Uniwersytet im. Adama Mickiewicza w Poznaniu
Wydział Matematyki i Informatyki

Marcin Walas

Wnioskowanie czasowo-przestrzenne w systemie Question Answering

Rozprawa doktorska
napisana pod kierunkiem
prof. UAM dra hab.
Krzysztofa Jassema

Poznań, 2013

Dziękuję mojemu promotorowi
profesorowi Krzysztofowi Jassemowi
za pomoc, wsparcie i cierpliwość.

Spis treści

Spis tabel

Spis rysunków

Rozdział 1

Wprowadzenie

1.1. Motywacja

Przedmiotem rozprawy jest opracowanie algorytmów wnioskowania czasowego i przestrzennego dla systemów odpowiadania na pytania (systemów QA). Zadaniem systemu QA jest dostarczenie użytkownikowi zwięzłej i dokładnej odpowiedzi na pytanie zadane w języku naturalnym.

Systemy QA nabierają we współczesnym świecie coraz większego znaczenia dzięki dynamicznemu rozwojowi systemów rozpoznawania mowy (ang. *Automatic Speech Recognition*, w skrócie ASR) oraz systemów syntezy mowy (ang. *Text to Speech Recognition*, w skrócie TTS). Połączenie systemów ASR i TTS z systemami QA pozwala na opracowanie systemu dialogowego, który w sposób naturalny będzie odpowiadał na pytania użytkownika.

W rozprawie omówiono autorskie algorytmy odpowiadające na pytania z aspektem czasowym i przestrzennym, Przykładem pytania tego typu jest: *Czy* ***w Wielkopolsce*** *były mecze Euro 2012?* Należy zauważyć, że w przypadku takich pytań proste metody, polegające na analizie statystycznej pokrycia słów występujących w pytaniu, mogą okazać się niewystarczające. W bazie wiedzy, z której korzysta system QA, może znajdować się przesłanka bardziej szczegółowa, na przykład zawarta w następującym zdaniu: *Drugi mecz Euro 2012 w Poznaniu: Chorwaci napędzili nam trochę strachu.*[1] W takim przypadku odpowiedź może zostać uzyskana z podanego zdania, ale tylko w przypadku gdy system QA:

- potrafi zidentyfikować nazwy *Poznań* oraz *Wielkopolska* jako nazwy odpowiednich obiektów w świecie,
- przechowuje informacje o obiektach przestrzennych świata (w tym przypadku, że *Poznań znajduje się w Wielkopolsce*),
- potrafi wywnioskować, że skoro mecz Euro 2012 odbył się w Poznaniu oraz Poznań znajduje się w Wielkopolsce, to w Wielkopolsce był mecz Euro 2012.

[1] Źródło: http://www.mmpoznan.pl/416708/2012/6/15/

Wszystkie trzy z wymienionych problemów zostały omówione w niniejszej rozprawie.

1.2. Teza i cele pracy

Tezę pracy można sformułować następująco:

Algorytm wnioskowania czasowego i przestrzennego oparty o rachunki RCC i algebrę Allena istotnie poprawia jakość odpowiedzi systemu Question Answering.

W celu zweryfikowania powyższej tezy zdefiniowano następujące zadania:

1. **Opracowanie algorytmu zbierania wiedzy przestrzennej pochodzącej z różnych źródeł.**
   Baza wiedzy przestrzennej stanowi podstawę wnioskowania przestrzennego. Dane zawarte w bazie reprezentują wiedzę o miejscach w świecie i ich wzajemnym położeniu względem siebie. Baza zawiera takie informacje jak:
   - podział administracyjny (podział na państwa, jednostki administracyjne),
   - miasta i ich położenie (np. w jakim kraju leży dane miasto),
   - obiekty interesujące (w skrócie POI od ang. *point of interest*) i ich położenie (np. w jakim mieście znajduje się dane POI).

   Istnieje kilka dostępnych baz wiedzy przestrzennej: baza Geonames[2], DBPedia[3] czy Krajowy Rejestr Urzędowy Podziału Terytorialnego Kraju TERYT[4]. Dane zawarte w wymienionych bazach są jednak albo niepełne, albo zawierają stosunkowo mało wpisów w języku polskim. Stąd istnieje potrzeba stworzenia wyczerpującej bazy wiedzy zawierającej polskie nazwy przechowywanych w bazie miejsc.
2. **Opracowanie algorytmu odpowiadania na pytania typu rozstrzygnięcia z aspektem czasowym i przestrzennym.**
   Algorytm ma wykorzystywać rachunek RCC5 do reprezentacji relacji przestrzennych oraz algebrę Allena do reprezentacji wiedzy czasowej. Rachunki tego typu są stosowane w systemach dialogowych i eksperckich np.: działający w języku angielskim system SHAKEN [UCHS2002] czy polski system rozwijany na Uniwersytecie im. Adama Mickiewicza *POLINT-112-SMS* [VMO+2010]. Zastosowanie to obejmuje jednakże głównie systemy o wyspecjalizowanej dziedzinie (ang. *closed-domain*). Proponowane rozwiązanie jest przeznaczone dla systemów o niesprecyzowanej dziedzinie (ang. *open-domain*).

[2] http://www.geonames.org
[3] http://pl.dbpedia.org
[4] http://www.stat.gov.pl/bip/36_PLK_HTML.htm

3. **Implementacja powyższych algorytmów w autorskim systemie QA.**
   Zadanie polega na implementacji powyższych algorytmów w dwóch systemach:
   a) **HipiSwot** — system zbierania wiedzy z internetowych portali informacyjnych i baz wiedzy geograficznej,
   b) **Hipisek.pl** — system odpowiadania na pytania działający w języku polskim wykorzystujący bazę wiedzy zebraną przez system HipiSwot.

   Implementacja algorytmów obejmuje opracowanie narzędzi przetwarzania języka naturalnego, a w szczególności: narzędzia do oznaczenia jednostek nazwanych w tekście polskim oraz narzędzia do pozyskiwania relacji przestrzennych i czasowych z tekstu polskiego. W implementacji wykorzystany jest pakiet PSI-Toolkit [Jas2012]. Implementacja algorytmów jest ewaluowana na zebranym korpusie pytań testowych.

1.3. Struktura pracy

Niniejsza praca składa się z pięciu rozdziałów oraz z czterech dodatków. W rozdziale 2 przedstawiono aparat matematyczny wykorzystywany w opracowanych algorytmach. Omówiono podstawowe pojęcia związane z algebrą Allena oraz rodziną rachunków RCC. Rozdział 3 zawiera opis stanu badań nad systemami odpowiadania na pytania oraz bazami wiedzy. W rozdziale omówiono rozwój dziedziny systemów odpowiadania na pytania oraz przedstawiono wybrane systemy, w których podjęto próbę zastosowania mechanizmów wnioskowania czasowego i przestrzennego. Ponadto opisano bazową wersję systemu Hipisek.pl, stworzonego przez autora niniejszej rozprawy. W systemie tym zaimplementowano algorytmy opisane w niniejszej rozprawie. W rozdziale 4 opisano metodę zbierania wiedzy przestrzennej, w której wykorzystano wnioskowanie jakościowe. Wnioskowanie zrealizowano za pomocą rachunków RCC5 oraz RCC8. Rozdział 5 zawiera opis algorytmów odpowiadania na pytania rozstrzygnięcia, w których występuje aspekt czasowy i przestrzenny. Algorytmy wykorzystują algebrę Allena do wnioskowania czasowego i rachunek RCC5 do wnioskowania przestrzennego. Rozdział 6 zawiera opis zagadnień implementacyjnych związanych z zastosowaniem opisanych w rozprawie algorytmów w systemie do zbierania bazy wiedzy HipiSwot oraz w systemie odpowiadania na pytania Hipisek.pl. Rozdział zawiera także wyniki ewaluacji opisanych w pracy algorytmów.

Dodatek A zawiera tabele złożeń rachunków używanych w pracy. Przedstawiono tabele złożeń algebry Allena, rachunku RCC5 i RCC8. W dodatku B przedstawiono dokładny opis wybranych podklas podatnych rachunków RCC5 i RCC8, które miały istotne znaczenie w analizie algorytmów wnioskowania. Dodatek C zawiera opis fragmentu taksonomii typów jednostek używanych w systemie Hipisek.pl. W dodatku D przedstawiono taksonomię typów relacji używanych w systemie Hipisek.pl.

1.4. Użyte symbole

$\top$ — relacja uniwersalna
$\bot$ — relacja pusta
A_{13} — zbiór relacji bazowych algebry Allena
2^{Allen} — zbiór wszystkich relacji (dysjunktywnych) algebry Allena
R_8 — zbiór relacji bazowych rachunku RCC8
2^{RCC8} — zbiór wszystkich relacji (dysjunktywnych) rachunku RCC8
R_5 — zbiór relacji bazowych rachunku RCC5
2^{RCC5} — zbiór wszystkich relacji (dysjunktywnych) rachunku RCC5
$\hat{H}_8$ — podklasa podatna rachunku RCC8, zdefiniowana w pracy [RN1997]
$\hat{H}_5$ — podklasa podatna rachunku RCC5, zdefiniowana w pracy [RN1997]
R' — relacja odwrotna do relacji R
$R_1 \circ R_2$ — złożenie relacji R_1 i R_2

1.5. Słownik

Algebra Allena — (and. *Allen's algebra*), inaczej algebra przedziałów Allena, jest to rachunek służący do wnioskowania czasowego, który składa się z trzynastu relacji reprezentujących wzajemne położenie dwóch przedziałów domkniętych.

Algorytm PC — w skrócie PC (od ang. *path-consistency*), heurystyczny algorytm odpowiadający na pytanie czy dana sieć ograniczeń wybranego rachunku jest sprzeczna. Idea algorytmu polega na konsekwentnym usuwaniu ograniczeń z sieci ograniczeń dla kolejnych trójek wierzchołków z wykorzystaniem operacji złożenia relacji. Dla większości rachunków w przypadku zwrócenia odpowiedzi `TRUE` (nie wykryto relacji pustej między wierzchołkami sieci ograniczeń) odpowiedź algorytmu może być niepoprawna. Skutkiem ubocznym algorytmu PC jest utworzenie wysubtelnienia sieci ograniczeń. Wynikowa sieć ograniczeń ma ten sam zbiór interpretacji, co sieć wejściowa.

Gazeter — (ang. *gazetteer*) baza obiektów geograficznych, zawierająca informacje geograficzne i geopolityczne.

Jednostka nazwana — ciągły fragment tekstu, odnoszący się do danego bytu np.: nazwa miejsca, imię i nazwisko osoby, wyrażenie czasu.

Ontologia — formalna reprezentacja wiedzy za pomocą pojęć i relacji zachodzących między pojęciami.

PC — patrz **algorytm PC**

RCC — (ang. *Region Connection Calculus*) rodzina rachunków ograniczeń wykorzystywana do modelowania relacji przestrzennych regionów. Oparty o relację

$C(x, y)$ (od ang. *connection*) interpretowaną jako relację połączenia dwóch regionów x, y. Do rodziny RCC należą na przykład rachunki: RCC3, RCC5 i RCC8.

RCC3 — jeden z rachunków RCC zawierający trzy relacje: DR (rozłączny), EQ (równy) oraz ONE (nachodzący).

RCC5 — jeden z rachunków RCC zawierający pięć relacji: DR (rozłączny), EQ (równy), PO (częściowo pokrywa się z), PP (jest podzbiorem właściwym) oraz PPI (relacja odwrotna do PP). Posiada bogatszy język od RCC3, który pozwala na modelowanie zawierania się jednego regionu w drugim.

RCC8 — jeden z rachunków RCC zawierający osiem relacji: DC (niepołączony), EQ (równy), EC (połączony brzegiem), PO (częściowo pokrywa się z), TPP (jest podzbiorem właściwym, stykającym się brzegiem), $NTPP$ (jest podzbiorem właściwym, ale niestykającym się brzegiem), oraz $TPPI$ i $NTTPI$ (relacje odwrotne odpowiednio do TPP i $NTPP$). Posiada bogatszy język od RCC5, który dodatkowo uwzględnia brzeg regionów.

Problem Podatny — problem decyzyjny, który może zostać rozwiązany na deterministycznej maszynie Turinga w wielomianowym czasie.

Sieć ograniczeń — (ang. *constraint network*) graf reprezentujący stan wiedzy na temat danego zbioru obiektów, poprzez wykorzystanie relacji wybranego rachunku ograniczeń (np.: algebry Allena, rachunków z rodziny RCC). Wierzchołki sieci ograniczeń reprezentują obiekty, natomiast krawędzie etykietowane są relacją jaka zachodzi między danymi dwoma obiektami. W niniejszej pracy rozpatrujemy grafy pełne. W przypadku braku wiedzy na temat relacji zachodzącej między danymi wierzchołkami występującymi w sieci ograniczeń przyjmujemy, że zachodzi między nimi relacja uniwersalna. Ponadto jeśli między wierzchołkami i oraz j zachodzi dana relacja, to między j oraz i zachodzi jej relacja odwrotna.

System QA o niesprecyzowanej dziedzinie — system QA odpowiadający na pytania z tematyki ogólnej (w praktyce nieograniczonej).

System QA o wyspecjalizowanej dziedzinie — system QA działający w ramach ograniczonej tematyki pytań.

Token — łańcuch znakowy, składający się ze zgrupowanych symboli tekstu wejściowego. Kryterium grupowania zależy od danego zagadnienia przetwarzania. Tokenami mogą być na przykład: wyrazy, liczby, znaki interpunkcyjne, jednostki nazwane (np. adresy internetowe).

Tokenizacja — podział tekstu na tokeny.

Wydobywanie Informacji — (ang. *Information Extraction*, IE) zadanie pozyskiwania danych ustrukturyzowanych z danych nieustrukturyzowanych (np. z tekstu zapisanego w języku naturalnym).

Wyszukiwanie Informacji — (ang. *Information Retrieval*, IR) zadanie odnalezienia materiału tekstowego w obszernej kolekcji dokumentów, który pasuje do za-

pytania składającego się ze zbioru słów kluczowych. Zadanie IR realizują m.in. wyszukiwarki internetowe (np. Google, Bing).

Rozdział 2

Rachunki czasowe i przestrzenne

W niniejszym rozdziale opisuję dwa formalizmy służące do reprezentacji wiedzy czasowej i przestrzennej. Są to:

- algebra Allena,
- rodzina rachunków RCC.

2.1. Algebra Allena

Algebra Allena została wprowadzona przez Jamesa F. Allena w 1983 r. [All1983] Była to jedna z pierwszych prób formalizacji wnioskowania jakościowego.

Rozpatrzmy następujący zestaw zdań w języku naturalnym (w niniejszym podrozdziale nazywany **przykładem Waldemara**):[1]

- Waldemara nie było w pokoju, kiedy wskutek przepięcia światło zgasło.
- Ale kiedy światło znowu się zapaliło, Waldemar był już w pokoju.

Powyższe zdania odnoszą się do trzech różnych wydarzeń: krótkiego okresu awarii, czasu, w którym Waldemar jest w pokoju oraz okresu, w którym światło jest zgaszone.

Zgodnie z sugestią Allena czas reprezentowany jest przez przedziały na prostej rzeczywistej. Na potrzeby rozprawy przyjmiemy za [Lig2012] następującą definicję przedziału czasowego:

Definicja 2.1. Przedziałem czasowym (w sensie Allena) nazywamy parę liczb rzeczywistych t_1, t_2 takich że $t_1 < t_2$.

Stąd wydarzenia z przykładu Waldemara możemy reprezentować za pomocą trzech przedziałów:

- p — krótki przedział czasowy, w którym zdarzyło się przepięcie,
- w — przedział czasowy, w którym Waldemar jest w pokoju,
- z — przedział czasowy, w którym światło jest zgaszone.

[1] Przykład wzorowany jest na jednym z przykładów z pracy [All1983] zamieszczonym w książce [Lig2012].

Tabela 2.1. Formalna definicja relacji bazowych algebry Allena

$x\ P\ y$	$y\ PI\ x$	$x_1 < x_2 < y_1 < y_2$
$x\ M\ y$	$y\ MI\ x$	$x_1 < x_2 = y_1 < y_2$
$x\ O\ y$	$y\ OI\ x$	$x_1 < y_1 < x_2 < y_2$
$x\ S\ y$	$y\ SI\ x$	$x_1 = y_1 < x_2 < y_2$
$x\ D\ y$	$y\ DI\ x$	$y_1 < x_1 < x_2 < y_2$
$x\ F\ y$	$y\ FI\ x$	$y_1 < x_1 < x_2 = y_2$
$x\ EQ\ y$	$y\ EQ\ x$	$x_1 = y_1 \wedge x_2 = y_2$

2.1.1. Relacje bazowe

Allen [All1983] definiuje trzynaście relacji, jakie mogą zachodzić między dwoma przedziałami czasowymi x oraz y. Nazwy trzynastu relacji zdefiniowanych przez Allena to:

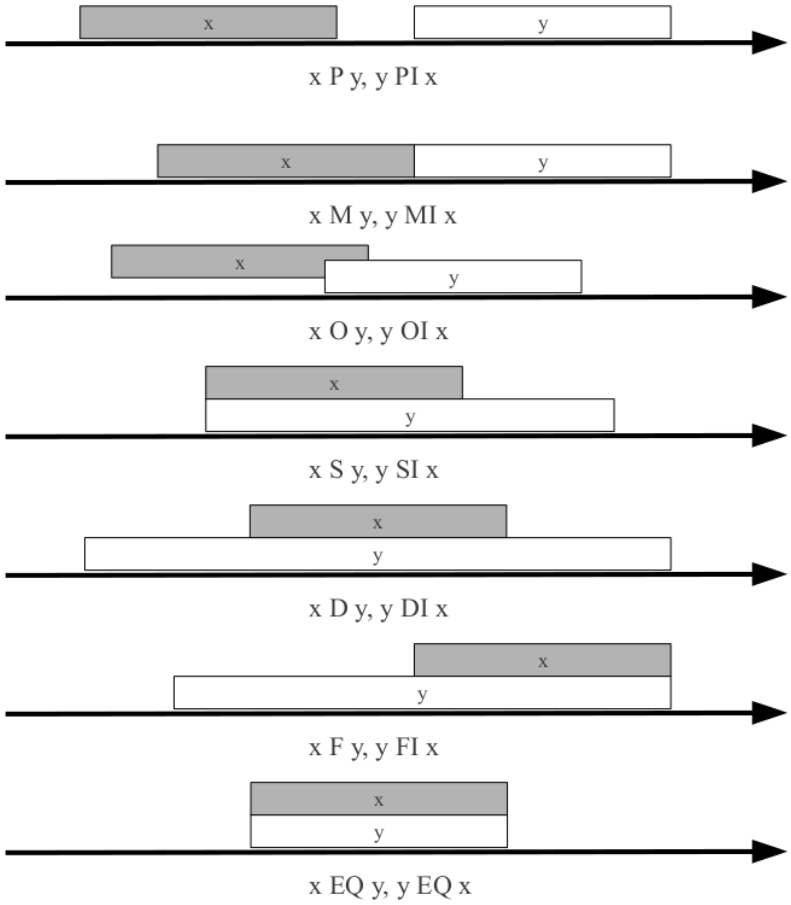
- x P y — x poprzedza y (ang. *precedes*),
- x PI y — relacja odwrotna do P,
- x M y — x styka się z y (ang. *meets*),
- x MI y — relacja odwrotna do M,
- x O y — x nachodzi na y (ang. *overlaps*),
- x OI y — relacja odwrotna do O,
- x S y — x zaczyna się równo z y (ang. *starts*),
- x SI y — relacja odwrotna do S,
- x D y — x trwa podczas y (ang. *is during*),
- x DI y — relacja odwrotna do D,
- x F y — x kończy się równo z y (ang. *finishes*),
- x FI y — relacja odwrotna do F,
- x EQ y — x równe y (ang. *equals*).

Uwaga 2.1. Dla zachowania spójności z rachunkami z rodziny RCC (patrz podrozdział 2.2), relacje algebry Allena zapisane zostały wielkimi literami. W literaturze relacje te oznaczane są zwykle małymi literami.

Interpretacje relacji na prostej rzeczywistej zostały przedstawione na rysunku 2.1.

Trzynaście relacji Allena definiujemy porównując ułożenie początków i końców dwóch przedziałów. Formalne definicje relacji (za [Lig2012]) zachodzących między dwoma przedziałami czasowymi $x = (x_1, x_2)$ oraz $y = (y_1, y_2)$ zostały przedstawione w tabeli 2.1. Trzynaście zdefiniowanych relacji nazywamy **relacjami bazowymi**. Zbiór relacji bazowych algebry Allena oznaczamy symbolem A_{13}.

Ważną własnością relacji ze zbioru A_{13} jest własność JEPD [Lig2012].



Rysunek 2.1. Relacje algebry Allena przestawione na prostej rzeczywistej

Definicja 2.2. JEPD (ang. *Jointly Exhaustive Pairwise Disjoint*) jest to własność zbioru relacji U zdefiniowanych na zbiorze Z, polegająca na tym, że każde dwa obiekty należące do Z są w jednej z relacji z U (JE, zbiór relacji U jest wyczerpujący) oraz każde dwie relacje należące do U są rozłączne (PD, parami rozłączne).

Własność JEPD dla zbioru A_{13} wynika bezpośrednio z formalnych definicji relacji bazowych [Lig2012].

Konsekwencjami własności JEPD są:

- dla dowolnych dwóch przedziałów czasowych x i y musi zachodzić jedna z relacji bazowych,
- dla dowolnych przedziałów czasowych x i y:

$$\forall_{R \in A_{13}, S \in A_{13}} [xRy \wedge xSy \Rightarrow R = S]$$

Rozpatrując przykład Waldemara należy zauważyć, że relacje bazowe okazują się niewystarczające do reprezentacji informacji zawartej w przykładowych zdaniach. Na przykład zdanie pierwsze z przykładu Waldemara oznacza, że między przedzia-

łami czasowymi w (*Waldemar jest w pokoju*) oraz p (*przepięcie prądu*) zachodzi jedna z relacji: P, PI, M, MI. W tym celu (za [Lig2012]) wprowadzamy pojęcie **relacji dysjunktywnej**:

Definicja 2.3. Relacją dysjunktywną (w skrócie **relacją**) nazywamy dowolny podzbiór zbioru relacji bazowych.

Uwaga 2.2. W niniejszej rozprawie wszystkie relacje dysjunktywne rozpatrywanych rachunków nazywane są relacjami. Relacje składające się z jednej relacji bazowej są z nią utożsamiane, dlatego w ich zapisie pomijamy nawiasy klamrowe (zapisując np. P zamiast $\{P\}$).

Zbiór wszystkich relacji algebry Allena oznaczamy symbolem 2^{Allen}. Spośród wszystkich relacji ze zbioru 2^{Allen} wyróżniamy dwie relacje:

- relację uniwersalną $\top$ — zbiór wszystkich relacji bazowych,
- relację pustą $\bot$ — zbiór pusty.

Korzystając z relacji ze zbioru 2^{Allen} możemy reprezentować relacje między przedziałami p, w, z z przykładu Waldemara w następujący sposób:

- w $\{P, PI, M, MI\}$ p (*Waldemara nie było w czasie przepięcia*),
- p $\{O, M\}$ z (*światło zgasło, bo wystąpiło przepięcie*),
- z $\{O, S, D\}$ w (*Waldemar był już w pokoju, kiedy światło znów się zapaliło*).

2.1.2. Sieć ograniczeń

Do reprezentowania wiedzy czasowej wykorzystujemy **sieć ograniczeń** (ang. *constraint network*). Za [Lig2012] przyjmujemy następującą definicję sieci ograniczeń:

Definicja 2.4. Siecią ograniczeń nazywamy parę (N, C), gdzie N jest skończonym zbiorem wierzchołków, natomiast $C : N \times N \to 2^{Allen}$ jest przyporządkowaniem każdej parze obiektów (i, j) z N elementu $C(i, j)$ należącego do 2^{Allen}. Przyporządkowany element $C(i, j)$ nazywamy **ograniczeniem** (ang. *constraint*).

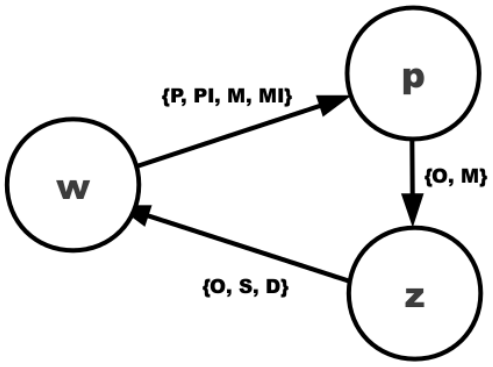
Przykład sieci ograniczeń reprezentującej wydarzenia z przykładu Waldemara przedstawia rysunek 2.2.

W pracy rozważamy sieci ograniczeń znormalizowane (w sensie Ligozata), w których zachodzi:

- $C(i, i) = \{EQ\}$ dla każdego wierzchołka i,
- $C(i, j) = C(j, i)'$ dla każdej pary wierzchołków i oraz j.[2]

Uwaga 2.3. Na rysunkach reprezentujących sieć ograniczeń, pomijamy krawędzie etykietowane relacją uniwersalną.

[2] Apostrofem oznaczamy relację odwrotną (patrz podrozdział 2.1.3.)



Rysunek 2.2. Sieć ograniczeń reprezentująca przykład Waldemara

2.1.3. Operacje na relacjach

Na zbiorze relacji 2^{Allen} definiujemy cztery operacje:

- $R \cup S$ — suma relacji R i S,
- $R \cap S$ — iloczyn relacji R i S,
- R' — odwrotność relacji R,
- $R \circ S$ — złożenie relacji R i S.

Operacje sumy i iloczynu są zwykłymi operacjami teoriomnogościowymi na zbiorach. W dalszej części podrozdziału omawiamy operacje odwrotności i złożenia.

Odwrotność relacji

Każda relacja bazowa algebry Allena ma z definicji relację odwrotną (przy czym relacją odwrotną do relacji równości EQ jest ta sama relacja). Dla relacji bazowej $r \in A_{13}$ przez r' oznaczamy odpowiadającą jej relację odwrotną.

Korzystając z tego faktu definiujemy relację odwrotną w następujący sposób:

Definicja 2.5. Relacją odwrotną do relacji $R \in 2^{Allen}$ jest zbiór relacji:

$$R' = \bigcup_{r \in R} \{r'\}$$

Korzystając z operacji odwrotności możemy wydedukować w przykładzie Waldemara, że skoro $z\{O, S, D\}w$, to $w\{OI, SI, DI\}z$.

Złożenie relacji

Operację złożenia dla relacji bazowych definiujemy w następujący sposób:

Definicja 2.6. Złożeniem relacji bazowych $r, s \in A_{13}$ jest zbiór:

$$r \circ s = \{(x, y) \in U \times U : \exists_{z \in U}(x\ r\ z) \wedge (z\ s\ y)\}$$

gdzie U oznacza uniwersum wszystkich przedziałów czasowych.

Złożenie relacji bazowych wyrażane jest za pomocą tak zwanej **tablicy złożeń** [Lig2012] [All1983]. Tablica złożeń dla algebry Allena przedstawiona została w dodatku A.

Dla relacji (dysjunktywnych) algebry Allena złożenie definiujemy w następujący sposób (za [Lig2012]):

Definicja 2.7. Złożenie dwóch relacji $R, S \in 2^{Allen}$ jest sumą zbiorów będących złożeniem relacji bazowych $r \circ s$, gdzie $r \in R, s \in S$:

$$R \circ S = \bigcup_{r \in R, s \in S} [r \circ s]$$

Złożenie relacji pozwala na wydedukowanie informacji na temat relacji zachodzących między dwoma przedziałami. Rozpatrzmy dla przykładu dwie relacje z przykładu Waldemara: $p\{O, M\}z$ oraz $z\{O, S, D\}w$. Złożenie tych relacji jest równe:

$$\{O, M\} \circ \{O, S, D\} = O \circ O \cup O \circ S \cup O \circ D \cup M \circ O \cup M \circ S \cup M \circ D$$

Z tablicy złożeń mamy:

$$O \circ O = \{P, M, O\}$$

$$O \circ S = \{O\}$$

$$O \circ D = \{O, S, D\}$$

$$M \circ O = \{P\}$$

$$M \circ S = \{M\}$$

$$M \circ D = \{O, S, D\}$$

Co daje nam:

$$\{O, M\} \circ \{O, S, D\} = \{P, M, O, S, D\}$$

Stąd korzystając ze złożenia relacji możemy wywnioskować, że między zdarzeniami p oraz w zachodzi jedna z relacji ze zbioru $\{P, M, O, S, D\}$.

2.1.4. Niesprzeczność

Definicja 2.8. Konfiguracją nazywamy skończony zbiór przedziałów czasowych.

Konfiguracja jest pojęciem wyrażającym konkretne ułożenie przedziałów czasowych. Mając daną konfigurację możemy przyporządkować jej sieć ograniczeń.

Przykładowa konfiguracja dla sieci ograniczeń zaprezentowanej na rysunku 2.2 (dla przykładu Waldemara) została przedstawiona na rysunku 2.3. Zaprezentowana

konfiguracja wyraża sytuację, w której podczas przepięcia światło gaśnie, a Waldemar przychodzi do pokoju kiedy światło jest ciągle zgaszone.

Rysunek 2.3. Przykład konfiguracji spełniającej ograniczenia sieci ograniczeń z przykładu Waldemara

Zauważmy, że konkretna konfiguracja pozwala na zdefiniowanie relacji bazowej zachodzącej między przedziałami. Dla konfiguracji z rysunku 2.3 zachodzi: $p\ O\ z$, $z\ O\ w$, $w\ PI\ p$. Wszystkie z wymienionych relacji zawierają się w ograniczeniach sieci ograniczeń z rysunku 2.2. W takiej sytuacji mówimy, że konfiguracja spełnia wszystkie ograniczenia sieci ograniczeń.

Problem niesprzeczności danej sieci ograniczeń dotyczy przyporządkowania odwrotnego. Pytanie brzmi: *Czy mając daną sieć ograniczeń N można znaleźć konfigurację, która spełnia wszystkie ograniczenia N?* Dla przykładu Waldemara sieć ograniczeń z rysunku 2.2 jest niesprzeczna, ponieważ udało się znaleźć konfigurację, która spełnia wszystkie ograniczenia.

Za [Lig2012] przyjmujemy następującą definicję:

Definicja 2.9. Sieć ograniczeń N jest **niesprzeczna** jeżeli istnieje konfiguracja, która spełnia wszystkie ograniczenia N.

Vilain et al. udowodnili w pracy [VKvB1990] następujące twierdzenie:[3]

Twierdzenie 2.1. *Rozstrzygnięcie, czy dana sieć ograniczeń algebry Allena jest niesprzeczna, należy do klasy problemów NP-zupełnych.*

Konsekwencją twierdzenia (przy założeniu, że $P \neq NP$) jest brak efektywnych algorytmów rozwiązujących problem niesprzeczności algebry Allena. W praktyce najczęściej wykorzystywanymi algorytmami heurystycznymi (przybliżającymi rozwiązanie) są algorytmy z grupy PC (ang. *path-consistent*).

2.1.5. Algorytm PC

Algorytm PC jest adaptacją algorytmów wykorzystywanych w dziedzinie CSP (ang. *Constraint Satisfaction Problem*, problem spełnienia warunków) na dziedzinę rachunków ograniczeń (w szczególności algebry Allena).

Za [Lig2012] przyjmujemy następującą definicję:

[3] W pracy podano sformułowanie twierdzenia dostosowane do wykorzystywanej terminologii.

Algorytm 2.1: Revise

```
Data: Sieć ograniczeń (N, C), trzy wierzchołki sieci ograniczeń (i, j, k)
Result: true jeśli ograniczenie C(i, j) zostało zmienione, wpp. false
1  begin
2      if C(i, k) = ⊤ ∨ C(k, j) = ⊤ then
3          return false
4      end
5      S ← ⊥ ;
6      foreach a ∈ C(i, k) do
7          foreach b ∈ C(k, j) do
8              S ← S ∪ (a ∘ b) ;
9          end
10     end
11     if C(i, j) ⊆ S then
12         return false
13     end
14     C(i, j) ← (C(i, j) ∩ S) ;
15     C(j, i) ← C(i, j)′;
16     return true
17 end
```

Definicja 2.10. Sieć ograniczeń (N, C) nazywana jest **ścieżkowo niesprzeczną** (ang. *path-consistent*, w skrócie PC) jeżeli dla każdej trójki wierzchołków $i, j, k \in N$ zachodzi:

$$C(i, j) \subseteq (C(i, k) \circ C(k, j))$$

Idea algorytmu PC polega na sukcesywnym stosowaniu złożenia relacji dla trójek wierzchołków z sieci ograniczeń, dopóki w pojedynczym przebiegu (wszystkich wierzchołków sieci) złożenie nie wprowadza żadnych zmian w sieci ograniczeń. Wynikiem działania algorytmu jest sieć ograniczeń, która jest ścieżkowo niesprzeczna. W pracy podajemy algorytm za książką [Lig2012]. Jest to wersja algorytmu opracowana przez Dechter et al. [DMP1991]. W pracy tej pokazano, że jest to algorytm wielomianowy.

Głównym elementem algorytmu PC jest funkcja *Revise* przedstawiona na listingu 2.1. Zadaniem tej funkcji jest obliczenie złożenia relacji między wierzchołkami (i, k) praz (k, j). Ponadto funkcja ta przyporządkowuje ograniczeniu $C(i, j)$ relację będącą iloczynem złożenia $C(i, k) \circ C(k, j)$ (wynik tego iloczynu przechowywany jest w zmiennej pomocniczej S) oraz poprzednio zapisanej wartości $C(i, j)$. Jeżeli ograniczenie zmienia swoją wartość, funkcja ta zwraca wartość logiczną *prawda*. W przeciwnym przypadku (ograniczenie $C(i, j)$ pozostało bez zmian) funkcja zwraca wartość logiczną *fałsz*.

Algorytm PC został przedstawiony na listingu 2.2. W głównej pętli algorytmu PC rozpatrywane są wszystkie trójki wierzchołków przetwarzanej sieci ograniczeń. Dla

każdej trójki wierzchołków uruchamiana jest funkcja *Revise*. Następnie sprawdzane jest, czy funkcja zmieniła ograniczenie sieci. Jeśli tak, to sprawdzane jest czy otrzymano ograniczenie będące relacją pustą $\perp$. W takim przypadku algorytm kończy się (otrzymanie relacji pustej oznacza wykrycie sprzeczności). Główna pętla algorytmu kończy się, gdy sieć ograniczeń jest stabilna (w pojedynczym przebiegu nie zmieniono żadnego z ograniczeń).

Algorytm 2.2: Algorytm PC

```
   Data: Sieć ograniczeń (N, C) mająca n wierzchołków
 1 begin
 2     change ← false ;
 3     repeat
 4         change ← false ;
 5         forall the (k, i, j) ← 1 ... n do
 6             if (i, j, k) są parami różne then
 7                 if Revise(i,k,j) then
 8                     if C(i, j) = ⊥ then
 9                         return
10                     end
11                     change ← true
12                 end
13             end
14         end
15     until change = false;
16 end
```

Wynikiem działania algorytmu PC jest powstanie wysubtelnienia sieci ograniczeń (sieci wysubtelnionej) [Lig2012]:

Definicja 2.11. Sieć ograniczeń (N, C') nazywamy **wysubtelnioną siecią ograniczeń** (wysubtelnieniem, ang. *refinement*) sieci ograniczeń (N, C), jeżeli dla każdej pary wierzchołków $(i, j) \in N \times N$ zachodzi $C'(i, j) \subseteq C(i, j)$.

Wysubtelniona sieć ograniczeń powstała w wyniku działania algorytmu PC jest spełniona przez ten sam zbiór konfiguracji spełniający wejściową sieć ograniczeń. Przyjmujemy, że jeśli algorytm PC wyprodukuje wysubtelnienie, które zawiera etykietowanie relacją pustą, to wejściowa sieć ograniczeń jest sprzeczna. W przeciwnym przypadku przyjmujemy, że wejściowa sieć ograniczeń jest niesprzeczna.

W [All1983] pokazano, że algorytm PC nie generuje nowej sprzeczności. Oznacza, to że jeśli algorytm PC zgłosi sprzeczność, to wynik ten jest poprawny. W ogólności jednak, jeśli algorytm PC zgłosi że sieć jest niesprzeczna, to nadal możliwe jest że sieć jest sprzeczna. Kontrprzykład pochodzący od Kautza podany został przez Allena w pracy [All1983]. Stąd algorytm PC jest tylko częściową procedurą rozstrzygania niesprzeczności danej sieci.

Istnieją podklasy algebry Allena, w których algorytm PC zawsze zgłasza odpowiedź poprawną. Ponieważ algorytm ten działa w czasie wielomianowym, podklasy te są podklasami podatnymi. [4]

2.2. Rodzina rachunków RCC

Rachunki z rodziny RCC (ang. *Region Connection Calculus*) są historycznie jedną z pierwszych prób formalizacji jakościowej wiedzy przestrzennej, opartą o prace Allena. Zostały opracowane niezależnie przez Randella, Cui i Cohna [RCC1992] oraz przez Egenhofera i jego współpracowników [Ege1989] [Ege1991]. W pracy rozpatrujemy podejście podane przez Randella.

2.2.1. Definicja relacji

Obiekty przestrzenne (nazywane dalej regionami) są reprezentowane w rachunku RCC jako podzbiory pewnej przestrzeni topologicznej U. Rachunki z rodziny RCC są oparte o pojedynczą relację C (relację połączenia, od ang. *connection*). Typowa interpretacja relacji $C(a, b)$ jest następująca: regiony a oraz b są w relacji C wtw gdy domknięcia a oraz b mają punkt wspólny (za [Ren2002]).

Relacja C powinna być zwrotna i symetryczna, co wyrażają dwa następujące aksjomaty:

$$\forall_x C(x, x)$$

$$\forall_{x,y}[C(x, y) \Rightarrow C(y, x)]$$

W pracy [RCC1992] za pomocą relacji $C(x, y)$ zdefiniowano następujące relacje (w nawiasach podano zamierzoną przez autorów pracy interpretację definiowanej relacji):

- $DC(x, y)$ (ang. *disconnected*, x i y nie są połączone),
- $P(x, y)$ (ang. *part*, x jest częścią y),
- $PP(x, y)$ (ang. *proper part*, x jest podzbiorem właściwym y),
- $EQ(x, y)$ (ang. *equals*, x jest równe y),
- $O(x, y)$ (ang. *overlaps*, x pokrywa się z y),
- $PO(x, y)$ (ang. *partially overlaps*, x częściowo pokrywa się z y),
- $DR(x, y)$ (ang. *discrete*, x jest oddzielne od y),
- $EC(x, y)$ (ang. *externally connected*, x jest połączone brzegiem z y),

[4] Podklasy podatne algebry Allena nie zostały wykorzystane w autorskich algorytmach wnioskowania, dlatego nie zostały omówione. W podrozdziale 2.2.6 omówiono podklasy podatne rachunków z rodziny RCC, które zostały wykorzystane w opracowanych metodach wnioskowania.

- $TPP(x, y)$ (ang. *tangential proper part*, x jest stycznym podzbiorem właściwym y),
- $NTPP(x, y)$ (ang. *non-tangential proper part*, x jest niestycznym podzbiorem właściwym y),
- $PI(x, y)$ (relacja odwrotna do P),
- $PPI(x, y)$ (relacja odwrotna do PP),
- $TPPI(x, y)$ (relacja odwrotna do TPP),
- $NTPPI(x, y)$ (relacja odwrotna do $NTPP$).

Poniżej znajduje się formalna definicja wymienionych relacji (za [Ren2002]):

$$DC(x, y) \equiv_{def} \neg C(x, y)$$

$$P(x, y) \equiv_{def} \forall_z [C(z, x) \Rightarrow C(z, y)]$$

$$PP(x, y) \equiv_{def} P(x, y) \wedge \neg P(y, x)$$

$$EQ(x, y) \equiv_{def} P(x, y) \wedge P(y, x)$$

$$O(x, y) \equiv_{def} \exists_z [P(z, x) \wedge P(z, y)]$$

$$PO(x, y) \equiv_{def} O(x, y) \wedge \neg P(x, y) \wedge \neg P(y, x)$$

$$DR(x, y) \equiv_{def} \neg O(x, y)$$

$$EC(x, y) \equiv_{def} C(x, y) \wedge \neg O(x, y)$$

$$TPP(x, y) \equiv_{def} PP(x, y) \wedge \exists_z [EC(z, x) \wedge EC(z, y)]$$

$$NTPP(x, y) \equiv_{def} PP(x, y) \wedge \neg \exists_z [EC(z, x) \wedge EC(z, y)]$$

$$PI(x, y) \equiv_{def} P(y, x)$$

$$PPI(x, y) \equiv_{def} PP(y, x)$$

$$TPPI(x, y) \equiv_{def} TPP(y, x)$$

$$NTPPI(x, y) \equiv_{def} NTPP(y, x)$$

W pracy [RCC1992] wyróżniono osiem z wyżej wymienionych relacji, które tworzą rachunek RCC8. Bennett zaproponował w pracy [Ben1994] użycie tylko pięciu relacji, tworząc rachunek RCC5.

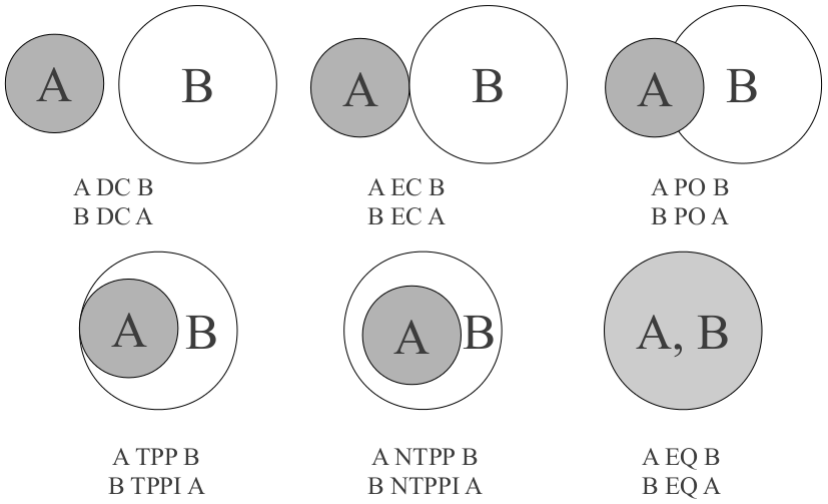
2.2.2. RCC8

Rachunek RCC8 oparty jest o zbiór ośmiu relacji:

$$R_8 = \{DC, EC, PO, EQ, TPP, TPPI, NTPP, NTPPI\}$$

Relacje wchodzące w skład zbioru R_8 są **relacjami bazowymi** rachunku RCC8. Podobnie jak w algebrze Allena zbiór relacji R_8 ma własność JEPD [Lig2012] (patrz podrozdział 2.1.1). Zbiór wszystkich relacji (dysjunktywnych) rachunku RCC8 oznaczamy symbolem 2^{RCC8}.

Rysunek 2.4 przedstawia zamierzoną interpretację relacji bazowych rachunku RCC8.



Rysunek 2.4. Relacje rachunku RCC8 przedstawione na płaszczyźnie rzeczywistej

2.2.3. RCC5

Rachunek RCC5 oparty jest o zbiór pięciu relacji:

$$R_5 = \{DR, PO, EQ, PP, PPI\}$$

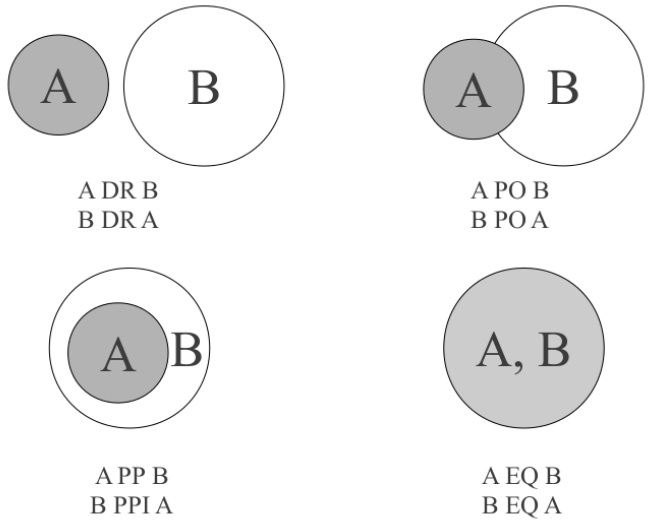
Relacje wchodzące w skład zbioru R_5 są **relacjami bazowymi** rachunku RCC5. Relacje bazowe spełniają własność JEPD. Zbiór wszystkich relacji (dysjunktywnych) rachunku RCC5 oznaczamy symbolem 2^{RCC5}.

Rysunek 2.5 przedstawia zamierzoną interpretację relacji bazowych rachunku RCC5.

Podstawową różnicą między rachunkami RCC5 oraz RCC8 jest brak rozróżnienia brzegów w rachunku RCC5. W rachunku tym można reprezentować jedynie relacje wyrażające zawieranie się regionów (lub ich części).

Należy zauważyć, że rachunek RCC5 jest podklasą rachunku RCC8 [Ben1994]:

- relacja DR rachunku RCC5 jest równoważna relacji $\{DC, EC\}$ rachunku RCC8,
- relacja PP rachunku RCC5 jest równoważna relacji $\{TPP, NTPP\}$ rachunku RCC8,



Rysunek 2.5. Relacje rachunku RCC5 przedstawione na płaszczyźnie rzeczywistej

- relacja PPI rachunku RCC5 jest równoważna relacji $\{TPPI, NTPPI\}$ rachunku RCC8,

2.2.4. Operacje na relacjach

Operacje sumy, iloczynu, odwrotności oraz złożenia definiujemy analogicznie jak dla algebry Allena (patrz podrozdział 2.1.3).

W przypadku relacji odwrotnych dla relacji bazowych zachodzi:

- W rachunku RCC8
  - dla relacji bazowych TPP, $NTPP$ relacją odwrotną są odpowiednio $TPPI$ oraz $NTPPI$,
  - dla relacji bazowych EQ, DC, EC, PO relacją odwrotną jest ta sama relacja.
- W rachunku RCC5
  - dla relacji bazowej PP relacją odwrotną jest PPI,
  - dla relacji bazowych EQ, DR, PO relacją odwrotną jest ta sama relacja.

Złożenie relacji obliczane jest za pomocą tablicy złożeń. Tablice złożeń dla rachunków RCC8 oraz RCC5 podane za pracą [Ren2002] znajdują się w dodatku A.

2.2.5. Niesprzeczność

Sieć ograniczeń rachunków RCC8 i RCC5 definiujemy podobnie jak dla algebry Allena. (patrz podrozdział 2.1.2). Definiujemy także pojęcie konfiguracji dla rachunków RCC8 i RCC5 (dla odróżnienia nazywane na potrzeby pracy konfiguracją przestrzenną).

Definicja 2.12. Konfiguracją przestrzenną nazywamy dowolny skończony zbiór regionów.

Za pracą [RN1997] ograniczymy się do regionów interpretowanych jako dziedziny domknięte pewnej przestrzeni topologicznej.

Korzystając z pojęcia konfiguracji problem niesprzeczności dla rachunku RCC5 i RCC8 definiujemy jako:

Definicja 2.13. Sieć ograniczeń N (rachunku RCC5 lub RCC8) jest **niesprzeczna** jeżeli istnieje konfiguracja przestrzenna, która spełnia wszystkie ograniczenia N.

Renz i Nebel w pracy [RN1997] udowodnili następujące twierdzenie:

Twierdzenie 2.2. *Rozstrzygnięcie, czy dana sieć ograniczeń rachunku RCC5 jest niesprzeczna, należy do klasy problemów NP-zupełnych.*

Renz i Nebel wykorzystują fakt, że rachunek RCC5 jest podklasą rachunku RCC8 otrzymując następujący wniosek:

Wniosek 2.3. *Rozstrzygnięcie, czy dana sieć ograniczeń rachunku RCC8 jest niesprzeczna, należy do klasy problemów NP-zupełnych.*

Podobnie jak w przypadku algebry Allena podstawowym algorytmem heurystycznym rozstrzygania niesprzeczności rachunków RCC8 i RCC5 jest algorytm PC. W dalszej części podrozdziału (podrozdział 2.2.6) rozpatrujemy podklasy rachunków RCC8 i RCC5, w których algorytm PC daje zawsze poprawną odpowiedź. Podklasy te zostały opisane przez Renza i Nebela w pracy [RN1997],

2.2.6. Podklasy podatne rachunków RCC8 i RCC5

W celu zidentyfikowania podklasy podatnej rachunku RCC8 Renz i Nebel analizowali, które relacje rachunku RCC8 mogą być reprezentowane za pomocą klauzuli Horna [RN1997]. Zbiór takich relacji oznaczyli symbolem H_8. Korzystając z faktu, że problem spełnialności klauzuli Horna (HORNSAT) jest podatny, Renz i Nebel udowadniają następujące twierdzenie [RN1997]:

Twierdzenie 2.4. *Problem rozstrzygnięcia, czy dana sieć ograniczeń podklasy rachunku RCC8 H_8 jest niesprzeczna (problem $RSAT(H_8)$), można wielomianowo zredukować do problemu HORNSAT, stąd $RSAT(H_8) \in P$.*

Następnie pokazali, że domknięcie takiego zbioru ze względu na operacje iloczynu, odwrotności i złożenia relacji również jest podklasą podatną. Zbiór ten oznaczono przez $\hat{H}_8$. Prowadzi to do następującego twierdzenia:

Twierdzenie 2.5. *Problem rozstrzygnięcia, czy dana sieć ograniczeń podklasy rachunku RCC8 $\hat{H}_8$ jest niesprzeczna, jest problemem podatnym, czyli $RSAT(\hat{H}_8) \in P$.*

W pracy [RN1997] obliczono, jakie relacje należą do zbioru $\hat{H}_8$, udowadniając następujące twierdzenie:

Twierdzenie 2.6. *$\hat{H}_8$ zawiera następujące 148 relacji:*

$$\hat{H}_8 = 2^{RCC8} \setminus (N_1 \cup N_2 \cup N_3)$$

gdzie:

$$\begin{aligned} N_1 =& \{R \in 2^{RCC8} : \{PO\} \nsubseteq R \wedge \\ & (\{TPP, TPPI\} \subseteq R \vee \{NTPP, NTPPI\} \subseteq R)\} \end{aligned}$$

$$\begin{aligned} N_2 =& \{R \in 2^{RCC8} : \{PO\} \nsubseteq R \wedge \\ & (\{TPP, NTPPI\} \subseteq R \vee \{TPPI, NTPP\} \subseteq R)\} \end{aligned}$$

$$\begin{aligned} N_3 =& \{R \in 2^{RCC8} : \{EQ\} \subseteq R \wedge \\ & ((\{NTPP\} \subseteq R \wedge \{TPP\} \nsubseteq R) \vee (\{NTPPI\} \subseteq R \wedge \{TPPI\} \nsubseteq R))\} \end{aligned}$$

Zbiór 148 relacji należących do zbioru $\hat{H}_8$ przedstawiony został *explicite* w dodatku B. Warto zauważyć, że zbiór $\hat{H}_8$ zawiera wszystkie relacje bazowe rachunku RCC8 oraz relację uniwersalną.

Korzystając z faktu, że rachunek RCC5 jest podklasą rachunku RCC8, Renz i Nebel definiują podklasę $\hat{H}_5$ jako przecięcie klasy $\hat{H}_8$ oraz zbioru relacji rachunku RCC5. W pracy [JD1997] podano 28 relacji należących do podklasy $\hat{H}_5$.

Zbiór 28 relacji należących do zbioru $\hat{H}_5$ przedstawiony został *explicite* w dodatku B. Podobnie jak w przypadku klasy $\hat{H}_8$, zbiór $\hat{H}_5$ zawiera wszystkie relacje bazowe rachunku RCC5 oraz relację uniwersalną.

Ponadto Renz i Nebel pokazali w pracy [RN1997] następujące twierdzenie:

Twierdzenie 2.7. *Dla podklasy $\hat{H}_8$ rachunku RCC8 metoda ścieżkowej niesprzeczności (algorytm PC) rozstrzyga problem niesprzeczności tej sieci.*

Oznacza to, że wejściowa sieć ograniczeń (która trafia do algorytmu PC), etykietowana relacjami ze zbioru $\hat{H}_8$ jest niesprzeczna wtedy i tylko wtedy gdy algorytm PC nie wyprodukuje relacji pustej.

Prostym wnioskiem z twierdzenia 2.7 jest analogiczna własność podklasy $\hat{H}_5$:

Wniosek 2.8. *Dla podklasy $\hat{H}_5$ rachunku RCC5 metoda ścieżkowej niesprzeczności (algorytm PC) rozstrzyga problem niesprzeczności tej sieci.*

Rozdział 3

Systemy QA

Niniejszy rozdział jest wprowadzeniem w tematykę systemów QA ze szczególnym uwzględnieniem zagadnień wnioskowania przestrzennego i czasowego.

W pierwszym podrozdziale wyjaśniam, na czym polega zadanie systemów QA. Następnie przechodzę do przedstawienia historii rozwoju dziedziny, które kończy opis obecnego stanu wiedzy. Później przedstawiam wybrane sposoby podziału systemów QA, po czym omawiam główne problemy badawcze.

W drugiej części rozdziału opisuję bazy wiedzy, które są jednym z podstawowych zasobów wykorzystywanych w systemach QA. Przedstawiam różne typy baz wiedzy i metody ich pozyskiwania. Ze względu na tematykę pracy ograniczam się do baz wiedzy przestrzennej. Następnie omawiam sposoby reprezentacji wiedzy przestrzennej i czasowej za pomocą standardów: TimeML oraz SpatialML.

Rozdział kończy opis autorskiego systemu QA. System Hipisek.pl jest prototypem polskiego systemu QA opracowanym przeze mnie w pracy magisterskiej. W procesie rozwoju systemu zaimplementowano algorytmy opisane w niniejszej rozprawie doktorskiej. W tym rozdziale ograniczam się do opisania bazowej wersji systemu (sprzed dokonania usprawnień).

3.1. Opis systemów QA

Zadaniem systemów QA (ang. *Question Answering* — odpowiadających na pytania) jest dostarczenie **dokładnej**, **precyzyjnej** i **użytecznej** odpowiedzi na pytanie zadane przez użytkownika w języku naturalnym. W pracy ograniczam się do opisu systemów QA z dostępem tekstowym. W systemach tego typu pytanie jest wprowadzane za pomocą tekstu (wyrażenia w języku naturalnym), natomiast odpowiedź wyświetlana jest w postaci krótkiego akapitu (czasem uzupełnionego elementami graficznymi).

Na przykład szukając odpowiedzi na pytanie: *Czy w tym roku urodzi się dziecko Kate i Williama?* użytkownik systemu QA może oczekiwać odpowiedzi składającej

się ze stwierdzenia (prawda/fałsz) z ewentualnym krótkim wyjaśnieniem. Przykładowa odpowiedź zwracana przez system może wyglądać tak[1]:

> Tak. Dziecko księżnej Cambridge Catherine i księcia Williama urodzi się w lipcu — czytamy w oficjalnym oświadczeniu.

Powyższa odpowiedź jest:

- **dokładna** — zawiera wszystkie informacje oczekiwane przez użytkownika,
- **precyzyjna** — nie zawiera żadnych informacji redundantnych oraz nie wymaga od użytkownika wykonania żadnej dodatkowej pracy (np. kliknięcie w odnośnik),
- **użyteczna** — spełnia potrzebę informacyjną użytkownika.

Systemy QA idą więc o krok dalej w stosunku do wyszukiwarek internetowych. Tradycyjne wyszukiwarki internetowe jako odpowiedź na zapytanie dostarczają zbiór odnośników do stron. Użytkownik musi samodzielnie wyszukać odpowiedź pośród dziesiątek (czasem nawet setek tysięcy) znalezionych stron. Różnicę między systemami QA a wyszukiwarkami internetowymi trafnie wyraża Jefrrey Pomerantz w pracy [Pom2005][2]:

> Celem zadania QA jest pozyskanie małych porcji tekstu, które zawierają faktyczną odpowiedź na pytanie, w przeciwieństwie do listy dokumentów, tradycyjnie zwracanych przez systemy wyszukiwania informacji.

3.1.1. Historia systemów QA

Pierwsze systemy QA były wyspecjalizowanymi programami, w których starano się symulować kompetencje językowe człowieka. Najciekawsze prototypy powstałe na tym etapie badań to:

- **Baseball** (1961) — system dostępu do bazy danych na temat rozgrywek w baseball. Odpowiadał na pytania postaci: *„Z kim przegrała drużyna Red Sox w dniu 5 lipca?”*. Pytanie zamieniane było na *zapytanie do bazy* za pomocą specjalnego procesora tekstu. Następnie program szukał odpowiedzi w dedykowanej bazie danych. [GWCL1961]
- **Altair** (1966) — system pozwalający na dostęp do bazy danych astronomicznych za pomocą pytań formułowanych w języku naturalnym. Podobnie jak Baseball Altair był programem tłumaczącym zdania w języku angielskim na kwerendy do bazy danych. [VH1966]
- **Parry** (1971) — program typu *chat-bot*, którego zadaniem było prowadzenie dialogu. Program symulował zachowanie pacjenta chorego na schizofrenię i był jedną z pierwszych prób komputerowej symulacji dialogu człowiek-maszyna. [Col1971]

[1] Źródło: http://www.tvn24.pl
[2] Tłumaczenie własne.

- **Lunar** (1972) — system stworzony przez NASA, którego głównym zadaniem było udostępnienie naturalnego interfejsu do bazy danych próbek geologicznych z Księżyca. System przyjmował pytania formułowane w języku naturalnym, które następnie były tłumaczone na zapytania do bazy danych. Obsługiwał pytania mające formę poleceń, takich jak: *Zidentyfikuj wszystkie próbki, w których znaleziono szkło.* [WKNW1972]
- **QUALM** (1978) — system będący implementacją ogólnego modelu odpowiadania na pytania. Był znacznie bliższy współczesnym systemom QA niż wyżej wymienione przykłady. Zastosowany model przetwarzania obejmował dwie fazy: zrozumienia pytania oraz znalezienia odpowiedzi. Pierwsza faza polegała na kategoryzacji pytania oraz zdefiniowaniu potrzeby informacyjnej, którą wyrażał użytkownik w pytaniu. W fazie drugiej system określał, ile informacji powinno się znaleźć w odpowiedzi oraz szukał odpowiedzi w bazie wiedzy. [Leh1977] QUALM miał jednak szereg ograniczeń związanych z zastosowanym modelem *Conceptual Dependency* (pol. *Zależność Pojęciowa*). Ograniczenia te powodowały niemożność zastosowania do rzeczywistych zbiorów dokumentów pochodzących z różnorodnych dziedzin. [HSJP2003]

Pierwsze systemy QA charakteryzowały się dużą skutecznością i wysoką jakością prezentowanych odpowiedzi. Jednakże działały w ramach wyspecjalizowanych dziedzin, a opracowanych rozwiązań nie można było łatwo rozszerzyć. Przez następne dwie dekady naukowcy próbowali różnych podejść do problemu odpowiadania na pytania począwszy od podejścia symbolicznego (związanego z ogólnymi mechanizmami reprezentacji wiedzy i rozumowania) do empirycznego (w którym główny nacisk kładzie się na analizę języka). [SH2007]

Wraz z powstaniem Internetu i jego dynamicznym rozwojem wzrosło znaczenie **wyszukiwarek internetowych**. Zadaniem wyszukiwarki internetowej jest odnalezienie adresu strony internetowej na podstawie kilku słów kluczowych podanych przez użytkownika. Tak zdefiniowane zadanie można rozumieć jako uproszczone zadanie systemu QA. Popularność wyszukiwarek internetowych oraz ich komercyjny sukces spowodowały, że twórcy systemów QA zaczęli przywiązywać większą wagę do uniwersalności rozwiązań oraz możliwości ich stosowania na znacznych zbiorach danych (najczęściej w postaci kolekcji dokumentów tekstowych). Jednym z przejawów tej zmiany było zapoczątkowanie w 1999 roku corocznej ewaluacji systemów QA o niesprecyzowanej dziedzinie podczas konferencji TREC[3] w *ścieżce QA* (ang. *QA track*).

Na początku dwudziestego wieku powstał szereg dokumentów, w których nakreślano kierunki rozwoju systemów QA. Celem opracowania dokumentów było

[3] http://www.trec.nist.gov

zapewnienie optymalnego rozwoju dziedziny i zapewnienie komercyjnego sukcesu. W raporcie [BCC+2001] z roku 2001 członkowie *Komitetu Mapy Drogowej Systemów Q&A* (ang. *Q&A Roadmap Committee*) wskazywali, że użyteczne systemy QA powinny przestrzegać następujących standardów:

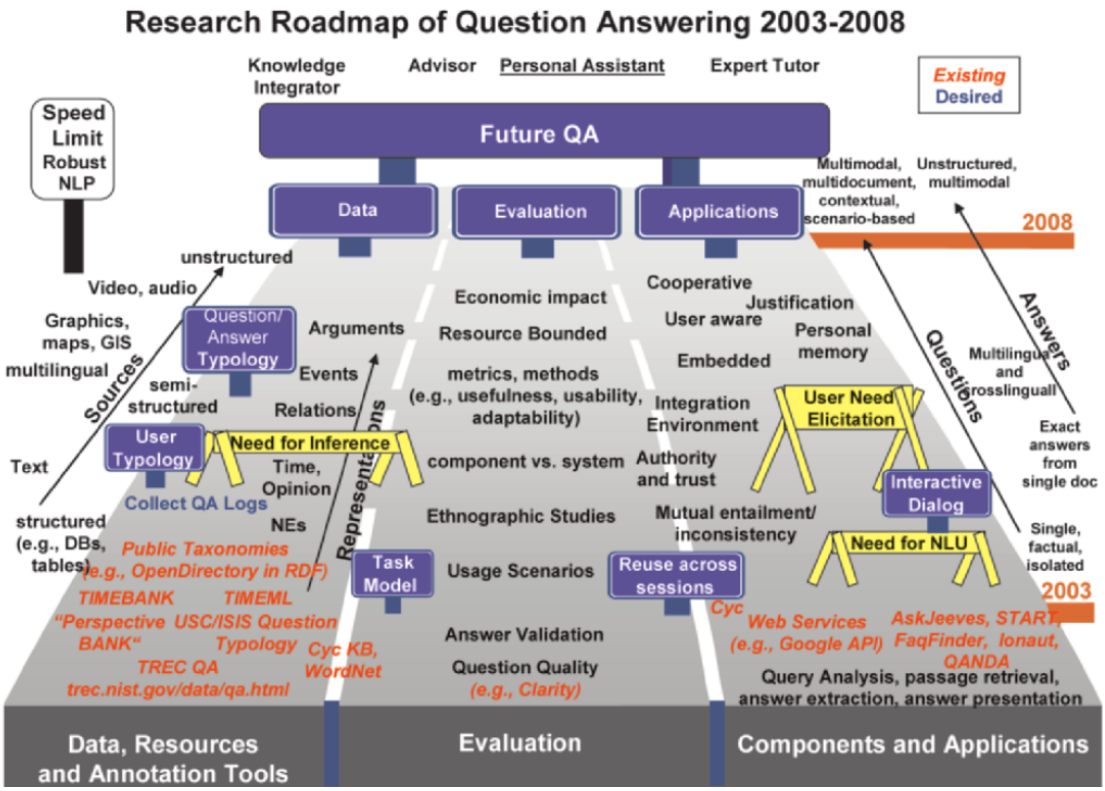
- **aktualność** — baza wiedzy, z której pozyskiwana jest odpowiedź, powinna być nieustannie aktualizowana nowymi danymi (np. na temat aktualnych wydarzeń),
- **dokładność** — zła odpowiedź jest gorsza niż brak odpowiedzi. Powinien istnieć mechanizm wykrywania sprzeczności w bazie wiedzy. Dodatkowo, dla zwiększenia dokładności, system powinien zawierać bazę wiedzy na temat świata oraz posiadać mechanizmy symulujące wnioskowanie.
- **użyteczność** — wiedza systemu powinna być dostosowana do konkretnych potrzeb użytkowników. Powinna istnieć możliwość wykorzystywania niejednolitych źródeł wiedzy (np. dokumentów tekstowych, stron internetowych, baz danych, obrazów, plików wideo).
- **pełność** — pożądane są wyczerpujące odpowiedzi na pytanie. Implikuje to potrzebę stosowania zarówno baz wiedzy ogólnej, jak i specjalizowanych baz wiedzy, a informacje z nich pochodzące należy zapamiętywać i wykorzystywać w procesie wnioskowania.
- **adekwatność** — odpowiedź na pytanie musi być zgodna z danym kontekstem. Ewaluacja systemów powinna być skoncentrowana na użytkowniku (który powinien być ostatecznym sędzią, czy odpowiedź jest poprawna).

Rok później, w roku 2002 podczas warsztatów QA konferencji LREC, opracowano **mapę drogową systemów QA**. Mapa ta, wraz ze zmianami zaproponowanymi podczas *Wiosennego Sympozjum AAAI* na temat *Nowych Kierunków w QA* [May2003] była zbiorem wytycznych badań w dziedzinie QA na lata 2002–2008. Mapa składa się z trzech torów (ang. *tracks*):

- dane, zasoby i narzędzia,
- ewaluacja,
- komponenty, programy i zastosowania.

W każdym z tych torów zdefiniowano szereg problemów, które należało rozwiązać, by osiągnąć nadrzędny cel: zwiększenie produktywności prowadzonych badań oraz uczynienie powstających systemów efektywniejszymi i bardziej użytecznymi. Mapa została przedstawiona na rysunku 3.1.

W roku 2008 grupa badaczy spotkała się w celu przedyskutowania stanu badań w dziedzinie systemów QA. Na spotkaniu powstał dokument [FNA+2008], w którym autorzy, odnosząc się do zadań zawartych w raporcie [BCC+2001] oraz mapy dro-



Rysunek 3.1. Mapa drogowa badań nad systemami QA (za [SH2007]).

gowej opracowanej w latach 2002–2003, wyznaczyli standardy **otwartego rozwoju systemów QA**, którego celem były:

- zwiększenie transparentności prowadzonych badań,
- promowanie rozwiązań ogólnych,
- łatwiejszy przepływ wiedzy między ośrodkami badawczymi.

Ponadto w raporcie zdefiniowano pięć **wyzwań**, z których każde było ocenione według trudności następujących elementów:

- trudność pytań,
- zakres dziedziny,
- czas odpowiedzi,
- dokładność,
- pewność, że odpowiedź jest poprawna,
- użyteczność systemu,
- trudność języka zapytań,
- trudność języka naturalnego użytego w dokumentach składających się na bazę wiedzy.

Wyzwaniami opisanymi w raporcie są:

1. **TREC QA** — jest to zadanie odpowiedzi na 500 pytań za pomocą bazy wiedzy składającej się z około miliona dokumentów tekstowych zapoczątkowane w ramach konferencji TREC. W wyzwaniu tym największa trudność bierze się z wymaganej wysokiej dokładności odpowiedzi i szerokiego zakresu działania systemu (dokumenty mogą dotyczyć dowolnych tematów).
2. **TAC QA**[4] — jest to zadanie polegające na znalezieniu odpowiedzi na 500 pytań (podobnie jak wyzwanie *TREC QA*), jednakże w przeciwieństwie do *TREC QA*, w wyzwaniu należy wykorzystać znacznie większą kolekcję dokumentów (około 3,2 miliona artykułów). Trudność wyzwania została oceniona podobnie jak *TREC QA*, z nieznacznie podwyższoną oceną trudności języka w dokumentach źródłowych (ze względu na częściej występujący język potoczny).
3. **Jeopardy!** — wyzwanie polega na wygraniu (za pomocą systemu QA) serii teleturniejów typu „Jeopardy!".[5] Trudność wyzwania została oceniona wyżej niż dwóch wyżej wymienionych, głównie ze względu na większy nacisk na szybkość działania systemu (odpowiedź musi zostać znaleziona w kilka sekund) oraz wymaganie dobrej oceny pewności znalezionej odpowiedzi (program powinien odpowiadać tylko na te pytania, których odpowiedzi jest pewien).
4. **Uczenie przez czytanie** — wyzwanie inspirowane było jednym z projektów DARPA[6]. W zadaniu system otrzymuje pewien wybrany tekst z książki (np. dotyczącej kardiologii). Następnie system powinien odpowiedzieć na szczegółowe pytania dotyczące wejściowego tekstu. Odpowiedzi powinny zawierać wyjaśnienia, których jakość oceniana jest przez człowieka. W tym wyzwaniu jednym z najtrudniejszych elementów jest nacisk na użyteczność systemu (ze względu na konieczność dostarczenia wyjaśnień odpowiedzi).
5. **Nieprzerwane dochodzenie** — w odróżnieniu od pozostałych wyzwań zadanie wykracza poza odpowiedź na jedno pytanie. Celem jest spełnienie potrzeby informacyjnej użytkownika, poprzez prowadzenie dialogu i dopytywanie. Przez konieczność obsługi skomplikowanej interakcji z użytkownikiem największą trudnością w wyzwaniu jest wymóg wysokiej użyteczności systemu.

W kontekście opisanych wyżej wyzwań spektakularnym sukcesem było opracowanie przez firmę IBM systemu **Watson** [FBCC+2010]. Głównym celem twórców systemu było podjęcie wyzwania **Jeopardy!**.

Sercem systemu jest superkomputer o znacznej mocy obliczeniowej, który potrafi przetwarzać bogate zbiory danych. Baza wiedzy systemu składa się z gigabajtów danych, wśród których są takie zasoby jak: cała zawartość encyklopedii internetowej

[4] http://www.nist.gov/tac/tracks/2008/qa

[5] W Polsce teleturniej ten emitowany był pod nazwą „Vabank". Zabawa polega na zadaniu pytania do odpowiedzi wyświetlonej przez prezentera programu.

[6] DARPA — Agencja Zaawansowanych Obronnych Projektów Badawczych Departamentu Obrony Stanów Zjednoczonych (ang. *Defense Advanced Research Projects Agency*).

Wikipedia, baza DBPedia (patrz podrozdział 3.2), ontologie (np. WordNet), słowniki oraz setki milionów zaindeksowanych stron internetowych.

14 lutego 2011 roku odbył się turniej, w którym Watson zmierzył się z dwoma mistrzami teleturnieju Jeopardy! Kenem Jenningsem oraz Bradem Rutterem. Watson **wygrał rywalizację**. Sukces ten był później kilkukrotnie powtarzany i stanowi istotne osiągnięcie w pracy nad systemami QA. Obecnie firma IBM pracuje nad komercjalizacją projektu.

Imponująca baza wiedzy oraz szereg zaawansowanych algorytmów nie pozwoliła jednak wyeliminować wszystkich błędów. We wspomnianej wyżej finałowej rozgrywce Watson źle odpowiedział na pytanie z kategorii *Miasta Stanów Zjednoczonych* podając jako odpowiedź... **Toronto**.[7]

3.1.2. Podział systemów QA

Istnieje szereg różnych kryteriów, według których można podzielić systemy QA. W pracy skupię się na następujących kryteriach podziału:

- zakres dziedziny,
- obsługa kontekstu,
- złożoność analizy (płytka lub głęboka analiza).

Zakres dziedziny

Ze względu na zakres dziedziny wyróżniamy dwa rodzaje systemów QA:

- o wyspecjalizowanej dziedzinie (ang. *closed domain*),
- o niesprecyzowanej dziedzinie (ang. *open domain*).

Systemy o wyspecjalizowanej dziedzinie działają w ramach ograniczonej tematyki pytań, często ograniczając się tylko do niektórych klas pytań. Pierwsze systemy QA należały do tej kategorii. Obecnie najpopularniejszym zastosowaniem tej grupy systemów QA są programy typu chat-bot[8] umieszczane na stronach internetowych jako wirtualni asystenci.

Systemy o niesprecyzowanej dziedzinie działają zwykle na obszernej bazie wiedzy (bardzo często bazą wiedzy jest cały Internet), a ich zadaniem jest odpowiadanie na szeroki zakres pytań (często zakres pytań jest w praktyce nieograniczony). Systemy tego typu powinny także na bieżąco aktualizować swoją wiedzę poprzez pozyskiwanie nowych faktów na temat zmieniającego się świata.

Obsługa kontekstu

Wyróżnię tutaj dwa rodzaje kontekstu:

[7] Źródło: http://www-03.ibm.com/innovation/us/watson/related-content/toronto.html
[8] Program chat-bot — program symulujący dialog z człowiekiem.

- kontekst zadającego pytanie,
- kontekst dialogu.

Pierwszy z wymienionych kontekstów grupuje wszystkie informacje dotyczące użytkownika zadającego pytanie. W szczególności:

- tożsamość użytkownika — rozróżnienie pytań pochodzących od różnych nadawców,
- aktualny czas — kiedy zadawane jest pytanie,
- miejsce — w jakim miejscu znajduje się zadający pytanie (np. w jakim mieście),
- profil użytkownika — zainteresowania i zwyczaje użytkownika (np. pozyskane na podstawie wcześniej zadanych pytań).

Ten rodzaj kontekstu jest często obsługiwany przez wyszukiwarki internetowe (np. Google, Bing). Zapytanie wysłane do wyszukiwarek internetowych przez użytkowników o różnym profilu (np. mieszkających w różnych miejscach na Ziemi) może zwrócić znacznie różniące się wyniki.

Kontekst dialogu wykracza poza przetwarzanie jednego pytania. System QA obsługujący ten rodzaj kontekstu powinien symulować rozmowę użytkownika z komputerem. W szczególności program powinien potrafić:

- śledzić temat rozmowy,
- umieć zidentyfikować odniesienia do poprzednich pytań (np. wyrażane za pomocą anafory, porównaj pytanie: *Gdzie leży **to** miasto?*),
- umieć podtrzymać rozmowę stosując techniki dopytywania i wyjaśniania.

Przykładem systemów tego typu są wirtualni asystenci i programy typu chat-bot. W dalszej części pracy, skupię się na pierwszym rodzaju kontekstu.

Powierzchniowe i głębokie QA

Ze względu na charakter metod użytych w procesie odpowiadania wyróżniamy dwa typy systemów:

- powierzchniowe QA (ang. *shallow QA*),
- głębokie QA (ang. *deep QA*).

Metody używane w **powierzchniowym QA** bazują na analizie statystycznej tekstu. W procesie odpowiadania zakłada się, że odpowiedź jest jawnie podana w materiale źródłowym. Dzięki temu zadanie QA upraszcza się do znalezienia fragmentu tekstu pasującego do pytania. W systemach tego typu nie przeprowadza się semantycznej analizy tekstu. Przetwarzanie tekstów jest zwykle ograniczone do poziomu znakowego, niekiedy uzupełnionego podstawowymi informacjami językowymi

(takim jak formy bazowe wyrazów lub części mowy). Najczęstszymi technikami używanymi w tym podejściu są: wyszukiwanie słów kluczowych oraz użycie reguł.

W **głębokim QA** używa się znacznie bardziej zaawansowanych metod przetwarzania pytania. Przeprowadza się analizę semantyczną tekstów źródłowych, a bazę wiedzy przechowuje się w sposób ustrukturyzowany (np. w postaci ontologii). W procesie szukania odpowiedzi stosuje się wnioskowanie oraz integrację różnych źródeł odpowiedzi. Odpowiedź jest często generowana automatycznie.

3.1.3. Zagadnienia badawcze

Poniżej znajduje się wybór głównych zagadnień badawczych zadania QA. Wyboru dokonano za pracą [BCC+2001].

Klasyfikacja pytań

Do klasycznych klasyfikacji pytań zaliczamy modele zaproponowane przez Wendy Lehnert [Leh1977] oraz Artura Graessera [GP1994].

Model Lehnert oparto na trzynastu kategoriach. Taksonomia pytań została zbudowana w oparciu o teorię reprezentacji pamięci zwaną *Conceptual Dependency* (pol. Zależność Pojęciowa). W modelu wprowadzono pojęcie **fokusu pytania**. Fokus pytania został zdefiniowany jako potrzeba informacyjna, którą wyraża pytanie. Ilustruje to następujący przykład:

> Dlaczego John pojechał ostatniej nocy do McDonald'sa na wrotkach?

Naiwną odpowiedzią na tak postawione pytanie jest:

> Odpowiedź: Bo był głodny.

Lehnert zauważa, że w istocie pytanie powinno być zrozumiane następująco:

> Dlaczego John pojechał na wrotkach zamiast iść, wziąć samochód albo użyć jakiegokolwiek innego racjonalnego środka transportu?

Przykład ten pokazuje, że bez dobrze zidentyfikowanego fokusu pytania, znalezienie poprawnej odpowiedzi (tzn. takiej, która usatysfakcjonuje użytkownika) może być niemożliwe.

W modelu Graessera [GP1994] pytania zostały podzielone na osiemnaście kategorii, z których większość została zaczerpnięta z modelu Lehnert. Empirycznie dowiedziono, że zaproponowana taksonomia jest kompletna, tzn. obejmuje wszystkie zapytania, jakie mogą pojawić się w trakcie rozmowy. Sprawdzenie obejmowało następujące przypadki użycia:

- studenci czytający fragmenty tekstu,
- osoby używające komputera,

- obywatele zadający pytania w gazetach.

Wymienione modele klasyczne mają jednak swoje ograniczenia. Najważniejszym ograniczeniem jest trudność zastosowania wymienionych wyżej modeli w systemach o niesprecyzowanej dziedzinie. Implikuje to nieustanną potrzebę badań nad teoriami klasyfikacji pytań. W ramach badań szuka się odpowiedzi na takie pytania jak:

- jakie są kryteria formułowania taksonomii pytań?
- jaka jest zależność między klasą pytania a trudnością znalezienia odpowiedzi?
- jakie są kryteria trudności pytania?

Obsługa kontekstu

Omawiając podział systemów QA w podrozdziale 3.1.2, jako jedno z kryteriów wyróżniłem kontekstowość. Kontekst jest istotnym elementem badań nad systemami QA. Występuje zarówno w zadawanym pytaniu, jak i w dokumentach będących źródłem odpowiedzi.

Rozpatrzmy następujący przykład[9]:

> Czy w zeszłym roku przewidziano koniec świata?

Zauważmy, że fraza *w zeszłym roku* oznacza inny przedział czasowy w zależności od momentu zadania pytania. Spowodowane jest to zjawiskiem zmiany kontekstu pytania (w tym przypadku kontekstu czasowego).

Inny rodzaj kontekstu ma związek z niejednoznacznością języka. Rozpatrzmy następujący przykład:

> Gdzie są Włochy?

W zależności od kontekstu pojęcie *Włochy* może oznaczać zarówno państwo w Europie jak i jedną z dzielnic Warszawy. Rozwiązanie tego problemu może polegać na przykład na powiązaniu użytkownika z miejscem w którym zadaje pytanie (użytkownikowi mieszkającemu w Warszawie prawdopodobnie chodzi o dzielnicę), analizując historię pytań zadanych przez użytkownika (użytkownik pytający o wycieczki turystyczne pewnie ma na myśli nazwę państwa) bądź stosując heurystyki statystyczne (znaczenie nazwy *Włochy* jako nazwy państwa występuje częściej).

Badania nad kontekstem obejmują takie zadania jak:

- sformułowanie modelu kontekstu (użytecznego z punktu widzenia systemów QA),
- modelowanie kontekstu definiowanego przez pytania zadane przez użytkownika bądź grupę użytkowników,

[9] Pytanie dotyczy przepowiedni końca świata, który miał mieć miejsce w grudniu 2012 roku. Przepowiednia związana była z końcem kalendarza Majów i przez długi czas była szeroko komentowana w mediach.

- sprawdzenie wpływu kontekstu na przetwarzanie pytania, w tym zbadanie jak istotny jest kontekst w znalezieniu odpowiedzi przez konkretne systemy QA.

Problem kontekstu odgrywa istotną rolę w implementacji algorytmu odpowiadania na pytania rozstrzygnięcia opisanego w rozdziale 5.

Źródła wiedzy

Aby odpowiedzieć na pytanie, system QA musi dysponować wyczerpującą bazą wiedzy, w której może szukać odpowiedzi. Z tego powodu prowadzone są intensywne prace nad opracowaniem obszernych baz wiedzy dla systemów QA.

Główne zadania zbierania baz wiedzy dla systemów QA polegają na:

- opracowaniu różnorodnych formatów baz danych,
- opracowaniu metod wykorzystania baz danych na potrzeby systemów QA,
- opracowaniu baz danych zawierających wyczerpujące dane z różnych dziedzin.

Podstawowym źródłem wiedzy (szczególnie w systemach o niesprecyzowanej dziedzinie) jest **kolekcja dokumentów tekstowych**. Ze względu na **nieustrukturyzowany** charakter takiej bazy wiedzy, jej wykorzystanie jest jednak ograniczone oraz obarczone trudnościami technicznymi (konieczność zarządzania obszernymi zbiorami danych). Dane w takim formacie dostarczane były podczas ewaluacji QA w ramach konferencji TREC [VT2000]. Do tak zorganizowanych danych stosuje się techniki zaczerpnięte z klasycznego zadania **wyszukiwania informacji** np.: tworzenie indeksu dokumentów oraz wyszukiwanie słów kluczowych (patrz np. praca [MRS2008]).

Innym źródłami wiedzy są **bazy ustrukturyzowane**. Do baz tego typu zaliczamy na przykład ontologie i bazy wiedzy przestrzennej.

Ontologią nazywamy formalną reprezentację wiedzy za pomocą pojęć i relacji zachodzących między pojęciami. Jedną z najintensywniej rozwijanych ontologii jest angielski WordNet [Fel1998]. Istnieje szereg przykładów wykorzystania WordNetu jako bazy wiedzy systemu QA [CFH2008]. Najczęściej jest on wykorzystywany do obsługi wiedzy ogólnej o świecie.

Bazy wiedzy przestrzennej są przykładem bazy specjalizowanej, która opisuje pewien fragment świata rzeczywistego. Przykładem systemu używającego bazy wiedzy przestrzennej jest projekt opisany w pracy [FR2006]. System używa między innymi bazy GNIS[10] (ang. *Geographic Names Information System*, baza obejmuje terytorium Stanów Zjednoczonych) oraz bazy GEOnet[11] (baza obejmuje miejsca z całego świata, poza Stanami Zjednoczonymi). Jest to przykład systemu o wyspecjalizowanej dziedzinie, działający dla pytań o tematyce geograficznej.

[10] http://geonames.usgs.gov/
[11] http://earth-info.nga.mil/gns/html/

Problem zbierania wiedzy jest szerzej omówiony w podrozdziale 3.2. Autorska metoda zbierania bazy wiedzy, przedstawiona jest w rozdziale 4.

Zaawansowane wnioskowanie

Nie zawsze odpowiedź dana jest jawnie w danych źródłowych. Rozpatrzmy następujący fragment dokumentu:[12]

> Kot pogryzł psa na Wildzie. Agresywny kot pogryzł mojego psa! — takie nietypowe zgłoszenie otrzymał w piątek patrol Straży Miejskiej.

Załóżmy, że system QA odpowiada na pytanie:

> Czy w Poznaniu kot pogryzł psa?

Odpowiedź na to pytanie nie znajduje się bezpośrednio w dokumencie źródłowym. System QA musi najpierw *zinterpretować* pojęcie *Wilda* jako dzielnicę miasta Poznań. Następnie powinien **wywnioskować**, że skoro Wilda znajduje się w Poznaniu, to opisywane wydarzenie również miało miejsce w Poznaniu.

Inny przykład obejmuje wyrażenia czasowe. Rozpatrzmy fragment dokumentu opublikowanego w lipcu 2011 roku:[13]

> Innym wielkim twórcą teatralnym, którego zobaczymy na Malcie, jest Jan Fabre. — W zeszłym roku gościł u nas z monodramem, tym razem zobaczymy jego zupełnie inne przedstawienie: duże, wręcz monumentalne — zapowiada Anna Reichel.

Załóżmy, że w roku 2013 system QA ma odpowiedzieć na następujące pytanie:

> Czy w zeszłym roku Jan Fabre był na Festiwalu Malta?

Zauważmy, że użycie płytkich metod przetwarzania, które nie korzystają z wnioskowania spowoduje podanie błędnej odpowiedzi. System działający powierzchniowo dopasuje identyczne fragmenty frazy *w zeszłym roku*. Frazy te (ze względu na kontekst) oznaczają jednak różne pojęcia.

W pytaniu fraza *w zeszłym roku* odnosi się do aktualnego czasu, oznacza więc **rok 2012**. W treści dokumentu fraza ta odnosi się do daty opublikowania artykułu, przez co oznacza **rok 2010**. Dopiero głęboka analiza tekstu połączona z analizą kontekstu i wnioskowaniem może doprowadzić do poprawnej odpowiedzi.

Te oraz podobne przykłady są motywacją do badań nad mechanizmami wnioskowania w systemach QA. W pracy ograniczam się do wnioskowań czasowych i przestrzennych. Poniżej przedstawiam wybrane systemy QA, w których wykorzystano wnioskowanie.

[12] Źródło: http://www.mmpoznan.pl/403569/2012/2/19/
[13] Źródło: http://www.mmpoznan.pl/377344/2011/7/1/

Uribe 2002

W pracy [UCHS2002] opisano zastosowanie wnioskowania przestrzennego w systemie SHAKEN. Jest to system o wyspecjalizowanej dziedzinie, który służy jako interfejs dostępu do bazy danych. Pytania mają postać twierdzeń logicznych. Odpowiadanie polega na dowiedzeniu prawdziwości twierdzenia. Przykładowe pytania mogą mieć postać:

- jeśli **nić DNA** jest częścią **jądra** i **jądro** jest wewnątrz **komórki**, to **nić DNA** jest wewnątrz **komórki**,
- jeśli **komórka 1** oraz **komórka 2** są rozłączne i **nić DNA** jest częścią **komórki 1**, to **nić DNA** nie jest częścią **komórki 2**,
- jeśli **jądro** jest wewnątrz **komórki** oraz **komórka** jest w miejscu **X**, to **jądro** jest w miejscu **X**.

W procesie wnioskowania wykorzystano rachunek RCC8 oraz wariant algorytmu *path-consistency* (algorytmu PC, patrz podrozdział 2.1.5).

Problemem systemu była szybkość działania, którą zbadano na wybranym pytaniu, nazwanym przez twórców systemu *pytaniem przykładowym*. Pierwsza wersja algorytmu znajdowała odpowiedź na przykładowe pytanie w czasie pięciu minut, a w procesie dowodzenia generowanych było dziesiątki tysięcy formuł logicznych. Po optymalizacji autorzy osiągnęli szybkość trzech sekund.

Harabagiu 2005

W pracy [HMC+2005] opisano system QA Power-Answer-2, zrealizowany w *ścieżce QA* konferencji TREC. W zbiorze testowym pytań TREC 2005 istotną rolę odgrywały pytania zawierające pewne odnośniki czasowe. W związku z tym, autorzy systemu Power-Answer-2 zdecydowali się dodać moduł wnioskowania czasowego oparty o system LCC [MCH2005].

Algorytm wnioskowania czasowego za pomocą LCC można w skrócie przedstawić w następujący sposób:

1. Zidentyfikuj daty pojawiające się w tekście.
2. Znajdź wydarzenia w pytaniu i zbiorze kandydatów na odpowiedzi.
3. Zunifikuj informacje czasowe w pytaniu z informacjami czasowymi znalezionymi w zbiorze kandydatów (sprawdź czy są ze sobą zgodne). Preferuj kandydatów, którzy pasują do wszystkich warunków czasowych w pytaniu.

   Rozpatrzmy następujące pytanie przykładowe:

   Kto był dyrektorem DePauw w 1999 roku?

Wyżej wymienione pytanie przykładowe jest zamieniane na zapytanie o dane osobowe dyrektora DePauw College, z warunkiem czasowym *w 1999 roku*.

Wnioskowanie zostało zrealizowane z użyciem logiki pierwszego rzędu. Wydarzenia i relacje temporalne były reprezentowane za pomocą ontologii SUMO (ang. *Suggested Upper Merged Ontology*) [NP2001].

Chociaż aż 16% pytań o fakt[14] w TREC 2005 zawierało odniesienia czasowe, to zastosowanie wnioskowania czasowego przyniosło tylko dwuprocentowy wzrost w ocenie systemu. Autorzy tłumaczą to znacznym odsetkiem pokrywania się znakowego pytań i odpowiedzi zawartych w źródłowych dokumentach. Wskazuje to także na duży potencjał rozwoju metody.

Vetulani 2010

Praca [VMO+2010] opisuje system POLINT-112-SMS, w którym zastosowano mechanizmy wnioskowania czasowego i wnioskowania przestrzennego. Jest to system QA o wyspecjalizowanej dziedzinie działający w języku polskim. Zadaniem systemu jest zarządzanie informacjami na temat zdarzeń dziejących się na ograniczonej przestrzeni (np. na meczu piłkarskim) w ograniczonym czasie. System zbiera informacje pochodzące od agentów terenowych wysyłane za pomocą wiadomości tekstowych SMS oraz odpowiada na pytania agentów. Przykładową informacją wysłaną do systemu może być: *bójka na południu sektora A*. Przykładowe pytanie to: *Gdzie jest bójka?*.

Wnioskowanie czasowo-przestrzenne zrealizowano wykorzystując model XCDC (ang. *eXtended Cardinal Direction Calculus*) [LVO2009], który jest rozszerzeniem klasycznego formalizmu jakościowego CDC [GE2001]. Model XCDC został uzupełniony o zależności czasowe, dzięki czemu system wnioskuje korzystając z następujących typów relacji:

- bezwzględne relacje kierunkowe: północ, południe, wschód, zachód,
- względne relacje kierunkowe: lewo, prawo, przed, za,
- relacje topologiczne: w, przy, otoczony, poza,
- relacje czasowe: po, wcześniej, równocześnie, w trakcie, około.

W procesie wnioskowania wykorzystywana jest operacja złożenia relacji i konstruowana jest **ścieżka złożeń**.

Na przykład załóżmy, że system posiada dwa fakty:

- A znajduje się na północ od B,
- B znajduje się na północ od C,

System wykorzystuje wnioskowanie, aby dowieść z dwóch wyżej wymienionych faktów, że *A znajduje się na północ od C*.

[14] **Pytanie o fakt** — jest to pytanie dotyczące pewnej pojedynczej informacji na temat wydarzenia, np.: Kiedy Polska wstąpiła do Unii Europejskiej?

System POLINT-112-SMS jest przykładem systemu o wyspecjalizowanej dziedzinie, który w istotny sposób bazuje na dziedzinie działania. Oprócz faktów wprowadzonych przez agentów system wykorzystuje bazę wiedzy ogólnej „PolNet-Polish WordNet" [VWO+2009], będącą ontologią typu WordNet (patrz podrozdział 3.2). Baza została poszerzona o słownictwo dziedzinowe z zakresu „bezpieczeństwa wewnętrznego".

Obsługiwany zbiór relacji został opracowany w oparciu o specyficzny język związany z zadaniem systemu. Moduł wnioskowania działa wyłącznie na faktach wprowadzonych przez agentów, przez co ich liczba jest ograniczona. Zastosowanie tego modelu w systemach o niesprecyzowanej dziedzinie może powodować kłopoty techniczne (rozszerzenie dziedziny) i obliczeniowe (ilość przetwarzanych informacji).

3.2. Bazy wiedzy

Opisując projekt Watson wskazałem obszerną bazę wiedzy jako istotny element systemu. Opracowanie bazy wiedzy przestrzennej jest też jednym z zadań mojej pracy. W tym miejscu przedstawiam stan badań i osiągnięć w dziedzinie zbierania wiedzy na potrzeby systemów QA.

3.2.1. Charakterystyka baz wiedzy

Każda kolekcja danych może stanowić bazę wiedzy dla systemu QA. Na przykład w TREC bazą wiedzy była kolekcja dokumentów tekstowych. Dane w takiej bazie nie mają jednolitego charakteru i wymagają dalszego przetworzenia. Na potrzeby pracy bazę wiedzy składającą się wyłącznie z nieprzetworzonych dokumentów tekstowych nazywamy **nieustrukturyzowaną bazą wiedzy**.

Na drugim biegunie mamy system POLINT-112-SMS, w którym bazą wiedzy jest ontologia. Baza tego typu charakteryzuje się znacznie większym stopniem formalizacji i organizacji przechowywanych danych. Pozwala na złożone przetwarzanie danych (np. umożliwia wnioskowanie). Bazy tego typu nazywamy w niniejszej rozprawie **ustrukturyzowanymi bazami wiedzy**.

Możemy wyróżnić dwa rodzaje wiedzy przechowywanej w bazach wiedzy:[15]

- **wiedza ilościowa** (ang. *quantitative knowledge*) — jest to wiedza zorganizowana w postaci mierzalnych wartości (np. położenie geograficzne, wysokość nad poziomem morza, powierzchnia),

[15] Należy zaznaczyć, że podział ten nie jest rozłączny, a pojęcia te nie są antonimami. W zależności od zastosowania dany rodzaj wiedzy może być postrzegany jako drugi (np. podział administracyjny może być zarówno przykładem wiedzy ilościowej jak i jakościowej).

- **wiedza jakościowa** (ang. *qualitative knowledge*) — jest to wiedza zorganizowana w formie relacji miedzy obiektami w bazie (np. położenie obiektu wewnątrz drugiego obiektu).

W dalszej części rozdziału skupię się na ustrukturyzowanych bazach wiedzy przestrzennej, wykorzystywanych w procesie wnioskowania przestrzennego. Przykładami baz tego typu są gazetery i ontologie.

Gazetery

Gazeter (ang. *gazetteer*) jest to baza obiektów geograficznych, zawierająca informacje geograficzne i geopolityczne. Może zawierać takie rodzaje danych jak:

- nazwa danego obiektu i jej warianty (np. nazwa w różnych językach, nazwy potoczne i zwyczajowe),
- położenie geograficzne (długość i szerokość geograficzna),
- wymiary (np. wysokość, powierzchnia),
- informacje geopolityczne (np. ludność, gęstość zaludnienia),
- podział administracyjny,
- kategoria obiektu (np. las, jezioro, miasto, rzeka itp.).

Gazeter może dotyczyć zarówno całej Ziemi, jak i wybranego obszaru (np. danego państwa lub regionu).

Do popularnych gazeterów należą:

- **Geographic Names Information System (GNIS)**[16] — jest to baza informacji geograficznych obejmująca teren Stanów Zjednoczonych oraz Antarktydy, opracowana przez *U.S. Board on Geographic Names* (ang. *Amerykański Zarząd Nazw Geograficznych*).
- **NGA GEOnet Names Server (GNS)**[17] — jest to druga część zasobów GNIS obejmująca obiekty geograficzne z całego świata (z wyłączeniem Stanów Zjednoczonych i Antarktydy dostępnych w GNIS).
- **Geonames.org**[18] — jest to baza informacji geograficznej z całego świata, powstała poprzez integrację różnych otwartych źródeł oraz dane ręcznie wprowadzone przez użytkowników serwisu.
- **DBPedia**[19] — jest to projekt opracowania ustrukturyzowanej bazy wiedzy, bazujący na informacjach dostępnych w internetowej encyklopedii Wikipedia. Dane geograficzne stanowią podzbiór przechowywanych danych. Dostępne są polskojęzyczne nazwy niektórych pojęć przechowywanych w bazie.

[16] http://nhd.usgs.gov/gnis.html
[17] http://earth-info.nga.mil/gns/html/index.html
[18] http://www.geonames.org
[19] http://dbpedia.org

- **Narodowy Gazeter Polski**[20] — jest to wykaz nazw obiektów wodnych (tzw. *hydronimów*) opracowany przez Komisję Standaryzacji Nazw Geograficznych poza Granicami Rzeczpospolitej Polskiej (KSNG).
- **TERYT**[21] — Krajowy Rejestr Urzędowy Podziału Terytorialnego Kraju (TERYT), jest to wykaz miejscowości polskich wraz z podziałem administracyjnym, który został opracowany przez Główny Urząd Statystyczny (GUS). Pozwala na opracowanie bazy wiedzy jakościowej (która miejscowość znajduje się w której jednostce podziału terytorialnego).

Dostępne zasoby geograficzne są obszerne, jednakże ich zastosowanie w polskojęzycznym systemie QA o niesprecyzowanej dziedzinie jest kłopotliwe ze względu na:

- brak danych w języku polskim — większość obszernych zasobów zawiera nazwy w języku angielskim, gazetery w języku polskim dotyczą zwykle terytorium Polski,
- wyspecjalizowany charakter gazeterów i brak niektórych typów obiektów (np. budynków, zabytków),
- niejednorodność danych — różne gazetery zawierają różne rodzaje danych. Format danych jest niejednorodny.

Ontologia

Ontologia jest formalną reprezentacją wiedzy. Zawiera **pojęcia** oraz **relacje**, jakie zachodzą między pojęciami.

Najczęściej używanymi relacjami w ontologiach są:

- **hiponimia** — A jest **hiponimem** B wtw A jest rodzajem B (np. każdy *kot* jest *ssakiem*, więc *kot* jest hiponimem *ssaka*),
- **hiperonimia** — relacja odwrotna do relacji **hiponimii**,
- **meronimia** — A jest **meronimem** B wtw A jest częścią B (np. *centrum miasta* jest meronimem *miasta*),
- **holonimia** — relacja odwrotna do relacji **meronimii**.

W zagadnieniach związanych z przetwarzaniem języka naturalnego wykorzystuje się głównie ontologie typu WordNet[22], w których główny nacisk kładzie się na językową warstwę pojęć przechowywanych w bazie.

Przykładowymi WordNetami są:

- **Princeton WordNet** — najstarszy i jeden z najpopularniejszych WordNetów rozwijany od 1985 roku na Uniwersytecie Princeton. Jest opracowany dla języka angielskiego i zawiera prawie 120 tysięcy pojęć. [Fel1998]

[20] http://ksng.gugik.gov.pl/narodowy_gazeter_polski_t_01.php
[21] http://www.stat.gov.pl/bip/36_PLK_HTML.htm
[22] W języku polskim ontologię tego rodzaju nazywa się czasem **słowosiecią**.

- **plWordNet** — projekt rozwijany od 2005 roku na Politechnice Wrocławskiej. Zawiera pojęcia w języku polskim mapowane na Princeton WordNet. Obejmuje ponad 110 tysięcy pojęć. [PSB2009]
- **PolNet** — projekt rozwijany na Uniwersytecie im. Adama Mickiewicza obejmujący słownictwo języka polskiego. Projekt został rozpoczęty w roku 2007. Zawiera ponad 13 tys. pojęć. [VWO+2009]

Ontologie służą jako reprezentacja wiedzy dowolnego typu, dlatego możliwe jest przechowywanie w niej także informacji geograficznej. Na przykład pojęciu **Poznań** w słowosieci *plWordNet* odpowiada następujący układ relacji:[23]

```
Synset: Poznań
Relacje:
    meronimia: miejsce
        { Poznań } jest meronimem (typu miejsce) { Wielkopolska }
            { Wielkopolska } jest meronimem
                    (typu miejsce) { Polska }
                { Polska }
                      jest meronimem (typu miejsce)
                      { Europa Środkowa }
```

Zaprezentowany wyżej fragment słowosieci opisuje położenie *Poznania* wewnątrz *województwa Wielkopolskiego*, które znajduje się wewnątrz *Polski*, a która jest częścią *Europy Środkowej*. Przykład ten pokazuje, że w ontologiach można przechowywać wiedzę jakościową, użyteczną w procesie wnioskowania przestrzennego.

3.2.2. Metody pozyskiwania baz wiedzy

Ręczne wprowadzanie danych

Najprostszą metodą zbierania wiedzy jest jej ręczne wprowadzenie przez grupę ekspertów danej dziedziny. Ze względu na znaczny koszt takiego przedsięwzięcia jest to metoda nieefektywna i czasochłonna. Prowadzi jednak do opracowania danych wysokiej jakości. Ręczne wprowadzanie stosowane jest często w początkowych fazach opracowywania danych, bądź tylko dla niektórych (kluczowych) podzbiorów danych opisywanej w bazie dziedziny.

Wariantem metody jest ręczne wprowadzanie danych przez społeczność użytkowników. Model ten zastosowano tworząc bazę **Geonames**. Jedną z możliwości serwisu jest funkcja modyfikacji dowolnego pojęcia przez dowolnego użytkownika sieci Internet (jest to tzw. *mechanizm wiki*). Zaangażowanie całej społeczności Internetu pozwala na znaczne rozszerzenie zasobów. Skutkiem ubocznym jest spadek jakości

[23] Źródło: http://plwordnet.pwr.wroc.pl/wordnet/wordnet/Poznań

danych w bazie, które wymagają weryfikacji przez moderatora (np. administratora serwisu).

Automatyczne Wydobywanie Informacji

Wydobywanie Informacji (ang. *Information Extraction*, IE) zadanie pozyskiwania danych ustrukturyzowanych z danych nieustrukturyzowanych (np. z tekstu zapisanego w języku naturalnym). Metody IE mogą służyć do tworzenia baz wiedzy.

Podejście to zastosowano w projekcie **DBPedia** [AL2007] [BLK+2009]. Źródłem danych DBPedii jest internetowa encyklopedia Wikipedia. Wikipedia jest źródłem *częściowo ustrukturyzowanym.*[24] Na potrzeby DBPedii opracowano algorytm IE, który bazując na ustrukturyzowanej części Wikipedii, wyciąga dane w formie ustrukturyzowanej. Algorytm składa się z następujących etapów:

1. Znajdź wszystkie strony Wikipedii, które zawierają szablon.
2. Wyciągnij szablony ze stron i wybierz te, które mogą zawierać ustrukturyzowane dane.
3. Przetwórz każdy szablon wyciągając odpowiednie krotki danych.
4. Sprawdź poprawność danych przetworzonych w poprzednim kroku.
5. Zidentyfikuj klasę danych na podstawie kategorii strony, z której została pozyskana lub rodzaju szablonu.

Zaprezentowane rozwiązanie pozwala na uzyskanie wyników wysokiej jakości stosunkowo małym kosztem. W algorytmie pominięto jednak tekstową zawartość Wikipedii (np. treść artykułu). Potencjalne rozwinięcia projektu polegają na rozszerzeniu algorytmu wydobywania na tekstową część Wikipedii, bądź zastosowaniu innych źródeł częściowo ustrukturyzowanych (katalogi, wykazy) lub nawet nieustrukturyzowanych (cała sieć Internet).

Podsumowując, należy stwierdzić, że zaletą metod automatycznych jest ich efektywność. Stosunkowo niskim kosztem można opracować obszerne zbiory danych. Z drugiej strony jakość zebranych danych może być znacznie niższa od danych zebranych ręcznie.

Integracja istniejących źródeł

Ze względu na mnogość dostępnych zasobów wiedzy przestrzennej jedną z możliwych metod opracowania bazy wiedzy jest integracja istniejących źródeł szczegółowych w jedno ogólne.

Metodę integrowania istniejących źródeł zastosowano tworząc bazę **Geonames**. Serwis korzysta między innymi z następujących źródeł danych:[25]

[24] Częściowe ustrukturyzowanie Wikipedii wynika np. z zastosowania tzw. *infoboksów*, czyli szablonów fragmentów stron zawierających konkretne informacje, zwykle zorganizowane w prostej tabeli.

[25] Źródło: http://www.geonames.org/data-sources.html

- gazetery narodowe np.: GNA, GNIS, GeoBase[26] (gazeter narodowy Kanady),
- strony informacji turystycznej np.: OurAirports[27], Hotels.com[28], LateRooms.com[29],
- serwisy informacji statystycznych np.: Statystyki Danii[30], dane Narodowego Instytutu Statystycznego Bułgarii[31].

Zintegrowanie wielu różnych źródeł pozwoliło na uzyskanie wyczerpujących danych. Niejednorodność źródeł prowadzi jednak do następujących problemów:

- niespójność danych wewnątrz bazy,
- kumulowanie się błędów występujących w źródłach służących integracji,
- niejednoznaczności nazw, np. nazwa *Poznań* oznaczająca *zamek* jako pierwsze z wyszukanych znaczeń.[32]

System integrujący różne źródła musi mieć mechanizm rozwiązujący przynajmniej część z wymienionych wyżej problemów, aby wynikowa baza danych miała zadowalającą jakość.[33]

3.3. Reprezentacja czasu i przestrzeni

Wymaganiem systemu przetwarzającego wiedzę czasową i przestrzenną jest formalna reprezentacja wiedzy. Zadanie opracowania takiej reprezentacji było jednym z elementów mapy drogowej QA. W tym podrozdziale przedstawiam dwa najpopularniejsze formalizmy: TimeML (opisujący wiedzę czasową) oraz SpatialML (opisujący wiedzę przestrzenną).

3.3.1. TimeML

Język TimeML (ang. *Time Markup Language*) służy do opisu wyrażeń czasowych i wydarzeń. Został stworzony na potrzeby programu systemów QA prowadzonego przez AQUAINT[34] w ramach sześciomiesięcznego warsztatu TERQAS [PCI+2003]. Najnowsza wersja języka, o nazwie ISO-TimeML, została zaakceptowana jako międzynarodowy standard [PLBR2010].

Celem opracowania języka było umożliwienie:

[26] http://www.geobase.ca
[27] http://www.ourairports.com
[28] http://www.hotels.com
[29] http://www.laterooms.com
[30] http://www.dst.dk/da/
[31] http://www.nsi.bg
[32] http://www.geonames.org/search.html?q=Poznań
[33] W bazie Geonames.org jako metodę weryfikacji danych zastosowano opisany wyżej mechanizm wiki. Każdy użytkownik serwisu może zmodyfikować pojęcie w bazie usuwając znalezione błędy.
[34] Zaawansowane QA dla Wywiadu (ang. *Advanced Question Answering for Intelligence*).

- oznaczenia wydarzeń i opisania ich umiejscowienia w czasie,
- uporządkowania wydarzeń na linii czasu,
- wnioskowania czasowego (także na kontekstowych wyrażeniach czasowych takich jak *w zeszłym roku*).

Oznaczenia wyrażeń czasowych za pomocą języka TimeML są wprowadzane za pomocą znaczników XML. Przykładowe oznaczenie dla frazy *1 stycznia* może mieć postać:

```
<TIMEX3 tid="t1" type="DATE" value="xxxx-01-01" >
1 stycznia
</TIMEX3>
```

W powyższym przykładzie znacznik `TIMEX3` oznacza podstawowy znacznik reprezentujący wyrażenie czasowe. Atrybut `type` określa typ jednostki (w tym przypadku jest to data). Atrybut `value` określa wartość jednostki. Zwróćmy uwagę, że w powyższym przykładzie wartość daty nie jest w pełni zdefiniowana (brakuje roku).

Podzbiór TimeML został wykorzystany do reprezentacji wiedzy czasowej w algorytmach wnioskowania czasowego opisanych w niniejszej pracy (patrz rozdział 5). Pełny opis języka można znaleźć w pracy [PLBR2010] lub na stronie internetowej projektu TimeML.[35]

3.3.2. SpatialML

Głównym celem języka SpatialML (ang. *Spatial Markup Language*) jest oznaczenie nazw miejsc występujących w tekście oraz przyporządkowanie ich do odpowiadających pojęć w bazie wiedzy przestrzennej [MDH+2010].

Model opisu języka SpatialML składa się z miejsc (znacznik `PLACE`) oraz połączeń między miejscami (znacznik `LINK`). Język ten umożliwia:

- zdefiniowanie położenia obiektu za pomocą współrzędnych geograficznych,
- określenie orientacji obiektu (np. według stron świata),
- określenie relacji między obiektami (w tym relacji jakościowych).

Przykładowe oznaczenie dla frazy *zachodnie Pomorze* może mieć postać:

```
zachodnie <PLACE mod="W" country="PL" form="NAM">Pomorze</PLACE>
```

W powyższym przykładzie znacznik `PLACE` wyznacza granice jednostki przestrzennej. Jest to nazwa własna (atrybut `form` ma wartość `NAM`), która znajduje się w kraju o kodzie `PL` (atrybut `country`). Modyfikator zdefiniowany za pomocą atrybutu `mod` wskazuje, że chodzi o zachodnią część regionu.

[35] http://www.timeml.org

SpatialML umożliwia także określenie jakościowych relacji między miejscami. Schemat relacji został oparty o rachunek RCC8 [RCC1992] [CBGG1997] z drobną modyfikacją.[36] Oznaczenie wykorzystujące relacje dla frazy *teatr w Poznaniu* może mieć postać:

```
<PLACE id="1" form="NOM">teatr</PLACE>
<SIGNAL id="2">w</SIGNAL>
<PLACE country="PL" id="3" form="NAM">Poznaniu</PLACE>
<LINK source="1" target="3" signals="2" linkType="IN"/>
```

W powyższym przykładzie oznaczono dwa miejsca: teatr i Poznań. Miejsca te połączone są relacją `IN` (atrybut `linkType` znacznika `LINK`). Relacja ta została wyrażona za pomocą słowa *w* (znacznik `SIGNAL`).

Język SpatialML stanowi podstawę reprezentacji wiedzy przestrzennej w systemie QA rozwijanego w niniejszej pracy (patrz rozdział 5).

3.4. Bazowa wersja autorskiego systemu QA

W pracy magisterskiej [Wal2009] opracowałem prototyp polskiego systemu QA odpowiadającego na pytania na temat aktualnych wiadomości prasowych.[37] Jest to system o niesprecyzowanej dziedzinie, w którym skupiłem się na obsłudze pytań o czas i miejsce. Bazowa wersja systemu wykorzystuje powierzchniowe metody odpowiadania na pytania.

Bazą wiedzy systemu jest kolekcja artykułów prasowych pozyskiwanych automatycznie z popularnych serwisów plotkarskich oraz „newsowych". Artykuły zostały zaindeksowane za pomocą maszyny indeksującej Sphinx.[38] Wiadomości są pozyskiwane za pomocą specjalnych „wrapperów". Wrapper jest prostym skryptem pozwalającym na wyciągnięcie ze strony HTML interesujących treści (np. tytułu artykułu, treści artykułu, daty opublikowania itp.).[39] Baza jest aktualizowana codziennie, dzięki czemu system może odpowiadać na pytania dotyczące bieżących wydarzeń.

Pytanie przetwarzane jest w następujący sposób:

1. **Parsowanie pytania** — pytanie zamieniane jest na jego formalną reprezentację QQuery (patrz podrozdział 3.4.1). Wykorzystywany jest parser płytki Puddle[40]. Pytanie przetwarzane jest do QQuery za pomocą zbioru reguł.

[36] Relacje `TPP` i `NTTP` zastąpiono relacją `IN`, która odpowiada relacji `PP` rachunku RCC5.

[37] Adres serwisu: http://www.hipisek.pl

[38] http://sphinxsearch.com

[39] Część wrapperów została opracowana przez studentów: Przemysława Iwanka, Szymona Jóźwiakowskiego, Kamila Wylegały i Tomasza Śliwińskiego.

[40] Autor Leszek Manicki. Parser ten jest obecnie częścią pakietu PSI-Toolkit (http://psi-toolkit.wmi.amu.edu.pl/)

2. **Znalezienie źródeł odpowiedzi** — w bazie wiedzy szukane są dokumenty tematycznie powiązane z pytaniem. Spośród znalezionych kandydatów wybierane są fragmenty, które potencjalnie mogą zawierać odpowiedź.
3. **Pozyskanie odpowiedzi** — na znalezionych fragmentach i sparsowanym pytaniu uruchamiany jest zestaw metod odpowiadania (patrz podrozdział 3.4.2). Powstaje zbiór kandydatów odpowiedzi.
4. **Ocena odpowiedzi** — kandydaci zostają ocenieni i uszeregowani w kolejności. Odpowiedzi zostają wyświetlone użytkownikowi.

W procesie przetwarzania pytania wykorzystano narzędzia przetwarzania języka naturalnego z pakietu PSI-Toolkit [Jas2012]. Wykorzystano następujące elementy pakietu:

- tokenizator,
- segmenter,
- parser płytki Puddle.

3.4.1. Reprezentacja QQuery

Pytanie w systemie Hipisek jest reprezentowane za pomocą reprezentacji QQuery. Reprezentacja QQuery jest wzorowana na reprezentacji pytania QTarget używanej w systemie Webclopedia [LGH+2000]. QQuery składa się z następujących elementów:

- **typ pytania** — opisuje typ oczekiwanej odpowiedzi (np. pytanie o czas, miejsce, osobę),
- **temat pytania** — opisuje najważniejsze pojęcie w pytaniu,
- **akcja pytania** — opisuje stan lub czynność jaką wykonuje temat,
- **ograniczenia** — zawiera pozostałe słowa kluczowe występujące w pytaniu.

Wprowadzenie wyróżnionych pól tematu i akcji pytania motywowane było obsługą fokusu pytania z modelu Lehnert (patrz podrozdział 3.1.3).

Pomocniczym elementem QQuery są **frazy wyszukujące**, czyli fragmenty tekstu, w pobliżu których może znajdować się odpowiedź. Pojęcie to najlepiej ilustruje przykład reprezentacji pytania *Gdzie kot pogryzł psa?* za pomocą następującej struktury QQuery:

```
AnswerType: place
QQuery:
  topic: kot[kot]
  action: pogryzł[pogryźć]
  constraints: psa[pies]
Phrases:
  [kot pogryzł psa]
```

W powyższym przykładzie frazą wyszukującą jest wyrażenie *kot pogryzł psa*, ponieważ wystąpienie takiego wyrażenia w tekście źródłowym wskazuje na wysokie prawdopodobieństwo wystąpienia odpowiedzi.

3.4.2. Metody odpowiadania

Pythia Answerer

Metoda **Pythia Answerer** została opisana w pracy magisterskiej [Wal2009]. Jej rozbudowana wersja została przedstawiona w pracy [WJ2010]. Metoda bazuje na znalezieniu zdań pasujących do pytania. Badane jest pokrycie elementów QQuery przez słowa ze zdania-kandydata oraz zdań sąsiadujących. Dodatkowo wykorzystano własny moduł rozpoznawania jednostek nazwanych (ang. *Named Entity Recognition*, NER). Moduł NER zaimplementowany został z wykorzystaniem parsera płytkiego Puddle będącego częścią pakietu PSI-Toolkit. Zadaniem modułu NER jest sprawdzenie, czy w odpowiedzi znajduje się poszukiwany typ jednostki nazwanej.

Na przykład przetwarzając następujące pytanie:

> Gdzie kot pogryzł psa?

Jedną z odpowiedzi zwracanych przez metodę jest:

> Kot pogryzł psa na Wildzie. Agresywny kot pogryzł mojego psa! — takie nietypowe zgłoszenie otrzymał w piątek patrol Straży Miejskiej.

Odpowiedź została wybrana ze względu na wystąpienie jednostki nazwanej *Wilda*, która jest zgodna z oczekiwanym typem odpowiedzi (pytanie o miejsce).

Simple Search Answerer

Metoda **Simple Search Answerer** polega na transformacji pytania na szereg zapytań do tradycyjnej wyszukiwarki internetowej. Odpytana zostaje cała zaindeksowana baza wiedzy.

Szczególnym typem pytań przetwarzanych w ten sposób są **pytania rozstrzygnięcia**. Pytania tego typu zostały potraktowane jako szczególny przypadek wnioskowania w języku naturalnym (ang. *Natural Language Inference*, NLI).

Zadanie NLI polega na rozstrzygnięciu, czy wniosek sformułowany w języku naturalnym wynika bezpośrednio ze zbioru przesłanek, również sformułowanych w języku naturalnym. Zauważmy, że pytanie rozstrzygnięcia możemy potraktować jako wniosek zadania NLI. Zbiorem przesłanek staje się wtedy zbiór fragmentów dokumentów, które potencjalnie zawierają odpowiedź. W takim ujęciu zadanie QA staje się zadaniem NLI.

Załóżmy, że system QA przetwarza pytanie: *Czy w zeszłym roku przewidziano koniec świata?*. System znalazł następujący fragment dokumentu jako potencjalnie zawierający odpowiedź:[41]

> Według przepowiedni Majów 21.12.2012 nastąpi koniec świata. W miniony weekend w Dzień Dobry TVN Justyna Steczkowska postanowiła zabrać głos w tej sprawie. Artystka najwyraźniej długo nad tym myślała, bo to, co powiedziała w studio wprawiło w zdumienie nawet prowadzących show. Nie sądzę, żeby był koniec świata taki, że nagle nastanie ciemność.

Wystarczy teraz pokazać, że z powyższego fragmentu wynika wniosek: *w zeszłym roku przewidziano koniec świata*.

Korzystając z tej intuicji dostosowano metodę odpowiadania *Simple Search Answerer* do pytań rozstrzygnięcia. W tym celu wykorzystano prosty model *bag-of-words* opisany np. w pracy [Mac2009].

Metoda odpowiadania na pytania rozstrzygnięcia oparta o *Simple Search Answerer* jest metodą bazową dla metod odpowiadania opisanych w rozdziale 5.

3.4.3. Rozwój systemu

W pracy doktorskiej do systemu dodano następujące elementy:

- Bazę wiedzy przestrzennej, która powstała poprzez integrację dostępnych źródeł, z wykorzystaniem algorytmu ujednoznaczniania opisanego w rozdziale 4.
- Metodę odpowiadania na pytania rozstrzygnięcia z aspektem czasowym i przestrzennym wykorzystującą algorytm opisany w rozdziale 5.
- Narzędzia oznaczania i przetwarzania wiedzy przestrzennej i czasowej opisane w rozdziale 6.

Prezentacja aktualnej wersji systemu wraz z ewaluacją została zamieszczona w rozdziale 6.

[41] Źródło: http://www.lansik.pl/31370/koniec-swiata-nastapi-na-poziomie-energetycznym

Rozdział 4

Metoda zbierania wiedzy przestrzennej w systemie HipiSwot

W niniejszym rozdziale opisuję proces zbierania bazy wiedzy przestrzennej, który został zaimplementowany w autorskim systemie HipiSwot.

Głównym celem rozdziału jest opis autorskiego rozwiązania problemu ujednoznacznienia pojęć w bazie wiedzy. Problem ten występuje, gdy dwa obiekty mają identyczne nazwy, ale nazwy te odnoszą się do różnych pojęć. Na przykład nazwa *Drawa* oznacza zarówno rzekę będącą dopływem Noteci, jak i inną rzekę będącą dopływem Dunaju. Zaproponowane rozwiązanie problemu ujednoznacznienia polega na wykorzystaniu wnioskowania jakościowego (w szczególności wnioskowania przestrzennego).

Rozdział rozpoczynam od przedstawienia procesu zbierania wiedzy w systemie HipiSwot. Wyjaśniam w jaki sposób przechowywane są informacje zbierane przez system. Następnie przedstawiam problem niejednoznaczności pojęć zbieranych w bazie.

W dalszej części rozdziału opisuję autorski algorytm ujednoznaczniania oparty o wnioskowanie przestrzenne w rachunku RCC5. Na przykładach demonstruję działanie algorytmu, a następnie przeprowadzam jego analizę opisując jego własności i ograniczenia.

Następnie przedstawiam rozszerzenie algorytmu ujednoznaczniania, które polega na wykorzystaniu informacji ilościowych zbieranych przez system (np. informacje geopolityczne takie jak powierzchnia regionu, liczba ludności) i wykorzystaniu ich w procesie wnioskowania jakościowego.

W procesie ujednoznaczniania za pomocą relacji ilościowych biorę pod uwagę niespójność danych zbieranych przez źródła (np. różne źródła podają różną liczbę ludności danego miejsca). Wprowadzam trzy relacje między wartościami liczbowymi zbieranymi przez system HipiSwot: *equal*, *aprox* oraz *notequal* (interpretowane odpowiednio jako: *równe*, *około* oraz *różne*). Pokazuję, że relacje te można modelować w rachunku RCC8.

Rozdział kończy prezentacja wyników działania systemu zbierania bazy wiedzy. Prezentuję rozmiar zebranej bazy wiedzy, porównując bazową wersję systemu HipiSwot (bez mechanizmów ujednoznaczniania) z wersją rozbudowaną o mechanizmy

ujednoznaczniania. Przedstawiam również wyniki eksperymentów pomiaru szybkości działania algorytmów ujednoznaczniania.

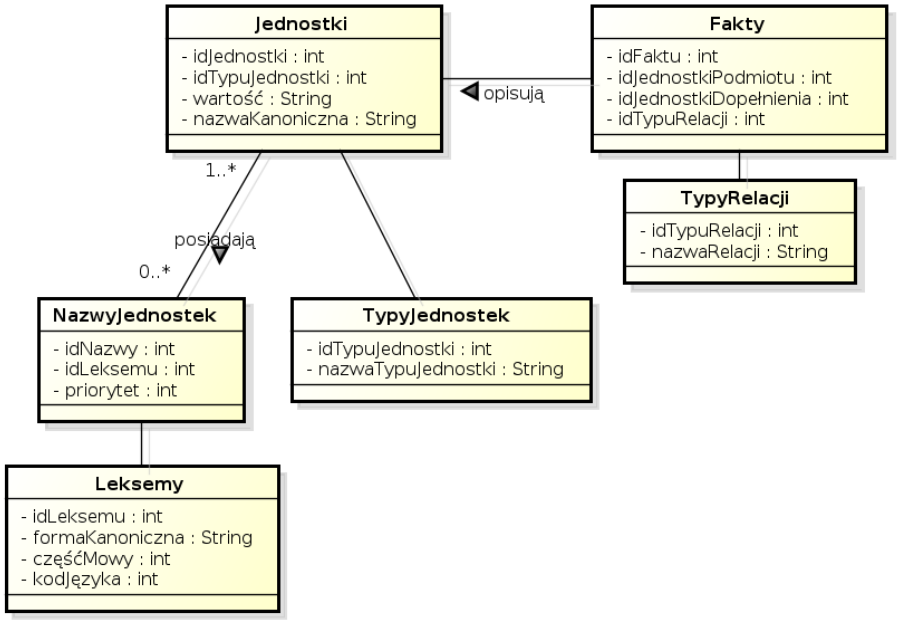
Rozdział jest rozwinięciem artykułu [WJ2011].

4.1. Charakterystyka bazy wiedzy

4.1.1. Budowa bazy wiedzy

Wykorzystywana w niniejszej rozprawie baza wiedzy ustrukturyzowanej jest uproszczoną ontologią (patrz podrozdział 3.2.1). W bazie przechowujemy **jednostki** i **fakty**.

Poglądowy schemat bazy wiedzy przedstawiony został na rysunku 4.1.



Rysunek 4.1. Poglądowy schemat bazy wiedzy wykorzystywanej w rozprawie

Jednostki

Jednostka jest odzwierciedleniem obiektu świata rzeczywistego (np. państwa, miasta, regionu) lub pojęcia abstrakcyjnego (np. długości geograficznej, liczby ludności). Atrybutami jednostki są:

- typ jednostki — określa kategorię jednostki, jednostka może mieć przyporządkowany dokładnie jeden typ,[1]
- nazwa kanoniczna — nazwa identyfikująca jednostkę,

[1] Opis typów jednostek używanych w systemach Hipisek i HipiSwot, wraz z przykładami, znajduje się w dodatku C.

- warianty nazw — zbiór nazw opcjonalnych jednostki (np. nazwy w innych językach),
- wartość jednostki (opcjonalne) — mierzalna wartość związana z jednostką.

Pojęcia **jednostki** i **jednostki nazwanej** (omawiane w podrozdziale 3.4.2) nie są tożsame. Pojęcie jednostki zostało wprowadzone na potrzeby pracy i jest wzorowane na **pojęciach** (ang. *concepts*) z których zbudowana jest klasyczna ontologia (patrz podrozdział 3.2). Jednostka nazwana jest sposobem wyrażenia jednostki w języku naturalnym (w szczególności w tekście). W szczególności jednostka może być reprezentowana przez różne jednostki nazwane. Na przykład jednostka *Warszawa [miasto]*, może być reprezentowana przez następujące jednostki nazwane: Warszawa (nazwa oficjalna), Varsovie (nazwa w języku francuskim).

Jednostki dzielimy na dwie grupy:

- **pojęcia** — odzwierciedlają jednostki abstrakcyjne np.: imiona, nazwiska, wartości liczbowe,
- **obiekty** — odzwierciedlają obiekty świata rzeczywistego np.: państwa, regiony, miasta.

Podział na grupy wyrażony jest za pomocą typu jednostki (patrz taksonomia typów w dodatku C). Jednostki, których typ jest podtypem potomnym *pojęcia*, należą do grupy pojęć. Analogicznie, jednostki, których typ jest podtypem potomnym *obiektu*, należą do grupy obiektów. Na przykład jednostka o typie *miasto* jest obiektem, bo typ *miasto* jest typem potomnym do typu *obiekt*. Pojęcia mogą mieć przyporządkowaną wartość (np. wartość liczbową).

Przykładem obiektu jest następująca jednostka reprezentująca stolicę Polski:

- nazwa kanoniczna: Warszawa,
- typ jednostki: miasto,
- warianty nazw: Warsaw, Varsovie.

Przykładem pojęcia jest jednostka reprezentująca liczbę ludności Warszawy (wydobyta z tekstu *1,7 miliona*):

- nazwa kanoniczna: 1,7 miliona,
- typ jednostki: liczba,
- wartość: 1700000.

Fakty

Fakty reprezentują relacje zachodzące między jednostkami. Jeden fakt modeluje informację o powiązaniu relacją dwóch jednostek.

Fakt składa się z:

- podmiotu i dopełnienia — odnośników do jednostek, między którymi zachodzi relacja,
- typu relacji.

Typy relacji zachodzących między obiektami tworzą taksonomię.[2] Na potrzeby zbierania wiedzy przestrzennej używamy następujących typów relacji:

- **relacje przestrzenne**:
  - jest położony w — *is located in*, region A znajduje się w pełni w regionie B,
  - jest częściowo położony w — *is partially located in*, region A znajduje się w całości w regionie B lub pokrywa się z regionem B,
  - częściowo pokrywa się z — *overlaps*, region A częściowo pokrywa się z regionem B,
- **relacje ilościowe** — relacje geograficzne i geopolityczne:
  - liczba ludności,
  - stolica,
  - długość geograficzna,
  - szerokość geograficzna,
  - wysokość nad poziomem morza,
  - powierzchnia,
  - długość.

Na przykład, aby zakodować informację, że: *Warszawa znajduje się w Polsce*, używamy następującego faktu: *(Warszawa [miasto], jest położony w, Polska [państwo])*. Podobnie wiedzę ilościową o liczbie ludności Warszawy przechowujemy za pomocą faktu: *(Warszawa [miasto], liczba ludności, 1700000 [liczba])*.

Założenie o zamkniętości świata

Zbierając bazę wiedzy przyjęto założenie o zamkniętości świata (ang. *Closed World Assumption*) [Rei1977]. Zakładamy, że w bazie wiedzy zostały zawarte wszystkie fakty pozytywne. Fakty negatywne (np. o rozłączności dwóch regionów) wynikają z niemożności wywnioskowania faktu pozytywnego. Oznacza to, że jeśli nie można dowieść, że regiony A i B są w jednej z wykorzystywanych typów relacji przestrzennych, to są one rozłączne.

Rozpatrzmy przykład rzeki *Drawa* (dopływ Dunaju), która przepływa przez następujące państwa: Austrię, Chorwację, Słowenię, Węgry i Włochy. W bazie wiedzy przechowujemy pięć faktów:

- *(Drawa [rzeka], częściowo pokrywa się z, Austria [państwo])*,
- *(Drawa [rzeka], częściowo pokrywa się z, Chorwacja [państwo])*,

[2] Opis typów relacji używanych w systemach Hipisek i HipiSwot, wraz z przykładami, znajduje się w dodatku D.

- *(Drawa [rzeka], częściowo pokrywa się z, Słowenia [państwo])*,
- *(Drawa [rzeka], częściowo pokrywa się z, Węgry [państwo])*,
- *(Drawa [rzeka], częściowo pokrywa się z, Włochy [państwo])*.

Korzystając z założenia o zamkniętości świata, zakładamy że wymienione pięć państw tworzy zbiór **wszystkich państw** przez które przepływa rzeka. Implikuje to, że każdy region (który nie pokrywa się z lub nie jest położony w jednym z pięciu wymienionych państw) jest traktowany jako **rozłączny** z rzeką Drawą.

Monotoniczność faktów

W procesie zbierania faktów podmiotem jest zawsze obiekt „mniejszy". Wielkość obiektów ma charakter umowny i wynika z taksonomii typów jednostek. Podstawowym kryterium wielkości jest możliwość położenia obiektu w innym (jeśli obiekt A może być położony w całości w obiekcie B, to obiekt A jest mniejszy). W ogólności jednak kolejność jednostek w faktach ma charakter konwencji, do której należy się dostosować opracowując źródło na potrzeby systemu HipiSwot.

Jako przykład rozpatrzmy następujący fakt: *(Drawa [rzeka], częściowo pokrywa się z, Austria [państwo])*. Fakt ten można równoważnie zapisać w postaci: *(Austria [państwo], częściowo pokrywa się z, Drawa [rzeka])*, ale ze względu na zapewnienie kryterium monotoniczności nie jest to poprawny fakt w bazie wiedzy (przyjmujemy bowiem, że typ *rzeka* jest typem obiektu „mniejszego" niż obiektu o typie *państwo*).

Zakładamy, że w bazie wiedzy nie przechowujemy faktów, które mogą zostać wywnioskowane z innych. Na przykład załóżmy, że w procesie zbierania bazy wiedzy dodamy do bazy wiedzy następujące fakty:

- *(Poznań [miasto], jest położony w, województwo Wielkopolskie [jednostka administracyjna pierwszego rzędu])*,
- *(województwo Wielkopolskie [jednostka administracyjna pierwszego rzędu], jest położony w, Polska [państwo])*.

Dodanie wyżej wymienionych faktów **nie implikuje** dodania faktu, że *(Poznań [miasto], jest położony w, Polska [państwo])*. Fakt o położeniu Poznania w Polsce można wywnioskować z dwóch wyżej wymienionych faktów poprzez proste złożenie relacji.

Monotoniczność bazy wiedzy ma na celu zmniejszenie przestrzeni przeszukiwania dla algorytmów ujednoznaczniających opisanych w dalszej części rozdziału.

4.1.2. Proces zbierania wiedzy w systemie HipiSwot

Źródła danych

Tworząc bazę wiedzy przestrzennej przyjąłem metodykę integracji dostępnych źródeł opisaną w podrozdziale 3.2.2.

System integruje dane z następujących źródeł:

- **Źródła ustrukturyzowane**: baza wiedzy Geonames, baza wiedzy DBPedia, rejestr TERYT.
- **Źródła częściowo ustrukturyzowane**: wybrane strony encyklopedii internetowej Wikipedia (np.: wykazy państw świata, wykazy rzek), serwisy informacji turystycznej: Miasteria[3] oraz WartoZwiedzic.pl[4], wykazy informacji i listy miast dostępne w internecie (np. serwis MongoBay[5]).
- **Źródła wydobywane**: informacje o położeniu miejsc opisywanych w encyklopedii internetowej Wikipedia pozyskane na podstawie kategorii artykułu opisującego dane miejsce.[6]

Zadanie systemu HipiSwot

Każde źródło przetwarzane przez system HipiSwot dodaje zbiór faktów. Zbiór faktów składa się z następujących elementów:

- jednostki bazowej — jednostki, która jest podmiotem wszystkich faktów ze zbioru faktów,
- tablicy faktów — niepustej tablicy zawierającej fakty, których podmiotem jest jednostka bazowa.

Na przykład przetwarzając dane serwisu Geonames na temat Poznania, do systemu HipiSwot trafia następujący zbiór faktów:

```
Poznań [miasto]
- jest położony w, Polska [państwo]
- długość geograficzna, 16,92993 [liczba]
- szerokość geograficzna, 52,40692 [liczba]
- liczba ludności, 570352 [liczba]
```

W powyższym przykładzie jednostką bazową jest jednostka *Poznań [miasto]*. Tablica faktów zawiera cztery elementy reprezentujące położenie w Polsce, współrzędne geograficzne oraz liczbę ludności.

[3] http://www.miasteria.pl

[4] http://www.wartozwiedzic.pl

[5] http://world.mongabay.com/polish/population/pl.html

[6] Na przykład artykuł o *Akademii Lubrańskiego* został na polskiej Wikipedii przyporządkowany do kategorii *Zabytki Poznania*. Na tej podstawie do bazy wiedzy dodany został fakt, że: *(Akademia Lubrańskiego [miejsce], jest położony w, Poznań [miasto])*.

Zbiór faktów zawiera nazwy jednostek. **Głównym zadaniem** systemu HipiSwot jest połączenie nazw ze zbioru faktów z jednostkami znajdującymi się w bazie wiedzy lub dodanie do bazy nowych jednostek, w przypadku gdy połączenie takie nie jest możliwe.

4.1.3. Schemat przetwarzania źródła

Źródła faktów są przetwarzane przez system w kolejności odpowiadającej jakości źródła (ocenianej subiektywnie). Schemat przetwarzania źródła można zilustrować za pomocą następujących kroków:

1. Dla każdego zbioru faktów F dostarczanego przez źródło wykonaj:
   a) Sprawdź czy jednostki występujące w F są już zapisane w bazie wiedzy, jeśli tak, to utwórz z nich zbiór kandydatów do połączenia ze zbiorem faktów.
   b) Jeśli jednostka z F nie ma kandydata do połączenia, to oznacz ją jako nową jednostkę do dodania w bazie.
   c) Wybierz najlepsze połączenie (interpretację) jednostek ze zbioru faktów z jednostkami występującymi w bazie wiedzy.
   d) Zapisz tak zinterpretowany zbiór faktów w bazie wiedzy (dodając nowe jednostki jeśli takie wystąpiły).

Kluczowym krokiem tego schematu jest wybór najlepszego połączenia jednostek ze zbioru faktów z jednostkami zapisanymi w bazie. Zauważmy, że w niektórych przypadkach zbiór kandydatów może zawierać więcej niż jedną jednostkę (w przypadku nazw niejednoznacznych). Związany jest z tym **problem ujednoznaczniania pojęć** omówiony w podrozdziale 4.1.4. Przez najlepsze połączenie rozumiemy w tym przypadku połączenie, które łączy tylko te jednostki, które odpowiadają tym samym obiektom lub pojęciom, które reprezentują.

Nawiązując do schematu bazy wiedzy przedstawionego w podrozdziale 4.1.1, zadanie wybrania najlepszego połączenia można opisać jako problem przypisania identyfikatorów jednostek z bazy wiedzy do nazw jednostek znajdujących się w przetwarzanym zbiorze faktów (w przypadku gdy baza wiedzy zawiera już odpowiednie jednostki) lub przypisania identyfikatora pustego — NULL (w przypadku gdy baza wiedzy nie zawiera odpowiedniej jednostki).

4.1.4. Problem ujednoznacznienia pojęć

Nazwa obiektu geograficznego nie jest jednoznacznym identyfikatorem. Wiele nazw jest wieloznacznych i może odnosić się do różnych obiektów geograficznych. Przykładowymi nazwami wieloznacznymi są:

- *Poznań* (może odnosić się do stolicy Wielkopolski lub wsi w okolicach Lublina),

- *Cambridge* (może odnosić się do miasta w Anglii lub miasta w Stanach Zjednoczonych, nazwa ta funkcjonuje także jako potoczne określenie *Uniwersytetu w Cambridge*),
- *Praga* (może odnosić się do dzielnicy Warszawy lub stolicy Czech),
- *Drawa* (może odnosić się do jednego z dopływów Noteci, innej europejskiej rzeki będącej dopływem Dunaju lub wsi w województwie warmińsko-mazurskim).

Jednym ze sposobów wyeliminowania wieloznaczności jest użycie typu jednostki. Heurystyka ujednoznacznienia polega w tym przypadku na rozróżnieniu jednostek, których typy są niezgodne. **Niezgodnymi typami** są takie dwa typy, które są różne oraz dla których nie jest prawdą, że jeden jest potomkiem lub przodkiem drugiego w taksonomii typów. **Zgodne typy** to takie, które nie są niezgodne. Zgodnymi typami są na przykład typy *obszar zamieszkały* oraz *miasto* (ponieważ typ *miasto* jest potomkiem typu *obszar zamieszkały* w taksonomii typów jednostek), natomiast niezgodnymi typami są typy *dzielnica* oraz *miasto* (które reprezentują różne gałęzie taksonomii typów jednostek).

Prezentowany wyżej przykład pokazuje jednak, że heurystyka wykorzystująca typ jednostki jest niewystarczająca do ujednoznacznienia nazw. W powyższym przykładzie heurystyka pozwoliłaby co prawda ujednoznacznić nazwę *Poznań* (typy *wieś* i *miasto* są niezgodne) oraz nazwę *Praga* (typy *dzielnica* i *miasto* są niezgodne), ale pozostałe dwie nazwy (*Cambridge* oraz *Drawa*) nie zostałyby w pełni ujednoznacznione.

W podrozdziałach 4.2 i 4.3 przedstawiam autorską metodę ujednoznacznia pojęć wykorzystującą wnioskowanie jakościowe. Metoda ta (w połączeniu z heurystyką wykorzystującą typ jednostki) pozwala w szczególności na ujednoznacznienie wszystkich nazw wymienionych w przykładzie prezentowanym na początku podrozdziału 4.1.4.

Procedura ujednoznaczniania wykorzystywana jest w kroku wyboru najlepszego połączenia jednostek ze zbioru faktów z jednostkami znajdującymi się już w bazie schematu przetwarzania zbioru faktów przedstawionego w podrozdziale 4.1.3.

4.2. Algorytm ujednoznaczniania pojęć wykorzystujący rachunek RCC5

4.2.1. Opis algorytmu

Danymi wejściowymi do algorytmu ujednoznacznienia jest zbiór faktów. Wynikiem działania algorytmu jest **interpretacja** zbioru faktów.

Definicja 4.1. Interpretacją zbioru faktów (w skrócie interpretacją) nazywamy przyporządkowanie jednostek występujących w zbiorze faktów do jednostek występujących w bazie wiedzy lub do jednostek pustych.

W interpretacji zbioru faktów mogą występować wartości `NULL` oznaczające **jednostkę pustą**. Jednostka pusta wskazuje na konieczność dodania nowej jednostki do bazy wiedzy.

Algorytm ujednoznaczniania jest przedstawiony na listingu 4.1.

Algorytm 4.1: WybierzInterpretację

```
   Data: Zbiór faktów Z
   Result: Interpretacja
1  begin
2  |  znajdź zbiór interpretacji I jednostek ze zbioru faktów Z ;
3  |  foreach interpretacja i ze zbioru I do
4  |  |  if CzyInterpretacjaNiesprzeczna(i, Z) then
5  |  |  |  return i
6  |  |  end
7  |  end
   |  /* wszystkie interpretacje są sprzeczne                    */
   |  /* nie można dodać do bazy                                 */
8  |  return NULL
9  end
```

W pierwszym kroku algorytmu ujednoznacznienia budowana jest kolekcja interpretacji, która zawiera wszystkie interpretacje przetwarzanego zbioru faktów. Kolekcja interpretacji powstaje poprzez wyszukanie w bazie wiedzy wszystkich jednostek, które mają tę samą nazwę kanoniczną lub jeden z wariantów nazw, jak jednostki występujące w ujednoznacznianym zbiorze faktów Z. Przy tworzeniu kolekcji interpretacji wykorzystywana jest heurystyka ujednoznacznienia za pomocą zgodności typów jednostek.[7]

Kolekcja interpretacji uzupełniona jest o jednostki puste.[8]. Kolejność występowania jednostek w kolekcji interpretacji ma znaczenie. Na końcu kolekcji znajdują się zawsze jednostki puste (wprowadzamy nowe jednostki tylko w ostateczności).

Ze względów technicznych zabronione jest interpretowanie jednostek będących pojęciami. Pojęcia będą zawsze interpretowane jako jednostki puste (zawsze dodamy je jako nowe jednostki w bazie).

Na przykład dla zbioru faktów składającego się z jednostki bazowej *Drawa [rzeka]* i pojedynczego faktu *(Drawa [rzeka], jest położony w, Polska [państwo])* oraz dla bazy wiedzy zawierającej jedną jednostkę *Polska [państwo] id:1* kolekcja interpre-

[7] Patrz podrozdział 4.1.4.

[8] Jednostki puste oznaczamy słowem kluczowym `NULL`.

tacji składa się z następujących elementów (w przyporządkowaniu podano numery identyfikatorów jednostek w bazie):

1. (Drawa [rzeka] → NULL, Polska [państwo] → 1),
2. (Drawa [rzeka] → NULL, Polska [państwo] → NULL),

Definicja 4.2. Interpretację i dla zbioru faktów Z nazywamy **niesprzeczną** z bazą wiedzy B wtedy i tylko wtedy gdy dodanie zbioru faktów Z do bazy wiedzy zgodnie z interpretacją i nie powoduje powstania sprzeczności w bazie wiedzy. W przeciwnym przypadku mówimy, że interpretacja jest **sprzeczna**.

Dla każdej interpretacji z kolekcji interpretacji sprawdzana jest jej niesprzeczność z całą dostępną bazą wiedzy. Pierwsza niesprzeczna z bazą wiedzy interpretacja jest zwracana jako poprawne ujednoznacznienie. W przypadku, gdy wszystkie interpretacje są sprzeczne, zwracana jest wartość pusta, która oznacza niemożność dodania zbioru faktów do bazy wiedzy.

Sprawdzenie niesprzeczności interpretacji i wykonujemy wykorzystując wnioskowanie jakościowe, za pomocą funkcji *CzyInterpretacjaNiesprzeczna* przedstawionej na listingu 4.2.

Algorytm 4.2: CzyInterpretacjaNiesprzeczna

```
Data: Interpretacja i, zbiór faktów Z
Result: Wartość logiczna
1 begin
2 |  F ← zbiór faktów Z zinterpretowany przez i ;
3 |  N ← StworzSiecOgraniczen(F, RCC5) ;
4 |  if PathConsistent(N) then
5 |  |  return TRUE
6 |  else
7 |  |  return FALSE
8 |  end
9 end
```

Funkcja *CzyInterpretacjaNiesprzeczna* wykorzystuje rachunki ograniczeń do stworzenia sieci ograniczeń (patrz podrozdział 2.1.2). Następnie w metodzie *PathConsistent* wykorzystywany jest algorytm PC (patrz podrozdział 2.1.5 oraz listing 2.2). Metoda *PathConsistent* zwraca:

- wartość logiczną *prawda*, gdy algorytm PC nie wyprodukuje relacji pustej (przyjmujemy wtedy, że wejściowa sieć ograniczeń jest niesprzeczna),
- wartość logiczną *fałsz*, gdy algorytm PC wyprodukuje relację pustą (przyjmujemy wtedy, że wejściowa sieć ograniczeń jest sprzeczna).

W celu utworzenia sieci ograniczeń zbiór faktów musi zostać zinterpretowany w wybranym rachunku. Tylko niektóre fakty są modelowane przez wybrany rachunek (np. za pomocą rachunku RCC5 nie jest modelowany fakt o typie relacji *liczba ludności*, ponieważ taki fakt nie dotyczy relacji przestrzennych obiektów). Aby określić, które fakty mają być modelowane przez wybrany rachunek, wprowadzimy pojęcie **typu relacji modelowanej przez rachunek**.

Definicja 4.3. Typ relacji modelowanej przez rachunek R to taki typ relacji, który jest równoważny pewnej relacji r z rachunku R.

Definicja 4.4. Fakt jest **modelowany przez rachunek** R, jeśli jego typ relacji jest modelowany przez rachunek R.

Na przykład typ relacji *jest częściowo położony w* jest modelowany przez rachunek RCC5, ponieważ jest równoważny relacji $\{PO, PP\}$. W dalszej części podrozdziału ograniczymy się do modelowania typów relacji przestrzennych za pomocą rachunku RCC5.

Typy relacji modelujemy w rachunku RCC5 w następujący sposób:[9]

- *jest położony w* $\rightarrow PP$,
- *jest częściowo położony w* $\rightarrow \{PP, PO\}$,
- *częściowo pokrywa się z* $\rightarrow PO$.

Modelując jednostki przestrzenne w rachunku RCC5 zakładamy, że jednostki są przybliżone za pomocą regionów w euklidesowej przestrzeni topologicznej. Podejście tego typu jest opisane na przykład w pracy [Gab2009].

Wykorzystując fakty, których typy relacji są modelowane przez rachunek RCC5 (czyli relacje przestrzenne), tworzymy sieć ograniczeń. Proces ten realizuje funkcja przedstawiona na listingu 4.3.

Algorytm tworzenia sieci ograniczeń polega na dodawaniu do sieci ograniczeń faktów pochodzących z bazy wiedzy (modelowanych w danym rachunku), tak długo dopóki sieć ograniczeń nie jest stabilna. Sieć jest stabilna wtedy, gdy w pojedynczym przebiegu pętli algorytmu 4.3 nie dodano nowych jednostek do sieci (zbiór $NEWENTINTIES$ jest pusty).

W sieci ograniczeń jednostki (podmiot lub dopełnienie faktu) zamieniane są na wierzchołki sieci. Relacje między jednostkami zamieniane są na krawędzie. Przyjmujemy, że jeśli dwa wierzchołki nie są połączone krawędzią, to zachodzi między nimi relacja uniwersalna danego rachunku.

Tworząc sieć ograniczeń wykorzystujemy semantykę typów jednostek do stworzenia relacji danego rachunku. Transformacja semantyki typów na relacje danego rachunku jest wykonywana za pomocą **reguł semantycznych**. Przykładowa reguła

[9] Niewymienione typy relacji nie są modelowane przez rachunek RCC5.

Algorytm 4.3: StworzSiecOgraniczen

```
   Data: zbiór faktów F, rachunek R
   Result: Sieć ograniczeń N
 1 begin
 2     T ← F (pomocniczy zbiór faktów);
 3     N ← NULL (sieć wynikowa);
 4     repeat
 5         NEWENTINTIES ← NULL ;
 6         foreach fakt f w F do
 7             if f jest modelowany przez rachunek R then
 8                 e ← zamień typ relacji f na relację rachunku R ;
 9                 dodaj podmiot lub dopełnienienie f do NEWENTINTIES
                   (chyba, że jednostka należy już do N);
10                 dodaj e do N ;
11             end
12         end
13         foreach jednostka j ze zbioru NEWENTINTIES do
14             NEWFACTS ← znajdź w bazie wiedzy wszystkie fakty, które
               mają ten sam podmiot co j ;
15             dodaj NEWFACTS do T ;
16         end
17         F ← T ;
18         T ← NULL ;
19     until NEWENTINTIES == NULL;
       /* Uruchomienie reguł semantycznych                              */
20     foreach para jednostek (e1, e2) z N, które nie są połączone w N do
21         spróbuj połączyć (e1, e2) korzystając z reguł semantycznych
22     end
23 end
```

semantyczna (dla rachunku RCC5) określa na przykład, że dwie jednostki o typie *państwo* są rozłączne (ponieważ nie ma dwóch państw o pokrywających się terytoriach).

Reguły semantyczne działają wyłącznie na typach porównywanych jednostek bądź ich wartościach. W szczególności reguły wykonują następujące operacje:

- przypisanie parze typów jednostek zdefiniowanej relacji (z uwzględnieniem taksonomii typów), np.: dwie jednostki o typie państwo są rozłączne,
- przypisanie zdefiniowanej relacji na podstawie porównania wartości, np.: daty o wartości roku odpowiednio 2012 oraz 2013 są w relacji *poprzedza*.[10]

Aby zachować pełność algorytmu zakładamy, że reguły semantyczne dotyczą tylko wybranego podzbioru relacji rachunku RCC5, mianowicie:[11]

[10] O reprezentacji wiedzy czasowej napisano szerzej w podrozdziale 5.2.3.

[11] W implementacji algorytmu w systemie HipiSwot użyto wyłącznie relacji `DR`. Reguły seman-

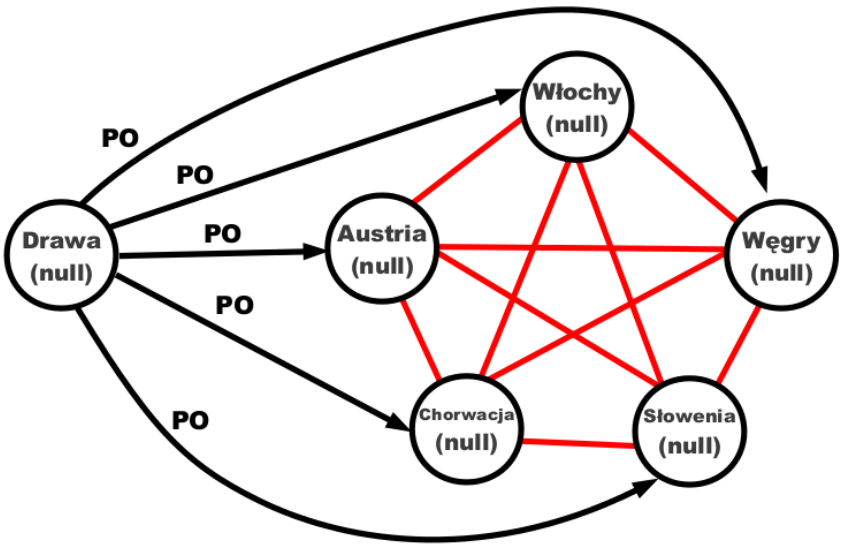
Uwaga 4.1. Tworząc reguły semantyczne używamy jedynie relacji należących do zbioru $\hat{H}_5$ (tzn. podklasy podatnej rachunku RCC5).

Powyższe założenie ma znaczenie w analizie algorytmu ujednoznaczniania (patrz podrozdział 4.2.3).

Rozpatrzmy następujący przykład. Do pustej bazy wiedzy dodajemy położenie rzeki Drawa, która przepływa przez pięć państw: Austrię, Chorwację, Słowenię, Węgry i Włochy. Przetwarzany zbiór faktów wygląda następująco:

```
Drawa [rzeka]
- częściowo pokrywa się˜z, Austria [państwo]
- częściowo pokrywa się˜z, Chorwacja [państwo]
- częściowo pokrywa się˜z, Słowenia [państwo]
- częściowo pokrywa się˜z, Węgry [państwo]
- częściowo pokrywa się˜z, Włochy [państwo]
```

Ponieważ baza wiedzy jest pusta, wszystkie jednostki są interpretowane jako nowe jednostki. Sieć ograniczeń zawiera sześć wierzchołków reprezentujących pięć państw oraz rzekę. Relacje *częściowo pokrywa się z* zostały przetransformowane do relacji `PO` rachunku RCC5. Reguły semantyczne dodały relacje `DR` między każdymi dwoma państwami. Wynikowa sieć ograniczeń zawiera sześć wierzchołków i została przedstawiona na rysunku 4.2. Relacja rozłączności regionów (`DR`) jest na rysunku oznaczona kolorem czerwonym.



Rysunek 4.2. Przykładowa sieć ograniczeń dla rzeki Drawy przepływającej przez Austrię, Chorwację, Słowenię, Węgry i Włochy

tyczne w tej implementacji ograniczają się do stwierdzenia, że pewne obiekty danych dwóch typów są rozłączne. Jednakże na potrzeby analizy algorytmu wystarczające jest ogólniejsze założenie podane wyżej.

4.2.2. Przykład działania algorytmu

Przykład 4.1: Dodanie faktów do pustej bazy wiedzy

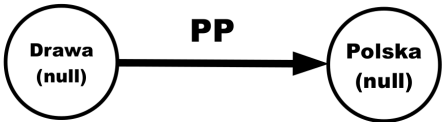
Rozpatrzmy przykład dodawania następującego zbioru faktów do pustej bazy wiedzy:

```
Drawa [rzeka]
- jest położony w, Polska [państwo]
- długość, 185,9 km [wartość metryczna]
```

W pierwszym kroku tworzona jest kolekcja interpretacji. Ponieważ baza wiedzy jest pusta, kolekcja interpretacji zawiera tylko jeden element:

- (Drawa [rzeka] → NULL, Polska [państwo] → NULL, 185,9 km [wartość metryczna] → NULL),

Następnie sprawdzana jest niesprzeczność interpretacji z bazą wiedzy. Tworzona jest sieć ograniczeń. Zbiór faktów zawiera tylko jeden fakt modelowany przez rachunek RCC5: *(Drawa [rzeka], jest położony w, Polska [państwo])*. Fakt ten jest zamieniany na relację `PP` w tworzonej sieci ograniczeń. Ponieważ baza wiedzy jest pusta, do sieci nie zostają dołączone żadne inne relacje. Reguły semantyczne nie zostają uruchomione (jedyne jednostki w sieci są już połączone krawędzią). Wynikowa sieć ograniczeń jest przedstawiona na rysunku 4.3



Rysunek 4.3. Wynikowa sieć ograniczeń dla rzeki Drawy z przykładu 4.1

Stworzona sieć ograniczeń nie zawiera relacji pustej, dlatego interpretacja zostaje zaakceptowana przez system. Jednostki i fakty zostają zapisane w bazie. Do bazy wiedzy dodane zostają następujące nowe jednostki (w nawiasach podano identyfikatory jednostek w bazie wiedzy):

- *Drawa [rzeka] (id:1)*,
- *Polska [państwo] (id:2)*,
- *185,9 [wartość metryczna] (id:3)*.

W bazie wiedzy zapisane zostają następujące fakty:

- *(Drawa [rzeka] (id:1), jest położony w, Polska [państwo] (id:2))*,
- *(Drawa [rzeka] (id:1), długość, 185,9 [wartość metryczna] (id:3))*.

Przykład 4.2: Dodanie niejednoznacznych faktów

Rozpatrzmy przykład dodania do bazy wiedzy (będącej wynikiem Przykładu 4.1.) zbioru faktów dotyczących innej rzeki Drawy, która przepływa przez następujące państwa: Austrię, Chorwację, Słowenię, Węgry i Włochy. Dla przejrzystości przykładu rozpatrzymy tylko jeden z faktów: *(Drawa [rzeka], częściowo pokrywa się z, Austria [państwo])*.

Przetwarzany zbiór faktów wygląda następująco:[12]

```
Drawa [rzeka]
- częściowo pokrywa się~z, Austria [państwo]
```

W bazie wiedzy znajduje się jedna jednostka o takiej samej nazwie kanonicznej (Drawa) i zgodnym typie (rzeka) jak jednostka bazowa zaprezentowanego wyżej zbioru faktów. Dlatego utworzona kolekcja interpretacji składa się z dwóch elementów:

- (Drawa [rzeka] → 1, Austria [państwo] → NULL),
- (Drawa [rzeka] → NULL, Austria [państwo] → NULL),

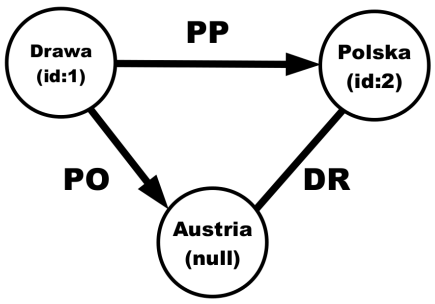
Pierwsza interpretacja z zaprezentowanej wyżej kolekcji interpretacji oznacza próbę ujednoznacznienia rzeki Drawy, z aktualnie przetwarzanego zbioru faktów, z rzeką Drawą, która już znajduje się w bazie wiedzy. Druga interpretacja oznacza wprowadzenie rzeki Drawy jako nowej jednostki. W obydwu przypadkach jednostka *Austria [państwo]* wprowadzana jest do bazy wiedzy jako nowa jednostka.

W kolejnym kroku następuje sprawdzenie niesprzeczności interpretacji z bazą wiedzy. W tym celu tworzona jest sieć ograniczeń. Pierwsza przetwarzana interpretacja zawiera połączenie jednostki *Drawa [rzeka]* z jednostką znajdującą się w bazie wiedzy. Wstępna sieć ograniczeń (przed wczytaniem faktów z bazy) zawiera dwa wierzchołki reprezentujące jednostkę z bazy wiedzy (*Drawa [rzeka]*) oraz nową jednostkę (*Austria [państwo]*). Wierzchołki połączone są relacją *PO*.

W kolejnym kroku tworzenia sieci ograniczeń dodawane są fakty z bazy wiedzy. W bazie wiedzy znajduje się następujący fakt na temat rzeki Drawy *(Drawa [rzeka] id:1, jest położony w, Polska [państwo] id:2)*. Fakt ten jest modelowany przez rachunek RCC5, dlatego zostaje dodany do sieci ograniczeń. Pętla wyszukująca nowe jednostki z bazy wiedzy zostaje przerwana, bo baza wiedzy nie zawiera więcej faktów modelowanych w rachunku RCC5, których podmiotem jest dodana w tym przebiegu pętli jednostka *Polska [miasto]*.

Reguły semantyczne zostają wykorzystane do połączenia jednostek: *Polska [państwo]* oraz *Austria [państwo]* relacją `DR` (zakładamy, że każde dwa państwa są rozłączne). Sieć wynikową przedstawiono na rysunku 4.4.

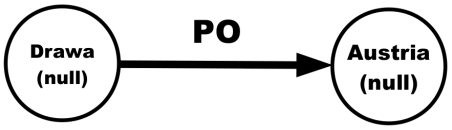
[12] Należy zwrócić uwagę, że tym razem jednostki połączone są relacją *częściowo pokrywa się z*. Relacja ta została zdefiniowana na poziomie źródła danych przez programistę przygotowującego źródło do przetworzenia przez system HipiSwot.



Rysunek 4.4. Wynikowa sieć ograniczeń dla rzeki Drawy z przykładu 4.2 (pierwsza interpretacja)

Dla przetwarzanej interpretacji sieć ograniczeń okazuje się być sprzeczna. Wykorzystując bowiem złożenie relacji między wierzchołkami: *Drawa*, *Polska*, *Austria* $PP \circ DR$ otrzymujemy relację DR. Tymczasem bezpośrednim etykietowaniem między wierzchołkami *Drawa* oraz *Austria* jest relacja PO. Iloczyn tych relacji daje relację pustą. Interpretacja łącząca rzekę Drawa, z rzeką już dodaną do bazy wiedzy zostaje odrzucona.

Rozpoczyna się proces sprawdzenia niesprzeczności drugiej interpretacji, w której rzeka Drawa jest dodawana jako nowa jednostka. Tym razem tworząc sieć ograniczeń nie zostaje dodany fakt, o położeniu rzeki w Polsce (ponieważ nazwa Drawa, w tej interpretacji, oznacza inną jednostkę). Wynikowa sieć ograniczeń zawiera więc wyłącznie dwie jednostki występujące w przetwarzanym zbiorze faktów. Sieć tą przedstawiono na rysunku 4.5.



Rysunek 4.5. Wynikowa sieć ograniczeń dla rzeki Drawy z przykładu 4.2 (druga interpretacja)

Sieć ograniczeń dla interpretacji drugiej (w której wszystkie jednostki są dodawane jako nowe), po zastosowaniu algorytmu PC nie zawiera relacji pustej, dlatego interpretacja zostaje użyta do dodania zbioru faktów do bazy wiedzy. Do bazy wiedzy zostają dodane dwie nowe jednostki i jeden fakt.

Prezentowany przykład pokazuje, że dzięki zastosowaniu wnioskowania przestrzennego system miał możliwość rozróżnienia dwóch rzek, których nazwy są identyczne.

4.2.3. Analiza algorytmu

Algorytm PC jako kryterium ujednoznaczniania

Sieć ograniczeń tworzona przez funkcję *StworzSiecOgraniczen* i używana w algorytmie ujednoznaczniania posiada następującą własność:

Własność algorytmu 4.1. *Dla dowolnej pary wierzchołków v i w sieci ograniczeń N zbudowanej przez algorytm StworzSiecOgraniczen przedstawiony na listingu 4.3 etykieta krawędzi między v a w jest albo:*

1. *relacją uniwersalną (brak wiedzy na temat relacji między jednostkami odpowiadającymi wierzchołkom v i w),*
2. *relacją pochodzącą z modelowania w rachunku RCC5 pewnego faktu między obiektami v i w, bądź relacją odwrotną do modelowanej,*
3. *relacją pochodzącą z reguł semantycznych.*

Z powyższej własności wynika następujący wniosek:

Wniosek 4.2. *Sieć ograniczeń N modelująca wiedzę przestrzenną, używany w algorytmie ujednoznaczniania ma krawędzie etykietowane wyłącznie relacjami ze zbioru $\hat{H}_5$ (będącego jedną z podklas podatnych rachunku RCC5).*

Dowód. Korzystając z własności 4.1 wystarczy pokazać, że wszystkie typy relacji wymienione we własności należą do zbioru relacji $\hat{H}_5$.

Rozpatrzmy trzy przypadki etykiety krawędzi sieci ograniczeń N:

1. **Krawędź etykietowana jest relacją uniwersalną.**
   Relacja uniwersalna należy do zbioru $\hat{H}_5$.
2. **Krawędź etykietowana jest relacją pochodzącą z modelowania w rachunku RCC5 pewnej relacji faktu między obiektami v i w, bądź relacją odwrotną do modelującej.**
   Należy pokazać, że typy relacji modelowanych przez rachunek RCC5 i odwrotności relacji modelującej dany typ relacji należą do $\hat{H}_5$. Mamy trzy typy relacji modelowanych przez RCC5:
   - *jest położony w*, modelowana relacją PP (odwrotność relacji PPI),
   - *jest częściowo położony w*, modelowana relacją $\{PP, PO\}$ (odwrotność relacji $\{PPI, PO\}$),
   - *częściowo pokrywa się z*, modelowana relacją PO (odwrotność relacji PO).

   Wszystkie wymienione wyżej relacje należą do zbioru $\hat{H}_5$.
3. **Krawędź etykietowana jest relacją pochodzącą z reguł semantycznych.**
   Korzystamy z założenia budowania reguł semantycznych 4.1. Zgodnie z tym z założeniem w regułach dopuszczamy wyłącznie relacje pochodzące ze zbioru $\hat{H}_5$.

□

Powyższy wniosek prowadzi bezpośrednio do dowodu następującej własności algorytmu:

Własność algorytmu 4.3. *Metoda CzyInterpretacjaNiesprzeczna używana przez algorytm ujednoznaczniania zawsze daje odpowiedź poprawną.*

Dowód. Z wniosku 4.2 wiemy, że w sieci ograniczeń występują wyłącznie etykietowania relacjami z podklasy $\hat{H}_5$. Renz i Nebel ([RN1997] oraz [JD1997]) pokazali, że w takim przypadku algorytm PC dla rachunku RCC5 zawsze daje poprawną odpowiedź. □

Złożoność algorytmu

Własność algorytmu 4.4. *Funkcja StworzSiecOgraniczen przedstawiona na listingu 4.3 ma złożoność wielomianową względem liczby jednostek w bazie wiedzy.*

Funkcja *StworzSiecOgraniczen* składa się z dwóch pętli:

1. pętli `repeat`, której zadaniem jest przetworzenie faktów z bazy wiedzy,
2. pętli `for`, której zadaniem jest uruchomienie reguł semantycznych.

Oznaczmy przez n liczbę jednostek w bazie wiedzy oraz przez k liczbę jednostek w zbiorze faktów na wejściu algorytmu. Zakładamy, że liczba faktów na wejściu algorytmu jest mała (tzn. $k \ll n$).

Pierwsza z wymienionych pętli zakończy swoje działanie, kiedy w jej pojedynczym przebiegu nie zostaną dodane żadne jednostki. W najgorszym przypadku w jednym przebiegu dodawana jest jedna nowa jednostka. Możemy więc asymptotycznie ograniczyć z góry liczbę przebiegów pętli `repeat` przez $O(n)$.

Pętla `repeat` składa się z dwóch wywołań pętli `for`:

- liczbę wywołań pierwszej pętli `for` ograniczymy z góry przez maksymalną liczbę faktów przestrzennych przechowywanych w bazie wiedzy; zakładając że nie przechowujemy faktów redundantnych (tzn. takich, których typ relacji, podmiot oraz dopełnienie są takie same), to możemy ograniczyć liczbę faktów przez liczbę krawędzi w grafie złożonym ze wszystkich jednostek; stąd liczbę wywołań pętli można ograniczyć z góry przez $O(n^2)$,
- liczbę wywołań drugiej pętli `for` ograniczymy z góry przez liczbę jednostek w bazie wiedzy (w jednym przebiegu do sieci dodano wszystkie jednostki z bazy wiedzy).

Sumarycznie pętla `repeat` wykonuje się więc $O(n^3)$ razy.

Pozostaje analiza pętli odpowiedzialnej za reguły semantyczne. Liczbę wykonań tej pętli można ograniczyć z góry przez liczbę par jednostek w bazie wiedzy (ponieważ założyliśmy, że liczba faktów na wejściu jest mała). Oznacza to, że wywołań pętli jest $O(n^2)$.

Wewnątrz pętli uruchamiane są reguły semantyczne. Liczba reguł jest skończona oraz stała. Ponadto reguły dotyczą prostych operacji w postaci sprawdzenia zgodności typów jednostek, bądź porównań wartości jednostek. Oznacza to, że uruchomienie reguł semantycznych na danej parze jednostek wiąże się z wykonaniem liczby operacji ograniczonej przez stałą.

Podsumowując, całość pracy, funkcji `StworzSiecOgraniczen` jest zdominowana przez pierwszą pętlę `repeat`. Liczbę jej wywołań można oszacować z góry przez $O(n^3)$. Stąd wielomianowa złożoność funkcji.

Własność algorytmu 4.5. *Algorytm ujednoznaczniania przedstawiony na listingu 4.1 ma złożoność wielomianową ze względu na liczbę jednostek w bazie wiedzy oraz wykładniczą ze względu na liczbę jednostek w zbiorze faktów wejściowych Z.*

Złożoność algorytmu ujednoznaczniania wynika z:

1. Złożoności funkcji *StworzSiecOgraniczen* (patrz własność 4.4),
2. Konieczności sprawdzenia wszystkich kombinacji interpretacji jednostek w bazie wiedzy:

Oznaczmy przez n liczbę jednostek w bazie wiedzy oraz przez k liczbę jednostek w zbiorze faktów na wejściu algorytmu.

Sprawdzenie wszystkich kombinacji interpretacji może się wiązać z wykładniczym czasem wykonania (względem liczby jednostek w przetwarzanym zbiorze faktów Z). W pesymistycznym przypadku każda z jednostek ze zbioru faktów wymaga ujednoznacznienia. Liczbę jednostek, w stosunku do których sprawdzamy możliwość niejednoznaczności możemy oszacować przez liczbę jednostek w bazie wiedzy n. Oznacza to, że zbiór interpretacji składa się z maksymalnie $O((n+1)^k)$ interpretacji, które w pesymistycznym przypadku (interpretacja przyporządkowująca wszystkie jednostki jako nowe, oznaczone przez NULL, jest jedyną poprawną) są w całości przetworzone przez główną pętlę algorytmu. Stąd jej sumaryczny czas działania jest wykładniczy względem liczby jednostek w zbiorze faktów na wejściu algorytmu.

Powyższa analiza wskazuje, że istotnym problemem jest konieczność sprawdzenia kombinacji wszystkich interpretacji. Niektóre źródła wiedzy dostarczają zbiorów faktów, składających się ze znacznej liczby jednostek (np. rzeki przepływające przez kilkanaście państw). Dodanie takich zbiorów faktów do bazy wiedzy trwa istotnie dłużej.

Wyniki eksperymentów pomiaru czasu działania algorytmu ujednoznaczniania znajdują się w podrozdziale 4.4.2. Potwierdzają one zasygnalizowany wyżej problem związany z wykładniczą złożonością algorytmu ujednoznaczniania.

4.3. Rozszerzenie algorytmu o rachunek RCC8

4.3.1. Motywacja

Część źródeł przetwarzanych przez system HipiSwot, oprócz informacji jakościowej o wzajemnym ułożeniu obiektów geograficznych, dostarcza także informacje ilościowe, takie jak: zaludnienie regionu, wysokość, długość (np. długość rzeki), współrzędne geograficzne. Natomiast informacje jakościowe dostarczane przez źródła są czasem bardzo ogólne (np. położenie miasta z dokładnością do kraju, zamiast jednostki administracyjnej niższego rzędu). Aby zwiększyć jakość wiedzy zbieranej przez system wprowadzono możliwość ujednoznacznienia w oparciu o informacje ilościowe dostarczane przez źródła.

Pierwotna (naiwna) wersja ujednoznaczniania w oparciu o informacje ilościowe polegała na porównaniu dwóch wartości związanych tą samą relacją. Jeśli wartości były równe, to uznawano, że dotyczą tej samej jednostki. Załóżmy na przykład, że dodajemy do bazy wiedzy fakt dotyczący miasta danego kraju, wraz z liczbą ludności tego miasta (informacje takie pochodzą np. z wykazów największych miast świata serwisu MongoBay). Kryterium ujednoznacznienia polega w tym przypadku, na sprawdzeniu, czy w bazie wiedzy znajduje się już informacja na temat liczby ludności danej interpretacji. Jeśli tak, to porównujemy obie wartości. Jeśli wartości są różne, to interpretacja jest odrzucana.

Jednakże takie podejście jest niewystarczające. Problem polega na znacznych różnicach w danych liczbowych podawanych przez różne źródła. Na przykład Wikipedia podaje, że liczba ludności miasta Boston (Stany Zjednoczone) wynosi 625087,[13] natomiast wykaz serwisu MongoBay podaje, że liczba ludności tego miasta wynosi 594034.[14]

Przyczynami różnic są na przykład:

- aktualność danych (np. liczba ludności zmienia się w czasie),
- dokładność danych (np. liczba ludności podana z dokładnością do tysięcy).

W związku z tym należy dopuścić pewien margines błędu między wartościami liczbowymi przechowywanymi w bazie wiedzy. W niniejszym podrozdziale pokazuję, że ujednoznacznianie bazujące na porównywaniu wartości liczbowych z pewnym marginesem błędu można zrealizować za pomocą wnioskowania jakościowego w ramach wcześniej przedstawionego algorytmu ujednoznaczniania.

[13] http://en.wikipedia.org/wiki/Boston
[14] http://world.mongabay.com/polish/population/pl.html

Zaletą zaproponowanego rozwiązania jest prostota. Algorytm ujednoznaczniania zaprezentowany w podrozdziale 4.2 nie musi być w istotny sposób zmieniany w celu obsługi relacji między wartościami liczbowymi.

4.3.2. Modelowanie relacji ilościowych w rachunku RCC8

Wartości liczbowe zbierane przez system HipiSwot reprezentujemy przez liczby rzeczywiste podawane z pewną dokładnością ϵ. Dokładność jest stałą, będącą dodatnią liczbą rzeczywistą, ustalaną dla wszystkich wartości danej sieci ograniczeń.

Zdefiniujemy trzy relacje zachodzące między wartościami liczbowymi: *equal* (równość), *aprox* (około) oraz *notequal* (różność).

Definicja 4.5. Relacje *equal*, *aprox* oraz *notequal* dla dwóch liczb $x \in \mathbb{R}, y \in \mathbb{R}$ definiujemy w następujący sposób:

- $equal(x, y) \Leftrightarrow |x - y| = 0$,
- $aprox(x, y) \Leftrightarrow |x - y| > 0 \wedge |x - y| < \epsilon$,
- $notequal(x, y) \Leftrightarrow |x - y| \geqslant \epsilon$.

Spostrzeżenie 4.6. *Zbiór relacji* $Q3 = \{equal, aprox, notequal\}$ *zdefiniowanych na zbiorze liczb rzeczywistych dla dodatniej stałej* $\epsilon \in \mathbb{R}$ *posiada własność JEPD.*

Powyższa własność wynika bezpośrednio z definicji relacji ze zbioru $Q3$. Moduł dwóch liczb rzeczywistych jest zawsze liczbą nieujemną. Ponieważ stała ϵ jest dodatnia, to po prawej stronie równoważności definiujących relacje mamy podział całego zbioru liczb nieujemnych na trzy rozłączne przedziały, co bezpośrednio implikuje własność JEPD.

Relacje ze zbioru $Q3$ modelujemy za pomocą rachunku RCC8. W tym celu zdefiniujemy relację połączenia $C(x, y)$ dla dwóch liczb rzeczywistych. Relację $C(x, y)$ zdefiniujemy w następujący sposób:

Definicja 4.6.

$$\forall_{x,y \in \mathbb{R}}[C(x, y) \Leftrightarrow |x - y| < \epsilon]$$

Lemat 4.7. *Relacja* $C(x, y)$ *spełnia warunki modelu RCC, to znaczy jest zwrotna i symetryczna.*

Lemat wynika bezpośrednio z definicji relacji $C(x, y)$ oraz własności modułu.

Pokażemy teraz, że zachodzi równoważność miedzy relacjami ze zbioru $Q3$ oraz zbiorem relacji $\{EQ, DC, ONE_{EC}\}$, gdzie:[15]

$$ONE_{EC} := \{EC, PO, TPP, NTPP, TPPI, NTPPI\}$$

[15] Nazwa relacji wskazuje na powiązanie z relacją *ONE* rachunku RCC3. Zachodzi: $ONE_{EC} = ONE \cup EC$.

Dla uproszczenia zbiór ten oznaczymy przez $RCC8_{Q3}$.

Udowodnienie takiej równoważności pozwoli nam na modelowanie relacji ze zbioru $Q3$ za pomocą podzbioru relacji rachunku RCC8.

Zaczniemy od spostrzeżenia, że:

Spostrzeżenie 4.8. *Zbiór relacji* $RCC8_{Q3} = \{EQ, DC, ONE_{EC}\}$ *posiada własność JEPD.*

Powyższa własność wynika bezpośrednio z własności JEPD relacji rachunku RCC8, oraz definicji zbioru $RCC8_{Q3}$.

Twierdzenie 4.9. *W modelu RCC8 skonstruowanym z wykorzystaniem relacji* $C(x, y)$ *zdefiniowanej w 4.6 zachodzi:*

$$\forall_{x,y\in\mathbb{R}}[DC(x,y) \Leftrightarrow notequal(x,y)]$$

$$\forall_{x,y\in\mathbb{R}}[EQ(x,y) \Leftrightarrow equal(x,y)]$$

$$\forall_{x,y\in\mathbb{R}}[ONE_{EC}(x,y) \Leftrightarrow aprox(x,y)]$$

Dowód. Udowodnimy prawdziwość dwóch pierwszych formuł. Następnie wykorzystamy własność JEPD obu zbiorów relacji, aby pokazać prawdziwość ostatniej formuły.

1. $\forall_{x,y\in\mathbb{R}}[DC(x,y) \Leftrightarrow notequal(x,y)]$

   Z definicji relacji DC oraz relacji *notequal*:

   $$\forall_{x,y\in\mathbb{R}}[\neg C(x,y) \Leftrightarrow |x-y| \geqslant \epsilon]$$

   $$\forall_{x,y\in\mathbb{R}}[\neg(|x-y| < \epsilon) \Leftrightarrow |x-y| \geqslant \epsilon]$$

   Prowadzi to do tautologii:

   $$\forall_{x,y\in\mathbb{R}}[|x-y| \geqslant \epsilon \Leftrightarrow |x-y| \geqslant \epsilon]$$

2. $\forall_{x,y\in\mathbb{R}}[EQ(x,y) \Leftrightarrow equal(x,y)]$

   Z definicji relacji EQ oraz relacji *equal*:

   $$\forall_{x,y\in\mathbb{R}}[P(x,y) \wedge P(y,x) \Leftrightarrow |x-y| = 0]$$

   Z definicji relacji P:

   $$\forall_{x,y\in\mathbb{R}}\{\forall_{z\in\mathbb{R}}[C(z,x) \Rightarrow C(z,y)] \wedge \forall_{z\in\mathbb{R}}[C(z,y) \Rightarrow C(z,x)] \Leftrightarrow |x-y| = 0\}$$

$$\forall_{x,y\in\mathbb{R}}\{\forall_{z\in\mathbb{R}}[(C(z,x)\Rightarrow C(z,y))\wedge(C(z,y)\Rightarrow C(z,x))]\Leftrightarrow|x-y|=0\}$$

$$\begin{aligned}\forall_{x,y\in\mathbb{R}}\{\forall_{z\in\mathbb{R}}[(|z-x|<\epsilon\Rightarrow|z-y|<\epsilon)\\ \wedge(|z-y|<\epsilon\Rightarrow|z-x|<\epsilon)]\Leftrightarrow|x-y|=0\}\end{aligned}\tag{4.1}$$

Mamy dwa przypadki:

a) $|x-y|=0$

To oznacza, że: $x=y$, więc lewa strona równoważności redukuje się do:

$$\forall_{z\in\mathbb{R}}[(|z-x|<\epsilon\Rightarrow|z-x|<\epsilon)\wedge(|z-x|<\epsilon\Rightarrow|z-x|<\epsilon)]$$

Jest to tautologią.

b) $\neg(|x-y|=0)$

To oznacza, że: $x\neq y$. Pokażemy, że w takim przypadku istnieje $z\in\mathbb{R}$ które nie spełnia warunku:

$$(|z-x|<\epsilon\Rightarrow|z-y|<\epsilon)\wedge(|z-y|<\epsilon\Rightarrow|z-x|<\epsilon)\tag{4.2}$$

Rozpatrzmy dwa przypadki:

i. $|x-y|\geqslant\epsilon$

Jedna z liczb x i y jest większa (z założenia $\neg(|x-y|=0)$).

Wybierzmy większą z nich. Bez straty ogólności przyjmijmy, że jest to x.

Wybierzmy stałą $\alpha\in\mathbb{R}$ taką, że $0<\alpha<\epsilon$.

Przyjmiemy $z:=x+\alpha$.

Oczywiście zachodzi: $|z-x|<\epsilon$

Jednakże:

$$|z-y|<\epsilon\Leftrightarrow|x+\alpha-y|<\epsilon\Leftrightarrow|(x-y)+\alpha|<\epsilon$$

Skoro $x>y$ oraz $\alpha>0$, to wyrażenie to jest równoważne następującemu:

$$|x-y|+\alpha<\epsilon$$

Tymczasem założyliśmy $|x-y|\geqslant\epsilon$. Otrzymaliśmy sprzeczność.

ii. $|x-y|<\epsilon$

Ponownie wybierzmy większą z liczb x i y. Bez straty ogólności przyjmijmy, że jest to x.

Przyjmiemy $z:=x+\epsilon-|x-y|$.

Oczywiście zachodzi: $|z-x|<\epsilon$, jednakże:

$$|z-y| < \epsilon \Leftrightarrow |x+\epsilon-|x-y|-y| < \epsilon \Leftrightarrow |(x-y)-|x-y|+\epsilon| < \epsilon$$

Skoro $x > y$, dochodzimy do sprzeczności:

$$|\epsilon| < \epsilon$$

Pokazaliśmy, że w każdym przypadku istnieje $z \in \mathbb{R}$, takie że warunek 4.2 nie jest spełniony.

Pokazaliśmy, że w każdym przypadku równoważność z formuły 4.1 jest prawdziwa. Oznacza to, że formuła 4.1 jest tautologią, czyli istotnie zachodzi:

$$\forall_{x,y\in\mathbb{R}}[EQ(x,y) \Leftrightarrow equal(x,y)]$$

3. $\forall_{x,y\in\mathbb{R}}[ONE_{EC}(x,y) \Leftrightarrow aprox(x,y)]$
   Załóżmy, że formuła nie jest spełniona. Oznacza to, że:

$$\exists_{x,y\in\mathbb{R}}\{\neg[ONE_{EC}(x,y) \Leftrightarrow aprox(x,y)]\}$$

Weźmy takie dwie liczby $x, y \in \mathbb{R}$, które spełniają powyższą negację:

$$\neg[ONE_{EC}(x,y) \Leftrightarrow aprox(x,y)]$$

Mamy dwa przypadki:

a) zachodzi $ONE_{EC}(x,y)$
   To oznacza, że nie zachodzi: $aprox(x,y)$.
   Z własności JEPD zbioru relacji $Q3 = \{equal, aprox, notequal\}$ wiemy, że musi zachodzić albo $equal(x,y)$, albo $notequal(x,y)$. Stąd, korzystając z wyżej udowodnionych równoważności, zachodzi albo $EQ(x,y)$, albo $DC(x,y)$.
   Z własności JEPD zbioru relacji $\{EQ, DC, ONE_{EC}\}$ otrzymujemy więc, że nie może zachodzić $ONE_{EC}(x,y)$. Otrzymaliśmy sprzeczność.

b) nie zachodzi $ONE_{EC}(x,y)$
   Dowód metodą nie wprost prowadzimy analogicznie jak w przypadku *zachodzi* $ONE_{EC}(x,y)$.

□

Powyższe twierdzenie pokazuje, że relacje ze zbioru $Q3$ możemy modelować za pomocą podzbioru relacji rachunku RCC8. W dalszej części rozdziału pokażę, w jaki sposób zastosować relacje $Q3$ w procesie zbierania bazy wiedzy i algorytmie ujednoznaczniania.

4.3.3. Modyfikacje algorytmu ujednoznaczniania

Modyfikacje algorytmu ujednoznaczniania polegają na:
1. Dodaniu reguł transformacji faktów ilościowych do relacji $Q3$,
2. Dodaniu reguł semantycznych dla jednostek z faktów ilościowych,
3. Tworzeniu więcej niż jednej sieci ograniczeń i niezależnym sprawdzaniu niesprzeczności każdej z sieci.

Reguły modelowania faktów ilościowych

Zdefiniujemy dwa dodatkowe typy relacji używane w faktach:

- *równy* — jednostka A ma wartość dokładnie równą B,
- *przybliżony* — jednostka A ma wartość około równą B (być może równą).

Zakładamy, że w bazie wiedzy wszystkie zapisane fakty są dokładne. Stąd fakty ilościowe pochodzące z bazy wiedzy transformowane są do typu relacji *równy* (odpowiadającej relacji *equal* ze zbioru $Q3$).

Fakt ilościowy pochodzący z nowego źródła uznawany jest za niedokładny. Taki fakt transformowany jest do relacji typu *przybliżony* (odpowiadającej sumie relacji *equal* oraz *aprox*).

Relacje *równy* i *przybliżony* modelowane są w rachunku RCC8 zgodnie z twierdzeniem 4.9 tzn.:

- *równy* $\rightarrow EQ$,
- *przybliżony* $\rightarrow \{EQ, ONE_{EC}\}$.

W ten sposób przetwarzamy następujące typy relacji:

- liczba ludności,
- długość geograficzna,
- szerokość geograficzna,
- wysokość nad poziomem morza,
- powierzchnia,
- długość.

Zauważmy, że dla wymienionych wyżej typów relacji, dopełnienie faktu jest pewną wartością liczbową (np.: liczbą lub wartością metryczną).

Uwaga 4.2. Zakładamy, że dany fakt ilościowy jest dla danej jednostki unikalny. Oznacza to, że nie zapisujemy do bazy wiedzy faktu ilościowego, który już znajduje się w bazie wiedzy (taki fakt wykorzystywany jest tylko do ujednoznacznienia).

Reguły semantyczne

Reguły semantyczne dla jednostek z faktów ilościowych działają na wartościach tych jednostek (a nie typach jednostek). Obsługujemy wyłącznie jednostki mające wartość liczbową (wartości metryczne i liczby).

Reguły semantyczne polegają na porównaniu wartości związanych z jednostkami oraz przypisaniu im jednej z relacji *equal*, *aprox* oraz *notequal* zgodnie z definicją 4.5. Następnie relacje te zostają modelowane w rachunku RCC8 zgodnie z twierdzeniem 4.9. Wartość ϵ ustalana jest dla całej sieci ograniczeń (przed uruchomieniem reguł semantycznych).[16]

Tworzenie dwóch sieci ograniczeń

Aby umożliwić wykorzystanie innego rachunku wymagana jest jedynie zmiana funkcji *CzyInterpretacjaNiesprzeczna*, tak aby tworzyć dwie sieci ograniczeń. Zmodyfikowany[17] algorytm tworzenia tej funkcji znajduje się na listingu 4.4.

Algorytm 4.4: CzyInterpretacjaNiesprzeczna

```
   Data: Interpretacja i, zbiór faktów Z
   Result: Wartość logiczna
 1 begin
       /* Relacje przestrzenne modelowane za pomocą RCC5          */
 2     F ← zbiór faktów Z zinterpretowany przez i ;
 3     N1 ← StworzSiecOgraniczen(F, RCC5) ;
 4     if not PathConsistent(N1) then
 5         return FALSE
       /* Relacje ilościowe modelowane za pomocą RCC8             */
 6     N2 ← StworzSiecOgraniczen(F, RCC8) ;
 7     if not PathConsistent(N2) then
 8         return FALSE
       /* Wszystkie sieci są niesprzeczne                         */
 9     return TRUE
10 end
```

4.3.4. Przykład działania algorytmu

Przykład 4.3: Dodanie nowej jednostki

Rozpatrzmy przykład dodawania do bazy wiedzy następującego zbioru faktów:[18]

[16] W implementacji algorytmu ujednoznacznienia w systemie HipiSwot wartość ϵ jest równa: 10^{d-1}, gdzie d to liczba cyfr największej wartości liczbowej w sieci ograniczeń (licząc tylko cyfry części całkowitej).

[17] W implementacji algorytmu zastosowano mechanizmy polimorfizmu, dzięki czemu przy dodawaniu nowego rachunku nie są wymagane żadne modyfikacje.

[18] http://en.wikipedia.org/wiki/Boston,_New_York

```
Boston [miasto]
- jest położony w, Stany Zjednoczone [państwo]
- liczba ludności, 8023 [liczba]
```

W bazie wiedzy znajdują się następujące fakty:[19]

```
Boston [miasto] id:1, jest położony w, Stany Zjednoczone [państwo] id:2
Boston [miasto] id:1, liczba ludności, 625087 [liczba]
```

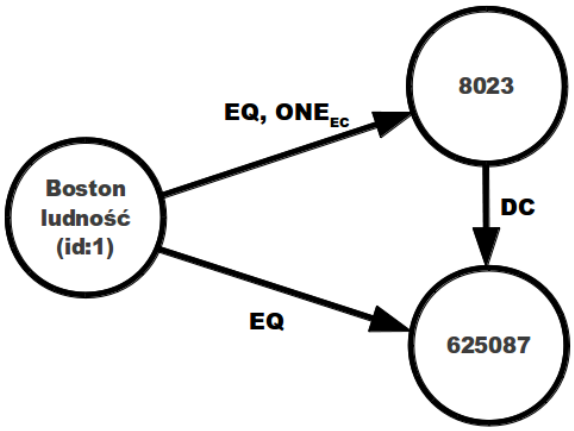
Kolekcja interpretacji dla przetwarzanego zbioru faktów przedstawiona została poniżej:[20]

- (Boston [miasto] → 1, Stany Zjednoczone [państwo] → 2),
- (Boston [miasto] → 1, Stany Zjednoczone [państwo] → NULL),
- (Boston [miasto] → NULL, Stany Zjednoczone [państwo] → 2),
- (Boston [miasto] → NULL, Stany Zjednoczone [państwo] → NULL).

Algorytm ujednoznaczniania wykorzystujący wyłącznie relacje przestrzenne zaakceptuje pierwszą (**błędną**) interpretację.

W drugim kroku rozpocznie się ujednoznacznianie z wykorzystaniem faktów ilościowych. Fakt pochodzący z bazy wiedzy uznany jest za dokładny i przetransformowany na typ relacji *równy* (modelowany w RCC8 przez relację EQ). Fakt dodawany przez źródło uznawany jest za niedokładny i przetransformowany na typ relacji *przybliżony* (co w RCC8 modelujemy jako $\{EQ, ONE_{EC}\}$).

Wartość ϵ zostaje ustalona na $10^5 = 100000$. Reguły semantyczne między wartościami liczbowymi 8023 oraz 625087 dodają relację DC między jednostkami (bowiem $|625087 - 8023| \geqslant \epsilon$). Wynikową sieć ograniczeń przedstawia rysunek 4.6.



Rysunek 4.6. Wynikowa sieć ograniczeń dla miasta Boston z przykładu 4.3

[19] http://en.wikipedia.org/wiki/Boston

[20] Pojęcia zawsze dodawane są jako nowe jednostki, dlatego w prezentowanym przykładzie pojęcia zostały pominięte w kolekcji interpretacji.

Sieć ograniczeń jest sprzeczna. Złożenie relacji między wierzchołkami: *Boston*, *8023* oraz *625087* jest równe:

$$\{EQ, ONE_{EC}\} \circ DC = (EQ \circ DC) \cup (ONE_{EC} \circ DC)$$

$$= (EQ \circ DC) \cup (\{EC, PO, TPP, NTPP, TPPI, NTPPI\} \circ \{DC\})$$

Z tablicy złożeń relacji dla rachunku RCC8 mamy:

$$EQ \circ DC = \{DC\}$$

$$EC \circ DC = \{DC, EC, PO, TPPI, NTPPI\}$$

$$PO \circ DC = \{DC, EC, PO, TPPI, NTPPI\}$$

$$TPP \circ DC = \{DC\}$$

$$NTPP \circ DC = \{DC\}$$

$$TPPI \circ DC = \{DC, EC, PO, TPPI, NTPPI\}$$

$$TPPI \circ DC = \{DC, EC, PO, TPPI, NTPPI\}$$

Stąd:

$$\{EQ, ONE_{EC}\} \circ DC = \{DC, EC, PO, TPPI, NTPPI\}$$

Tymczasem krawędź łącząca bezpośrednio wierzchołki *Boston* oraz *625087* etykietowana jest relacją EQ. Część wspólna obu zbiorów relacji jest pusta. Algorytm PC zgłasza sprzeczność. Ze względu na wykrycie sprzeczności algorytm odrzuca interpretację jako błędną.

Podobny proces jest przeprowadzony dla kolejnej interpretacji, jednakże również w tym przypadku ujednoznacznianie za pomocą wartości liczbowych odrzuca interpretację. Dopiero trzecia interpretacja wprowadzająca nową jednostkę: *Boston [miasto]* zostaje zaakceptowana.

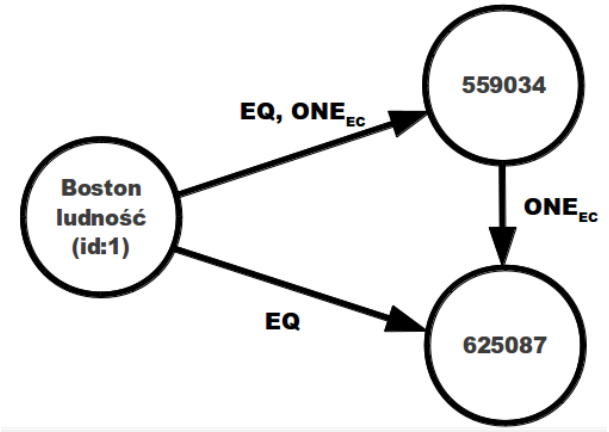
Przykład 4.4: Dodanie niedokładnych danych liczbowych

Rozpatrzmy przykład dodawania do bazy wiedzy następującego zbioru faktów:[21]

```
Boston [miasto]
- jest położony w, Stany Zjednoczone [państwo]
- liczba ludności, 559034 [liczba]
```

Podobnie jak w poprzednim przykładzie w bazie wiedzy znajdują się następujące fakty:[22]

[21] http://world.mongabay.com/polish/population/pl.html
[22] http://en.wikipedia.org/wiki/Boston



Rysunek 4.7. Wynikowa sieć ograniczeń dla miasta Boston z przykładu 4.4

```
Boston [miasto] id:1, jest położony w, Stany Zjednoczone [państwo] id:2
Boston [miasto] id:1, liczba ludności, 625087 [liczba]
```

Kolekcja interpretacji dla przetwarzanego zbioru faktów jest identyczna jak w poprzednim przykładzie (z dokładnością do wartości liczbowych).

Algorytm ujednoznaczniania rozpoczyna działanie od sprawdzenia interpretacji, w której obie jednostki przestrzenne zostają połączone z jednostkami w bazie wiedzy. Ujednoznacznianie za pomocą relacji przestrzennych akceptuje taką interpretację. Rozpoczyna się ujednoznacznianie z wykorzystaniem faktów ilościowych.

Tworzona sieć ograniczeń jest niemalże identyczna z siecią ograniczeń stworzoną w poprzednim przykładzie. Jedyną różnicą jest krawędź dodana przez reguły semantyczne. Wartość ϵ zostaje ustalona na $10^5 = 100000$. Reguły semantyczne między wartościami liczbowymi 594034 oraz 625087 dodają relację ONE_{EC} między jednostkami (bowiem $0 < |625087 - 594034| < \epsilon$). Wynikową sieć ograniczeń przedstawia rysunek 4.7.

Sieć ograniczeń nie zawiera relacji pustej, dlatego ostatecznie interpretacja zostaje wybrana jako poprawna.

Do bazy wiedzy nie zostaną jednak dodane nowe fakty. Fakt o położeniu miasta Boston w Stanach Zjednoczonych znajdował się już w bazie wiedzy, podobnie jak fakt o liczbie ludności miasta Boston. Gdyby jednak zastosowano naiwne ujednoznacznienie (oparte o dokładne porównanie liczby ludności) do bazy wiedzy zostałaby wprowadzona nowa jednostka dotycząca miasta Boston (co zakładamy, że byłoby błędem).

4.3.5. Analiza zmodyfikowanego algorytmu

Własność algorytmu 4.10. *W algorytmie ujednoznaczniania za pomocą wartości liczbowych, sieć ograniczeń etykietowana jest relacjami ze zbioru:*

$$RCC8_{Q3} = \{EQ, DC, ONE_{EC}, \{EQ, ONE_{EC}\}, \top\}$$

Dowód. Podobnie jak w dowodzie wniosku 4.2 wykorzystamy fakt, że etykietowania krawędzi w sieci ograniczeń mogą pochodzić z trzech źródeł:

1. **Krawędź etykietowana jest relacją uniwersalną.**
   Relacja uniwersalna należy do zbioru $RCC8_{Q3}$.
2. **Krawędź etykietowana jest typem relacji pochodzącym z modelowania w rachunku RCC8 pewnego faktu między obiektami v i w, bądź relacją odwrotną do modelowanej.**
   W przypadku faktów ilościowych dopuszczamy tylko dwa typy relacji: *równy* oraz *przybliżony*. Wymienione typy relacji modelowane są odpowiednio relacjami: EQ oraz $\{EQ, ONE_{EC}\}$. Relacje te należą do zbioru $RCC8_{Q3}$.
   Co więcej relacje odwrotne mają postać:

$$EQ' = EQ$$

$$\{EQ, ONE_{EC}\}' = \{EQ, ONE_{EC}\}$$

3. **Krawędź etykietowana jest relacją pochodzącą z reguł semantycznych.**
   Założyliśmy, że reguły semantyczne budujemy używając wyłącznie relacji *equal*, *notequal*, *aprox*, które są modelowane przez odpowiednio: EQ, DC oraz ONE_{EC}.

□

Powyższa własność prowadzi bezpośrednio do następujących wniosków:

Wniosek 4.11. *Zbiór Q3 relacji rachunku RCC8 używanych w procesie ujednoznacznia za pomocą wartości liczbowych jest podzbiorem klasy $\hat{H}_8$.*

Wniosek 4.12. *Algorytm PC uruchomiony na sieci ograniczeń stworzonych przez modelowanie relacji ze zbioru Q3 zawsze daje odpowiedź poprawną.*

Dowód. Renz i Nebel [RN1997] pokazali, że w podklasie $\hat{H}_8$ dla rachunku RCC8 algorytm PC daje zawsze poprawną odpowiedź. Jest to podklasa podatna rachunku RCC8. □

Ostatnia obserwacja jest związana z zapisywaniem nowych faktów do bazy wiedzy. Założyliśmy, że w przypadku gdy dany fakt ilościowy dla danej jednostki został już zapisany w bazie wiedzy, to ten sam fakt pochodzący z innego źródła nie zostaje

zapisany do bazy wiedzy (wykorzystywany jest tylko do ujednoznacznienia) patrz uwaga 4.2.

Spostrzeżenie 4.13. *Ujednoznacznianie za pomocą relacji ilościowych operuje zawsze na sieci ograniczeń o liczbie wierzchołków* $\leqslant 3$.

Powyższa obserwacja wynika z faktu, że w bazie wiedzy może znajdować się tylko jeden fakt ilościowy danego typu dla danej jednostki. Dzięki tej własności algorytmu nie doprowadzimy do sprzeczności w bazie.

4.4. Analiza zebranych danych

4.4.1. Wielkość zebranej bazy wiedzy

W systemie HipiSwot przetworzono łącznie 9 źródeł wymienionych w podrozdziale 4.1.2. Zebrano łącznie 298002 faktów (w tym prawie 190 tys. faktów przestrzennych) oraz 261063 jednostek (w tym prawie 150 tys. jednostek przestrzennych).

Aby zbadać wpływ zastosowania algorytmów ujednoznaczniania porównano dwie wersje systemu HipiSwot:

- **wersję bazową** — w tej wersji systemu algorytmy ujednoznaczniania były wyłączone,
- **wersję ostateczną** — w tej wersji systemu zastosowano algorytmy ujednoznaczniania.

Obie wersje systemu HipiSwot zostały uruchomione na tym samym zbiorze źródeł wiedzy. W tabeli 4.1 przedstawiono liczby zebranych jednostek i faktów w obu wersjach systemu.

Tabela 4.1. Porównanie wielkości bazy wiedzy stworzonej w bazowej wersji systemu HipiSwot oraz w wersji rozbudowanej o algorytmy ujednoznaczniania

	Wersja bazowa	Wersja ostateczna
Całkowita liczba jednostek	232 710	261 063
Liczba jednostek przestrzennych	111 891	147 085
Całkowita liczba faktów	307 668	298 002
Liczba jakościowych faktów przestrzennych	191 715	186 829

W ostatecznej wersji systemu zebrano o prawie 30 tys. więcej jednostek. Jednostki te wprowadzone zostały w wyniku ujednoznacznienia. Algorytm ujednoznacznienia wykrył, że dołączenie pewnych faktów do jednostek znajdujących się w bazie spowoduje powstanie sprzeczności, dlatego zamiast tego wybrano interpretację wprowadzającą nową jednostkę do bazy.

Jakość zebranego materiału zmierzono na korpusie pytań testowych poprzez ewaluację algorytmów odpowiadania na pytania (patrz rozdział 6).

4.4.2. Eksperymentalne badanie czasu działania

Aby zbadać czas działania implementacji algorytmu ujednoznaczniania opracowano testową bazę wiedzy obejmującą informację o dziesięciu państwach. Każde państwo zostało podzielone na 10 jednostek administracyjnych pierwszego rzędu, z których każda została podzielona na 10 jednostek administracyjnych drugiego rzędu. Przez każdą jednostkę administracyjną drugiego rzędu przepływało 10 rzek. Wszystkie jednostki miały unikalne nazwy.

Eksperyment polegał na dodaniu do bazy wiedzy zbioru faktów na temat rzek o nazwach pokrywających się z nazwami rzek z testowej bazy wiedzy (jednakże leżących w innym regionie, więc wymagających ujednoznacznienia). Zbadano trzy grupy faktów:

1. `PP1` — rzeki położone w państwie,
2. `PP2` — rzeki położone w jednostce administracyjnej pierwszego rzędu,
3. `PP3` — rzeki położone w jednostce administracyjnej drugiego rzędu.

Dla każdej z wymienionych grup zmierzono czas znalezienia interpretacji dla próbki testowej składającej się z 50, 100, 150, 200, 250 zbiorów faktów. Każdy zbiór faktów zawierał pojedynczy fakt z relacją *jest położony w*. Eksperyment został uruchomiony na komputerze z procesorem Intel Core i3–2370M CPU 2.40GHz oraz 8 GB pamięci operacyjnej RAM. Wyniki pomiarów czasu przedstawione zostały w tabeli 4.2. W tabeli czas został podany w sekundach. Parametr n w nagłówku tabeli oznacza liczbę faktów w próbce.

Tabela 4.2. Wyniki eksperymentu sprawdzenia czasu działania algorytmów ujednoznacznienia dla relacji PP

	$n = 50$	$n = 100$	$n = 150$	$n = 200$	$n = 250$
Zbiór PP1	23,56	47,48	70,97	94,89	118,44
Zbiór PP2	24,65	49,21	74,17	98,83	123,46
Zbiór PP3	26,06	52,00	78,32	103,86	129,67

Wyniki przedstawione w tabeli 4.2 pokazują nieznaczne różnice w czasie działania algorytmu dla każdej z próbek. Średnio jeden fakt ujednoznaczniany był w około pół sekundy. Najdłużej trwało przetwarzanie zbioru z grupy `PP3`. W grupie tej znajdowały się najbardziej szczegółowe fakty wymagające budowania większych sieci ograniczeń, stąd dłuższy sumaryczny czas działania algorytmu.

Drugi eksperyment polegał na sprawdzeniu, jak zmienia się czas działania algorytmu w przypadku wzrostu rozmiaru zbioru faktów na wejściu algorytmu. W tym celu opracowano trzy grupy zbiorów testowych:

1. `PO1` — rzeki przepływające przez z państwa,
2. `PO2` — rzeki przepływające przez jednostki administracyjne pierwszego rzędu,
3. `PO3` — rzeki przepływające przez jednostki administracyjne drugiego rzędu.

Dla każdej z wymienionych grup zmierzono czas znalezienia interpretacji dla próbki testowej składającej się z 50 zbiorów faktów. Wygenerowano próbki, w których każdy zbiór faktów zawierał 1, 2, 3, 4, 5 faktów połączonych relacją *częściowo pokrywa się z*. Eksperyment został uruchomiony na komputerze z procesorem Intel Core i3–2370M CPU 2.40GHz oraz 8 GB pamięci operacyjnej RAM. Wyniki pomiarów czasu przedstawione zostały w tabeli 4.3. W tabeli czas został podany w sekundach. Parametr n w nagłówku tabeli oznacza liczbę faktów w próbce.

Tabela 4.3. Wyniki eksperymentu sprawdzenia czasu działania algorytmów ujednoznacznienia dla relacji PO

	$n=1$	$n=2$	$n=3$	$n=4$	$n=5$
Zbiór PO1	23,63	38,56	69,06	134,13	268,49
Zbiór PO2	24,80	41,99	79,63	157,09	318,79
Zbiór PO3	26,06	48,21	97,07	194,09	398,23

Wyniki pokazują, że zgodnie z analizą przedstawioną w podrozdziale 4.2.3 wraz ze wzrostem rozmiaru zbioru faktów na wejściu algorytmu obserwujemy znaczny wzrost czasu działania (w praktyce wykładniczy). Najdłużej przetwarzany był zbiór zawierający informacje najbardziej szczegółowe (zbiór `PO3`).

Z powyższych eksperymentów wynika, że największym problemem w zaproponowanym algorytmie ujednoznaczniania jest rozmiar zbioru faktów na wejściu algorytmu. W zbiorze przetwarzanych danych zbiory faktów o większej liczbie elementów nie są częste i ograniczają się zwykle do kilku lub kilkunastu elementów. Jednakże w ogólnym zastosowaniu należy mieć na uwadze, że zbiory faktów na wejściu algorytmu nie powinny zawierać zbyt wielu elementów.

4.5. Podsumowanie

W rozdziale omówiłem problem ujednoznaczniania pojęć w procesie pozyskiwania bazy wiedzy przestrzennej. Przedstawiłem autorski algorytm ujednoznaczniania oparty o wnioskowanie przestrzenne. Następnie omówiłem proces rozbudowy algorytmu ujednoznaczniania w oparciu o relacje ilościowe.

Algorytmy ujednoznaczniania zostały zaimplementowane w autorskim systemie zbierania wiedzy przestrzennej HipiSwot. W końcowej części rozdziału omówiłem zebraną za pomocą systemu HipiSwot bazę wiedzy oraz wpływ zastosowania algorytmów ujednoznaczniania. Przedstawiłem również wyniki eksperymentów dotyczących szybkości działania algorytmów ujednoznaczniania.

Rozdział 5

Algorytmy odpowiadania na pytania

W niniejszym rozdziale opisuję autorskie algorytmy odpowiadania na pytania rozstrzygnięcia, w których występuje aspekt czasowy lub przestrzenny. Algorytmy odpowiadania na pytania zostały oparte o wnioskowanie czasowe i przestrzenne z wykorzystaniem rachunku RCC5 oraz algebry Allena.

Rozdział rozpoczynam od charakterystyki problemu odpowiadania na pytania rozstrzygnięcia. Następnie opisuję sposób reprezentacji wiedzy przestrzennej i czasowej za pomocą jednostek i faktów (wprowadzonych w rozdziale 4).

W kolejnym podrozdziale przedstawiam główny moduł wnioskujący, wykorzystywany w algorytmach odpowiadania. Omawiam algorytm wnioskowania, który jest rozwinięciem algorytmów zaprezentowanych w rozdziale 4. W procesie wnioskowania wykorzystywane są: rachunek RCC5 i algebra Allena. Odpowiedzi wyszukiwane są w dwóch bazach wiedzy:

- bazie artykułów prasowych,
- bazie faktów przestrzennych o świecie.[1]

W dalszej części rozdziału opisuję algorytmy odpowiadania na pytania rozstrzygnięcia. Na potrzeby pracy wyróżniam dwie grupy pytań: pytania w postaci kwerendy oraz pytania z warunkami. Opisuję algorytmy odpowiadania na pytania z obydwu wprowadzonych grup pytań. Dla każdego z algorytmów przedstawiam przykład jego działania.

Rozdział kończy analiza działania algorytmu wnioskowania. Ewaluację algorytmów odpowiadania oraz szczegóły implementacyjne opisuję w rozdziale 6.

Niniejszy rozdział jest rozwinięciem artykułu [Wal2012].

5.1. Przedstawienie problemu

Pytania rozstrzygnięcia (czasem nazywane pytaniami polarnymi, ang. *polar questions*) stanowią odrębną klasę pytań. Intencją zadającego pytanie tego typu jest uzyskanie jednej z trzech odpowiedzi:

[1] Baza faktów przestrzennych została zebrana za pomocą metod opisanych w rozdziale 4.

- `TRUE` — potwierdzenie hipotezy zawartej w pytaniu,
- `FALSE` — zaprzeczenie hipotezy zawartej w pytaniu,
- `UNKNOWN` — niemożność odpowiedzenia na pytanie.

W języku polskim pytania tego typu rozpoczynają się najczęściej od słowa *czy*. Przykładowym pytaniem należącym do tej klasy jest pytanie: *Czy Jan Fabre był uczestnikiem Festiwalu Malta?*.

W pracy rozpatrujemy pytania rozstrzygnięcia z aspektem czasowym lub przestrzennym. Na potrzeby pracy przyjmiemy, że pytania tego typu zawierają odnośnik do pewnego miejsca w świecie lub punktu w czasie. W szczególności rozpatrujemy następujące typy odnośników przestrzennych i czasowych:

- odnośnik dotyczący położenia w danym miejscu (np. w mieście, kraju),
- odnośnik dotyczący odbywania się danego wydarzenia w danym czasie, przed danym czasem, lub po danym czasie (np. w danym roku, przed daną datą).

Przykładowymi pytaniami należącymi do tej klasy pytań są:

- Czy Poznań znajduje się w Polsce?
- Czy Uniwersytet Adama Mickiewicza znajduje się w Polsce?
- Czy w Poznaniu kot pogryzł psa?
- Czy w zeszłym roku Jan Fabre był na Festiwalu Malta?
- Czy w zeszłym roku w Azji pies urodził kota?
- Czy w tym roku w Wielkopolsce kot pogryzł psa?

W podrozdziale 3.4.2 przedstawiono powierzchniową metodę odpowiadania na pytania rozstrzygnięcia zaimplementowaną w bazowej wersji systemu Hipisek.pl. Metoda powierzchniowa polega na znalezieniu fragmentu tekstu, który w znacznej części pokrywa się ze zbiorem słów występujących w pytaniu. Na przykład wystąpienie w artykule prasowym frazy: *Poznań to miasto znajdujące się w Polsce* implikuje odpowiedź `TRUE` na pytanie *Czy Poznań znajduje się w Polsce?*

Metoda powierzchniowa jest niewystarczająca w przypadku obsługi większości pytań zawierających aspekt czasowy i przestrzenny. W procesie przetwarzania pytań tego typu często niezbędne są informacje, które nie występują bezpośrednio w tekście źródłowym. Rozpatrzmy na przykład następujący fragment artykułu prasowego opublikowanego 23 lipca 2012 roku:[2]

> Niesamowite! W Korei Południowej suka urodziła kota — twierdzi właściciel psa. — To święta prawda — zaklina się Koreańczyk Jeong Pyong-bong (63 l.), mówiąc o niezwykłych narodzinach w jego gospodarstwie.

[2] Źródło: http://www.fakt.pl/Pies-urodzil-kota-w-Korei-Poludniowej,artykuly,169899,1.html

Znalezienie odpowiedzi na pytanie zadane w roku 2013: *Czy w zeszłym roku w Azji pies urodził kota?* wymaga wykorzystania informacji, że *Korea Południowa znajduje się w Azji* oraz, że *wydarzenie miało miejsce w czasie zeszłego roku.* Informacje te nie są wyrażone bezpośrednio w tekście źródłowym.

Jako rozwiązanie problemu znalezienia odpowiedzi na pytania rozstrzygnięcia zawierające aspekt przestrzenny lub czasowy proponujemy wykorzystanie wnioskowania jakościowego. Główna idea algorytmu polega na pozyskaniu z tekstu źródłowego pewnej relacji przestrzennej (lub odpowiednio czasowej) dotyczącej rozpatrywanego pytania oraz wykorzystaniu rachunków ograniczeń (RCC5 dla wnioskowania przestrzennego oraz algebry Allena dla wnioskowania czasowego) do sprawdzenia niesprzeczności pozyskanej relacji z relacją zawartą w pytaniu.

5.2. Założenia algorytmów odpowiadania

5.2.1. Źródła odpowiedzi

Zakładamy, że w procesie wyszukiwania odpowiedzi na pytanie korzystamy z następujących źródeł:

- **nieustrukturyzowanej bazy wiedzy** — kolekcji dokumentów tekstowych,
- **ustrukturyzowanej bazy wiedzy** — bazy informacji przestrzennych.

Do pozyskiwania informacji z nieustrukturyzowanej bazy wiedzy wykorzystujemy metody powierzchniowe, przy pomocy narzędzi do pozyskiwania relacji przestrzennych i czasowych z fragmentów tekstu. Opis takich narzędzi zaimplementowanych w systemie Hipisek.pl znajduje się w rozdziale 6.1.1.

Ustrukturyzowana baza wiedzy służy do zarządzania tzw. wiedzą podstawową o świecie (ang. *naive world knowledge*). Na potrzeby pracy przyjmiemy, że baza ta przechowuje informacje przestrzenne na temat miejsc na świecie i ich wzajemnym położeniu. Ustrukturyzowana baza wiedzy wykorzystywana w systemie Hipisek.pl powstała z wykorzystaniem metod opisanych w rozdziale 4.

Obydwie wymienione bazy wiedzy zawierają fakty. Jednakże fakty pochodzące z różnych typów baz wiedzy różnią się jakością. W ogólności zakładamy, że fakty pochodzące z ustrukturyzowanej bazy wiedzy charakteryzują się wyższą jakością niż fakty pozyskiwane z nieustrukturyzowanej bazy wiedzy.

W pracy przyjmiemy, że fakty pozyskiwane z fragmentów tekstu pochodzących z bazy nieustrukturyzowanej nazywamy **faktami wydobytymi**. Fakty pochodzące z bazy ustrukturyzowanej nazywamy **faktami pewnymi**.

W stosunku do faktów pewnych i faktów wydobytych przyjmujemy podobnie jak w procesie tworzenia bazy wiedzy przestrzennej założenie o zamkniętości świata

(patrz podrozdział 4.1.1). Według tego założenia zbiór faktów pewnych i wydobytych w pełni opisuje całą wiedzę o świecie. Odpowiedzi negatywne wynikają z wywnioskowania sprzeczności z dostępnych faktów pewnych i wydobytych.

Takie założenie może prowadzić do błędnego działania systemu w przypadku obsługi pytań o wydarzenia cykliczne lub odbywające się wielokrotnie. Na przykład rozpatrzmy pytanie: *Czy igrzyska olimpijskie odbyły się w 2008 roku?* Załóżmy, że system przechowuje informację o igrzyskach olimpijskich mających miejsce w roku 2012 (natomiast nie ma informacji o igrzyskach z roku 2008). W takim przypadku system zwraca błędną odpowiedź negatywną, uzasadniając ją niezgodnością dat. Aby odpowiedź systemu była poprawna, baza wiedzy (w postaci faktów pewnych lub wydobytych) powinna zostać uzupełniona o brakujący fakt dotyczący igrzysk olimpijskich z roku 2008.

W opisywanym rozwiązaniu założenie o zamkniętości świata zostało nieznacznie osłabione. Przyjęto, że jeśli system **nie ma żadnej informacji** na temat danego wydarzenia, to odpowiedzią systemu jest *UNKNOWN* (brak możliwości odpowiedzi na dane pytanie).

5.2.2. Reprezentacja pytania

Ze względu na charakter pytania wyróżnimy na potrzeby pracy dwie grupy pytań rozstrzygnięcia:

- pytania w postaci kwerendy,
- pytania z warunkami.

Pytania w postaci kwerendy są to pytania, które mogą być reprezentowane za pomocą pojedynczego faktu, w którym zarówno podmiot i dopełnienie reprezentują jednostki z używanej bazy wiedzy ustrukturyzowanej. Proces znalezienia odpowiedzi na pytania tego typu redukuje się do potwierdzenia (lub falsyfikacji) danego faktu. Przykładowymi pytaniami tego typu są:

- Czy Korea Południowa jest w Azji?
- Czy Toronto znajduje się w USA?
- Czy Uniwersytet Adama Mickiewicza jest w Polsce?

Pytania z warunkami są to pytania rozstrzygnięcia, w których odnośnik przestrzenny lub czasowy jest traktowany jako modyfikator (dalej nazywany **warunkiem**) bazowej hipotezy zawartej w pytaniu. Pojęcie to najłatwiej wyjaśnić na przykładzie pytania: *Czy w zeszłym roku w Azji pies urodził kota?* Bazowe pytanie (pozbawione odnośników przestrzennych i czasowych) ma postać: *Czy pies urodził kota?* Warunkami w tym przypadku są następujące frazy:

- *w Azji* — odnośnik przestrzenny ograniczający pytanie do terenu kontynentu azjatyckiego,
- *w zeszłym roku* — odnośnik czasowy ograniczający pytanie do wydarzeń odbywających się w zeszłym roku (co uwzględniając kontekst zadania pytania oznacza rok 2012).

Odpowiedź na pytanie z warunkami polega na znalezieniu odpowiedzi na pytanie bazowe (pozbawione warunków), a następnie sprawdzeniu czy znaleziona odpowiedź jest zgodna z wszystkimi warunkami. Innymi słowy w przypadku pytania: *Czy w zeszłym roku w Azji pies urodził kota?* znalezienie odpowiedzi polega na:

- **znalezieniu odpowiedzi bazowej** — wyszukaniu w bazie wiedzy potwierdzenia, czy kiedykolwiek i gdziekolwiek pies urodził kota,
- **sprawdzeniu warunków** — sprawdzeniu, czy wyszukana odpowiedź miała miejsce *w Azji* oraz *w roku 2012*.

5.2.3. Reprezentacja wiedzy w pytaniach

Wiedzę w pytaniu reprezentują jednostki i fakty opisane w rozdziale 4. Reprezentacja wiedzy przestrzennej w procesie odpowiadania na pytania jest tożsama z reprezentacją wiedzy przestrzennej w procesie zbierania bazy wiedzy (patrz podrozdział 4.1.1).

Reprezentacja wiedzy czasowej również została zrealizowana przy pomocy jednostek i faktów. W niniejszym podrozdziale zawarto opis taksonomii typów jednostek czasowych i typów relacji czasowych obsługiwanych w zaprezentowanych w niniejszej pracy algorytmów odpowiadania na pytania.

Zagadnienia implementacyjne związane z pozyskiwaniem relacji przestrzennych i czasowych z tekstowej reprezentacji pytania zostały opisane w rozdziale 6.

Jednostki czasowe

Wyróżniamy dwie podstawowe grupy jednostek czasowych:

- **absolutne wyrażenia czasowe** — wyrażenia czasowe odwołujące się do konkretnych punktów w czasie (np. *czerwiec 2012*),
- **względne wyrażenia czasowe** — wyrażenia czasowe odwołujące się do innych jednostek czasowych bądź kontekstu (np. *zeszły rok*).

W procesie wnioskowania wszystkie jednostki czasowe są traktowane jak przedziały czasowe. Na przykład jednostka *czerwiec 2012* jest traktowana jako przedział od 1 czerwca 2012 do 30 czerwca 2012.

Pełny opis wykorzystywanych typów jednostek czasowych znajduje się w dodatku C.

Typy relacji czasowych występujących w pytaniu

Na potrzeby przetwarzania pytań rozstrzygnięcia z aspektem czasowym używamy następujących typów relacji:[3]

- w trakcie — *is during time*, jednostka A ma/miała miejsce w trakcie jednostki B (przedziały czasowe jednostek mają część wspólną),
- dokładnie w trakcie — *is during strict time*, jednostka A ma/miała miejsce dokładnie w trakcie jednostki B (przedział czasowy jednostki A jest w całości zawarty w przedziale czasowym jednostki B),
- przed — *before time*, jednostka A ma/miała miejsce przed jednostką B,
- po — *after time*, jednostka A ma/miała miejsce po jednostce B,
- zaczyna się — *start time*, jednostka A zaczyna się w czasie jednostki B,
- kończy się — *end time*, jednostka A kończy się w czasie jednostki B,

Na przykład, aby zakodować informację, że w roku 2012 pies urodził kota, wykorzystamy następujący fakt: *(pies urodził kota [wydarzenie], dokładnie w trakcie, 2012 rok [data])*.

5.2.4. Modelowanie wiedzy w algorytmach odpowiadania

Wiedzę przestrzenną modelujemy za pomocą rachunku RCC5 w sposób przedstawiony w rozdziale 4. Wiedzę czasową modelujemy za pomocą algebry Allena.

Modelowanie wiedzy czasowej w algebrze Allena

Wszystkie jednostki czasowe utożsamiamy z przedziałami czasowymi. Typy relacji czasowych modelujemy w algebrze Allena w następujący sposób:

- w trakcie
  - — jeśli dopełnienie jest typu *data* $\rightarrow \{D, EQ\}$,
  - — wpp. $\rightarrow \{D, DI, O, OI, S, SI, F, FI, EQ\}$,
- dokładnie w trakcie $\rightarrow \{D, EQ\}$,
- przed $\rightarrow \{P, M\}$,
- po $\rightarrow \{PI, MI\}$,
- zaczyna się $\rightarrow \{S, SI\}$,
- kończy się $\rightarrow \{F, FI\}$.

Reguły semantyczne opierają się o prostą arytmetykę wartości jednostek czasowych. Na przykład, jeśli porównujemy dwie jednostki typu data o wartościach odpowiednio: *4 lutego 2012* oraz *23 września 2012*, to reguły semantyczne dodają

[3] Opis typów relacji używanych w systemie Hipisek, wraz z przykładami, znajduje się w dodatku D.

między tymi jednostkami relację `P` (pierwsza data poprzedza drugą). Reguły semantyczne wykorzystują wszystkie relacje bazowe algebry Allena.

5.3. Algorytm wnioskowania

Głównym elementem algorytmów odpowiadania na pytania rozstrzygnięcia zaprezentowanych w niniejszej pracy jest **algorytm wnioskowania**. Danymi wejściowymi do algorytmu wnioskowania jest fakt (nazywany dalej **faktem weryfikowanym**), który należy potwierdzić (lub sfalsyfikować) oraz kolekcja dodatkowych faktów (nazywanych dalej **faktami wydobytymi**), które mają być wykorzystane w procesie wnioskowania. Algorytm wnioskowania wykorzystuje wyłącznie ustrukturyzowaną bazę wiedzy, natomiast nie wykorzystuje bezpośrednio nieustrukturyzowanej bazy wiedzy. Zakładamy, że odpowiednie fakty (przydatne w procesie wnioskowania) pochodzące z nieustrukturyzowanej bazy wiedzy zostały pozyskane.[4] Fakty te są danymi wejściowymi do algorytmu wnioskowania (są to fakty wydobyte).

Wnioskowanie składa się z dwóch kroków:

- **krok potwierdzenia** — w tym kroku zadaniem algorytmu jest wykazanie, że fakt weryfikowany jest prawdziwy,
- **krok falsyfikacji** — w tym kroku zadaniem algorytmu jest wykazanie, że fakt weryfikowany jest fałszywy.

Wynikiem działania algorytmu wnioskowania jest jedna z trzech wartości:

- `TRUE` — fakt weryfikowany jest prawdziwy,
- `FALSE` — fakt weryfikowany jest fałszywy,
- `UNKNOWN` — nie można wykazać czy fakt weryfikowany jest prawdziwy czy fałszywy (brak informacji).

W przypadku gdy algorytm wnioskowania zwraca odpowiedź `TRUE` lub `FALSE`, to mówimy że wnioskowanie zakończyło się sukcesem, a fakt weryfikowany nazywamy **faktem zweryfikowanym**. W przeciwnym przypadku wnioskowanie kończy się porażką, a fakt weryfikowany nazywany jest **faktem niezweryfikowanym**.

W procesie wnioskowania wykorzystujemy rachunki ograniczeń do modelowania wiedzy przestrzennej i czasowej. Dla każdego faktu weryfikowanego budowana jest sieć ograniczeń (odpowiednio rachunku RCC5 dla faktu przestrzennego lub algebry Allena dla faktu czasowego) oraz uruchamiany jest na niej algorytm PC. Przyjmujemy następujące założenia:

[4] Opis narzędzi do wyciągania faktów z nieustrukturyzowanej bazy wiedzy zaimplementowanych w systemie Hipisek.pl znajduje się w podrozdziale 6.1.3.

- jeśli w kroku potwierdzenia w sieci ograniczeń (powstałej w wyniku działania algorytmu PC) możliwe jest znalezienie krawędzi między wierzchołkami odpowiadającymi jednostkom z faktu weryfikowanego, która etykietowana jest relacją modelującą typ relacji faktu weryfikowanego (np. relacją `PP` dla typu relacji *jest położony w*) i sieć ograniczeń nie zawiera relacji pustej, to przyjmujemy że fakt weryfikowany jest prawdziwy,
- jeśli w kroku falsyfikacji otrzymana sieć ograniczeń zawiera relację pustą, to przyjmujemy że fakt weryfikowany jest fałszywy,
- jeśli żaden z warunków nie zachodzi przyjmujemy, że nie można wykazać, czy fakt weryfikowany jest prawdziwy czy fałszywy.

Sieć ograniczeń jest budowana w sposób podobny jak w algorytmie 4.3 przedstawionym w podrozdziale 4.2. Zastosowano następujące modyfikacje algorytmu budowy sieci ograniczeń:

- dodano kolekcję faktów dodatkowych (faktów wydobytych) jako opcjonalne źródło faktów dla budowanej sieci ograniczeń,
- wprowadzono rozróżnienie sieci budowanych w kroku potwierdzenia i kroku falsyfikacji (w kroku potwierdzenia dodawane są jedynie jednostki z faktu weryfikowanego, ponieważ typ relacji ma być wyprowadzony z pozostałych faktów, natomiast w kroku falsyfikacji dodawany jest cały fakt weryfikowany)

Schemat algorytmu wnioskowania został przedstawiony na listingu 5.1.

W pierwszym kroku algorytmu wnioskowania tworzona jest sieć ograniczeń. Funkcja `StworzSiecOgraniczenDoWeryfikacji` tworzy sieć ograniczeń w sposób podobny jak funkcja `StworzSiecOgraniczen` przedstawiona na listingu 4.3 w podrozdziale 4.2. Modyfikacje funkcji polegają na tym że:

- rachunek w którym tworzone są etykietowania krawędzi sieci (RCC5 lub algebra Allena) jest wybierany na podstawie typu relacji faktu weryfikowanego,
- do sieci trafiają wszystkie modelowania faktów wydobytych,
- modelowanie relacji faktu weryfikowanego **nie jest** wykorzystywane w procesie tworzenia sieci ograniczeń.

Ponownie rozpoczynamy od kroku potwierdzenia. Na stworzonej sieci ograniczeń N uruchamiany jest algorytm PC. Sprawdzane jest, czy sieć będąca wynikiem działania algorytmu PC zawiera relację pustą. Jeśli tak (wykryto sprzeczność), to zwracana jest wartość `UNKNOWN`. Sprzeczność na tym etapie oznacza bowiem, że sprzeczność występuje w jednej z baz wiedzy (ustrukturyzowanej lub nieustrukturyzowanej), ponieważ modelowanie faktu weryfikowanego nie zostało na tym etapie dodane do sieci ograniczeń.

Algorytm 5.1: WeryfikujFakt

```
   Data: Fakt weryfikowany f, Kolekcja faktów wydobytych E
   Result: Jedna z wartości: {TRUE, FALSE, UNKNOWN}
 1 begin
       /* Krok potwierdzenia                                          */
 2     N ← StworzSiecOgraniczenDoWeryfikacji(f, E) ;
 3     r_f ← relacja modelująca typ relacji f ;
 4     if PathConsistent(N) then
 5         if N etykietowany jest relacją r_f między wierzchołkami
           odpowiadającymi jednostkom z f then
 6             return TRUE
 7         end
           /* Krok falsyfikacji                                       */
 8         dodaj etykietowanie r_f do N ;
 9         if not PathConsistent(N) then
10             return FALSE
11         else
12             return UNKNOWN
13         end
14     else
15         return UNKNOWN
16     end
17 end
```

Skutkiem ubocznym działania algorytmu PC jest utworzenie wysubtelnienia sieci (w którym dodano relacje będące wynikiem złożenia, patrz podrozdział 2.1.5). W kolejnym kroku algorytmu wnioskowania sprawdzamy czy sieć ograniczeń zawiera modelowanie faktu weryfikowanego. Jeśli tak, zwracana jest wartość `TRUE`, ponieważ udało się wykazać relację zachodzącą między jednostkami faktu weryfikowanego.

W przeciwnym przypadku rozpoczyna się krok falsyfikacji. Do sieci N dokładana jest krawędź modelująca fakt weryfikowany. Ponownie uruchamiany jest algorytm PC. Jeśli w wyniku jego działania powstanie sieć, która zawiera relację pustą (wykryto sprzeczność), to algorytm zwraca wartość `FALSE`. Wykazano, że nie zachodzi relacja między jednostkami faktu weryfikowanego.

Przykłady działania algorytmu wnioskowania znajdują się w podrozdziałach 5.4.2 oraz 5.5.2 (w ramach przykładów odpowiadania na konkretne pytania).

5.4. Algorytm odpowiadania na pytania w postaci kwerendy

5.4.1. Opis algorytmu

Istotą pytań w postaci kwerendy jest rozstrzygnięcie, czy fakt zawarty w pytaniu (dalej nazywany w skrócie faktem kwerendy) jest sprzeczny bądź niesprzeczny z dostępną bazą wiedzy.

Algorytm odpowiadania na pytania tego typu składa się z następujących kroków:

1. Znajdź fakt pytania w ustrukturyzowanej bazie wiedzy. Jeśli fakt znajduje się w bazie, to zwróć odpowiedź `TRUE`.
2. Uruchom algorytm wnioskowania na fakcie kwerendy (bez faktów wydobytych). Jeśli fakt kwerendy jest zweryfikowany, to zwróć wynik algorytmu wnioskowania jako odpowiedź.
3. Wykorzystaj nieustrukturyzowaną bazę wiedzy do znalezienia faktów wydobytych E.
4. Dla każdego faktu wydobytego e wykonaj:
   a) Jeśli e jest równe faktowi kwerendy, to zwróć odpowiedź `TRUE`,
   b) Uruchom algorytm wnioskowania na fakcie kwerendy wraz z faktem wydobytym e. Jeśli fakt kwerendy jest zweryfikowany, to zwróć wynik algorytmu wnioskowania jako odpowiedź.
5. Jeśli nie udało się zweryfikować faktu kwerendy zwróć odpowiedź `UNKNOWN`.

W pierwszym kroku algorytmu fakt kwerendy wyszukiwany jest w ustrukturyzowanej bazie wiedzy (szukana informacja może być zachowana bezpośrednio w bazie wiedzy). Proces ten nie wymaga wnioskowania.

W kolejnym kroku wykorzystywane jest wnioskowanie wyłącznie na faktach z bazy wiedzy. Na tym etapie nie są wykorzystywane fakty wydobyte, które traktujemy jako źródło wiedzy gorszej jakości.

Jeśli wnioskowanie nie powiedzie się, to rozpoczyna się wykorzystanie nieustrukturyzowanej bazy wiedzy. W pierwszym kroku tworzymy kolekcję faktów wydobytych. Fakty są pozyskiwane za pomocą metod opisanych w podrozdziale 6.1. Następnie każdy z faktów wydobytych wykorzystywany jest pojedynczo w procesie wnioskowania. Oznacza to, że algorytm wykorzystuje wnioskowanie na maksymalnie jednym fakcie wydobytym.[5]

Jeśli fakt kwerendy jest zweryfikowany przez algorytm wnioskowania, to oprócz wartości odpowiedzi zwracana jest również utworzona w procesie wnioskowania sieć ograniczeń (jako wyjaśnienie odpowiedzi). W przypadku, gdy odpowiedź powstała

[5] Nieustrukturyzowana baza wiedzy jest źródłem gorszej jakości, dlatego dopuszczamy wykorzystanie tylko jednego faktu wydobytego, aby uchronić się przed propagacją błędów.

poprzez wykorzystanie faktu wydobytego, dodatkowo zwracany jest fragment tekstu, z którego pozyskano fakt wydobyty.

5.4.2. Przykład działania algorytmu

Przykład 5.1: Pytanie kwerendy

Rozpatrzmy następujące pytanie: *Czy Uniwersytet Adama Mickiewicza znajduje się w Polsce?* Przyjmijmy, że ustrukturyzowana baza wiedzy zawiera następujący fakt:

`Poznań [miasto], jest położony w, Polska [państwo]`

Dodatkowo załóżmy, że w nieustrukturyzowanej bazie wiedzy znajduje się następujący fragment artykułu prasowego:[6]

> [...] na dnie jeziora zalegają teraz nagromadzone z tego wszystkiego osady, które najpierw trzeba by było usunąć. Osady niebezpieczne nie są, badali je naukowcy z Uniwersytetu Adama Mickiewicza w Poznaniu [...]

Rozpatrywane pytanie należy do grupy pytań w postaci kwerendy i jest reprezentowane za pomocą następującego faktu kwerendy: *(Uniwersytet Adama Mickiewicza [jednostka], znajduje się w, Polska [państwo])*. Zadaniem algorytmu odpowiadania na pytanie jest potwierdzenie (lub sfalsyfikowanie) faktu kwerendy.

W pierwszym kroku algorytmu w bazie wiedzy ustrukturyzowanej wyszukiwany jest fakt dokładnie równy faktowi kwerendy. W naszym przykładzie baza wiedzy nie zawiera takiego faktu.

W kolejnym kroku algorytm wykorzystuje wnioskowanie i ustrukturyzowaną bazę wiedzy do znalezienia odpowiedzi. Wnioskowanie rozpoczyna się od kroku potwierdzenia. Algorytm wnioskowania buduje sieć ograniczeń składającą się z dwóch wierzchołków reprezentujących podmiot i dopełnienie faktu kwerendy (tzn. jednostkę *Uniwersytet Adama Mickiewicza* oraz jednostkę *Polska*). Nie jest dodawana krawędź pomiędzy wierzchołkami (jej uzyskanie jest celem kroku potwierdzenia).

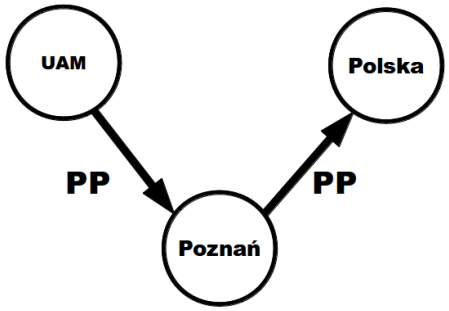
Ponieważ baza wiedzy nie zawiera faktów, których podmiotem jest jedna z wymienionych jednostek wynikowa sieć ograniczeń zawiera tylko dwa wierzchołki (nie zostają dodane żadne fakty). Krok potwierdzenia nie powodzi się (stworzona sieć ograniczeń nie zawiera krawędzi łączącej podmiot i dopełnienie faktu kwerendy).

Rozpoczyna się krok falsyfikacji. Algorytm dodaje cały fakt kwerendy do sieci ograniczeń, tworząc sieć ograniczeń składającą się z dwóch wierzchołków: *Uniwersytet Adama Mickiewicza* oraz *Polska*. Wierzchołki połączone są krawędzią `PP`, pochodzącą z modelowania typu relacji *jest położony w* z faktu kwerendy. Sieć ograniczeń

[6] Źródło: http://www.mmpoznan.pl/5458/2009/6/11/kiedy-wykapiemy-sie-w-jeziorze-swarzedzkim

nie zawiera relacji pustej, co oznacza że krok falsyfikacji również się nie powodzi (nie udało się wykryć sprzeczności).

W następnym kroku algorytmu wykorzystywana jest wiedza wydobyta z artykułu prasowego. Załóżmy, że z fragmentu tekstu pozyskano następujący fakt wydobyty: *(Uniwersytet Adama Mickiewicza [jednostka], znajduje się w, Poznań [miasto])*. Algorytm wnioskowania wykorzystuje ten fakt, tworząc w kroku potwierdzenia sieć ograniczeń składającą się z trzech wierzchołków reprezentujących jednostki z faktu kwerendy oraz faktu wydobytego, tzn.: *Uniwersytet Adama Mickiewicza*, *Poznań*, *Polska*. Dodawana jest relacja `PP` między jednostkami *Uniwersytet Adama Mickiewicza* oraz *Poznań*, która modeluje typ relacji faktu wydobytego. Następnie z bazy wiedzy dodawany jest fakt: *(Poznań [miasto], jest położony w, Polska [państwo])*. Dodanie faktu powoduje dodanie krawędzi `PP` między jednostkami *Poznań* i *Polska*. Sieć ograniczeń przedstawiona jest na rysunku 5.1.



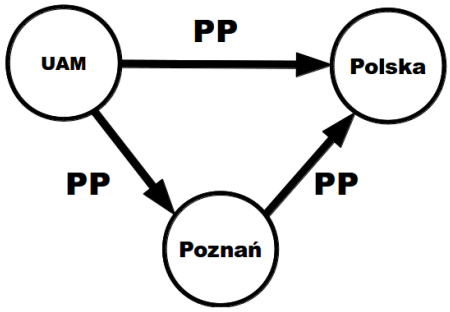
Rysunek 5.1. Sieć ograniczeń stworzona w procesie wnioskowania z wykorzystaniem wiedzy wydobytej (przed uruchomieniem algorytmu PC)

Na powstałej sieci ograniczeń uruchamiany jest algorytm PC, który tworzy wysubtelnioną sieć ograniczeń. Korzystając ze złożenia relacji algorytm PC dodaje etykietowanie PP między wierzchołkami *UAM* oraz *Polska*. Wykorzystywane jest złożenie między wierzchołkami: *UAM*, *Poznań* oraz *Polska* postaci: $PP \circ PP = PP$. Brak jest krawędzi bezpośredniej między wierzchołkami *UAM* oraz *Polska* (co interpretowane jest jako połączenie relacją uniwersalną). Nowe etykietowanie dodane przez algorytm PC ma więc postać:

$$PP \cap \top = PP$$

Wynikowa sieć ograniczeń jest przedstawiona na rysunku 5.2.

Wysubtelniona sieć ograniczeń (będąca wynikiem algorytmu PC) nie zawiera relacji pustej. Algorytm sprawdza, czy uzyskano krawędź między jednostkami faktu kwerendy, która odpowiada modelowaniu typu relacji faktu kwerendy. Sieć ograniczeń zawiera taką krawędź (jest to krawędź `PP`) między jednostkami *Uniwersytet Adama Mickiewicza* oraz *Polska*. Krok potwierdzenia zakończył się sukcesem. Al-



Rysunek 5.2. Sieć ograniczeń stworzona w procesie wnioskowania z wykorzystaniem wiedzy wydobytej (po uruchomieniu algorytmu PC)

gorytm zwraca odpowiedź `TRUE` na zadane pytanie. Jako wyjaśnienie odpowiedzi zwracana jest stworzona sieć ograniczeń oraz fragment tekstu z którego pozyskano fakt wydobyty. Odpowiedź wyświetlana przez system została przedstawiona na rysunku 5.3.

5.5. Algorytm odpowiadania na pytania z warunkami

5.5.1. Opis algorytmu

Pytanie z warunkami reprezentowane jest za pomocą następującej struktury:

- **pytanie bazowe** — przetwarzane pytanie pozbawione odnośników przestrzennych i czasowych,
- **warunki** — kolekcja faktów reprezentująca odnośniki przestrzennej i czasowe.

Warunki są faktami, których podmiot jest niezdefiniowany (warunki wyrażają bowiem fakty, których podmiotem jest samo pytanie bazowe). Przyjmiemy, że niezdefiniowany podmiot możemy utożsamić z podmiotem dowolnego faktu wydobytego z tekstu, który jest źródłem odpowiedzi na pytanie bazowe. Warunki mogą być różnych typów (tzn. w pytaniu mogą występować zarówno warunki przestrzenne i czasowe).

Algorytm odpowiadania na pytania z warunkami wykorzystuje odpowiedź bazową na pytanie bazowe. Odpowiedź bazowa pozyskiwana jest metodami powierzchniowymi opisanymi w podrozdziale 3.4.2.

Głównym celem algorytmu odpowiadania na pytania z warunkami jest weryfikacja warunków. Wykorzystywany jest algorytm wnioskowania opisany w podrozdziale 5.3. Przyjmujemy następujące założenia:

- Jeśli wszystkie warunki zostały zweryfikowane, to:

Rysunek 5.3. Odpowiedź na pytanie *Czy Uniwersytet Adama Mickiewicza znajduje się w Polsce?*

  - — Jeśli wszystkie warunki są prawdziwe (odpowiedź `TRUE`), to odpowiedź bazowa zwracana jest jako odpowiedź wynikowa (wraz z odpowiednimi sieciami ograniczeń).
  - — W przeciwnym przypadku (istnieje warunek zweryfikowany z wynikiem `FALSE`), to **zanegowana** odpowiedź bazowa zwracana jest jako odpowiedź wynikowa (wraz z odpowiednimi sieciami ograniczeń).
- Jeśli choć jeden warunek nie jest zweryfikowany, to nie jest zwracana odpowiedź wynikowa (brak odpowiedzi).

Wszystkie warunki weryfikowane są niezależnie. Wynikiem weryfikacji jest jedna z wartości: `TRUE`, `FALSE` oraz `UNKNOWN`. Weryfikacja warunku c dla odpowiedzi bazowej, której źródłem jest tekst p odbywa się w następujących krokach:

1. Pozyskaj wszystkie relacje z tekstu p tworząc zbiór faktów wydobytych E.[7]
2. Dla każdego faktu wydobytego e z E wykonaj:
   a) Połącz podmiot faktu e z podmiotem faktu c (traktuj jednostki jako równe).
   b) Jeśli e jest równe c (po połączeniu), to zwróć wartość `TRUE` (pozyskany fakt z tekstu spełnia warunek).
   c) W przeciwnym przypadku uruchom algorytm wnioskowania na c z faktem wydobytym e.
   d) Jeśli warunek został zweryfikowany, to zwróć wynik algorytmu wnioskowania.
3. Jeśli nie udało się zweryfikować warunku za pomocą faktów wydobytych, to zwróć `UNKNOWN`.

Podobnie jak w przypadku pytań w postaci kwerendy w procesie wnioskowania dopuszczamy wykorzystanie maksymalnie jednego faktu wydobytego.

Jeśli wszystkie warunki zostaną zweryfikowane, to oprócz wartości odpowiedzi zwracane są również sieci ograniczeń utworzone w procesie wnioskowania oraz fragment tekstu z którego pozyskano fakt wydobyty. Dane służą do wygenerowania wyjaśnienia odpowiedzi.

5.5.2. Przykłady działania algorytmu

Przykład 5.2: Pytanie z warunkiem czasowym

Rozpatrzmy następujące pytanie: *Czy Jan Fabre był w zeszłym roku uczestnikiem Festiwalu Malta?* Przyjmijmy, że pytanie zadawane jest w roku 2013.

Załóżmy, że w nieustrukturyzowanej bazie wiedzy znajduje się następujący fragment artykułu prasowego (data opublikowania: 1 lipca 2011 r.):[8]

[7] Pod uwagę brane są tylko typy relacji zgodne z typem relacji warunku (np. dla warunku przestrzennego pozyskiwane są tylko relacje przestrzenne).

[8] Źródło: http://www.mmpoznan.pl/377344/2011/7/1/festiwal-malta-juz-za-chwile

> Innym wielkim twórcą teatralnym, którego zobaczymy na Malcie jest Jan Fabre. — W zeszłym roku gościł u nas z monodramem, tym razem zobaczymy jego zupełnie inne przedstawienie: duże, wręcz monumentalne — zapowiada Anna Reichel.

Rozpatrywane pytanie należy do grupy pytań z warunkami i jest reprezentowane jako następująca struktura:

- **Pytanie bazowe** — *Czy Jan Fabre był uczestnikiem Festiwalu Malta?*
- **Warunki** — *(*, w trakcie, zeszły rok [data]=2012)*

Zauważmy, że wykorzystując kontekst zadającego pytanie frazie *w zeszłym roku* została przypisana wartość odpowiadająca rokowi 2012.

W pierwszym kroku odpowiadania na pytanie znajdowana jest odpowiedź bazowa dla pytania bazowego (wykorzystywane są mechanizmy powierzchniowe). Na podstawie fragmentu tekstu przedstawionego na początku przykładu algorytm znajduje odpowiedź bazową `TRUE`.

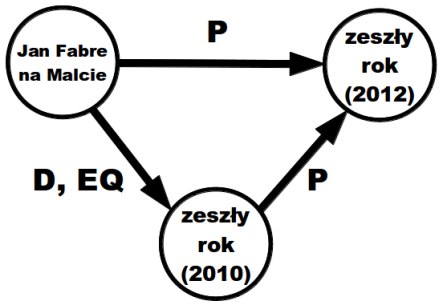
Rozpoczyna się proces weryfikacji warunków pytania. Zadaniem algorytmu jest rozstrzygnięcie czy warunek jest zgodny lub sprzeczny z wiedzą zawartą w źródłach odpowiedzi bazowej (w naszym przykładzie jest to fragment artykułu). Dla warunku uruchomiony zostaje algorytm wnioskowania.

Wykorzystywany jest następujący fakt wydobyty z artykułu:[9] *(Jan Fabre gościł z monodramem [wydarzenie], w trakcie, zeszły rok [data]=2010)*. Zwróćmy uwagę, że w tym przypadku frazie *w zeszłym roku* przyporządkowano wartość *roku 2010* (wynika to z uwzględnienia kontekstu dokumentu). Rozpoczyna się krok potwierdzenia faktu. Tworzona jest sieć ograniczeń składająca się z trzech wierzchołków: *wydarzenie*, *zeszły rok=2010*, *zeszły rok[data]=2012*. Dodawana jest krawędź pochodząca z modelowania typu relacji faktu wydobytego. Typ relacji *w trakcie* jest dla dat modelowany za pomocą relacji `D, EQ`. Reguły semantyczne dodają krawędź `P` między wierzchołkami *zeszły rok[data]=2010* oraz *zeszły rok[data]=2012*. Po uruchomieniu algorytmu PC otrzymujemy wysubtelnioną sieć ograniczeń przedstawioną na rysunku 5.4.

Krawędź między wierzchołkami odpowiadającymi jednostkom z przetwarzanego warunku jest etykietowana relacją `P`. Etykietowanie to dodał algorytm PC poprzez zastosowanie złożenia relacji między wierzchołkami: *Jan Fabre*, *zeszły rok (2010)* oraz *zeszły rok (2012)*:

$$\{D, EQ\} \circ P = (D \circ P) \cup (EQ \circ P)$$

[9] W ogólności algorytm może wykorzystać w tym miejscu bazę wiedzy czasowej, jednak ponieważ w implementacji algorytmu nie wykorzystaliśmy takiej bazy, to w opisie przykładów pytań z aspektem czasowym pomijamy ten krok.



Rysunek 5.4. Sieć ograniczeń stworzona w procesie wnioskowania (krok potwierdzenia)

Z tablicy złożeń relacji dla algebry Allena mamy:

$$D \circ P = P$$

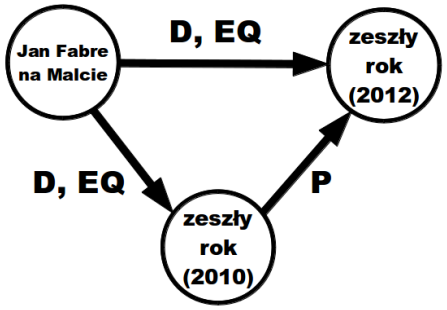
$$EQ \circ P = P$$

Stąd:

$$\{D, EQ\} \circ P = P$$

Etykietowanie relacją P nie jest zgodne z relacją modelującą typ relacji warunku (którą jest `D, EQ`). Krok potwierdzenia faktu nie powodzi się.[10]

Algorytm wnioskowania przystępuje do kroku falsyfikacji faktu. Do sieci ograniczeń zostaje dodany cały fakt z warunku pytania (razem z modelowaniem typu relacji) oraz fakt wydobyty. Sieć ograniczeń zawiera trzy wierzchołki odpowiadające wydarzeniu, datom 2010 rok oraz 2012 rok. Utworzona sieć ograniczeń przedstawiona jest na rysunku 5.5.



Rysunek 5.5. Przykład 5.2: Sieć ograniczeń stworzona w procesie wnioskowania (krok falsyfikacji)

W wyniku działania algorytmu PC powstaje sieć, która zawiera relację pustą. Biorąc złożenie relacji między wierzchołkami: *Jan Fabre*, *zeszły rok (2010)* oraz *zeszły*

[10] Powyższy wynik możemy interpretować jako wykazanie, że wydarzenie miało miejsce **przed** rokiem 2012.

rok (2012) otrzymujemy:

$$\{D, EQ\} \circ P = P$$

Tymczasem krawędź łącząca bezpośrednio wierzchołki *Jan Fabre* oraz *zeszły rok (2012)* etykietowana jest zbiorem relacji $\{D, EQ\}$. Część wspólna obu zbiorów relacji jest pusta. Algorytm PC wykrył sprzeczność.

Krok falsyfikacji kończy się sukcesem. Odpowiedź bazowa zostaje zanegowana, a jako wyjaśnienie zwrócona zostaje utworzona sieć ograniczeń. Odpowiedź wyświetlana przez system została przedstawiona na rysunku 5.6.

Przykład 5.3: Pytanie z warunkiem czasowym i przestrzennym

Rozpatrzmy następujące pytanie: *Czy w tym roku w Wielkopolsce kot pogryzł psa?* Przyjmijmy, że pytanie zadawane jest w roku 2013. Ustrukturyzowana baza wiedzy przestrzennej zawiera następujące fakty:

```
Wilda [dzielnica], jest położony w, Poznań [miasto]
Poznań [miasto], jest położony w, Wielkopolska
       [jednostka administracyjna 1.˜rzędu]
```

Załóżmy, że w nieustrukturyzowanej bazie wiedzy znajduje się następujący fragment artykułu prasowego (data opublikowania: 19 lutego 2012 r.):[11]

> Kot pogryzł psa na Wildzie.
>
> Agresywny kot pogryzł mojego psa! — takie nietypowe zgłoszenie otrzymał w piątek patrol Straży Miejskiej.

Rozpatrywane pytanie należy do grupy pytań z warunkami i jest reprezentowane jako następująca struktura:

- **Pytanie bazowe** — *Czy kot pogryzł psa?*
- **Warunki**:
  1. *(*, jest położony w, Wielkopolska [jednostka administracyjna 1. rzędu])*
  2. *(*, w trakcie, ten rok [data]=2013)*

W pierwszym kroku poszukiwania odpowiedzi wykorzystywane są metody powierzchniowe do znalezienia odpowiedzi na pytanie bazowe. Wykorzystywany jest fragment artykułu z nieustrukturyzowanej bazy wiedzy do stworzenia odpowiedzi bazowej `TRUE`.

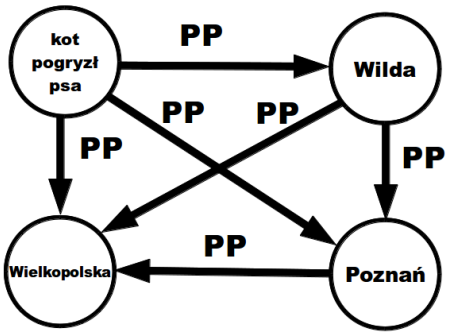
Rozpoczyna się proces weryfikacji warunków pytania. W tym przykładzie należy zweryfikować dwa warunki. Warunki weryfikowane są niezależnie.

W procesie weryfikacji warunku przestrzennego: *(*, jest położony w, Wielkopolska [jednostka administracyjna 1. rzędu])* algorytm wnioskowania wykorzystuje ustruk-

[11] Źródło: http://www.mmpoznan.pl/403569/2012/2/19/kot-pogryzl-psa-na-wildzie

Rysunek 5.6. Odpowiedź na pytanie *Czy Jan Fabre był w zeszłym roku uczestnikiem Festiwalu Malta?*

turyzowaną bazę wiedzy przestrzennej oraz fakt wydobyty z tekstu: *(kot pogryzł psa [wydarzenie], jest położony w, Wilda [dzielnica])*. Następuje utożsamienie podmiotu warunku z podmiotem faktu wydobytego. Na sieci ograniczeń uruchamiany jest algorytm PC. Wynikowa sieć nie zawiera relacji pustej. W wysubtelnionej sieci ograniczeń dodano trzy relacje PP korzystając ze złożenia relacji $PP \circ PP = PP$. Wysubtelniona sieć ograniczeń jest zaprezentowana na rysunku 5.7.



Rysunek 5.7. Przykład 5.3: Wysubtelniona sieć ograniczeń stworzona w procesie wnioskowania dla warunku przestrzennego

Algorytm wnioskowania sprawdza etykietowanie krawędzi między wierzchołkami odpowiadającymi podmiotowi i dopełnieniu z warunku. Krawędź etykietowana jest relacją `PP`, która modeluje typ relacji warunku. Oznacza to, że algorytm wnioskowania korzystając z dostępnych baz wiedzy wykazał prawdziwość warunku. Krok potwierdzenia kończy się sukcesem.

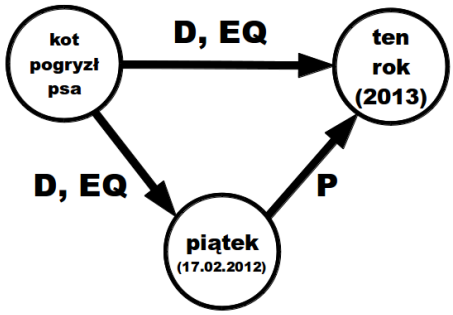
Kolejnym krokiem jest weryfikacja drugiego warunku pytania. Jest to warunek czasowy: *(*, w trakcie, ten rok [data]=2013)*. Wykorzystywany jest następujący fakt wydobyty z tekstu: *(kot pogryzł psa [wydarzenie], w trakcie, piątek [data]=17.02.2012)*. Zauważmy, że przypisując frazie *w piątek* wartość daty równą *17.02.2012* wykorzystany został kontekst dokumentu.

Krok potwierdzający nie powodzi się. Algorytm wnioskowania wykazał, że wydarzenie miało miejsce przed rokiem 2013. W kroku falsyfikacji tworzona jest sieć ograniczeń zaprezentowana na rysunku 5.8.

Uruchomienie algorytmu PC na rozpatrywanej sieci prowadzi do powstania relacji pustej. Zachodzi podobna sytuacja jak w przykładzie 5.2. Krok falsyfikacji kończy się sukcesem. Warunek zostaje potwierdzony jako fałszywy. Jako wyjaśnienie zwracany jest utworzona sieć ograniczeń.

Algorytm integruje wyniki weryfikacji dwóch warunków. Pierwszy warunek został zweryfikowany jako prawdziwy, drugi — jako fałszywy. Oznacza to, że ostatecznie wynikiem weryfikacji jest wartość fałsz. Odpowiedź bazowa zostaje zanegowana. Algorytm zwraca odpowiedź `FALSE` wraz z dwiema sieciami ograniczeń pochodzącymi

z procesu wnioskowania.[12] Odpowiedź wyświetlana przez system została przedstawiona na rysunku 5.9.



Rysunek 5.8. Przykład 5.3: Sieć ograniczeń stworzona w procesie wnioskowania dla warunku czasowego

5.6. Analiza algorytmu wnioskowania

5.6.1. Algorytm PC w procesie wnioskowania

Spostrzeżenie 5.1. *Dla faktów przestrzennych algorytm PC (wykorzystywany w algorytmie wnioskowania) daje zawsze poprawną odpowiedź.*

Powyższe spostrzeżenie wynika bezpośrednio z własności opisanych w podrozdziale 4.2.3. Podobnie jak w algorytmie ujednoznaczniania w sieci ograniczeń wykorzystywanej w algorytmie wnioskowania wykorzystujemy relacje pochodzące z podklasy podatnej rachunku RCC5 (podklasy $\hat{H}_5$). Algorytm PC uruchomiony na tym zbiorze relacji daje zawsze poprawną odpowiedź.

Niestety podobne spostrzeżenie dla wnioskowania czasowego nie jest w ogólności prawdziwe. W przypadku relacji czasowych wykorzystujemy efektywnie wszystkie relacje podstawowe algebry Allena (ze względu na zastosowanie arytmetyki dat) wraz z relacją uniwersalną. Na takim zbiorze algorytm PC w ogólności nie daje zawsze odpowiedzi poprawnej. Jednakże w przypadku wiedzy czasowej nie jest wykorzystywana ustrukturyzowana baza wiedzy, co w znaczny sposób redukuje wielkość sieci ograniczeń.

Spostrzeżenie 5.2. *W przypadku niewykorzystania bazy wiedzy ustrukturyzowanej w algorytmie wnioskowania budowane są sieci ograniczeń o maksymalnej liczbie wierzchołków równej* $2(n+1)$*, gdzie* n *oznacza liczbę faktów wydobytych.*

Ponieważ nie jest wykorzystywana ustrukturyzowana baza wiedzy, to źródłem jednostek (których odpowiednikiem są wierzchołki sieci ograniczeń) są jedynie fakty

[12] Wynik algorytmu można interpretować w następujący sposób: **prawdą** jest że w Wielkopolsce *kot pogryzł psa*, **ale nieprawda** że wydarzenie to miało miejsce w 2013 roku.

Rysunek 5.9. Odpowiedź na pytanie *Czy w tym roku w Wielkopolsce kot pogryzł psa?*

będące danymi algorytmu. Na wejściu algorytmu mamy zawsze jeden fakt (fakt weryfikowany) oraz kolekcję n opcjonalnych faktów wydobytych (wśród których niektóre jednostki mogą się pokrywać). Stąd liczba wierzchołków sieci ograniczeń ograniczona jest w takim przypadku z góry przez $2(n + 1)$.

Spostrzeżenie 5.3. *Dla faktów czasowych, jeżeli w algorytmie wnioskowania wykorzystywany jest najwyżej jeden fakt wydobyty, którego podmiot jest utożsamiany z podmiotem faktu weryfikowanego, to algorytm PC (wykorzystywany w algorytmie) daje zawsze poprawną odpowiedź.*

Zauważmy, że w taki właśnie sposób wykorzystano algorytm wnioskowania dla wiedzy czasowej. Używany jest dokładnie jeden fakt wydobyty, którego podmiot jest zawsze utożsamiany z podmiotem faktu weryfikowanego. Zgodnie ze spostrzeżeniem 5.2 oznacza to, że utworzona sieć ograniczeń ma maksymalnie 3 wierzchołki (dwa wierzchołki zostały utożsamione). Algorytm PC na tak niewielkiej sieci daje zawsze poprawną odpowiedź.

5.6.2. Złożoność algorytmu wnioskowania

Dla algorytmu wnioskowania, podobnie jak w przypadku algorytmu ujednoznaczniania (patrz podrozdział 4.2.3) możemy sformułować następujące spostrzeżenie:

Własność algorytmu 5.4. *Funkcja StworzSiecOgraniczenDoWeryfikacji ma złożoność wielomianową względem liczby jednostek w bazie wiedzy.*

Jak wspomniano w podrozdziale 5.3, funkcja ta jest tożsama z funkcją *StworzSiecOgraniczen* wykorzystywaną w algorytmie ujednoznaczniania. Zastosowane modyfikacje nie mają istotnego wpływu na złożoność.

Własność algorytmu 5.5. *Algorytm wnioskowania przedstawiony na listingu 5.1 ma złożoność wielomianową ze względu na liczbę jednostek w bazie wiedzy oraz wielomianową ze względu na liczbę jednostek użytych w procesie wnioskowania.*

Złożoność algorytmu wnioskowania wynika z:
1. Złożoności funkcji *StworzSiecOgraniczenDoWeryfikacji*,
2. Złożoności algorytmu PC wykorzystanym w procesie wnioskowania (jest ona wielomianowa ze względu na liczbę wierzchołków sieci ograniczeń).

5.7. Podsumowanie

W niniejszym rozdziale przedstawiłem autorskie algorytmy odpowiadania na pytania rozstrzygnięcia z aspektem czasowym i przestrzennym. Algorytmy odpowiadania wykorzystują rachunki ograniczeń: rachunek RCC5 (dla wiedzy przestrzennej)

oraz algebrę Allena (dla wiedzy czasowej). Omówiłem na wybranych przykładach działanie algorytmów. Ewaluacja algorytmów na korpusie pytań została przedstawiona w rozdziale 6.

Rozdział 6

Opis systemu Hipisek

W niniejszym rozdziale przedstawiam opis techniczny kluczowych elementów systemu Hipisek, dotyczących przetwarzania pytań rozstrzygnięcia z aspektem czasowym i przestrzennym.

Rozdział rozpoczynam od opisania zagadnień implementacyjnych systemu Hipisek. Opis obejmuje moduły służące do oznaczania i normalizacji jednostek czasowych i przestrzennych w tekście niesformatowanym, pozyskiwania faktów czasowych i przestrzennych z tekstu niesformatowanego oraz przygotowania sieci ograniczeń wykorzystywanej w algorytmach odpowiadania na pytania.

W drugiej części rozdziału przedstawiam ewaluację implementacji algorytmu odpowiadania na pytania rozstrzygnięcia z aspektem czasowym i przestrzennym w systemie Hipisek.

Pierwszym z omówionych eksperymentów jest ewaluacja jakości bazy wiedzy przestrzennej zebranej z wykorzystaniem algorytmów opisanych w rozdziale 4. Ewaluacja polegała na manualnym sprawdzeniu losowej próbki jednostek z bazy wiedzy.

Następnie przedstawiam eksperyment ewaluacji algorytmów odpowiadania na pytania rozstrzygnięcia. Ewaluacja została przeprowadzona na zebranym korpusie pytań. Opisuję, w jaki sposób zebrano korpus pytań oraz przedstawiam jego charakterystykę. Rozdział kończy przedstawienie wyników systemu na korpusie pytań testowych oraz wyników pomiaru szybkości działania systemu.

6.1. Zagadnienia implementacyjne

W niniejszym podrozdziale opisuję wybrane moduły systemu Hipisek, wykorzystywane w procesie pozyskiwania odpowiedzi na pytania rozstrzygnięcia. Opisanymi modułami są:

- moduł oznaczania jednostek czasowych i przestrzennych w tekście (HipiNER),
- moduł normalizacji jednostek czasowych i przestrzennych (HipiNEN),
- moduł wydobywania faktów czasowych i przestrzennych z tekstu (HipiRE),
- moduł przygotowujący sieć ograniczeń dla wybranego typu relacji.

6.1.1. Oznaczanie jednostek czasowych i przestrzennych

Oznaczanie jednostek czasowych i przestrzennych w tekście jest szczególnym przypadkiem zadania rozpoznawania jednostek nazwanych (ang. *Named Entity Recognition*).

Zadanie oznaczenia jednostek czasowych i przestrzennych polega na znalezieniu ciągłych fragmentów tekstu odnoszących się do informacji czasowych i przestrzennych. Oznaczone fragmenty tekstu nazywane są **jednostkami nazwanymi** (od ang. *named entities*). Przykładowymi jednostkami nazwanymi są:

- *Czy **Poznań** znajduje się w **Polsce**?* — jednostki przestrzenne, odnoszące się odpowiednio do miasta oraz państwa.
- *Czy w **2012 roku** przewidziano koniec świata?* — jednostka czasowa, odnosząca się do roku 2012.
- *Czy w **tym roku** w Wielkopolsce kot pogryzł psa?* — jednostka czasowa (kontekstowa), odnosząca się do roku w którym wypowiedziano dane zdanie.

Moduł rozpoznawania jednostek nazwanych w systemie Hipisek (**HipiNER**, od ang. *Hipisek Named Entity Recognition*) został oparty na dwóch mechanizmach rozpoznających:

- **gazeterze** — wykorzystującym bazę wiedzy przestrzennej,
- **mechanizmie regułowym** — wykorzystującym zbiór reguł.

Gazeter modułu HipiNER

Gazeter modułu HipiNER wykorzystuje bazę wiedzy przestrzennej do poszukiwania nazw z bazy wiedzy występujących w przetwarzanym fragmencie tekstu. W celu usprawnienia mechanizmu dla nazw występujących w formie odmienionej (np. *w Poznaniu*) wykorzystano lematyzator (korzystający ze słownika SJP.pl[1] poszerzonego o materiał utworzony w projekcie Dylemat [GW2009] i Logofag [Wal2010]) oraz algorytm odmiany przez analogię (służący do znalezienia formy podstawowej wyrazów nieznajdujących się w słowniku).

Oprócz oznaczenia jednostek w tekście gazeter wykonuje jednocześnie zadanie normalizacji jednostek (patrz podrozdział 6.1.2). W tym przypadku normalizacja polega na przyporządkowaniu oznaczonych fragmentów tekstu do odpowiednich jednostek znajdujących się w bazie wiedzy.

[1] Słownik SJP.pl udostępniany na licencji GPL http://www.sjp.pl

Mechanizm regułowy modułu HipiNER

Mechanizm regułowy został napisany z wykorzystaniem parsera płytkiego Puddle, będącego częścią pakietu PSI-Toolkit [Jas2012] [GJJD2012].[2] Parser płytki Puddle [Man2009] został oparty o parser SPADE [BP2008].

Reguła rozpoznawania składa się z zestawu wyrażeń regularnych, które uruchamiane są na stokenizowanym tekście.[3] Opcjonalnie wykorzystywane są informacje językowe (np. forma podstawowa wyrazu, część mowy).

Przykładową regułą modułu HipiNER jest:

```
Rule: "month name"
Match: [base~"(?:styczeń|luty|marzec|kwiecień|maj| \
               czerwiec|lipiec|sierpień|wrzesień| \
               październik|listopad|grudzień)"];
Eval: group(NE_MONTH_NAME, 1);
```

Powyższa reguła, o nazwie *month name*, znajduje token, którego forma podstawowa odpowiada jednej z nazw miesięcy. Następnie token ten oznaczany jest typem *nazwa miesiąca* (`NE_MONTH_NAME`).

Reguły mogą korzystać także z oznaczeń innych reguł. Na przykład poniższa reguła wykorzystuje oznaczenie nazw miesięcy (wykonane np. przez regułę o nazwie *month name*):

```
Rule "12 sierpnia 2001r."
Match: [orth~"(?:3[01]|[12][0-9]|0?[1-9])"] \
       [type=NE_MONTH_NAME] \
       [orth~"[0-9]{4}"] \
       [orth~"roku|r\.?|rok"]?;
Eval: group(NE_DATE,1);
```

Powyższa reguła, o nazwie *12 sierpnia 2001r.*, znajduje ciąg tokenów składający się z trzech lub czterech elementów. Forma ortograficzna pierwszego tokenu musi spełniać wyrażenie regularne oznaczające liczbę dni w dacie. Drugi token musi być oznaczony typem *nazwa miesiąca* (`NE_MONTH_NAME`). Forma ortograficzna trzeciego tokenu musi spełniać wyrażenie regularne oznaczające rok (sekwencja czterech cyfr). Ostatni token (opcjonalny) wyraża różne warianty zapisania wyrazu *rok* (w formie podstawowej, odmienionej lub skróconej). Ciąg tokenów spełniających kryteria powyższej reguły oznaczany jest typem *data* (`NE_DATE`).

Przykładowymi fragmentami tekstu, do którego dopasowuje się reguła są:

---

[2] http://psi-toolkit.wmi.amu.edu.pl

[3] Przed uruchomieniem modułu HipiNER uruchamiany jest tokenizator pakietu PSI-Toolkit.

- 16 czerwca 1904
- 16 czerwiec 1904
- 16 czerwca 1904 r.

Mechanizm regułowy służy także do oznaczenia nazw jednostek przestrzennych, które nie występują w bazie wiedzy (czyli takich, których nie oznaczono mechanizmem gazeter). Tak oznaczone jednostki nie są normalizowane (patrz podrozdział 6.1.2), ale są wykorzystywane w procesie pozyskiwania faktów przestrzennych (patrz podrozdział 6.1.3). Przykładową regułą tego typu jest:

```
Rule "ulica Lipowa"
Match: [base~"ulica|ul[.]"/i] \
       [orth~"[A-ZĄĆĘŁŃÓŚŹŻ][A-ZĄĆĘŁŃÓŚŹŻa-ząćęłńóśźż]+|[0-9]+"]+;
Eval: group(NE_ADDRESS,2);
```

Powyższa reguła, o nazwie *ulica Lipowa*, oznaczy ciąg tokenów typem *adres* (`NE_ADDRESS`). Ciąg tokenów składa się z tokenu o formie bazowej *ulica* lub *ul.* (w sprawdzaniu ignorowana jest wielkość liter, co wyrażone jest modyfikatorem `/i` umieszczonym po wyrażeniu regularnym) oraz niepustego ciągu tokenów będących wyrazami pisanymi wielką literą lub liczbą.

Reguły rozpoznawania wspierają także ujednoznacznianie jednostek przestrzennych w procesie normalizacji (patrz podrozdział 6.1.2). Reguły tego typu służą do określenia typu oznaczonej jednostki na podstawie kontekstu.

Rozpatrzmy następujący przykład reguły:

```
Rule "wieś Poznań"
Left: [base~"wieś"/i];
Match: [orth~"[A-Zążśźęćółńą̇ŻŚŹĘĆÓŁŃ][a-zążśźęćółń]+"]+;
Eval: group(NE_VILLAGE, 2);
```

Za pomocą powyższej reguły oznaczany jest niepusty ciąg tokenów, których forma tekstowa jest wyrazem pisanym wielką literą, jeżeli przed oznaczonym ciągiem wystąpi wyraz o formie podstawowej *wieś*. Reguła ta dopasowuje się na przykład do fragmentu tekstu: *wieś Poznań*. Dzięki zastosowaniu tej reguły normalizator odrzuci jedną z możliwych interpretacji nazwy *Poznań* (jako miasto).[4]

6.1.2. Normalizacja jednostek czasowych i przestrzennych

W systemie Hipisek normalizacja jednostek czasowych i przestrzennych polega na transformacji oznaczonych jednostek czasowych i przestrzennych do postaci formalnej, niezbędnej do dalszego przetwarzania. W procesie normalizacji można wyróżnić dwa aspekty:

[4] Dokładny opis powyższego problemu znajduje się w podrozdziale 6.1.2.

- normalizację **jednostek przestrzennych**, która polega na przyporządkowaniu oznaczonej jednostce odpowiedniej jednostki z bazy wiedzy przestrzennej,
- normalizację **jednostek czasowych**, która polega na utworzeniu logicznej reprezentacji danej jednostki.

Zadanie normalizacji wykonuje moduł HipiNEN (od ang. *Hipisek Named Entity Normalization*).

Jednostki przestrzenne

W przypadku normalizacji jednostek przestrzennych przetwarzane są wyłącznie jednostki, które oznaczone były za pomocą mechanizmu typu gazeter (czyli takie, których nazwa znajduje się w zebranej bazie wiedzy). W takim przypadku zadanie normalizacji redukuje się do odpowiedniego zapytania do bazy wiedzy przestrzennej.

Problemem w przypadku normalizacji jednostek przestrzennych jest niejednoznaczność nazw (porównaj identyczny problem w przypadku zbierania bazy wiedzy przestrzennej opisany w podrozdziale 4.2). Niektóre nazwy mogą odnosić się do różnych pojęć. Na przykład nazwa *Poznań* odnosi się do stolicy Wielkopolski lub niewielkiej wsi w województwie Lubelskim.

Jako częściowe rozwiązanie tego problemu zastosowano zestaw heurystyk ujednoznaczniających. W przypadku gdy jedna nazwa odnosi się do więcej niż jednej jednostki w bazie wiedzy, jednostki te sortowane są według zestawu heurystyk. Ostatecznie wynikiem normalizacji jest jedna jednostka z bazy wiedzy dla danej nazwy (o najwyższej ocenie według zaagregowanej oceny heurystyk).

Wykorzystano następujące heurystyki:

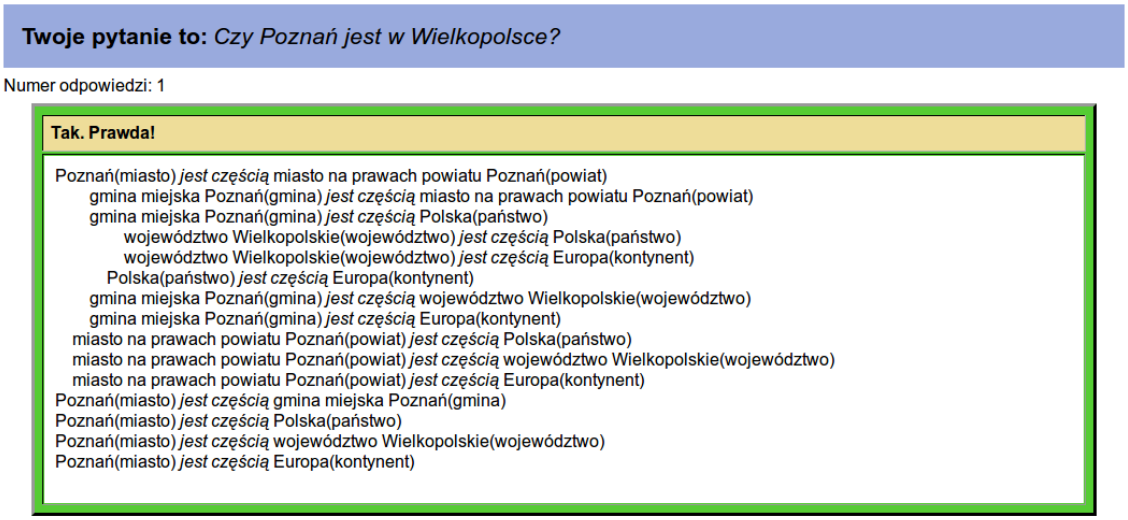
- występowanie słów kluczowych — w przypadku wykrycia pewnych słów kluczowych w pobliżu oznaczonej jednostki (zwykle przed jej oznaczeniem) premiowane są jednostki, o typie zgodnym[5] z typem przypisanym do słowa kluczowego (np. wystąpienie słowa *osada* przed nazwą *Poznań* spowoduje premiowanie jednostki o typie *wieś*),
- priorytety typów jednostek — niektórym typom jednostek mają wyższy priorytet (np. typy kontynent i państwo mają ustawiony wysoki priorytet),
- kryterium liczby ludności — jednostki przestrzenne o większej liczbie ludności są traktowane jako bardziej prawdopodobna interpretacja,
- kryterium zawierania — zakładamy że kontekstem przestrzennym jest Polska, dlatego jednostki położone na terenie Polski są premiowane.

Działanie heurystyk ujednoznaczniających ilustrują następujące przykłady:

[5] Pojęcie zgodnych typów jednostek zostało wprowadzone w podrozdziale 4.1.4

- *Czy Poznań jest w Wielkopolsce?*
- *Czy **wieś** Poznań jest w Wielkopolsce?*

W pierwszym przypadku system Hipisek utożsamia nazwę *Poznań* z nazwą miasta będącego stolicą województwa Wielkopolskiego. Do ujednoznacznienia wykorzystano heurystykę wykorzystującą liczbę ludności oraz priorytety typów jednostek (typ *miasto* ma ustawiony wyższy priorytet niż typ *wieś*). Odpowiedzią na tak zinterpretowane pytanie jest **tak**. Z wyjaśnienia dostarczanego przez system dowiadujemy się, że *Poznań* jest położony w *gminie miejskiej Poznań*, która jest położona w *województwie Wielkopolskim*. Odpowiedź systemu Hipisek została przedstawiona na rysunku 6.1.



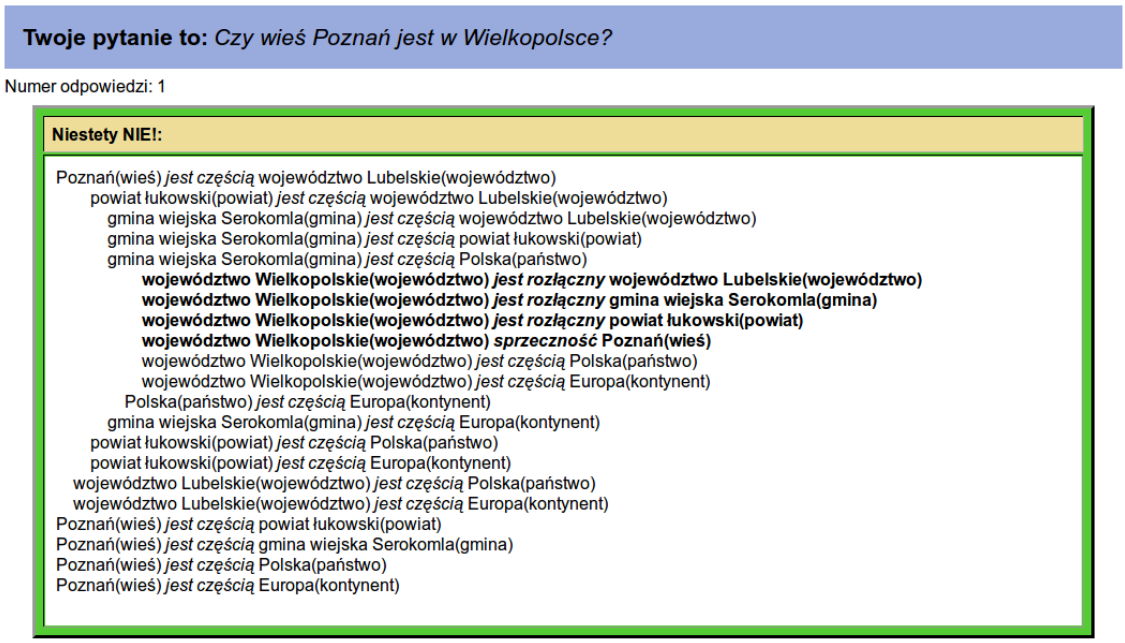
Rysunek 6.1. Odpowiedź na pytanie *Czy Poznań jest w Wielkopolsce?*

W drugim przypadku system Hipisek utożsamia nazwę *Poznań* z nazwą **wsi**. Odpowiedzią na tak zinterpretowane pytanie jest **nie**. Z wyjaśnienia dostarczanego przez system dowiadujemy się, że *wieś Poznań* jest położona w *gminie Serokomla*, która jest położona w *powiecie łukowskim*, który jest położony w *województwie Lubelskim*. Ponieważ każde dwa województwa są regionami rozłącznymi, system stwierdził nieprawdziwość hipotezy zawartej w pytaniu. Odpowiedź systemu Hipisek została przedstawiona na rysunku 6.2.

Jednostki czasowe

Normalizacja jednostek czasowych w systemie Hipisek polega na transformacji oznaczonej jednostki do formalnej reprezentacji daty. Reprezentacja ta została oparta o uproszczony format TIMEX3 (zredukowany do dat). Data w systemie Hipisek składa się z następujących pól:

- rok,



Rysunek 6.2. Odpowiedź na pytanie *Czy wieś Poznań jest w Wielkopolsce?*

- numer miesiąca w roku,
- numer dnia w miesiącu,
- numer dnia w tygodniu,
- godzina,
- minuta,
- sekunda.

Każde z pól może mieć jedną z dwóch typów wartości:

- wartość liczbowa,
- wartość `niezdefiniowana` — brak informacji na temat wartości danego pola.

Mechanizm transformacji

Transformacja jednostek czasowych do formalnej reprezentacji jest przeprowadzana przez mechanizm regułowy. Format reguł został oparty o formalizm reguł SPADE [BP2008].

Reguła transformacji składa się z czterech elementów:

- **nazwy reguły**,
- **typu jednostki**, dla której uruchamiana jest reguła,
- **sekcji match**, która definiuje do jakiego ciągu tokenów dopasuje się reguła,
- niepustego ciągu **instrukcji konwersji**.

Instrukcje konwersji są predefiniowanymi operacjami, które mogą być uruchamiane na wybranych tokenach przetwarzanej jednostki. Obejmują następujące operacje:

- ustawienie jednego z pól daty,
- operacje arytmetyczne na dacie i dacie kontekstowej,
- modyfikacje tokenu (np. usunięcie znaków, podział według wyrażenia regularnego, normalizacja wielkości liter, pobranie formy bazowej wyrazu itp.),
- transformacja nazw (np. nazwy miesiąca na numer miesiąca).

Przykładową regułą transformującą jest:

```
Rule: day_number + month_name
Entity: date
Match: <used~([01]|[12][0-9]|0?[1-9])> <ne=month_name>
Convert: 1 => to_property('day_number')
Convert: 2 => month_name_to_month_number
```

Powyższa reguła działa na jednostkach, które oznaczono typem *data* (`date`). Reguła jest dopasowywana do jednostek składających się z dwóch tokenów. Forma tekstowa pierwszego tokenu musi spełniać wyrażenie regularne będące opisem dopuszczalnej liczby dni w miesiącu (od 1 do 31). Drugi token powinien być jednostką nazwaną o typie *nazwa miesiąca* (`month_name`). Na dopasowanej jednostce uruchomione są dwie instrukcje konwersji:

- pobranie wartości tekstowej pierwszego tokenu i zapisanie do wartości pola *numer dnia w miesiącu*,
- transformacja nazwy miesiąca pobranej z formy tekstowej drugiego tokenu do wartości pola *numer miesiąca w roku*.

Opisywana reguła uruchomiana na oznaczonej jednostce czasowej o postaci tekstowej: *16 czerwca*, utworzy formalną reprezentację daty, w której ustawione są następujące wartości pól:

- numer miesiąca w roku — wartość `6`,
- numer dnia w miesiącu — wartość `16`,
- pozostałe pola — wartość `niezdefiniowana`.

Problem kontekstu czasowego

W procesie normalizacji jednostek czasowych niezbędne jest uwzględnienie kontekstu. W systemie Hipisek wyróżniono dwa rodzaje kontekstu:

- kontekst wypowiedzi — aktualny czas,
- kontekst dokumentu — data opublikowania dokumentu z którego pochodzi przetwarzany tekst.

Dzięki uwzględnieniu dwóch rodzajów kontekstu jednostka czasowa reprezentująca względne wyrażenie czasowe (np. *w zeszłym roku*) jest normalizowana do różnych

wartości, w zależności od kontekstu, w jakim została użyta (porównaj przykłady pytań z warunkami czasowymi w podrozdziale 5.5.2).

Z problemem obsługi kontekstu związany jest problem identyfikacji *fokusu pytania* (patrz opis modelu Lehnert w podrozdziale 3.1.3). Problem ten ilustruje następujący przykład pytania:

> Czy w lutym kot pogryzł psa?

W zależności od *fokusu pytania* pytanie może być interpretowane dwojako:

1. *Czy w lutym [**tego roku**] kot pogryzł psa?*
2. *Czy w lutym [**dowolnego roku**] kot pogryzł psa?*

Jednostka *lutym* jest przetransformowana mechanizmem regułowym na datę, w której zdefiniowane jest jedno pole (*numer miesiąca w roku*). Pole *rok* jest niezdefiniowane, co interpretowane jest jako dowolna wartość. Oznacza to, że pytanie *Czy w lutym kot pogryzł psa?* jest interpretowane przez system jako:

> Czy w lutym **dowolnego roku** kot pogryzł psa?

W obecnej wersji systemu druga interpretacja pytania (*w lutym tego roku*) nie jest obsługiwana. Użytkownik musi jawnie wyrazić fokus, aby uzyskać taką interpretację.

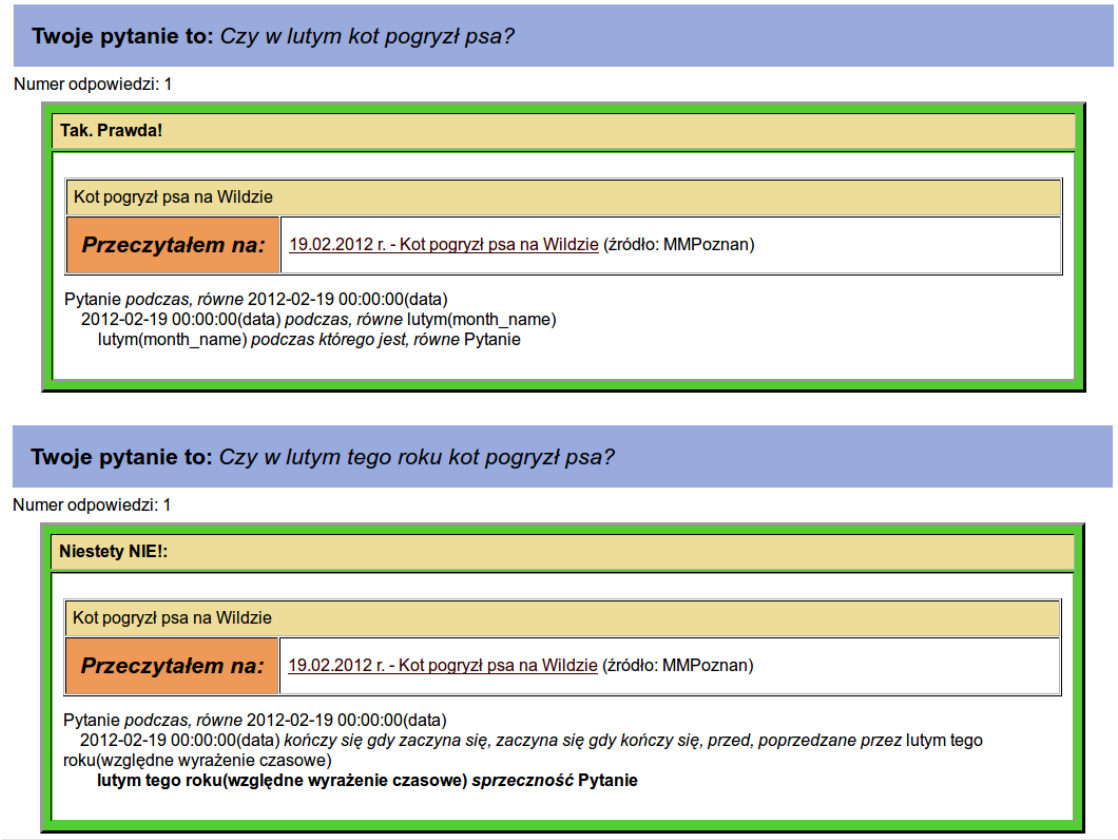
Na rysunku 6.3 przedstawiono odpowiedzi systemu Hipisek na pytanie: *Czy w lutym kot pogryzł psa?* oraz pytanie z jawnie wyrażonym fokusem: *Czy w lutym tego roku kot pogryzł psa?* System zwrócił różne odpowiedzi, ze względu na różną interpretację fokusu pytania.

6.1.3. Wydobywanie relacji czasowych i przestrzennych

Wydobywanie relacji czasowych i przestrzennych jest odrębnym zagadnieniem badawczym dziedziny przetwarzania języka naturalnego. Zadanie polega na identyfikacji relacji semantycznych zachodzących między jednostkami występującymi w tekście. Przykładem takiej relacji jest np. relacja *być stolicą kraju*. Prace nad wydobywaniem relacji semantycznych z tekstów w języku polskim prowadzone są na przykład przez Michała Marcińczuka [Mar2013].

W systemie Hipisek zadanie wydobywania relacji czasowych i przestrzennych polega na powiązaniu oznaczonych jednostek czasowych i przestrzennych jednym z typów relacji wykorzystywanych w systemie Hipisek (patrz dodatek D) i stworzeniu faktu.[6] Zadanie wydobywania jest wykonywane przez moduł HipiRE (od ang. *Hipisek Relation Extraction*).

[6] Stąd dalej zadanie to nazywane jest wydobywaniem faktów.



Rysunek 6.3. Ilustracja problemu fokusu pytania w systemie Hipisek

Wynikiem działania modułu HipiRE jest zbiór faktów.[7] Pozyskane fakty mogą mieć niezdefiniowany podmiot (podobnie jak warunki wykorzystywane w algorytmie odpowiadania na pytania rozstrzygnięcia opisane w rozdziale 5).

Moduł HipiRE jest oparty o zbiór reguł. Reguły zostały zapisane w formalizmie wzorowanym na formalizmie reguł systemu SPADE [BP2008].

Reguła rozpoznawania składa się z następujących elementów:

- **nazwy reguły**,
- **sekcji match**, która definiuje do jakiego ciągu tokenów dopasuje się reguła,
- **opisu utworzonego faktu**, który składa się, z opisu podmiotu, dopełnienia oraz typu relacji.

Reguły uruchamiane są na stokenizowanym tekście, w którym oznaczono i znormalizowano jednostki czasowe i przestrzenne.

Przykładowa reguła rozpoznawania faktu przestrzennego jest przedstawiona poniżej:

[7] Pojęcie faktu zostało wprowadzone w podrozdziale 4.1.1.

```
Rule: Wisła płynie przez Kraków
Match: <ne=stream>+ <base~płynąć> <used~przez> <ne=place>+
Subject: stream(\1)
Object: place(\4)
PredicateName: overlaps
```

Powyższa reguła dopasuje się do ciągu tokenów składającego się z:

- niepustego ciągu tokenów oznaczonych typem jednostki przestrzennej *rzeka* (w treści reguły oznaczonej terminem angielskim *stream*),
- tokenu będącego wyrazem o formie podstawowej *płynąć*,
- tokenu, którego postać tekstowa jest równa łańcuchowi *przez*,
- niepustego ciągu tokenów oznaczonych typem jednostki przestrzennej *miejsce* (w treści reguły oznaczonej terminem angielskim *place*).

Rozpatrywana reguła dopasuje się na przykład do następującego fragmentu tekstu:[8]

Warta płynie przez Poznań

Dla powyższego tekstu przykładowa reguła utworzy następujący fakt: *(Warta [rzeka], częściowo pokrywa się z, Poznań [miasto])*.

Fakty wydobywane z niezdefiniowanym podmiotem

Niektóre reguły nie definiują konkretnej jednostki jako podmiotu wydobytego faktu. W takim przypadku podmiotem faktu staje się całe zdanie. Podmiot taki traktujemy jako niezdefiniowany. Tak wydobyte fakty nie są wykorzystywane w algorytmie odpowiadania na pytania w postaci kwerendy (patrz podrozdział 5.4), natomiast mogą być wykorzystane w algorytmie odpowiadania na pytania z warunkami (patrz podrozdział 5.5).

Reguły tego typu służą do oznaczenia faktów, których pełne oznaczenie wymagałoby bardziej zaawansowanych metod niż wykorzystywane w systemie Hipisek metody powierzchniowe.

Przykładową regułą należącą do tej grupy reguł jest:

```
Rule: ostatnia szansa (dzień tygodnia)
Match: <normal~(?:w|we)> <ne=day_of_week_name>+
Subject: \sentence
Object: \2
PredicateName: is during strict time
```

[8] W dopasowaniu typów jednostek wykorzystywana jest taksonomia typów jednostek, dlatego typ jednostki *Poznań [miasto]*, jest dopasowany do typu *miejsce* (typ *miasto* jest jednym z potomków typu *miejsce* w taksonomii typów jednostek).

Powyższa reguła dopasuje się do ciągu tokenów, w którym pierwszy ma postać tekstową równą łańcuchowi *w* lub *we*, natomiast pozostałe zostały oznaczone jako typ jednostki *nazwa dnia tygodnia* (`day_of_week_name`).

Załóżmy, że reguła jest uruchomiona na następującym pytaniu, które zadano 16 czerwca 2013 roku:

> Czy w piątek jest zakończenie roku szkolnego?

Dla powyższego tekstu reguła utworzy fakt postaci: *(*, dokładnie w trakcie, 2013–06–21 [data])*.[9] Należy zwrócić uwagę, że jednostka czasowa *piątek* została znormalizowana do wartości daty 21 czerwca 2013 z wykorzystaniem kontekstu dokumentu (patrz podrozdział 6.1.2).

Reguły, w których podmiot jest niezdefiniowany, traktowane są jako reguły *ostatniej szansy* i są uruchomiane na końcu przetwarzania.

6.1.4. Przygotowanie sieci ograniczeń

Krawędzie w sieci ograniczeń (wykorzystywanej w procesie wnioskowania) mogą pochodzić z dwóch źródeł:

- bazy wiedzy ustrukturyzowanej,
- bazy wiedzy nieustrukturyzowanej.

Budowanie sieci ograniczeń w oparciu o bazę wiedzy ustrukturyzowanej zostało opisane w podrozdziałach 4.2 oraz 5.3. W niniejszym podrozdziale opisane zostały mechanizmy umożliwiające wykorzystanie bazy wiedzy nieustrukturyzowanej do budowy sieci ograniczeń.

Wykorzystanie bazy wiedzy nieustrukturyzowanej

Baza wiedzy nieustrukturyzowanej składa się z kolekcji dokumentów tekstowych. Każdemu dokumentowi w bazie wiedzy przyporządkowane są następujące informacje dodatkowe:

- tytuł artykułu,
- słowa kluczowe,
- data opublikowania artykułu,
- źródło artykułu (odnośnik do strony internetowej).

[9] Podobnie jak w zapisie warunków pytania (patrz podrozdział 5.5) niezdefiniowany podmiot oznaczamy gwiazdką.

Treść dokumentów jest zaindeksowana za pomocą wyszukiwarki pełnotekstowej Sphinx[10]. Wyszukanie dokumentu w bazie polega na wysłaniu zapytania, mającego postać typowego zapytania do wyszukiwarki internetowej.[11]

W celu pozyskania faktów do budowy sieci ograniczeń wykonywane są następujące zadania:

- wyszukanie dokumentów związanych z przetwarzanym pytaniem,
- pozyskanie akapitów (fragmentów dokumentów, potencjalnie zawierających fakty związane z przetwarzanym pytaniem),
- uruchomienie mechanizmu HipiRE, na pozyskanych akapitach.

Powyższy proces wykorzystuje mechanizmy wyszukiwania dokumentów i akapitów bazowej wersji systemu Hipisek pierwotnie wykorzystywanych w metodzie odpowiadania *Pythia Answerer* (patrz podrozdział 3.4.2).[12]

Wyszukanie dokumentów

Celem kroku wyszukania dokumentów jest ograniczenie przestrzeni przeszukiwania do dokumentów istotnych w przetwarzaniu danego pytania. W tym celu wykorzystywana jest formalna reprezentacja pytania za pomocą struktury QQuery (patrz podrozdział 3.4.1). Budowany jest szereg zapytań do wyszukiwarki Sphinx wykorzystujących temat pytania, akcję pytania, ograniczenia oraz frazy wyszukujące.

Wynik wyszukiwarki Sphinx jest sortowany zgodnie z oceną. Najwyżej ocenione dokumenty trafiają do dalszego przetwarzania.

Pozyskiwanie akapitów

W systemie Hipisek.pl **akapitem** nazywamy ciągły fragment artykułu składającego się z sekwencji kolejnych zdań artykułu źródłowego. Akapit budowany jest w oparciu o **zdanie bazowe**. Do akapitu trafia zdanie bazowe wraz z ustaloną liczbą zdań poprzedzających i następujących po zdaniu bazowym.

Pozyskiwanie akapitów polega na utworzeniu ze zbioru wyszukanych dokumentów kolekcji fragmentów tekstów (akapitów) posortowanej według oceny przydatności w znalezieniu odpowiedzi na przetwarzane pytanie.

W bazowej wersji systemu Hipisek krok pozyskiwania akapitów był utożsamiony z krokiem pozyskiwania odpowiedzi (akapity były wynikiem działania bazowego mechanizmu odpowiadającego *Pythia Answerer*). W obecnej wersji systemu krok

[10] http://sphinxsearch.com/

[11] Szczegółowy opis budowy bazy wiedzy nieustrukturyzowanej znajduje się w pracy [Wal2009].

[12] W niniejszej pracy zawarto tylko krótkie wprowadzenie do metod wyszukiwania dokumentów i akapitów, skupiając się na zagadnieniach związanych bezpośrednio z obsługą pytań rozstrzygnięcia. Dokładny opis metod wyszukiwania dokumentów związanych z przetwarzanym pytaniem oraz pozyskiwania akapitów znajduje się w pracy [Wal2009].

ten jest wydzielony ze względu na wykorzystywanie akapitów w różnych mechanizmach odpowiadających (w szczególności w mechanizmie odpowiadania na pytania rozstrzygnięcia) oraz mechanizmach budowania bazy wiedzy przestrzennej.

Krok pozyskiwania akapitów polega na przyporządkowaniu każdemu zdaniu danego dokumentu oraz jego otoczeniu oceny według następujących metryk:[13]

- ocena według metody frazy wyszukującej — polega na przypisaniu oceny danemu zdaniu, w którego *pobliżu* występuje jedna (lub więcej) fraz wyszukujących.
- ocena według metody wystąpienia tematu — polega na przypisaniu oceny danemu zdaniu, w zależności od występowania w zdaniu lub jego sąsiedztwie tematu pytania, akcji pytania bądź ograniczeń.

W przypadku przekroczenia ustalonej wartości progowej dowolnej z wymienionych metryk zdanie (wraz z ustaloną liczbą zdań sąsiadujących) służy do zbudowania akapitu (stając się zdaniem bazowym). Ocena zdania tworzy ocenę utworzonego akapitu.

Do dalszego przetwarzania trafiają najwyżej ocenione akapity, posortowane według oceny.

Wykorzystanie narzędzia wydobywającego fakty

Każdy z akapitów stanowi odrębne źródło odpowiedzi dla danego pytania. Oznacza to, że akapity mogą dostarczyć różnych odpowiedzi na przetwarzane pytanie. Wszystkie znalezione odpowiedzi są wyświetlane użytkownikowi posortowane według oceny akapitu z którego pochodzą.

Na przykład na pytanie *Czy w Poznaniu kot pogryzł psa?* system Hipisek znajduje dwie różne odpowiedzi (pierwszą pozytywną, drugą negatywną). Pierwsza odpowiedź dotyczy wydarzenia pogryzienia kota na Wildzie (dzielnica Poznania),[14] natomiast druga odpowiedź dotyczy podobnego wydarzenia, które miało miejsce w Lidzbarku Warmińskim.[15]

Dla każdego z akapitu uruchamiany jest moduł *HipiRE* według następującego schematu:

1. Ustaw zdanie przetwarzane jako zdanie budujące akapit,
2. Dla zdania przetwarzanego wykonaj:
   a) Pozyskaj wszystkie fakty ze zdania przetwarzanego,
   b) Dla każdego ze znalezionych faktów f: spróbuj znaleźć odpowiedź na pytanie wykorzystując fakt f.
   c) Jeśli udało się znaleźć odpowiedź, to zakończ przetwarzanie akapitu.

[13] Wymienione metryki zostały dokładnie opisane w pracy [Wal2009] w podrozdziałach 6.2.1. oraz 6.2.2.

[14] Źródło odpowiedzi: http://www.mmpoznan.pl/403569/2012/2/19/kot-pogryzl-psa-na-wildzie

[15] Źródło odpowiedzi: http://www.fakt.pl/Szok-Kot-pogryzl-psa-,artykuly,67256,1.html

3. Jeśli nie znaleziono odpowiedzi na pytanie, to wróć do punktu 2, ustawiając jako zdanie przetwarzane niewykorzystane zdanie poprzedzające/następujące po zdaniu budującym akapit **dopóki** nie sprawdzisz wszystkich zdań akapitu.

Należy podkreślić, że fakty pozyskane przez moduł *HipiRE* wykorzystywane są pojedynczo. Wynika to z następującego założenia:

Uwaga 6.1. W algorytmie wnioskowania zakładamy, że jednocześnie wykorzystywany jest najwyżej **jeden** fakt wydobyty z bazy wiedzy nieustrukturyzowanej.

Powyższe ograniczenie ma na celu minimalizację błędów systemu związanych z niedoskonałością narzędzi przetwarzania nieustrukturyzowanej bazy wiedzy (w szczególności modułu pozyskiwania faktów czasowych i przestrzennych *HipiRE*).

Na przykład załóżmy, że w procesie odpowiadania na pytanie *Czy w Poznaniu kot pogryzł psa?* znaleziono następujący akapit (zdanie budujące akapit zostało pogrubione):[16]

1. *Lidzbarscy policjanci wyjaśniają tragiczną w skutkach bójkę kota z psem.*
2. *Według wstępnych ustaleń śledczych i przesłuchanych świadków wynika, że to kocur zaatakował psa, gdy ten spacerował ze swoim panem.*
3. *Dramat rozegrał się na jednym z osiedli w sobotę w Lidzbarku Warmińskim (Warmia) około godziny 17:00*
4. ***Wtedy oficer dyżurny policji otrzymał zgłoszenie o bójce kota z psem.***
5. *— Na miejscu policjanci ustalili, że na spacerującego z psem mężczyznę miał zaatakować kot, który drapał psa pazurami go i naskoczył mu na kark — relacjonuje kom. Jolanta Wójcik oficer prasowy z Lidzbarka Warmińskiego.*
6. *— Na pomoc psu przyszedł jego pan, który miał odepchnąć kota nogą — dodaje komisarz.*

W pierwszym kroku moduł *HipiRE* pozyskuje fakty ze zdania budującego akapit (o numerze 4). Zdanie to nie zawiera żadnych faktów przestrzennych, dlatego następnie moduł *HipiRE* przetwarza zdanie poprzedzające zdanie budujące (o numerze 3). Ze zdania tego pozyskany jest fakt o niezdefiniowanym podmiocie: *(*, jest położony w, Lidzbark Warmiński [miasto])*. Moduł odpowiadający wykorzystuje pozyskany fakt, do znalezienia odpowiedzi **nie**, na zadane pytanie. Akapit (wraz z siecią ograniczeń utworzoną w procesie wnioskowania) zostaje wyświetlony użytkownikowi jako wyjaśnienie odpowiedzi.

[16] Źródło: http://www.fakt.pl/Szok-Kot-pogryzl-psa-,artykuly,67256,1.html

6.2. Ewaluacja

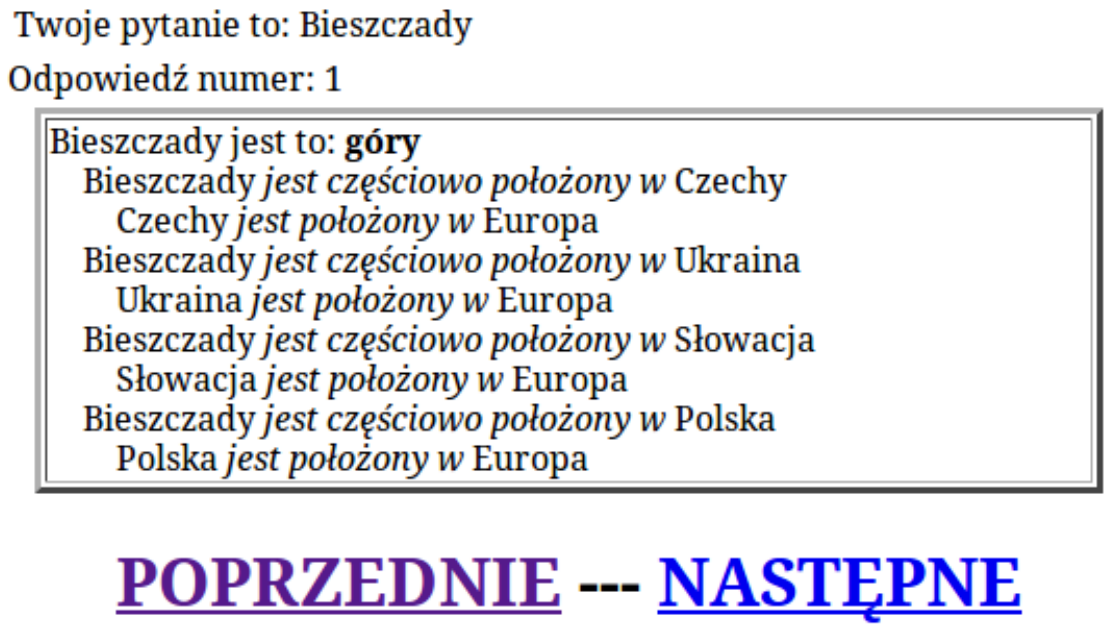
6.2.1. Ewaluacja zebranej bazy wiedzy

Eksperyment ewaluacji zebranej bazy wiedzy polegał na ręcznym sprawdzeniu próbki 100 jednostek. W tym celu dla każdej z wylosowanych jednostek utworzono sieć ograniczeń z wykorzystaniem dwóch wersji baz wiedzy przestrzennej:

- wersji bazowej — powstałej bez użycia algorytmu ujednoznaczniania,
- wersji ostatecznej — powstałej z użyciem algorytmu ujednoznaczniania.

Sieci ograniczeń dla każdej z jednostek z próbki zostały wyświetlone testerowi w obydwu wersjach. Kolejność sieci była losowa. Tester nie wiedział, z której wersji pochodzi dana sieć ograniczeń.

Sieci ograniczeń z obydwu wersji bazy wiedzy były pojedynczo wyświetlane testerowi. Przykładową sieć ograniczeń wyświetlana testerowi przedstawia rysunek 6.4. Zadaniem testera była ocena jakości wyświetlonej sieci ograniczeń w skali od 0 do 10 (im wyższa ocena tym lepiej). Ocenę wykonało trzech testerów. Wyniki testów zostały przedstawione w tabeli 6.1.



Rysunek 6.4. Przykładowa sieć ograniczeń wyświetlana testerowi podczas ewaluacji bazy wiedzy

Eksperyment potwierdził wzrost jakości bazy wiedzy. Średnia ocena testerów dla ostatecznej wersji bazy wiedzy wzrosła o prawie **39%**.

Tabela 6.1. Wyniki ewaluacji zebranej bazy wiedzy przestrzennej

	Wersja bazowa	Wersja ostateczna
Suma ocen	1424	1974
Średnia ocena	4,74	6,58
Zmiana oceny	n/d	**+38,8%**

6.2.2. Ewaluacja działania systemu QA

Ze względu na brak systemu QA o podobnych funkcjonalnościach do systemu Hipisek działającego w języku polskim ewaluacja systemu polegała na porównaniu dwóch jego wersji:

- wersji bazowej — z wyłączonymi algorytmami wnioskowania, wykorzystującej wyłącznie powierzchniowe metody wyszukiwania odpowiedzi opisane w podrozdziale 3.4.2,
- wersji ostatecznej — w włączonymi wszystkimi algorytmami wnioskowania.

Wersje systemu zostały porównane na zebranym korpusie pytań. Korpus ten został udostępniony na stronie internetowej projektu[17] oraz stronie internetowej Pracowni Systemów Informacyjnych Wydziału Matematyki i Informatyki na Uniwersytecie im. Adama Mickiewicza w Poznaniu.[18]

Korpus pytań

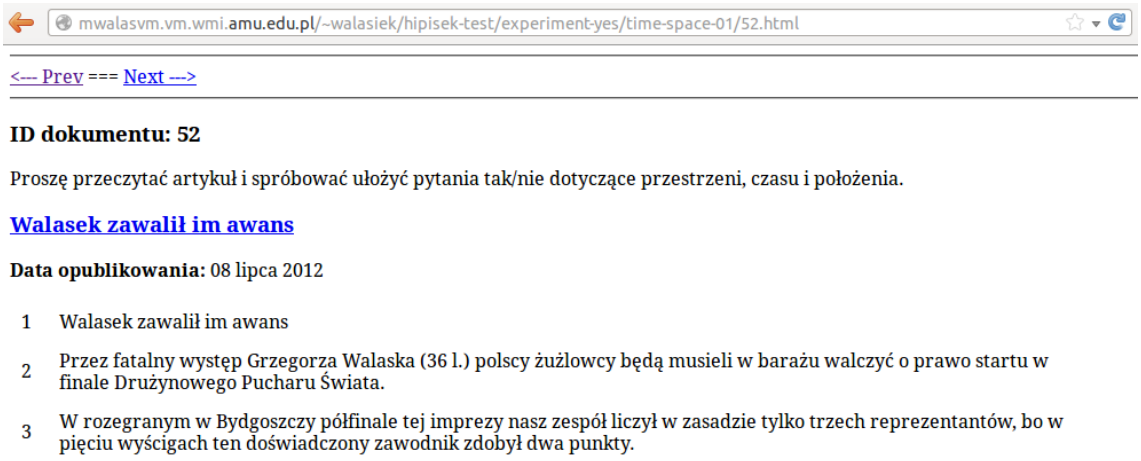
Eksperyment ewaluacji systemu Hipisek jest wzorowany na ewaluacji systemów QA podczas konferencji TREC (patrz rozdział 3). W ramach tej konferencji udostępniano zestaw dokumentów tekstowych, w których należało szukać odpowiedzi na pytania wchodzące w skład zbioru ewaluacyjnego (odpowiedź na każde pytanie musiała znaleźć się w jednym z dokumentów z dostarczonego zestawu).

Aby stworzyć podobny zasób w języku polskim dla kolekcji dokumentów z dziedziny tekstów obsługiwanych przez system Hipisek (tzn. wiadomości internetowych) wykorzystano pracę testerów.

Każdy z testerów otrzymał losowy zbiór dokumentów zaindeksowany przez system Hipisek. Wyświetlony dokument składał się z następujących elementów:

- tytułu dokumentu,
- daty opublikowania dokumentu,
- źródła dokumentu (odnośnik do strony internetowej),
- numerowanych kolejnych zdań treści dokumentu.

[17] http://www.hipisek.pl
[18] http://psi.amu.edu.pl



Rysunek 6.5. Fragment dokumentu wyświetlanego testerowi podczas eksperymentu zbierania korpusu pytań

Fragment przykładowego dokumentu wyświetlonego testerowi został przedstawiony na rysunku 6.5.

Zadaniem testera było zadanie pytań rozstrzygnięcia, w których występuje aspekt czasowy lub przestrzenny, do wyświetlonego dokumentu. Testerzy mieli przyjąć datę opublikowania dokumentu jako czas zadania pytania. Oprócz zadanego pytania, testerzy mieli za zadanie zdefiniować:

- oczekiwaną odpowiedź na pytanie — jedną z trzech wartości: tak, nie, nie wiadomo,[19]
- identyfikator dokumentu, którego dotyczy pytanie,
- numer lub numery zdań, z którego wynika odpowiedź.[20]

Dostęp do eksperymentu odbywał się przez stronę internetową **test.hipisek.pl**, na której w skrócie wyjaśniono zasady i cel eksperymentu.

Wynikiem pracy czterech testerów przetworzono ponad 370 dokumentów i opracowano zestaw pytań składający się z ponad 2100 pytań. Korpus ten został podzielony na trzy części:

- trenującą — służącą do usprawnienia mechanizmów odpowiadających (np. pisanie reguł oznaczania, normalizacji i wydobywania),
- strojącą — służącą do dostrojenia elementów systemu Hipisek (np. dobranie parametrów, sprawdzanie zestawów reguł),
- oceniającą — służącą do oceny systemu.

Statystyki zebranego korpusu pytań zostały zebrane w tabeli 6.2.

[19] W praktyce ostatnia z wartości nie była używana.

[20] W wersji systemu opisywanej w niniejszej pracy nie wykorzystano tego przyporządkowania. Dane te zostały jednak zebrane ze względu na plany wykorzystania metod uczenia maszynowego.

Tabela 6.2. Statystyki zebranego korpusu pytań

Liczba pytań w korpusie	2104
— zbiór trenujący	210
— zbiór strojący	210
— zbiór oceniający	1684
Ogółem pytań z odpowiedzią tak	1185
Ogółem pytań z odpowiedzią nie	919
Ogółem liczba wykorzystanych dokumentów	372

Analiza wybranych pytań prowadzi do interesujących wniosków na temat korpusu.

Po pierwsze znaczna część pytań w korpusie ma sztuczną strukturę oraz wymaga dość zaawansowanej głębokiej analizy tekstu (w dodatku niekoniecznie związanej z wnioskowaniem czasowym i przestrzennym). Przykładem ilustrującym zjawisko tego typu jest pytanie: *Czy na przejeździe kolejowym koło Pobiedzisk zginęło dwóch mężczyzn i pies?* Pytanie to zostało zadane do dokumentu, w którym wystąpił następujący fragment tekstu:

> POBIEDZISKA: Tragiczny wypadek — zginęły 2 kobiety i dziecko. Do tragicznego wypadku na przejeździe kolejowym koło Pobiedzisk, w miejscowości Falkowo, doszło w sobotę po godzinie 15. Na miejscu zginęły dwie kobiety i dziecko.

Innym zaobserwowanym zjawiskiem jest używanie przez testerów pewnych *potocznych* lub *idiomatycznych* wyrażeń przestrzennych. Zjawisko to ilustruje przykład pytania: *Czy Doda postanowiła polecieć* ***do ciepłych krajów****?* Wyrażenie *do ciepłych krajów* nie jest obsługiwane przez system Hipisek, choć jest wyrażeniem przestrzennym. Obsługa tego typu wyrażeń znacznie wykracza poza prace wykonane w ramach niniejszej rozprawy.

Podobnym zjawiskiem jest używanie przez testerów dość *nietypowych* wyrażeń czasowych i przestrzennych. Przykładem ilustrującym ten problem jest pytanie: *Czy Christina wróciła do formy sprzed ciąży?* Intencją testera było zapewne zawarcie aspektu czasowego we frazie *sprzed ciąży*. Obsługa tego typu zjawisk językowych jest (podobnie jak w przypadku frazy *do ciepłych krajów*) poza zakresem niniejszej rozprawy.

Najważniejszym problemem jest częste dopasowanie przez testerów pytań do treści dokumentu. Problem ten polega na kopiowaniu przez testerów fragmentów tekstu z dokumentu i włączaniu go do tworzonych pytań. Przykładem pytania ilustrującego ten problem jest jedno z najdłuższych pytań z zebranego korpusu: *Czy funkcjonariu-*

sze z Centralnego Biura Śledczego i Izby Celnej w Poznaniu w 2012 roku przechwycili nielegalny tytoń? Pytanie to zostało zadane do następującego fragmentu dokumentu:

> Nielegalny transport tytoniu przechwycili funkcjonariusze z Centralnego Biura Śledczego i Izby Celnej w Poznaniu. Wartość przechwyconego tytoniu oszacowano na około 800 tysięcy złotych.

Niniejszy przykład pokazuje, że pytanie zostało utworzone poprzez skopiowanie fragmentu tekstu i dodaniu prostego odnośnika czasowego za pomocą frazy *w 2012 roku*. Taki sposób konstrukcji pytań premiuje powierzchniowe mechanizmy odpowiadania (które bazują na pokryciu słów z pytania słowami z tekstu źródłowego), co może powodować fałszowanie wyników ewaluacji.[21]

Korpus został opublikowany:

- na stronie Pracowni Systemów Informacyjnych: **http://psi.amu.edu.pl/pl/index.php?title=Do_pobrania**
- na stronie projektu Hipisek.pl: **http://www.hipisek.pl** (zakładka *„zasoby”*).

Opis eksperymentu ewaluacji

Eksperyment ewaluacji polegał na uruchomieniu testowanych wersji systemu na zbiorze oceniającym korpusu pytań. System, oprócz pytania, otrzymywał informację o dokumencie, z którego miał zaczerpnąć odpowiedź (miało to na celu wyeliminowanie problemu kontekstowości pytań zbioru testowego). Odpowiedź zwracana przez system była porównywana z odpowiedzią zdefiniowaną przez testera.

Precyzja i pokrycie zostały obliczone w następujący sposób:

$$\text{precyzja} = \frac{\text{liczba poprawnych odpowiedzi}}{\text{liczba wszystkich pytań na które udzielono odpowiedzi}}$$

$$\text{pokrycie} = \frac{\text{liczba poprawnych odpowiedzi}}{\text{liczba pytań w korpusie}}$$

Eksperyment ewaluacji został uruchomiony na komputerze z procesorem Intel Core i3–2370M CPU 2.40GHz oraz 8 GB pamięci operacyjnej RAM. Wyniki eksperymentu przedstawione zostały w tabeli 6.3.

[21] Powyższy problem może zostać zminimalizowany poprzez opracowanie zbioru par dokumentów. Każda para dotyczyć powinna tego samego wydarzenia. Tester układałby pytania korzystając z pierwszego dokumentu, natomiast system odpowiadał korzystając z drugiego dokumentu z pary. Powyższy pomysł został zasugerowany przez dra Filipa Gralińskiego, jednakże ze względu na brak odpowiedniego materiału nie został zrealizowany.

Tabela 6.3. Wyniki ewaluacji systemu Hipisek

	Wersja bazowa	Wersja ostateczna
# pytań w korpusie	1684	1684
# poprawnych odpowiedzi	727	838
# pytań na które udzielono odpowiedzi	1101	1081
precyzja	0,66	0,78
pokrycie	0,43	0,50
f-score	0,52	0,61
średni czas przetwarzania pytania (sek.)	0,27	0,88
całkowity czas przetwarzania pytań (sek.)	449	1482

Dzięki zastosowaniu algorytmów wnioskowania udało się uzyskać znaczny wzrost oceny *f-score*. Wartość *f-score* wzrosła o 0,09 punktu (wzrost o 17%). Wzrost ten uzyskano dzięki znacznej poprawie jakości znalezionych odpowiedzi (w wersji ostatecznej liczba pytań, na które udzielono poprawnej odpowiedzi wzrosła o 111).

Zastosowanie algorytmów wnioskowania skutkuje jednak znacznym spowolnieniem działania systemu. W wersji bazowej przetworzenie jednego pytania zajmowało średnio 0,27 sekundy, natomiast w wersji ostatecznej wartość ta wzrosła prawie trzykrotnie. Przyczynami wzrostu są:

- wykorzystanie dodatkowych narzędzi przetwarzania języka naturalnego (np. modułów HipiNEN i HipiRE),
- konieczność przeszukiwania bazy wiedzy przestrzennej (która nie jest wykorzystywana w wersji bazowej),
- złożoność algorytmów wnioskowania.

Rozdział 7

Podsumowanie

7.1. Realizacja zadań pracy doktorskiej

Pierwsze zadanie pracy doktorskiej polegało na opracowaniu algorytmu zbierania wiedzy przestrzennej z różnych źródeł. Głównym problemem w zadaniu była niejednoznaczność nazw obiektów przestrzennych. Problem ten rozwiązano poprzez zastosowanie wnioskowania jakościowego. Wykorzystano rachunki RCC5 i RCC8. Opracowany algorytm wykorzystuje relacje przestrzenne między obiektami zbieranymi przez system (np. położenie w danym kraju, województwie, gminie) w celu ujednoznacznienia pojęć (np. nazwy dwóch miejscowości Poznań, jednej leżącej w Wielkopolsce oraz drugiej znajdującej się w województwie Lubelskim). Ponadto algorytm wykorzystuje wartości metryczne zbierane przez system (np. liczbę ludności, powierzchnię). Zastosowanie wnioskowania jakościowego pozwoliło na porównywanie wartości metrycznych w sposób przybliżony (za pomocą relacji *około*).

Drugie zadanie polegało na opracowaniu algorytmu odpowiadania na pytania rozstrzygnięcia, w których występuje aspekt czasowy i/lub przestrzenny. Wyróżniono dwa typy pytań: pytania typu kwerendy oraz pytania z warunkami. Opracowano dwa algorytmy odpowiadania na pytania: algorytm odpowiadania na pytania typu kwerendy oraz algorytm odpowiadania na pytania z warunkami. Algorytmy wykorzystują wnioskowanie jakościowe. Wiedzę przestrzenną zamodelowano wykorzystując rachunek RCC5, natomiast wiedzę czasową zamodelowano za pomocą algebry Allena. Obydwa opracowane algorytmy wykorzystują metody powierzchniowe (opracowane w ramach wcześniejszych prac nad systemem Hipisek.pl oraz w pracy magisterskiej autora niniejszej rozprawy doktorskiej), które zostały połączone z metodami wykorzystującymi wnioskowanie.

Trzecie zadanie polegało na implementacji i ewaluacji wszystkich opracowanych algorytmów. Opisane w niniejszej rozprawie algorytmy zostały zaimplementowane w systemach HipiSwot (autorskim systemie do zbierania bazy wiedzy) oraz Hipisek.pl (autorskim systemie odpowiadania na pytania). Za pomocą systemu HipiSwot opracowano bazę wiedzy przestrzennej wykorzystywaną w procesie odpowiada-

nia na pytania. Przeprowadzono ewaluację działania algorytmu ujednoznaczniania, ewaluację zebranej bazy wiedzy przestrzennej oraz ewaluację działania algorytmów odpowiadania na pytania. Ewaluacja odpowiadania na pytania została przeprowadzona w oparciu o zebrany korpus pytań.

7.2. Wymierny rezultat pracy

W niniejszym podrozdziale wymienione zostały efekty pracy nad każdym zadaniem niniejszej rozprawy.

7.2.1. Zbieranie wiedzy przestrzennej

- opracowanie algorytmu ujednoznacznia pojęć, który wykorzystuje wnioskowanie jakościowe,
- system do zbierania wiedzy przestrzennej HipiSwot,
- implementacja algorytmu ujednoznacznia pojęć w systemie HipiSwot,
- baza wiedzy przestrzennej zebrana za pomocą systemu HipiSwot.

7.2.2. Algorytmy odpowiadania na pytania

- algorytm wnioskowania wykorzystującego rachunki RCC5, RCC8 oraz algebrę Allena,
- algorytm tworzenia sieci ograniczeń z bazy wiedzy ustrukturyzowanej i nieustrukturyzowanej,
- algorytm odpowiadania na pytania rozstrzygnięcia z aspektem przestrzennym reprezentowanym za pomocą kwerendy,
- algorytm odpowiadania na pytania rozstrzygnięcia z aspektem czasowym lub przestrzennym reprezentowanym za pomocą warunków,
- taksonomia typów jednostek i typów relacji przestrzennych,
- reguły semantyczne i modelowanie typów relacji przestrzennych i czasowych w rachunkach RCC5 oraz algebrze Allena.

7.2.3. Implementacja systemu QA

- implementacja algorytmów odpowiadania w systemie QA Hipisek,
- opracowanie narzędzi do oznaczania jednostek czasowych i przestrzennych za pomocą mechanizmów regułowych i słownikowych dla języka polskiego,
- opracowanie narzędzi do normalizacji jednostek czasowych i przestrzennych za pomocą mechanizmów regułowych dla języka polskiego,
- opracowanie narzędzi do wydobywania relacji czasowych i przestrzennych za pomocą mechanizmów regułowych dla języka polskiego,

- opracowanie polskiego korpusu pytań rozstrzygnięcia.

7.3. Unikalny wkład badań

Unikalnym wkładem badań autora pracy w tematykę systemów QA jest:

- opracowanie autorskiej metody ujednoznaczniania pojęć dla systemu integrującego dane pochodzące z różnych źródeł, z wykorzystaniem wnioskowania jakościowego,
- opracowanie autorskiej metody odpowiadania na pytania rozstrzygnięcia z aspektem czasowym i przestrzennym wykorzystującą wnioskowanie jakościowe,
- opracowanie i udostępnienie prototypu systemu QA działającego dla języka polskiego i wykorzystującego opracowane algorytmy odpowiadania na pytania,
- opracowanie polskiego korpusu pytań rozstrzygnięcia na zdefiniowanym zbiorze dokumentów tekstowych.

7.4. Kierunki rozwoju

7.4.1. Systemy zbierania wiedzy

Algorytm ujednoznaczniania może zostać zaadaptowany do różnego rodzaju systemów zbierania wiedzy, jako jedno z kryterium walidacji zebranej wiedzy. W szczególności, w przypadku zbierania wiedzy w sposób nienadzorowany (np.: za pomocą metod automatycznych, lub z udziałem sieci społecznościowej, której użytkownicy nie są zaufanymi ekspertami), algorytm ujednoznaczniania może służyć do identyfikacji błędnie zebranych danych lub do identyfikacji wandalizmów.

7.4.2. Rozszerzenie na inne klasy pytań

Algorytmy odpowiadania na pytania przedstawione w niniejszej pracy mogą być rozszerzone na inne klasy pytań. Zauważmy, że warunki (zdefiniowane w pytaniach rozstrzygnięcia z warunkami) mogą wystąpić w dowolnym pytaniu. Dla przykładu rozpatrzmy następujące pytania o czas:

- *Gdzie były mecze Euro 2012?*
- *Gdzie* ***w Wielkopolsce*** *były mecze Euro 2012?*

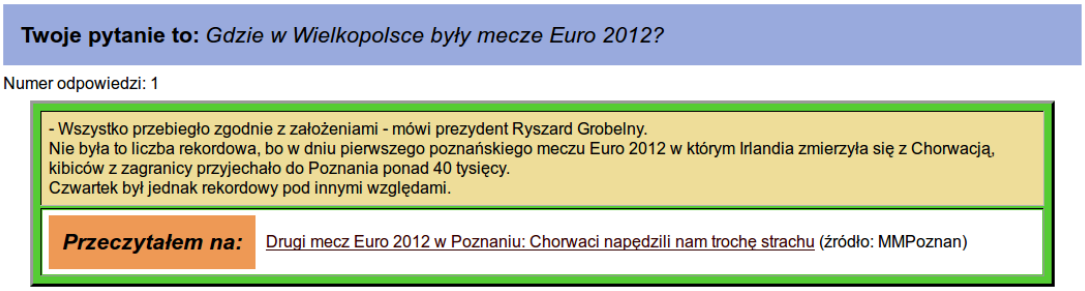
Pierwsze z wymienionych pytań jest zwykłym pytaniem o miejsce. Drugie pytanie jest pytaniem o miejsce z **warunkiem przestrzennym** ograniczającym fokus pytania do obszaru Wielkopolski. Zaprezentowane w niniejszej rozprawie algorytmy odpowiadania mogą zostać zaadoptowane do obsługi pytań z warunkami występującymi w dowolnych klasach pytań. Szkic algorytmu polega na usuwaniu odpowiedzi,

które nie spełniają warunków zawartych w pytaniu (podobnie jak w pytaniach rozstrzygnięcia).

W momencie pisania niniejszej rozprawy trwają prace nad implementacją wymienionego rozwiązania. Rysunek 7.1 przedstawia odpowiedzi systemu Hipisek.pl na pytanie *Gdzie były mecze Euro 2012?* natomiast rysunek 7.2 przedstawia odpowiedź systemu Hipisek.pl na pytanie *Gdzie w Wielkopolsce były mecze Euro 2012?* W drugim przypadku system uwzględnił zawężony fokus pytania (do terenu Wielkopolski), dlatego usunął odpowiedzi, które nie spełniają warunku (tzn. odpowiedzi związane z meczami na terenie Wrocławia, Gdańska i Warszawy).

Wstępne wyniki prac wskazują, że opracowane algorytmy mogą zostać rozszerzone na inne klasy pytań. Zaproponowane w pracy algorytmy mogą zostać zaadaptowane do innych systemów QA lub systemów dialogowych, w których istotną rolę odgrywa aspekt czasowy lub przestrzenny (i które nie ograniczają się do pytań rozstrzygnięcia).

Rysunek 7.1. Odpowiedź na pytanie *Gdzie były mecze Euro 2012?*



Rysunek 7.2. Odpowiedź na pytanie *Gdzie w Wielkopolsce były mecze Euro 2012?*

Dodatek A

Tablice złożeń rachunków użytych w pracy

A.1. Algebra Allena

W rozprawie podano tablicę złożeń relacji algebry Allena za książką [Lig2012]. W tablicy złożeń użyto następujących skrótów:

- $[P, D] = \{P, M, O, S, D\}$
- $[D, PI] = \{D, F, OI, MI, PI\}$
- $[P, DI] = \{P, M, O, FI, DI\}$
- $[DI, PI] = \{DI, SI, OI, MI, PI\}$
- $[O, OI] = \{O, S, D, FI, EQ, F, DI, SI, OI\}$
- $[O, DI] = \{O, FI, DI\}$
- $[DI, OI] = \{DI, SI, OI\}$
- $[O, D] = \{O, S, D\}$
- $[P, O] = \{P, M, O\}$
- $[D, OI] = \{D, F, OI\}$
- $[OI, PI] = \{OI, MI, PI\}$
- $[FI, F] = \{FI, EQ, F\}$
- $[S, SI] = \{S, EQ, SI\}$

Tablica złożeń została przedstawiona w tabeli A.1.

A.2. RCC8

W rozprawie podano tablicę złożeń relacji rachunku RCC8 za książką [Ren2002]. W tablicy złożeń użyto następujących skrótów:

- $[DC, NTPP] = \{DC, EC, PO, TPP, NTPP\}$
- $[DC, NTPPI] = \{DC, EC, PO, TPPI, NTPPI\}$
- $[DC, EQ] = \{DC, EC, PO, TPP, TPPI, EQ\}$

Tabela A.1. Tablica złożeń algebry Allena

$\circ$	EQ	P	PI	D	DI	O	OI	M	MI	S	SI	F	FI
EQ	EQ	P	PI	D	DI	O	OI	M	MI	S	SI	F	FI
P	P	P	$\top$	$[P,D]$	P	P	$[P,D]$	P	$[P,D]$	P	P	$[P,D]$	P
PI	PI	$\top$	PI	$[D,PI]$	PI	$[D,PI]$	PI	$[D,PI]$	PI	$[D,PI]$	PI	PI	PI
D	D	P	PI	D	$\top$	$[P,D]$	$[D,PI]$	P	PI	D	$[D,PI]$	D	$[P,D]$
DI	DI	$[P,DI]$	$[DI,SI]$	$[O,OI]$	DI	$[O,DI]$	$[DI,OI]$	$[O,DI]$	$[DI,OI]$	$[O,DI]$	DI	$[DI,OI]$	DI
O	O	P	$[DI,SI]$	$[O,D]$	$[P,DI]$	$[P,O]$	$[O,OI]$	P	$[DI,OI]$	O	$[O,DI]$	$[O,D]$	$[P,O]$
OI	OI	$[P,DI]$	PI	$[D,OI]$	$[DI,SI]$	$[O,OI]$	$[OI,PI]$	$[O,DI]$	PI	$[D,OI]$	$[OI,PI]$	OI	$[DI,OI]$
M	M	P	$[DI,SI]$	$[O,D]$	P	P	$[O,D]$	P	$[FI,F]$	M	M	$[O,D]$	P
MI	MI	$[P,DI]$	PI	$[D,OI]$	PI	$[D,OI]$	PI	$[S,SI]$	PI	$[D,OI]$	PI	MI	MI
S	S	P	PI	D	$[P,DI]$	$[P,O]$	$[D,OI]$	P	MI	S	$[S,SI]$	D	$[P,O]$
SI	SI	$[P,DI]$	PI	$[D,OI]$	DI	$[O,DI]$	OI	$[O,DI]$	MI	$[S,SI]$	SI	OI	DI
F	F	P	PI	D	$[DI,SI]$	$[O,D]$	$[OI,PI]$	M	PI	D	$[OI,PI]$	F	$[FI,F]$
FI	FI	P	$[DI,SI]$	$[O,D]$	DI	O	$[DI,OI]$	M	$[DI,OI]$	O	DI	$[FI,F]$	FI

- $[EC, NTPP] = \{EC, PO, TPP, NTPP\}$
- $[EC, NTPPI] = \{EC, PO, TPPI, NTPPI\}$
- $[PO, NTPP] = \{PO, TPP, NTPP\}$
- $[PO, NTPPI] = \{PO, TPPI, NTPPI\}$
- $[DC, EC] = \{DC, EC\}$
- $[TPP, NTPP] = \{TPP, NTPP\}$
- $[TPPI, NTPPI] = \{TPPI, NTPPI\}$
- $[PO, EQ] = \{PO, EQ, TPP, TPPI\}$

Tablica złożeń została przedstawiona w tabeli A.3.

A.3. RCC5

W rozprawie podano tablicę złożeń relacji rachunku RCC5 za książką [Ren2002]. Tablica złożeń została przedstawiona w tabeli A.2.

Tabela A.2. Tablica złożeń rachunku RCC5

$\circ$	DR	PO	PP	PPI	EQ
DR	$\top$	$\{DR, PO, PP\}$	$\{DR, PO, PP\}$	DR	DR
PO	$\{DR, PO, PPI\}$	$\top$	$\{PO, PP\}$	$\{DR, PO, PPI\}$	PO
PP	DR	$\{DR, PO, PP\}$	PP	$\top$	PP
PPI	$\{DR, PO, PPI\}$	$\{PO, PPI\}$	$\{PO, PP, PPI, EQ\}$	PPI	PPI
EQ	DR	PO	PP	PPI	EQ

Tabela A.3. Tablica złożeń rachunku RCC8

$\circ$	DC	EC	PO	TPP	NTPP	TPPI	NTPPI	EQ
DC	$\top$	$[DC, NTPP]$	$[DC, NTPP]$	$[DC, NTPP]$	$[DC, NTPP]$	DC	DC	DC
EC	$[DC, NTPPI]$	$[DC, EQ]$	$[DC, NTPP]$	$[EC, NTPP]$	$[PO, NTPP]$	$[DC, EC]$	DC	EC
PO	$[DC, NTPPI]$	$[DC, NTPPI]$	$\top$	$[PO, NTPP]$	$[PO, NTPP]$	$[DC, NTPPI]$	$[DC, NTPPI]$	PO
TPP	DC	$[DC, EC]$	$[DC, NTPP]$	$[TPP, NTPP]$	NTPP	$[DC, EQ]$	$[DC, NTPPI]$	TPP
NTPP	DC	DC	$[DC, NTPP]$	NTPP	NTPP	$[DC, NTPP]$	$\top$	NTPP
TPPI	$[DC, NTPPI]$	$[EC, NTPPI]$	$[PO, NTPPI]$	$[PO, EQ]$	$[PO, NTPP]$	$[TPPI, NTPPI]$	NTPPI	TPPI
NTPPI	$[DC, NTPPI]$	$[PO, NTPPI]$	$[PO, NTPPI]$	$[PO, NTPPI]$	$\top$	NTPPI	NTPPI	NTPPI
EQ	DC	EC	PO	TPP	NTPP	TPPI	NTPPI	EQ

Dodatek B

Podklasy podatne rachunków RCC8 i RCC5

B.1. RCC8: Podklasa $\hat{H}_8$

$\bot$	{EQ}	{NTPPI}
{TPPI}	{TPPI, EQ}	{TPPI, NTPPI}
{TPPI, NTPPI, EQ}	{NTPP}	{TPP}
{TPP, EQ}	{TPP, NTPP}	{TPP, NTPP, EQ}
{PO}	{PO, EQ}	{PO, NTPPI}
{PO, TPPI}	{PO, TPPI, EQ}	{PO, TPPI, NTPPI}
{PO, TPPI, NTPPI, EQ}	{PO, NTPP}	{PO, NTPP, NTPPI}
{PO, NTPP, TPPI}	{PO, NTPP, TPPI, NTPPI}	{PO, TPP}
{PO, TPP, EQ}	{PO, TPP, NTPPI}	{PO, TPP, TPPI}
{PO, TPP, TPPI, EQ}	{PO, TPP, TPPI, NTPPI}	{PO, TPP, TPPI, NTPPI, EQ}
{PO, TPP, NTPP}	{PO, TPP, NTPP, EQ}	{PO, TPP, NTPP, NTPPI}
{PO, TPP, NTPP, TPPI}	{PO, TPP, NTPP, TPPI, EQ}	{PO, TPP, NTPP, TPPI, NTPPI}
{PO, TPP, NTPP, TPPI, NTPPI, EQ}	{EC}	{EC, EQ}
{EC, NTPPI}	{EC, TPPI}	{EC, TPPI, EQ}
{EC, TPPI, NTPPI}	{EC, TPPI, NTPPI, EQ}	{EC, NTPP}
{EC, TPP}	{EC, TPP, EQ}	{EC, TPP, NTPP}
{EC, TPP, NTPP, EQ}	{EC, PO}	{EC, PO, EQ}
{EC, PO, NTPPI}	{EC, PO, TPPI}	{EC, PO, TPPI, EQ}
{EC, PO, TPPI, NTPPI}	{EC, PO, TPPI, NTPPI, EQ}	{EC, PO, NTPP}
{EC, PO, NTPP, NTPPI}	{EC, PO, NTPP, TPPI}	{EC, PO, NTPP, TPPI, NTPPI}
{EC, PO, TPP}	{EC, PO, TPP, EQ}	{EC, PO, TPP, NTPPI}
{EC, PO, TPP, TPPI}	{EC, PO, TPP, TPPI, EQ}	{EC, PO, TPP, TPPI, NTPPI}

{EC, PO, TPP, TPPI, NTPPI, EQ}	{EC, PO, TPP, NTPP}	{EC, PO, TPP, NTPP, EQ}
{EC, PO, TPP, NTPP, NTPPI}	{EC, PO, TPP, NTPP, TPPI}	{EC, PO, TPP, NTPP, TPPI, EQ}
{EC, PO, TPP, NTPP, TPPI, NTPPI}	{EC, PO, TPP, NTPP, TPPI, NTPPI, EQ}	{DC}
{DC, EQ}	{DC, NTPPI}	{DC, TPPI}
{DC, TPPI, EQ}	{DC, TPPI, NTPPI}	{DC, TPPI, NTPPI, EQ}
{DC, NTPP}	{DC, TPP}	{DC, TPP, EQ}
{DC, TPP, NTPP}	{DC, TPP, NTPP, EQ}	{DC, PO}
{DC, PO, EQ}	{DC, PO, NTPPI}	{DC, PO, TPPI}
{DC, PO, TPPI, EQ}	{DC, PO, TPPI, NTPPI}	{DC, PO, TPPI, NTPPI, EQ}
{DC, PO, NTPP}	{DC, PO, NTPP, NTPPI}	{DC, PO, NTPP, TPPI}
{DC, PO, NTPP, TPPI, NTPPI}	{DC, PO, TPP}	{DC, PO, TPP, EQ}
{DC, PO, TPP, NTPPI}	{DC, PO, TPP, TPPI}	{DC, PO, TPP, TPPI, EQ}
{DC, PO, TPP, TPPI, NTPPI}	{DC, PO, TPP, TPPI, NTPPI, EQ}	{DC, PO, TPP, NTPP}
{DC, PO, TPP, NTPP, EQ}	{DC, PO, TPP, NTPP, NTPPI}	{DC, PO, TPP, NTPP, TPPI}
{DC, PO, TPP, NTPP, TPPI, EQ}	{DC, PO, TPP, NTPP, TPPI, NTPPI}	{DC, PO, TPP, NTPP, TPPI, NTPPI, EQ}
{DC, EC}	{DC, EC, EQ}	{DC, EC, NTPPI}
{DC, EC, TPPI}	{DC, EC, TPPI, EQ}	{DC, EC, TPPI, NTPPI}
{DC, EC, TPPI, NTPPI, EQ}	{DC, EC, NTPP}	{DC, EC, TPP}
{DC, EC, TPP, EQ}	{DC, EC, TPP, NTPP}	{DC, EC, TPP, NTPP, EQ}
{DC, EC, PO}	{DC, EC, PO, EQ}	{DC, EC, PO, NTPPI}
{DC, EC, PO, TPPI}	{DC, EC, PO, TPPI, EQ}	{DC, EC, PO, TPPI, NTPPI}
{DC, EC, PO, TPPI, NTPPI, EQ}	{DC, EC, PO, NTPP}	{DC, EC, PO, NTPP, NTPPI}
{DC, EC, PO, NTPP, TPPI}	{DC, EC, PO, NTPP, TPPI, NTPPI}	{DC, EC, PO, TPP}
{DC, EC, PO, TPP, EQ}	{DC, EC, PO, TPP, NTPPI}	{DC, EC, PO, TPP, TPPI}
{DC, EC, PO, TPP, TPPI, EQ}	{DC, EC, PO, TPP, TPPI, NTPPI}	{DC, EC, PO, TPP, TPPI, NTPPI, EQ}
{DC, EC, PO, TPP, NTPP}	{DC, EC, PO, TPP, NTPP, EQ}	{DC, EC, PO, TPP, NTPP, NTPPI}

{DC, EC, PO, TPP, NTPP, TPPI}	{DC, EC, PO, TPP, NTPP, TPPI, EQ}	{DC, EC, PO, TPP, NTPP, TPPI, NTPPI}
$\top$		

B.2. RCC5: Podklasa $\hat{H}_5$

Zbiór relacji podklasy $\hat{H}_5$ przedstawiony został za pracą [JD1997].

$\bot$	{DR}	{PO}
{DR, PO}	{PP}	{DR, PP}
{PO, PP}	{DR, PO, PP}	{PPI}
{DR, PPI}	{PO, PPI}	{DR, PO, PPI}
{PO, PP, PPI}	{DR, PO, PP, PPI}	{EQ}
{DR, EQ}	{PO, EQ}	{DR, PO, EQ}
{PP, EQ}	{DR, PP, EQ}	{PO, PP, EQ}
{DR, PO, PP, EQ}	{PPI, EQ}	{DR, PPI, EQ}
{PO, PPI, EQ}	{DR, PO, PPI, EQ}	{PO, PPI, EQ}
$\top$		

Dodatek C

Taksonomia wybranych typów jednostek

W poniższym zestawieniu zebrano najważniejsze typy jednostek przetwarzane w systemie Hipisek. W tabeli pominięto opis szczegółowych typów, pozostawiając tylko opis typów ogólnych.

Taksonomia jednostek przestrzennych została oparta o klasyfikację jednostek serwisu Geonames.[1]

Typ	Opis
entity	**Nadtyp**: — **Typy potomne**: concept, object **Opis**: Nadrzędny typ wszystkich jednostek.
concept	**Nadtyp**: entity **Typy potomne**: first_name, surname, value **Opis**: Pojęcia abstrakcyjne.
first_name	**Nadtyp**: concept **Opis**: Imię osoby. Np.: Marcin, Jan, Stefan, Anna, Renata.
surname	**Nadtyp**: concept **Opis**: Nazwisko osoby. Np.: Kowalski, Nowak, Walas.
value	**Nadtyp**: concept **Typy potomne**: link, number, quantity, temporal **Opis**: Jednostka reprezentująca mierzalną wartość. Np.: liczba (1, 3, 4), wartość metryczna (12 metrów, 100 kg).
link	**Nadtyp**: value **Opis**: Odnośnik do strony internetowej.
temporal	**Nadtyp**: value **Typy potomne**: date, relative_date, temporal_value, time **Opis**: Dowolne wyrażenie czasowe.

[1] http://www.geonames.org/export/codes.html

date	**Nadtyp**: temporal **Opis**: Wyrażenie daty. Np.: 12 maja 2013 roku.
relative_date	**Nadtyp**: temporal **Opis**: Względne wyrażenie czasowe. Np.: w zeszłym roku, w zeszłym miesiącu, pojutrze.
temporal_value	**Nadtyp**: temporal **Typy potomne**: day_of_month_number, day_of_week_name, hour_number, minute_number, month_name, month_number, part_of_day, second_number, year_number **Opis**: Absolutna wartość dowolnej jednostki czasowej. Np.: numer roku, numer miesiąca w roku, numer dnia w miesiącu.
time	**Nadtyp**: temporal **Opis**: Wyrażenie godziny. Np.: 17:00.
object	**Nadtyp**: entity **Typy potomne**: place **Opis**: Obiekty świata rzeczywistego.
place	**Nadtyp**: object **Typy potomne**: administrative_territory, area, forest, mountain_object, populated_place, spot, water_object **Opis**: Dowolna jednostka przestrzenna.
administrative_territory	**Nadtyp**: place **Typy potomne**: administrative_division, country **Opis**: Jednostka przestrzenna wynikająca z podziału administracyjnego świata. Np.: państwo, województwo, gmina.
administrative_division	**Nadtyp**: administrative_territory **Typy potomne**: district, first_order_administrative_division, fourth_order_administrative_division, parish, second_order_administrative_division, third_order_administrative_division **Opis**: Jednostka administracyjna danego państwa np.: województwo, powiat, gmina.
country	**Nadtyp**: administrative_territory **Opis**: Państwo, np.: Polska, USA, Wielka Brytania, Francja.
area	**Nadtyp**: place **Typy potomne**: continent, desert, island **Opis**: Dowolny obszar geograficzny, w szczególności kontynent, wyspa, region. Np.: Antarktyda, Mazowsze, Małopolska.

forest	**Nadtyp**: place **Opis**: Obszar leśny. Np.: Puszcza Kurpiowska, Nadleśnictwo Jegiel, Park Narodowy Imienia Stefana Żeromskiego.
mountain_object	**Nadtyp**: place **Typy potomne**: canyon, mountain, mountains, peak **Opis**: Dowolny obiekt górski. W szczególności szczyt, masyw, przełęcz, dolina. Np.: Przełęcz Tylicka, Góra Świętego Jana, Kotlina Sandomierska, Garb Podlaski.
populated_place	**Nadtyp**: place **Typy potomne**: city, section_of_city, village **Opis**: Dowolny obszar zamieszkały, w tym miasta, wsie i osady. Np.: Warszawa, Poznań, Szczecin, Łapki Małe.
spot	**Nadtyp**: place **Typy potomne**: address, airport, automatic_teller_machine, bridge, building, cave, cementery, gate **Opis**: Obiekt interesujący (ang. *point-of-interest*). Np.: bankomat, zabytek, budynek użyteczności publicznej, restauracja.
building	**Nadtyp**: spot **Typy potomne**: bank, castle, church, museum, palace, school, university **Opis**: Dowolny budynek. Np.: Teatr im. Stanisława Witkiewicza.
water_object	**Nadtyp**: place **Typy potomne**: bank, bay, gulf, lagoon, lake, ocean, pond, sea, spring, stream, swamp, water_canal, water_channel, waterfall **Opis**: Dowolny obiekt wodny. W szczególności rzeka, jezioro, strumyk, ocean, wodospad. Np.: Jezioro Warpno, Wisła, Drawa, Wodospad Kamieńczyk.

Dodatek D

Taksonomia relacji

W poniższym zestawieniu zebrano najważniejsze typy relacji przetwarzane w systemie Hipisek. Nazwy typów podano tak jak występują w implementacji systemu Hipisek, to znaczy w języku angielskim. W wyjaśnieniach typów relacji przyjęto, że jednostka A jest podmiotem faktu, a jednostka B dopełnieniem faktu.

Typ	Opis
relation	**Nadtyp**: — **Typy potomne**: attribute, spatial relation, temporal relation **Opis**: Nadrzędny typ wszystkich relacji.
attribute	**Nadtyp**: relation **Typy potomne**: geoattribute, population **Opis**: Własność, atrybut. Pewna dana związana z jednostką, będącą podmiotem faktu. Dla danej jednostki atrybut danego typu jest zawsze unikatowy, np.: dla jednostki Polska występuje tylko jeden atrybut o typie *population*.
geoattribute	**Nadtyp**: attribute **Typy potomne**: elevation, latitude, length, longitude, surface **Opis**: Atrybut przestrzenny: długość i szerokość geograficzna, długość (np. rzeki), powierzchnia, wysokość nad poziomem morza.
population	**Nadtyp**: attribute **Opis**: Liczba ludności danej jednostki.
spatial relation	**Nadtyp**: relation **Typy potomne**: is located in, is partially located in, overlaps **Opis**: Nadtyp wszystkich relacji przestrzennych.
is located in	**Nadtyp**: spatial relation **Opis**: A jest położone w B. Jednostka A znajduje się w całości w jednostce B, np.: województwo Wielkopolskie **jest położone w** Polsce.

is partially located in	**Nadtyp**: spatial relation **Opis**: A jest częściowo położone w B. Jednostka A znajduje się w części lub w całości w jednostce B, np.: rzeka Odra **jest częściowo położona w** Polsce. Relacja wyraża pewną niedokładność informacji przechowywanej w bazie (nie wiemy czy jednostka A znajduje się w jednostce B w części, czy też w całości).
overlaps	**Nadtyp**: spatial relation **Opis**: A pokrywa B. Jednostka A znajduje się częściowo w jednostce B, np.: Tatry **pokrywają** Polskę (bo tylko część Tatr znajduje się w Polsce).
temporal relation	**Nadtyp**: relation **Typy potomne**: after time, before time, end time, is during time, start time **Opis**: Nadtyp wszystkich relacji czasowych.
after time	**Nadtyp**: temporal relation **Opis**: Jednostka A ma/miała miejsce po jednostce B.
before time	**Nadtyp**: temporal relation **Opis**: Relacja odwrotna do *after time*.
end time	**Nadtyp**: temporal relation **Opis**: Jednostka A kończy się w czasie jednostki B. Na przykład: (praca, end time, 16:00).
is during time	**Nadtyp**: temporal relation **Typy potomne**: is during strict time **Opis**: Jednostka A ma/miała miejsce w czasie jednostki B, przy czym przedział czasowy odpowiadający czasowi trwania jednostki A nie musi być w całości zawarty w przedziale czasowym odpowiadającemu jednostce B (może tylko na niego nachodzić).
is during strict time	**Nadtyp**: is during time **Opis**: Jednostka A ma/miała miejsce w czasie jednostki B, przy czym przedział czasowy odpowiadający czasowi trwania jednostki A jest w całości zawarty w przedziale czasowym odpowiadającemu jednostce B.
start time	**Nadtyp**: temporal relation **Opis**: Jednostka A zaczyna się w czasie jednostki B. Na przykład: (praca, start time, 8:00).

Podziękowania

Na początek pragnę podziękować panu profesorowi Wojciechowi Buszkowskiemu i panu profesorowi Adamowi Przepiórkowskiemu za cenne uwagi, które pozwoliły udoskonalić niniejszą pracę.

Żadna praca we współczesnym świecie nie odbywa się w izolacji. Podczas pracy nad systemem Hipisek.pl oraz pisania niniejszej rozprawy korzystałem z pomocy wielu ludzi, którym chcę w tym miejscu serdecznie podziękować.

- Marcin Dziubek
- Filip Graliński
- Jam Harcerz
- Przemysław Iwanek
- Krzysztof Jassem
- Marcin Junczys-Dowmunt
- Szymon Jóźwiakowski
- Leszek Manicki
- Michał Marcińczuk
- Karolina Pędziwiatr
- Justyna Stachewicz
- Kuba Syroka
- Mariusz Tański
- Kamil Wylegała
- Tomasz Śliwiński

Ponadto dziękuję administratorom Wydziału Matematyki i Informatyki, za pomoc w administracji maszyną wirtualną na której umieszczony był prototyp systemu Hipisek.pl.

Formatując niniejszą rozprawę korzystałem z przykładu autorstwa doktora Marcina Borkowskiego zamieszczonego na stronie internetowej: mbork.pl.

Bibliografia

[AL2007] S. Auer and J. Lehmann, *What have Innsbruck and Leipzig in common? Extracting Semantics from Wiki Content*, ESWC, 2007, pp. 503–517.

[All1983] J. F. Allen, *Maintaining knowledge about temporal*, Intervals CACM **26** (1983), 832–844.

[BCC+2001] J. Burger, C. Cardie, V. Chaudhri, R. Gaizauskas, S. Harabagiu, D. Israel, C. Jacquemin, C.-Y. Lin, S. Maiorano, G. Miller, D. Moldovan, B. Ogden, J. Prager, E. Riloff, A. Singhal, R. Shrihari, T. Strzalkowski, E. Voorhees, and R. Weischedel, *Issues, tasks and program structures to roadmap research in Question & Answering (Q&A)*, NIST, 2001. Nieopublikowany.

[Ben1994] B. Bennett, *Spatial reasoning with propositional logic*, Principles of Knowledge Representation and Reasoning: Proceedings of the 4th International Conference, 1994, pp. 51–62.

[BLK+2009] C. Bizer, J. Lehmann, G. Kobilarov, S. Auer, C. Becker, R. Cyganiak, and S. Hellmann, *DBpedia - A crystallization point for the Web of Data*, Web Semantics **7** (September 2009), no. 3, 154–165.

[BP2008] A. Buczyński and A. Przepiórkowski, ♠ *Demo: An Open Source Tool for Partial Parsing and Morphosyntactic Disambiguation*, Proceedings of the Sixth International Language Resources and Evaluation (LREC'08), 2008, pp. 28–30.

[CBGG1997] A. G. Cohn, B. Bennett, J. Gooday, and N. M. Gotts, *Qualitative Spatial Representation and Reasoning with the Region Connection Calculus*, GeoInformatica **1** (1997), no. 3, 275–316.

[CFH2008] P. Clark, C. Fellbaum, and J. Hobbs, *Using and extending WordNet to support Question-Answering*, Proceedings of the fourth global WordNet conference (GWC'08), 2008.

[Col1971] K. M. Colby, *Artificial paranoia*, Artificial Intelligence, 1971.

[DMP1991] R. Dechter, I. Meiri, and J. Pearl, *Temporal Constraint Networks*, Artif. Intell. **49** (1991), no. 1-3, 61–95.

[Ege1989] M. J. Egenhofer, *A Formal Definition of Binary Topological Relationships*, FODO, 1989, pp. 457–472.

[Ege1991] M. J. Egenhofer, *Reasoning about binary topological relations*, SSD, 1991, pp. 143–160.

[FBCC+2010] D. Ferrucci, E. Brown, J. Chu-Carroll, J. Fan, D. Gondek, A. A. Kalyanpur, A. Lally, J. W. Murdock, E. Nyberg, J. Prager, N. Schlaefer, and C. Welty, *Building Watson: An Overview of the DeepQA Project*, AI Magazine (2010).

[Fel1998] C. Fellbaum (ed.), *WordNet: An Electronic Lexical Database (Language, Speech, and Communication)*, illustrated edition, The MIT Press, 1998.

[FNA+2008] D. Ferrucci, E. Nyberg, J. Allan, K. Barker, E. Brown, J. Chu-Carroll, A. Ciccolo, P. Duboue, J. Fan, D. Gondek, E. Hovy, B. Katz, A. Lally, M. McCord, P. Morarescu, B. Murdock, B. Porter, J. Prager, T. Strzalkowski, C. Welty, and W. Zadrozny, *IBM Research Report. Towards the Open Advancement of Question Answering Systems*, IBM, 2008.

[FR2006] D. Ferrés and H. Rodríguez, *Experiments Adapting an Open-Domain Question Answering System to the Geographical Domain Using Scope-Based Resources* (2006).

[Gab2009] N. Gabrielli, *Investigation of the Tradeoff between Expressiveness and Complexity in Description Logics with Spatial Operators*, Universita degli Studi di Verona. Dipartimento di Informatica, 2009. Rozprawa doktorska.

[GE2001] R. K. Goyal and M. J. Egenhofer, *Cardinal direction between extended spatial objects*, IEEE Transactions on Knowledge and Data Engineering, 2001.

[GJJD2012] F. Graliński, K. Jassem, and M. Junczys-Dowmunt, *PSI-Toolkit: Natural Language Processing Pipeline*, Computational Linguistics — Applications (2012).

[GP1994] A. C. Graesser and N. K. Person, *Question Asking during Tutoring*, American Educational Research Journal **31** (1994), no. 1, 104–137.

[GW2009] F. Graliński and M. Walas, *Looking for new words out there*, Proceedings of the International Multiconference on Computer Science and Information Technology, 2009October, pp. 213–218.

[GWCL1961] B. F. Green., A. K. Wolf, C. Chomsky, and K. Laughery, *Baseball: an automatic question answerer*, Proceedings of the western joint computer conference, 1961, pp. 219–224.

[HMC+2005] S. M. Harabagiu, D. I. Moldovan, C. Clark, M. Bowden, A. Hickl, and P. Wang, *Employing Two Question Answering Systems in TREC 2005*, TREC, 2005.

[HSJP2003] S. M. Harabagiu, M. Steven J, and M. A. Pasca, *Open-domain textual question answering techniques*, Natural Language Engineering (2003), 1–38.

[Jas2012] K. Jassem, *PSI-Toolkit — how to turn a linguist into a computational linguist*, Proceedings of 15th International Conference on Text, Speech and Dialogue, 2012.

[JD1997] P. Jonsson and T. Drakengren, *A Complete Classification of Tractability in RCC-5*, Journal of Artificial Intelligence Research (1997), 211–221.

[Leh1977] W. G. Lehnert, *The Process of Question Answering*, Yale, 1977.

[LGH+2000] E. H. Laurie, L. Gerber, U. Hermjakob, M. Junk, and C. yew Lin, *Question Answering in Webclopedia*, In Proceedings of the Ninth Text REtrieval Conference (TREC-9), 2000, pp. 655–664.

[Lig2012] G. Ligozat, *Qualitative spatial and temporal reasoning*, ISTE Ltd. and John Wiley and Sons Ltd., 2012.

[LVO2009] G. Ligozat, Z. Vetulani, and J. Osiński, *Spatio-temporal aspects of the monitoring of complex events*, Proceedings of the 21st International Joint Conference on Artificial Intelligence: Workshop on Spatial and Temporal Reasoning, 2009.

[Mac2009] B. MacCartney, *Natural Language Inference*, Stanford University, 2009. Rozprawa doktorska.

[Man2009] L. Manicki, *Płytki parsing języka francuskiego*, Uniwersytet im. Adama Mickiewicza, 2009. Praca magisterska.

[Mar2013] M. Marcińczuk, *Ekstrakcja informacji o relacjach semantycznych między jednostkami identyfikacyjnymi z dokumentów tekstowych*, Politechnika Wrocławska, 2013. Rozprawa doktorska.

[May2003] M. T. Maybury (ed.), *New Directions in Question Answering, Papers from 2003*

AAAI Spring Symposium, Stanford University, Stanford, CA, USA, AAAI Press, 2003.

[MCH2005] D. Moldovan, C. Clark, and S. Harabagiu, *Temporal context representation and reasoning*, Proceedings of the International Joint Conferenceon Artificial Intelligence (IJCAI), 2005.

[MDH+2010] I. Mani, C. Doran, D. Harris, J. Hitzeman, R. Quimby, J. Richer, B. Wellner, S. A. Mardis, and S. Clancy, *SpatialML: annotation scheme, resources, and evaluation*, Language Resources and Evaluation **44** (2010), no. 3, 263–280.

[MRS2008] C. D. Manning, P. Raghavan, and H. Schütze, *Introduction to Information Retrieval*, Cambridge University Press, 2008.

[NP2001] I. Niles and A. Pease, *Towards a standard upper ontology*, Proceedings of the international conference on Formal Ontology in Information Systems - Volume 2001, 2001, pp. 2–9.

[PCI+2003] J. Pustejovsky, J. Castaño, R. Ingria, R. Saurí, R. Gaizauskas, A. Setzer, and G. Katz, *Timeml: Robust specification of event and temporal expressions in text*, Fifth International Workshop on Computational Semantics (IWCS-5), 2003.

[PLBR2010] J. Pustejovsky, K. Lee, H. Bunt, and L. Romary, *ISO-TimeML: An International Standard for Semantic Annotation*, Proceedings of the Seventh International Conference on Language Resources and Evaluation (LREC'10), 2010.

[Pom2005] J. Pomerantz, *A linguistic analysis of question taxonomies*, Journal of the American Society for Information Science and Technology **56(7)** (2005), 715–728.

[PSB2009] M. Piasecki, S. Szpakowicz, and B. Broda, *A Wordnet from the ground up*, Oficyna Wydawnicza Politechniki Wrocławskiej, 2009.

[RCC1992] D. A. Randell, Z. Cui, and A. G. Cohn, *A Spatial Logic based on Regions and Connection*, Proceedings 3rd International Conference On Knowledge Representation and Reasoning, 1992.

[Rei1977] R. Reiter, *On Closed World Data Bases*, Logic and Data Bases, 1977, pp. 55–76.

[Ren2002] J. Renz, *Qualitative spatial reasoning with topological information*, Springer-Verlag, Berlin, Heidelberg, 2002.

[RN1997] J. Renz and B. Nebel, *On the Complexity of Qualitative Spatial Reasoning: A Maximal Tractable Fragment of the Region Connection Calculus*, IJCAI (1), 1997, pp. 522–527.

[SH2007] T. Strzalkowski and S. Harabagiu, *Advances in Open Domain Question Answering*, 1st ed., Springer Publishing Company, Incorporated, 2007.

[UCHS2002] T. E. Uribe, V. Chaudhri, P. J. Hayes, and M. E. Stickel, *Qualitative Spatial Reasoning for Question-Answering: Axiom Reuse and Algebraic Methods*, AAAI Spring Symposium on Mining Answers from Texts and Knowledge Bases, 2002march.

[VH1966] J. F. Vallee and J. A. Hynek, *An Automatic Question-Answering System for Stellar Astronomy*, Publications of the Astronomical Society of the Pacific **78** (1966).

[VKvB1990] M. Vilain, H. Kautz, and P. van Beek, *Readings in qualitative reasoning about physical systems*, 1990, pp. 373–381.

[VMO+2010] Z. Vetulani, J. Marciniak, J. Obrebski, G. Vetulani, A. Dabrowski, M. Kubis, J. Osiński, J. Walkowska, P. Kubacki, and K. Witalewski, *Language resources and text processing technologies. The POLINT-112-SMS system as example of application of Human Language Technology in the public security area.*, Adam Mickiewicz University Press, 2010.

[VT2000] E. M. Voorhees and D. M. Tice, *Building a Question Answering test collection*,

Proceedings of the 23rd Annual International ACM SIGIR Conference on Research and Development in Information Retrieval, 2000, pp. 200–207.

[VWO+2009] Z. Vetulani, J. Walkowska, T. Obrębski, J. Marciniak, P. Konieczka, and P. Rzepecki, *An Algorithm for Building Lexical Semantic Network and Its Application to PolNet — Polish WordNet Project*, Human Language Technology. Challenges of the Information Society, 2009, pp. 369–381.

[Wal2009] M. Walas, *Pozyskiwanie wiedzy z bazy artykułów/tekstów za pomocą aplikacji typu wyszukiwarki internetowej odpowiadającej na pytania o czas i miejsce*, Uniwersytet im. Adama Mickiewicza, 2009. Praca magisterska.

[Wal2010] M. Walas, *Classification of translation pairs for the purpose of creating domain dictionaries*, Speech and Language Technology, 2010, pp. 105–113.

[Wal2012] M. Walas, *How to answer yes/no spatial questions using qualitative reasoning?*, Computational Linguistics and Intelligent Text Processing. Lecture Notes in Computer Science (LNCS), 2012, pp. 330–341.

[WJ2010] M. Walas and K. Jassem, *Named Entity Recognition in a Polish Question Answering System*, Intelligent Information Systems. New Approaches., 2010, pp. 181–192.

[WJ2011] M. Walas and K. Jassem, *Spatial reasoning and disambiguation in the process of knowledge acquisition*, Proceeding of the 5th Language and Technology Conference, 2011, pp. 420–424.

[WKNW1972] W. A. Woods, R. M. Kaplan, and B. L. Nash-Webber, *The Lunar Sciences Natural Language Information System: Final Report*, Cambridge, MA, 1972.