Anna Sågvall Hein
UCDL
Centrum för Datorlingvistik
Uppsala Universitet

UPPSALA CHART PARSER, Version 2 (UCP-2) - En översikt.

1. Inledning

Uppsala Chart Parser är en lingvistisk processor för analys av naturligt språk. Den representerar en vidareutveckling av de grundläggande ideerna i The General Syntactic Processor (se Kaplan 73) och utnyttjar härvid vidare ideer och erfarenheter presenterade av M. Kay (Kay 75, 77a och 77b), A. Sågvall Hein (Sågvall 77a, 77b, 78 och 80a) och M. Carlsson (Carlsson 80 och 81).

För närmare upplysningar om bakgrunden till UCP samt om hur den är relaterad till tidigare arbeten, se Sågvall 80b.

2. Allmänt om UCP.

Den centrala datastrukturen i UCP är en chart. Den fungerar som en 'anteckningsbok', där allt som äger rum under bearbetningen av ett språkligt uttryck (ordform, fras, sats) noteras. Charten representerar det obearbetade språkliga uttrycket, delanalyser samt den (eller de) resulterande analyserna, s.k. passiv information. Vidare innehåller den information om hur bearbetningen skall fortskrida, s.k. processinformation.

Charten är en riktad graf, bestående av noder (vertices) och bågar (edges). Noderna är numrerade och bågarna är etiketterade.

Man skiljer mellan inaktiva och aktiva bågar.

De inaktiva bågarna bär i sina etiketter den passiva informationen t.ex. lingvistiska beskrivningar över språkliga enheter. Initialt representeras det språkliga uttryck som skall analyseras (analysuttrycket) avm en enkel chart bestående av inaktiva bågar, där varje båge representerar en karaktär i analysuttrycket. Under bearbetningens gång introduceras kontinuerligt nya bågar i charten, vilka representerar partiella analysresultat. Den slutliga analysen lagras i etiketten till den (eller de) inaktiva båge (eller bågar) som omspänner hela charten från första till sista vertex. Om analysen inte lyckas, återfinns inte någon sådan båge i charten.

Uppsala Chart Parser, Version 2 (UCP-2) - En översikt
Anna Sågvall Hein
Proceedings of NODALIDA 1981, pages 95-116

De aktiva bågarna innehåller i sina etiketter antingen namn på grammatiska regler som skall appliceras eller namn på lexikon (eller den punkt i ett visst lexikon) där lexikonsökning skall ske.

Utnyttjande av chartstrukturen gör det möjligt att

- känna igen samt beakta samtliga ambiguiteter i en sats
- analysera satsfragment
- undvika upprepad igenkänning av satsfragment

Hela bearbetningen i UCP tar formen av en följd av bearbetningssteg, tasks, som genereras och exekveras. Exekvering av en task innebär ett försök att tillämpa en grammatisk regel på en båge i charten eller en jämförelse mellan en båge i charten och en bokstav i något uppslagsord i något lexikon. (Uppslagsorden i lexikonen ligger lagrade som bokstavsträd.)

Nya tasks genereras automatiskt som följd av att nya bågar introduceras i charten. Detta sker enligt en allmän princip som säger att en task skall genereras för varje väg, path, i charten bestående av en aktiv edge följd av en inaktiv edge.

Introduktionen av nya bågar i charten styrs från de grammatiska reglerna och lexikonartiklarna. Kontrollen är sålunda decentraliserad.

När en task har genererats överförs den (scheduleras) till en agenda, varifrån den sedan hämtas (selecteras) för exekvering.

Agendan samt charten ger en komplett beskrivning av status på bearbetningen i varje moment.

UCP arbetar med godtyckligt antal lexikon och grammatikor.

I en och samma grammatiska formalism uttrycks såväl fonotaktiska som morfotaktiska och syntaktiska regler.

Formalismen är rent procedural.

I den procedurala formalismen uttrycker vi också informationen i de enskilda lexikonartiklarna.

Så snart ett uppslagsord återfunnits i något lexikon vidtar en interpretering av lexikoninformationen helt i analogi med interpreteringen av de grammatiska reglerna.

För varje enskilt lexikon kan vi också specificera vilka aktioner som skall utföras då ett lexikonsökningssteg lyckats. Även detta uttrycks i den grammatiska formalismen. Se vidare 4.1.1.1.

3. Den grammatiska formalismen

Den grammatiska formalismen inkluderar ett specificerat format, en struktur, för beskrivning av lingvistiska enheter samt en uppsättning grammatiska operatorer.

3.1 Strukturformatet

En struktur är en lista av egenskaper, där varje egenskap formuleras som ett attribut och ett värde.

3-1 visar en struktur som beskriver den svenska frasen 'i denna film'.

```
(SYN.CONST = PREP.PHRASE
 1 = (CAT = PREP
      LEX = I)
 2 = (SYN.CONST = NP
      1 = (CAT = DETER
           LEX = DENNA
           UTR.NEUTR = UTR
           ...)
      2 = (CAT = NOUN
           LEX = FILM
           GENDER = REAL
           UTR.NEUTR = UTR
           ...)))
```

Figure 3-1:

Ett attribut är ett (med några undantag, se nedan samt 3.3) fritt valt namn, en atom, t.ex. SYN.CONST (för syntaktisk konstituent) i figuren.

Ett värde är antingen en atom, t.ex. PREP.PHRASE i figuren eller en struktur, t.ex. (SYN.CONST = NP, 1 = (CAT = DETER, LEX = DENNA, ...), 2 = (CAT = NOUN, LEX = FILM, ...)).

Lingvistiska beskrivningar i form av strukturer lagras i etiketterna till bågarna i charten.

Med hjälp av de grammatiska operatorerna formuleras primitiva grammatiska operationer.

En grammatik består av en uppsättning grammatiska regler.

En grammatisk regel består av följd, en sekvens, av grammatiska operationer. I en grammatisk regel uttrycker man de operationer som skall utföras med avseende på en given inaktiv båge. _ Vilken den inaktiva bågen är specificeras i aktuellt bearbetningssteg (task).

Då vi formulerar en grammatisk regel refererar vi till den strukturella beskrivningen av den inaktiva bågen med hjälp av det reserverade attributnamnet *. (Den inaktiva bågens strukturbeskrivning utgör värdet på attributet *).

Även den aktiva bågens etikett har utrymme för en strukturbeskrivning. Det är där som interpretatorn lagrar den struktur som byggs upp under analysen av ett visst språkligt uttryck. Till den aktiva bågens strukturbeskrivning refererar vi i formuleringen av den grammatiska regeln med hjälp av attributnamnet &.

I 3-2 ger vi ett exempel på attributet & och dess värde för en aktiv edge.

```
(& = (SYN.CONST = PREP.PHRASE
      1 = (CAT = PREP
            LEX = I)))
```

Figure 3-2:

Den aktiva edge, vars strukturbeskrivning, egenskapslista, visas i 3-2, representerar en delanalys av uttrycket 'i denna film'.

3-3 visar denna edge _ den riktade bågen från 1 till 3 _ i den aktuella chartstrukturen.

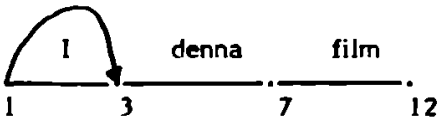


Figure 3-3:

Anm. 1 Figuren visar endast de bågar och noder som är väsentliga för att visa den aktiva edgen i sitt sammanhang.

Anm. 2 De inaktiva edgarna är inte riktade till skillnad från de aktiva. För detaljer rörande implementeringen av chartstrukturen, se Carlsson 80 och 81.

Förutom strukturbeskrivningen innehåller de aktiva bågarna också information om vilken regel (eller lexikonbåge vid lexikonsökningen) som skall tillämpas, samt uppgift om i vilket bearbetningssteg, task, som de genererats.

3-4 presenterar den fullständiga utskriften av den aktiva bågen i exemplet.

```
1--3  CREATOR: 39
      FEATURES: (& = (SYN.CONST = PREP.PHRASE
                      1 = (CAT = PREP
                           LEX = I)))
      LR-ACTION: (CONTINUE,
                  <* SYN.CONST> = 'NP,
                  <& :new> :=: <*>,
                  STORE),
                 RULEEXIT:(PREP:PHRASE)
```

Anm. De grammatiska operatorerna som förekommer under rubriken L(eft to) R(ight) ACTION förklaras nedan i 3.2.

Figure 3-4:

3.2 De grammatiska operatorerna

Med hjälp av de grammatiska operatorerna kan vi

- utföra tester,
- bygga upp lingvistiska beskrivningar, samt
- styra den vidare bearbetningen genom att införa nya bågar i charten

3.2.1 Testoperatorer

Testerna som utförs i de grammatiska operationerna avser egenskapslistorna hos den aktiva och den passiva båge som bearbetningssteget gäller.

Med hjälp av PATH-operatorn (notation: < >) kan vi extrahera värdet på en viss egenskap.

T.ex. <& SYN.CONST> applicerat på den aktiva edgen i 3-4 returnerar värdet PREP.PHRASE.

Med PATH-operatorn kan vi komma åt värdet av en egenskap på godtyckligt djup i en struktur.

T.ex. <& 1 CAT> applicerat på samma struktur ger värdet PREP.

En PATH-operation kan också returnera en hel delstruktur.

T.ex. <& 1> i exemplet ovan returnerar (CAT = PREP, LEX = I).

Samma resultat får vi genom att använda oss av det reserverade attributnamnet :LAST, som returnerar den sista delstrukturen i en strukturbeskrivning. Operationen skulle då formuleras <& :LAST>.

Alla operationer returnerar ett värde, antingen ett faktiskt värde som i exemplen ovan vilket då tolkas som TRUE eller ett negativt värde, NIL. Om den egenskap man frågar efter i PATH-operationen saknas i strukturen, så returneras värdet NIL. Ett PATH-uttryck kan sålunda i sig fungera som en test.

EQUALITY-operatorn (notation: =) används för att testa på likhet mellan två PATH-uttryck eller mellan ett PATH-uttryck och en atom (ett visst ord).

T.ex <& 1 LEX> = 'PREP i exemplet ovan, skulle returnera värdet TRUE. (Atomer måste markeras med ett enkelt anföringstecken enligt LISP-konvention.)

Det finns också en NEGATIONs-operator (notation: NOT) som t.ex. ger oss möjlighet att formulera ett test 'icke lika med'.

Med hjälp av OR-operatorn (notation: /) kan vi uttrycka alternativ, t.ex.

```
(<* FORM> = 'DEF /
 <* NUMB> = 'PLUR /
 <* PROPR> = T /
 <* CBLTY> = 'MASS)
```

i en NP-regel som krav på att ett ensamt substantiv skall få betraktas som en NP.

OR-operatorn returnerar TRUE om någon av dess operationer lyckas. Efterföljande operationer utförs ej (dependent disjunction).

Konditionala regler kan formuleras med hjälp av CONDITIONs- operatorn (notation: IF... THEN ... ELSE).

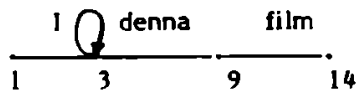
3.2.2 Presentation av SAME, ADVANCE, STORE och MAJORPROCESS i anslutning till ett exempel ur SVE.GRAM

För att illustrera den fortsatta presentationen av de grammatiska operatorerna anför vi i 3-5 en grammatisk regel för igenkänning av en NP-konstituent bestående av en determinerare och ett substantiv i svenska. Regeln är hämtad ur vårt fragment av en svensk grammatik för UCP.

```
DETER.NP
    <& SYN.CONST> :=: 'NP,
    <* CAT> = 'DETER,
    <& :NEW> :=: <*>,
    ADVANCE,
    <* CAT> = 'NOUN,
    <& :LAST NUMB> = <* NUMB>,
    <& :LAST UTR.NEUTR> = <* UTR.NEUTR>,
    (<& :LAST LEX> = <* DENNA>,
     <* FORM> = 'INDEF /
     <* FORM> = 'DEF),
    <& :NEW> :=:<*>,
    STORE,
    MAJORPROCESS(NP)
```

Figure 3-5:

I 3-6 presenterar vi en förenklad chartstruktur som visar en situaton där DETER.NP regeln aktiveras.




```
3--3 CREATOR: 13
     LR-ACTION: DETER.NP

3--9 CREATOR: 12
     FEATURES: (* = (CAT = DETER
                     LEX = DENNA
                     NUMB = SING
                     UTR.NEUTR = UTR
                     DEF.INDEF = DEF
                     FORM = INDEF))

...
```

Figure 3-6:

En grammatisk regel består av en SEQUENCE (notation: ,) av operationer. Under interpreteringen av en regel exekveras de olika operationerna till dess någon misslyckas (returnerar värdet NIL). Sekvensen som sådan returnerar värdet TRUE om alla de ingående operationerna lyckas (returnerar värdet TRUE). Om någon operation misslyckas, så returnerar sekvensen som sådan värdet NIL.

Sekvenser kan uppträda som argument till andra operatorer (t.ex. i konditionala uttryck och OR-uttryck).

DETER.NP regeln i 3-5 är en sekvens bestående av 11 operationer.

Den första operationen är en tilldelningssats, formulerad med hjälp av SAME-operatorna (notation: :=:). I operationen tilldelas attributet SYN.CONST i den aktiva bågens egenskapslista värdet NP. Med andra ord, så börjar vi här bygga upp en lingvistisk beskrivning av den konstituent vi förväntar oss att känna igen.

I nästa sats testar vi på den inaktiva bågens kategoritillhörighet. Vi kräver att den skall ha egenskapen CAT = DETER. (Till den syntaktiska kategorin DETER hänför vi t. ex. 'denna' och 'den här'.

I den tredje satsen gör vi en ny tilldelning, där vi använder oss av det reserverade attributnamnet :NEW samt av path-uttrycket <*>.

Satsen innebär att hela den inaktiva bågens egenskapslista <*> ges som värde till attributet :NEW i den aktiva bågens strukturbeskrivning. Användningen av det reserverade attributnamnet :NEW får till följd att det byggs upp en ny egenskap, vars attribut är en siffra, som är en enhet högre än tidigare numrerade attribut. Denna facilitet används för att kunna numrera delkonstituenter i en konstituent.

ADVANCE-operatorn är en av de 5 process-operatorerna. Med en process-operator menar vi en operator som lägger in nya bågar i charten och därigenom för bearbetningen framåt. (Alla processoperatorerna returnerar värdet TRUE.)

ADVANCE-operatorn har den effekten att en ny aktiv båge läggs in i charten. Denna båges egenskapslista innehåller attributet & med tillhörande värde. Den innehåller vidare resten av den grammatiska sekvens i vilken den ingår. Intuitivt svarar ADVANCE-operatorn mot att man tar ett steg framåt i analysuttrycket; en eller flera nya tasks kommer att genereras vilka gäller den nya aktiva bågen och följande inaktiva båge/bågar.

3-7 visar chartstrukturen (i förenklad form) efter exekveringen av ADVANCE-satsen i regeln DETER.NP.

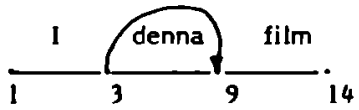
Den fortsatta interpreteringen av DETER.NP-regeln kommer sålunda att ske i en ny task som gäller den aktiva bågen från 3 till 9 samt den passiva bågen från 9 till 14.

Här testas först den inaktiva bågens kategoritillhörighet. Därpå testas överensstämmelse med avseende på egenskaperna UTR.NEUTR och NUMB. (Obs. Om någon test skulle misslyckas så avbryts exekveringen av tasken, eftersom operationerna ingår i en sekvens.)

För referens till den sist igenkända delkonstituenten i den aktiva edgen använder man sig av det reserverade attributnamnet :LAST.

Därpå följer en OR-operation. (Obs. Både 'denna' och 'den här' är kategoriserade som determinerare. Alternativen motiveras av att de kräver indefinit resp. definit form på substantivet.)

Efter OR-operationen följer en STORE-operation. STORE tillhör processoperatorerna. En STORE-operation innebär att en ny inaktiv edge införs i charten. Den får som strukturbeskrivning värdet av attributet &, dvs. den övertar den aktiva bågens




```
3--9 CREATOR: 29
     FEATURES: (& = (SYN.CONST = NP
                     1 = (LEX = DENNA
                          CAT = DETER
                          NUMB = SING
                          UTR.NEUTR = UTR
                          FORM = INDEF))
     LR-ACTION: (CONTINUE,
          (<CAT *> = 'NOUN,
          <& :LAST NUMB> = <* NUMB>,
          <& :LAST UTR.NEUTR> = <* UTR.NEUTR>,
          (<& :LAST LEX> = 'DENNA, <* FORM> = 'INDEF /
           <* FORM> = 'DEF),
          <& :NEW> :=: <*>,
          STORE,
          MAJORPROCESS(NP))
          RULEEXIT: DETER.NP)

9--14 CREATOR: 21
   FEATURES: (* = (CAT = NOUN
                   LEX = FILM
                   UTR.NEUTR = UTR
                   NUMB = SING
                   FORM = INDEF
                   ...))

...
```

Figure 3-7:

egenskapslista. Den omspänner den följd av bågar som regeln omfattar. I vårt exempel medför STORE att en ny inaktiv edge läggs in i charten från vertex 3 till vertex 14.

DETER.NP-regeln avslutas med operationen MAJORPROCESS(NP). Operatorn MAJORPROCESS är den tredje av processoperatorerna. Den tar som argument namnet på en grammatisk regel (eller namnet på en grammatik eller ett på ett lexikon). NP är namnet på en regel i vår svenska grammatik. Operationen MAJORPROCESS(NP) innebär att en ny aktiv edge, vars LR-ACTION är NP, införs i charten. Denna aktiva edge går från och till den nod i vilken aktuell regel aktiverades, dvs. i vårt exempel i vertex 4. I NP-regeln beskrivs NP-konstituenter, och regeln inkluderar igenkänning av efterställda bestämningar, dvs. relativa satser och prepositionsfraser. Tillgång till operatorn MAJORPROCESS innebär möjlighet att arbeta med ett look-ahead. Den bidrar till att göra analysen datadriven. Först när vi ser vad vi har initierar vi en viss process.

I 3-8 presenteras en analys av uttrycket 'I denna film från Kanada' enligt vårt nuvarande grammatikfragment för svenska.

```
I denna film från Kanada:

(SYN.CONST = PREP.PHRASE
 1 = (CAT = PREP
      LEX = I)
 2 = (SYN.CONST = NP
      1 = (SYN.CONST = NP
           1 = (CAT = DETER
                LEX = DENNA
                ...)
           2 = (CAT = NOUN
                LEX = FILM
                ...))
      2 = (SYN.CONST = PREP.PHRASE
           1 = (CAT = PREP
                LEX = FRÅN)
           2 = (SYN.CONST = NP
                1 = (CAT = NOUN
                     LEX = KANADA
                     PROPR = T
                     ...))))
```

Figure 3-8:

3.2.3 Presentation av PROCESS i anslutning till ett exempel ur SVE.DIC

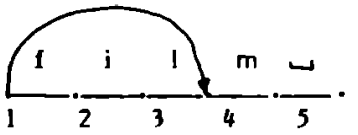
3-9 visar den lexikoninformation som är associerad med uppslagsordet 'film' i vår svenska applikation av UCP. Lexikonartikeln är hämtad från huvudlexikonet SVE.DIC.

```
FILM : <& LEX> :=: 'FILM,
       <& MORPH.CAT> :=: 'ROOT,
       <& ROOT.CAT> :=: 'NOUN,
       <& ROOT> :=: 'FILM,
       <& GENDER> :=: 'REAL,
       <& ANIM.INANIM> :=: 'INANIM,
       <& CBLTY > :=: 'COUNT,
       <& PROPR > :=: 'NO,
       <& UTR.NEUTR> :=: 'UTR,
       <& DECL> :=: '-ER,
       STORE,
       MAJORPROCESS(NOUN),
       ADVANCE,
       (<* CHAR :TYPE> = 'SEP, PROCESS(SEP)/
        (<* CHAR> = 'E, PROCESS(NUMB), PROCESS(FORM)/
         <* CHAR> = 'S, PROCESS(CASE), PROCESS(SVE.DIC)),
         MAJORPROCESS(COMPOUND))
```

Figure 3-9:

3-10 visar en (förenklad) chartstruktur, där denna 'regel' aktiveras. Exemplet gäller analysuttrycket 'film'.

Sekvensen i 3-9 inleds med 10 tilldelningssatser. Därpå följer en STORE-operation, som leder till att en ny inaktiv edge från 1 till 5 läggs in i charten. Denna edge har en egenskapslista som överensstämmer med tilldelningarna ovan.




```
1--2 CREATOR: 0
     FEATURES: (* = (CHAR = F))

2--3 CREATOR: 0
     FEATURES: (* = (CHAR = I))

3--4 CREATOR: 0
     FEATURES: (* = (CHAR = L))

4--5 CREATOR: 0
     FEATURES: (* = (CHAR = M))

5--6 CREATOR: 0
     FEATURES: (* = (CHAR =  ))

1--4 CREATOR: 6
     FEATURES: NIL
     LR-ACTION: <& LEX> :=: 'FILM,
                <& MORPH.CAT> :=: 'ROOT,
                 ...
 ...
```

Figure 3-10:

MAJORPROCESS(NOUN) innebär att en ny aktiv edge från och till vertex 1 läggs in i charten. Dess LR-ACTION är en hänvisning till den grammatiska regeln NOUN, som specificerar strukturen för ett svenskt substantiv. I kraft av den allmänna taskgenereringsprincipen kommer denna edge att leda till att substantivregeln kommer att börja tillämpas på stammen 'film'.

ADVANCE-operationen innebär att vi tar ett steg (edge) framåt i charten för att kunna se följande karaktär.

Här har vi specificerat 2 huvudalternativ

1. karaktären har typmarkering SEParator

2. karaktären är identisk med e eller s

För det första alternativet har vi utnyttjat en facilitet i UCP som gör det möjligt att lagra och återvinna egenskaper hos de enskilda karaktärerna. Sålunda har mellanslaget och skiljetecknen fått typbeteckningen 'separator'. Med hjälp av det reserverade attributnamnen :TYPE kan vi fråga på denna egenskap. Övrig information om separatorerna finns lagrad i ett separatorlexikon (SEP), där varje separator ligger som ett eget uppslagsord.

PROCESS är den 4.e processoperatorn. Den initierar processerande med avseende på en viss grammatik eller ett visst lexikon. I argumentet till PROCESS anger man namnet på grammatiken (eller visst läge i en aktiv grammatik) eller på lexikonet. Processerandet startar i den vertex i vilken den aktuella aktiva edgen slutar. (Jfr. MAJORPROCESS som initierar processerande i den vertex där den aktuella aktiva edgen börjar.)

PROCESS(SEP) leder i vårt exempel till att lexikonsökning i separatorlexikonet startar i vertex 5. Om påföljande karaktär är 'e' så initieras processerande i både numeruslexikonet (NUMB), jfr. 'filmer', och i formlexikonet (FORM), jfr. 'filmen'. Är påföljande karaktär 's' så initieras lexikonsökning i kasuslexikonet (CASE), jfr. 'films'.

Inte i något av de sistnämnda fallen kan man utesluta möjligheten att andra ledet i en sammansättning följer. Därav MAJORPROCESS(COMPOUND), PROCESS(SVE.DIC).

3.2.4 Presentation av MINORSTORE samt ALTERNATION och RULE CALL i anslutning till ett exempel ur SVE.GRAM

MINORSTORE fungerar som STORE med den skillnaden att den inaktiva edge som läggs in omspänner samma sekvens av edgar som den aktuella aktiva edgen. (STORE-operatorn lägger in en inaktiv edge som omspänner samma följd av edgar som den aktiva samt följande inaktiva.) MINORSTORE i kombination med ADVANCE kompletterar våra möjligheter till ett look-ahead av en edge (jfr. MAJORPROCESS).

ALTERNATION är en operator som möjliggör parallellprocesserande. Den används i situationer där man inte t. ex. med OR-operatorn kan välja alternativ utan måste gå fram med flera tolkningar.

I 3-11 illustreras användningen av ALTERNATION i den grammatiska regeln NO.NOUN.ENDING i vår experimentgrammatik SVE.GRAM.

```
NO.NOUN.ENDING:

    (<& DECL> = '-, (<& NUMB> ::: 'SING // <& NUMB> ::: 'PLUR) /
     <& NUMB> ::: 'SING),
    <& FORM> ::: 'INDEF,
    <& CASE> ::: 'COMMON,
    STORE;
```

Figure 3-11:

Regeln i 3-11 aktiveras i en situation där man har en följd av en substantivisk stam och en separator. Utgående från stammens deklinationstyp gör den aktuella tilldelningar. Med hjälp av OR-operatorn gör den åtskillnad mellan substantiv av deklinationstyp '-' (ingen ändelse i pluralis) och övriga. För substantiv tillhöriga den förstnämnda typen görs två parallella tilldelningar till attributet NUMB (numerus), nämligen SING och PLUR. Övriga substantiv tilldelas värdet SING för numeruskategorin. I charten avspeglas parallella tilldelningar i form av alternativa edgar.

3-11 exemplifierar ytterligare en facilitet i den grammatiska formalismen. För att göra grammatiken överskådlig och för att slippa upprepa sekvenser av operationer som förekommer på flera ställen i grammatiken eller i flera lexikonartiklar, kan vi extrahera sådan sekvenser och föra upp dem under egna namn i grammatiken. Tack vare RULE CALL faciliteten kommer dessa sekvenser att interpreteras då deras namn påträffas i en regel. NO.NOUN.ENDING-regeln i figuren är en sådan sekvens. De grammatiska operationerna i regeln kommer att exekveras då interpretatorn träffar på namnet NO.NOUN.ENDING i regeln NOUN.

3.3 Reserverade attributnamn

I 3.1 nämns att attributnamn med några undantag när kan väljas fritt. Det finns tre typer av reserverade attributnamn, nämligen

1. & och * ,

2. :NEW och :LAST, samt

3. övriga attributnamn, prefigerade med : , t.ex. :TYPE.

Dessa tre typer av attributnamn behandlas olika av interpretatorn.

& och * behandlas som de fritt valda attributnamnen, dvs. de tolkas bokstavligt.

2. utlöser vid interpreteringen subrutinanrop. :NEW har den effekten att ett attributnamn skapas. Det skapade attributnamnet är en siffra, som är en enhet högre än det högsta redan förefintliga sifferattributet i den aktuella aktiva edgens strukturbeskrivning (jfr. 3-5). :LAST i ett path-uttryck identifieras av interpretatorn som en referens till det sista attributet i en given strukturbeskrivning (jfr. 3-5).

3. förekommer enbart i path-uttryck som avser karaktärer. Förekomst av prefixet : på ett attributnamn signalerar till interpretatorn att egenskapen inte ligger lagrad i den inaktiva edgens egenskapslista utan som en global egenskap till karaktären i fråga. Denna facilitet gör det möjligt att utifrån grammatiken kunna komma åt globalt lagrade egenskaper avseende karaktärerna.

I vår svenska applikation har vi bl. a. utnyttjat :-konventionen för att kunna fråga på globalt lagrade fonetiska särdrag utifrån fonologiska regler. Exempel på detta ges i 4.1.1.

4. Processintegrering

Som framgått av presentationen av de grammatiska operatorerna har man fullständig frihet att initiera olika slags processer från en grammatisk regel eller en lexikonartikel. Härvid utnyttjar man operatorerna PROCESS och MAJORPROCESS. Man har möjlighet att experimentera med såväl regelstyrd (top-down) som datastyrd (bottom-up) bearbetning tillika med en kombination av båda.

Förutom från grammatiska regler och lexikonartiklar kan man initiera godtyckligt processerande från de aktioner som kan specificeras att gälla för sökningen i ett givet lexikon (se 4.1.1.1).

Exemplen ovan har visat samspelet mellan grammatiska regler och lexikonartiklar. Nedan illustrerar vi processintegrering avseende lexikonsökning och omskrivning.

4.1 Lexikonsökning och omskrivning

Ett problem som ständigt dyker upp i samband med språkanalys är frågan om hur man skall hantera fonologiska och morfofonematiska alternationer. Vanligtvis definierar man s. k. omskrivningsregler (rewriting rules) vilka återför alternanterna till en kanonisk lexikonform.

Nästa fråga gäller hur interpreteringen av omskrivningsreglerna skall integreras med övrig bearbetning.

Den grammatiska formalismen i UCP erbjuder en notation i vilken omskrivningsreglerna kan definieras.

Vad gäller integreringen av omskrivningsprocessen med övriga processer, så har man en gynnsam situation i UCP.

För ett givet lexikon kan man specificera vilka aktioner som skall utföras i samband med att en lexikonsökningstask lyckas. Detta innebär, att man utifrån signaler i analysuttrycket kan aktivera olika omskrivningsprocesser.

Nedan presenterar vi en strategi för integrering av omskrivning och lexikonsökning på ett svenskt exempel.

4.1.1 Ett exempel från svenskan

I många svenska ord förekommer s. k. flyktig vokal. Vi tänker på fall som 'cykel' och 'cyklar', 'äpple' och 'äppelträd', 'kommen' och 'komna' etc.

I dessa fall kan man tänka sig att betrakta formen utan flyktig vokal som den kanoniska lexikonformen, och via någon omskrivningsregel återföra den utvidgade formen till den kanoniska.

4-1 ger exempel på hur en sådan regel skulle kunna se ut i vår grammatiska notation. I 4-1 utnyttjar vi en möjlighet, som vi tidigare inte nämnt, nämligen den att kunna ge argument till STORE-operatorn.

STORE utan argument har den effekten att en ny edge läggs in i charten. Denna edge omspänner samma följd av bågar som den aktuella aktiva edgen inklusive närmast följande inaktiva edge. Den övertar den aktiva edgens egenskapslista.

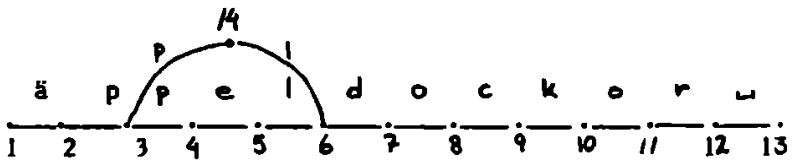
Om STORE ges med argument, så lägger den in lika många nya inaktiva edgar som

```
UNSTABLE.VOWEL
    (<* CHAR :MODE> = 'STOP/<* CHAR> = 'S/<* CHAR> = 'M),
    <& FIRST.CONS CHAR> :=: <* CHAR>,
    <& FIRST.CONS REWR> :=: 'UNSTABLE.VOWEL,
    ADVANCE,
    <* CHAR :TYPE> = 'VOWEL,
    <& FIRST.CONS UNSTABLE.VOWEL> :=: <* CHAR>,
    ADVANCE,
    (<* CHAR :MODE> = 'LIQUID/<* CHAR> = 'N),
    <& SEC.CONS CHAR> :=: <* CHAR>,
    NOT <& FIRST.CONS CHAR> = <& SEC.CONS CHAR>,
    <& SEC.CONS REWR> :=: 'UNSTABLE.VOWEL,
    <& SEC.CONS UNSTABLE.VOWEL> :=:
      <& FIRST.CONS UNSTABLE.VOWEL>,
    STORE(<& FIRST.CONS>,<& SEC.CONS>)
```

Figure 4-1:

antalet argument. I argumenten specificeras egenskapslistorna för de olika bågarna. Följden av inaktiva edgar som läggs in i charten omspänner samma följd av edgar som den som STORE utan argument inför.

Appliceras omskrivningsregeln UNSTABLE.VOWEL i vertex 3 i en chart som representerar ordet 'äppeldockor' så får vi följande resultat (se 4-2).




```
3--3 CREATOR: 9
       LR-ACTION: UNSTABLE.VOWEL

3--4  CREATOR: 0
      FEATURES: (* = (CHAR = P))

3--14 CREATOR: 12
      FEATURES: (* = (CHAR = P
                      REWR = UNSTABLE.VOWEL
                      UNSTABLE.VOWEL = E))
4--5 CREATOR: = 0
     FEATURES: (* = (CHAR = E))

5--6 CREATOR: = 0
     FEATURES: (* = (CHAR = L))

14--6 CREATOR: 12
      FEATURES: (* = (CHAR = L
                      REWR = UNSTABLE.VOWEL
                      UNSTABLE.VOWEL = E))
```

Figure 4-2:

Vi har sålunda en väg i charten (1 - 2 - 3 - 14 - 6) som svarar mot uppslagsordet i lexikonet.

```
ÄPPL  <& MORPH.CAT> ::: 'ROOT,
      <& ROOT> ::: 'ÄPPL,
      <& ROOT.CAT> ::: 'NOUN,
      (IF <* UNSTABLE.VOWEL>
         THEN <* UNSTABLE.VOWEL> = 'E,
              STORE,
              ADVANCE,
              PROCESS(SVE.DIC),
              MAJORPROCESS(COMPOUND)
         ELSE ...
```

Figure 4-3:

I 4-3 ges lexikonartikeln till roten 'äppl'. Som framgår av 4-3 så testar vi i lexikonartikeln på omskrivning samt flyktig vokal innan vi går vidare. (ELSE-alternativet gäller 'äpple, äpplet' etc.)

Ännu har vi inte berört hur vi får regeln UNSTABLE.VOWEL att aktiveras i vertex 3 i exemplet. Det sker utifrån en lexikonsökningstask på följande sätt.

4.1.1.1 Aktioner i samband med lexikontasks
För varje lexikon i UCP specificeras 4 obligatoriska 'entries', nämligen

1. SEARCH-KEY
2. FORWARD-ACTION
3. GATHERING-RULE
4. TOTAL-HIT-RULE-DEFAULT

I SEARCH-KEY anges på vilket attribut jämförelsen skall ske. I våra applikationer har vi arbetat med CHAR som söknyckel.

I FORWARD-ACTION anges hur bearbetningen skall gå vidare då man funnit överensstämmelse mellan en karaktär i analysuttrycket och en karaktär i bokstavsträdet (lexikonet). I våra applikationer har vi där ADVANCE som aktion, vilket innebär att man stegar sig fram till nästa karaktär.

I GATHERING-RULE kan man ange övriga aktioner som man vill utföra vid en träff.

I TOTAL-HIT-RULE-DEFAULT kan man specificera vad som skall ske i det fall uppslagsordet inte har någon information associerad med sig. Vi har där STORE.

I GATHERING-RULE för SVE.DIC (vårt svenska huvudlexikon) ger vi signalen för att UNSTABLE.VOWEL regeln skall aktiveras.

4-4 visar GATHERING-RULE för SVE.DIC.

```
GATHERING-RULE
    ADVANCE,
    (<* CHAR :MODE> = 'STOP / <* CHAR> = 'S / <* CHAR> = 'M),
    PROCESS(SVE.PHON);
```

Figure 4-4:

SVE.PHON är namnet på en svensk fonologisk grammatik, vars första regel är UNSTABLE.VOWEL. PROCESS(SVE.PHON) leder till ett försök att tillämpa UNSTABLE.VOWEL på den karaktär som uppfyller villkoren i satsen ovan.

5. Indexgrammatiken

Under analysen av ett visst uttryck införs kontinuerligt nya bågar i charten och inga bågar raderas ut.

Om vi t.ex. gör en satsanalys, kommer charten till slut att innehålla många olika slags bågar. Det är fråga om edgar som representerar (omskrivna och icke omskrivna) karaktärer, morfer, ordformer och syntaktiska konstituenter.

Låter vi den allmänna taskgenereringsprincipen verka utan restriktioner, så kommer det att genereras många inherent improduktiva tasks. Det är t. ex. meningslöst att generera en lexikonsökningstask där den inaktiva edgen representerar en syntaktisk konstituent.

Vi måste ha en möjlighet att kunna uttrycka vilka slags processer som skall verka på vilka slags lingvistiska enheter. Detta gör vi i en särskild grammatik, den s. k. indexgrammatiken. Indexgrammatiken är del av den språkspecifika informationen i en UCP parser.

I 5-1 ger vi några regler ur indexgrammatiken i vår svenska applikation.

```
SVE.DIC: CHAR;

NOUN: MORPH.CAT;

NP: CAT / SYN.CONST;

UNSTABLE.VOWEL: CHAR, NOT UNSTABLE.VOWEL;

...
```

Figure 5-1:

Den först regeln i 5-1 säger följande: Processering med avseende på lexikonet SVE.DIC skall endast gälla inaktiva edgar, vilka innehåller attributet CHAR, dvs. vid lexikonsökning i lexikonet SVE.DIC skall endast karaktär-bågar beaktas.

Den andra regeln säger, att processering med avseende på den grammatiska regeln NOUN1 endast skall gälla inaktiva edgar vilka innehåller attributet MORPH.CAT, dvs. vid interpretering av den grammatiska regeln NOUN skall endast morf-bågar beaktas och så vidare.

> Anm. UCP håller själv reda på vilka namn som refererar till lexikon och vilka som refererar till grammatikor och grammatiska regler.

6. Implementering

UCP består av ett kärnprogram, ett utskriftsprogram och en editor.

Kärnprogrammet anropas via funktionen PROCESS. Den tar tre argument, nämligen analysuttrycket (ett ord eller en hel sats omgiven av parenteser), namn på den överordnade grammatiken samt optionellt namnet på aktuell indexgrammatik.

T.ex.

```
(PROCESS (I denna film från Kanada får vi träffa en kvinna.)
         SVE.GRAM
         SVE.INDEX)
```

Bearbetningen startar med att ett initialt chart byggs upp. Därpå appliceras den första regeln i grammatiken på den första noden i charten. I vår svenska applikation lyder den första regeln

```
START.RULE
    PROCESS(SVE.DIC);
```

Vad som därpå sker bestäms av hur analysuttrycket ser ut samt av den språkspecifika informationen (lexika och grammatikor). Kontrollen över bearbetningen är sålunda decentraliserad. Den vilar på grammatikor och lexika.

Om analysen lyckas returnerar PROCESS värdet True i annat fall NIL. Analysresultatet skrivs ut med funktionen PRINTRESULT av CHART.

UCP körs vanligen interaktivt. Efter ett PROCESS-anrop hamnar man automatiskt i TRACE-mode. Därvid har man möjlighet att få de olika bearbetningsstegen (tasken) utskrivna på bildskärmen. Man kan också välja att gå ur TRACE-mode och enbart titta på slutresultatet. För de olika kommandona i 'trace-paketet', se Carlsson 81.

Med hjälp av utskriftsprogrammet TYPE kan man skriva ut såväl lexika (eller enskilda lexikonartiklar) som grammatikor (eller enskilda regler). Man kan också göra t. ex. (TYPE K) och få information om globalt lagrade egenskaper avseende karaktären K. UCP håller själv reda på vad som är lexikon, vad som är grammatik och vad som är en enskild karaktär.

Programmet EDIT är ett mycket viktigt hjälpprogram. Via EDIT editerar man i förefintliga lexika och grammatikor samt får hjälp med att bygga upp nya sådana. EDIT körs interaktivt.

UCP är implementerad i ett subset av BBN INTERLISP (se Teitel 74), nämligen Uppsala INTERLISP (se UDAC: 75). Den körs mot en IBM 370.

Angående implementeringen, se Carlsson 80 och 81.

7. Aktuella tillämpningar

Många ideer rörande utformningen av UCP har kommit fram under arbetet på en processmodell för ordigenkänning i finska (se Sågvall 78 och 80b).

Huvudapplikationen rör grammatisk analys av svenska. Den är bl. a. att betrakta som ett delprojekt inom projektet Datorsimulering av textförståelseprocessen. Projektet ingår i forskningsprogrammet för Uppsala Programming Methodology and Artificial Intelligence Laboratory (UPMAIL), delvis finansierat av Styrelsen för Teknisk Utveckling (STU). Projektet bedrivs i samarbete med Mats Carlsson, UCDL och UPMAIL.

Arbete pågår också med att definiera lexika och grammatik för automatisk morfologisk analys av serbo-kroatiska i UCP. Detta görs inom ramen för JUBA-projektet vid Slaviska institutionen i Lund (se Đurović 79).

UCP kan betraktas som ett redskap för studier av lingvistiska processer och deras samspel.

8. Planerad vidareutveckling

Den mest näraliggande vidareutvecklingen av UCP rör task-kommunikation och i anslutning därtill utarbetande av en metodik för sofistikerad schedulering och selectering av tasks.

9. Referenser

Carlsson 1980
M. Carlsson
Uppsala Chart Parser 1 - Program documentation
Report no. UCDL-R-80-2
Center for Computational Linguistics
Uppsala University, 1980

Carlsson 1981
M. Carlsson
Uppsala Chart Parser 2 - Program documentation
Report no. UCDL-R-81-1
Center for Computational Linguistics
Uppsala University (under utgivning)

Ďurovič 79
Durovic, L. & Stankovski, M.
Hemspråksutvecklingen hos serbokroatisk/kroatoserbisk-talande barn i Sverige,
Slaviska institutionen
Lund, 1979

Kaplan 1973
R. Kaplan
A General Syntactic Processor.
I: Rustin, R. (red.),
Natural Language Processing.
New York: Algorithmic Press, 1973
ss. 193-241

Kay 73
M. Kay
The MIND System
I: Rustin, R. (red.)
Natural Language Processing
New York: Algorithmic Press, 1973
ss. 155-188

Kay 75
M. Kay
Syntactic Processing and the Functional Sentence Perspective.
I: Schank, R. & Nash-Webber, B.L. (red.)
Theoretical Issues in Natural Language Processing (TINLAP-1)
Cambridge, Mass., 1975
ss.6-9

Kay 77a
M. Kay
Morphological and Syntactic Analysis.
I: Zampolli, A. (red)
Linguistic Structures Processing
Amsterdam: North-Holland, 1977
ss. 131-234

Kay 77b
M. Kay
Reversible Grammar.
I: Handbook for the 1977 Nordic Summer School in Computational Linguistics
Palo Alto, Xerox PARC, 1977

Sågvall 77a
A. Sågvall Hein

Chartanalys och morfologi.
I: Rapporter från Språkdata 3.
Föredrag från en konferens i Göteborg 10-11 okt. 1977
ss. 87-93

Sågvall 77b
A. Sågvall Hein
An Approach to the Construction of a Text Comprehension System for X-ray Reports.
I: Schneider, W. & Sågvall Hein, A. (red.)
Computational Linguistics in Medicine
Amsterdam: North-Holland, 1977
ss. 91-99

Sågvall 78
A. Sågvall Hein
Finnish Morphological Analysis in the Reversible Grammar System.
I: Proceedings from the 7th International Conference on Computational Linguistics, Bergen, 14-18 August, 1978
(under utgivning)

Sågvall 80a
A. Sågvall Hein
An Outline of a Computer Model of Finnish Word Recognition.
Fenno-ugrica suecana 3/1980
Finsk-ugriska institutionen
Uppsala Universitet, 1980

Sågvall 80b
A. Sågvall Hein
An Overview of the Uppsala Chart Parser Version 1 (UCP-1)
Report no. UCDL-R-80-1
Center for Computational Linguistics
Uppsala Universitet, 1980

Teitel 74
Teitelman, W., Hartley, A. K., Goodwin, J. W., Lewis, D. C., Bobrow, D. G., Masinter, L. M.
INTERLISP reference manual.
Palo Alto, Xerox PARC, 1974

UDAC: 75 INTERLISP/360 and /370 user reference manual.
Uppsala University Data Center
Uppsala, 1975