A.Ü. TÖMER Dil Dergisi, S.30, Nisan 1995

Bilgisayar Destekli Sözdizimi Çalışması ve PROLOG

Yrd.Doç.Dr. Mustafa ÇAKIR*

Giriş

Bilgisayar, Christopher Strachy tarafından 'simgeleri işlemek için kullanılan bir makine' olarak tanımlanmaktadır (Aktaran, Türker 1994:137). Görünüşte yalın olan bu terim, gerçekte çağdaş bir sayısal bilgisayarın yapabileceği bütün görevleri kapsayan genişliktedir. Bu tanıma göre, bilgisayara girdi denilen bir simgeler dizisi verilir; bunlar önceden makineye yerleştirilen ve değişmez kurallar bütünü olan PROGRAM'a göre işlenir ve sonunda çıktı denilen bir başka simgeler dizisi üretilir. PROLOG-Program dili de bu tür simgelerle işleyen bir sistemdir.

PROLOG, "PROgramming in LOGic" sözünün kısaltılmış biçimidir ve ilk olarak 70'li yılların başında Marsilya Üniversitesi'nde ve Edinburg kentlerinde ortaya çıkmıştır. Daha sonraları ise Londra'da Imperial College'da geliştirilmiş ve geniş kitleler tarafından tanınması bu yolla olmuştur. PROLOG doğrudan mantığa dayalı *bildirgesel* (deklerativ) bir program dilidir. Burada kullanılan deklerativ terimi, *süreçsel* (prozedural) terimine karşılık olarak kullanılmaktadır. Hem sıradan bir bilgisayar dili hem de gelişmiş bütün üst düzeydeki bilgisayar program dilleri bir süreçle ilgili dillerdir. Süreçsel bir dilde, verilen bir problemi çözmek için (bir yemek tarifinde olduğu gibi) aşama aşama çalışılır. Program bitirilip çalıştırılmaya başlandığında ise önceden yazılan öngörülere uygun olan sıraya göre çalışır. Bildirgesel bir program dilinde ise, **olgular** ve **kurallar**'dan yola çıkılır (Bkz. Kraml & Schrodt 1990: 98). Yazılan programı çalıştırmak için bir **öngörü** (Hipotezden) oluşturulur ve bilgisayar bu öngörüyü kendisine daha önceden verilmiş olan olgu ve kurallar yardımıyle sınar. Buna günlük konuşulan dile yakın olması bakımından aşağıdaki "PROLOG-Programı" örnek olarak verilebilir (açıklama yapılan satırlar /*...*/ ve % ile gösterilmiştir):

İstanbul bir şehirdir.
Bolu bir şehirdir.
Eskişehir bir şehirdir.
Londra bir şehirdir.

İstanbul'un nüfusu 7,500 000 kişidir.
Bolu'nun nüfusu 70.000 kişidir.
Eskişehir'in nüfusu 700.000 kişidir.

---

[*] Anadolu Üniversitesi Eğitim Fakültesi Öğretim Üyesi

/* buraya kadar "olgular" dan söz edildi.*/

70.000 kişiden daha az nüfusu olan bir kent küçük bir şehirdir.
1.000.000 kişiden daha fazla insan yaşayan bir kent büyük şehirdir.

/*Bu bilgiler iki ayrı "kural"dır.*/
/*Şimdi "öngörü"leri ve buna bağlı olarak kullanılacak program sonuçlarını verelim:*/

Öngörü 1: İstanbul bir büyük şehirdir. (Sonuç: doğru)
Öngörü 2: Eskişehir bir büyük şehirdir. (Sonuç: yanlış)
Öngörü 3: Bolu bir küçük şehirdir. (Sonuç: doğru)
Öngörü 4: Eskişehir bir küçük şehirdir. (Sonuç: yanlış)
Öngörü 5: Eskişehir ne bir büyük şehir ne de bir küçük şehirdir. (Sonuç: doğru)
Öngörü 6: Eskişehir bir şehir değildir. (Sonuç: yanlış)
Öngörü 7: Londra bir büyük şehirdir. (Sonuç: yanlış)

Yukarıda verilen yanıtlar şöyle değerlendirilir: Eğer sonuç **yanlış**sa bundan iki tür sonuç çıkarmak olasıdır. Ya programdaki öngörüler (Öngörü 6'daki gibi) eksik verilmiştir; ya da verilen bilgi işlenememektedir (Öngörü 7'deki gibi).

PROLOG-Programı yukarıda olduğu gibi günlük konuşma dilinde yazılmaz. Bunun için kullanılan özel bir kodlama sistemi vardır (Bkz.: Knaus & Jay 1988: 13; Knaus 1988: 13).

```
şehirdir (istanbul).
şehirdir (eskişehir).
...
nüfusu vardır (eskişehir, 700.000).
nüfusu vardır (bolu, 70.000).
...
_ küçük şehirdir (X) : _ bir _ şehirdir (X),
                       _ nüfusu vardır (X,Y),
                       Y< 70.000
```

Yukarıda, mantığa uygun olarak **olgular** ve **biçimler** oluşturuldu. Verilmek istenen gönderge (Prädikat) (_bir şehirdir, _nüfusu vardır) şeklinde delillerden (Argumente) oluşan bir liste halinde oluşturuldu. Program yazımı sırasında özel adların yazımı konusundaki büyük yazım kuralı burada uygulanmamaktadır. Her türlü ad küçük yazılmaktadır. Kurallar iki bölümden oluşur ve bölümler :- sembolü ile birbirinden ayırılır. Kuralın ilk bölümü sadece tek bir terimi (Term) yani her hangi bir yapıyı içerir; ikinci bölümü ise birden fazlasını içerebilir. Kural yazımına örnek olarak

```
P:- Q, S.
```

verilebilir. Buradaki P, Q ile S doğru ise doğru; yani P, Q ile S kanıtlanabilirse kanıtlanabilir (program başarıyla çalıştırılabilir).

Kurallar yazılırken genellikle değişkenler de kullanılır. Bu değişkenler, program yazımı ilerledikçe bilinmeyen semboller için kullanılacak boşlukları oluşturacaktır. Değişkenler yazılırken ya ilk harfleri büyük yazılır ya da ilk harfin altı çizilir. PROLOG-Programının önemli bir diğer özelliği de oluşturulan listelerdir. Listeler köşeli ayraçlar içine yazılır ve listeyi oluşturan birimler virgül ile birbirinden ayrılırlar. Örneğin, [küçük, çocuk, uyuyor] ifadesi üç elemandan oluşan bir listedir. PROLOG listeleri her zaman birleştirilebilir ya da ayrılabilir.

Olguları öngörülerle ifade etmesi, formal yazım kurallarının betimlenmesinde kullanılabilmesi, PROLOG programının en takdir edilen, beğenilen özelliğidir. Bu özellik sayesinde PROLOG sisteminde bir tümce bir öngörü olarak verilebilir ve bu tümce, programda çalışılan dilbilgisi kuralına göre analiz edilebilir. Bu (İngilizcede teknik bir terimle **Parsing** olarak anılan) analizler, diğer program dillerinde de benzer kurallara bağlı olarak kullanılmaktadır (Bkz. Lenders & Willée 1986:106). Bu analizler, bir bilgisayar dilinin programının gerçekte kullanılabilir bir makine diline çevirilmesindeki ilk aşama olarak kabul edilmektedir.

Prolog ile Sözdizimi

Yazılım dili kurallarının günlük konuşma diline uyarlanabilmesi ve PROLOG-Sisteminin normal konuşma dilinin dilbilgisel açıdan araştırılmasında kullanılması mümkündür. Ancak burada normal yazı dilinin ya da her hangi bir bilgisayar yazılımı için kullanılan dilin kuralları ile sözdizimi için kullanılacak bilgisayar dilinin kuralları arasında bir fark vardır. Bu fark, bir yazılım için mevcut bir kuralın her zaman geçerli olması ve o dille ilgili çeviri programı için hazırlanmış olmasından kaynaklanmaktadır. Sözdizimi analizi yapacak olan programın görevi ise bir dilin yaratılacak olası tümcelerinin dilin kurallarına uygun olup olmadığını araştıracak olmasıdır.

Aynı dili konuşan bireyler, konuşma sırasında ifade edilen tümcelerin dilbilgisi kurallarına uygun olup olmadığına bakmaksızın birbirlerini anlayabilirler; buna karşın kurulan tümcenin dilbilgisi açısından doğruluğunu ya da kullandıkları ifadelerin dilbilgisi kurallarına uygunluğunu test edemezler. Bu tür dilsel analizleri yapmak ise dilbilimcilerin görevidir. Doğal dillerin dilsel analizlerinin bilgisayar yoluyla yapılması görüşü, aslında formal bilgisayar dillerinin tümce analizinde kullanılabileceği görüşüdür. Almanca ya da Türkçede bir tümcenin doğru ya da yanlış kurulduğunu tespit etmemiz mümkün, ama bu görüşlerimizi dayandırabileceğimiz bir kural ve öngörülerin olması gerekmektedir. Bu öngörüleri de (deneysel fizik alanında olduğu gibi) PROLOG yardımıyle işlememiz gerekir. Bu durumda yaptığımız işlem de bir noktada "deneysel dilbilim" çalışması olacaktır. PROLOG yazılımında Parsing kullanıldığından, bu yazım için de geçerli olan bir yeni yazım biçimi ("definite clause grammar notation" veya kısaca DCG) geliştirilmiştir. Böylelikle dilbilgisi kuralları ("Ersetzungsregeln" ya da "rewriting rules" da olduğu gibi) daha kolay bir şekilde yazılım olarak gösterilebilmektedir.

```
s     --> np, vp
np    --> noun.
vp    --> verb.
noun  --> [çocuk].
verb  --> [uyuyor].
```

Böylece, yukarıda görüldüğü şekilde PROLOG ile [çocuk, uyuyor] şeklinde yazılan tümce, kullanılan dilbilgisine göre doğru bir tümce olarak kabul edilebilir. Burada görüldüğü gibi, kullanılan ilk üç kural sözdizimi, kalan ikisi de sözcükbilgisi ile ilişkilidir.

Bu tür bir program, bilgisayarda bir metin gibi yazılır ve bir adla (örn. satz.dec.) kaydedilir sonra da PROLOG sistemine

consult (satz).

komutuyla girilir.

Bu tümcenin analizi için ilerki aşamalarda gerekli komut

phrase (s, [çocuk, uyuyor]).

şeklinde görülecek ve alınacak yanıt da

yes

olacaktır.

PROLOG bize bu aşamada verilen tümcenin nasıl analiz edilmesi gerektiğini değil; doğru analiz edilip edilemediğini verecektir. Tümcenin analizi sırasında bir anlam ifade edecek değişik argümanların kullanıldığı biraz daha karışık bir yazım biçiminde, tümce şu yapısal özelliği (Alm.: Phrasenstrukturbaum in Klammernotation) gösterecektir:

```
s (s(N,V)) --> np (N), vp (V).
np (np(N)) --> noun (N).
vp (vp(V)) --> verb (V).
noun (noun (çocuk)) --> [çocuk].
verb (verb (uyuyor)) --> [uyuyor].
```

Buna ek olarak analizin sonucunu verecek bir parçayı da parse olarak adlandıralım.

```
parse (Satz) : - phrase (s (S), Satz), write (S), nl.
parse (Satz) : -write ('Tümce analizi yapılamıyor.'), nl.
```

Önceden *write* diye tanımlanan birim tümceyi ekrana çıkarmaya ve *nl* ise ekranda yeni bir satır açmaya yarıyor. Oysa burada bir parça iki ayrı kural kullanılarak tanımlandı. Bunun nedeni, ilk kuraldan istenen sonuç alınamazsa PROLOG'da ikinci seçenekten yararlanılsın. Yani ilk kural olan *parse* (...) anlaşılamazsa, sadece metin verebilen ve her zaman başarılı olabilen ikinci kural kullanılsın. Bu durumda yapılan analizin başarılı olamaması durumunda da PROLOG *yes* yanıtı verecektir. Buna göre oluşturulan program aşağıdaki gibi olacaktır:

parse ([çocuk, uyuyor]).

Böyle bir program değişik şekillerde oluşturulabilir. Doğal dillerin analizinde karşılaşılan esas sorun, analizi yapılan dilde analizi yapılan tümcelerin sadece bir bölümünün kullanılabilmesi ve analizi yapılan dilin kullanılan söz dağarcığının yetersiz kalmasıdır. Bunun için de oldukça geniş kapsamlı bir sözlük ile uygun alıntıların yer alacağı bir dağarcık kullanılması gerekir. Bu aşamada sözcüklerin yetersiz kalmasından kaynaklanabilecek program yetersizliğinin önüne geçmek için "kendiliğinden öğrenen" yani ilgili yerde gereken sözcüğü sorabilen ve gerekli sözcüğü belleğine kullanıcı yardımıyle alabilen bir program geliştirilmesi de düşünülebilir.

Tümce analizi yapan programların geliştirilmiş şekilleri dilbilim araştırmaları açısından ilginç gelebilir. Yukarıda verilen parçalar aşama aşama geliştirilebilir ve Türkçenin tümce yapısının betimlenmesinde de kullanılabilir. Ya da örneğin Almancanın tümce yapısına uygulanabilir ve dil öğretiminde bu tür bir programdan yararlanılabilir.

Kolay Bir Parsing-Programı

Yukarıda açıklanmaya çalışılan dilbilgisi programının bir versiyonu Almanca için hazırlanmış ve bilgisayara ASCII-Zeichen olarak kaydedilerek (bazı özellikleri saklı tutularak) tam çıktısı alınmıştır.

consult (...).

Datei (namens ...) laden; die Datei "syntax" enthält z.B. das Programm mitsamt den Syntaxregeln, die Datei "lexikon" die Lexikoneinträge.

phrase (...).

Der Satz ... (als Liste [..., ...] geschrieben) soll syntaktisch analysiert werden.

speichere _lexikon (...).

Die derzeit gerade bekannten Lexikonregeln sollen in der Datei namens ... (z.B. "lexikon") gespeichert
werden.

Bu arada, dilbilgisel bir sınıflandırma için duruma göre pek çok kural ortaya çıkabileceği görülmektedir; yani, "oder"-ilişkisine göre duruma uyan kuralın kullanımı mümkündür.

Buna göre yazılan program aşağıdaki gibi olacaktır:

```
/* Örnek Bir Tümce Analizi Programı*/[1]

parse (L) :-
        assert (firstpass),
        phrase (s (S), L),
        !
        write ('Tümce aşağıdaki gibi analiz edilecektir : ' ),
        nl,
        write (S),
        nl.
parse (L) :-
        retractall (firstpass),
        phrase (s (S), L),
```

---

[1] Burada örnek olarak verilen program, LPA PROLOG Professional, V.2.0 (Logic Programming Associates, London) ile oluşturulmuştur. Bu Edinburgh-Syntax türündeki PROLOG uyarlamasıdır ve piyasada bulunan PROLOG adı altındaki programlarla uyumludur. Programın bundan başka, bu çalışmaya yeni başlayanlar için önerilebilecek PROLOG Public-Domain-Version'u da vardır. Ancak dikkat edilmesi gereken bir konu, bu program Türkiye'nin yanı sıra daha başka ülkelerde de bulunan ve genel PROLOG standartlarına uymayan Turbo-Prolog (Borland International) programından farklı bir program olma özelliğini taşımaktadır. Bu program için, yukarıdaki programda kapsamlı bir değişiklik yapmak gerekmektedir. (Burada anılan bu durumun yeniden test edilmesi için yardımcı olan Almanca Öğretmenliği Programı'nda görevli çalışma arkadaşım **Okt.Abdullah KUZU**'ya teşekkür ederim)

```
        !,
        write ('Tümce aşağıdaki gibi analiz edilecektir: ' ),
        nl,
        write (S),
        nl.

parse (L) :-
        write ('Tümce analiz edilemiyor!'),
        nl.
lex_suche (Word, Class) :-
        def (lex_eintrag),
        lex_eintrag (Word, Class)
        !
lex_suche (Word, Class) :-
        not def (firstpass),
        write ('Gehört " ' ),
        write (Word),
        write (' " der Wortklasse " ' ),
        write (Class),
        write (' " an? ' )
        nl,
        get (Ans),
        % put (Ans),
        on (Ans, [74, 106])
        assert (lex_eintrag (Word, Class)),
        lex_suche (Word, Class).
% Lexikonregeln werden zur Laufzeit dynamisch erzeugt; anschließend
% eine Regel zum Speichern des Lexikons; geladen wird mit consult (Name)
speichere_lexikon (Name) :- save (Name, lex_eintrag).

% sözdizimine ilişkin kurallar:

s (s (N, V))          --> np (N), vp (V).                    % Satz
np (np (D, N))        --> det (D), ng (N).                   % Nominalphrase
np (np (N))           --> ng (N).                            % Nominalphrase
np (np (N))           --> noun (N).                          % Nominalgruppe
np (np (A, N))        --> adj (A), ng (N).                   % Nominalgruppe
vp (vp (I))           --> iv (I).                            % Verbalphrase
vp (vp (T, N)         --> tv (T), np (N).                    % Verbalphrase
det (det (D))         --> [D], {lex_suche (D, det)}.         % Determinativ
noun (noun (N))       --> [N], {lex_suche (N, noun)}.        % Nomen
adj (adj (A))         --> [A], {lex_suche (A, adj)}.         % Adjektiv
iv (iv (V))           --> [V], {lex_suche (V, iv)}.          % intransitives Verb
tv (tv (V))           --> [V], {lex_suche (V, tv)}.          % transitives Verb
```

Almanca Sözdizimi İçin Kolay Bir Örnek Program

Burada birbirini takip eden bir dizi argümandan oluşan bir yapı oluşturuldu. Bunun için tümce ile başlandı. Tümce terimi, çeşitli kuram ve anlayışlara göre değişken içerikler kapsamaktadır. Bu bakımdan söz konusu terimi tek bir tanıma indirgeme olanağı bulunmamaktadır. Verilen tanımlardan birinin yansıttığı anlayış her yönden eleştirilmekte, ikincisi öğelerle tümce arasındaki ilişkilere yer vermediği için yetersiz görülmektedir (Bkz. Vardar 1980: 145). Yine günlük bilgi dağarcığına en yakın tanımı alalım. Buna göre tümce, geleneksel dilbilgisinde anlam açısından eksiksiz sayılan, bir kesinti ya da durakla sınırlanan söz anlamında kullanılmaktadır. Bu bağlamda yan tümceleri yukarıdaki tanıma uymadığı ve bir istek, duygu ya da düşünceyi tam olarak yansıtamadığı için eksiksiz bir tümce olarak kabul etmemek gerekmektedir. Hangi durumda olursa olsun, tümce yukarda sayılan sınıflandırmadaki bir gruba girmektedir. Yukarıda verilen Türkçe tümcede (Çocuk uyuyor.) olduğu gibi, bir **özne** (Subjekt) ve bir **yüklem**den (Prädikat) oluşan Almanca bir örnek alalım.

Otto schläft.

Bu tümceyi PSG (Phrasenstrukturgrammatik) 'ye göre kısaca tanımlamak gerekirse,

S ----> NP+VP
NP ----> N
VP ----> V

Bu tümce formülünü, ***betimsel dilbilgisi*** yaklaşımına göre 'Tümce, NP ile VP'den oluşmaktadır.' diye okumak mümkündür. Ayrıca bilgisayar için yazılım hazırlanırken ok işareti yerine "ersetze [das Symbol X] durch [das Symbol Y]" (... sembolünün yerine ... sembolünü koy) şeklinde bir ifade de kullanılır. Bu yaklaşıma da ***üretici dilbilgisi*** yaklaşımı denilmektedir. Burada verilen tümce yukarıdaki formülde oldukça yalın bir şekilde görülmesine rağmen, bilgisayar dili söz konusu olunca yaklaşık olarak aşağıdaki şekilde ifade edilir:

$[_{S}[_{NP}[_{N}[\text{Otto}]][_{VP}[_{V}[\text{schläft}]]].$

Buna uygun yazılmış PROLOG dili sözdizimi kuralı ise daha yalın:

```
s (s (N, V))          --> np (N), vp (V).
np (np (N))           --> noun (N).
vp (vp (V))           --> verb (V).
```

Bu yapı PROLOG dilinde ise yaklaşık olarak şöyle ifade edilir:

```
s (np (noun(otto) ), vp (verb (schläft)))
```

Bu yeni biçim, tam olarak ilk verilen örnek kadar anlaşılır olmasa da doğrudur. Ama ilerideki aşamalarında yukarıda görüldüğünden de karmaşık bir şekil alacaktır. Bir NP Almancada bir tanımlık (Artikel) ile birlikte de kullanılabilir; örn.: *der Student* vb. gibi. Bu durumda belirli tanımlık ile belirsiz tanımlık kullanımını ayırdetmek gerekir. Bundan başka, hiç bir ad bulunmaksızın, tanımlık kendi başına da kullanılabilir (Der hat mir ein Strafmandat verpaßt). Bundan başka, tanımlıklar, adla birlikte ya da adın yerine de kullanılabilirler. Bu durumda yeni bir terim kullanmakta yarar vardır. Bunun için **Determinativ** (kısaca **Det**) denilebilir. Kural olarak da NP---> Det + N ya da tanımlık sürekli kullanılmak zorunda olmadığından, NP----> (Det) N. PROLOG programında bu iki durumu belirtmek için iki kural vardır. Bunlar:

np (np (N))   ----> noun (N).
np (np (D, N)) ----> det (D), np (N).

Der Student (öğrenci) sözcüğü de belli durum ya da özelliklere sahip olabilir. Bunlar, sıfatlar yardımyla açıklanabilir. *Der müde Student* (yorgun öğrenci) ya da sadece *müde Studenten* gibi. Bu bağlamda, "Sıfatı söz dizimi kombinasyonunda nereye yerleştireceğiz?" ya da "Çoğul kullanıldığında tanımlık neden kullanılmadı?" türünde sorularla karşılaşılabilir. Bu sorulara yanıt verebilmek için, bu kategorilerden birisini, NP grubuna, Adjektiv + Nomen biçiminde yazmak olasıdır. Çünkü bu grup **Nominalgruppe** (ad grubu) olarak geçmektedir (kısaca NG). Ayrıca, PROLOG için de NG den N yapabilmek için yeni bir formüle gereksinim duyulmaktadır. Bu:

ng (ng (N)) -----> noun (N).

veya bir Adjektiv (sıfat) ve bir N den oluşan

ng (ng (A,N)) -----> adj (A), ng (N).

şeklinde yazılır. Böylelikle ilk başta sahip olunan NP, artık değişik bir şekilde, NG yi de içerebilecek şekilde yeniden yazılmış oldu. Bu durum da yeni bir kural yazımını gerektiriyor. Bu kural:

np (np(N)) ----> ng (N).

biçimindedir. Bir tanımlık + ad durumunda da NG durumu söz konusudur. Bu duruma uygun NP kuralı da aşağıdaki gibi olacaktır:

np (np (D,N)) ----> det (D), ng (N).

Yazılan bu sözdizimi kuralı aşağıdaki tümceleri tanımlayabilir:

Studenten schlafen.
s (np (ng (noun (studenten))), vp (verb (schlafen)))

Der Student schläft.
s (np (det (der), ng (noun (student))), vp (verb (schläft)))

Müde Studenten schlafen.
s (np (ng (adj (müde), ng (noun (student)))), vp (verb (schlafen)))

Der müde Student schläft.
s (np (det (der), ng (adj (müde), ng (noun (polizist)))), vp (verb (schläft)))

Öğrenciler yorgun olabilirler ya da olmayabilirler. Arada içlerinden yaramazlık yapanları da çıkabilir ve söz gelimi kopya çekmeye yeltenenler olabilir. Bunlardan kopyada yakalananlar da olabilir kuşkuşusuz. Bunun yanısıra olayları anlatmak için kullanılan eylemler geçişli bir fiil de olabilirler. Geçişli fiiller, zorunlu olarak bir NP (tümleç) ile kullanılmak durumundadırlar. Bağıl dilbilgisi çalışmalarında değerlilik kuralına göre de bir NP, kullanılan eyleme bağlı olarak bir tümleyici, bir -e hali veya bir -i halinde kullanılabilir. Burada, PROLOG programı yazılırken, belirtilen bu durumların da dikkate alınması gerekir. Bunun için de yaklaşık olarak aşağıdaki kural geçerli olur:

vp (vp (I))   --------> iv (I)
.vp (vp (T, N)   --------> tv (T), np (N).

Bu kuralın geçerliğini test etmek için aşağıdaki tümceyi yazalım.

Der wachsame Lehrer schnappt den Studenten. (=Uyanık öğretmen öğrenciyi yakaladı.)

s (np (det (der), ng (adj (wachsame), ng (noun (lehrer)))), vp (tv (schnappt), np (det (den) ng (noun (studenten))))).

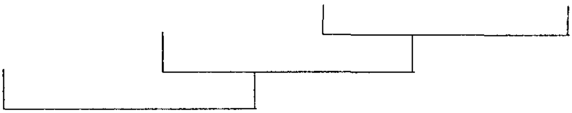
Buraya kadar uygulamaya çalışılan PROLOG-Parser programı bütün sözcük türlerini doğru betimleyebilmişse hiç sorun çıkarmadan çalışacaktır. Burada özellikle biçim bilgisini ilgilendiren konulara hiç değinilmedi. Bu nedenle, bazı ilginç ve çözümü zevkli sorunlar çıkabilir. Öğretmen sadece uyanık (wachsam) değil; çok uyanık (sehr wachsam) da olabilir. Bu durumda yazılan program genişletilebilir. Bunun için gerekli kural:

ng (ng (A, N)) ------> adj (A), ng (N).

şeklindedir. PROLOG bu verilen kurala göre tümcenin yeni şeklini analiz edecektir.

der sehr wachsame Lehrer

np (det (der), ng (adj) (sehr), ng (adj (wachsame), ng (noun (lehrer))))



Böylece arzu edilen program oluşturuldu, ama yapılacak işlem bitirilmedi. PROLOG ile yazılan bu programa göre yukarıda genişletilmiş örnekte diğer bütün sıfatlar eş değer olarak algılandı. Söz gelimi yukarıda sözü edilen öğrenciler, çalışıyor olabilirler (arbeitende Studenten); belki iyi çalışıyor olabilirler (gut arbeitende Studenten), hatta çok iyi çalışıyor olabilirler (sehr gut arbeitende Studenten). Bunun için yukarıda sağa doğru genişletilmiş olan örnek sözcük türleri doğru tanımlanması koşuluyla, sağa doğru kaydırma kuralı aynen uygulanabilir. Burada yapılan işlemin adı, Almanca işlemlerde olduğu gibi **seri bağlama kuralı** (Serialisierungsregeln) olarak bilinebilir. Burada aslolan, yapılacak bu tür işlemlerde olası tümce ya da ad grubunu düşünerek program oluşturmaktır. Programda, gerekiyorsa, sözcüklerin içerdikleri ya da içermesi olası yan anlamları da düşünmek ve bu sisteme göre programın oluşumunu sağlamaya çalışmak gerekmektedir. Bu konu ile ilgili olarak Hans Glinz (1971: Kap.16) tarafından duruma, dereceye ve derecelendirmeye bağlı bir sınıflandırma yapılmaktadır. Bu sınıflandırma, özellikle sıfatlarda PROLOG programının yazılmasında büyük kolaylık sağlamaktadır.[2]

Yazılan programlarda tümce açısından dil karmaşıklaştıkça, yukarıda açıklanan yöntem kullanılmak suretiyle sağdan sola doğru açımlama yapılarak sonuca varılır. Program bu yönüyle, belki diğer bilgisayar programlarından daha kolay değildir. Özellikle tümce analizi boyutundaki sorunlar üzerindeki çalışmalar hala daha sürdürülmektedir. Ancak, dilbilimin gerektirdiği yapısal analizlere ve geliştirmeye çok açık olması, programın tercih edilmesini sağlamaktadır.

---

[2] Viyana Üniversitesi'nde bu konuda bir proje çalışması yapılmıştır. İlgi duyanlar PROLOG üzerine bilgi almak üzere Willibad KRAML ile irtibat kurabilirler. E-Mail: A7511 daa @ AWIUNI11.

KAYNAKÇA

Bergen, Karl von: **LISP für Linguisten: ein Grundkurs zur Einführung in die Computerlinguistik**. Frankfurt/aM; Berlin; New York; Paris; Wien: Lang, 1992 (Hamburger englische Linguistik Praktika Bd. 4).

Glinz, Hans: **Deutsche Grammatik II**. Frankfurt/aM. 1971.

Knaus, Rodger: *Faster and Smarter Queries.* **AI Expert's Toolbok**. May 1988, SS. 13-18.

Knaus, Rodger & : *A Simple PROLOG DBMS [data base management system].* **AI Expert's Toolbok**. April 1988, SS.13-21.

Kraml Willibald & Schrodt, Richard: *Syntax lernen mit PROLOG.* In: **Ide: Zeitschrift für den Deutschunterricht in Wissenschaft und Schule**. 2/1990. SS. 98-114.

Lenders, Winfried & Willée, Gerd: **Linguistische Datenverarbeitung: ein Lehrbuch**. Opladen: Westdeutscher Verlag, 1986.

Türker, Faruk: *Dilbilimin Çalışma Alanlarında Bilgisayar Kullanımı.* **Dilbilim Araştırmaları. 1991**. Ankara: Hitit Yayınları, 1991. SS.137-143.

Vardar, Berke vd.: **Dilbilim ve Dilbilgisi Terimleri Sözlüğü**. Ankara: TDK-Yay.No: 471, 1980.