyższym priorytetcie i wraz z jego argumentami zastępowany jest odpowiednim wyrażeniem tak, jak w przypadku nawiasów. W celu ułatwienia późniejszej analizy, działania zamieniane są od razu do postaci prefiksowej. Dla dwóch pierwszych nawiasów podstawienia są jednoetapowe:

NAWIAS[1]:
    ZM[1] <- + 2 x

NAWIAS[2]:

W trzecim nawiasie dokona się kilka podstawień w kolejności od najwyższego priorytetu i od lewej strony.

\sin NAWIAS[1] ^ 2 - NAWIAS[2] * \pi
    ZM[3] <- \sin NAWIAS[1]


Tym sposobem w każdej zmiennej ZM[x] znajduje się dokładnie jedno działanie, a w każdym nawiasie NAWIAS[x] znajduje się dokładnie jedna zmieniona ZM[y] albo liczba lub stała lub zmieniona w sensie matematycznym (np. x lub \pi lub 123.456). W takiej sytuacji nie ma problemów z kolejnością działań - reprezentacja jest jednoznaczna.

Tak przygotowane dane łatwo teraz konwertować do standardu MathML. Poniższy pseudokod wyjaśnia sposób konwersji.

FOR po wszystkich ZM[i]
    $0 := ZM[i]$ // ułatwia analizę
    Zamień $1 na odpowiedni znacznik MathML;
    IF $2 jest liczbą
        $2 := "<cn>$2</cn>";

106
ELSE IF $2$ jest zmienną lub symbolem
    $2$ := "<ci>$2$</ci>";
ELSE
    $2$ := "<apply>$2</apply>";
IF istnieje $3$
    IF $3$ jest liczbą
        $3$ := "<cn>$3$</cn>";
    ELSE IF $3$ jest zmienną lub symbolem
        $3$ := "<ci>$3$</ci>";
    ELSE
        $3$ := "<apply>$3</apply>";
    ZM[i] := $1$"$2$"$3$;
ELSE
    ZM[i] := $1$"$2$";

Teraz wystarczy ponownie podstawić pod ciągi ZM[i] i NAWIAS[y] odpowiadające im wyrażenia, przez proste podmienianie wzorców. Otrzymane wyrażenia należy poprzedzić odpowiednimi nagłówkami (w szczególności dodać znaczek różniczkowania, którego argumentem jest całe wyrażenie) i zakończyć ich domknięciami i wypisać wynik do pliku.

Wyżej przedstawiony program jest skrytem, który przyjmuje jako argument nazwę pliku z danymi wejściowymi.

Chcąc przetransformować do MathML rozważane wcześniej wyrażenie \( \frac{d}{dx}(\sin(2+x))^2-(a/2)*\pi \) należy zwrócić uwagę na priorytety działań. Pominięcie nawiasów wokół funkcji sinus byłoby interpretowane przez skrypt tak, jak zapis powyższy, natomiast Czytelnicy mogliby to rozumieć jako sinus kwadratu sumy.

Niech powyższe wyrażenie zostanie zapisane do pliku we.txt w celu transformatji. Niech plik ze skryptem nazywa się run. Program wywołujemy komandą ./run we.txt. Wynik zapisany zostanie w pliku o nazwie wyjście.xml.

7.6.2. Transformacja XML do BibiX

W ramach tej części projektu powstał jedynie skrypt transformujący XML do BibiX. Jako że pliki typu MathML są zgodne z XML, ich budowa jest hierarchiczna, reprezentująca drzewo XML. Skrypt XSLT wglębia się w drzewo przekształcając je stopniowo do formatu zgodnego BibiX. Zestaw operatorów został ograniczony do minimum, są to: dodawanie, odejmowanie, mnożenie, dzielenie, potęgowanie, logarytmowanie, podstawowe funkcje trygonometryczne i pierwiastkowanie.

Idea algorytmu polega na przyporządkowaniu za pomocą polecenia match, poszczególnym poleceniom MathML, polecen systemu składni BibiX. A następnie, dzięki wykorzystaniu polecenia X-path takich jak self (oś własna elementu) i following-sibling (Oś obejmująca rodzeństwo następujące po węźle jej wystąpienia) zostaje określony zakres ich stosowania, mówiący jakie elementy obej-
7. Transformacja danych za pomocą szablonów

mowane są przez dane działanie. Wykrycie zmiennej i zastąpienie jej własnym symbolem:

```xml
<xsl:template match="m:bvar">
    <xsl:apply-templates/>
    <xsl:if test="following-sibling::m:bvar">
        ,
    </xsl:if>
</xsl:template>
```

Wykrycie ułamka wraz z jego stopniem i zamiana na odpowiednią instrukcję \LaTeX

```xml
<xsl:template match="m:apply[*[1][self::m:root]]">
    <xsl:text>\sqrt</xsl:text>
    <xsl:if test="m:degree!=2">
        [\xsl:apply-templates select="m:degree/*"]
    </xsl:if>
    {\xsl:apply-templates select="*[position()>1 and not(self::m:degree)]"}/
</xsl:template>
```

Aby całość była kompilowalnym plikiem \LaTeX, stworzony szablon XSLT dodaje preambułę dokumentu. Przykładowa sesja transformacji mogłaby wyglądać następująco:

```
- saxon -in <wejscie.xml> XMLtoLATEX.xsl <wyjście.tex>
- pdflatex wyjście.tex
```

Literatura

8.1. Modele informacyjne

Modele informacyjne są reprezentacją koncepcji, relacji, reguł i operacji w wybranym zagadnieniu. Dostarczają one podzielną, stabilną i zorganizowaną strukturę informacyjnych wymagań dla rozpatrywanej dziedziny [1].

Wyrażenie model informacyjny w ogólności odnosi się do określenia konkretnych modeli rzeczy takich jak np. urządzenia, budynki czy procesy technologiczne. Model informacyjny składa się z modelu urządzenia (procesu) oraz opisującej go dokumentacji, zawierającej właściwości, relacje oraz operacje, które mogą zostać wykonane na modelu. Składowe modelu mogą odzwierciedlać obiekty świata rzeczywistego (np. części maszyny) lub mogą być elementami abstrakcyjnymi (np. aplikacja komputerowa).

Model informacyjny dostarcza formalizmu opisującego wybrane zagadnienie, bez narzucania, jak ten opis ma dokładnie wyglądać oraz jak ma zostać zrealizowany (np. zaimplementowany). Modele informacyjne służą przedstawieniu informacji z danej dziedziny w sposób ogólny, podkreślając generalne koncepcje, które są przedstawione w ustrukturalizowany sposób.


Przykładem modelu informacyjnego może być model firmy przemysłowej. Powinien on zawierać m.in. opis struktury organizacyjnej firmy, jak i jej wewnętrznej strategii oraz podstawowe koncepcje, cele i wytyczne poszczególnych działów. Należałyby również zamieścić w nim rodzaje informacji, jakie powinny przepływać pomiędzy poszczególnymi działami. Model taki mógłby posłużyć np. do przedstawienia struktury firmy nowym pracownikom lub w celu zaprezentowania podstawowych koncepcji kontrahentom.
8. Budowa modeli informacyjnych i ich profilowanie

8.1.1. Profilowanie modeli informacyjnych

Modele informacyjne są najczęściej tworzone do konkretnych zastosowań. Koncepcja modeli informacyjnych jest dość ogólnym i szerokim pojęciem. Taka definicja z jednej strony wiąże się ze sporą elastycznością podczas korzystania z nich, jednak z drugiej trudno wybrać konkretny model do konkretnego zastosowania. Aby temu sprostać często tworzone są indywidualne profile do określonych potrzeb.

W najogólniejszym sensie profilowanie modeli informacyjnych można zdefiniować jako zawężanie lub rozszerzanie o nowe elementy określonego modelu. Przykładowo, format plików SVG (ang. Scalable Vector Graphics) można byłoby rozszerzyć o znaczniki opisujące przedefiniowane złożone obiekty, będące najczęściej wykorzystываемymi w konkretnym zagadnieniu. Sprofilowanie takie umożliwiłoby m.in. przejrzystsze wykorzystanie tego formatu, jego szybsze przesyłanie (zwłaszcza w przypadku dużej ilości danych) oraz prostsze definiowanie określonych obiektów.

8.2. Języki modelowania

8.2.1. XML

XML (ang. Extensible Markup Language) jest prostym, elastycznym i niezależnym od platformy językiem przeznaczonym do formalnego reprezentowania różnych danych w ustrukturalizowany sposób. Zastosowanie XML umożliwia łatwą wymianę dokumentów pomiędzy różnymi systemami i znacząco przychodzi się do popularności tego języka w dobie Internetu. XML jest przede wszystkim językiem, który umożliwia łatwą wymianę dokumentów pomiędzy różnymi systemami. XML jest przede wszystkim językiem, który umożliwia łatwą wymianę dokumentów pomiędzy różnymi systemami. XML jest przede wszystkim językiem, który umożliwia łatwą wymianę dokumentów pomiędzy różnymi systemami. XML jest przede wszystkim językiem, który umożliwia łatwą wymianę dokumentów pomiędzy różnymi systemami. XML jest przede wszystkim językiem, który umożliwia łatwą wymianę dokumentów pomiędzy różnymi systemami. XML jest przede wszystkim językiem, który umożliwia łatwą wymianę dokumentów pomiędzy różnymi systemami. XML jest przede wszystkim językiem, który umożliwia łatwą wymianę dokumentów pomiędzy różnymi systemami. XML jest przede wszystkim językiem, który umożliwia łatwą wymianę dokumentów pomiędzy różnymi systemami. XML jest przede wszystkim językiem, który umożliwia łatwą wymianę dokumentów pomiędzy różnymi systemami. XML jest przede wszystkim językiem, który umożliwia łatwą wymianę dokumentów pomiędzy różnymi systemami. XML jest przede wszystkim językiem, który umożliwia łatwą wymianę dokumentów pomiędzy różnymi systemami. XML jest przede wszystkim językiem, który umożliwia łatwą wymianę dokumentów pomiędzy różnymi systemami. XML jest przede wszystkim językiem, który umożliwia łatwą wymianę dokumentów pomiędzy różnymi systemami. XML jest przede wszystkim językiem, który umożliwia łatwą wymianę dokumentów pomiędzy różnymi systemami. XML jest przede wszystkim językiem, który umożliwia łatwą wymianę dokumentów pomiędzy różnymi systemami. XML jest przede wszystkim językiem, który umożliwia łatwą wymianę dokumentów pomiędzy różnymi systemami. XML jest przede wszystkim językiem, który umożliwia łatwą wymianę dokumentów pomiędzy różnymi systemami. XML jest przede wszystkim językiem, który umożliwia łatwą wymianę dokumentów pomiędzy różnymi systemami. XML jest przede wszystkim językiem, który umożliwia łatwą wymianę dokumentów pomiędzy różnymi systemami. XML jest przede wszystkim językiem, który umożliwia łatwą wymianę dokumentów pomiędzy różnymi systemami. XML jest przede wszystkim językiem, który umożliwia łatwą wymianę dokumentów pomiędzy różnymi systemami. XML jest przede wszystkim językiem, który umożliwia łatwą wymianę dokumentów pomiędzy różnymi systemami. XML jest przede wszystkim językiem, który umożliwia łatwą wymianę dokumentów pomiędzy różnymi systemami. XML jest przede wszystkim językiem, który umożliwia łatwą wymianę dokumentów pomiędzy różnymi systemami. XML jest przede wszystkim językiem, który umożliwia łatwą wymianę dokumentów pomiędzy różnymi systemami. XML jest przede wszystkim językiem, który umożliwia łatwą wymianę dokumentów pomiędzy różnymi systemami. XML jest przede wszystkim językiem, który umożliwia łatwą wymianę dokumentów pomiędzy różnymi systemami. XML jest przede wszystkim językiem, który umożliwia łatwą wymianę dokumentów pomiędzy różnymi systemami. XML jest przede wszystkim językiem, który umożliwia łatwą wymianę dokumentów pomiędzy różnymi systemami. XML jest przede wszystkim językiem, który umożliwia łatwą wymianę dokumentów pomiędzy różnymi systemami. XML jest przede wszystkim językiem, który umożliwia łatwą wymianę dokumentów pomiędzy różnymi systemami. XML jest przede wszystkim językiem, który umożliwia łatwą wymianę dokumentów pomiędzy różnymi systemami. XML jest przede wszystkim językiem, który umożliwia łatwą wymianę dokumentów pomiędzy różnymi systemami. XML jest przede wszystkim językiem, który umożliwia łatwą wymianę dokumentów pomiędzy różnymi systemami. XML jest przede wszystkim językiem, który umożliwia łatwą wymianę dokumentów pomiędzy różnymi systemami. XML jest przede wszystkim językiem, który umożliwia łatwą wymianę dokumentów pomiędzy różnymi systemami. XML jest przede wszystkim językiem, który umożliwia łatwą wymianę dokumentów pomiędzy różnymi systemami. XML jest przede wszystkim językiem, który umożliwia łatwą wymianę dokumentów pomiędzy różnymi systemami. XML jest przede wszystkim językiem, który umożliwia łatwą wymianę dokumentów pomiędzy różnymi systemami. XML jest przede wszystkim językiem, który umożliwia łatwą wymianę dokumentów pomiędzy różnymi systemami. XML jest przede wszystkim językiem, który umożliwia łatwą wymianę dokumentów pomiędzy różnymi systemami. XML jest przede wszystkim językiem, który umożliwia łatwą wymianę dokumentów pomiędzy różnymi systemami. XML jest przede wszystkim językiem, który umożliwia łatwą wymianę dokumentów pomiędzy różnymi systemami. XML jest przede wszystkim językiem, który umożliwia łatwą wymianę dokumentów pomiędzy różnymi systemami. XML jest przede wszystkim językiem, który umożliwia łatwą wymianę dokumentów pomiędzy różnymi systemami. XML jest przede wszystkim językiem, który umożliwia łatwą wymianę dokumentów pomiędzy różnymi systemami. XML jest przede wszystkim językiem, który umożliwia łatwą wymianę dokumentów pomiędzy różnymi systemami. XML jest przede wszystkim językiem, który umożliwia łatwą wymianę dokumentów pomiędzy różnymi systemami. XML jest przede wszystkim językiem, który umożliwia łatwą wymianę dokumentów pomiędzy różnymi systemami. XML jest przede wszystkim językiem, który umożliwia łatwą wymianę dokumentów pomiędzy różnymi systemami. XML jest przede wszystkim językiem, który umożliwia łatwą wymianę dokumentów pomiędzy różnymi systemami. XML jest przede wszystkim językiem, który umożliwia łatwą wymianę dokumentów pomiędzy różnymi systemami. XML jest przede wszystkim językiem, który umożliwia łatwą wymianę dokumentów pomiędzy różnymi systemami. XML jest przede wszystkim językiem, który umożliwia łatwą wymianę dokumentów pomiędzy różnymi systemami. XML jest przede wszystkim językiem, który umożliwia łatwą wymianę dokumentów pomiędzy różnymi systemami. XML jest przede wszystkim językiem, który umożliwia łatwą wymianę dokumentów pomiędzy różnymi systemami. XML jest przede wszystkim językiem, który umożliwia łatwą wymianę dokumentów pomiędzy różnymi systemami. XML jest przede wszystkim językiem, który umożliwia łatwą wymianę dokumentów pomiędzy różnymi systemami. XML jest przede wszystkim językiem, który umożliwia łatwą wymianę dokumentów pomiędzy różnymi systemami. XML jest przede wszystkim językiem, który umożliwia łatwą wymianę dokumentów pomiędzy różnymi systemami. XML jest przede wszystkim językiem, który umożliwia łatwą wymianę dokumentów pomiędzy różnymi systemami. XML jest przede wszystkim językiem, który umożliwia łatwą wymianę dokumentów pomiędzy różnymi systemami. XML jest przede wszystkim językiem, który umożliwia łatwą wymianę dokumentów pomiędzy różnymi systemami. XML jest przede wszystkim językiem, który umożliwia łatwą wymianę dokumentów pomiędzy różnymi systemami. XML jest przede wszystkim językiem, który umożliwia łatwą wymianę dokumentów pomiędzy różnymi systemami. XML jest przede wszystkim językiem, który umożliwia łatwą wymianę dokumentów pomiędzy różnymi systemami. XML jest przede wszystkim językiem, który umożliwia łatwą wymianę dokumentów pomiędzy różnymi systemami. XML jest przede wszystkim językiem, który umożliwia łatwą wymianę dokumentów pomiędzy różnymi systemami. XML jest przede wszystkim językiem, który umożliwia łatwą wymianę dokumentów pomiędzy różnymi systemami. XML jest przede wszystkim językiem, który umożliwia łatwą wymianę dokumentów pomiędzy różnymi systemami. XML jest przede wszystkim językiem, który umożliwia łatwą wymianę dokumentów pomiędzy różnymi systemami. XML jest przede wszystkim językiem, który umożliwia łatwą wymianę dokumentów pomiędzy różnymi systemami. XML jest przede wszystkim językiem, który umożliwia łatwą wymianę dokumentów pomiędzy różnymi systemami. XML jest przede wszystkim językiem, który umożliwia łatwą wymianę dokumentów pomiędzy różnymi systemami. XML jest przede wszystkim językiem, który umożliwia łatwą wymianę dokumentów pomiędzy różnymi systemami. XML jest przede wszystkim językiem, który umożliwia łatwą wymianę dokumentów pomiędzy różnymi systemami. XML jest przede wszystkim językiem, który umożliwia łatwą wymianę dokumentów pomiędzy różnymi systemami. XML jest przede wszystkim językiem, który umożliwia łatwą wymianę dokumentów pomiędzy różnymi systemami. XML jest przede wszystkim językiem, który umożliwia łatwą wymianę dokumentów pomiędzy różnymi systemami. XML jest przede wszystkim językiem, który umożliwia łatwą wymianę dokumentów pomiędzy różnymi systemami. XML jest przede wszystkim językiem, który umożliwia łatwą wymianę dokumentów pomiędzy różnymi systemami. XML jest przede wszystkim językiem, który umożliwia łatwą wymianę dokumentów pomiędzy różnymi systemami. XML jest przede wszystkim językiem, który umożliwia łatwą wymianę dokumentów pomiędzy różnymi systemami. XML jest przede wszystkim językiem, który umożliwia łatwą wymianę dokumentów pomiędzy różnymi systemami. XML jest przede wszystkim językiem, który umożliwia łatwą wymianę dokumentów pomiędzy różnymi systemami. XML jest przede wszystkim językiem, który umożliwia łatwą wymianę dokumentów pomiędzy różnymi systemami. XML jest przede wszystkim językiem, który umożliwia łatwą wymianę dokumentów pomiędzy różnymi systemami. XML jest przede wszystkim językiem, który umożliwia łatwą wymianę dokumentów pomiędzy różnymi systemami. XML jest przede wszystkim język...
zasoby w tych strukturach wyszukujemy za pomocą słów kluczowych skojarzonych z zagadnieniem. Niestety większość z otrzymanych informacji jest niewłaściwa z punktu widzenia naszych potrzeb. W związku z tym przydatny okazałby się jednolity model opisu tych zasobów.


W modelu RDF dany zasób może posiadać tylko jeden opis, który z kolei może składać się z wielu właściwości. Każda właściwość ma jedną wartość (rys. 8.1). Opis w postaci trójki zasób, własność, wartość jest cechą charakterystyczną modelu RDF. Poniżej przedstawiono dwa przykłady zapisu tych samych trójek, pierwszy w języku RDF/XML, a drugi w Notation 3 ([http://en.wikipedia.org/wiki/Notation_3](http://en.wikipedia.org/wiki/Notation_3)).

Listing 8.1: Zapis trójek w formacie RDF/XML

```xml
<rdf:RDF
   xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
   xmlns:dc="http://purl.org/dc/elements/1.1/"/>
  <rdf:Description
      rdf:about="http://en.wikipedia.org/wiki/Tony_Benn">
    <dc:title>Tony Benn</dc:title>
    <dc:publisher>Wikipedia</dc:publisher>
  </rdf:Description>
</rdf:RDF>
```

Listing 8.2: Zapis trójek w formacie Notation 3

```not
@prefix dc: <http://purl.org/dc/elements/1.1/>. 

<http://en.wikipedia.org/wiki/Tony_Benn>
  dc:title "Tony Benn";
  dc:publisher "Wikipedia".
```

Opisanie zasobów w oparciu o przytoczony uniwersalny model pozwala ujednolicić treść względem określonego szablonu. Podejście takie umożliwia programom komputerowym trafniej interpretować zawartość zasobów, a w następstwie indeksować, katalogować oraz łączyć je w dziedzinie z innymi zasobami o podobnej tematyce. Taki sposób opisu zasobów zapewnia użytkownikowi wydajniejszy i pewniejszy dostęp do informacji.
8. Budowa modeli informacyjnych i ich profilowanie

Rys. 8.1: Model opisu RDF.

8.2.3. OWL

Język OWL (ang. *Web Ontology Language*) został stworzony by usprawnić komunikację pomiędzy komputerami podłączonymi do Internetu ([http://www.w3.org/TR/owl-features/](http://www.w3.org/TR/owl-features/)). Nadaje on formalne znaczenie informacjom zawartym w sieci, dzięki czemu łatwiejsze staje się ich przetwarzanie. OWL można przedstawić w strukturze czterowarstwowej jako produkt zbudowany na fundamentach innych języków. Poszczególne warstwy tej struktury to:

- warstwa ontologii (OWL),
- warstwa schema (RDF Schema),
- warstwa metadanych (RDF),

OWL występuje w trzech odmianach o rosnącym poziomie złożoności:

- **OWL Lite** – jest przeznaczony dla użytkowników, którzy potrzebują jedynie hierarchii klasyfikacji oraz podstawowych ograniczeń. Definiuje powiązania z RDF Schema, metody sprawdzania równości i nierówności klas lub właściwości, metody operowania na właściwościach oraz wybór określonych wartości z klas.
- **OWL DL** – jest przeznaczony dla użytkowników wymagających pełnego języka OWL przy jednoczesnym zapewnieniu obliczalności wyrażeń (wszelkie konkluzje są obliczalne, a wszelkie obliczenia osiąganą wynik w skończonym czasie). Definiuje taki sam zestaw możliwości jak OWL Lite, ale ponadto posiada dodatkowy zestaw możliwych operacji na klasach i właściwościach. Wymaga też separacji typów (klasa nie może być jednocześnie indywidual lub właściwością, a właściwość nie może być jednocześnie indywidual lub klasą). Ponadto właściwości w OWL DL muszą być relacjami pomiędzy dwoma klasami albo relacjami pomiędzy instancjami klas i typami danych RDF lub XML Schema.
- **OWL Full** – podobnie jak OWL DL zapewnia on dostęp do pełnego języka OWL, ale nie wprowadza tak restrykcyjnych ograniczeń, przez co nie może zapewnić pełnej obliczalności wyrażeń. W szczególności oznacza to, że nie wymaga separacji typów oraz nie ogranicza właściwości. W przeciwieństwie do dwóch poprzednich, pozwala na użycie klas jako instancji.

Ponieważ każda z tych odmian jest rozwinięciem poprzedniej, prawdziwe są następujące stwierdzenia:
8.2. Języki modelowania

- Każda poprawna ontologia OWL Lite jest również poprawną ontologią OWL DL
- Każda poprawna ontologia OWL DL jest również poprawną ontologią OWL Full
- Każda poprawna konkluzja OWL Lite jest również poprawną konkluzją OWL DL
- Każda poprawna konkluzja OWL DL jest również poprawną konkluzją OWL Full

8.2.4. UML


UML można opisać jako czterowarstwowy model, gdzie kolejne warstwy to:
- metamodel MOF (Meta Object Facility),
- metamodel UML,
- modele UML,
- dane użytkownika.

8.2.5. Różnice pomiędzy językami UML i OWL

Porównanie UML oraz OWL może odbywać się ze względu na różne kryteria:
- Składnia języka – język UML posiada reprezentację diagramową oraz reprezentację w postaci XMLa nazwaną XML Metadata Interchange (XMI), jednakże ta druga jest wtórna i tworzona na podstawie diagramów. W tym kontekście jest jedynie pewnego rodzaju konwersją podstawowej reprezentacji UMLa. OWL natomiast jest językiem bezpośrednio bazowanym na XMLu.
- Semantykę – oba języki służą innym celom. Podczas gdy OWL pozwala reprezentować wiedzę o systemie, UML został stworzony głównie z myślą o tworzeniu systemu (oprogramowania). Mimo to celem obu jest reprezentacja systemu jako całości. Oba języki są ukierunkowane obiektowo, co oznacza, że głównym komponentem reprezentacji wiedzy jest obiekt oraz jego relacje z innymi obiektami. Oba języki mają też dwie podstawowe warstwy reprezentacji wiedzy: wiedzę konkretną (udowodnioną) oraz abstrakcyjną (terminologiczną).
- Założenia dotyczące otwartego i zamkniętego świata – wyróżnić można dwa przeciwwstawne założenia dotyczące świata: świat otwarty lub zamknięty. W modelu zamkniętym nie ma potrzeby zamykania definicji klasyfikatora lub obiektu, gdyż zamknięcie jest domyślnie przyjmowane. W modelu otwartym natomiast wyraźne aksjomaty zamykające są niezbędne, by określić co nie może dotyczyć danej koncepcji. UML jest ukierunkowany na modelowanie danych oraz tworzenie systemów, a więc nawet przy tworzeniu modelu konceptualnego, reprezentowana wiedza jest traktowana jako kompletna, a w przypadku OWL-a model może potencjalnie reprezentować wiedzę częściową.
8. Budowa modeli informacyjnych i ich profilowanie

Tab. 8.1: Porównanie UMLa z OWLem

<table>
<thead>
<tr>
<th></th>
<th>UML Lite</th>
<th>UML Full</th>
<th>OWL</th>
</tr>
</thead>
<tbody>
<tr>
<td>Zorientowany obiektowo</td>
<td>tak</td>
<td>tak</td>
<td>tak</td>
</tr>
<tr>
<td>Założenie otwartego świata</td>
<td>nie</td>
<td>nie</td>
<td>tak</td>
</tr>
<tr>
<td>Właściwości globalne</td>
<td>nie</td>
<td>nie</td>
<td>tak</td>
</tr>
<tr>
<td>Synonimy</td>
<td>nie</td>
<td>nie</td>
<td>tak</td>
</tr>
<tr>
<td>Metaklasy</td>
<td>tak*</td>
<td>tak*</td>
<td>tak</td>
</tr>
<tr>
<td>Universalna koncepcja</td>
<td>nie</td>
<td>tak*</td>
<td>tak</td>
</tr>
</tbody>
</table>

* - oznacza występowanie założenia tylko w ograniczonym zakresie

• Założenia i synonimy dotyczące unikalnych nazw – ponieważ OWL ma na celu zapewnienie wsparcia dla sieci semantycznej, narzuca to pewne ograniczenia na język, takie jak konieczność wspierania rozproszonych ontologii. Aby to osiągnąć, OWL umożliwia tworzenie definicji synonimów dla klas, właściwości oraz indywidualnych opisów. Natomiast UML wymaga, aby definicje klas oraz właściwości były unikalne.
• Zasięg atrybutów i właściwości – UML nie ogranicza również globalnego zakresu atrybutów i asocjacji z powodu założenia zamkniętego świata. Także w OWLu właściwości mają globalny zasięg.

Skrócone porównanie podstawowych cech UMLa z OWLem przedstawiono w tab. 8.1.

8.3. Wybrane języki opisu geometrii

8.3.1. GML


Dokumenty GML muszą być zgodne ze schematami określającymi strukturę ich zawartości. Wszystkie dostępne schematy GML w wersji 3.1.1 znajdują się pod adresem [http://schemas.opengis.net/gml/3.1.1/base/](http://schemas.opengis.net/gml/3.1.1/base/). Oto niektórych z nich:

- Podstawowe schematy języka GML – gmlBase.xsd,
- Podstawowe (Proste) komponenty – basicTypes.xsd,
- Obiekty złożone – geometryBasic.xsd, geometryPrimitives.xsd, geometryAggregates.xsd, geometryComplexes.xsd,
- Obiekty geograficzne – feature.xsd, dynamicFeature.xsd, coverage.xsd, observation.xsd, grids.xsd,
- Zapis topologii – topology.xsd,
- Jednostki miar – units.xsd, measures.xsd,
8.3. Wybrane języki opisu geometrii


8.3.2. X3D

X3D jest XMLowym standardem ISO (ISO/IEC 19775/19776/19777) pozwalającym przechowywać różnorodne informacje graficzne i geometryczne [http://www.web3d.org/x3d/]. Oznacza to możliwość opisu złożonych obiektów geometrycznych oraz ich atrybutów fizycznych (masa, tarcie) i graficznych (tekstura), jak również informacji o środowisku (oświetlenie, mgła, grawitacja itp.). Od wersji 3.2 format ten pozwala również definiować przeguby łączące poszczególne obiekty, a tym samym tworzyć łańcuchy kinematyczne. Istnieje szereg edytorów przeznaczonych bezpośrednio do edycji plików X3D (np. X3D-Edit, SwirlX3D, Flux Studio); również niektóre edytorzy graficzne umożliwiają eksport do formatu X3D (np. Blender). Jednakże w przypadku Blendeda pamiętać należy, iż nie zapisuje on ogólnych danych o geometrii obiektów (np. promieni kuli lub wysokości/szerokości sześcianu), ale jedynie współrzędne wszystkich wierzchołków danej bryły.

Format ten definiuje 6 profili różniących się dostępnymi znacznikami oraz ich atrybutami. Przedstawione zostały one w tab. 8.2 wraz z krótką charakterystyką (każdy z opisów jest prawdziwy dla kolejnych profili w tabeli).

Zakres funkcjonalności X3D można modyfikować nie tylko poprzez wybór jednego z powyższych profili. Oprócz nich użytkownik ma do dyspozycji różnorodne komponenty grupujące zestawy węzłów pod względem funkcjonalności, a poszczególne komponenty zwykle mogą mieć określony poziom, definiujący zakres dostępności węzłów z danego komponentu. Deklaracja używanego profilu odbywa się poprzez dopisanie odpowiedniego atrybutu w głównym węźle X3D, np.:

```xml
<X3D profile="Immersive" version="3.2">
```

Natomiast używane komponenty deklaruje się w nagłówku, np.:
8. Budowa modeli informacyjnych i ich profilowanie

Tab. 8.2: Charakterystyka profili formatu X3D

<table>
<thead>
<tr>
<th>Profil</th>
<th>Opis</th>
</tr>
</thead>
<tbody>
<tr>
<td>Core</td>
<td>Definiuje jedynie podstawowy zestaw znaczników służących do opisu brył geometrycznych</td>
</tr>
<tr>
<td>Interchange</td>
<td>Zawiera znaczniki wspomagające animację sceny</td>
</tr>
<tr>
<td>Interactive</td>
<td>Zawiera znaczniki opisujące interakcję użytkownika ze sceną</td>
</tr>
<tr>
<td>MPEG-4 interactive</td>
<td>Wprowadza wsparcie dla standardu MPEG-4</td>
</tr>
<tr>
<td>Immersive</td>
<td>Wprowadza pełen zestaw znaczników kontroli nawigacji i sensorów</td>
</tr>
<tr>
<td>Full</td>
<td>Rozszerza zakres możliwości wielu znaczników oraz wprowadza kilka nowych (np. zapewniające powiązania z CADem)</td>
</tr>
</tbody>
</table>

Każdy plik X3D musi określać, z którego ze standardowych profili korzysta. Ponadto może mieć określone wykorzystywane komponenty. Oprócz nich użytkownik może definiować własne rozszerzenia funkcjonalności przy pomocy prototypów. W tym celu stosuje się znaczniki: `<ProtoDeclare>` do rozpoczęcia definiowania nowego węzła, `<ProtoInterface>` do określenia atrybutów węzła (przy pomocy `<field>`) oraz `<ProtoBody>` opisujący wygląd węzła.

8.4. Realizacja profilu X3D


Aby to osiągnąć konieczne było głównie zawężenie możliwości X3D, aby opis modelu był równoważny z opisem wymaganym przez silnik fizyczny ODE (co również umożliwi łatwą konwersję z ODE do VTK). W tym celu plik oficjalnego schematu X3D w wersji 3.2 został poddany edycji i usunięto z niego wszelkie niepotrzebne (a jednocześnie opcjonalne) elementy i atrybuty. Utworzony profil zawiera wszystkie konieczne węzły X3D, a z węzłów opcjonalnych pozostawia jedynie następujące:

- Podstawowe bryły geometryczne

116
8.5. Realizacja parsera X3D

Utworzony profil X3D pozwolił na niemalże bezpośrednią konwersję modelu danych X3D na model wymagany przez ODE. Ogólne założenie dotyczące obiektów jest podobne w obu formatach - definiowane są geometrie (Shape w X3D, Geom w ODE) oraz bryły sztywne (RigidBody w X3D, Body w ODE). Bryła sztywna jest rozumiana jako obiekt fizyczny (czyli posiadający właściwości fizyczne jak masa lub chropowatość powierzchni), którego kształt określa jedna lub więcej geometrii. Dodatkowo bryła sztywna może być połączona z innymi bryłami sztywnymi przy pomocy przegubów (Joint).

W przypadku konwersji z jednego modelu na drugi trzeba jednak mieć na uwadze kilka drobnych różnic. Przykładowo rotacja brył w X3D jest określana na podstawie 4 parametrów - trzy z nich definiują wektor wokół którego następuje obrót, a czwarty określa kąt obrotu. W ODE rotacja jest reprezentowana przez macierz rotacji, którą można wygenerować automatycznie na podstawie kątów Eulera, jednakże kąty te są liczone w odwrotnym kierunku niż w X3D, więc końcowa jest zmiana znaku podanego kąta. Inną różnicą jest opis kształtu cylindra, co sugerują już nazwy atrybutów. W X3D cylinder posiada atrybut height, czyli wysokość, natomiast w ODE analogicznym parametrem jest length, czyli długość. Można z tego wyciągnąć wniosek (potwierdzony w praktyce), że w X3D cylinder jest domyślnie umieszczony „w pionie” (np. kolumna), natomiast w ODE cylinder jest domyślnie umieszczony „w poziomie” (np. koło). W tab. 8.3 przedstawiono analogiczne elementy obu modeli informacyjnych.

Utworzony parser przetwarza pliki X3D zgodnie z określonym profilem, a następnie wizualizuje model opisany w pliku (patrz rys. 8.3). Do skompilowania kodu źródłowego parsera konieczne jest Qt w wersji 4.5.1 lub nowszej oraz
Tab. 8.3: Zestawienie użytych w projekcie elementów

<table>
<thead>
<tr>
<th>Opis</th>
<th>X3D</th>
<th>typ</th>
<th>funkcja</th>
</tr>
</thead>
<tbody>
<tr>
<td>Bryła sztywna</td>
<td>&lt;RigidBody&gt;</td>
<td>dBodyID</td>
<td>dCreateBody()</td>
</tr>
<tr>
<td>Geometria prostopadłościanu</td>
<td>&lt;Box&gt;</td>
<td>dGeomID</td>
<td>dCreateBox()</td>
</tr>
<tr>
<td>Geometria kuli</td>
<td>&lt;Sphere&gt;</td>
<td>dGeomID</td>
<td>dCreateSphere()</td>
</tr>
<tr>
<td>Geometria cylindra</td>
<td>&lt;Cylinder&gt;</td>
<td>dGeomID</td>
<td>dCreateCylinder()</td>
</tr>
<tr>
<td>Przegub obrotowy</td>
<td>&lt;SingleAxisHingeJoint&gt;</td>
<td>dJointID</td>
<td>dJointCreateHinge()</td>
</tr>
<tr>
<td>Przegub krzyżakowy</td>
<td>&lt;DoubleAxisHingeJoint&gt;</td>
<td>dJointID</td>
<td>dJointCreateHinge2()</td>
</tr>
<tr>
<td>Przegub translacyjny</td>
<td>&lt;SliderJoint&gt;</td>
<td>dJointID</td>
<td>dJointCreateSlider()</td>
</tr>
<tr>
<td>Przegub kulisty</td>
<td>&lt;BallJoint&gt;</td>
<td>dJointID</td>
<td>dJointCreateBall()</td>
</tr>
<tr>
<td>Przegub uniwersalny</td>
<td>&lt;UniversalJoint&gt;</td>
<td>dJointID</td>
<td>dJointCreateUniversal()</td>
</tr>
<tr>
<td>Przegub obrotowy z silnikiem</td>
<td>&lt;MotorJoint&gt;</td>
<td>dJointID</td>
<td>dJointCreateAMotor()</td>
</tr>
</tbody>
</table>

VTK w wersji 5.0 lub nowszej. Ponadto zalecane jest skorzystanie z wygenerowanego pliku Makefile do kompilacji (na Linuxie), lub alternatywnie wygenerowanie tego pliku przy pomocy Qt Creatora na podstawie pliku projektu (x3d-parser.pro). Razem z programem dostarczony został przykładowy model robota Pioneer (pioneer.x3d w podkatalogu models).

Poniżej zamieszczono fragment kodu X3D opisującego tylną oś i kółko robota (patrz rys. 8.3).

Listing 8.3: Fragment kodu X3D opisującego tylną oś i kółko robota

```xml
<RigidBody DEF="bar_body" position="0 -1.075 -1.5">
  <CollidableShape>
    <Shape DEF="bar1">
      <Cylinder height="0.5" radius="0.07"/>
    </Shape>
  </CollidableShape>
</RigidBody>
```

118
8.5. Realizacja parsera X3D

Rys. 8.3: Przykładowy model robota zwizualizowany przez parser.

```xml
</Shape>
</CollidableShape>
<CollidableShape rotation="1 0 0 1.570796" translation="0 -0.18 -0.2">
  <Shape DEF="bar2">
    <Cylinder height="0.4" radius="0.07"/>
  </Shape>
</CollidableShape>
</RigidBody>

<RigidBody DEF="back_wheel_body" orientation="0 0 1 1.570796" position="0 -1.23 -1.85">
  <CollidableShape>
    <Shape DEF="back_wheel">
      <Cylinder height="0.1" radius="0.3"/>
    </Shape>
  </CollidableShape>
</RigidBody>

<SingleAxisHingeJoint anchorPoint="0 -1.23 -1.85" axis="1 0 0">
  <RigidBody USE="bar_body"/>
</SingleAxisHingeJoint>
```
8. Budowa modeli informacyjnych i ich profilowanie

8.5.1. Opis programu


8.5.2. Wnioski

Sprofilowanie modelu informacyjnego X3D pozwoliło na znacznie uproszenie parsera, gdyż musi on jedynie przetwarzać elementy i atrybuty zawarte w danym profilu. Ponadto zredukowany został również rozmiar pliku xsd opisującego utworzony profil - oryginalny Schema X3D w wersji 3.2 ma rozmiar ponad 400kB, natomiast Schema profilu ma rozmiar jedynie około 60kB. Dzięki temu programy sprawdzające poprawność modelu na podstawie pliku Schema będą w stanie wykonać taką weryfikację znacznie szybciej. Również z punktu widzenia użytkownika korzystanie z takiego profilu jest wygodniejsze; nie musi on poznać złożonych struktur pełnego X3D (z których większość nie miałaby zastosowania w opisie modelu robota), a jedynie niewielki zestaw elementów, które w prosty i jednoznaczny sposób pozwalają opisać geometrię robota.

Literatura

9.1. Wprowadzenie

W dzisiejszych czasach coraz powszechniejsze stają się urządzania poruszające się samodzielnie, orientujące się w przestrzeni i podejmujące decyzje na podstawie sygnałów z posiadanym sensorów. Dane pozyskiwane w ten sposób mogą być jednak niedokładne lub błędne. Stąd też poprawne działanie tych urządzeń mocno zależy od szybkości zastosowanej metody wnioskowania, nawet w przypadku posiadania nieprecyzyjnych danych, a co za tym idzie, niepełnej informacji.


Dość popularnym sposobem przedstawiania wiedzy jest piramida wiedzy (rys. 9.1). Podstawę piramidy tworzą dane, które po odpowiednim przetworzeniu przekształcają się w informacje. Po kolejnym przetworzeniu i umieszczaniu informacji w odpowiednim kontekście powstaje wiedza. Przetworzenie i zrozumienie wiedzy prowadzi do mądrości. Określenie mądrości odnosi się jedynie do ludzi (nie będzie ono tu dalej rozwijane).
9. Reprezentacja informacji niepewnej i niepełnej

Kolejnym dość często używanym pojęciem jest „informacja”. Termin ten pochodzi od łacińskich określeń: wyobrażenie, wyjaśnienie, zawiadomienie. Pojęcie to występuje w teorii informacji i jest definiowane jako miara:

\[ I = \log\left(\frac{1}{p}\right) = -\log(p) \]

gdzie \( p \) jest prawdopodobieństwem otrzymania przez odbiorcę określonej wiadomości spośród skończonego zbioru wiadomości, które może emitować źródło. Ilość informacji zawarta w przekazanej wiadomości jest zatem tym większa, im prawdopodobieństwo otrzymania tej wiadomości jest mniejsze. Jednostkami ilości informacji są: szanon (ang. shannon), bit (jednostka ilości miejsca zajętego przez informację), nat (jednostka ilości informacji mierzonej przez logarytm naturalny ilości możliwości) oraz ban lub hartlej (ang. hartley, jednostka ilości informacji mierzona ilością cyfr dziesiętnych potrzebnych do jej zapisania) (1 nat = \( \log_2(e) \) bitów, 1 bit = ln(2) natów, 1 ban = ln(10) natów = \( \log_2(10) \) bitów).

Informacje można podzielić na: niedokładne, niepewne, niepełne. W niniejszej pracy przedstawione i omówione zostaną informacje niepewne i niepełne. Informacja niepewna jest to informacja, w której wystąpienie danego zjawiska jest określone z pewną dokładnością. Informacja niepełna jest to informacja, w której znana jest tylko część informacji oraz możliwa jest zmiana wnioskowania po dodaniu kolejnej, nowej informacji.

9.2. Reprezentacja wiedzy

Wiedzę, jak już wspomniano, można podzielić na dwie podstawowe kategorie: wiedzę niepewną i wiedzę niepełną. Poniżej zostaną omówione metody reprezentacji wiedzy obu kategorii (założono, że czytelnik zna podstawy logiki klasycznej).

9.2.1. Reprezentacja wiedzy niepewnej

Niepewność odzwierciedla poziom niezgodności informacji z rzeczywistością. Pojawia się, gdy informacja jest nieprzecyzyjna, gdy granice zbiorów przekazywanych wartości są niejednoznaczne. Niepewność informacji jest wyrażana
9.2. Reprezentacja wiedzy


**Metody probabilistyczne**

Metody probabilistyczne jeszcze niedawno były jedynymi metodami rozwiązywania problemów informacji niepewnej i pomimo wprowadzenia nowych metod dalej są najczęściej stosowane. Pojęciem pierwotnym w teorii prawdopodobieństwa jest pojęcie przestrzeni zdarzeń elementarnych $\Theta$ (zbioru niepodzielnych i rozłącznych wyników obserwacji). Punktem wyjściowym dla różnych probabilistycznych metod reprezentacji informacji niepewnej jest twierdzenie Bayesa. Wykorzystuje się w nich prawdopodobieństwo opisane zależnością:

$$P(A|B) = \frac{P(B|A)P(A)}{P(B)}$$ (9.1)

Oznacza ono prawdopodobieństwo wystąpienia zdarzenia $A$ pod warunkiem wystąpienia warunku $B$. Jest to reguła: Jeśli $B$ to $A$ (A może zostać uznanie jako prawdziwe wtedy, kiedy B jest uznanie za prawdziwe). Wyjaśnia to poniższy przykład.

U pewnego pacjenta przeprowadzono test na obecność wirusa SARS (ang. Severe Acute Respiratory Syndrome), czyli zespół ostrej ciężkiej niewydolności oddechowej, i wypadł on pozytywnie. Co w takiej sytuacji ma zrobić lekarz, hospitalizować pacjenta i rozpocząć leczenie? Czy jest to niekonieczne? Należy pamiętać, że przeprowadzony test nigdy nie jest całkowicie niezawodny. Jeśli jest dobry, to zapewnia bardzo wysoką skuteczność otrzymania poprawnego wyniku w przypadku obecności wirusa. Inną, bardzo ważną cechą dobrego testu jest wysokie prawdopodobieństwo otrzymania wyniku negatywnego w przypadku nieobecności wirusa. Powyższe stwierdzenia można zapisać w postaci:

- $P(T^\oplus|SARS)$ - prawdopodobieństwo otrzymania wyniku pozytywnego w przypadku obecności wirusa
- $P(T^\ominus|SARS)$ - prawdopodobieństwo otrzymania wyniku negatywnego w przypadku nieobecności wirusa.

Wymienione powyżej informacje nie są przydatne dla lekarza, którego interesuje wartość $P(SARS|T^\oplus)$ albo $P(\neg SARS|T^\ominus)$, czyli prawdopodobieństwo, że dany pacjent ma SARS. Niech, dla przykładu,

$$P(SARS) = 0.0001$$  
$$P(T^\oplus|SARS) = 0.95$$  
$$P(T^\ominus|SARS) = 0.90$$

W celu obliczenia $P(SARS|T^\oplus)$ wykorzystywane jest zależność:

$$P(SARS|T^\oplus) = \frac{P(T^\oplus|SARS)P(SARS)}{P(T^\oplus)}$$
9. Reprezentacja informacji niepewnej i niepełnej

Wartość $P(T^\oplus)$ wyliczana jest z zależności:

$$P(T^\oplus) = P(T^\oplus|SARS)P(SARS) + P(T^\oplus|\neg SARS)P(\neg SARS)$$

$$P(T^\oplus) = 0.95 \cdot 0.0001 + 0.01 \cdot 0.9999 = 0.000095 + 0.09999 = 0.100085$$

Po podstawieniu wszystkich wartości do wzorów:

$$P(SARS|T^\oplus) = \frac{0.95 \cdot 0.0001}{0.100085} = 0.00094919$$

Otrzymany wynik jest prawie o rząd większy od wartości prawdopodobieństwa wystąpienia wirusa. Nasuwa się w takim razie kolejne pytanie, czy należy zacząć kosztowne i bardzo wyniszczające organizm leczenie pacjenta? W celu dokładniejszej diagnozy stosuje się rozbudowaną wersję reguł Bayesa - trwałej niezależności warunkowej. W celu uściślenia diagnozy stosuje się drugi test o innych właściwościach (obliczony, tak jak w przykładzie wcześniejszym, lecz z innymi danymi), a następnie oblicza się prawdopodobieństwo wystąpienia SARS jako uwarunkowania wyników obu testów. W tym celu wykorzystuje się zależność:

$$P(SARS|T^\oplus_1, T^\oplus_2) = \frac{P(SARS \cap T^\oplus_1 \cap T^\oplus_2)}{P(T^\oplus_1 \cap T^\oplus_2)}$$

$$= \frac{P(T^\oplus_1 \cap T^\oplus_2|SARS)P(SARS)}{P(T^\oplus_1 \cap T^\oplus_2)}$$

$$= \frac{P(T^\oplus_1|SARS)P(T^\oplus_2|SARS)P(SARS)}{P(T^\oplus_1 \cap T^\oplus_2)}$$

W przypadku, gdy oba testy miałyby identyczną charakterystykę jak w obliczonym wcześniej przykładzie, to otrzymany pozytywny wynik z obu testów wskałby na prawdopodobieństwo blisko 100 razy większe niż w przypadku braku informacji.

$$P(SARS|T^\oplus_1, T^\oplus_2) = \frac{P(T^\oplus_1|SARS)P(T^\oplus_2|SARS)P(SARS)}{P(T^\oplus_1 \cap T^\oplus_2)}$$

Przy metodach probabilistycznych często stosowana jest wielkość nazywana entropią. Entropia jest to funkcjonal przyporządkowujący funkcji prawdopodobieństwa nieujemną liczbę. Jej zadaniem jest mierzenie zawartości informacyjnej rozpatrywanego zbioru zadań lub zdarzeń. Istnieje kilka określen entropii:

- entropia Shannona
  $$H_n(P) = -\sum_{i=1}^{n} p_i \log p_i$$
- entropia ważona
9.2. Reprezentacja wiedzy

\[ H_n(P; W) = - \sum_{i=1}^{n} w_i p_i \log p_i \]

- entropia ważona stopnia \( \beta \)

\[ H_n^\beta(P; W) = (2^{1-\beta} - 1)^{-1} \sum_{i=1}^{n} p_i \log p_i \]

Współczynnik pewności \( CF \)

Wprowadzenie współczynnika pewności \( CF \) do reprezentacji informacji niepełnej miało na celu zmniejszenie wymagań dotyczących dużej ilości danych i uniknięcie niewygodnych obliczeń związanych z wnioskowaniem probabilistycznym.

Pierwotnie współczynnik \( CF \) zdefiniowany był jako przyrost prawdopodobieństwa warunkowych. Pozwala na połączenie stopnia wiedzy oraz niewiedzy oraz przedstawienia ich w postaci jednej liczby. Został on określony przez twórców Shortliffe’a i Buchanana jako różnica miar \( MB \) i \( MD \). Miara wiarygodności \( MB(h, e) \) reprezentująca stopień potwierdzenia hipotezy \( h \) przez obserwacje \( e \). \( MD(h, e) \) jest to miara niepotwierdzenia hipotezy \( h \) przez \( e \). W literaturze można dość często spotkać się z interpretacją probabilistyczną tych wielkości:

\[
CF(h, e) = \begin{cases} 
1 & P(h) = 1, \\
MB & P(h|e) > P(h), \\
0 & P(h|e) = P(h), \\
-MD & P(h|e) < P(h), \\
-1 & P(h) = 0,
\end{cases}
\]

\[
MB(h, e) = \begin{cases} 
\frac{P(h|e) - P(h)}{1-P(h)} & P(h|e) > P(h)), \\
0 & \text{w przeciwnym przypadku,}
\end{cases}
\]

\[
MD(h, e) = \begin{cases} 
\frac{P(h) - P(h|e)}{P(h)} & P(h|e) < P(h)), \\
0 & \text{w przeciwnym przypadku,}
\end{cases}
\]

gdzie \( P(h) \) jest prawdopodobieństwem \( a \ priori \) hipotezy \( h \), \( P(h|e) \) prawdopodobieństwem \( a \ posteriori \). Powyższe podejście pozwala na interpretację przyrostową prawdopodobieństwa:

\[
P(h|e) = \begin{cases} 
P(h) + CF(h, e)[1 - P(h)], & CF(h, e) > 0, \\
P(h) - |CF(h, e)|P(h), & CF(h, e) < 0,
\end{cases}
\]

W systemach MYCIN (medyczne systemy ekspертowe) wiedza jest zapamiętywana w postaci reduły \( \text{jeśli } e, \text{to } h. \) Do każdej reguły powiązana jest pewna liczba \( CF \), reprezentująca zmiany wiarygodności hipotezy dla danej obserwacji \( e \). \( CF \) znajduje się w przedziale \([-1, 1]\). Podczas wnioskowania w modelu współczynnika pewności \( CF \) w oparciu o działanie interpretatora reguł, następuje zjawisko przechodzenia z reguły do reguły. Efektem przejścia jest powstanie drzewa wywodzącego wybrane i uaktywnione reguły wraz z ich kolejnością. Każdy z faktów może posiadać swój współczynnik pewności. Fakty te tworzą przesłanki pewnych reguł.
9. Reprezentacja informacji niepewnej i niepełnej

- przesłanka reguły zawiera wyrażenie zwierające operator AND (\&)

Jeżeli \( e_1 \) i (AND) \( e_2 \), to \( h \)

\[
CF(h, e_1 \& e_2) = \min\{CF(e_1, h), CF(e_2, h)\}
\]

- przesłanka reguły zawiera wyrażenie zawierające operator OR (\|)

Jeżeli \( e_1 \) lub (OR) \( e_2 \), to \( h \)

\[
CF(h, e_1 | e_2) = \max\{CF(e_1, h), CF(e_2, h)\}
\]

Dość często występują różne kombinacje połączeń reguł, z których wynika jedna konkluzja. Rozróżniamy dwa podstawowe układy, gdy:

- hipoteza \( h \) jest konkluzją więcej niż jednej reguły (połączenie równoległe),

\[
CF(h, e_1, e_2) = \begin{cases} 
CF(h, e_1) + CF(h, e_2) - CF(h, e_1)CF(h, e_2) & \text{dla } CF(h, e_1) \geq 0, CF(h, e_2) \geq 0 \\
CF(h, e_1) + CF(h, e_2) + CF(h, e_1)CF(h, e_2) & \text{dla } CF(h, e_1) < 0, CF(h, e_2) < 0
\end{cases}
\]

- hipoteza \( h \) jest konkluzją połączenia szeregowego reguł,

\[
CF(h, e_1) = \begin{cases} 
CF(e_1, e_2)CF(h, e_2) & \text{dla } CF(e_1, e_2) \geq 0 \\
- CF(e_1, e_2)CF(h, \neg e_2) & \text{dla } CF(e_1, e_2) < 0
\end{cases}
\]

Wartości dodatnie odpowiadają zwiększeniu wiarygodności hipotezy, natomiast wartości ujemne jej zmniejszeniu. Wartości przechodzą przez sieci wnioskowania zgodnie z określającymi je funkcjami. Różne typy połączenia zostały przedstawione na rys. 9.2.

Po przeprowadzeniu obliczeń oraz połączeniu niepewności \((e_1, e_2, e_5 \rightarrow h)\) dla przykładu zamieszczonego na rys. 9.3 można wnioskować, że posiada on współczynnik pewności równy \( CF = 0.059375 \). Ponieważ dziedzina ta ciągle się rozwija w późniejszych czasach zostały wprowadzone różne modyfikacje mające na celu likwidację niekonsekwencji między definicją współczynnika \( CF \) a funkcjami łączącymi niepewności informacji, jednakże w poniższej pracy nie będą one omawiane.

Logika rozmyta

Klasyczna logika operuje na dwóch wartościach (0,1) lub prawda i fałsz. Dodatkowo granica między jedną wartością a drugą jest wyraźna i jednoznaczna. Do logiki rozmytej należy rozszerzenie klasycznego rozumowania w taki sposób, aby
9.2. Reprezentacja wiedzy

Rys. 9.2: Połączenia równoległe i szeregowe w modelu współczynnika pewności $CF$.

Rys. 9.3: Propagacja współczynnika $CF$ przez przykładową sieć wnioskowania.
9. Reprezentacja informacji niepewnej i niepełnej

była ona bliższa ludzkiemu rozumowaniu. Logika ta wprowadza wartości pośrednie między standardowe 0 i 1. Rozmyte granice pomiędzy 0 i 1 dają możliwość zaistnienia wartości z tego przedziału (np.: prawie falso, w połowie prawda). Przykładem takiego rozumowania może być określenie wieku ludzi czy też wyznaczenie granicy wieku między ludźmi młodymi, w wieku średnim i starszymi. Logika klasyczna zmusza do przyjęcia stałych granic, np.: ludzie młodzi (0-30 lat), ludzie w wieku średnim (30-40 lat) i osoby starsze (po 40 roku życia). Jednak w subiektywnej ocenie samych osób przyjęcie ustalonych granic nie jest już tak oczywiste. Czy 28 latem zawsze powinien być zakwalifikowany do grupy osób młodych? Rozmycie granic podziału przedstawiono na rys. 9.4.

![Rys. 9.4: Przykład granic podziału w logice klasycznej i logice rozmytej.](image)

Logika rozmyta jest stosowana tam, gdzie trudno jest posługiwać się logiką klasyczną (na przykład przy opisie matematycznym złożonego procesu lub w przypadku, w którym wyliczanie zmiennych potrzebnych do rozwiązania jest niemożliwe). Najczęściej wykorzystywana jest w sterownikach, które mogą być zamontowane w prostych urządzeniach typu lodówka czy pralka. Można ją spotkać również w bardziej skomplikowanych urządzeniach, służących do przetwarzania obrazów, rozwiązywania problemów ulicznych czy unikania kolizji. Sterowniki wykorzystujące logikę rozmytą stosowane są też w połączeniu z sieciami neuronowymi.

Definicja zbiorów rozmytych opiera się na pewnym formalizmie matematycznym. Podobnie jak w klasycznej algebrze zbiorów, na zbiorach rozmytych można wykonywać pewne operacje. Podstawowymi operacjami wykonywanymi na zbiorach rozmytych są: negacja (NOT), suma (OR), iloczyn (AND). Operacje te mogą być realizowane w różny sposób, np. według wzorów zamieszczonych w tab. 9.1, 9.2, a w wyniku ich stosowania można uzyskać różne rezultaty.

9.2.2. Reprezentacja informacji niepełnej

Jedną z metod symbolicznego przetwarzania informacji niepełnej jest wykorzystanie klasycznej logiki pierwszego rzędu. Logika ta posiada wiele cennych właściwości. Pozwala ona na reprezentacje informacji niepełnej przez stwierdzenie, że:
### 9.2. Reprezentacja wiedzy

Tab. 9.1: Podstawowe operatory t-normy.

<table>
<thead>
<tr>
<th>Nazwa operacji</th>
<th>Wzór</th>
</tr>
</thead>
<tbody>
<tr>
<td>minimum (MIN)</td>
<td>$\mu_{A\cap B}(x) = \text{MIN}(\mu_A(x), \mu_B(x))$</td>
</tr>
<tr>
<td>iloczyn (PROD)</td>
<td>$\mu_{A\cap B}(x) = \mu_A(x) \cdot \mu_B(x)$</td>
</tr>
<tr>
<td>iloczyn Hamachera</td>
<td>$\mu_{A\cap B}(x) = \frac{\mu_A(x) \cdot \mu_B(x)}{\mu_A(x) + \mu_B(x) - \mu_A(x) \cdot \mu_B(x)}$</td>
</tr>
<tr>
<td>iloczyn Einstaina</td>
<td>$\mu_{A\cap B}(x) = \frac{\mu_A(x) \cdot \mu_B(x)}{2 - \mu_A(x) + \mu_B(x) - \mu_A(x) \cdot \mu_B(x)}$</td>
</tr>
<tr>
<td>iloczyn drastyczny</td>
<td>$\mu_{A\cap B}(x) = \begin{cases} \text{MIN}(\mu_A(x), \mu_B(x)) \text{ dla } \text{MAX}(\mu_A(x), \mu_B(x)) = 1 \ 0 \text{ poza tym} \end{cases}$</td>
</tr>
<tr>
<td>różnica ograniczona</td>
<td>$\mu_{A\cap B}(x) = \text{MAX}(0, \mu_A(x) + \mu_B(x) - 1)$</td>
</tr>
</tbody>
</table>

Tab. 9.2: Podstawowe operatory s-normy.

<table>
<thead>
<tr>
<th>Nazwa operacji</th>
<th>Wzór</th>
</tr>
</thead>
<tbody>
<tr>
<td>maksimum (MAX)</td>
<td>$\mu_{A\cup B}(x) = \text{MAX}(\mu_A(x), \mu_B(x))$</td>
</tr>
<tr>
<td>suma algebraiczna</td>
<td>$\mu_{A\cup B}(x) = \mu_A(x) + \mu_B(x) - \mu_A(x) \cdot \mu_B(x)$</td>
</tr>
<tr>
<td>suma Hamachera</td>
<td>$\mu_{A\cup B}(x) = \frac{\mu_A(x) + \mu_B(x) - 2 \mu_A(x) \cdot \mu_B(x)}{1 - \mu_A(x) \cdot \mu_B(x)}$</td>
</tr>
<tr>
<td>suma Einstaina</td>
<td>$\mu_{A\cup B}(x) = \frac{\mu_A(x) \cdot \mu_B(x)}{1 + \mu_A(x) \cdot \mu_B(x)}$</td>
</tr>
<tr>
<td>suma drastyczna</td>
<td>$\mu_{A\cup B}(x) = \begin{cases} \text{MAX}(\mu_A(x), \mu_B(x)) \text{ dla } \text{MIN}(\mu_A(x), \mu_B(x)) = 0 \ 1 \text{ poza tym} \end{cases}$</td>
</tr>
<tr>
<td>suma ograniczona</td>
<td>$\mu_{A\cup B}(x) = \text{MIN}(1, \mu_A(x) + \mu_B(x))$</td>
</tr>
</tbody>
</table>

- coś ma pewną wartość, bez wykazywania tej rzeczy: $(\exists x)(P(x))$
- wszystkie elementy danej klasy mają pewną własność, bez wyliczania elementów tej klasy: $(\forall x)(P(x) \Rightarrow Q(x))$
- przynajmniej jedno z dwóch stwierdzeń jest prawdziwe, bez rozstrzygania, które: $P \lor Q$
- pewne twierdzenie jest fałszywe (w przeciwieństwie do nie mówienia, że jest ono prawdziwe): $\neg P$ - założenia domkniętego świata

Wymienione punkty pozwalają na przetwarzanie informacji niepełnej. Pierwsze trzy są oczywiste i nie będą tu wyjaśniane. Ostatni odnosi się do bardzo ciekawego zagadnienia, który wyjaśnić można następującym przykładem z życia. Niech będą dostępne dwie bazy informacji: pierwsza – zawierająca listę losów wygrywających w loterii, druga – zawierająca spis telefonów. Z informacji zawartej w pierwszej bazie możliwe jest wyciągnięcie poprawnego wniosku, że jeśli nu-
mer danego losu nie znajduje się na liście, to właściciel losu nie wygrał. W drugim przypadku nie jest to już takie oczywiste. Brak danego nazwiska z książki telefonicznej może oznaczać kilka z rzeczy, niekoniecznie brak posiadania telefonu przez daną osobę. Możliwe powody, dla których dana osoba nie znajduje się w spisie abonentów, to: zastrzeżenie numeru, brak telefonu, nieaktualność danych.

9.3. Posługiwanie się wiedzą niepewną i niepełną w praktyce

W celu praktycznego przetestowania metod przetwarzania wiedzy niepewnej i niepełnej zaproponowano przeprowadzenie eksperymentów polegających na eliminacji niepewnych i niepełnych pomiarów sensorycznych podczas budowy mapy otoczenia. Założono, że źródłem danych pomiarowych będzie czujnik odległości Sharp GP2D12 połączony z serwem. Dokładności pomiaru wynika z dokładności czujnika, a niepełność pomiaru wynika ze skoku serw podczas obrotu (dokładności i pełności pomiarów zależy od wielkości tego skoku).

W celu zminimalizowania niepewności pomiarowych każde badanie otoczenia będzie przeprowadzone kilkukrotnie, a następnie wyciągnięta zostanie średnia otrzymanych wyników. Niepełność pomiarów ograniczona będzie poprzez zmniejszenie skoku serw, czyli zwiększenie próbek na skanowanej powierzchni.

9.3.1. Realizacja projektu


Mikrokontroler Atmega32 jest reprezentantem 8-bitowej rodziny opartej na architekturze RISC. Ze względu na swoją cenę oraz łatwości programowania zarówno w systemie Linux jak i Windows mikrokontroler ten cieszy się dość dużą popularnością. Do podstawowych jego własności należą:

- architektura AVR
  - 131 instrukcji - większość jednocyklowych
  - 32 x 8-bit rejestrów ogólnego przeznaczenia
  - możliwość pracy statycznej (0Hz)
  - do 16 MIPS przy 16MHz
  - wbudowany 2-cyklowy układ mnożący
- nieulotne pamięci danych i programu
  - 32K bajty programowanej w systemie pamięci programy Flash trwałość: 10000 cykli zapis/ kasowanie
9.3. Posługiwanie się wiedzą niepewną i niepełną w praktyce

Rys. 9.5: Widok skonstruowanego układu.

- Obszar Boot Code z Lock Bits: Programowanie w systemie przez program
  w obszarze Boot Operacje Read-While-Write
- 1024 bajty EEPROM trwałość ponad 100000 cykli zapis/ kasowanie
- 2K bajty wewnętrznej pamięci danych SRAM
- zabezpieczenie oprogramowania przez odczytem
- interfejs JTAG
  - Boundary-Scan
  - Funkcja On-chip Debug
  - programowanie Flash, EEPROM, fuse i lock-bitów przez JTAG
- urządzenia dodatkowe
  - dwa 8-bit liczniki z odrębnymi preskalerami u trybami porównania
  - jeden 16-bit licznik z oddzielnym preskalarem, trybem porównania i prze-
    chwytywania
  - licznik czasu rzeczywistego z oddzielnym oscyloskopem
  - cztery kanały PWM
  - interfejs TWI
  - programowany USART
  - interfejs SPI
  - programowalny watchdog z oddzielnym oscyloskopem
  - komputer analogowy
- specjalne cechy mikrokontrolera
  - samoczynny reset po włączeniu zasilania i dekoder napięcia zasilającego
  - przestrzajny wewnętrzny oscyloskop RC
9. Reprezentacja informacji niepewnej i niepełnej

Rys. 9.6: Schemat skonstruowanego układu.
9.3. Posługiwanie się wiedzą niepewną i niepełną w praktyce

- zewnętrzne i wewnętrzne źródło przerwań
- 6 trybów obniżonego poboru mocy

- I/O
  - 32 programowalne linie wejścia / wyjścia
  - obudowy: 40-pin DIL, 44 TQFP, 44 MLF

- zakres napięć zasilania:
  - 2,7 - 5,5V dla Atmega32L
  - 4,5 - 5,5V dla Atmega32

- prędkość pracy
  - 0 - 8MHz dla Atmega32L
  - 0 - 16MHz dla Atmega32

9.3.2. Przeprowadzone eksperymenty

Po uruchomieniu układu czujnik wysyła sygnał podczerwony, który na zasadzie odbicia od przeszkody powraca do czujnika, gdzie odczytywany jest jego kąt padania. Na podstawie wartości kąta sygnału powrotnego wyznaczana jest odległość, która na wyjściu układu reprezentowana jest w postaci napięcia. Wartość napięcia jest odczytywany przy pomocy portu analogowego. Poprzez analizę charakterystyk napięciowych obliczane są wynikowe odległości.

Tryby pracy układu

W przypadku pierwszego trybu pracy aplikacji w sposób ciągły zbierane są poszczególne wartości mierzonego odcinka, a następnie, po usrednieniu 80 ostatnich wyników, podawana jest odległość od przeszkody. Uśrednianie odbywa się w sposób ciągły. sygnał pobrany z czujnika w postaci odpowiedniej wartości jest umieszczany na 80 pozycji w liście. Aby stworzyć na niej miejsce, pierwszy element jest usuwany z listy a wszystkie pozostałe przesuwają się o jedno miejsce (co pokazano na rys. 9.7).

Celem eksperymentu jest pokazanie użytkownikowi zmienności odczytów stałej odległości w danej chwili. Użytkownik będzie mógł zaobserwować błędy pomiarów wynikających z różnego typu zakłóceń czy niedokładności sprzętowej. Wada pierwszego trybu pracy jest duża różnica pomiędzy pojedynczymi wynikami (uzyskane wyniki chwiotowe trudno jest traktować jako wynik poprawny).

Rys. 9.7: Schemat przejścia informacji w liście.
9. Reprezentacja informacji niepewnej i niepełnej

Metoda odczytu odległości


![Rys. 9.8: Okno wizualizacji pomiarów.](image)

Inną metodą odczytu informacji z mikrokontrolera jest ciągła obserwacja odczytywanych danych i wypisywanie ich w programie SimpleTerm SE. Wyświetlane w ten sposób dane obarczone są dość dużym błędem (o który już wcześniej wspomniano). Na rys. 9.9 przedstawiono przykładowy pomiar odległości.

9.3.3. Wnioski

W niniejszym rozdziale omówiono różne metody reprezentowania wiedzy niepełnej i niepewnej. Następnie pokazano praktyczny przykład urządzenia, z którego można pozyskać tego typu wiedzę. Metoda w nim wykorzystana opierała się na uśrednianiu wyników pomiarowych w sposób ciągły. Jej celem było określenie odległości z największym prawdopodobieństwem przy jak najmniejszym błędzie. Wykorzystany czujnik okazał się bardzo wrażliwy na wszelkie zakłócenia zewnętrzne.
9.3. Posługiwanie się wiedzą niepewną i niepełną w praktyce

Rys. 9.9: Przykładowy pomiar odległości w programie SimpleTerm SE.

Literatura


10.1. Temporalne bazy danych

10.1.1. Wprowadzenie


Temporalne bazy danych są rozwojem tradycyjnych baz danych. Pozwalają na przechowywanie zależności czasowych i formułowanie zapytań z uwzględnieniem czasu. Ich funkcjonowanie opiera się na temporalnych modelach danych. Im poświęcona jest dalsza część niniejszego rozdziału.

10.1.2. Modelowanie czasu w bazach danych

Modelowanie czasu na potrzeby temporalnych baz danych wymaga, aby w łatwy sposób dało się zintegrować jego reprezentację z modelem danych, przy
10.1. Temporalne bazy danych


Aspekt czasu wprowadzany jest do temporalnych baz danych poprzez mechanizm etykiet czasowych (ang. timestamp). Wyróżnione są tam dwie zasadnicze wielkości:

• **Valid time** — interwał czasu, w którym fakt jest, był lub będzie prawdziwy w rzeczywistym świecie. Obrazując to na przykładzie: adresy zamieszkania poszczególnych osób są ważne aż do momentu przeprowadzki. Baza danych przechowująca takie informacje zostaje wzbogacona o pola: **valid from**, **valid to**. Pola te muszą być wypełnione podczas dodawania danych do bazy.

• **Transaction time** — interwał czasu, w którym fakt jest aktualny w bazie danych. Inaczej mówiąc, jest to czas, w którym fakt jest przechowywany. **Valid time** przydatny jest do odtwarzania historii w odniesieniu do świata rzeczywistego. **Transaction time** jest przydatny od opisywania historii zmian. Na przykład po wprowadzeniu błędnych danych zamieszkania należy dokonać ich Korekty. Stare dane zostaną zastąpione nowymi, a przedział czasu, w którym stare dane obowiązywały, zostanie zamknięty i zostanie otwarty nowy przedział dla nowych danych. Dzięki wielkości **Transaction time** możliwe jest śledzenie modyfikacji danych. Wracając do przykładu, gdy dane dane adresowe celowo zakłamano, aby np. uniknąć płacenia podatków, przedłużenie historii zmian umożliwi odtworzenie starego adresu. Śledzenie wielkości **transaction time** odbywa się automatycznie przez bazę danych, nie ma więc sensu jej manualne modyfikowanie, jest to wręcz zabronione.

10.1.3. Rodzaje temporalnych baz danych

Ze względu na sposób reprezentacji czasu temporalne bazy danych można podzielić na następujące kategorie.

10. Reprezentacja wiedzy zmieniającej się w czasie


- **Bitemporalna** – zapewnia kompleksową obsługę czasu i wykorzystywana jest w rozbudowanych systemach bazodanowych. W bazie danych tego typu wykorzystuje się koncepcję bazy *historycznej* i bazy *odwracalnej* (posiada funkcjonalność obu typów). Na rys. [10.1d] przedstawiono wyobrażenie bitemporalnej bazy danych. Jak widać, jest ona połączeniem obydwu powyższych przedstawionych. Nowe kontenery reprezentujące kolejne stany bazy powstają zarówno wtedy, gdy dane zmieniają swój stan w odniesieniu do realnego świata (oś *valid time*), jak i w przypadku modyfikacji danych (oś *transaction time*).

10.1.4. Znaczniki czasowe

Podstawowym mechanizmem pozwalającym na śledzenie zmian stanów bazy danych jest mechanizm etykiet czasowych (*timestamp*). Okres, w jakim dany fakt jest prawdziwy w świecie rzeczywistym (bądź też okres, w którym dane są przechowywane w bazie) jest reprezentowany przez znacznik czasowy. Problem polega na tym, co dokładnie powinno zostać poddane takiemu oznaczeniu. Poniżej przedstawiono sposoby nanoszenia znaczników czasowych na dane wg [1]:

138
10.1. Temporalne bazy danych

Rys. 10.1: Wyobrażenia baz danych przedstawione w osiach czasów: valid time, transaction time. a) baza typu snapshot, b) baza historyczna, c) baza odwracalna, d) baza bitemporalna.

- **Tuple timestamping** (etykietowanie krotek) — pojęcie odnosi się do relacyjnych baz danych, przy czym ma też swój odpowiednik w bazach obiektowych. Każda krotka w danej relacji posiada znacznik czasowy. W historycznych bazach danych każda krotka posiada znacznik czasowy wyrażony w valid time, w odwracalnych bazach jest to transaction time, z kolei w bazach bitemporalnych – w obydwu wielkościach. Realizowane jest to poprzez dodanie odpowiednich atrybutów określających przedział czasu. Wadą takiego rozwiązania jest to, że część danych jest duplikowana. Na przykład można wyobrazić sobie krotkę przechowującą informacje o tym, że pracownik „A” w danym okresie zarabiał 100 zł. Następnie dostąpił podwyżkę i zarabia teraz 200 zł. Fakt, że pracownik zarabia 100 zł, stracił ważność i zostanie utworzona nowa krotka zawierająca
10. **Reprezentacja wiedzy zmieniającej się w czasie**

uaktualnioną wartość zarobków. Problem polega na tym, że podczas tworzenia nowej krótki atrybuty takie jak imię, nazwisko itp. zostaną również skopiowane.

- **Atribute timestamping** (etykietowanie atrybutów) — tutaj znaczniki czasowe nanoszone są na każdy atrybut z osobną. Modyfikacja pojedynczej wartości nie wymaga, jak poprzednio, kopiowania wszystkich pozostałych atrybutów. Historia zmian wartości przechowywana jest osobno dla każdego z atrybutów.

### 10.1.5. Przykłady tempoporanych baz danych

Chociaż problem temporalnego reprezentowania danych jest już znany i rozpatrywany od dłuższego czasu, trudno znaleźć kompletne implementację języka TSQL (temporalnego SQL, którego wersję 2.0 opracowano już pod koniec lat dziewięćdziesiątych). Producenci takich baz danych jak Oracle Database i PostgreSQL stosują własne, typowe dla nich sposoby reprezentacji temporalnych danych, niezgodne z TSQL.

Jednym z rozwiązań, w którym zaimplementowano temporalną wersję języka zapytań, jest aplikacja TimeDB. Jest to nakładka na standardową bazę danych transformująca zapytania z języka TSQL2 do SQL. Rozwiązanie to dokładnie opisane w podrozdziale [10.3]. Innym ciekawym i dość nietypowym rozwiązaniem jest wykorzystanie języka XML do przechowywania i zarządzania danymi zmieniającymi się w czasie. Rozwiązanie to przedstawiono w podrozdziale [10.2].

### 10.2. Temporalne rozszerzenie języka XML

Modelowanie i przetwarzanie temporalnej bazy danych w XML obszernie omówiono w [3]. W niniejszym podrozdziale przedstawiono najważniejsze aspekty tego zagadnienia wraz z podstawowymi informacjami o samym XML.

#### 10.2.1. Język XML


W XML-u dane przechowywane są w sposób tekstowy, opisywane poprzez znaczniki, w obrębie których te dane się znajdują. Ponieważ dokument XML zapisany jest tekstowo, nie występuje problem niekompatybilności pomiędzy różnymi systemami komputerowymi. Dlatego też język nadaje się szczególnie do przechowywania danych wymienianych za pośrednictwem Sieci. Najważniejsze elementy składni języka XML, w omawianym kontekście, to:

- pierwszy wiersz jest deklaracją XML;
10.2. Temporalne rozszerzenie języka XML

- istnieje element (znacznik) główny (ang. root - korzeń);
- znacznik rozpoczyna lewy nawias kątowy (<), a kończy prawy (>), pomiędzy którymi znajduje się nazwa znacznika i opcjonalnie lista parametrów z przypisanymi wartościami np.: <znacznik parametr="wartość">. Każdy znacznik otwierający posiada swój odpowiednik zamykający, złożony z nazwy poprzedzonej ukośnikiem, tutaj </znacznik>;
- pomiędzy znacznikiem otwierającym i zamykającym mogą znaleźć się konkretnie dane np.: <em>Tekst</em>;
- mogą istnieć znaczniki pojedyncze, wtedy ukośnik znajduje się przed znakiem >, np.: <znacznik parametr="wartość" /> jest odpowiednikiem <znacznik parametr="wartość"></znacznik>;
- znaczniki mogą zagnieździć się, ustanawiana jest tym samym relacja rodzic-potomek;
- znaczniki nie mogą się przeplatać, np.: zabroniona jest składnia <pierwszy> <drugi> </pierwszy> </drugi>.


Listing 10.1: Przykładowy dokument XML

```xml
<?xml version="1.0" encoding='UTF-8'?>
<element atrybut="wartosc">
  <owoc nazwa="gruszka">
    <charakterystyka kolor="#00ff00" masa="0.31"/>
    <opis>Owoc pochodzi z<br />
     <region>Suwalszczyzny</region> . . .</opis>
  </owoc>
  !— Komentarz —>
</element>
```


Popularność i otwartość języka skłania do zastanowienia się nad jego przydatnością w zarządzaniu informacją zmieniającą się w czasie. Okazuje się, że istnieją już pewne podejścia do realizacji tego typu zadań. Opracowane koncepcje zarządzania danymi temporalnymi w XML posiadają wiele wspólnych cech, dlatego w rozdziale tym skupiono się na wynikach jednej pracy 3. 141
10. Reprezentacja wiedzy zmieniającej się w czasie

10.2.2. Temporalny dokument XML

Dokument XML przedstawiany jest jako graf skierowany, z wyróżnionym wierzchołkiem początkowym (korzeniem dokumentu) i trzema rodzajami wierzchołków, których podział wynika ze specyfiki XML (wartości, atrybuty, elementy). Wyróżnione są ponadto dwa rodzaje krawędzi takiego grafu: zawierania i referencji. Te ostatnie łączą specjalny atrybut $\text{REF}$ elementu z innym elementem. Krawędzie zawierania albo łączą element z atrybutem, wartością, innym elementem, albo łączą atrybut z jego wartością. Każdy węzeł w grafie ma swój unikalny numer ($\text{ID}$).

Czas w dokumencie XML opisywany jest przy pomocy etykietowania krawędzi grafu. Etykieta z przedziałem czasowym opisuje pewną krawędź $(a) \xrightarrow{[t_1,t_2]} (b)$, że w czasie od $t_1$ do $t_2$ element $b$ należał do $a$. Krawędź zapisuje się w wygodnej do analizowania formie $e(n_i, n_j, T_e)$, gdzie $n_i$, $n_j$ to łączone elementy, natomiast $T_e$ to omawiany przedział czasowy. Natomiast w XML zapisuje się to inaczej. Fragment dokumentu z takimi etykietami przedstawiono na listingu 10.2. Ponieważ w XML element nie może mieć dwóch atrybutów o tej samej nazwie, wprowadza się dodatkowy znacznik zawierający listę atrybutów jako swoje elementy.

Listing 10.2: Opis czasu w XML

```xml
<osoba imie="Maria">
  <ATTRIBUTES>
    <nazwisko Time:From="0" Time:To="t1-1">Kowalska</nazwisko>
    <nazwisko Time:From="t1" Time:To="Now">Nowak</nazwisko>
  </ATTRIBUTES>
</osoba>
```

Czas opisuje się dyskretnie. Wyróżnia się dwa specjalne znaczniki czasowe: czas utworzenia dokumentu i chwilę obecną. Opisuje się tylko czas istnienia w bazie danych (ang. transaction time), ale wprowadzenie czasu prawdziwości faktów (ang. valid time) nie powinno nastręczać trudności.

Czas życia elementu definiuje się jako sumę przedziałów czasowych wszystkich wchodzących do krawędzi zawierania. Czas życia elementu głównego to przedział od $t_0$ (czas założenia bazy) do teraz.

Temporalny dokument XML można zdefiniować jak następuje [3].

**Definicja 10.2.1** Temporalny dokument XML to graf poszerzony o etykiety temporalne, który spełnia poniższe warunki:

1. Suma przedziałów czasowych krawędzi zawierania wychodzących z elementu zawiera się w czasie życia elementu.
2. Suma przedziałów czasowych krawędzi zawierania wchodzących do elementu tworzy jeden zwarty przedział czasowy, a ich iloczyn jest przedziałem pustym.

3. Dla każdej chwili czasu $t$ podgraf złożony ze wszystkich krawędzi zawierania $e_c$ takich, że $t$ należy do przedziału czasowego $e_c$, jest drzewem, którego korzeń jest elementem głównym. Takie drzewo nazywane jest zrzutem dokumentu w czasie $t$ i oznaczane jako $D(t)$.

4. ID elementu pozostaje niezmienne we wszystkich zrzutach dokumentu.

5. Dla każdej krawędzi zawierania $e_z(n_i, n_j, T_{e_z})$, jeśli $n_j$ jest atrybutem typu REF, więc istnieje krawędź $e_r(n_j, n_k, T_{e_r})$, zachodzi $T_{e_z} = T_{e_r}$.

6. Przedział czasowy krawędzi referencyjnej zawiera się w czasie życia elementu, na który ona wskaże.

Graf, który nie spełnia definicji 10.2.1 w pewnym zakresie, nazywany jest nie-spójnym. Rozróżnia się cztery typy niespójności (w [3] zaproponowano szereg algorytmów wykrywających i naprawiających niespójności):

1. istnieje wychodząca krawędź zawierania, której przedział czasowy wychodzi poza czas życia elementu;
2. przedziały czasowe krawędzi zawierania wchodzących do elementu nie tworzą ciągłego przedziału lub ich iloczyn nie jest przedziałem pustym;
3. istnieje cykl w jakimś zrzucie dokumentu;
4. istnieją dwa elementy z takim samym ID.

Modyfikacje danych w temporalnym XML odbywa się wg następujących zasad. Nowy element można dodać tylko do elementu bieżącego. Element „A”, do którego dodawany jest nowy element „B” przestaje być bieżącym elementem. Górna granica jego przedziału czasowego przestaje być „Now”, na rzecz chwili wykonywania tej operacji $t_1$ pomniejszonej o 1. Przedział czasowy nowego elementu rozpoczyna się od $t_1$ i kończy na „Now”. Po operacji dodawania elementu naruszone zostaje spójność typu drugiego i należy to naprawić, stosując na przykład algorytm podany w [3].


10.2.3. Język zapytań dla temporalnego XML

Język XPath 2.0

Język ścieżek XML, XPath 2.0 (ang. XML Path Language) służy do adresowania zbiorów elementów w dokumencie XML. Jest podzioborem XQuery (ang. XML Query Language), służącego do przeszukiwania dokumentów XML. Specyfikacje zarówno XPath, jaki i XQuery, znajdują się na stronach W3C.

Ścieżka w XPath składa się z następujących elementów:
10. Reprezentacja wiedzy zmieniającej się w czasie

osi będących zbiorami węzłów o określonym pokrewieństwie wobec odpowiednich węzłów, względem których dokonuje się przeszukiwania (bieżących);

nazw badanych węzłów;

predykatów, za pomocą których testowane są odpowiednie węzły, np. węzeł[element = 'szukany']


Przykładowe zapytanie, pozwalające wybrać imiona tych studentów z grupy, którzy mają element obecności większy od 5:

/grupa/student[obecności > 5]/imię

Temporalny XPath

TXPath to zaproponowane w \[3\] rozszerzenie XPath pozwalające na formułowanie zapytań dla temporalnego XML. Wybierane elementy to pary (element, przedział czasowy) takie, że element spełnia dane wyrażenie nieprzerwanie w przedziale przedział czasowy. Zapytanie:

//kierunek[nazwa="automatyka i robotyka"]//
student[@from <= 2005 and @to='Now']


• Zapytanie distinct-values(wyrażenie) zwraca jeden element w przypadku, kiedy kilka takich samych elementów spełniających wyrażenie posiada nachodzące na siebie lub występujące tuż po sobie przedziały czasowe.

• Agregacja XPath 2.0 może być stosowana do przedziałów czasowych. Zapytanie o imiona graczy, którzy grali w klubie Magic, kiedy Kowalski po raz pierwszy dołączył do tego klubu ma następującą postać (za \[3\]):

let $m= min(//klub[nazwa='Magic']//
gracz[imię='Kowalski']/@from)
return
//klub[nazwa='Magic']//gracz[$m >= @from
and $m <= @to]/imię

• Łatwo wygenerować zrzut dla konkretnego czasu, jak w przykładowym zapytaniu o studentów, którzy studiowali 16.12.2009:

//kierunek[nazwa="automatyka i robotyka"]//student[
@from => '16.12.2009' and @to <= '16.12.2009']
10.3. Przykład zastosowanie temporalnej bazy danych

W celu zobrazowania korzyści, jakie w praktyce daje przetwarzanie danych temporalnych, napisano prosty program śledzący aktywność użytkowników systemu operacyjnego. Informacje pozyskane w ten sposób mogą pozwolić na konfigurowanie systemu z uwzględnieniem preferencji wybranych jego użytkowników. Tego typu program może być również wykorzystany do analizowania aktywności pracowników firmy.

10.3.1. Ogólna charakterystyka rozwiązania

Aplikacja zajmuje się przechowywaniem informacji na temat użytkowników oraz uruchamianych przez nich procesów w systemie operacyjnym opartym o jądro Linux. Zapisywane są interwały czasowe, w których poszczególni użytkownicy są zalogowani oraz interwały, w których poszczególne programy są używane. Prosty interfejs użytkownika pozwala na wyświetlanie wybranych danych w formie tabeli. Dodatkowo, jeżeli dostępny jest program Gnuplot (http://www.gnuplot.info/), możliwa jest graficzna reprezentacja wyników zapytań. Aplikację zbudowano korzystając z:

- temporalnego relacyjnego systemu zarządzania bazą danych (TimeDB 2.2, http://www.timeconsult.com/Software/Software.html);
- programu na bieżąco uaktualniającego zapisane dane;
- interfejsu użytkownika przyjmującego zapytania i wyświetlającego odpowiedzi.

Dwa ostatnie elementy powyższej listy to programy napisane w języku Java przez autorów tego rozdziału. Korzystano z „Sun Java(TM) Development Kit (JDK) 5.0” oraz „Sun Java(TM) Runtime Environment (JRE) 5.0”. Wersja 5.0 wspomnianego zestawu narzędzi Java jest najwyższą kompatybilną ze sterownikiem JDBC dla aktualnie dostępnej najnowszej bazy danych Oracle Express Edition.

10.3.2. Dane temporalne i baza danych

Podstawową częścią aplikacji jest TimeDB — nakładka na tradycyjną relacyjną bazę danych, interpretująca zapytania o charakterze temporalnym. Jest to aplikacja napisana w języku Java, która do prawidłowego działania wymaga bazy danych Oracle, Sybase lub IBM Cloudscape. Aplikacja komunikuje się z systemem zarządzania bazą danych poprzez sterownik JDBC. Udostępnia ona użytkownikowi możliwość formułowania zapytań w języku TSQL2, transformuje je na standardowy język zapytań SQL rozumiany przez DBMS. TimeDB posiada własny interfejs, TDBCI, do komunikacji z aplikacją użytkownika. Na rys. 10.2 zobrazowano strukturę rozwiązania wykorzystującego TimeDB.

W zapytaniach do TimeDB zwyczajne zapytanie języka SQL poprzedza się tzw. „flagą czasu”. Modele zapytań:
10. Reprezentacja wiedzy zmieniającej się w czasie

Rys. 10.2: TimeDB jako warstwa pośrednicząca między programem użytkowym a DBMS (wg dokumentacji pakietu).

- *snapshot* - zapytanie odnosi się tylko do danych prawdziwych w danej chwili (select * from proces);
- *sequenced* - zapytanie jest powtórzone dla wszystkich przechowywanych stanów bazy danych (validtime period [2009/6/12-14-forever) select *

Nakładka TimeDB jest dostępna jedynie w wersji beta (nie jest wersja finalna). Posiada wiele ograniczeń, które nie pozwalają na jej zastosowanie w systemie produkcyjnym. Kluczowe ograniczenia występujące w tej wersji oprogramowania, to:

- rozpatrywanie tylko **valid time**, **brak transaction time** i **bazy typu bitemporal**;
- **brak operacji update**, dostępne są tylko **insert** i **delete**.

Dodatkowo bardzo istotnym ograniczeniem jest brak pełnej implementacji języka TSQL2. Mianowicie:

- interwały czasu mogą być tylko reprezentowane przez stałą wartość (przykład: validtime period [1980-1990) select ...),
- nie ma możliwości odwołania się do stempli czasowych.

Istnieje starsza wersja TimeDB o większych możliwościach, posiadająca interfejs programowania w języku Perl. Nie została ona przez autorów tego opracowania sprawdzona. Dokładniejszy opis oraz przykłady zapytań dostępne są w dokumentacji TimeDB.

10.3.3. Dwuczesciowa struktura rozwiązania

Rozwiązanie składa się z dwóch osobnych programów. Pierwszy program zajmuje się uaktualnianiem informacji o załogowanych użytkownikach i urucho-
10.3. Przykład zastosowanie temporalnej bazy danych

![Diagram](image)

**Rys. 10.3: Schemat wzajemnych powiązań poszczególnych składowych rozwiązania.**

mionych przez nich procesach. W tym celu z częstotliwością 1 Hz wywoływane zostaje polecenie systemowe `ps`. Wynik tego polecenia jest następnie przetwarzany i analizowany w poszukiwaniu zmian w stosunku do stanu zarejestrowanego w chwili poprzedniej. Program ten docelowo może zostać przekształcony w daemon systemowy. Drugi program to klient, którego zadaniem jest tłumaczenie prostych polecień użytkownika na zapytania TSQL i wyświetlanie wyników. Strukturę stworzonego oprogramowania przedstawiono na rys. [10.3]

Aplikacja klienta pozwala na formułowanie temporalnych zapytań dotyczących logowań użytkowników, uruchamianych programów. Obsługiwane są następujące zapytania:

- **users_active** — aktualnie zalogowani użytkownicy,
- **users_history** — historia logowań,
- **user_history 'user'** — historia logowań danego użytkownika,
- **users_active_in 'period'** — użytkownicy zalogowani w danym przedziale czasu,
- **proces_active** — aktualnie uruchomione procesy,
- **proces_history** — historia aktywności procesów,
- **proces_history 'proces'** — historia aktywności danego procesu,
- **user_proces_active 'user'** — aktualnie uruchomione procesy danego użytkownika,
- **user_proces_history 'user'** — historia uruchamianych procesów przez danego użytkownika,
- **proces_active_in 'period'** — procesy uruchomione w danym przedziale czasu,
- **user_proces_active_in 'user' 'period'** — procesy uruchomione w danym przedziale czasu przez danego użytkownika.
10. Reprezentacja wiedzy zmieniającej się w czasie

10.3.4. Przykładowe zapytania i odpowiedzi

Wyszukiwanie użytkowników zalogowanych w danym przedziale czasowym
Poniżej przedstawiono przykładowe zapytanie o użytkowników zalogowanych do systemu w danym przedziale czasu i uzyskaną odpowiedź.

$ java Client users_active_in 2010/1/5~23:12:48-2010/1/5~23:30:00

::Request::
......
[2010/1/5~23:12:48-2010/1/5~23:30:00) domin
[2010/1/5~23:12:48-2010/1/5~23:30:00) rdkt
[2010/1/5~23:12:48-2010/1/5~23:25:49) krystek

Wyszukiwanie procesów uruchomionych w danym przedziale czasowym
Poniżej przedstawiono przykładowe zapytanie o procesy uruchomione w danym przedziale czasu i uzyskaną odpowiedź.

$ java Client proces_active_in 2010/1/5~22:30-2010/1/5~22:35

::Request::
.................
[2010/1/5~22:33:14-2010/1/5~22:33:15) bash 20543 rdkt
[2010/1/5~22:33:14-2010/1/5~22:33:15) bash 20541 rdkt
[2010/1/5~22:33:14-2010/1/5~22:35) bash 20531 rdkt
[2010/1/5~22:33:14-2010/1/5~22:35) sshd 20530 rdkt
[2010/1/5~22:33:53-2010/1/5~22:33:58) less 20714 rdkt
[2010/1/5~22:34:13-2010/1/5~22:34:24) ssh 20791 rdkt
[2010/1/5~22:34:29-2010/1/5~22:35) ssh 20852 rdkt
[2010/1/5~22:34:31-2010/1/5~22:35) zsh 20862 domin

Graficzna reprezentacja wyników
Ponieważ analiza tak przedstawionych rezultatów nie jest dla człowieka łatwa ani intuicyjna, wprowadzono możliwość graficznego przedstawienia danych otrzymanych w wyniku zapytania o procesy. W takim przypadku generowane przez program są dwa pliki: dane w formacie Gnuplota (data.dat) i plik z listą poleceń Gnuplota do wygenerowania wykresu (plotscript). Oś odciętych reprezentuje tutaj czas, oś rzędnych — PID. Każdemu procesowi nadaje się etykietę z nazwą użytkownika i nazwą procesu. Przykład tak utworzonego raportu przedstawiono na rys. [10.4].

W celu wygenerowania wykresu można uruchomić program Gnuplot z polówki poleceniem:

$ gnuplot plotscript -persist
10.3. Przykład zastosowanie temporalnej bazy danych

Rys. 10.4: Przykład raportu w programie Gnuplot.

Do oglądania wykresów zawierających duże ilości danych niezbędna jest funkcja powiększania wybranych obszarów wykresu. Taką możliwość udostępnia Gnuplot z terminalem (wyjściem) ustawionym na „wxt”.

Literatura

11.1. Wprowadzenie


Oprócz filozoficznej problematyki natury czasu, logika temporalna znajduje zastosowanie głównie w informatyce — w temporalnych bazach danych, przy analizie wykonywania programów i ich poprawności. Wśród innych zastosowań można wymienić lingwistykę (analiza czasów gramatycznych), sztuczną inteligencję (planowanie kolejności akcji, przetwarzanie wiedzy zmiennej w czasie) i fizykę (analizowanie nowych teori, czas — czwarty wymiar).

Logiki temporalne można podzielić na dwie grupy — z liniową oraz rozgałęzioną strukturą czasu. Logiki te mogą traktować czas jako ciągły lub, co jest częściej spotykane, jako dyskretny. Najprostszą i zarazem najbardziej rozpowszechnioną jest LTL (ang. linear temporal logic), używająca dyskretnego i liniowego modelu czasu. Rozszerzeniem LTL na rozgałęzione modele czasu są CTL* oraz CTL, w których dodano dwa dodatkowe operatory wraz z ograniczeniami. W dalszej części rozdziału przedstawiono podstawowe własności wspomnianych logik.

11.2. Podstawowe operatory logik temporalnych

11.2.1. Operatory czasu liniowego

Zależności czasowe można opisywać za pomocą zwykłego rachunku predykatów, jednak dowodzenie twierdzeń w tej notacji jest dość trudne i niezbyt wygodne. Logika temporalna została stworzona specjalnie z myślą o wnioskowaniu
11.2. Podstawowe operatory logik temporalnych

z użyciem czasu. Umożliwia bezpośrednie określanie relacji między zmiennymi wyrażającymi czas, przez co zapis staje się prostszy i bardziej elegancki.

Zdaniową logikę temporalną zbudowano z rachunku zdań pozbawionego kwantyfikatorów: ∀ — dla każdego oraz ∃ — istnieje takie. Dozwolone są dowolne zmienne zdaniowe, wszystkie spójniki, w tym: ¬ — negacja, ∧ — koniunkcja, ∨ — alternatywa, → — implikacja i ↔ — równoważność. Dodatkowo wprowadzono trzy prefiksowe, unarne (jednoargumentowe) operatory temporalne:

• □ — zawsze, oznaczany również G (od ang. globally),
• ◊ — kiedyś, także F (od ang. finally),
• ○ — następny, X (od ang. next).

Operator uniwersalny □ nieformalnie znaczy „dla każdej chwili w przyszłości”, a zapis □p tłumaczy się jako: „od danego momentu już zawsze będzie zachodziło p”. Z kolei ◊ to egzystencjonalny operator oznaczający „dla pewnej chwili w przyszłości”, a ◊q oznacza, że kiedyś w przyszłości będzie miało miejsce q. Należy wspomnieć, że operatory te mają priorytet spójnika logicznego negacji (¬).

Twierdzenie 11.2.1 □p → ◊p

Twierdzenie 11.2.2 (Dualność) □p ↔ ¬◊¬p


Twierdzenie 11.2.3 □p ↔ ◊◊p

Trzeci operator unarny następny (○) można zdefiniować w logikach temporalnych w czasie dyskretnym. Czas taki charakteryzuje się tym, że jest podzielony na momenty, w których stan świata pozostaje niezmieniony. Przejścia między stanami są nieskończenie krótkie. Model ten nadaje się do odwzorowania pracy procesora, gdzie dyskretnym momentem odpowiadają takty zegara. W przypadku kiedy czas jest ciągły można zastąpić go dyskretnym, co wprowadza pewne uproszczenia w obliczeniach.

11. Logika temporalna

zdaniowych. Pierwsze z nich (11.2.4) tłumaczy się następująco: \( r \) będzie kiedyś fałszywe, a w kolejnej dyskretnej chwili \( p \) będzie prawdziwe. Natomiast drugie (11.2.5): zawsze będzie tak, że jeśli w pewnej chwili zajdzie \( q \), to w następnej chwili przestanie zachodzić.

Twierdzenie 11.2.4 \( \Diamond (\neg r \land \Box p) \)

Twierdzenie 11.2.5 \( \Box (q \rightarrow \Box \neg q) \)

11.2.2. Operatory poprzedzania

Poznane operatory nie pozwalają na wyrażenie, że \( p \) zachodzi przed \( q \). Do tego celu zostały wprowadzone dodatkowe, dwuargumentowe operatory temporalne:

- \( p U q \) — (od ang. until) kiedyś nastąpi \( q \), ale do tego czasu będzie \( p \),
- \( p W q \) — (od ang. waiting for) \( p \) będzie zachodziło w trakcie czekania na \( q \),
- \( p R q \) — (od ang. release) \( q \) będzie zachodziło tak długo aż nie zajdzie \( p \).

11.2.3. Operatory przeszłości

Ani operatory modalne ani operatory poprzedzania nie są w stanie wyrazić wymagania, że pewne zdarzenie musiało nastąpić od czasu wystąpienia innego wydarzenia. Z tego powodu powstały kolejne dwa binarne operatory — operatory przeszłości:

- \( p S q \) — odkład (ang. since); wystąpienie \( q \) poprzedza wystąpienie \( p \),
- \( p B q \) — wstecz (ang. back-to); jeśli było \( q \) to po nim było \( p \).

Operator \( B \) jest słabszą wersją \( S \). Na ich podstawie definiuje się operatory unarne przeszłości:

- \( ♦ \) — kiedyś (ang. once); operator egzystencjalny, definiowany jako \( ♦ p = \text{true} S p \),
- \( ⊟ \) — jak dotąd (ang. so-far); uniwersalny, definiowany następująco: \( ⊟ p = \neg ♦ \neg p \),
- \( ⊖ \) — poprzednio; jest to wersja \( ₜ \) dotyka przeszłości.

Operator \( ⊖ \) wymaga krótkiego wyjaśnienia. Przy zapisie \( ⊖ q \) oznacza, że zachodzi on w stanie \( s_i \), jeżeli \( q \) zachodziło w \( s_{i-1} \). Przypadek szczególnym jest stan \( s_0 \), dla którego formuła \( ⊖ q \) nie może być spełniona, ponieważ ten stan ten nie ma poprzednika. Dlatego powstała słabsza wersja \( ⊖ \), która odgórnie przyjmuje, że w \( s_0 \) formuła jest spełniona.

11.2.4. Operatory czasu rozgałęzionego

W logice temporalnej czasu rozgałęzionego „powracają” kwantyfikatory, które już „pożegnano” przy omawianiu operatorów czasu liniowego. Do formalnego zdefiniowania logiki temporalnej z modelem czasu rozgałęzionego kwantyfikatory są niezbędne, gdyż wraz ze znawanymi doskonale \( □ \) (zawsze) oraz \( ♦ \) (kiedyś)
11.3. Liniowa logika temporalna

tworzą one operatory złożone, umożliwiające kwantyfikację po ścieżkach oraz po stanach ścieżek wybranych przez kwantyfikator:

• \( \forall \Box \) — dla wszystkich ścieżek \( \pi \) i wszystkich stanów \( s \in \pi \),
• \( \exists \Box \) — dla pewnej ścieżki \( \pi \) i wszystkich stanów \( s \in \pi \),
• \( \forall \Diamond \) — dla wszystkich ścieżek \( \pi \) i pewnego stanu \( s \in \pi \),
• \( \exists \Diamond \) — dla pewnej ścieżki \( \pi \) i pewnego stanu \( s \in \pi \).

11.3. Liniowa logika temporalna

11.3.1. Definicja

Dfnicja logiki LTL (ang. linear temporal logic) jest następująca.

Dfnicja 11.3.1 (LTL) Jeżeli relacja \( \rho \) spełnia warunek, że dla każdego stanu istnieje dokładnie jeden stan, do którego można przejść bezpośrednio z tego stanu (różnego od niego samego), to logikę taką nazywamy logiką temporalną czasu liniowego (LTL).

11.3.2. Składnia

Jest to najprostsza logika temporalna omawiana w tym rozdziale. Jak sama nazwa wskazuje, logika ta wykorzystuje liniowy oraz dyskretny model czasu. Stosuje się tu pięć podstawowych operatorów temporalnych, tzn.:

• \( G \) \( \rho \) (od ang. globally), oznaczający, że od tej chwili włącznie, \( \rho \) będzie zachodziło już zawsze,
• \( F \) \( \rho \) (od ang. finally), oznaczający, że kiedyś (w przyszłości, ale nie wiadomo dokładnie kiedy) zajdzie \( \rho \),
• \( X \) \( \rho \) (od ang. next), oznaczający, że \( \rho \) zajdzie w chwili, która nastąpi po obecnej,
• \( qU \rho \) (od ang. until), oznaczający, że kiedyś zajdzie \( \rho \), ale zanim to nastąpi będzie \( q \),
• \( qR \rho \) (od ang. release), oznaczający, że \( \rho \) będzie zachodziło tak długo, aż zajdzie \( q \).

Formuły LTL buduje się ze zdań atomowych \( AP = p_1, p_2, p_3, \ldots \) w następujący sposób:

• wszystkie \( p_i \in AP \) są formułami LTL,
• jeśli \( p \) jest formułą LTL to \( *p \) jest również formułą LTL, gdy \( * \in \{\neg, G, F, X \} \),
• jeśli \( p_1 \) oraz \( p_2 \) są formułami LTL to \( p_1 * p_2 \) jest również formułą LTL, gdy \( * \in \{\land, \lor, \rightarrow, U, R \} \).

Wszystkie powyższe operatory można zastąpić wyrażeniami zbudowanymi z tylko dwóch operatorów: \( X \) oraz \( U \).

11.3.3. Ważniejsze twierdzenia

Poniżej przedstawiono wybrane twierdzenia logiki temporalnej.

153
11. Logika temporalna

Twierdzenie 11.3.2 (Przechodniość) $Gp \iff Gp$

Twierdzenie 11.3.2 oznacza, że złożenie dwóch operatorów $G$ jest równoważne jednemu takiego operatorowi. Jest to prawdziwe dla dowolnej liczby składanych operatorów, również dla operatora $F$.

Twierdzenie 11.3.3 (Rozdzielność) $X(p \land q) \iff (Xp \land Xq)$

Twierdzenie 11.3.3 o rozdzielności $X$ względem koniunkcji.

Twierdzenie 11.3.4 (Rozdzielność) $G(p \land q) \implies (Gp \land Gq)$

Twierdzenie 11.3.4 rozdzielność $G$ względem koniunkcji.

Twierdzenie 11.3.5 (Rozdzielność) $(Gp \land Gq) \implies (G(p \land q))$

Twierdzenie 11.3.5 - relacja odwrotna do (11.3.4) również jest prawdziwa.

Twierdzenie 11.3.6 (Wymiana) $GXp \iff XGp$

Twierdzenie 11.3.6 - operatorzy $G$ i $X$, kiedy występują w swoim bezpośrednim sąsiedztwie, można zamieniać miejscami.

11.3.4. Zastosowania

Do praktycznych zastosowań LTL należy weryfikacja algorytmów. Sprawdza się czy:

- jeśli chce to kiedyś wejść do sekcji krytycznej,
- jeśli nieskończenie wiele razy chce wejść do sekcji krytycznej, to nieskończoność wiele razy wejdę do sekcji krytycznej,
- jeśli wejdę do sekcji krytycznej to kiedyś z niej wyjdę.

Kolejnym zastosowaniem jest weryfikacja całych programów. W metodzie tej z kolei sprawdza się czy:

- jeśli chce to dostanę kiedyś dostęp do procesora,
- jeśli chce nieskończoność wiele razy dostać dostęp do procesora to nieskończenie wiele razy go kiedyś dostanę.

Kolejnym ważnym zastosowaniem jest model checking, czyli automatyczna weryfikacja danego modelu pod względem zadanych własności. Jednym z popularniejszych programów stworzonych do weryfikowania poprawności modeli jest SPIN (ang. Simple Promela Interpreter). Do sprawdzania poprawności wykorzystuje się w nim język modelowania - PROMELA.

11.4. Logika CTL*

CTL* (ang. computation tree logic) to logika rozszerzająca możliwości LTL na drzewiaste, tzn. rozgałęzione modele czasu. Dodano tu dwa dodatkowe operatory unarne, będące odpowiednikami kwantyfikatorów $\forall$ i $\exists$.
11.5. Logika CTL

• $A p$ — dla każdej ścieżki zachodzi $p$,
• $E p$ — istnieje taka ścieżka dla której zachodzi $p$.

Występuje tu ograniczenie składania operatorów, tzn. każdy operator właściwy dla LTL musi być poprzedzony przez operator $A$ lub $E$.

11.5. Logika CTL

11.5.1. Definicja

Definicja 11.5.1 (CTL) Jeżeli relacja $p$ nie spełnia warunku takiego, że dla każdego stanu istnieje dokładnie jeden stan, do którego można przejść bezpośrednio z tego stanu (różnego od niego samego), to logikę taką nazywamy logiką temporalną czasu rozgałęzionego (CTL).

11.5.2. Składnia

Logika ta jest ograniczeniem CTL* i również w tym przypadku ograniczenie dotyczy składania operatorów. Mianowicie operatorzy mogą występować tylko i wyłącznie w parze: operator ścieżkowy, stanowy. Wszystkie dopuszczalne połączenia to: $A \mathcal{F}$, $E \mathcal{F}$, $A \mathcal{X}$, $E \mathcal{X}$, $A \mathcal{U}$, $E \mathcal{U}$, $A \mathcal{R}$, $E \mathcal{R}$. Każdy spośród tych operatorów można zapisać przy pomocy trzech par: $E \mathcal{U}$, $E \mathcal{A}$ i $E \mathcal{X}$.

Składnię CTL od strony formalnej przedstawia się jak następuje. Niech $V = \{v_1, v_2, \ldots\}$ będzie zbiorem zmiennych zdaniowych. Zbiór formuł stanowych $FS$ to najmniejszy zbiór, taki że:

- $V \subseteq FS$,
- $\bot \in FS$,
- jeśli $p, q \in FS$, to także $(p \rightarrow q) \in FS$,
- jeśli $\alpha \in FP$, to także $(A \alpha), (E \alpha) \in FS$.

Zbiór formuł ścieżkowych $FP$ to najmniejszy zbiór, taki że:

- jeśli $p, q \in FS$, to $(p \mathcal{U} q) \in FP$,
- jeśli $p \in FS$, to $(A p), (E p) \in FS$.

Spójniki $\neg, \land, \lor, \rightarrow$ definiuje się w formułach stanowych za pomocą $\bot$ oraz $\rightarrow$.

11.5.3. Semantyka

W logice CTL do weryfikacji prawdziwości formuł używa się struktur Kripkego. Jest to (intuicyjnie) proces, z którego każdym stanem związuje się zbiór zmiennych zdaniowych (faktów), które są w tym stanie prawdziwe. Wraz z upływem czasu i wykonywaniem się procesu, prawdziwość zmiennych zdaniowych ulega zmianie. Struktura Kripkego to czwórka uporządkowana $(S, R, I, \delta)$, gdzie

- $(S, R, I)$ jest procesem,
- Funkcja $\delta : S \rightarrow V$ przyporządkowuje każdemu stanowi zbiór $f$ zmiennych zdaniowych prawdziwych w tym stanie.
11. Logika temporalna

Funkcję $\delta$ możemy uważać za wartościowanie zmiennych zdaniowych, ale wartościowanie to zależy od stanu, w którym znajduje się proces.

Mając strukturę Kripkego i formułę CTL można rozstrzygać, czy jest ona prawdziwa w tej strukturze. Najpierw jednak należy zdefiniować prawdziwość formuły stanowej w określonym zadany stanie. Trzeba także określić prawdziwość formuły ścieżkowej dla zadanej ścieżki.

Niech $K = (S, R, I, \delta)$ będzie strukturą Kripkego, a $s \in S$ dowolnym stanem.

1. Definiowana jest relacją $K, s \models \phi$ dla formuły stanowej $\phi$:
   - dla $\nu \in V$ mamy $K, s \models \nu$ wtedy gdy $\nu \in \delta(s)$,
   - nieprawda, że zachodzi $K, s \models \bot$,
   - $K, s \models \phi \rightarrow \psi$ wtedy gdy (jeśli $K, s \models \phi$ to $K, s \models \psi$),
   - $K, s \models \exists \alpha$ wtedy gdy dla wszystkich $p \in \text{Path}_s$ zachodzi $K, p \models \alpha$,
   - $K, s \models \forall \alpha$ wtedy gdy dla pewnej ścieżki $p \in \text{Path}_s$ zachodzi $K, p \models \alpha$.

2. Definiowana jest $K, p \models \alpha$ dla formuły ścieżkowej $\alpha$:
   - $K, p \models G\phi$ wtedy gdy $\forall i \in K, p(i) \models \phi$,
   - $K, p \models F\phi$ wtedy gdy $\exists i \in K, p(i) \models \phi$,
   - $K, p \models X\phi$ wtedy gdy $K, p(1) \models \phi$.

Można powiedzieć, że struktura Kripkego $K = (S, R, I, \delta)$ jest modelem dla formuły stanowej $\phi$ wtedy gdy dla każdego $s \in I$ zachodzi $K, s \models \phi$.

11.5.4 Przykładowe formuły

Poniżej przedstawiono przykładowe formuły w logice CTL.

- $\forall p$ — $p$ zachodzi w każdym stanie każdej ścieżki,
- $\exists p$ — być może zawsze będzie miało miejsce $p$,
- $\exists p$ — być może kiedyś nastąpi $p$,
- $\forall p$ — kiedyś na pewno będzie $p$,
- $\forall p \land p$ — zaczynając od dowolnego stanu osiągalnego możemy kiedyś osiągnąć $p$,
- $\forall p \land p$ — zachodzi nieskończoną ilość razy,
- $\forall(p \rightarrow \exists q)$ — jeśli znajdziemy się w stanie spełniającym $p$, to na pewno kiedyś dojdziemy do stanu spełniającego $q$.

11.5.5 Zastosowania

Dana jest struktura Kripkego $K$ oraz formuła stanowa $p$. Sprawdzić, czy $K \models p$.

Zadanie to można sprowadzić do wielokrotnego przeszukiwania grafu, który reprezentuje proces. Tak więc sprawdzenie, czy dana struktura Kripkego jest modelem dla zadanej formuły czy też nie, jest problemem rozstrzygальным. Można to wykorzystać do weryfikacji programów współbieżnych w następujący sposób:

- zidentyfikuj kluczowe stany procesów składające się na program współbieżny,
- utwórz strukturę Kripkego modelującą przeplot ciągów wykonawczych tych procesów,
11.6. Zastosowanie LTL w praktyce

- wyrażać własności żywotności i bezpieczeństwa w postaci formuł logiki CTL,
- sprawdzić, czy utworzona struktura Kripkego jest modelem dla powyższych formuł.

Narzędziem, za pomocą którego można przeprowadzić to zadanie, jest np. SMV. Jako praktyczny przykład realizacji tego zadania wykorzystany zostanie kod w języku SMV, mający na celu weryfikację poprawności algorytmu Peterson’a dla dwóch procesów. Jest to algorytm stosowany w programowaniu współbieżnym, mający na celu realizację sekcji krytycznych. Pozwala on na dostęp dwóch procesów do niepodzielnych zasobów, nie wywołując przy tym konfliktu mimo, że używają do komunikacji pamięci współdzielonej. Algorytm ten w pseudokodzie przedstawia się następująco: Kolejny listing to kod w SMV, który realizuje rozwiązanie omawianego problemu.

**11.6. Zastosowanie LTL w praktyce**

W tym podrozdziale zostanie zaprezentowany sposób weryfikacji różnych systemów przy pomocy programu Spin oraz LTL. Program Spin jest to narzędzie sprawdzające poprawność modeli pod względem zdefiniowanych przez użytkownika reguł. Reguły są tworzone z wykorzystaniem LTL. W celu sprawdzenia poprawności, program w każdym ze stanów porównuje czy stworzona reguła jest prawdą, jeśli tak to oznacza błąd modelu. Sposób sprawdzania zmusza użytkownika do tworzenia reguł, które nigdy nie powinny zajść w modelu.

**11.6.1. Opis narzędzia**


**PROMELA**


- `int x` - deklaracja jednej zmiennej,

Zmiana wartości zmiennych może odbywać się tylko w procesie. Procesy deklaruje się w następujący sposób:
11. Logika temporalna

proctype A()
{
    byte state;
    state = 3;
}

Taka deklaracja nie powoduje, że dany proces będzie uruchomiony. Do uruchomienia wszystkich procesów służy proces `init`. W nim przy użyciu komendy `run` powodujemy wykonywanie się danego procesu. Procesy komunikują się pomiędzy sobą przy pomocy zmiennych globalnych bądź też za pomocą kanałów. Kanał deklaruje się w następujący sposób:

can kanal=[10] of {int}

Wysyłanie i odczytywanie informacji z kanału odbywa się przy użyciu znaków `!` oraz `?`.

- kanal `! wiadomosc` - powoduje wyslanie wartości zmiennej `wiadomosc` do kanału, o ile nie jest zapelniony,
- kanal `? wiadomosc` - powoduje odczytanie wiadomości z kanału, o ile kanał nie jest pusty.


**Sposób użycia SPIN'a**

Chcąc użyć programu Spin do weryfikacji danego systemu, trzeba jego model zapisać w języku `PROMELA`. Mając dany model zapisany w pliku `*.pml` można stworzyć reguły w logice `LTL` sprawdzające poprawność systemu. Oczywiście ta reguła musi być zrozumiała przez program, jednak tutaj można wykorzystać bezpośrednio program `spin` wywołany z opcją `-f`. Przykładem zapisania do pliku zrozumiałej dla programu formuły może być wywołanie:

```
spin -f '!([](!(P && L))' > logika.ltl
```

Reguła `[](!(P \&\& L))` może oznaczać, żeby zawsze było spełnione, aby silnik nie obracał się w prawo i w lewo jednocześnie. W celu sprawdzenia tego warunku do programu musi zostać wprowadzona zaprzeczona reguła, ponieważ aplikacja weryfikuje dany model sprawdzając czy kiedykolwiek dane wyrażenie będzie prawdą. Operatory modalne dostępne w programie to:

- `[]` - zawsze,
- `<>` - kiedyś,
- `X` - następny.

Kolejnym krokiem jest zdefiniowanie zmiennych `P` oraz `L`, tak aby odpowiadały rzeczywistym parametrom modelu. W tym celu trzeba otworzyć plik `logika.ltl`
11.6. Zastosowanie LTL w praktyce

i na jego początku, wykorzystując makrodefinicje podobnie jak w języku C, zde-
finiować zmienne. Może to wyglądać następująco:

#define P silnikObracaSieWPrawo
#define L silnikObracaSieWLewo

Następnym krokiem jest utworzenie weryfikatora sprawdzającego warunek zapi-
sany w LTL. W tym celu wywołuje się program z opcjami, jak poniżej:

spin -a -N logika.ltl plik_modelu.pml

Powstały pliki pan.* z różnymi rozszerzeniami, pan.c zawiera główny kod we-
ryfikatora, a pan.h jest typowym plikiem nagłówkowym, natomiast pliki pan.b, pan.m oraz pan.t zawierają informacje potrzebne do wymiany informacji pomię-
dzy kolejnymi stanami. Powyższe pliki trzeba skompilować, np. poleceniem:

gcc -o pan pan.c

Wywołanie programu pan (weryfikatora) powoduje wyświetlenie wielu informa-
cji. Komunikaty mówią o tym czy weryfikacja ukończyła się sukcesem oraz które
stany w danych procesach nigdy nie zostały osiągnięte, co pozwala na optymaliz-
ację modelu. Program spin tworzy kod w języku C, ponieważ zwiększa to szyb-
kość sprawdzania wszystkich stanów i tylko w ten sposób może w miarę sensown-
ym czasie zweryfikować dany system.

11.6.2. Weryfikacja systemu stworzonego na sterownikach PLC

Weryfikacja systemu pokazanego na rys. 11.1 została dokonana przez utwo-
rzenie odpowiednich reguł w LTL. Sam model w języku PROMELA dostępny jest
na stronie urlhttp://www.albertolluch.com/research/promelamodels, na której
znajduje się również wiele innych gotowych przykładów. Przestrzony proces
ma za zadanie produkcję soli o zmniejszonym stężeńiu. Podstawą produkcji jest
stężona sól znajdujaca się w zbiorniku B1 oraz woda w zbiorniku B2. W zbiorniku
B3 jest mieszana woda z solą. Gotowa mieszanka jest transportowana do zbiorni-
ika B4 a dalej do B5. W zbiorniku B5 mieszanka jest podgrzewana, a powstająca
para transportowana do zbiornika B6. W zbiorniku B6 para jest schładzana, po-
wstaje woda, która następnie jest wpompowywana z powrotem do zbiornika B2.
W zbiorniku B5 pozostaje podgrzana sól, która jest transportowana do zbiornika
B7, tam schładzana i dalej wpompowywana z powrotem do zbiornika B1.

Zmienne zdefiniowane w modelu to:
• empty - oznacza, że dany zbiornik jest pusty,
• sol42C - sól schłodzona,
• sol42H - sól podgrzana,
• sol70C - zimna mieszanina wody z solą,
• sol70H - gorąca mieszanina wody z solą,
• water28C - zimna woda,
• water28H - gorąca woda (para).
11. Logika temporalna

Przykłady

Pierwszym celem postawionym przed systemem przedstawiony w 11.6.2 jest nieprzerwanne działanie. Można to zapisać przy użyciu formuły:

\[ []<>B^3=\text{sol70C} \land []<>B^3=\text{empty} \]
11.6. Zastosowanie LTL w praktyce

Powyższa reguła mówi, że zbiornik B3 zawsze kiedyś zostanie wypełniony mieszanką soli oraz że zawsze kiedyś zostanie opróżniony. Czyli działania są naprzemiennie powtarzane, co zapewnia ciąłą pracę systemu. Po przeprowadzeniu weryfikacji pod tym kątem otrzymuje się poniższe informacje:

(Spin Version 5.2.4 -- 2 December 2009)
+ Partial Order Reduction

Full statespace search for:
  never claim +
  assertion violations + (if within scope of claim)
  acceptance cycles - (not selected)
  invalid end states - (disabled by never claim)

State-vector 132 byte, depth reached 9999, errors: 0
  203248 states, stored
  394335 states, matched
  597583 transitions (= stored+matched)
  3904292 atomic steps
hash conflicts: 36467 (resolved)

Stats on memory usage (in Megabytes):
  28.687 equivalent memory usage for states
    (stored*(State-vector + overhead))
  23.481 actual memory usage for states (compression: 81.85%)
    state-vector as stored = 105 byte + 16 byte overhead
  2.000 memory used for hash table (-w19)
  0.305 memory used for DFS stack (-m10000)
  25.743 total actual memory usage
unreached in proctype B1toB3
  line 167, state 13, "assert(0)"
  line 176, state 32, "assert(0)"
  line 181, state 41, "-end-"
  (3 of 41 states)
unreached in proctype B2toB3
  line 190, state 13, "assert(0)"
  line 199, state 32, "assert(0)"
  line 204, state 41, "-end-"
  (3 of 41 states)

...
11. Logika temporalna

pan: elapsed time 13.5 seconds
pan: rate 15088.938 states/second

Wynika z tego, że system będzie działał bez przerwy. Podobnie można sprawdzić, czy w zbiornikach B1 oraz B2 mogłyby się znaleźć odpowiednio gorąca sól oraz gorąca woda. Reguły logiki temporalnej wyglądają następująco:

\[
[]!(B1==sol42H) \\
[]!(B2==water28H)
\]

Oczywiście trzeba pamiętać, że przy wprowadzaniu do programu SPIN należy wszystko zaprzeczyć oraz zamiast \(B1==sol42H\) wstawić zmienną logiczną, którą potem definiuje się w pliku *.ltl.

W celu pokazania przypadku negatywnego, założono, że nieporządane w systemie jest znalezienie się w zbiorniku B7 soli schłodzonej. Regułę można zapisać:

\[
[]!(B7==sol42C)
\]

Wynikiem działania programu jest:

warning: for p.o. reduction to be valid the never claim must be stutter-invariant (never claims generated from LTL formulae are stutter-invariant)

pan: claim violated! (at depth 5482)
pan: wrote plc.prm.trail

(Spin Version 5.2.4 -- 2 December 2009)
Warning: Search not completed
+ Partial Order Reduction

Full statespace search for:
never claim +
assertion violations + (if within scope of claim)
acceptance cycles - (not selected)
invalid end states - (disabled by never claim)

State-vector 132 byte, depth reached 5482, errors: 1
118 states, stored
20 states, matched
138 transitions (= stored+matched)
9847 atomic steps
hash conflicts: 0 (resolved)

2.501 memory usage (Mbyte)
11.7. Podsumowanie

W tym przypadku widać, że program zwrócił błąd co jest oczywiste, ponieważ sól w zbiorniku B7 jest schładzana, i dopiero jak będzie uznana za zimną jest transportowana dalej.

11.7. Podsumowanie


Program SPIN jest bardzo dobrym narzędziem, świadczy o tym chociażby, że jest używany w NASA. Przy jego pomocy można przeprowadzić weryfikację różnych modeli. Jedyny problem to zapisanie modelu w języku zrozumiałym przez program. W celu wprawnego posługiwania się tym narzędziem należy opanować teorię związaną z logiką temporalną oraz zapoznać się z dokumentacją i specyfikacją języka PROMELA. Kolejną zaletą tego programu jest to, że jest darmowy oraz dostępny pod różne platformy, m.in. pod Linux’a oraz Windows’a.

Literatura

12.1. Logika

Słowo „logika” pochodzi od greckiego słowa *logikos*, oznaczającego naukę o rozumowaniu. Nauka ta nie rozpatruje treści osądu czy otrzymanych wniosków, ale metody, które wykorzystuje się do otrzymywania wyników. Logika formalna interesuje się zasadami i metodami wnioskowania. Logika używana jest w większości intelektualnych czynności, ale głównie jest to dziedzina filozofii, matematyki i informatyki.

Mówiąc o logice ma się na myśli pewien rodzaj krytycznego myślenia. W filozofii dziedzina ta jest częścią epistemologii (teorii poznania), której głównym zagadnieniem jest: *skąd wiemy to, co wiemy*. W matematyce logika to nauka o wnioskowaniu, wykorzystująca język formalny. Jako dziedzina nauki, logika pochodzi z czasów Arystotelesa, który ustanowił logikę jako podstawę filozofii. Logika jest częścią klasycznego trivium

12.1.1. Krótko historia

Arystoteles jest nazywany ojcem logiki. Był on pierwszym filozofem, który badał tę dziedzinę naukowo. W swoim dziele „Organon” analizuje wiedzę oraz klasyfikuje rodzaje wnioskowania. Organon oznacza narzędzie. Tak Arystoteles określił logikę, która jest właśnie narzędziem i nie stanowi nauki.

Arystoteles wymyślił również syllogizm. Jest to rodzaj logicznej debaty, w której jedno twierdzenie (zwanе wnioskiem) jest konkluzją z dwóch innych przesłanek. Syllogizm jest rozumowaniem dedukcyjnym (nieco więcej informacji na temat syllogizmu zamieszczono w następnym podrozdziale [12.1.2]).

Uczeń Sokratesa, Euklejdes (Euklides) z Megary, zapoczątkował logikę megarejską, zwana dialektyką, która służyła sztuce prowadzenia sporów i rozmów.

---

1 Trivium – jedna z podgrup Siedmiu Sztuk Wyzwolonych (Siedmiu umiejętności godnych człowieka wolnego), zawiera gramatykę retorykę i logikę.
12.1. Logika

spekulatywnych (dialektyka - tutaj sztuka dyskutowania, zwłaszcza umiejętność dochodzenia do prawdy przez ujawnianie i przewyższanie sprzeczności w rozumowaniu przeciwnika [1]). Z twórczości Euklejdesa korzystali także Płat, Antystenes i Arystyp. Megarejczycy dociekieli trudności w prowadzeniu dyskusji, poszukiwali problemów, subtelnosci i paradoksów. Zenon z Kition i Chryzyp pracowali później nad stworzeniem ogólnej teorii implikacji.

Na podstawie badań megarejczyków i stoików powstał tzw. rachunek zdań stoików, obejmujący obszerny fragment współczesnego rachunku zdań. Zawiera on główne twierdzenia, dotyczące implikacji. Logika zdań jest bardziej abstrakcyjna od syllogistyki.

W 1607 roku sir Francis Bacon przedstawił formalną dyskusję na temat de dukcji indukcyjnej. Ta metoda była wcześniej przyjmuona przez logikę dedukcyjną, przedstawioną przez Arystotelesa. Wiele lat później, w roku 1854 George Boole opublikował książkę, w której przedstawił logikę symboliczną oraz podstawy logiki, którą teraz określa się mianem logiki Boola (bądź logikę boolowską). Dwadzieścia pięć lat później Frege rozpoczął nowoczesną logikę, wprowadzając kwantyfikatory.

12.1.2. Metody wnioskowania

Rozumowanie dedukcyjne (wcześniej wspomniany syllogizm) składa się z trzech części: głównego założenia, założenia drugorzędnego oraz wniosku. Na przykład:

1. Francja jest częścią Europy. (założenie główne)
2. Miasto Paryż jest częścią Francji. (założenie drugorzędne)
3. Paryż jest częścią Europy. (wniosek)


Sir F. Bacon powiedział, że naukowe wnioski mogą być osiągnięte tylko poprzez analizę, eksperymenty i dochodzenie, a całe poprawne wnioskowanie musi być przeprowadzone drogą indukcji. Rozumowania indukcyjne bywają uważane za główne narzędzie tzw. nauk empirycznych, przeciwwstawianych z tego powodu tzw. naukom dedukcyjnym (głównie matematyka i logika), posługujących się rozumowaniami dedukcyjnymi. Metoda stosowana przez nauki empiryczne, polega na zastosowaniu eksperymentu i obserwacji.

W rozumowaniu indukcyjnym wyróżniamy indukcję niezupelną, która polega na poznaniu jakiejs ogólnej prawidłowości na podstawie skończonej liczby zdań stwierdzających niektóre wystąpienia tej prawidłowości. Jest to jedno z podstawowych narzędzi nauk doświadczalnych, jednak jego stosowanie wymaga odpowiedniej metodologii. Kolejnym typem jest indukcja zupełna. Jest to wnioskowanie, w którym jakąś ogólną prawidłowość stwierdza się na podstawie sprawdzania wszystkich możliwych przypadków. Przykładem rozumowania przez in-
dukcję zupełną może być stwierdzenie (przez przeczytanie listy obecności), że obecny jest każdy uczeń. W praktyce naukowej zastosowania indukcji zupełnej są bardzo ograniczone. Istnieje bowiem wiele sytuacji, w których liczba możliwych wystąpień danego zdarzenia jest niezmiernie duża lub wręcz nieskończona.

Kolejnym typem jest indukcja eliminacyjna F. Bacona. Sprowadza się ona do sformułowania wyczerpującej listy hipotez na dany temat, które wzajemnie się wykluczają; a następnie dokonanie eliminacji z użyciem narzędzia, którym jest eksperyment. Zakłada się, że jeśli lista hipotez jest wyczerpująca, to musi się wśród nich znajdować także hipoteza prawdziwa. F. Bacon sformułował zasadę ograniczonej różnorodności świata, która zakłada, że dany temat można sformułować wyczerpujaco i przedstawić jako skończoną listę.


12.1.3. Logika w matematyce

W matematyce logika koncentruje się na analizowaniu zasad rozumowania oraz pojęć, związanych z wykorzystaniem sformalizowanych i uściślenionych metod oraz narzędzi matematyki. Logika matematyczna to dział matematyki wywodzący się z tzw. logiki tradycyjnej, wyodrębniony przez zastosowanie języka i metod matematycznych [3]. Matematyka wykorzystuje logikę dedukcyjną oraz indukcyjną. Kiedy matematyk poszukuje nowej koncepcji, najpierw poprzez wnioskowanie indukcyjne ustala przypuszczenie. Następnie próbuje to przypuszczenie udowodnić i zamienić w twierdzenie, używając dowodu opartego na innych twierdzeniach i aksjomatach. Teza staje się twierdzeniem przy wykorzystaniu rozumowania dedukcyjnego.

12.1.4. Podstawy logiki matematycznej

Zdanie - dowolne stwierdzenie, o którym można powiedzieć, że jest albo prawdziwe, albo fałszywe, i które nie może być jednocześnie i prawdziwe, i fałszywe. Na przykład „stół jest czerwony” jest zdaniem w sensie logiki. Sformułowanie „czy logika jest trudna?” zdaniem nie jest. Zdaniem przypisywana jest wartość logiczna prawda lub fałsz.

Formuły logiczne (formuły rachunku zdań) tworzy się przez łączenie zdań używając spółników: i (∧), lub (∨), jeżeli (⇒), wtedy i tylko wtedy (⇔), nie (¬). Wartość logiczna formuł zależy od wartości logicznej zdań.

Chcąc napisać program używając języka logicznego można opisać fragment rzeczywistości (bądź hipotetyczną rzeczywistość) i rozważyć czy dane fakty są spełnialne. Sprawdzanie czy formuła jest twierdzeniem danej teorii (to znaczy czy konkretny fakt spełniony jest w danej rzeczywistości), nazywany jest proble-
mem decyzyjnym w logice. Procedura umożliwiająca rozstrzygnięcie tego problemu to procedura decyzyjna. Aby zajmować się problemami decyzyjnymi należy zdefiniować pojęcia konsekwencji logicznej oraz teorii \[4\].

Niech \( U \) będzie zbiorem formuł, a \( \varphi \) dowolną formułą. Jeśli każdy model \( U \) jest jednocześnie modelem \( \varphi \) to mówimy, że \( \varphi \) jest konsekwencją logiczną \( U \), co zapisujemy \( U \models \varphi \). Mówimy się również czasami, że \( \varphi \) wynika logicznie z \( U \), który bywa nazywany zbiorem przesłanek.

Rozważmy na przykład formułę \( \varphi = p \lor \neg q \). Formuła \( \varphi \) jest konsekwencją logiczną zarówno zbioru \( U_1 = \{p\} \), czyli \( U_1 \models \varphi \), jak i \( U_2 = \neg q \), czyli \( U_2 \models \varphi \).

Jednocześnie \( \varphi \) nie jest konsekwencją logiczną zbioru \( U_3 = \neg p, q \), a więc \( U_3 \not\models \varphi \).

Własności konsekwencji logicznych:

1. Jeśli dla pewnego \( U = \{\varphi_1, \varphi_2, \ldots, \varphi_n\} \) i \( \psi \) zachodzi \( U \models \psi \) to formuła \( \varphi_1 \land \varphi_2 \land \cdots \land \varphi_n \Rightarrow \psi \) jest tautologią, inaczej: \( \models \varphi_1 \land \varphi_2 \land \cdots \land \varphi_n \Rightarrow \psi \) (tautologia – wyrażenie, które jest prawdziwe na mocy swojej formy/budowy). Nie możemy jednak powiedzieć, że tautologią jest \( U \models \psi \) ponieważ nie jest to formuła logiczna, a \( \models \) nie jest spójnikiem logicznym.
2. Jeśli \( U \models \varphi \) to dla dowolnej formuły \( \psi \) zachodzi: \( U \cup \psi \models \varphi \).
3. Jeśli \( U \models \varphi \) i \( \psi \) jest formułą prawdziwą, to zachodzi: \( U \setminus \{\psi\} \models \varphi \).

Zbiór formuł \( T \) jest nazywany teorią jeśli jest on zamknięty na konsekwencje (albo wynikanie) logiczne. Zbiór formuł \( T \) jest zamknięty na konsekwencje logiczne wtedy i tylko wtedy, gdy dla wszystkich formuł \( \varphi \) zachodzi zależność: jeśli \( T \models \varphi \), to \( \varphi \in T \). Elementy teorii \( T \) nazywane są twierdzeniami tej teorii. Teorią \( T(U) \) zbioru formuł \( U \) nazywa się zbiór \( T(U) = \{\varphi \mid U \models \varphi\} \).

Fakt: Dla dowolnego zbioru formuł \( U \) teoria tego zbioru \( T(U) \) jest teorią.

Twierdzenia matematyczne na ogół mają postać implikacji. Aby dowieść twierdzenie dany zbiór zdań, zwanych założeniami, \( \varphi_1, \varphi_2, \ldots, \varphi_n \), uznaje się za prawdziwe i dodaje się do nich nowe zdanie \( \psi \). \( \psi \) jest wnioskiem lub konkluzją, która wynika z \( \varphi_1, \ldots \) zgodnie z prawami logiki.

Dla formalnego wprowadzenia dowodów logicznych używa się reguł wnioskowania, postaci:

\[
\frac{\varphi_1, \varphi_2, \ldots, \varphi_n}{\psi}
\]

Znaczenie reguły wnioskowania jest takie, że jeśli wiemy, że prawdziwe są formuły \( \varphi_1, \varphi_2, \ldots, \varphi_n \) to możemy również uznać za prawdziwą formułę \( \psi \) [4].

Rozważmy teraz pojęcie dowodu. Załóżmy, że dany jest pewien zbiór formuł \( \Delta \) zwany zborem założeń, oraz formuła \( \psi \). Dowodem formuły \( \psi \) jest ciąg \( \varphi_1, \varphi_2, \ldots, \varphi_n, \psi \), składający się z formuł, z których każda spełnia jeden z warunków:

1. jest jedną z przesłanek,
2. jest tautologią,
3. została otrzymana w wyniku użycia jednej z reguł wnioskowania zastosowanej do formuł leżących na lewo od niej w dowodzie.
12. Programowanie deklaratywne i logika obliczeniowa

Regułę wnioskowania:

\[
\frac{\varphi_1, \varphi_2, \ldots, \varphi_n}{\psi}
\]

nazywa się poprawną wtedy i tylko wtedy, gdy poniższa formula jest tautologią.

\[
\varphi_1 \land \varphi_2 \land \cdots \land \varphi_n \Rightarrow \psi
\]

Sprawdzenie, czy wyrażenie rachunku zdań jest tautologią można przeprowadzić, wykorzystując postać normalną tego wyrażenia. W logice matematycznej istnieją dwie postaci normalne formuł: dysjunkcyjną postać normalną i koniunkcyjną postać normalną. Dla postaci koniunkcyjnej istnieją algorytmy wielomia- nowe sprawdzające czy formula jest tautologią. Formuła ma dysjunkcyjną postać normalną (DNF, ang. Disjunctive Normal Form), jeśli jest postaci:

\[
\bigvee_{i=1}^{n} \left( \bigwedge_{j=1}^{m} l_{ij} \right)
\]

gdzie \( l_{ij} \) są literałami dla \( i = 1, \ldots, n \) oraz \( j = 1, \ldots, m_i \). Inaczej mówiąc dysjunkcyjna postać normalna, to alternatywa koniunkcji literałów.

Każdą zmienną zdaniową i negację zmiennej zdaniowej nazwijmy literałem. Dla każdej formuły rachunku zdań istnieje równoważna jej formuła w postaci DNF. Można ją otrzymać przekształcając formułę zgodnie z prawami de Morgana i rozdzielności. Istnieje ponadto prosty schemat wyznaczania tej postaci na podstawie tabelki prawdy dowolnej formuły.

<table>
<thead>
<tr>
<th>( q )</th>
<th>( p )</th>
<th>( r )</th>
<th>( \ldots )</th>
<th>( \phi )</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
<td>( \ldots )</td>
<td>0</td>
</tr>
<tr>
<td>( \ldots )</td>
<td>( \ldots )</td>
<td>( \ldots )</td>
<td>( \ldots )</td>
<td>( \ldots )</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>1</td>
<td>( \ldots )</td>
<td>1</td>
</tr>
<tr>
<td>( \ldots )</td>
<td>( \ldots )</td>
<td>( \ldots )</td>
<td>( \ldots )</td>
<td>( \ldots )</td>
</tr>
<tr>
<td>( \ldots )</td>
<td>( \ldots )</td>
<td>( \ldots )</td>
<td>( \ldots )</td>
<td>( \ldots )</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>0</td>
<td>( \ldots )</td>
<td>1</td>
</tr>
</tbody>
</table>

Dla każdego wiersza tabeli, który ma jedynkę w kolumnie formuły \( \phi \) konstruuje się koniunkcję literałów odpowiadających wszystkim zmiennym zdaniowym formuły, z negacją tylko przy zmiennych, dla których w danym wierszu występuje zero:

\[
\phi \Leftrightarrow (\neg p \land \neg q \land r \land \ldots) \lor (\neg p \land q \land \neg r \land \ldots)
\]

Formuła wygenerowana zgodnie z tym schematem rzadko jest jednak optymalna, tzn. na ogólnie istnieje krótsza postać DNF równoważna danej formule.

Przykład zastosowania postaci DNF: formuła \( \phi = (p \land \neg q) \lor (\neg p \land q) \) jest w postaci DNF Równoważny, że jest ona równoważna formule \( p \oplus q \):
12.1. Logika

<table>
<thead>
<tr>
<th></th>
<th></th>
<th>≠</th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
</tbody>
</table>

Mówimy, że formuła zdaniowa jest w koniunkcyjnej postaci normalnej (ang. Con-junctive Normal Form, CNF), gdy jest koniunkcją alternatyw literałów. Co można zapisać:

\[ \bigwedge_{i=1}^{n} \left( \bigvee_{j=1}^{m_i} l_{ij} \right) \]

gdzie \( l_{ij} \) są literałami dla \( i = 1, \ldots, n \) oraz \( j = 1, \ldots, m_i \).

W programowaniu często logicznym wykorzystuje się formuły logiczne w postaci klauzul (język Prolog wykorzystuje klauzule Horna). Klauzula to alternatywa literałów. Zapisuje się ją w następujący sposób

\[ \bigvee_{j=1}^{m} l_j, \]

gdzie \( l_j \) to literały oraz \( j = 1, 2, \ldots, m \). Pusta klauzula jest nieprawdziwa. Klauzula Horna to klauzula, w której najwyżej jeden element jest niezanegowany.

12.1.5. Logika pierwszego rzędu

Język logiki pierwszego rzędu (logika pierwszego rzędu nazywana jest też rachunkiem predykatów lub rachunkiem kwantyfikatorów) można traktować jak rozszerzenie rachunku zdaniowego, pozwalające formułować stwierdzenia o zależnościach pomiędzy obiektami indywiduowymi (np. relacjach i funkcjach). Dzięki zastosowaniu kwantyfikatorów, odwołujących się do całej zbiorowości rozważanych obiektów, można w logice pierwszego rzędu wyrażać własności struktur relacyjnych oraz modelować rozumowania dotyczące takich struktur. Do zestawu symboli rachunku zdaniowego dodajemy następujące nowe składniki syntaktyczne:

• symbole operacji i relacji (w tym symbol równości \( = \));
• zmienne indywiduowe, których wartości mogą przebiegać rozważane dziedziny;
• kwantyfikatory, wciągające zmienne indywiduowe w formułach.

Symbole operacji i relacji są podstawowymi składnikami do budowy najprostszych formuł, tzw. formuł atomowych. Z tego względu w języku pierwszego rzędu rezygnuje się ze zmiennych zdaniowych [5]. Język logiki pierwszego rzędu, dzięki możliwości używania kwantyfikatorów, jest dość elastyczny. Można z jego pomocą wyrażać wiele nietrywialnych własności obiektów matematycznych. W szczególności interesować nas może definiowanie elementów i wyodrębnianie struktur o pewnych szczególnych w własnościach, czy też formułowanie kryteriów odróżniających jakieś struktury od innych.
12. Programowanie deklaratywne i logika obliczeniowa

Symbole operacji i relacji są podstawowymi składnikami do budowy najprostszych formuł, tzw. formuł atomowych. Z tego względu w języku pierwszego rzędu rezygnuje się ze zmiennych zdaniowych. Do podstawowych pojęć wykorzystywanych w rachunku predykatów należą sygnatura $\Sigma$, zmienneindywidualowe, zmienne atomowe.

Przez sygnaturę $\Sigma$ oznaczać będziemy rodzinę zbiorów $\Sigma_n^F$, dla $n \geq 0$ oraz rodzinę zbiorów $\Sigma_n^R$, dla $n \geq 1$. Elementy $\Sigma_n^F$ nazywamy symbolami operacji n-argumentowych, $\Sigma_n^R$ nazywamy symbolami relacji n-argumentowych. Przyjmujemy, że wszystkie te zbiory są parami rozłączne. W praktyce, sygnatura zwykle jest skończona i zapisuje się jako ciąg symboli. Np. ciąg złożony ze znaków $+\cdot 0\ 1$ (o znanej liczbie argumentów) tworzy sygnaturę języka teorii ciał.

Zbiór termów $\mathcal{T}(\Sigma)(X)$ nad sygnaturą $\Sigma$ i zbiorem zmiennych $X$ definiujemy inductyjnie. Zmienne indywidualne są termami. Dla każdego $n \geq 0$ i każdego symbolu operacji $f \in \Sigma_n^F$, jeśli $t_1, \ldots, t_n$ są termami, to $f(t_1, \ldots, t_n)$ jest też termem. $X$ jest nieskończonym przeliczalnym zbiorem symboli, nazywanymi zmiennymi indywidualnymi.

Dla każdego termu $T(\Sigma)(X)$ definiujemy zbiór $\mathcal{FV}(t)$ zmiennej występującej w $t$. $\mathcal{FV}(x) = x$ oraz $\mathcal{FV}(f(t_1, \ldots, t_n)) = \bigcup_{i=1}^n \mathcal{FV}(t_i)$. Formuły atomowe języka pierwszego rzędu definiujemy następująco.

Symbol fałszy $\bot$ jest formułą atomową. Dla każdego $n \geq 1$, każdego symbolu relacji $n$-argumentowej, oraz dla dowolnych termów $t_1, \ldots, t_n \in \mathcal{T}(\Sigma)(X)$, napis $r(t_1, \ldots, t_n)$ jest formułą atomową. Dla dowolnych termów $t_1, t_2$, napis $(t_1 = t_2)$ jest formułą atomową.

Formuły nad sygnaturą $\Sigma$ i zbiorem zmiennych indywidualnych $X$ definiujemy następująco. Każda formuła atomowa jest formułą drugiego rzędu. Jeśli $\phi, \psi$ są formułami drugiego rzędu nad sygnaturą $\Sigma$, to $\phi \lor \psi$ jest formułą drugiego rzędu nad sygnaturą $\Sigma$. Jeśli $\phi$ jest formułą drugiego rzędu nad sygnaturą $\Sigma$ i $x \in X$ jest zmienną indywidualną, to $\forall x \phi$ jest formułą drugiego rzędu nad sygnaturą $\Sigma$.

Jeśli $\phi$ jest formułą drugiego rzędu nad sygnaturą $\Sigma$, a $R$ jest symbolem relacji $k$-argumentowej z $\Sigma$, to $\exists R \phi$ jest formułą drugiego rzędu nad sygnaturą $\Sigma – \{R\}$.

Jedno z twierdzeń mówi, że logika drugiego rzędu nie ma żadnego pełnego i poprawnego systemu dowodowego.

12.1.6. Logika drugiego rzędu

Składnią logiki drugiego uzupełnia się przez rozszerzenie zbioru reguł składniowych dla logiki pierwszego rzędu o kwantyfikatory wiąjące symbole relacyjne. Definicja formuł drugiego rzędu jest indukcyjna, podobnie jak analogiczna definicja formuł drugiego rzędu. Jednak tym razem nie ustalamy sygnatury z góry.

Kaźda formuła atomowa nad sygnaturą $\Sigma$ jest formułą drugiego rzędu nad sygnaturą $\Sigma$. Jeśli $\phi, \psi$ są formułami drugiego rzędu nad sygnaturą $\Sigma$, to $\phi \lor \psi$ też jest formułą drugiego rzędu nad sygnaturą $\Sigma$.

Jeśli $\phi$ jest formułą drugiego rzędu nad sygnaturą $\Sigma$, to $\neg \phi$ też jest formułą drugiego rzędu nad sygnaturą $\Sigma$.

Jeśli $\phi$ jest formułą drugiego rzędu nad sygnaturą $\Sigma$, a $x \in X$ jest zmienną indywidualną, to $\exists x \phi$ jest też formułą drugiego rzędu nad sygnaturą $\Sigma$.

Jeśli $\phi$ jest formułą drugiego rzędu nad sygnaturą $\Sigma$ i $R$ jest symbolem relacji $k$-argumentowej z $\Sigma$, to $\exists R \phi$ jest formułą drugiego rzędu nad sygnaturą $\Sigma – \{R\}$.

Jedno z twierdzeń mówi, że logika drugiego rzędu nie ma żadnego pełnego i poprawnego systemu dowodowego.
12.2. Algorytmy wnioskowania

12.1.7. Logika obliczeniowa

Logika obliczeniowa jest działem matematyki zajmującym się teoretycznymi podstawami automatycznego wnioskowania. Ma takie same założenia, co logika matematyczna czyli: dokładność składni i semantyki, poprawność i kompletność rozumowania. Logika obliczeniowa uwzględnia też efektywność algorytmów oraz kładzie nacisk na automatyzację.

12.2. Algorytmy wnioskowania

12.2.1. Metoda tabel semantycznych

Metoda tabel semantycznych jest algorytmem do badania poprawności formuł rachunku zdań. Tabela $T$ dla formuły $A$ jest drzewem, którego każdy wierzchołek $n$ zawiera zbiór formuł $U(n)$. Początkowa $T$ składa się z pojedynczego wierzchołka (korzenia) zawierającego zbiór jedno elementowy {$A$}. Tworzenie tabeli semantycznej przebiega iteracyjnie przez wybór nieoznakowanego liścia $n$, zawierającego $U(n)$ i wykonanie jednego z następujących kroków algorytmu.

- Jeżeli $U(n)$ nie jest zbiorem literałów, to wybierz dowolna formułę $A$ z tego zbioru, niebędąca literałem. Jeśli $A$ jest typu $\alpha$, to utwórz nowy wierzchołek $n'$, będący potomkiem wierzchołka $n$ i umieść w nim $U(n') = ((U(n) - \{A\}) \cup \{\alpha_1, \alpha_2\})$.

- Jeżeli $A$ jest typu $\beta$, utwórz dwa nowe wierzchołki $n'$ oraz $n''$ jako następcy wierzchołka $n$. W wierzchołku $n'$ umieść $U(n) = (U(n) - \{A\}) \cup \beta_1$, a w wierzchołku $n''$ umieść $U(n'') = (U(n) - \{A\}) \cup \beta_2$.

- Jeżeli $U(n)$ (zbiór formuł w wierzchołku $n$) jest zbiorem literałów, to sprawdź, czy zawiera on parę literałów komplementarnych. Jeżeli tak, to oznakuj go jako domknięty, jeżeli nie, to oznakuj go jako otwarty.

Tabele semantyczne, której tworzenie zakończono nazywamy zakończoną. Tabele zakończona nazywamy domknięta, jeśli wszystkie liście są oznakowane jako domknięte. Jeżeli istnieje liść otwarty, to tabele nazywamy otwarta.

Formuła $A$ jest niespełniana wtedy, gdy zakończona tabela $T$ dla formuły $A$ jest domknięta. Formuła $A$ jest spełnialna wtedy, gdy $T$ jest otwarta. Formuła $A$ jest prawdziwa wtedy, gdy tabela semantyczna dla formuły $\neg A$ jest domknięta [6].

Konstrukcja tabel semantycznych odbywa się według reguł zapisanych w tab. [12.1]. Symbol $\uparrow$ oznacza negację koniunkcji (NAND). Znak $\downarrow$ oznacza negację alternatywy (NOR).

Problemy z tabelami semantycznymi:
12. Programowanie deklaratywne i logika obliczeniowa

Tab. 12.1: Reguły budowy tabel semantycznych

<table>
<thead>
<tr>
<th>α</th>
<th>α₁</th>
<th>α₂</th>
<th>β</th>
<th>β₁</th>
<th>β₂</th>
</tr>
</thead>
<tbody>
<tr>
<td>¬¬A₁</td>
<td>A₁</td>
<td></td>
<td>¬B₁ ∧ B₂</td>
<td>B₁</td>
<td></td>
</tr>
<tr>
<td>A₁ ∧ A₂</td>
<td>A₁</td>
<td>A₂</td>
<td>B₁ ⊕ B₂</td>
<td></td>
<td></td>
</tr>
<tr>
<td>¬(A₁ ∨ A₂)</td>
<td>A₁</td>
<td>¬A₂</td>
<td>B₁ ⇒ B₂</td>
<td>¬B₁</td>
<td>B₂</td>
</tr>
<tr>
<td>¬(A₁ ⇒ A₂)</td>
<td>A₁</td>
<td>¬A₂</td>
<td></td>
<td>¬B₁</td>
<td>B₂</td>
</tr>
<tr>
<td>A₁ ⇔ A₂</td>
<td>A₁</td>
<td>A₂</td>
<td>¬(B₁ ⇒ B₂)</td>
<td>B₁</td>
<td>B₂</td>
</tr>
<tr>
<td>A₁ ⊕ A₂</td>
<td>A₁</td>
<td>A₂</td>
<td>¬(B₁ ⇒ B₂)</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

• Tworzenie tabel semantycznych nie jest jednoznaczne.
• Zbiór aksjomatów może być nieskończony.
• Dla nie licznych systemów logicznych istnieją procedury decyzyjne takie, jak dla rachunku zdaniowego.
• Procedura decyzyjna daje tylko odpowiedzi „tak” lub „nie”, czyli nie możemy poznawać wyników pośrednich.

12.2.2. Rezolucja

Niech C₁, C₂ będą klauzulami takimi, że l ∈ C₁, ̄l ∈ C₂. Klauzule C₁, C₂ nazywamy kolidującymi i mówimy, że kolidują względem komplementarnych literałów l, ̄l. Rezolwentą klauzul C₁ i C₂ nazywamy klauzulę C postaci:

\[ \text{Rez}(C₁; C₂) = (C₁ \setminus \{l\}) \cup (C₂ \setminus \{\bar{l}\}). \]

Klauzule C₁ i C₂ nazywamy klauzulami macierzystymi dla C. Na przykład:

\[ \begin{align*}
C₁ &= \{pqr\} \\
C₂ &= \{q\bar{r}s\} \\
C &= ((pqr \setminus \{r\}) \cup ((q\bar{r}s) \setminus \{r\})) \\
&= \{pq\bar{s}\}
\end{align*} \]

Rezolwenta klauzul C₁ i C₂ jest spełnialna wówczas, gdy klauzule C₁ i C₂ są spełnialne. Dowód metodą rezolwencji:

1. Niech S będzie zbiorem klauzul.
   \[ S₀ := S; \quad i := 0. \]
2. Załóżmy, że utworzyliśmy zbiór Sᵢ.
3. Wybierz parę klauzul kolidujących C₁, C₂ ∈ Sᵢ, które jeszcze nie były wybrane.
4. Niech C będzie rezolwentą C₁ i C₂.
5. Jeśli C = ∅ (pusta klauzula), to zakończ algorytm stwierdzając, że S jest nie-spełniany.
6. W przeciwnym razie
   \[ Sᵢ₊₁ := Sᵢ \cup \{C\} \]

172
12.3. Programowanie deklaratywne

\[ i := i + 1; \text{wróć do3.} \]

7. Jeśli \( S_{i+1} = S_i \) dla wszystkich możliwych wyborów klausul kolidujących, to zakończ algorytm stwierdzając, że \( S \) jest spełnialny.

Wyprowadzenie klausuli pustej ze zbioru \( S \) oznacza, że zbiór \( S \) jest niespełniany. Aby wykazać metodą rozulocji, że zbiór \( D \subseteq \{ \overline{j} \} \) należy zapisać \( S = D \cup \{ \overline{j} \} \) w postaci klausulowej i wyprowadzić pustą klausulę \( S \). Wyprowadzenie klausuli pustej ze zbioru klausul \( S \) nazywamy dowodem \( j \) przez zaprzeczenie.

12.3. Programowanie deklaratywne

Język deklaratywny jest to język programowania, w którym programista zamiast definiowania sposobu rozwiązania zadania, czyli sekwencji kroków prowadzących do uzyskania wyniku (algorytm), opisuje samo rozwiązanie. Pojęcie - programowanie deklaratywne obejmuje paradygmaty programowania, takie jak:

• programowanie funkcyjne,
• programowanie logiczne,
• programowanie z ograniczeniami,
• język dziedzinowy.

12.3.1. Programowanie funkcyjne

Programowanie funkcyjne zamiast sekwencyjnie wykonywać zadania (jak w przypadku języków imperatywnych np. Pascal, Assembler, C), wyznacza jedynie wartości poszczególnych wyrażeń i składają się jedynie z funkcji.

Funkcje są podstawowymi elementami języka funkcyjnego. Główny program jest funkcją, której podajemy argumenty, a w zamian otrzymujemy wyznaczoną wartość – wynik działania programu. Główna funkcja składa się tylko i wyłącznie z innych funkcji, które z kolei składają się z jeszcze innych funkcji. Funkcje takie dokładnie odpowiadają funkcjom w czysto matematycznym znaczeniu – przyjmują pewną liczbę parametrów i zwracają wynik. Każda operacja wykonywana podczas działania funkcji, a nie mająca związku z wartością zwracaną przez funkcję, to efekt uboczny (np. operacje wejścia wyjścia, modyfikowanie zmiennych globalnych).

Funkcje, które nie posiadają efektów ubocznych nazywane są funkcjami czystymi (ang. pure function). Usunięcie efektów ubocznych pozwala na wyznaczanie wartości wyrażeń w dowolnej kolejności. Ta zasada obowiązuje również w innych paradygmatach programowania deklaratywnego. Funkcje są bardzo ważną częścią funkcyjnych języków programowania. Funkcje są traktowane jak wartości, tak samo jak \emph{Integer} lub \emph{String}. Funkcja może zwracać inną funkcję, może przyjmować funkcję jako parametr, może być skonstruowana jako połączenie dwóch funkcji. Daje to olbrzymie możliwości łączenia poszczególnych modułów w jeden program. Funkcja, która oblicza wartość pewnego wyrażenia,
12. Programowanie deklaratywne i logika obliczeniowa

może brać udział w obliczeniach na przykład jako argument, czyniąc w ten sposób funkcje jeszcze bardziej modularnymi. W programowaniu funkcyjnym nie występują pętle zamiast nich trzeba zastosować rekurencję. Przykładem języka funkcyjnego jest język Haskell.

Przykład programu do obliczania silni w języku Haskell:

```haskell
factorial :: Integer → Integer
factorial 0 = 1
factorial n = n * factorial (n-1)
```

Tę samą funkcję można zapisać krócej:

```haskell
factorial n = if n > 0 then n * factorial (n-1) else 1
```

Powyższy rekurencyjny opis silni przypomina opis z książek matematycznych. Większa część kodu jest podobna do standardowego zapisu matematycznego.

Kolejny przykład pokazuje sposób implementacji kalkulatora RPM (ang. Reverse Polish Notation)

```haskell
calc :: String → [Float]
calc = foldl f [] . words
where
  f (x:y:zs) "+" = (y + x):zs
  f (x:y:zs) "-" = (y - x):zs
  f (x:y:zs) "*" = (y * x):zs
  f (x:y:zs) "/" = (y / x):zs
  f xs y = read y : xs
```

12.3.2. Programowanie logiczne

Programowanie logiczne w ogólności jest to używanie logiki matematyczne przy programowaniu. Bazuje na fakcie, że dowodzenie twierdzeń metodą wnosowania w tyl (ang. backward reasoning), zamienia formułę podaną w formie implikacji: „Jeśli $A_1$ i $A_2$… i $A_n$ to $B$” na procedurę: „rozwiąż/pokaż że $B$ oraz rozwiąż/pokaż że $A_1$ i $A_2$… i $A_n$”.


Wiedza reprezentowana może być przez klauzule Horna, co w połączeniu z backward reasoning oznacza, że programowanie logiczne łączy deklaratywną oraz proceduralną reprezentację wiedzy.

Matematyka rozróżnia pojęcia: język obiektowy oraz metajęzyk, dlatego języki logiczne pozwalają programować na meta poziomie. Najprostszy meta interpreter nazywa się „Vanilla”.

```prolog
solve(true).
solve((A,B)):- solve(A), solve(B).
```
12.3. Programowanie deklaratywne

solve(A):- clause(A,B), solve(B).

W tym wypadku true reprezentuje pustą koniunkcję, klauzula clause(A,B) oznacza, że istnieje na poziomie obiektowym klauzula w formie A:-B. Przykładem języka logicznego jest Prolog oraz język Oz. Przykład kodu w języku Prolog (http://cs.union.edu/~striegnk/learn-prolog-now/):

woman(mia).
woman(jody).
woman(yolanda).
playsAirGuitar(jody).
happy(yolanda).
listensToMusic(yolanda):- happy(yolanda).

Pytania wprowadza się po znaku ?-

12.3.3. Programowanie z ograniczeniami

Programowanie z ograniczeniami bazuje na modelowaniu zadania jako problemu spełnienia ograniczeń. Ograniczenia są zależne od dziedzin zmiennych, których dotyczą. Najpopularniejszą i pierwszą dziedziną zmiennych była skończone podzbiory liczób naturalnych. Innymi dziedzinami są: skończone zbior, drzewa, rekordy, przedziały rzeczywiste. Najistotniejszą cechą i największą zaletą programowania z ograniczeniami jest ich propagacja. Zasadą działania propagacji jest usuwanie wartości nie spełniających ograniczeń z domen zmiennych. Programowanie z ograniczeniami poszukuje takiego stanu, w którym jak największej liczby ograniczeń jest spełnionych. Przykładowe języki to: B-Prolog, CHIP V5, Oz, Claire, Curry. Przykład napisany w języku B-Prolog przez Neng-Fa ZHOU na rozwiązywanie SUDOKU:

go:-
instance(N,A),
Vars @= [A[I,J] : I in 1..N, J in 1..N],
Vars :: 1..N,
foreach(I in 1..N, [Row],
   (Row @= [A[I,J] : J in 1..N], all_distinct(Row))),
foreach(J in 1..N, [Col],
   (Col @= [A[I,J] : I in 1..N], all_distinct(Col))),
M is floor(sqrt(N)),
foreach(I in 1..M, J in 1..M,
   [Square],
   (Square @= [A[I1,J1] : I1 in (I-1)*M+1..I*M,
               J1 in (J-1)*M+1..J*M],
    all_distinct(Square))),
labeling([ff],Vars),
12. Programowanie deklaratywne i logika obliczeniowa

foreach(I in 1..N,  
    (foreach(J in 1..N,[Aij],  
        (Aij @= A[I,J], format("2d ",[Aij])),nl)).

12.4. Przykład zastosowania


sendmore(Digits) :-  
    Digits = [S,E,N,D,M,O,R,Y], % Tworzenie zmiennych  
    Digits :: [0..9], % Przypisanie zmiennym dziedziny  
    S #\= 0,  % Ograniczenie: S != 0  
    M #\ = 0,  
    alldifferent(Digits), % Różne wartości zmiennych  
    1000*S + 100*E + 10*N + D + 1000*M + 100*O + 10*R + E #=  
    10000*M + 1000*O + 100*N + 10*E + Y, % Kolejny warunek  
    labeling(Digits). % Rozpoczęcie poszukiwania

Stworzenie algorytmu oraz zaimplementowanie go w którymkolwiek z języków imperatywnych nie jest tak proste jak w tym przykładzie. Innym przykładem może być program, rozwiązujący problem n-królowych (zwany problemem 8 hetmanów). Jest to problem rozmieszczenia królowych na szachownicy, tak, aby się nie szachowały. Poniższy kod, napisany w języku Curry, przedstawia rozwiązanie tego problemu.

queens options n l =  
    gen_vars n =:= l &  
    domain l 1 (length l) &  
    all_safe l &  
    labeling options l

all_safe [] = success  
all_safe (q:qs) = safe q qs l & all_safe qs

safe :: Int -> [Int] -> Int -> Success  
safe_ [] _ = success  
safe q (ql:qs) p = no_attack q ql p & safe q qs (p+#1)
12.4. Przykład zastosowania

no_attack q1 q2 p = q1 /=# q2 & q1 /= #q2+ #p & q1 /= # q2- #p

gen_vars n = if n==0 then [] else
   var : gen_vars (n-1) where var free

-- queens [] 8 1 where l free
-- queens [FirstFail] 16 1 where l free

12.4.1. Prolog

Najbardziej znanym i najczęściej używanym językiem programowania logicznego jest język Prolog. Prolog, wraz z jego różnymi odmianami, może służyć do programowania, wykorzystując prawie wszystkie paradigmy języków deklaracyjnych. Oto niektóre implementacje języka Prolog:

- SWI-Prolog,
- YAProlog,
- Visual Prolog,
- GNU Prolog,
- Ciao Prolog.

Prolog (fr. *Programmation en Logique*) to język programowania, służący do obliczeń symbolicznych, stworzony w 1972 roku przez Alaina Colmeraurera i Philippe’a Roussela [7]. Programowanie w Prologu odbywa się za pomocą definiowania faktów i relacji oraz zadawania pytań o relacje. Składnia języka składa się ze znaków oraz termów. Znaki to duże i małe litery, cyfry oraz symbole tj. +, -, ;, . . . . Natomiast termy to:

- Liczby - całkowite, rzadziej zmiennoprzecinkowe.
- Atomy - ciągi znaków, zaczynające się małą literą (czlowiek). Ciągi znaków umieszczone w apostrofach (‘czlowiek’) są to nazwy atomów. Ciągi znaków specjalnych tj. @, ====, ;, ;-, niektóre z nich mają predefiniowane znaczenie.
- Zmienna - ciągi znaków zaczynające się wielką literą lub podkreślением (X, _zmienna).
- Termy złożone - składające się z funktora, który musi być atomem oraz z sekwencji argumentów (rozdzielonych przecinkami) umieszczonej w nawiasach (ojciec(X, ojciec(Y, ela))).

Kod składa się z sekwencji klawuiz. Umożliwia wpisywanie faktów i predykatów (predykat(arg1, arg2,...)).. reguł oraz zapytań [8]. Reguły mogą mieć postać jest(światło) :- włączony(przycisk). Zapis :- oznacza „wtedy, gdy” lub „jeśli”. Ta reguła oznacza, że zdanie jest(światło) jest prawdziwe wtedy, gdy prawdziwe jest zdanie włączone(przycisk). Reguły mogą zawierać zmienne. Termy w klawuizach mogą być oddzielone przecinkami, które oznaczają koniunkcję lub średniaki, które oznaczają alternatywę. Na przykład reguła: ojciec(X, Y) :- rodzic(X, Y), jest_rodzaju_męskiego(Y). oznacza "dla każdych X i Y, jeśli rodzic(X, Y) i jest_rodzaju_męskiego(Y) to ojciec(X, Y)". Pytanie
12. Programowanie deklaratywne i logika obliczeniowa


Kod programu z pliku można wczytać do interpretera poleceń consult(nazwa pliku). Wcześniej wpisane w pliku fakty mogą być usuwane, bądź dodawane dynamicznie poleceń retract(fakt), oraz assert(fakt). W języku Prolog za pomocą reguł z użyciem zmiennych oraz rekurencji można tworzyć nowe predykaty. Umożliwia to między innymi obliczenia arytmetyczne, logiczne oraz operacje na listach.

12.4.2. Otter - system automatycznego wnioskowania


Reprezentowanie wiedzy

Zapisywanie reguł dozwolone jest zarówno w postaci standardowej, używając symboli logicznych, jak też w postaci klauzul. Przy tłumaczeniu zdań z języka naturalnego na język logiki problemy pojawiają się zarówno na poziomie semantyki jaki i syntaktyki. Niech jako przykład posłuży zdanie:

- „Do dyspozycji mamy 3 klocki. Każdy z nich jest mniejszy od poprzedniego.”

Zapisanie tej informacji w języku logiki wymaga pewnych przekształceń i analizy. Ze zdania powyżej można wyciągnąć następujące wnioski:

178
12.4. Przykład zastosowania

1. Istnieją: \( \text{klocek1, klocek2, klocek3} \), wszystkie z nich należą do zbioru \( \text{klocki} \).
2. \( \text{klocek1} \prec \text{klocek2} \).
3. \( \text{klocek2} \prec \text{klocek3} \).

Analizując 2 i 3, prawdziwe jest stwierdzenie, że \( \text{klocek1} \prec \text{klocek3} \). Zdanie 1 można zapisać w także: „W zbiorze klocki znajdują się 4 elementy.” Dają one następujący zapis w składni programu Otter:

1. \( \exists x (\text{Klocki}(x)). \)
2. \( \forall x (\text{Klocek1}(x)). \)
3. \( \forall x (\text{Klocek2}(x)). \)
4. \( \forall x (\text{Klocek3}(x)). \)

Zapisując warunek \( \text{klocek1} \prec \text{klocek2} \) w formie zdania Jeśli ... to otrzymamy:

- „Jeśli istnieje \( \text{klocek1} \) i istnieje \( \text{klocek2} \) to \( \text{klocek1} \prec \text{klocek2} \)

co można zapisać:

- \( \forall x \forall z (\text{Block1}(x) \land \text{Block2}(z) \rightarrow \text{Smaller}(x,z)). \)

Mając tak zdefiniowaną zależność, zapisanie reguły przechodniej dla \( \text{klocek1} \prec \text{klocek3} \), może mieć następującą postać:

- \( \forall x \forall z \forall p (\text{Smaller}(x,z) \land \text{Smaller}(z,p) \rightarrow \text{Smaller}(x,p)). \)

Jeśli zaszłyby potrzeba zapisania informacji o tym, który klocek jest większy, wygodnie jest korzystać z już zdefiniowanych reguł. W tym wypadku można zapisać:

- "Jeśli \( \text{klocek1} \prec \text{klocek2} \) to \( \text{klocek2} \succ \text{klocek1}. "
- \( \forall x \forall y (\text{Smaller}(x,y) \rightarrow \text{Bigger}(y,x)). \)

12.4.3. Program symulujący świat klocków

W celu zobrazowania deklaracyjnego podejścia do programowania zaplanowano stworzyć program symulujący świat klocków. Świat klocków można sobie wyobrazić jako stół, na którym znajdują się klocki różnego koloru. Świat ten ma swój zestaw reguł oraz pozwala na zmianę ułożenia klocków. Układ klocków musi być zgodny z ustalonymi regułami.

Struktura programu

Program składa się z graficznego interfejsu, w którym wyświetlane jest rozmieszczenie klocków. Aplikacja umożliwia użytkownikowi przemieszczanie klocków. Do programu dołączony jest plik z regułami, według których można dokony-
12. Programowanie deklaratywne i logika obliczeniowa

wać zmian w przedstawionym świecie. Plik z regułami można zmieniać i tworzyć swój własny zestaw reguł.

Część graficzna programu została napisana w języku C++ przy użyciu biblioteki Qt na platformie Linux. Obliczenia logiczne są wykonywane przez system automatycznego wnioskowania Otter. Komunikacja między programami odbywa się za pomocą plików i strumieni.

Działanie programu

Program zapisuje stan świata wraz ze zbiorom reguł do pliku. Następnie uru-
chamia program Otter z nazwą zapisanego pliku. Na wyjściu programu automa-
tycznego wnioskowania pojawi się zestaw poprawnych ruchów, które użytkownik
może wykonać. Po wykonaniu ruchu przez użytkownika program sprawdzi, czy
takie przestawienie klocków znalazło się na liście.

Niestety nie udało nam się znaleźć funkcji Otter, umożliwiającej wykrywanie
sprzeczności. Gdyby system Otter potrafił wysłać sygnał, gdy natrafi na sprzecz-
ność, wtedy program mógłby sprawdzać, czy dane ułożenie nie jest sprzeczne
z regułami.

Program mógłby stanowić podstawę do stworzenia aplikacji symulującej
wieże Hanoi albo manipulator, ustawiający elementy na linii produkcyjnej lub
w magazynie.

Literatura

MIARY PODOBIEŃSTWA W ONTOLOGIACH

R. Gierczak, T. Tylecki

W rozdziale tym zostaną przedstawione informacje wyjaśniające pojęcie ontologii oraz miar podobieństwa stosowanych przy porównywaniu ontologii. Poruszony będzie również temat narzędzi oraz języków do edycji i zapisu ontologii. Zostaną zaprezentowane przyjęte założenia projektowe, opis pomysłu realizacji zadania, a także sposób w jaki zostało wykonane przy pomocy dostępnych narzędzi.

13.1. Ontologia

Dostępna literatura zawiera kilka definicji tego słowa. W szeroko rozumianym podejściu filozoficznym oraz dużo bardziej nas interesującym podejściu informatycznym. Słowo ontologia wywodzi się z języka greckiego będącego zlepkiem słowa ὁν (on), w dopełniaczu ὅντος (ontos), oznaczającego „byt”, „rzeczywiste będący”, „istniejący” oraz słowa ὁλογία (-logia), oznaczającego „słowo”, „nauka”, „teoria” [6], stąd może oznaczać naukę o szeroko pojmowanym bycie, czyli o wszystkim. Termin ten został użyty w XVII w. Do spopularyzowania słowa przyczynił się Christian Wolff, który podzielił filozofię na ontologię, kosmologię i psychologię. Przed nim w swoich pracach używał tego terminu niemiecki teolog i filozof Johannes Clauberg, pierwszy raz słowo to zostało użyte w obecnej formie w słowniku filozoficznym napisanym przez pomorskiego teologa Johannesa Micraeliusa [7]. Ontologia według dzisiejszej definicji w pojęciu filozoficznym jest dziedziną metafizyki, która zajmuje się badaniem i wyjaśnianiem natury jak i kluczowych właściwości oraz relacji rządzących wszelkimi bytami, bądź głównych zasad i przyczyn bytu. Nie zajmuje się tym w jaki sposób człowiek postrzega świat, ale stawia pytania „jak można wszystko poklasyfikować”, „jakie klasy bytów są niezbędne do opisu i wnioskowania na temat zachodzących procesów?”, „jakie klasy bytów pozwalają wnioskować o prawdzie?”, „na podstawie jakich klas bytów można wnioskować o przyszłości”.

Rozwój techniki oraz próba wykorzystania maszyn w coraz to nowych zadańach spowodowała konieczność opisu rzeczywistości, w taki sposób aby był on również „rozumiały” przez maszyny. Próby opisu opierające się jedynie na lek-
13. Miary podobieństwa w ontologach

sykalnym podejściu do problemu nie dają w większości przypadków pożądan
nych efektów, ponieważ nie oddają one relacji zachodzących pomiędzy obiektami. Sprawdza się jedynie w przypadkach, gdzie występują zbiory słów o wspól
nym rdzeniu dające się w łatwy sposób kategoryzować, choć często mogą wystą
pić przypadki obiektów o wspólnym rdzeniu w nazwie znaczeniowo różniące się, dla których metoda ta zawodzi. Rozwiązanie takie jest również niepraktyczne, ze względu na opis obiektów w różnych językach, co nie zapewnia przenośności danego opisu rzeczywistości. Jednym ze sposobów poradzenia sobie z tym pro
blemem jest stworzenie zapisu uwzględniającego rzeczywiste powiązania mie
dzy obiektami. Zapis taki powinien oddawać znaczenie obiektów oraz umiesz
cać je w hierarchicznej strukturze umożliwiającej badanie powiązań pomiędzy
nymi. Właśnie w takim kontekście (informatycznym) termin ontologia pojawił się w 1967 r. w badaniach dotyczących modelowania danych, jednakże zyskał on do
piero na popularności wraz z rozwojem internetu i konieczności przetwarzania
informacji.

Można przytoczyć następującą definicję w znaczeniu informatycz
nym [2]: „specjalizacja konceptualizacji” (ang. specification of a conceptualiza
tion). Ontologia określa zbiór reprezentacyjnych jednostek elementarnych sta
nowiących dziedzinę modelu wiedzy. Jednostki elementarne są typowo kla
sam (zbiorami), atrybutami (właściwościami) i relacjami (powiązaniami między
członkami klas). Opis jednostek elementarnych niesie z sobą informacje o ich
znaczeniu oraz ograniczeniach ich stosowania. W kontekście systemów baz da
nych, ontologia może być rozpatrywana jako poziom abstrakcji modelu danych
analogicznie do hierarchicznych i relacyjnych modelów, przeznaczona do mode
lowania wiedzy o indywidualach, ich atrybutach oraz ich relacjach w stosunku do
innach indywidualiów.

Formalna definicja wg [1] przedstawia ontologię jako parę:

$$(O, L)$$

gdzie $O$ jest strukturą ontologii, a $L$ jest leksykonem ontologii. Struktura ontologii ma postać:

$$O = \{C, R, Hc, Rel, A\}$$

gdzie: $C$ - zbiór wszystkich pojęć wykorzystanych w modelu, $R$ - zbiór nietaksono
micznych relacji, definiowane jako nazwane połączenia między pojęciami, $Hc$
 - zbiór taksonomicznych relacji pomiędzy pojęciami (relacja $Hc(C_1; C_2)$ oznacza, że $C_1$ jest podpojęciem $C_2$), $Rel$ - funkcja $R \rightarrow C \times C$, która relacji nietaksonomicz
nej przyporządkowuje uporządkowaną parę pojęć; $A$ - zbiór aksjomatów.

Leksykon zawiera interpretacje rozumienia pojęć i relacji występujących mię
dzy nimi. Leksykon ma postać:

$$L = \{Lc, Lr, F, G\}$$

gdzie $Lc$ - definicja leksykonu dla zbioru pojęć, $Lr$ - definicja leksykonu dla zbioru
relacji, $F$ - referencje dla pojęć, $G$ - referencje dla relacji.
13.2. Sposoby zapisu ontologii


Język XML nie jest wystarczający by możliwe było jednoznaczne zdefiniowanie zasobów, ponieważ potrzebny jest standard, który pozwoliłby maszynom rozumieć identycznie znaczenie poszczególnych tagów, a także powiązań między nimi. Problem ten częściowo rozwiązuje RDF [10] (ang. Resource Description Framework), który jest specyfikacją modelu metadanych, określoną przez W3C. Celem RDF jest umożliwienie maszynowego przetwarzania abstrakcyjnych opisów zasobów w sposób automatyczny. Może służyć zarówno do wyszukiwania da-
13. Miary podobieństwa w ontologiiach


OWL (ang. Web Ontology Language) jest językiem wyższego poziomu, służącym do zapisu ontologii. Istnieją trzy odmiany języka OWL: OWL Lite, OWL DL, OWL Full. Język ten został zaprojektowany dla aplikacji, które mają za zadanie przetwarzanie zawartości informacji zamiast po prostu przedstawiać informacje ludziom. Korzysta z XML i RDF. Ontologia w OWL może zawierać następujące elementy:

- usystematyzowaną relację pomiędzy klasami,
- właściwości typu danych, opisującymi atrybuty elementów klas,
- właściwości obiektów, opisujące relacje pomiędzy elementami klas,
- instancje klas,
- instancje właściwości.

13.3. Miary podobieństwa


Hierarchiczna budowa reprezentacji ontologii sprawia, że możliwe jest zastosowanie metod pomiaru podobieństwa opierających się na powiązaniach zachodzących pomiędzy obiektami. Dzięki relacjom wiadomo jakie typu istnieją powiązania, co przenosi poziom pomiaru na wyższy poziom abstrakcji. W dalszej
13.3. Miary podobieństwa

części rozdziału zostanie zaprezentowane kilka zaczerpniętych z literatury miar wykorzystywanych do pomiaru podobieństwa w ontologii [12,13,14].

13.3.1. Podobieństwo wewnątrz ontologii

Odległość ontologiczna


\[ \text{Dist}(A,B) = |AX| + |BX| \]

Rys. 13.2: Przykładowy graf.

Przykładowo odległość pomiędzy węzłami słoń i wieloryb w grafie przedstawionym na rys. 13.2 wynosi 2 jako suma odległości do wspólnego przodka którym jest węzeł ssak. W tym przypadku wszystkie gałęzie grafu mają wartość równą 1. Nic jednak nie stoi na przeszkodzie, aby gałęzie grafu posiadały różne współczynniki wagowe.

Podejście informacyjne

Algorytm (odległość między klasami A i B):

\( n \) – liczba wszystkich obiektów (instancji) w modelu.

Wyznaczamy najmniejszego wspólnego przodka X.

\[ P(A) = \frac{\text{liczba wszystkich instancji klasy } A}{n} \]
\[ P(B) = \frac{\text{liczba wszystkich instancji klasy } B}{n} \]
\[ P(X) = \frac{\text{liczba wszystkich instancji klasy } X}{n} \]
\[ \text{Sim}(A,B) = 2\log \frac{P(X)}{\log(P(A) + \log(P(B))} \]

185
13. Miary podobieństwa w ontologiach

Odległość Levenshteina

W podstawowej wersji odległość Levenshteina ma zastosowanie do ciągów znaków, jako miara podobieństwa (odległości) napisów. Jest określona jako liczba operacji elementarnych, które należy zastosować, aby przekształcić jeden napis w drugi. Do operacji elementarnych należą m.in. usunięcie, wstawienie, zamiana znaku. Przykładowo dla słów *słon* i *słoń* odległość Levenshteina jest zerowa. Słowa te są identyczne i niewymagane jest zastosowanie jakichkolwiek operacji elementarnych do przekształcenia jednego ciągu znaków w drugi. W przypadku napisów *słon* i *słońce* odległość ta wynosi 2, gdyż konieczne jest dwukrotne zastosowanie operacji wstawienia znaku.

Dla ontologii odległość ta jest zdefiniowana podobnie, jako liczba zmian jakie należy wykonać by przekształcić jedną klasę w drugą. W tym przypadku zmiana jest rozumiana jako wstawienie, usunięcie modyfikacja atrybutu, wartości atrybutu, relacji lub typu relacji.

Każej z tych operacji elementarnych możliwe jest przypisanie funkcji wagowej. Uznając, że koszty modyfikacji są większe od kosztów usunięcia elementu, czy też wstawienie wartości atrybutu ma mniejszą wagę niż dodanie relacji. Wyznaczenie odległości w tej metodzie rozpoczyna się od identyfikacji atrybutów, ich wartości oraz relacji porównywanych klas. Kolejnym krokiem jest obliczenie liczby zmian lub sumy kosztów tych zmian potrzebnych do przekształcenia jednej klasy w drugą. W przypadku zastosowania funkcji wagowej określającej koszty poszczególnych transformacji, konieczne jest przypisanie zmiennej $W$ kosztem najgorszej transformacji. Odległość Levenshteina pomiędzy klasami $A$ i $B$ jest wtedy obliczana jako:

$$\text{Dist}(A, B) = \frac{\text{liczba zmian lub koszt transformacji}}{W}$$

Podejście wektorowe

W tym sposobie oceny podobieństwa każda klasa (obiekt) jest rozpatrywana jako element $k$-wymiarowej przestrzeni wektorowej, w którym każdy z wymiarów to pewna własność klasy. Przykładowo odnosząc się do reprezentacji jak na poniższym rysunku wektor ten może mieć następującą postać $\vec{k} = [\text{liczba koł, czy_ma_silnik, czy_m_kierownicę}]=\text{nem} do cech danej klasy. Ogólny sposób wyznaczenia odległości pomiędzy dwoma dowolnymi klasami $A$ i $B$ zakłada wyznaczenie długości wektora $|\vec{a}|$ reprezentującego klasę $A$. Wyznaczenie wektora $|\vec{b}|$, reprezentującego klasę $B$. Obliczenie podobieństwa jako odległości cosinusowej

$$\text{Sim}(A, B) = \frac{\vec{a} \cdot \vec{b}}{|\vec{a}| \cdot |\vec{b}|}$$

Chcąc obliczyć przykładowo podobieństwo pomiędzy obiektami *samochód* i *motocykl*, uwzględniając cechy klas jak dla wektora $\vec{k}$ (zobacz rys. 13.3) otrzymujemy odpowiednio wektory $\vec{a} = [4, 1, 1], \vec{b} = [2, 1, 1]$. Dla *karoc* wektor $\vec{c} = [4, 0, 0]$. 186
W przypadku naszego przykładu odległości między klasami obliczone z powyższych zależności są następujące:

- \( \text{Sim(samochód,motocykl)} = 0.96 \)
- \( \text{Sim(samochód,karoca)} = 0.94 \)
- \( \text{Sim(motocykl,karoca)} = 0.82 \)

Zaadaptowany algorytm TF-IDF

Ten sposób pomiaru podobieństw w ontologii bazuje na metodzie służącej do określania podobieństwa tematyki analizowanych tekstów. Każdy element ontologii tworzy osobny dokument. Dokument ten posiada nazwę, atrybuty i opis występujących relacji. Elementy te to słowa tego dokumentu. Każde słowo występujące w dokumentach ma wagę. Miara wyliczana w tym algorytmie oparta jest na często追问wości występowania danego słowa w dokumencie, oraz liczbie dokumentów, w których dane słowo występuje. Cały algorytm opiera się na następującym wzorze:

\[
TF-IDF(D, w) = \frac{TN(w)}{|D|} \cdot \log \frac{N}{DN(w)}
\]

gdzie:
- \( TN(w) \) – ilość występień słowa \( w \) w dokumencie \( D \),
- \( |D| \) – ilość wszystkich słów w dokumencie \( D \),
- \( DN(w) \) – ilość dokumentów, w których występuje słowo \( w \),
- \( N \) – ilość wszystkich dokumentów.

13.3.2. Podobieństwo pomiędzy ontologiami

Jedną z metod pomiaru podobieństwa pomiędzy encjami w dwóch różnych ontologii OWL.DL jest sposób zaproponowany w [3]. Zaproponowany sposób bazuje na informacjach wydobytych z encji. Encja w OWL.DL może być klasą, relacją, instancją lub każdą częścią ontologii. W metodzie tej próbuje się wydobyć tak wiele informacji, jak to jest możliwe z opisu encji. Wydobyte składowe są porównywane dając częściowe wartości prawdopodobieństwa, które są następnie przemnażane przez funkcję zmiennową i sumowane. W obliczaniu prawdopodobieństwa bierze się pod uwagę zdefiniowane w OWL.DL i RDF składowe takie jak np. rdf:type, owl:equivalentClass, które mogą być wydobyte z opisu encji. W przedstawionym systemie encja ontologii opisanej przy pomocy OWL.
13. Miary podobieństwa w ontologii

DL może być klasą która ma na nazwę (URI), bądź być klasą dowolną lub relacją. Encja jest opisana w OWL DL za pomocą jednostek elementarnych, takich jak rdf:id, rdfs:range, owl:subClassOf. Każda z tych jednostek elementarnych niesie z sobą kawałek wiedzy o całości znaczenia encji. Dlatego można rozpatrywać podobieństwo między encjami jako kombinację częściowych prawdopodobieństw, które są prawdopodobieństwami pomiędzy częściami opisu opisu używanych jednostek elementarnych. Kombinacja podobieństwa jest zmiennową wagową sumą oszacowaną z częściowych podobieństw. Ontologia OWL DL jest dokumentem RDF. Zostaje rozpatrzona jako zbiór trójek RDF. O jest ontologią, (s, p, o) trójką, gdzie s, p, o są odpowiednio podmiotem, predykatem i obiektem, O = (s, p, o). Dla każdej trójki zawartej w ontologii O, w opisie jednostki, p jest predykatem, który jest jednym z 33 RDF(S) lub OWL jednostek podstawowych. Dla trójk zostały zdefiniowane:

T(e) = {(e, p, o) | (e, p, o) ∈ O} - zbiór trójek RDF mających encje e jako podmiot

O(e) = {o | (e, p, o) ∈ T(e)} - zbiór obiektów RDF, które są częścią trójek mających e jako podmiot

E(e) = {(p, o) | (e, p, o) ∈ T(e)} - zbiór par predykat-obiekt, gdzie pierwszy jest predykatem, a drugi obiektem w trójce mającej e jako podmiot.

Podobieństwo między składowymi encji jest podobieństwem pomiędzy zbiorami par E(e1) oraz E(e2). Dla porównania tych dwóch zbiorów par został zaproponowany następujący algorytm:

a) Należy zidentyfikować zbiór predykatów P_{c1}, które zawierają predykaty par E(e1), mające podobne predykaty w parach w E(e2) i podobnie P_{c2}. P_{c} jest sumą tych dwóch zbiorów.

\[ P_{c1} = \{ p | p \in P(s_1) \land (\exists q \in P(s_2), SimPred(q, p) > 0) \} \]

\[ P_{c2} = \{ p | p \in P(s_2) \land (\exists q \in P(s_2), SimPred(q, p) > 0) \} \]

\[ P_{c} = P_{c1} \cup P_{c2} \]

SimPred(q, p) jest funkcjonalnym podobieństwem pomiędzy dwoma predykatami, które są wartościami RDF lub OWL. Jest ona zdefiniowana następująco (i) SimPres(p, p) = 1; (ii) SimPred(p, q) jest wartością z przedziału [0, 1] jeśli p ≠ q. Niektóre właściwości OWL mogą być rozpatrzone jako podobne semanticznie np. owl:cardinality i owl:maxCardinality.

b) Aby oszacować podobieństwa częściowe predykatów mogących pojawiać się kilkakrotnie w opisie encji np. rdfs:label, rdfs:subPropertyOf,..., można zgrupować obiekty w trójkach mające te same predykaty p. W ten sposób otrzymamy zbiór obiektów dla dwóch encji w dwóch ontologii: O(s_2, p) i O(s_1, p). Następnie obiekty z tych dwóch zbiorów są grupowane w pary w porządku do oszacowanego prawdopodobieństwa pomiędzy zbiorami obiektów. Do tworzenia par obiektów można wykorzystać następujący algorytm. Niech o_{1i} i o_{2j}
13.3. Miary podobieństwa

są obiektami w zbiorze, \( o_{1i} \) zawiera się w \( O(s_1, p) \), a \( o_{2j} \) w \( O(s_2, p) \). Należy znaleźć \( o_{1i} i o_{2j} \), których \( Sim_{total}(o_{1i}, o_{2j}) \) jest maksymalna. \( (o_{1i}, o_{2j}) \) jest parą obiektów. Usunąć \( o_{1i} \) ze zbioru \( O(s_1, p) \) i usunąć \( o_{2j} \) ze zbioru \( O(s_2, p) \). Następnie powtarzać znajdywanie par i usuwanie do momentu, gdy więcej par nie zostało znalezionych. Ostatecznie po procesie znajdywania par, podobieństwa pomiędzy parami obiektów są sumowane i dzielone przez maksymalną liczbę obiektów w obu zbiorach.

\[
Sim_{partial}(s_1, s_2, p) = \frac{\Sigma_{(o_{1i}, o_{2j}) \in Par_{ing}(O(s_1, p), O(s_2, p))} Sim_{total}(o_{1i}, o_{2j})}{\max(|O(s_1, p)|, |O(s_2, p)|)}
\]

c) Dla predykatów które mogą pojawić się jedynie raz w opisie encji, takich jak rdf:id, owl:complementOf itd. częściowe podobieństwo dwóch encji bazuje na podobieństwie pomiędzy dwoma obiektami z dwóch trójk: jeśli dwa obiekty są podobne ich klasy są również podobne.

d) Zależnie od modelu ontologii, opis encji (klas, relacji) może składać się z jednej lub kilku własności RDF(S)/OWL. Dana właściwość (predykat) może pojawić się w opisie, jednak może się zdarzyć, że jej nie będzie. W takim przypadku nie prawidłową praktyką byłoby wyliczanie częściowych podobieństw o takiej samej ustalonej wartości funkcji wagowej. Lepszym rozwiązaniem jest przypisanie różnych funkcji wagowych dla różnych predykatów, jak jest to zaprezentowane w poniższym wzorze:

\[
Sim_{component}(s_1, s_2) = \sum_{p_j \in P_c} \phi(w_{p_j}) Sim_{partial}(s_1, s_2, p_j)
\]

gdzie \( \phi(w) \) jest adaptacyjną funkcją modyfikującą wagę. W metodzie tej można zdefiniować 33 wagi odpowiadające jednostkom RDF(S)/OWL, tak aby ich suma wynosiła 1.

Podobieństwo między dwoma encjami może być wyprowadzone nie tylko poprzez podobieństwo opisu składników, ale także poprzez podobieństwo struktury grafu RDF, który je reprezentuje. Tu została zaproponowana metoda pomiaru podobieństwa jako stosunku liczby podobnych predykatów do maksymalnej liczby predykatów z obu encji.

\[
Sim_{graph}(s_1, s_2) = \frac{|P_c|}{\max(|P(s_1)|, |P(s_2)|)}
\]

Całkowite podobieństwo jest obliczane jako ważona suma podobieństwa grafu i składowych:

\[
Sim_{total} = w_{component} * Sim_{component} + w_{graph} * Sim_{graph},
\]

gdzie \( w_{component} + w_{graph} = 1 \).
13. Miary podobieństwa w ontologii

13.4. Zastosowania pomiaru podobieństwa w ontologii


![Wyszukiwarka Swoogle](http://swoogle.umbc.edu/)

Rys. 13.4: Wyszukiwarka Swoogle.

Miary podobieństwa można wykorzystać również do automatycznego generowania odpowiedzi na zapytania. Daje to możliwość budowania systemów eksperckich, których siła w dużym stopniu zależy od jakości oraz usystematyzowania danych. Jednak duży wpływ na jakość ma sposób wyszukiwania i dopasowania odpowiedzi zależny od zastosowanej metody określania podobieństwa, która ma znaczny wpływ na wychwytywanie związków między danymi.

190
13.5. Zrealizowany projekt


Literatura

13. 

Miary podobieństwa w ontologiach


Od redaktora i wydawcy

Czym jest wiedza?
Nad tym pytaniem zastanawiali się już starzyńczycy, głowili się filozofowie, tani nami główę psycholodzy, próbowali na nie odpowiadać praktyki. Choć wiedza południowo-kardynalna jest z informacją, nauką doświadczaniem, zbiorem faktów, nie posiada ona jednej, uniwersalnej definicji. W zależności od kontekstu odpowiedź na to pytanie może przybrać różną postać i formę.

Czym jest przetwarzanie wiedzy?
To kolejna niewiadoma, sięgająca w swej maturze do sposobów reprezentacji wiedzy, jej interpretacji i wykorzystania, włączając w to metody wnioskowania i podejmowania decyzji. Podobnie jak w pytaniu o wiedzę, mnogość możliwych odpowiedzi może tu być ogromna.

Czym jest komputerowe przetwarzanie wiedzy?
Odpowiedź na to pytanie jest projekcją sumy odpowiedzi na powyższe dwa pytania na płaszczyznę zdefiniowaną przezmiotnikiem „komputerowe”. Mówiąc prosćie jest to dziedzina, w której wykorzystuje się komputery do rozwiązania złożonych problemów zdefiniowanych na różnych poziomach abstrakcji. Wykazuje ona poza samą implementacją algorytmów ekstrahujących wartości parametrów opisujących obiekt, na przykład na poziomie sztucznej inteligencji i inteligencji istot żywych, tworząc pomost pomiędzy czołówką, co jest w stanie sami przekształcać, a czym, co umyka naszym zmysłem z powodu wielkiej złożoności albo szczegółowości.

W niniejszej książce zebrano opracowania wykonane przez studentów W roku Automatyki i robotyki w ramach kursu Komputerowe przetwarzanie wiedzy prowadzonego przez mnie na Politechnice Wrocławskiej w semestrze zimowym 2009/2010. Zgodnie z zarysowanym kształtem odpowiedzi na ostatnie z powyższych pytań, zadanie studentów polegało nie tyle na zaimplementowaniu jakiegoś opublikowanego bądź autorskiego algorytmu, co na posłużeniu się zdobywaną dzięki temu wiedzą do rozwiązania jakiegoś problemu na wyższym poziomie abstrakcji.

Zakres tematyczny opracowań można zawrzeć w następującym zestawieniu:
- Przetwarzania dokumentów
- Porozumienia societyalne
- Modulo algorytmowe
- Ochrona informacji w sieci
- Reprzisztacyja wiedzy
- Operacje na danych
- Logika, programowanie i wnioskowanie
- Ontologie informatyczne

Mam nadzieję, że lektura tych opracowań okaza się interesująca dla czytelnika.

Tomasz Kubik
Wrocław, wrzesień 2011