Instytut Informatyki Uniwersytetu Warszawskiego

Komputerowa weryfikacja opisu składni polskiej[1]

Janusz S. Bień
jsbien@mimuw.edu.pl

TR 96-06 (227)
maj 1996

[1] Referat przyjęty na LIV Zjazd Naukowy Polskiego Towarzystwa Językoznawczego, Toruń, 31.05-1.06.1996. Praca częściowo finansowana przez grant KBN, nr 8 S503 032 07 pt. *„Analizator morfologiczno-syntaktyczny dla obszernego podzbioru języka polskiego"* realizowany w Instytucie Informatyki UW w latach 1994-96 pod kierunkiem dra hab. Janusz S. Bienia

Instytut Informatyki
Uniwersytetu Warszawskiego
ul. Banacha 2
02-097 Warszawa
telefon: (48-22) 658-31-65
fax: (48-22) 658-31-64

Druk: Zakład Graficzny UW zam.nr. 242/96.

Spis treści

Institute of Informatics Warsaw University

Computer validation of a description of Polish syntax[1]

Janusz S. Bień

Abstract

The best way to validate a linguistic theory is to implement it as a computer program and to perform appropriate experiments. For the time being the most precise description of Polish syntax is the one developed by Marek Świdziński and published in the book *Formal Grammar of Polish Language* (Warsaw 1992, in Polish). Krzysztof Szafran's work on morphological analysis, based on solid linguistic basis, provided the necessary prerequisite for a syntactic parser. The AMOS-95 morphologico-syntactic analyser described in this paper consists of two parts. The parser, based on Świdziński's grammar, has been developed by the present author. The SAM-95 morphologic analyser, based on linguistic results of Zygmunt Saloni and Jan Tokarski, has been developed by Krzysztof Szafran. The first results of the experiments with the AMOS system look very promising. It is hoped that the system will make the understanding of Świdziński's grammar much easier, and in consequence will stimulate its further development and modifications.

[1]The development of the AMOS-95 system was sponsored by the KBN grant *Morphologico–syntactic analyser for a large subset of Polish* directed by Janusz S. Bień.

Komputerowa weryfikacja opisu składni polskiej

Janusz S. Bień

maj 1996

Streszczenie

Najlepszą metodą weryfikacji skomplikowanych teorii lingwistycznych jest zapisanie ich w postaci programu komputerowego i przeprowadzenie odpowiednich eksperymentów. Aktualnie najbardziej precyzyjny opis składni polskiej jest zawarty w książce Marka Świdzińskiego pt. *Gramatyka formalna języka polskiego* (Warszawa 1992). Bazujące na solidnych podstawach lingwistycznych prace Krzyszta Szafrana nad analizą morfologiczną otworzyły drogą do eksperymentów z analizą składniową. W tym celu stworzono system analizy morfologiczno–syntaktycznej AMOS–95, którego część składniowa została opracowana przez autora na podstawie gramatyki Świdzińskiego, zaś część morfologiczna (program SAM-95) — przez Szafrana na podstawie prac Zygmunta Saloniego i Jana Tokarskiego. Wstępne wyniki eksperymentów z systemem AMOS są bardzo obiecujące i pozwalają sądzić, że istotnie ułatwi on zainteresowanym lingwistom i informatykom zapoznanie się z gramatyką Świdzińskiego w wystarczającym stopniu, aby móc ją dalej rozwijać lub przystosowywać do konkretnych potrzeb.

1 Komputery a lingwistyka

Język naturalny jest obiektem bardzo złożonym. Jeśli jego opis nie ma ograniczać się do ogólnych rozważań, lecz ma go przedstawiać w sposób możliwie konkretny, wówczas taki opis również szybko staje się bardzo złożony. Choć wielu lingwistów potrafi skutecznie pracować przy pomocy tysięcy papierowych fiszek, najlepszym rozwiązaniem jest posługiwanie się odpowiednio oprogramowanym komputerem. Oczywiście, nie każda teoria lingwistyczna może być rozwijana lub weryfikowana za pomocą komputera, niezbędne jest, aby była ona sformułowana w sposób wystarczająco ścisły i jasny — nie jest to jednak zasadnicza trudność, ponieważ w ideale każda teoria lingwistyczna powinna spełniać te warunki.

Oczywiście, weryfikacja teorii lingwistycznych to tylko jeden z aspektów związków informatyki z lingwistyką, których pełne przestawienie wykracza poza ramy niniejszego artykułu. Warto jednak dodać, że rozwój tzw. technologii językowej (ang. *language technology*), czyli praktycznych i komercyjnych zastosowań takich narzędzi językowych jak np. programy do korekty ortograficznej, tworzy nowe zapotrzebowanie na szczegółowe opisy różnych aspektów języka naturalnego.[1]

2 Ścisłe opisy składni polskiej

Za pioniera ścisłego opisu składni polskiej jest słusznie uważany Henryk Misz z Uniwersytetu im. Mikołaja Kopernika w Toruniu[2]. Dla genezy omawianych tutaj prac istotnym wydarzeniem była jednak

[1] Całość tej problematyki proponuję nazywać *lingwistyką informatyczną*, zdefiniowaną jako badanie języka naturalnego z punktu widzenia potrzeb i możliwości automatycznego przetwarzania tekstów.

[2] Jego badania kontynuują Krystyna Kallas i Maria Szupryczyńska oraz ich uczniowie.

przede wszystkim praca habilitacyjna Zygmunta Saloniego, opublikowana w 1976 r. pod tytułem *Cechy składniowe polskiego czasownika* [9]. Stanowiła ona główne źródło inspiracji dla Stanisława Szpakowicza z Instytutu Informatyki Uniwersytetu Warszawskiego, który w 1978 r. w swojej pracy doktorskiej [14] przedstawił pierwszy formalny opis nietrywialnego podzbioru polszczyzny, dodatkowo zilustrowany działającym analizatorem syntaktycznym; zasadnicza część pracy została później opublikowana [15]. Praca ta wywarła w szczególności duży wpływ na Marka Świdzińskiego, który najpierw wspólnie ze Szpakowiczem, a następnie całkowicie samodzielnie kontynuował ten nurt badań, przedstawiając w 1987 r. swoje wyniki[3] jako pracę habilitacyjną [17], niemal bez zmian opublikowaną kilka lat później [18].

Wspomniana wyżej książka Świdzińskiego zawiera najpełniejszy obecnie formalny opis składni języka polskiego. Głównym tematem niniejszego artykułu jest właśnie opis aktualnie prowadzonych eksperymentów z gramatyką Świdzińskiego, a mówiąc ściślej — z opartym na tej gramatyce systemem analizy morfologiczno-syntaktycznej, nazywanym dalej **AMOS-95**. Świdziński pisze wprawdzie [18, s. 58]:

> Opis przedstawiony w niniejszej pracy ukierunkowany jest w większym stopniu lingwistycznie (empirycznie) niż informatycznie. Przyjmuję tutaj tak wysoki stopień szczegółowości empirycznej, że bezpośrednia implementacja nawet fragmentów podanej w tej pracy gramatyki nie wydaje się możliwa.

ze stanowiskiem tym jednak nigdy się nie zgadzałem. Co więcej, wyznawałem pogląd całkowicie przeciwny. Uważałem mianowicie, że do komputera należy wprowadzić opis najlepiej bez żadnych zmian i dopiero po przeprowadzeniu odpowiednich eksperymentów dokonać w razie potrzeby modyfikacji i rozszerzeń tego opisu. Ze względu na olbrzymią złożoność sformułowanej przez Świdzińskiego formalnej gramatyki języka polskiego, jej analiza „na piechotę" (czyli bez użycia komputera) wymaga tak silnej motywacji i determinacji, że trudno jej oczekiwać od typowego lingwisty, nawet jeśli specjalizuje się on w badaniach składniowych.

Można postawić sobie pytanie, dlaczego opisywane tutaj eksperymenty zostały przeprowadzone tak późno — prawie 10 lat od opublikowania pierwszej wersji gramatyki. Warto dodać, że gramatyka ta od samego początku była dostępna na nośniku komputerowym, ponieważ była ona przygotowywana do druku za pomocą komputera (w przypadku [17] był to mikrokomputer Sinclair Spectrum, w przypadku [18] — komputer PC/XT). Otóż prowadzone w Instytucie Informatyki UW doświadczenia z analizatorem Szpakowicza (por. m.in. [1], [2]) pokazały wyraźnie, że wszelkie szerzej zakrojone eksperymenty z analizą składniową są skrajnie uciążliwe, jeśli nie dysponuje się programem analizy morfologicznej, a na taki program trzeba było poczekać.

3 Analizator morfologiczny SAM-95

Możliwe są różne podejścia do problemu analizy morfologicznej. Jedną z pułapek czyhających zwłaszcza na informatyków jest założenie, że wystarczy stworzyć program korzystający z odpowiedniego słownika w postaci komputerowej bazy danych, a ktoś kiedyś wypełni tę bazę odpowiednią informacją lingwistyczną. W praktyce okazuje się jednak, że brak chętnych do tej niewdzięcznej pracy (lub — co na jedno wychodzi — brak funduszy do jej sfinansowania), a może się również okazać, że struktura bazy słownikowej nie jest dostatecznie ogólna i że nie wszystkie potrzebne hasła dają się do niej wprowadzić.

[3] Wyniki te oraz inne aspekty ścisłego opisu języka polskiego były dyskutowane głównie na seminarium Zygmunta Saloniego *Formalny opis języka naturalnego*, które odbywało się w Instytucie Informatyki UW od 1977 do 1990 roku.

Aby uniknąć tego typu niebezpieczeństw, w naszym zespole zdecydowaliśmy się na taką realizację analizy morfologicznej, która od samego początku będzie dysponować kompletną informacją lingwistyczną. Konkretnie rzecz biorąc, postanowiliśmy oprzeć się na schematycznym indeksie *a tergo* Jana Tokarskiego [20], znanym mi jeszcze w formie niedokończonego rękopisu (por. [20, s. 7]). Dzięki zaangażowaniu się kolegi Krzysztofa Szafrana w prowadzone przez Zygmunta Saloniego prace nad pośmiertnym wydaniem tego indeksu, a także w skład drukarski jego wersji ostatecznej, tekst indeksu był dostępny na nośniku komputerowym już w momencie skierowania go do druku. Pierwsza wersja programu analizy morfologicznej opracowanego przez Szafrana była gotowa w 1993, a jego opis został zawarty w pracy doktorskiej obronionej na Wydziale Polonistyki UW [10]. Była ona przeznaczona przede wszystkim do interakcyjnego hasłowania tekstu (patrz [11], [12]). Zmodyfikowana wersja tego programu, przystosowana do pracy w systemie AMOS, nosi nazwę system analizy morfologicznej **SAM-95**.

Do celów niekomercyjnych program SAM-95 jest dostępny bezpłatnie, również za pomocą sieci komputerowej[4], zaś jego opis użytkowy zawarty jest w raporcie [13], nie ma więc potrzeby omawiać go tutaj szczegółowo. Ograniczymy się zatem do podania jednego przykładu. Otóż dla tekstu

```
Jest dobrze.
```

otrzymamy w wyniku[5]

```
Jest %
{{ (3') < być(XII)+ } }%
dobrze.%
{{ () < dobrze(J)+ }%
{ (L) < dobro(nIII)+ } }%
```

Warto zwrócić uwagę, że jeśli w pliku wynikowym pominiemy informację zawartą w nawiasach klamrowych, a także wszystkie znaki procentu oraz wszystkie znaki znajdujące się za procentem do końca linii włącznie, to otrzymamy w wyniku plik wejściowy z zachowaniem oryginalnych spacji i znaków końca wiersza (oczywiście, pod warunkiem, że nawiasy klamrowe i znaki procenta nie wystąpują w tekście oryginalnym). Jest to zrobione celowo, aby przyszłe zastosowania systemu SAM mogły w razie potrzeby ukryć przed użytkownikiem wyniki analizy, ukazując mu tylko oryginalny tekst.

Do każdego słowa poddającego się analizie morfologicznej dopisywana jest w nawiasach klamrowych lista jego możliwych interpretacji; dla słowa `Jest` mamy tylko jedną interpretację, zaś dla słowa `dobrze` — dwie interpretacje. Każda interpretacja składa się z opisu — niekiedy pustego — danej formy fleksyjnej, oraz z opisu odpowiedniej formy podstawowej. Znak `+` oznacza, że dana forma nie jest tylko potencjalną, lecz jest potwierdzona przez tzw. słownik gramatyczny Zygmunta Saloniego [8]; pozostałe oznaczenia są całkowicie zgodne z *Indeksem* a tergo [20].

4 Rozszerzona gramatyka metamorficzna

Stosowany przez Świdzińskiego (a wcześniej przez Szpakowicza) formalizm to pewien wariant tzw. gramatyk metamorficznych, których twórcą jest Alain Colmerauer (por. np.[3]). Jedną z zalet tego

[4] Jego adres (tzw. URL czyli *Uniform Resource Locator*) to `ftp://ftp.mimuw.edu.pl/polszczyzna/SAM-95`.

[5] Poprawność interpretacji słowa *dobrze* jako miejscownika rzeczownika *dobro* ilustruje następujący przykład, podany mi przez Dorotę Kopcińską: *Zegarek ze złota stanowił całe jego dobro. O tym dobrze opowiadał godzinami.*

formalizmu jest jego dostępność w języku Prolog. Gramatyka metamorficzna składa się z reguł, w których po lewej stronie występuje jeden symbol nieterminalny i ewentualnie kilka symboli terminalnych, zaś prawa strona może być pusta lub zawierać dowolny ciąg symboli terminalnych, nieterminalnych i warunków. Oto kilka przykładowych reguł gramatyki Świdzińskiego:

```
KPRZYSŁ (RÓW, neg, NP, kl)
     = ZAIMNO (PRZYSŁ, p, rl, o, kl)                    (PS20)
     = ZAIMNEG (PRZYSŁ, p, rl, o, kl)                   (PS21)
       $ RÓWNE (neg, ANI.NIE).

 ZNAKKOŃCA (P)            = # ? .                      (INT1)

# f PRZECSP       = # f                                (INT5)
                     $ RÓWNE (f, ALBOWIEM.AŻ.BO.CHOCIAŻ.CHOĆ.CZY.
                              DOPÓKI.DOPÓTY.GDY.GDYŻ.IŻ.JAK.JEŚLI.
                              JEŻELI.KIEDY.NIM.PONIEWAŻ.PÓKI.PÓTY.
                              SKORO.WÓWCZAS.WTEDY.ZANIM.ŻE).
```

Dużymi literami zapisywane są symbole stałe, małe litery oznaczają zmienne, znak dolara poprzedza warunki, zaś znak # poprzedza symbol terminalny. Reguły o identycznej lewej stronie mogą być grupowane. Symbole w nawiasach okrągłych po prawej stronie nie są częścią reguł, lecz stanowią ich oznaczenia.

Na szczególną uwagę zasługuje reguła INT5, której lewa strona ma postać symbolu terminalnego, po którym następuje symbol nieterminalny. Uważny Czytelnik powinien zauważyć, że nie jest to zgodne z definicją gramatyki metamorficznej, i to nie bez powodu — tego typu konstrukcje stwarzają pewne problemy przy automatycznej analizie. Jednak ze względu na znaczenie tej reguły w gramatyce Świdzińskiego okazało się konieczne wprowadzenie w systemie AMOS odpowiedniego rozszerzenia.

Należy również pamiętać, że korzystanie przez Świdzińskiego z symboli terminalnych jest daleko idącym uproszczeniem, ponieważ w rzeczywistości mamy do czynienia nie z konkretnymi słowami, ale z wynikami — i to niekoniecznie jednoznacznymi — analizy morfologicznej. Tak więc w systemie AMOS nie używamy w praktyce symboli terminalnych, lecz dwóch specjalnych symboli nieterminalnych. Pierwszy z nich nosi nazwę `uje` (uniwersalna jednostka elementarna) i służy do odwoływania się do wyników analizy morfologicznej, drugi z nich to `inter` służący do odwoływania się do znaków interpunkcyjnych.

Stosowana przez Świdzińskiego forma zapisu była inspirowana przez tzw. składnię marsylską używaną w Prologu I, nie jest jednak z nią identyczna; co więcej, obecnie dominująca jest inna konwencja zapisu, nazywana składnią edynburską. Żadna z dostępnych wersji Prologu nie pozwala na stosowanie liter narodowych w symbolach, są one dopuszczalne tylko w tzw. napisach (ujętych w cudzysłowy). W systemie AMOS przyjęliśmy dla gramatyk metamorficznych składnię edynburską, co łącznie ze wspomnianymi wyżej modyfikacjami (oraz pewnymi technicznymi zmianami, których nie będziemy tutaj omawiać) daje nam następujący zapis przytoczonych wyżej reguł:

```
kprzysl(row, Neg, "np", Kl) -->
     s(ps20), zaimno(przysl, P, Rl, O, Kl);
     s(ps21), zaimneg(przysl, P, Rl, O, Kl),
       { rowneNeg( Neg, [ani,nie])}.
```

```
znakkonca(Z) --> s(int1), inter("?"), {Z = "p", !}.
przecsp(_)   --> s(int5), previous(F, _, _, _, _, _, _),
   { rowne(F, ["albowiem","aż","bo","chociaż","choć","czy","dopóki",
         "dopóty","gdy","gdyż","iż","jak","jeśli","jeżeli","kiedy",
         "nim","ponieważ","póki","póty","skoro","wówczas","wtedy",
         "zanim","że"]),
     !}.
```

Jak widać, lewa strona reguły oddzielona jest od prawej specjalną strzałką, prawe strony w grupie reguł oddzielane są średnikiem, warunki zaś ujmowane są w nawiasy klamrowe. Zmianie uległa też reprezentacja zmiennych, pojawiły się także tzw. zmienne anonimowe zapisywane w postaci znaku podkreślenia, zmienił się poza tym sposób reprezentacji list. Polskie litery tam, gdzie są istotne, występują w cudzysłowach, gdzie indziej są zastępowane przez odpowiednie litery łacińskie. Oznaczenia reguł stały się ich częścią, dzięki czemu mogą one występować w wynikach analizy i pomocniczych wydrukach.

Dzięki temu, że Marek Świdziński udostępnił nam swoją książkę [18] na nośniku komputerowym (za co mu serdecznie dziękuję), znacząca część tych przekształceń mogła być dokonana automatycznie. W następnych punktach omówimy te niezbędne zmiany, które niestety nie mogły być dokonane mechanicznie.

5 Strategia analizy syntaktycznej

Dla każdej gramatyki formalnej możliwe są różne strategie analizy (lub syntezy) zdań opisywanego przez nią języka. W przypadku gramatyk metamorficznych standardowo dostępna jest strategia zstępująca od lewej do prawej (ang. *top-down, left-to-right*). Strategia ta powoduje pewne problemy z niektórymi regułami rekurencyjnymi. Oto przykład kilku reguł z gramatyki Świdzińskiego:

```
ZR (wf, a, c, t, rl, o, neg, i, z)
     = ZSZ (wf, a, c, t, rl, o, neg, i, z).                (R1)
...
ZSZ (wf, a, c, t, rl, o, neg, i, z)
     = ZJ (wf, a, c, t, rl, o, neg, i, z, oz)              (S1)
       $ RÓŻNE (oz, LUB).
...
ZJ (wf, a, c, t, rl, o, neg, i, z, PRZEC)
     = ZP (wf, a, c, t, rl, o, neg, i, z).                 (J1)
...
ZP (wf, a, c, t, rl, o, neg, i, z)
     = ZE (wf, a, c, t, rl, o, wa, wb, wc, neg, i, z, ow). (P1)
...
ZE (wf, a, c, t, rl, o, wa, wb, wc, neg, i, z, ow)
     = ZR (wf, a, c, t, rl, o, neg, i, z1)                (E19)
       $ OBLZAL (z1, z, ow).
```

Z reguł tych wynika, że analiza zdania równorzędnego poprzez zdanie szeregowe, jednorodne, proste i elementarne doprowadza nas znowu do analizy zdania równorzędnego, przy czym nie następuje żadna

zmiana analizowanego napisu wejściowego. W rezultacie proces analizy zapętla się, nie dostarczając wyniku. Podobnie wygląda sprawa np. dla frazy wymaganej.

```
FW (tfw, k, a, c, rl, o, neg, i, z)
     = FW1 (tfw, k, a, c, rl, o, neg, i, z)                (WY1)
       $ RÓŻNE (tfw, NIC).
...
FW1 (a1, k, a, c, rl, o, neg, i, z)
     = FWE (BOK, a1, c, t, rl1, o1, wa, wb, wc, k, neg, i, z).
                                                           (WY6)
...
FWE (wf, a, c, t, rl, o, wa, wb, NIC, k, neg, i, z)
     = FW (wc, k, a, c, rl, o, neg, i, z1)                 (WE3)
       KWENEG (wf, a, c, t, rl, o, wa, wb, wc, k, neg, NI, z)
       $ RÓWNE (z, P.P'.P")
       $ RÓWNE (z1, NP.P).
```

Fraza wymagana to m.in. fraza właściwa (reguła WY1),która może mieć realizację bezokolicznikową (reguła WY6) w postaci frazy werbalnej elementarnej, ta zaś może rozpoczynać się od frazy wymaganej itd. (por. m.in. reguły WE3). Jest to przykład tzw. rekursji lewostronnej, która może doprowadzić do zapętlenia przy każdym wystąpieniu frazy wymaganej, nie tylko na początku zdania.

Choć w informatyce znane są różne metody eliminowania niepożądanych typów rekursji, ich użycie tutaj byłoby kłopotliwe m.in. dlatego, że przekształcone reguły zatraciłyby swój związek z gramatyką Świdzińskiego. W rezultacie każdy taki przypadek traktujemy indywidualnie, starając się wniknąć w jego sens lingwistyczny, a następnie dodając do odpowiednich reguł dodatkowe parametry i warunki.

6 Efektywność analizy syntaktycznej

Oryginalna gramatyka Świdzińskiego ma charakter statyczny, tj. nie zakłada ona żadnej szczególnej kolejności wykonywania reguł i warunków. Przekształcając ją w rutynowy sposób na program komputerowy, przesądzamy, że poszczególne elementy gramatyki są wykonywane w kolejności ich zapisania w pliku komputerowym. W przypadku samych reguł i symboli nieterminalnych prowadzi to do wspomnianej wyżej strategii zstępującej od lewej do prawej, ale w przypadku warunków sprawa staje się bardziej skomplikowana. Popatrzmy na następujący przykład

```
WYPOWIEDZENIE
     = # %                                                 (W1)
       ZR (wf, a, c, t, rl, o, neg, i, z)
       ZNAKKOŃCA (z).
...
KWER1 (wf, a, c, t, rl, o, wa, wb, wc, k, i, z)
     = FORMACZAS (wf, a, c, t, rl, o, wa, wb, wc, k, i, z) (WE30)
       $ RÓWNE (z, NP.NP'.NP")
     = FORMACZAS (wf, a, c, t, rl, o, wa, wb, wc, k, z)    (WE31)
       $ RÓWNE (z, NPt.NP't.NP"t.P.P'.P".PZ)
       $ RÓŻNE (t, ROZ)
```

```
= FORMACZAS (wf, a, c, t, rl, o, wa, wb, wc, k, z)    (WE32)
  $ RÓWNEt(z, AŻ1".AŻ2.BO.BO'.BO".BOWIEM.CHOCIAŻ.CO.CZY.
              DOPÓKI.GDY.GDY".JAK.JAKI.JEŚLI.KTO.KTÓRY.
              PODCZAS. PONIEWAŻ.ŻE). PONIEWAŻ.ŻE)
  $ RÓŻNE (wf, BOK)
  $ RÓŻNE (t, ROZ).
```

Ostateczna wartość parametrowi zależności `z` zostanie nadana dopiero po rozpoznaniu typu znaku interpunkcyjnego kończącego zdanie, nie jest więc ona znana, kiedy analizujemy główną konstrukcję werbalną. Załóżmy dla ustalenia uwagi, że zdanie może być interpretowane jako twierdzące, ale kończy się znakiem zapytania. Jeśli warunek potraktujemy jako podstawienie, to najpierw reguła WE30 nada parametrowi `z` wartość `NP`, zdanie zostanie zanalizowane jako zdanie szeregowe, po czym analiza ta zostanie odrzucona jako niezgodna z typem znaku końca. Ponieważ program nie dysponuje bezpośrednią informacją o przyczynie niepowodzenia, będzie musiał przeanalizować wszystkie możliwości innej interpretacji zdania, co doprowadzi go z czasem do warunku w regule WE30. Tym razem parametrowi `z` zostanie nadana wartość `NP'` i oczywiście sytuacja się powtórzy. Dopiero za siódmym razem reguła WE31 nada parametrowi `z` właściwą wartość `P`. Należy jednak pamiętać, że ten sam problem występuje również w innych regułach, a także dla innych parametrów. Nałożenie się tych czynników na siebie powoduje eksplozję kombinatoryczną możliwości i tak długi czas analizy zdania (rzędu kilku godzin na szybkim komputerze), że jest on absolutnie nie do przyjęcia.

W tym konkretnym przypadku najwłaściwszym rozwiązaniem jest sprawdzanie warunków konstrukcji werbalnej dopiero po otrzymaniu wartości przez parametr `z`, tj. praktycznie po zanalizowaniu całego napisu wejściowego. Choć pozornie wydaje się to trudne, w rzeczywistości jest to bardzo łatwe do zrealizowania, pod warunkiem jednak, że wykorzystywana implementacja języka Prolog dysponuje tzw. korutynami. Możemy wówczas „zamrozić" warunek do czasu nadania wartości wskazanej zmiennej lub w inny sposób opóźnić jego wykonanie. Niestety, nie można tego dokonywać mechanicznie, ponieważ wówczas niektóre zmienne nigdy nie otrzymałyby wartości i analiza uległaby zablokowaniu. Tak więc dla każdego warunku musimy najpierw przeanalizować jego sens lingwistyczny i dopiero potem wprowadzić odpowiednie modyfikacje.

Inną przyczynę nieefektywności oryginalnej gramatyki Świdzińskiego najlepiej zilustrować przykładem konstrukcji nominalnej z atrybutem. Dla przejrzystości jest ona wyrażona za pomocą 18 reguł. Jedna reguła opisuje realizację prostą, druga realizację niestandardową. Pozostałe 16 reguł dzieli się na dwie grupy. Jedna z nich opisuje realizację postaci: najpierw konstrukcja nominalna z inkorporacją, potem fraza przymiotnikowa; druga grupa różni się od pierwszej tylko szykiem: najpierw fraza przymiotnikowa, potem konstrukcja nominalna z inkorporacją. Wewnątrz każdej grupy reguły różnią się tylko wartościami parametrów i warunkami, przy czym warunki te są w obu grupach analogiczne — mamy więc łącznie tylko 8 warunków. Przy oryginalnych regułach w pechowym przypadku program musi 7 razy stworzyć i odrzucić poprawną analizę, aż trafi na tę regułę, która będzie zawierała warunek właściwy dla analizowanej sytuacji. Oryginalne 18 reguł można jednak zredukować do czterech, a 8 warunków skomasować w jeden, co pozwala na istotne przyspieszenie działania analizatora.

Istnieje jeszcze jedna możliwość prostego zwiększania efektywności analizy, którą zilustrujemy przykładem frazy luźnej.

```
FL (a, c, rl, o, neg, i, z)
     = FL1 (a, c, rl, o, neg, i, z).                    (LU1)
```

```
FL (a, c, rl, o, neg, i, P)
     = FL1 (a, c, rl, o, neg, i, z1)                    (LU2)
       FL (a, c, rl, o, neg, NI, z2)
       $ RÓWNE (P, z1.z2)
       $ RÓWNE (z1, P.NP)
       $ RÓWNE (z2, P.NP).
...
FL (a, c, rl, o, neg, i, z)
     = FL1 (a, c, rl, o, neg, i, z)                     (LU4)
       FL (a, c, rl, o, neg, NI, NP)
       $ RÓŻNE (z, P.PZ).
```

Jeśli analizowana fraza luźna jest zgodna z jedną z reguł LU2-LU4, to jej początek jest zgodny z regułą LU1. Tak więc najpierw zostanie użyta reguła LU1 (a fraza luźna w ogólnym przypadku może być bardzo skomplikowana), po czym ta poprawna analiza zostanie odrzucona, analizator zaś przejdzie do jednej z następnych reguł. Aby uniknąć takiego marnotrawstwa, należy skomasować reguły tak, aby po zanalizowaniu frazy luźnej właściwej (oznaczonej symbolem `FL1`) była możliwa w razie potrzeby kontynuacja w postaci rekurencyjnej analizy frazy luźnej; oczywiście, towarzyszyć temu musi odpowiednia modyfikacja warunków. W rezultacie zamiast czterech reguł LU1–LU4 otrzymujemy znacznie bardziej efektywną regułę następującą:

```
 fl(ob, A, C, Rl, O, Neg, I, Z)
      --> s(lu(Lu)), fl1(A, C, Rl, O, Neg, I, Z1),
          (  {Lu=1, Z=Z1}
           ; fl(ob, A, C, Rl, O, Neg, "ni", Z2),
             {  Lu=2, rowneZ( "p", [Z1,Z2]), rowneZ( Z1, ["p","np"]),
                      rowneZ( Z2, ["p","np"]), Z="p"
              ; Lu=3, Z1="pz", rowneZ( Z2, ["p","np"]), Z=Z1
              ; Lu=4, rozneZ( Z1, ["p","pz"]), Z2="np", Z=Z1}
          ).
```

7 Poprawność i adekwatność gramatyki Świdzińskiego

Przy tworzeniu gramatyki formalnej, podobnie jak przy pisaniu programu komputerowego, nie do uniknięcia są pomyłki i przeoczenia. Oto jeden z przykładów z książki [18, s. 365], poprawiony zresztą przez autora w udostępnionej nam wersji komputerowej.

```
FW1 (MIAN, k, a, c, rl, o, neg, i, z)
     = FNO (p, rl, o, neg, i, z, kl).                   (WY8)

FW1 (BIER, k, a, c, rl, o, neg, i, z)
     = FNO (DOP, rl, o, neg, i, z, kl).                 (WY9)
       $ RÓWNE (neg, ANI.NIE).
FW1 (BIER, k, a, c, rl, o, TAK, i, z)
     = FNO (BIER, rl, o, neg, i, z, kl).                (WY10)
```

```
FW1 (p, k, a, c, rl, o, neg, i, z)
     = FNO (p, rl, o, neg, i, z, kl)                    (WY11)
       $ RÓŻNE (p, MIAN.BIER.WOŁ).
```

Reguły te opisują nominalne realizacje frazy wymaganej. Reguła WY8 zawiera oczywisty błąd, ponieważ pozwala realizować wymaganie mianownikowe przez frazę nominalną w dowolnym przypadku. Po zmianie parametru `p` na `MIAN` ujawnia się niekonsekwencja polegająca na tym, że reguła WY8 byłaby szczególnym przypadkiem reguły WY11, gdyby nie warunek zawarty w tej ostatniej. W regule WY8 liczba, rodzaj i osoba frazy nominalnej — reprezentowane przez parametry `rl` i `o` — są uzgodnione z wymagającą tej frazy konstrukcją werbalną, ale to samo zachodzi również w pozostałych regułach! Oczywiście, jest to błędem i powyższe reguły powinny mieć postać następującą[6]:

```
FW1 (MIAN, k, a, c, rl, o, neg, i, z)
     = FNO (MIAN, rl, o, neg, i, z, kl).                (WY8)

FW1 (BIER, k, a, c, rl, o, neg, i, z)
     = FNO (DOP, rl1, o1, neg, i, z, kl).               (WY9)
       $ RÓWNE (neg, ANI.NIE).
FW1 (BIER, k, a, c, rl, o, TAK, i, z)
     = FNO (BIER, rl1, o1, neg, i, z, kl).             (WY10)

FW1 (p, k, a, c, rl, o, neg, i, z)
     = FNO (p, rl1, o1, neg, i, z, kl)                 (WY11)
       $ RÓŻNE (p, MIAN.BIER.WOŁ).
```

Trzeba z przyjemnością stwierdzić, że tego typu błędy sprawiały o wiele mniej kłopotów niż można było oczekiwać. W niektórych przypadkach poprawki były zupełnie oczywiste, w niektórych intencje autora wyjaśniały przykłady towarzyszące regule, w jeszcze innych wystarczyło sięgnąć do omówienia w zasadniczej części książki.

Metody oceny adekwatności gramatyk wyczerpująco omówił Mirosław Bańko w swoich pracach [1] i [2]. Zwrócił on m.in. uwagę na to, że w ogólnym przypadku gramatyka formalna przyporządkowuje jednemu zdaniu wiele możliwych interpretacji. Czasami różnią się one istotnie, ale przeważnie są to tylko różne warianty grupowania składników; np. gramatyka Szpakowicza zdaniu

> Położyła kucharka na stole: kartofle, buraki, marchewkę, fasolę, kapustę, pietruszkę, selery i groch.

przyporządkowuje 429 równie poprawnych interpretacji. Zdaniem Bańki — z którym się całkowicie zgadzamy — do oceny adekwatności zarówno obserwacyjnej, jak i opisowej, wystarczy brać pod uwagę *pierwszą* interpretację znalezioną przez analizator, pomimo tego, że jest praktycznie sprawą przypadku, która z możliwych interpretacji zostanie znaleziona jako pierwsza.

Przy przekształcaniu gramatyki Świdzińskiego na analizator syntaktyczny starałem się wyeliminować niektóre ewidentnie zbędne interpretacje. Dla przykładu, każde zdanie elementarne, które da się zanalizować za pomocą reguły E5 i E6 [18, s. 357], da się również zanalizować za pomocą reguły E4

[6] Dostrzeżenie wszystkich różnic między obu wersjami reguł WY8–WY11 pozostawiam Czytelnikowi jako ćwiczenie.

z pustą realizacją frazy luźnej. Aby uniknąć takich interpretacji, symbol `FL` frazy luźnej został uzupełniony o dodatkowy parametr, określający, czy dana fraza jest obligatoryjna (niepusta) czy fakultatywna (potencjalnie pusta). Oczywiście, reguła E4 została tak zmodyfikowana, aby występująca na początku zdania fraza luźna była obligatoryjna. Pomimo tych ograniczeń trzeba się jednak nadal liczyć z tym, że pierwsza znaleziona przez analizator interpretacja może być zaskakująca. Tak np. dzięki temu, że analiza morfologiczna obejmuje prawie wszystkie hasła słownika Doroszewskiego, dla zdania

Chciano dać dziewczynie kwiaty.

jako pierwsza może być znaleziona interpretacja, w której *dziewczynie* jest formą przymiotnika **dziewczyni** i określa rzeczownik *kwiaty*.

Ocena adekwatności opisowej gramatyki opiera się na ocenach wyników analiz poszczególnych zdań. Dla zdań zakceptowanych przez analizator należy ustalić, czy przyporządkowana zdaniu struktura jest właściwa — może to niekiedy budzić wątpliwości, ale w zasadzie tryb postępowania jest oczywisty. Nie jest natomiast oczywiste, w jaki sposób stwierdzić, czy zdanie nie zaakceptowane przez analizator zostało odrzucone z uzasadnionych powodów. Aby ułatwić to zadanie, analizator systemu AMOS produkuje częściowe drzewo analizy również dla zdań odrzuconych — cecha taka jest bardzo rzadko dostępna w analizatorach syntaktycznych. Użyta przez nas metoda została zainspirowana przez system ORBIS prezentowany ustnie przez Alaina Colmerauera na konferencji *Natural Language Understanding and Logic Programming* w Rennes w r. 1984 (pewien wariant metody Colmerauera został później opisany w artykule [7]).

Adekwatność obserwacyjna gramatyki abstrahuje od tego, jakie struktury są przyporządkowane zdaniu, istotne jest tylko, czy zdanie jest zaakceptowane czy odrzucone. W cytowanych pracach Mirosław Bańko sformułował wygodną miarę adekwatności obserwacyjnej, zdefiniowaną za pomocą formuły

$$A_{obs} = \frac{P_A + N_O}{Z},$$

gdzie Z to liczba analizowanych zdań, P_A to liczba zaakceptowanych zdań poprawnych, zaś N_O to liczba niepoprawnych zdań odrzuconych. Dla analizatora Szpakowicza testowanego na zbiorze przykładów z pracy [16] wartość adekwatności obserwacyjnej wyniosła 53%. W przyszłości zamierzamy ustalić wartość adekwatności obserwacyjnej również dla gramatyki Świdzińskiego.

8 Zasoby słownikowe systemu AMOS

W systemie AMOS moduł analizy morfologicznej SAM-95 oparty jest o schematyczny indeks *a tergo* polskich form wyrazowych [20], którego podstawowym zadaniem jest rozpoznawanie wszystkich napisów będących potencjalnie formami fleksyjnymi jakichś wyrazów. Ponieważ rozpatrywanie takich potencjalnych możliwości istotnie utrudniłoby przebieg analizy syntaktycznej, zachodzi potrzeba weryfikacji, czy hipotetyczny wyraz rzeczywiście istnieje w języku polskim. Obecnie problem ten można by rozwiązać za pomocą komputerowej wersji indeksu *a tergo* do słownika Doroszewskiego [5]. Od niedawna dzięki Robertowi Wołoszowi i zgodzie kierownictwa zespołu, który indeks opracował, jest on obecnie publicznie dostępny na nośniku komputerowym [23][7]. W momencie rozpoczynania prac nad systemem AMOS musieliśmy jednak oprzeć się na innych danych, otrzymanych od Zygmunta Saloniego, za które mu dziękujemy.

[7] Jego adres (URL) to `ftp://ftp.mimuw.edu.pl/pub/polszczyzna/a_terDor/`.

Informacja dostępna w wymienionych źródłach jest ograniczona, nie zawiera w szczególności informacji o aspekcie czasowników i o podrodzajach rzeczowników męskich. Dane te dla około 120 000 haseł zostały wprowadzone do komputera w ramach kierowanego przez Zygmunta Saloniego grantu KBN pt. *Słownik gramatyczny współczesnego języka polskiego* (por. [8]). Niestety, ze względu na odmienny i niedostatecznie konsekwentny format tych danych nie jest możliwe ich pełne i bezpośrednie wykorzystanie przez system AMOS, mamy jednak nadzieję, że problem ten uda nam się rozwiązać w niezbyt odległej przyszłości. Warto podkreślić, że informacje o aspekcie i podrodzajach nie są niezbędne do testowania gramatyki Świdzińskiego, utrudniają tylko interpretację wyników.

Gramatyka Świdzińskiego zakłada dostępność słownika zawierającego różne informacje gramatyczne o poszczególnych słowach, w szczególności schematy wymagań dla czasowników. Analizator syntaktyczny został skontruowany tak, że schematy wymagań nie są niezbędne do jego działania. Praktyka pokazała jednak, że ich brak zbyt często prowadzi do błędnych lub nieintuicyjnych wyników. W rezultacie konstruowany jest sukcesywnie prowizoryczny słownik własności gramatycznych dostosowany do aktualnie przetwarzanych przykładów. W przyszłości będzie można rozważyć wykorzystanie bazy schematów czasownikowych opracowywanej w ramach grantu KBN kierowanego przez Marka Świdzińskiego.

9 Dane testowe

Obecnie system AMOS jest wykorzystywany do przetwarzania przykładów zaczerpniętych z aneksu do książki [18]; zestaw ten nazywamy minikorpusem GFJP-A. Liczy on 660 przykładów, na które składają się zarówno zdania poprawne, jak i niepoprawne. W oryginale większość przykładów ma mniej lub bardziej dokładnie zaznaczoną strukturę, a miejsce ich wystąpienia jednoznacznie wskazuje na regułę, którą one ilustrują. Informacje te zostały usystematyzowane w sposób zilustrowany poniżej.

```
% 334

[W1; GFJP A5-1; 1]
Ja zostałem.
```

Oprócz właściwego przykładu widzimy tu dwa rodzaje komentarzy — wprowadzone znakiem procentu i ujęte w nawiasy prostokątne. Pierwszy komentarz to po prostu numer strony w książce [18], na której znajdują się następujące po nim przykłady. Drugi komentarz zawiera symbol reguły (R2), do której odnosi się przykład, symbol minikorpusu (GFJP) oraz lokalizacja przykładu — w tym przypadku aneks punkt 5.1. Ostatnia liczba to kolejny numer przykładu w korpusie.

```
[R2; GFJP A5-2.2; 5]
Ja zostałem, | a | kto pójdzie?

[R2; GFJP A5-2.2; 6]
Ona przyszła, <ja zaś zostałem, | ale | kto pójdzie>?

[*R3; GFJP A5-2.2; 12]
Kto został, | on natomiast pójdzie?

[E1; GFJP A5-6.1.1; 181]
Gdzie | idziecie _ _ _?
```

W przykładzie nr 5 mamy zaznaczoną strukturę zdania. W oryginale przecinek stanowi również wydzielony składnik, a poszczególne składniki są opatrzone odpowiednimi etykietami. Założyliśmy jednak, że zakodowanie tej informacji byłoby zbyt pracochłonne i zbytnio zaciemniałoby przykłady.

Przykład nr 6 jest pełnym zdaniem, ale regułę R2 ilustruje tylko ten jego fragment, który jest ujęty w nawiasy kątowe; struktura tego fragmentu jest zaznaczona w sposób omówiony wyżej. Przykład nr 12 to zdanie niepoprawne ilustrujące regułę R3. W przykładzie nr 181 dla poprawnego zaznaczenia struktury niezbędne było — podobnie jak w oryginale — wskazanie składników o realizacji pustej.

Warto podkreślić, że przetwarzanie przykładu przez system AMOS zachowuje te dodatkowe informacje, które są widoczne w ostatecznym wyniku i istotnie ułatwiają stwierdzenie, czy uzyskany wynik jest zgodny z intencjami autora gramatyki.

Drugi zestaw danych testowych to tzw. minikorpus GFJP-B zawierający pozostałe 1377 przykładów z książki [18]. Po przetworzeniu obu tych korpusów za pomocą systemu AMOS proces weryfikacji gramatyki Świdzińskiego uznamy za zakończony, co pozwoli zająć się jej oceną — w szczególności obliczeniem wartości adekwatności obserwacyjnej — i bardziej zaawansowanymi eksperymentami.

Warto tutaj wspomnieć, że były czynione już próby ręcznej weryfikacji gramatyki Świdzińskiego. Jak pisze on w pracy [19, s. 21], dla 855 zdań z korpusu słownika frekwencyjnego [6] ręczna symulacja procesu analizy dała poprawne wyniki dla ponad 800 zdań. Ciekawe, jakie będą wyniki systemu AMOS dla identycznych danych.

Ciekawe byłoby również sprawdzić działanie systemu na korpusie zawartym w książce Zygmunta Vetulaniego [21], której kompletny tekst na nośniku komputerowym uprzejmie udostępnił mi jej autor, za co jestem mu bardzo zobowiązany. Jest to korpus dialogów zgromadzony empirycznie i używany do eksperymentów z tzw. analizatorem BPII (ang. *Basic Polish for Information Interchange*). Choć analizator ten jest zorientowany na konkretne zastosowanie w systemach konwersacyjnych (por. [22]), porównanie go z systemem AMOS byłoby niewątpliwie interesujące.

10 Struktura systemu AMOS

System AMOS składa się z kilku programów, których współpraca jest koordynowana przez wywodzący się z systemu UNIX program `make`. Moduł analizy morfologicznej jest napisany w języku C, pozostałe moduły w języku Sicstus Prolog i perl; wykorzystywany jest również system składania tekstów TEX i inne narzędzia. Autorem koncepcji i programu analizy morfologicznej jest Krzysztof Szafran. Autorem formalizmu rozszerzonych gramatyk metamorficznych, modyfikacji gramatyki Świdzińskiego i koncepcji pozostałych modułów systemu jest niżej podpisany. Moduły te zaprogramował w językach Prolog i perl Marcin Woliński (student informatyki), przygotował on również odpowiednie definicje dla systemu TEX. Minikorpusy GFJP przygotowała Marta Nazarczuk (studentka polonistyki) na podstawie tekstu książki [18] skonwertowanego z formatu ChiWriter przez niżej podpisanego i Marcina Wolińskiego.

W całości system działa na komputerach PC z systemem operacyjnym DOS, ale większość modułów — w szczególności analizator syntaktyczny — może być bez żadnych zmian używana pod systemem operacyjnym UNIX; przeniesienie w razie potrzeby innych modułów pod system UNIX nie powinno stwarzać problemów.

Dane do przetworzenia należy wprowadzić do pliku o rozszerzeniu `852` (rozszerzenie to niesie informację, że polskie litery reprezentowane są zgodnie z tzw. stroną kodową 852); poszczególne zdania muszą być oddzielone pustym wierszem, ale ograniczenie to będzie usunięte w przyszłości. Następnym etapem jest przetworzenie tekstu przez analizę morfologiczną — wynikiem jest plik o tej samej nazwie, lecz o rozszerzeniu `amo`; przykład zawartości takiego pliku podaliśmy w punkcie 3 na s. 3. Kolejny etap,

to przetworzenie wyników analizy morfologicznej przez program, który żartobliwie nazywamy wywoływaczem. Ma on kilka zadań. Po pierwsze, rozpoznaje granice między zdaniami w pliku wejściowym. Po drugie, dla każdego zdania wypisuje komendę w języku Prolog uaktywniającą analizę syntaktyczną. Po trzecie, uzupełnia wyniki analizy morfologicznej wyrazów o niezbędne informacje gramatyczne pobrane z prowizorycznego słownika. Po czwarte, nadaje tym informacjom bardziej jawną i jednolitą strukturę. Tak więc po wykonaniu tej operacji możemy otrzymać następujący plik

```
:-analiza("Jest dobrze.
", [
  wa(1, [[
    wg("Jest", [
      wm("być", 'XII', v(ter), [km(sg, X0, X1, 3)], X2)])]], X3),
  wa(2, [[
    wg("dobrze", [
      wm("dobrze", 'J', a(psr), [], X4)])], [
    wg("dobrze", [
      wm("dobro", [n, 'III'], n(X5), [km(pl, [l(X6)], X7, X8)], X9)])]], X10),
  i(3, ".")]).
```

Symbol `wa` oznacza wyraz alfabetyczny, `wg` — wyraz grafemiczny, zaś `wm` — wyraz morfologiczny. Pojęcia te są używane zgodnie z definicjami przedstawionymi m.in. w książce [4]. Reprezentacja wyrazu alfabetycznego zawiera numer kolejny wyrazu oraz listę — w powyższym przykładzie jednoelementową — możliwych podziałów wyrazów alfabetycznych na wyrazy grafemiczne (por. przykład nr 4 zamieszczony w dodatku). Każdy możliwy podział jest reprezentowany przez listę — w naszym przykładzie znów jednoelementową — składowych wyrazów morfologicznych. Reprezentacja wyrazu morfologicznego rozpoczyna się od postaci hasłowej, potem następuje tzw. opis słownikowy będący w praktyce wzorem odmiany, następnie oznaczenie fleksemu, wartości kategorii morfologicznych i opis gramatyczny; oznaczenie fleksemu oraz kategorie morfologiczne oznaczane w zasadzie w sposób opisany w [4], choć na potrzeby systemu AMOS wprowadzono pewne rozszerzenia. Jak widać z naszego przykładu, nie wszystkie wartości muszą być ustalone, stąd na niektórych pozycjach występują nazwy zmiennych (zaczynające sie od litery `X`).

Następnym i niekiedy ostatnim etapem jest właściwa analiza syntaktyczna. Normalnie interesuje nas tylko wynik końcowy, otrzymywany na ekranie i w pliku o rozszerzeniu `asy`, ale w razie potrzeby możemy spowodować wypisywanie pewnych informacji o przebiegu przetwarzania. Kiedy wielkość otrzymanego jako wynik drzewa analizy syntaktycznej przekracza rozmiar ekranu czy jednej strony papieru, jego przeglądanie staje się dość kłopotliwe. Dlatego normalnie tworzymy dodatkowo plik o rozszerzeniu `plt` (polski LaTeX), który pozwala nam wydrukować wyniki (lub obejrzeć je na ekranie) za pomocą systemu TeX. Na tym etapie możemy również dokonać kompresji drzewa przez pominięcie niektórych gałęzi. Jak już wspominaliśmy, dla zdań odrzuconych otrzymujemy częściowe drzewo analizy, co ilustruje przykład nr 1 podany w dodatku. Znajdziemy tam także przykłady różnych poprawnych drzew analizy.

Warto dodać, że wynik składu za pomocą systemu TeX może być poddany dalszemu przetwarzaniu, np. w celu przekształcenia wyników do postaci nadającej się do udostępnienia w Internecie za pomocą „pajęczyny" czyli World Wide Web; wybrane wyniki systemu AMOS są już udostępnione w ten sposób pod adresem[8] `ftp://ftp.mimuw.edu.pl/pub/polszczyzna/amos/amos.html`.

[8] Adres ten należy podać przeglądarce World Wide Web; może on w przyszłości ulec zmianie.

11 Podsumowanie

W książce [18, s. 307] znajdujemy następujące stwierdzenie:

> W dyskusji nad pewnymi fragmentami niniejszej gramatyki Janusz Bień wyraził pogląd, że osiągnięte tu zostało maksimum możliwości przyjętego formalizmu — gramatyki metamorficznej, że, innymi słowy, niewiele więcej dałoby się uzyskać.

Zaszło tutaj pewne nieporozumienie. W roboczych dyskusjach — jeśli nie zawodzi mnie pamięć — wyrażałem bowiem nieco inny pogląd, że tak złożona gramatyka formalna zbliża się do granic możliwości percepcji czytelnika czy użytkownika, a może nawet je przekracza. Podstawowe znaczenie systemu AMOS widzę w możliwości stworzenia bogatej kolekcji zanalizowanych przykładów, które istotnie ułatwią zainteresowanych lingwistom i informatykom zapoznanie się z gramatyką Świdzińskiego w wystarczającym stopniu, aby móc ją dalej rozwijać lub przystosowywać do konkretnych potrzeb.

Bibliografia

[1] M. Bańko, Analiza polskich fraz rzeczownikowych testem adekwatności i efektywności parsera Szpakowicza. Praca magisterska (opiekun J. S. Bień), Instytut Informatyki UW, 1985.

[2] M. Bańko, *Niektóre problemy oceny adekwatności gramatyk (na przykładzie fragmentu gramatyki Szpakowicza)*. **Studia Gramatyczne** IX (1990), s. 55-72.

[3] J. S. Bień. *Lingwistyka informatyczna we Francji*. **Polonica**, III:234–255, 1977.

[4] J.S. Bień, **Koncepcja słownikowej informacji morfologicznej i jej komputerowej weryfikacji**. Wydawnictwa Uniwersytetu Warszawskiego, Warszawa 1991.

[5] **Indeks *a tergo* do *Słownika języka polskiego* pod redakcją Witolda Doroszewskiego**. Opr. zespół pod kierunkiem R. Grzegorczykowej i J. Puzyniny. PWN, Warszawa 1973.

[6] I. Kurcz, A. Lewicki, J. Sambor, K. Szafran, J. Woronczak, **Słownik frekwencyjny współczesnej polszczyzny pisanej**. Instytut Języka Polskiego PAN, Kraków 1990.

[7] P. Rincel, P. Sabatier, *Using the Same System for Analyzing and Synthesizing Sentences*, H. Karlgren (ed.), **COLING-90**, pp. 440-442.

[8] Zygmunt Saloni. Słownik gramatyczny języka polskiego — wersja przedwstępna. Praca niepublikowana, 1994.

[9] Z. Saloni. **Cechy składniowe polskiego czasownika**. Wrocław 1976.

[10] K. Szafran, Automatyczna analiza fleksyjna tekstu polskiego (na podstawie *Schematycznego indeksu a tergo* Jana Tokarskiego). Praca doktorska, Wydział Polonistyki UW, 1993.

[11] K. Szafran, *Automatic Lemmatisation of Texts in Polish — is it Possible?. First European Conference on Formal Description of Slavic Languages*, Leipzig, 30.11-2.12.1995 (w druku).

[12] K. Szafran, *Automatyczne hasłowanie tekstu polskiego*. **Polonica** (złożone do druku).

[13] K. Szafran. Analizator morfologiczny SAM-95 — opis użytkowy. Raport Instytutu Informatyki Uniwersytetu Warszawskiego TR 96–05 (226), maj 1996.

[14] S. Szpakowicz, Automatyczna analiza składniowa polskich zdań pisanych, praca doktorska, Instytut Informatyki UW, 1978.

[15] S. Szpakowicz, **Formalny opis składniowy zdań polskich**, 2. wyd. Wydawnictwa Uniwersytetu Warszawskiego 1986.

[16] S. Szpakowicz, M. Świdziński, *Formalna definicja równorzędnej grupy nominalnej we współczesnej polszczyźnie pisanej.* Warszawa 1981 (maszynopis powielony), także *Studia Gramatyczne* IX (1990), s. 9-54.

[17] M. Świdziński, Formalny opis składniowy polskich zdań o składniku zdaniowym. Wydział Polonistyki UW, Warszawa 1987 (maszynopis powielony).

[18] M. Świdziński, **Gramatyka formalna języka polskiego**. Wydawnictwa Uniwersytetu Warszawskiego 1992.

[19] M. Świdziński, *Od interpretacji do faktów językowych: weryfikacja empiryczna gramatyki formalnej.* **Biuletyn Polskiego Towarzystwa Językoznawczego** XLIX (1983), s. 15–24.

[20] Jan Tokarski (opracowanie i redakcja Zygmunt Saloni), **Schematyczny indeks *a tergo* polskich form wyrazowych**. Wydawnictwo Naukowe PWN, Warszawa 1993.

[21] Z. Vetulani, **Corpus of consultative dialogues**. Experimentally collected source data for A[rtificial]I[ntelligence] application. UAM Press, Poznań 1990.

[22] Z. Vetulani, *Język polski jako interfejs w komunikacji człowiek–komputer: system POLINT.* **Poznańska Szkoła Matematyczna**, Poznań 1995.

[23] R. Wołosz, *Errata do* Indeksu *a tergo* do *Słownika języka polskiego* pod redakcją Witolda Doroszewskiego. **Poradnik Językowy**, w druku.

A Przykładowe wyniki analizy

Na następnych stronach znajdują się kolejno:

1. przykład częściowej analizy zdania niepoprawnego,
2. pełne drzewo analizy zdania *Dał im pieniądze ojciec*,
3. skrócone drzewo analizy tego samego zdania,
4. skrócone drzewo analizy zdania *Czyżbyś poszedł?*,
5. skrócone drzewo analizy zdania *Ja zostanę, on przyjdzie*,
6. skrócone drzewo analizy zdania *Matka dała komu pieniądze?*,
7. skrócone drzewo analizy zdania *To, że przyjechała, pamiętano*,
8. skrócone drzewo analizy zdania *Wiem, że on przyjdzie.*

Dla znających książkę [18] interpretacja wyników powinna być oczywista. Symbole nieterminalne i ich parametry są z nielicznymi wyjątkami zgodne z książką; jeśli z jakichś powodów dodano nowe parametry, to znajdują się one zawsze na początku. W nawiasach prostokątnych znajdują się symbole reguł. Symbole składające się z jednej lub więcej cyfr zaczynające się od podkreślenia to nazwy zmiennych; ponieważ, jak już o tym była mowa, nie dysponujemy pełną informacją słownikową, często zmienna występuje np. na pozycji aspektu.

Najbardziej istotnym odstępstwem od oryginalnej gramatyki jest inna reprezentacja elipsy frazy wymaganej. W systemie AMOS wymagania są reprezentowane w postaci pary: realizacja wymagania i wymaganie. W przykładzie 8 w regule WE26 widzimy wymaganie mianownikowego o realizacji pustej, co jest zapisane jako `nic/mian`. Z zapisem takim mamy do czynienia tylko wtedy, kiedy wymagania czasownikowe zostały pobrane ze słownika, który obecnie jest niewielki. W pozostałych przypadkach odnotowane są tylko te wymagania, które rzeczywiście występują w konkretnym zdaniu — we wspomnianym przykładzie widać to w regule N_FCZP4, gdzie czasownikowi **przyjechać** system przypisał trzy wymagania puste, ponieważ żadne z jego prawdziwych wymagań nie zostało zrealizowane w tym konkretnym zdaniu.

Rodzaj oznaczany jest zgodnie z notacją wprowadzoną w książce [4]: rodzaj męski `m`, żeński `f`, nijaki `ne`, *plurale tantum* `p`. Jeśli z kontekstu można uzyskać dokładniejsze informacje o rodzaju, to jest to odpowiednio zapisane, np. równy mianownikowi biernik liczby mnogiej rzeczownika *pieniądz* w przykładach 2 i 7 ma oznaczenie rodzaju `m23`, co oznacza rodzaj męskozwierzęcy lub męskorzeczowy.

W przykładzie 4 widzimy, jak jeden wyraz alfabetyczny może być rozłożony na dwa wyrazy grafemiczne; niesamodzielność wyrazów grafemicznych jest zaznaczona występującym między nimi — a konkretnie przed drugim z nich — znakiem plus; przypominam, że stosujemy tutaj terminologię zgodną z książką [4].

Znajdujące się na wydrukach czasy analizy odnoszą się do komputera PC z procesorem 486DX i zegarem 66MHz oraz pamięcią operacyjną 8MB. Nie należy jednak do nich przywiązywać nadmiernej wagi, ponieważ są one bardzo niestabilne — drobne zmiany w regułach mogą niekiedy znacznie zwiększyć lub zmniejszyć czas analizy.

Ja ⟨zostawszy⟩.
Porażka analizy
Czas analizy 45.439 s.

wypowiedzenie	[w1]
zr(_1,_2,_3,_4,_5,_6,_7,"ni",_8)	[_26/2]
zsz(_1,_2,_3,_4,_5,_6,_9,"ni",_10)	[s2_4_6_8_10/2]
zp(_1,_2,_3,_4,_5,_6,_11,"ni",_10)	[_27/2]
ze(_1,_2,_3,_4,_5,_6,_12,_13,_14,_11,"ni",_10,_15)	[e16/7]
fl(ob,_2,_3,_5,_6,_11,"ni",_10)	[lu(_16)]
fl1(_2,_3,_5,_6,_11,"ni","np")	[lu10]
przec(nk)	[int12/12]
fwe(psu,_17,prze,_18,_19,1,_20,_21,_22,_23,_11,"ni",_24)	[we3_5_7_9_11_13-19/2]
fw(mian,_23,_17,prze,_19,1,_11,"ni",_25)	[_28/1]
fw1(mian,_23,_17,prze,_19,1,_11,"ni","np")	[wy8-11/5]
fno(mian,_19,1,_11,"ni","np","os")	[no1]
knodop(mian,_19,1,_11,"ni","np","os")	[no5]
knopm(mian,_19,1,_11,"ni","np","os")	[no12]
knoatr(mian,_19,1,_11,"ni","np","os")	[no19/3]
knoink(mian,_19,1,_11,"ni","np","os")	[no40]
knom(mian,_19,1,_11,"np","os")	[no45]
zaimos(rzecz,mian,_19,1)	[n_zo1]
zaimrzecz(mian,_19,zaimos(1))	[jel5b]
Ja : ja	

Dał | im | pieniądze | ojciec.
Analiza zakończona sukcesem
Czas analizy 1.154 s.

wypowiedzenie [w1]
- zr(os,_1,prze,ozn,m/poj,3,tak,"ni","np") [r1/2]
  - zsz(os,_1,prze,ozn,m/poj,3,tak,"ni","np") [s1/1]
    - zj(os,_1,prze,ozn,m/poj,3,tak,"ni","np",przec) [j1/1]
      - zp(os,_1,prze,ozn,m/poj,3,tak,"ni","np") [p1/2]
        - ze(os,_1,prze,ozn,m/poj,3,cel,bier,mian,tak,"ni","np",_2) [e3_10_13_6/4]
          - ff(os,_1,prze,ozn,m/poj,3,cel/cel,bier/bier,mian/mian,_3,tak,"ni","np",_2) [fi1]
            - ff1(os,_1,prze,ozn,m/poj,3,cel/cel,bier/bier,mian/mian,_3,tak,"ni","np",_2) [fi4]
              - fwe(p,os,_1,prze,ozn,m/poj,3,cel/cel,bier/bier,mian/mian,_3,tak,"ni","np") [we1/1]
                - kweneg(os,_1,prze,ozn,m/poj,3,cel/cel,bier/bier,mian/mian,_3,tak,"ni","np") [we23]
                  - kweink(os,_1,prze,ozn,m/poj,3,cel/cel,bier/bier,mian/mian,_3,"ni","np") [we26]
                    - kwer(os,_1,prze,ozn,m/poj,3,cel/cel,bier/bier,mian/mian,_3,"np") [we27-29]
                      - kwer1(os,_1,prze,ozn,m/poj,3,mian,cel,bier,_3,"np") [we30-35]
                        - formaczas(os,_1,prze,ozn,m/poj,3,mian,cel,bier,_3,"np") [n_fcz2]
                          - formaczasp(os,_1,prze,ozn,m/poj,3,mian,cel,bier,_3,"np",n) [n_fczp4]
                            - **Dał** : dać
                    - fl(pu,_1,prze,m/poj,3,tak,"ni","np") [lu14]
              - fw(cel,_3,_1,prze,m/poj,3,tak,"ni","np") [wy1/1]
                - fw1(cel,_3,_1,prze,m/poj,3,tak,"ni","np") [wy8-11/5]
                  - fno(cel,_4/mno,3,tak,"ni","np","os") [no1]
                    - knodop(cel,_4/mno,3,tak,"ni","np","os") [no5]
                      - knopm(cel,_4/mno,3,tak,"ni","np","os") [no12]
                        - knoatr(cel,_4/mno,3,tak,"ni","np","os") [no19/3]
                          - knoink(cel,_4/mno,3,tak,"ni","np","os") [no40]
                            - knom(cel,_4/mno,3,tak,"np","os") [no45]
                              - zaimos(rzecz,cel,_4/mno,3) [n_zo1]
                                - zaimrzecz(cel,_4/mno,zaimos(3)) [jel5]
                                  - **im** : on
              - fw(bier,_3,_1,prze,m/poj,3,tak,"ni","np") [wy1/1]
                - fw1(bier,_3,_1,prze,m/poj,3,tak,"ni","np") [wy8-11/5]
                  - fno(bier,m23/mno,3,_5,"ni","np","rzecz") [no1]
                    - knodop(bier,m23/mno,3,_5,"ni","np","rzecz") [no5]
                      - knopm(bier,m23/mno,3,_5,"ni","np","rzecz") [no12]
                        - knoatr(bier,m23/mno,3,_5,"ni","np","rzecz") [no19/3]
                          - knoink(bier,m23/mno,3,_5,"ni","np","rzecz") [no40]
                            - knom(bier,m23/mno,3,_5,"np","rzecz") [no46]
                              - formarzecz(bier,m23/mno) [n_forz1c]
                                - **pieniądze** : pieniądz
              - fw(mian,_3,_1,prze,m/poj,3,tak,"ni","np") [wy1/1]
                - fw1(mian,_3,_1,prze,m/poj,3,tak,"ni","np") [wy8-11/5]
                  - fno(mian,m/poj,3,tak,"ni","np","rzecz") [no1]
                    - knodop(mian,m/poj,3,tak,"ni","np","rzecz") [no5]
                      - knopm(mian,m/poj,3,tak,"ni","np","rzecz") [no12]
                        - knoatr(mian,m/poj,3,tak,"ni","np","rzecz") [no19/3]
                          - knoink(mian,m/poj,3,tak,"ni","np","rzecz") [no40]
                            - knom(mian,m/poj,3,tak,"np","rzecz") [no46]
                              - formarzecz(mian,m/poj) [n_forz1]
                                - **ojciec** : ojciec
  - znakkonca("np") [int2]
    - .

Dał | **im** | **pieniądze** | **ojciec.**
Analiza zakończona sukcesem
Czas analizy 1.098 s.

```
wypowiedzenie                                                                                                    [w1]
  zr(os,_1,prze,ozn,m/poj,3,tak,"ni","np")                                                                       [r1/2]
    ze(os,_1,prze,ozn,m/poj,3,cel,bier,mian,tak,"ni","np",_2)                                                    [e3_10_13_6/4]
      ff(os,_1,prze,ozn,m/poj,3,cel/cel,bier/bier,mian/mian,_3,tak,"ni","np",_2)                                 [fi1]
        kweneg(os,_1,prze,ozn,m/poj,3,cel/cel,bier/bier,mian/mian,_3,tak,"ni","np")                              [we23]
          kweink(os,_1,prze,ozn,m/poj,3,cel/cel,bier/bier,mian/mian,_3,"ni","np")                                [we26]
            formaczasp(os,_1,prze,ozn,m/poj,3,mian,cel,bier,_3,"np",n)                                           [n_fczp4]
              Dał : dać
            fl(pu,_1,prze,m/poj,3,tak,"ni","np")                                                             [lu14]
        fw(cel,_3,_1,prze,m/poj,3,tak,"ni","np")                                                             [wy1/1]
          zaimrzecz(cel,_4/mno,zaimos(3))                                                                    [jel5]
            im : on
        fw(bier,_3,_1,prze,m/poj,3,tak,"ni","np")                                                            [wy1/1]
          formarzecz(bier,m23/mno)                                                                           [n_forz1c]
            pieniądze : pieniądz
        fw(mian,_3,_1,prze,m/poj,3,tak,"ni","np")                                                            [wy1/1]
          formarzecz(mian,m/poj)                                                                             [n_forz1]
            ojciec : ojciec
  znakkonca("np")                                                                                            [int2]
    .
```

Czyżbyś poszedł?
Analiza zakończona sukcesem
Czas analizy 5.165 s.

```
wypowiedzenie                                                                                                    [w1]
  zr(os,_1,prze,ozn,m/poj,2,tak,"ni","p")                                                                        [r1/2]
    ze(os,_1,prze,ozn,m/poj,2,_2,_3,_4,tak,"ni","p",_5)                                                          [e14/2]
      pyt("czyżby","ni")                                                                                         [par2]
        partykula("czyżby")                                                                                      [jel2]
          Czyżby : czyżby
        zr(os,_1,prze,ozn,m/poj,2,tak,"ni","czyżby")                                                             [r1/2]
          ze(os,_1,prze,ozn,m/poj,2,nic,nic,nic,tak,"ni","czyżby",_6)                                  [e1_7_8_11_4/5]
            fl(ob,_1,prze,m/poj,2,tak,"ni","czyżby")                                                         [lu(1)]
              morfagl("ś",m/poj,2)                                                                           [jel1]
                +ś : ś
              ff(os,_1,prze,ozn,m/poj,3,nic/nic,nic/nic,nic/nic,_7,tak,"ni","czyżby",_6)                     [fi1]
                kweneg(os,_1,prze,ozn,m/poj,3,nic/nic,nic/nic,nic/nic,_7,tak,"ni","czyżby")                  [we23]
                  kweink(os,_1,prze,ozn,m/poj,3,nic/nic,nic/nic,nic/nic,_7,"ni","czyżby")                    [we26]
                    formaczasp(os,_1,prze,ozn,m/poj,3,nic,nic,nic,_7,"czyżby",n)                          [n_fczp4]
                      poszedł : pójść
                    fl(pu,_1,prze,m/poj,3,tak,"ni","np")                                                     [lu14]
              fw(nic,_7,_1,prze,m/poj,2,tak,"ni","np")                                                       [wy5/2]
              fw(nic,_7,_1,prze,m/poj,2,tak,"ni","np")                                                       [wy5/2]
              fw(nic,_7,_1,prze,m/poj,2,tak,"ni","np")                                                       [wy5/2]
  znakkonca("p")                                                                                             [int1]
    ?
```

Ja zostanę, on przyjdzie.
Analiza zakończona sukcesem
Czas analizy 19.726 s.

```
wypowiedzenie                                                                          [w1]
 zr(os,dk,przy,ozn,_1/poj,1,_2,"ni","np")                                            [r1/2]
  zj(os,dk,przy,ozn,_1/poj,1,_2,"ni","np",przec)                                  [_4129/1]
   zp(os,dk,przy,ozn,_1/poj,1,tak,"ni","np")                                         [p1/2]
    ze(os,dk,przy,ozn,_1/poj,1,mian,nic,nic,tak,"ni","np",_3)                 [e2_9_12_5/3]
     fw(mian,_4,dk,przy,_1/poj,1,tak,"ni","np")                                     [wy1/1]
      zaimrzecz(mian,_1/poj,zaimos(1))                                              [jel5b]
       Ja : ja
     ff(os,dk,przy,ozn,_1/poj,1,mian/mian,nic/nic,nic/nic,_4,tak,"ni","np",_3)        [fi1]
      kweneg(os,dk,przy,ozn,_1/poj,1,mian/mian,nic/nic,nic/nic,_4,tak,"ni","np")     [we23]
       kweink(os,dk,przy,ozn,_1/poj,1,mian/mian,nic/nic,nic/nic,_4,"ni","np")        [we26]
        formaczasp(os,dk,przy,ozn,_1/poj,1,mian,nic,nic,_4,"np",n)                [n_fczp2]
         zostanę : zostać
       fl(pu,dk,przy,_1/poj,1,tak,"ni","np")                                         [lu14]
     fw(nic,_4,dk,przy,_1/poj,1,tak,"ni","np")                                      [wy5/2]
     fw(nic,_4,dk,przy,_1/poj,1,tak,"ni","np")                                      [wy5/2]
   spoj(sz,przec,"ni")                                                              [spoj2]
    przec(nk)                                                                    [int13/7]
     ,
   zp(os,dk,przy,ozn,m/poj,3,tak,"ni","np")                                          [p1/2]
    ze(os,dk,przy,ozn,m/poj,3,mian,nic,nic,tak,"ni","np",_5)                  [e2_9_12_5/3]
     fw(mian,bier,dk,przy,m/poj,3,tak,"ni","np")                                    [wy1/1]
      zaimrzecz(mian,m/poj,zaimos(3))                                                [jel5]
       on : on
     ff(os,dk,przy,ozn,m/poj,3,mian/mian,nic/nic,nic/nic,bier,tak,"ni","np",_5)       [fi1]
      kweneg(os,dk,przy,ozn,m/poj,3,mian/mian,nic/nic,nic/nic,bier,tak,"ni","np")    [we23]
       kweink(os,dk,przy,ozn,m/poj,3,mian/mian,nic/nic,nic/nic,bier,"ni","np")       [we26]
        formaczasp(os,dk,przy,ozn,m/poj,3,mian,nic,nic,bier,"np",n)               [n_fczp2]
         przyjdzie : przyjść
       fl(pu,dk,przy,m/poj,3,tak,"ni","np")                                          [lu14]
     fw(nic,bier,dk,przy,m/poj,3,tak,"ni","np")                                     [wy5/2]
     fw(nic,bier,dk,przy,m/poj,3,tak,"ni","np")                                     [wy5/2]
 znakkonca("np")                                                                     [int2]
  .
```

Matka ⟨dała | komu⟩ pieniądze?
Analiza zakończona sukcesem
Czas analizy 0.769 s.

wypowiedzenie [w1]
- zr(os,_1,prze,ozn,f/poj,3,tak,"ni","p") [r1/2]
  - ze(os,_1,prze,ozn,f/poj,3,mian,cel,bier,tak,"ni","p",_2) [e2_9_12_5/3]
    - fw(mian,_3,_1,prze,f/poj,3,tak,"ni","np") [wy1/1]
      - formarzecz(mian,f/poj) [n_forz1]
        - **Matka** : matka
    - ff(os,_1,prze,ozn,f/poj,3,mian/mian,cel/cel,bier/bier,_3,tak,"ni","p",_2) [fi1]
      - kweneg(os,_1,prze,ozn,f/poj,3,mian/mian,cel/cel,bier/bier,_3,tak,"ni","p") [we23]
        - kweink(os,_1,prze,ozn,f/poj,3,mian/mian,cel/cel,bier/bier,_3,"ni","p") [we26]
          - formaczasp(os,_1,prze,ozn,f/poj,3,mian,cel,bier,_3,"p",n) [n_fczp4]
            - **dała** : dać
          - fl(pu,_1,prze,f/poj,3,tak,"ni","np") [lu14]
    - fw(cel,_3,_1,prze,f/poj,3,tak,"ni","p") [wy1/1]
      - zaimrzecz(cel,m1/poj,zaimpyt("kto")) [jel5a]
        - **komu** : kto
    - fw(bier,_3,_1,prze,f/poj,3,tak,"ni","np") [wy1/1]
      - formarzecz(bier,m23/mno) [n_forz1c]
        - **pieniądze** : pieniądz
  - znakkonca("p") [int1]
    - **?**

⟨To, że przyjechała⟩, pamiętano.
Analiza zakończona sukcesem
Czas analizy 91.153 s.

```
wypowiedzenie                                                              [w1]
 zr(bos,nd,prze,ozn,_1,3,tak,"ni","np")                                    [r1/2]
  ze(bos,nd,prze,ozn,_1,3,"że",nic,mian,tak,"ni","np",_2)                  [e2_9_12_5/3]
   fw("że",bier,nd,prze,_1,3,tak,"ni","np")                                [wy1/1]
    fw1("że",bier,nd,prze,_1,3,tak,"ni","np")                              [wy17-19/4]
     fzd("że",bier,_3,prze,ozn,tak,"ni")                                   [zd1]
      fzdkor("że",bier,_3,prze,ozn,tak,"ni")                               [zd38]
       kor(bier,"ni")                                                      [kor2]
        zaimrzecz(bier,ne/poj,kor)                                         [jel5a]
         To : to
       przec(nk)                                                           [int13/7]
        ,
       fzde("że",_3,prze,ozn,tak,"ni")                                     [zd42]
        spoj(po,"że","ni")                                                 [spoj2]
         spojnik("że")                                                     [jel4]
          że : że
        zr(os,_3,prze,ozn,f/poj,3,tak,"ni","że")                           [r1/2]
         ze(os,_3,prze,ozn,f/poj,3,nic,nic,nic,tak,"ni","że",_4)           [e3_10_13_6/4]
          ff(os,_3,prze,ozn,f/poj,3,nic/nic,nic/nic,nic/nic,_5,tak,"ni","że",_4)      [fi1]
           kweneg(os,_3,prze,ozn,f/poj,3,nic/nic,nic/nic,nic/nic,_5,tak,"ni","że")   [we23]
            kweink(os,_3,prze,ozn,f/poj,3,nic/nic,nic/nic,nic/nic,_5,"ni","że")      [we26]
             formaczasp(os,_3,prze,ozn,f/poj,3,nic,nic,nic,_5,"że",n)      [n_fczp4]
              przyjechała : przyjechać
            fl(pu,_3,prze,f/poj,3,tak,"ni","np")                           [lu14]
          fw(nic,_5,_3,prze,f/poj,3,tak,"ni","np")                         [wy5/2]
          fw(nic,_5,_3,prze,f/poj,3,tak,"ni","np")                         [wy5/2]
          fw(nic,_5,_3,prze,f/poj,3,tak,"ni","np")                         [wy5/2]
     przec(nk)                                                             [int13/7]
      ,
   ff(bos,nd,prze,ozn,_1,3,"że"/"że",nic/nic,nic/mian,bier,tak,"ni","np",_2)  [fi1]
    kweneg(bos,nd,prze,ozn,_1,3,"że"/"że",nic/nic,nic/mian,bier,tak,"ni","np")  [we23]
     kweink(bos,nd,prze,ozn,_1,3,"że"/"że",nic/nic,nic/mian,bier,"ni","np")    [we26]
      formaczasp(bos,nd,prze,ozn,_1,3,mian,"że",nic,bier,"np",n)           [n_fczp6]
       pamiętano : pamiętać
     fl(pu,nd,prze,_1,3,tak,"ni","np")                                     [lu14]
   fw(nic,bier,nd,prze,_1,3,tak,"ni","np")                                 [wy5/2]
   fw(nic,bier,nd,prze,_1,3,tak,"ni","np")                                 [wy5/2]
 znakkonca("np")                                                           [int2]
  .
```

Wiem⟨, że on przyjdzie⟩.
Analiza zakończona sukcesem
Czas analizy 4.780 s.

```
wypowiedzenie                                                                              [w1]
 zr(os,nd,ter,ozn,_1/poj,1,tak,"ni","np")                                                  [r1/2]
  ze(os,nd,ter,ozn,_1/poj,1,"że",nic,nic,tak,"ni","np",_2)                                 [e3_10_13_6/4]
   ff(os,nd,ter,ozn,_1/poj,1,"że"/"że",nic/nic,nic/nic,"nk",tak,"ni","np",_2)              [fi1]
    kweneg(os,nd,ter,ozn,_1/poj,1,"że"/"że",nic/nic,nic/nic,"nk",tak,"ni","np")            [we23]
     kweink(os,nd,ter,ozn,_1/poj,1,"że"/"że",nic/nic,nic/nic,"nk","ni","np")               [we26]
      formaczasp(os,nd,ter,ozn,_1/poj,1,"że",nic,nic,"nk","np",n)                          [n_fczp1]
       Wiem : wiedzieć
     fl(pu,nd,ter,_1/poj,1,tak,"ni","np")                                                  [lu14]
   fw("że","nk",nd,ter,_1/poj,1,tak,"ni","np")                                             [wy1/1]
    fw1("że","nk",nd,ter,_1/poj,1,tak,"ni","np")                                           [wy17-19/4]
     fzd("że","nk",dk,przy,ozn,tak,"ni")                                                   [zd1]
      fzdkor("że","nk",dk,przy,ozn,tak,"ni")                                               [zd39]
       przecsp(nk)                                                                         [int6]
        przec(nk)                                                                          [int13/7]
         ,
       fzde("że",dk,przy,ozn,tak,"ni")                                                     [zd42]
        spoj(po,"że","ni")                                                                 [spoj2]
         spojnik("że")                                                                     [jel4]
          że : że
        zr(os,dk,przy,ozn,m/poj,3,tak,"ni","że")                                           [r1/2]
         ze(os,dk,przy,ozn,m/poj,3,mian,nic,nic,tak,"ni","że",_3)                          [e2_9_12_5/3]
          fw(mian,bier,dk,przy,m/poj,3,tak,"ni","np")                                      [wy1/1]
           zaimrzecz(mian,m/poj,zaimos(3))                                                 [jel5]
            on : on
          ff(os,dk,przy,ozn,m/poj,3,mian/mian,nic/nic,nic/nic,bier,tak,"ni","że",_3)       [fi1]
           kweneg(os,dk,przy,ozn,m/poj,3,mian/mian,nic/nic,nic/nic,bier,tak,"ni","że")     [we23]
            kweink(os,dk,przy,ozn,m/poj,3,mian/mian,nic/nic,nic/nic,bier,"ni","że")        [we26]
             formaczasp(os,dk,przy,ozn,m/poj,3,mian,nic,nic,bier,"że",n)                   [n_fczp2]
              przyjdzie : przyjść
            fl(pu,dk,przy,m/poj,3,tak,"ni","np")                                           [lu14]
          fw(nic,bier,dk,przy,m/poj,3,tak,"ni","np")                                       [wy5/2]
          fw(nic,bier,dk,przy,m/poj,3,tak,"ni","np")                                       [wy5/2]
     przec(k)                                                                              [int8/10]
   fw(nic,"nk",nd,ter,_1/poj,1,tak,"ni","np")                                              [wy5/2]
   fw(nic,"nk",nd,ter,_1/poj,1,tak,"ni","np")                                              [wy5/2]
 znakkonca("np")                                                                           [int2]
  .
```

RAPORTY
INSTYTUTU INFORMATYKI UW
od 1994 roku

200. S. van Bakel, L.Liquori, S.Ronchi, P.Urzyczyn: *Comparing Cubes of Typed and Type Assignment Systems*

201. T.Nowicki: *A dynamical model of behavioural specifications*

202. W.M.Turski: *Behavioural specifications*

203. P.Urzyczyn: *Positive Recursive Type Assignment*

204. J.Tiuryn: *Equational Axiomatization of Bicoercibility for Polymorphic Types*

205. M.Iglewski, M.Kubica, J.Madey: *Trace Specifications of Non-deterministic Multi - object Modules*

206. L.Czaja: *Lattice of cause-effect structures and their set-theoretic representation*

207. L.Czaja: *Decomposition of cause-effect structures*

208. FMTA'95 Formal Specifications: Foundations, Methods, Tools and Applications

209. ANNUAL REPORT 1995

210. L.Czaja: *Behaviour of cause-effect structures (Properties of four semantics)*

211. A.Szałas: *Relational Calculus with Recursion Quantifiers*

212. L.Czaja: *Representing CSP-like system as cause-effect structures*

213. J.Tiuryn, M.Wand: *Untyped Lambda-Calculus with Input-Output*

214. W.Ogryczak: *On the Lexicographic Minimax Approach to Location Problems*

215. V.Pratt, J.Tiuryn: *Satisfiability of Inequalities in a Poset*

216. D.Niwiński: *Fixed points characterization of infinite behaviour of finite state systems*

217. K.Stencel: *Refined Simulation Techniques for the Trace Assertion Method*

218. M.Iglewski, M.Kubica, J.Madey, J.Mincer-Daszkiewicz, K.Stencel: *The Fun-Project: From Requirements Specification to Program Presentation*

219. J.Tiuryn, P.Urzyczyn: *The Subtyping Problem for Second-Order Types is Udecidable*

220. M.Benke: *Some Complexity Bounds for Subtype Inequalities*

221. G.Grudziński: *Intersection Types Spell Polymorphic Invariance for Strictness Analysis*

222. J.Tiuryn: *A Sequent Calculus for Subtyping Polymorphic Types*

223. W.Ogryczak: *Equitable Multiple Criteria Programming*

224. L.Czaja: *Cause-effect structure processes and their link with Mazurkiewicz's traces*

225. M.Jurdziński, M.Konarski, A.Schubert: *The EML Kit Version 1*

226. K.Szafran: *Analizator morfologiczny SAM-95 – opis użytkowy*