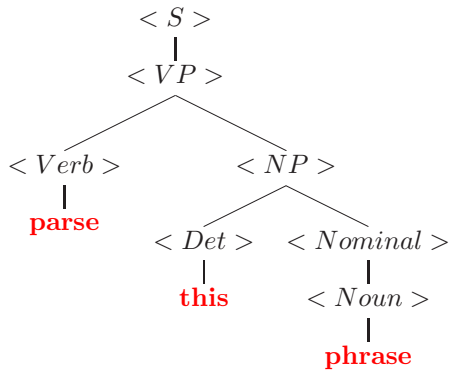
Συντακτική Ανάλυση Φυσικής Γλώσσας με τον αλγόριθμο early

Παπούλιας Νίκος
A.M: 2001030039

9 Ιουνίου 2005



Περίληψη

Η παρούσα αναφορά περιγράφει την λειτουργικότητα και τις λεπτομέρειες υλοποίησης ενός συντακτικού αναλυτή φυσικής γλώσσας , υλοποιούμενο με τον αλγόριθμο early .

Περιεχόμενα

1 Εγκατάσταση του αναλυτή και λειτουργικότητα

1.1 Εγκατάσταση

Η εγκατάσταση του αναλυτή γίνεται μέσω του script : "install.sh" το οποίο δίνεται μαζί με τον πηγαίο κώδικα του προγράμματος . Μετά την επιτυχή εκτέλεση του script δημιουργείται το εκτελέσιμο αρχείο "early" , το οποίο υλοποιεί τον αναλυτή .

1.2 Λειτουργικότητα

1.2.1 Είσοδος του προγράμματος

Ο αναλυτής δέχεται σαν είσοδο ένα αρχείο με την Context-Free γραμματική που θα χρησιμοποιήσει , και μία φράση η οποία ζητείται να αναλυθεί . Η δοθείσα γραμματική πρέπει να περιγράφεται σε απλουστευμένο BNF notation .

1.2.2 Εκτέλεση του προγράμματος

Η χρήση του εκτελέσιμου "early" είναι η εξής :

- early $< grammar - file - path >$ "expression to parse"

1.2.3 Έξοδος του προγράμματος

Η κύρια έξοδος του προγράμματος είναι ένα .dvi [1] αρχείο στο οποίο περιγράφονται

- τα συντακτικά δέντρα της φράσης που δόθηκε εάν αυτή αναγνωρίστηκε από την δοθείσα γραμματική ή ένα μήνυμα που πληροφορεί τον χρήστη ότι η δοθείσα φράση δεν αναγνωρίζεται από το συντακτικό .
- ο πίνακας early που κατασκευάστηκε από τον αλγόριθμο .
- οι κανόνες της γραμματικής που χρησιμοποιήθηκε .

Για την δημιουργία του .dvi αρχείου προυπόθεση είναι η ύπαρξη της LaTeX στο σύστημα του χρήστη , ενώ η προβολή του αρχείου μπορεί να γίνει σε οποιοδήποτε .dvi viewer που έρχεται μαζί με τις εκδόσεις του linux . Ο αναλυτής προσπαθεί αυτόματα να καλέσει την LaTeX και να προβάλει το .dvi μέσω του πολύ διαδεδομένου viewer: xdvi για το περιβάλλον των X-Windows .

Για την περίπτωση που δεν υπάρχει η LaTeX στο σύστημα του χρήστη , τυπώνεται στο standard output ο πίνακας early ο οποίος περιέχει όλες τις πληροφορίες που σχετίζονται με την αναγνώριση της δοθείσας φράσης .

2 Λεπτομέρειες Υλοποίησης

Ο αναλυτής αποτελείται κυρίως από τρία δομικά μέρη:

[1] Τα .dvi αρχεία είναι τα επεξεργασμένα αρχεία της mark-up γλώσσας TeX , στην οποία περιγράφεται η έξοδος του προγράμματος

2.1 Ανάγνωση της Context-Free γραμματικής

Ένα front-end για την Context-Free γραμματική , το οποίο αναλύει το αρχείο των κανόνων και δημιουργεί ένα symbol table το οποίο αντιστοιχεί έναν μοναδικό ακέραιο στα μη τερματικά και τερματικά σύμβολα της γραμματικής επαυξημένα με πληροφορίες όπως αν το σύμβολο είναι μέρος του λόγου κ.τ.λ . Επιπλέον οι κανόνες 'μεταφράζονται' από το front-end σε εσωτερική μορφή συνδεμένων λιστών ακεραίων και σχετίζονται με τα σύμβολα που τους παράγουν , ώστε να βρίσκονται σε μορφή εύκολα 'αναγνώσιμη' από τον αλγόριθμο "early" . Το front-end υλοποιήθηκε με τα meta-tools flex και bison και το BNF-notation που αναγνωρίζει είναι της μορφής :

- rule := tag → taglist
- rule := tag → terminal
- taglist := tag
- taglist := taglist tag

2.2 Ο αλγόριθμος early

Αφού διαβαστούν οι κανόνες της γραμματικής από τον αναλυτή η φράση που δόθηκε στο input χωρίζεται σε λέξεις και καλείται ο αλγόριθμος early , η υλοποίηση του οποίου σε C φαίνεται παρακάτω :

hearly.c

```
#include "operators.c"

void hearly_algorithm(char** input,word_index num_of_words){
        unsigned char i = 0;
        expression_list* expr_list;
        expression* expr;

        while(i<=num_of_words){
                expr_list = &(chart[i]);
                initialise_expression_list_scanner(expr_list);

                while(expr_list->scanner!=0){
                        expr = expr_list->scanner;
                        if(expr->dot!=0){
                                if(symbol_table[expr->dot->index]->is_part_of_speech==0){
                                        predictor_op(i,expr->from,expr->to,expr->dot->index);
                                }
                                else{
                                        if(i<num_of_words) scanner_op(i,input[i],expr);
                                }

                        }
                        else{
                                completer_op(i,expr);
                        }
                        scan_next_expression_list(expr_list);
                }
                i++;
        }
};
```

2.2.1 Επεξήγηση του αλγορίθμου

Ο αλγόριθμος early που υλοποιήθηκε πέρνει σαν είσοδο ένα array με τις λέξεις της υπο ανάλυση φράσης και το μέγεθος του array αυτού . Το early chart στο οποίο επενεργεί ο αλγόριθμος είναι global , οπότε και προσβάσιμο από τον αλγόριθμο . Το chart έχει αρχικοποιειθεί με το expression:

s0 : $< gamma > \rightarrow \quad . < S >$ [0,0] dummy-start-state []

το οποίο τοποθετείται στο chart[0] . Ο αλγόριθμος αποτελείται ουσιαστικά από δύο βρόγχους . Στον εξωτερικό βρόγχο *(γραμμές 8 εώς 29)* διατρέχουμε τον αλγόριθμο για ολόκληρο το input δηλαδή για όλες τις λέξεις αυτού *(γραμμή 8 η συνθήκη του while)* ενώ στον εσωτερικό βρόγχο *(γραμμές 12 εώς 27)* διατρέχουμε όλα τα expressions που βρίσκονται στο τρέχον chart (chart[i]) *(γραμμή 12 η συνθήκη του while)* . Τώρα στο τρέχον chart και για όλα τα expressions αυτού *(γραμμές 14 εώς 25)* ο αλγόριθμος επιλέγει :

- Αν το expression που εξετάζει **δεν είναι πλήρες** *(συνθήκη γραμμής 14)* και ένα σύμβολο που **δεν είναι μέρος του λόγου** περιμένει να αναγνωριστεί *(συνθήκη γραμμής 15)* καλεί τον **predictor** *(γραμμή 16)* .
- Αν το expression που εξετάζει **δεν είναι πλήρες** *(συνθήκη γραμμής 14)* και ένα σύμβολο που **είναι μέρος του λόγου** περιμένει να αναγνωριστεί *(συνθήκη γραμμής 15)* καλεί τον **scanner** *(γραμμή 19)* .
- Αν το expression που εξετάζει **είναι πλήρες** *(συνθήκη γραμμής 14)* καλεί τον **completer** *(γραμμή 16)* .

2.2.2 Οι predictor , scanner και completer

Οι τελεσταίοι predictor , scanner και completer περιγράφονται στο αρχείο πηγαίου κώδικα operators.c .

Ο τελεσταίος predictor δημιουργεί ένα καινούργιο expression στο τρέχον chart για κάθε μη τερματικό που δεν είναι μέρος του λόγου και περιμένει να αναγνωριστεί . Το expression αυτό έχει στο δεξί του μέλος το μη τερματικό αυτό σύμβολο και αριστερά όλα τα δυνατά derivations (δημιουργείται ένα καινούργιο expression για κάθε derivation) του συμβόλου αυτού . Οι πληροφορίες για την θέση του συμβόλου στο input εξαρτώνται από την θέση της επόμενης λέξης που περιμένει να αναγνώριστεί , έτσι για παράδειγμα κρατούμε πληροφορία ότι το τάδε σύμβολο περιμένει να αναγνωριστεί στην θέση q .

Ο τελεσταίος scanner δημιουργεί ένα καινούργιο expression στο επόμενο chart για κάθε μη τερματικό που είναι μέρος του λόγου και αναγνωρίστηκε στο input . Το expression αυτό έχει στο δεξί μέρος το μή τερματικό και στο αριστερό το τερματικό με το οποίο έγινε η αναγνώριση στο input . Οι πληροφορίες για την θέση του μη τερματικού στο input εξαρτώνται από την θέση της λέξης που αναγνωρίστηκε , έτσι για παράδειγμα κρατούμε πληροφορία ότι το τάδε μέρος του λόγου αναγνωρίστηκε μεταξύ των θέσεων q και $q + 1$.

Ο τελεσταίος completer δημιουργεί ένα καινούργιο expression στο τρέχον chart για κάθε σύμβολο(είτε είναι μέρος του λόγου είτε όχι) που έχει αναγνωριστεί μέχρι το τρέχον σημείου του input . Το καινούργιο expression είναι ταυτόσημο με κάποιο προηγούμενο από αυτό expression που είχε προσθεθεί σε αυτό ή στα προηγούμενα charts και περίμενε να αναγνωρίσει το εν λόγο σύμβολο στην συγκεκριμένη θέση . Σε σχέση με το input το καινούργιο expression βρίσκεται τόσες θέσεις δεξιότερα όσες είχαν αναγνωριστεί από το αρχικό ολοκληρωμένο expression .

2.3 Έξοδος του προγράμματος , δημιουργία συντακτικού δέντρου .

Για κάθε expression που δημιουργείται από τον completer αποθηκεύεται πληροφορία σχετικά με τα expressions που αναγνωρίζουν τα συντακτικά μέρη του δεξιού του μέλους .

Αφού λοιπόν ολοκληρωθεί ο αλγόριθμος early κάθε expression στο τελευταίο chart που έχει στο αριστερό μέλος το αρχικό σύμβολο της γραμματικής (κατα σύμβαση το $< S >$) και έχει αναγνωρίσει ως και την τελευταία λέξη του input μπορεί να μας δώσει και από ένα συντακτικό δέντρο για την προτασή μας αν επισκεφθούμε αναδρομικά τα expressions που αποθηκεύτηκαν από τον completer .

Η διαδικασία αυτή περιγράφεται στο αρχείο πηγαίου κώδικα output.c στα procedures : print-parse-trees-to-tex , print-tree και print-node .

3 Παραδείγματα εξόδου του προγράμματος

Στον φάκελο examples που δόθηκε μαζί με τον πηγαίο κώδικα περιέχονται παραδείγματα εξόδου του προγράμματος , που χρησιμοποιούν το αρχείο γραμματικής "rules" από το site του μαθήματος για την αναγνώριση των φράσεων "Book that flight" και "Does this flight include a meal" .

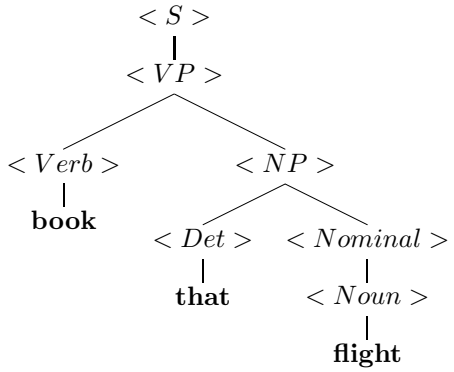
Parsing of the phrase : "book that flight" , with CFG from file "rules" using the early-parsing algorithm

THIS FILE IS AUTOMATICALLY GENERATED FROM THE EARLY PARSER
created by Nick Papoulias for the purposes of the NLP course
in the spring semester of 2005 at TUC .

June 9, 2005

Contents

1 Parse Trees



2 Contex Free Grammar from file : "rules"

- <S>→ <NP> <VP>
- <S>→ <Aux> <NP> <VP>
- <S>→ <VP>
- <NP>→ <Det> <Nominal>
- <NP>→ <Proper-Noun>
- <VP>→ <Verb>
- <VP>→ <Verb> <NP>
- <Aux>→ does
- <Det>→ that
- <Det>→ this
- <Det>→ a
- <Nominal>→ <Noun>

- <Nominal>→ <Noun> <Nominal>
- <Noun>→ book
- <Noun>→ flight
- <Noun>→ meal
- <Noun>→ money
- <Proper-Noun>→ Houston
- <Proper-Noun>→ TWA
- <Verb>→ book
- <Verb>→ include
- <Verb>→ prefer
- <Prep>→ from
- <Prep>→ to
- <Prep>→ on

3 EARLY chart

3.1 CHART[0]

s0 :	<gamma> →	. <S>	[0,0]	dummy-start-state	[]
s1 :	<S> →	. <NP> <VP>	[0,0]	predictor	[]
s2 :	<S> →	. <Aux> <NP> <VP>	[0,0]	predictor	[]
s3 :	<S> →	. <VP>	[0,0]	predictor	[]
s4 :	<NP> →	. <Det> <Nominal>	[0,0]	predictor	[]
s5 :	<NP> →	. <Proper-Noun>	[0,0]	predictor	[]
s6 :	<VP> →	. <Verb>	[0,0]	predictor	[]
s7 :	<VP> →	. <Verb> <NP>	[0,0]	predictor	[]

3.2 CHART[1]

s8 :	<Verb> →	book	[0,1]	scanner	[]
s9 :	<VP> →	<Verb>	[0,1]	completer	[s8]
s10 :	<VP> →	<Verb> . <NP>	[0,1]	completer	[s8]
s11 :	<S> →	<VP>	[0,1]	completer	[s9]
s12 :	<NP> →	. <Det> <Nominal>	[1,1]	predictor	[]
s13 :	<NP> →	. <Proper-Noun>	[1,1]	predictor	[]
s14 :	<gamma> →	<S>	[0,1]	completer	[s11]

3.3 CHART[2]

s15 :	<Det> →	that	[1,2]	scanner	[]
s16 :	<NP> →	<Det> . <Nominal>	[1,2]	completer	[s15]
s17 :	<Nominal> →	. <Noun>	[2,2]	predictor	[]
s18 :	<Nominal> →	. <Noun> <Nominal>	[2,2]	predictor	[]

3.4 CHART[3]

s19 :	<Noun> →	flight	[2,3]	scanner	[]
s20 :	<Nominal> →	<Noun>	[2,3]	completer	[s19]
s21 :	<Nominal> →	<Noun> . <Nominal>	[2,3]	completer	[s19]
s22 :	<NP> →	<Det> <Nominal>	[1,3]	completer	[s15 s20]
s23 :	<Nominal> →	. <Noun>	[3,3]	predictor	[]
s24 :	<Nominal> →	. <Noun> <Nominal>	[3,3]	predictor	[]
s25 :	<VP> →	<Verb> <NP>	[0,3]	completer	[s8 s22]
s26 :	<S> →	<VP>	[0,3]	completer	[s25]
s27 :	<gamma> →	<S>	[0,3]	completer	[s26]

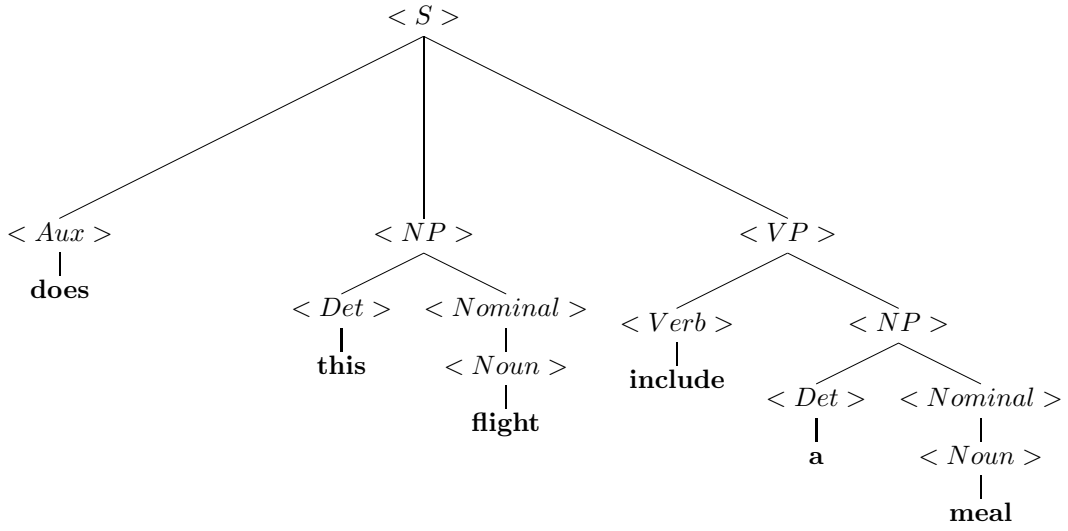
Parsing of the phrase : "does this flight include a meal" , with CFG from file "rules" using the early-parsing algorithm

THIS FILE IS AUTOMATICALLY GENERATED FROM THE EARLY PARSER
created by Nick Papoulias for the purposes of the NLP course
in the spring semester of 2005 at TUC .

June 9, 2005

Contents

1 Parse Trees



2 Contex Free Grammar from file : "rules"

- <S>→ <NP> <VP>
- <S>→ <Aux> <NP> <VP>
- <S>→ <VP>
- <NP>→ <Det> <Nominal>
- <NP>→ <Proper-Noun>
- <VP>→ <Verb>
- <VP>→ <Verb> <NP>
- <Aux>→ does
- <Det>→ that

- <Det>→ this
- <Det>→ a
- <Nominal>→ <Noun>
- <Nominal>→ <Noun> <Nominal>
- <Noun>→ book
- <Noun>→ flight
- <Noun>→ meal
- <Noun>→ money
- <Proper-Noun>→ Houston
- <Proper-Noun>→ TWA
- <Verb>→ book
- <Verb>→ include
- <Verb>→ prefer
- <Prep>→ from
- <Prep>→ to
- <Prep>→ on

3 EARLY chart

3.1 CHART[0]

s0 :	<gamma> →	. <S>	[0,0]	dummy-start-state	[]
s1 :	<S> →	. <NP> <VP>	[0,0]	predictor	[]
s2 :	<S> →	. <Aux> <NP> <VP>	[0,0]	predictor	[]
s3 :	<S> →	. <VP>	[0,0]	predictor	[]
s4 :	<NP> →	. <Det> <Nominal>	[0,0]	predictor	[]
s5 :	<NP> →	. <Proper-Noun>	[0,0]	predictor	[]
s7 :	<VP> →	. <Verb>	[0,0]	predictor	[]
s8 :	<VP> →	. <Verb> <NP>	[0,0]	predictor	[]

3.2 CHART[1]

s6 :	<Aux> →	does	[0,1]	scanner	[]
s9 :	<S> →	<Aux> . <NP> <VP>	[0,1]	completer	[s6]
s10 :	<NP> →	. <Det> <Nominal>	[1,1]	predictor	[]
s11 :	<NP> →	. <Proper-Noun>	[1,1]	predictor	[]

3.3 CHART[2]

s12 :	<Det> →	this	[1,2]	scanner	[]
s13 :	<NP> →	<Det> . <Nominal>	[1,2]	completer	[s12]
s14 :	<Nominal> →	. <Noun>	[2,2]	predictor	[]
s15 :	<Nominal> →	. <Noun> <Nominal>	[2,2]	predictor	[]

3.4 CHART[3]

s16 :	<Noun> →	flight	[2,3]	scanner	[]
s17 :	<Nominal> →	<Noun>	[2,3]	completer	[s16]
s18 :	<Nominal> →	<Noun> . <Nominal>	[2,3]	completer	[s16]
s19 :	<NP> →	<Det> <Nominal>	[1,3]	completer	[s12 s17]
s20 :	<Nominal> →	. <Noun>	[3,3]	predictor	[]
s21 :	<Nominal> →	. <Noun> <Nominal>	[3,3]	predictor	[]
s22 :	<S> →	<Aux> <NP> . <VP>	[0,3]	completer	[s6 s19]
s23 :	<VP> →	. <Verb>	[3,3]	predictor	[]
s24 :	<VP> →	. <Verb> <NP>	[3,3]	predictor	[]

3.5 CHART[4]

s25 :	<Verb> →	include	[3,4]	scanner	[]
s26 :	<VP> →	<Verb>	[3,4]	completer	[s25]
s27 :	<VP> →	<Verb> . <NP>	[3,4]	completer	[s25]
s28 :	<S> →	<Aux> <NP> <VP>	[0,4]	completer	[s6 s19 s26]
s29 :	<NP> →	. <Det> <Nominal>	[4,4]	predictor	[]
s30 :	<NP> →	. <Proper-Noun>	[4,4]	predictor	[]
s31 :	<gamma> →	<S>	[0,4]	completer	[s28]

3.6 CHART[5]

s32 :	<Det> →	a	[4,5]	scanner	[]
s33 :	<NP> →	<Det> . <Nominal>	[4,5]	completer	[s32]
s34 :	<Nominal> →	. <Noun>	[5,5]	predictor	[]
s35 :	<Nominal> →	. <Noun> <Nominal>	[5,5]	predictor	[]

3.7 CHART[6]

s36 :	<Noun> →	meal	[5,6]	scanner	[]
s37 :	<Nominal> →	<Noun>	[5,6]	completer	[s36]
s38 :	<Nominal> →	<Noun> . <Nominal>	[5,6]	completer	[s36]
s39 :	<NP> →	<Det> <Nominal>	[4,6]	completer	[s32 s37]
s40 :	<Nominal> →	. <Noun>	[6,6]	predictor	[]
s41 :	<Nominal> →	. <Noun> <Nominal>	[6,6]	predictor	[]
s42 :	<VP> →	<Verb> <NP>	[3,6]	completer	[s25 s39]
s43 :	<S> →	<Aux> <NP> <VP>	[0,6]	completer	[s6 s19 s42]
s44 :	<gamma> →	<S>	[0,6]	completer	[s43]