

Parsningsalgoritmer

OH-serie 1: introduktion

<http://stp.lingfil.uu.se/~matsd/uv/uv09/pa/>



UPPSALA
UNIVERSITET

Mats Dahllöf
Institutionen för lingvistik och filologi
April 2009

1

Algoritmer

- "en systematisk procedur som i ett ändligt antal steg anger hur man utför en beräkning eller löser ett givet problem" (NE).
- De ses oftast som deterministiska: de väljer i varje läge ett bestämt nästa steg.
- Ibland räknar man även icke-deterministiska system som algoritmer (exempel snart: transitionssystem). Blir deterministiska med hjälp av styrande mekanism.
- Den "abstrakta principen" bakom ett program. (En algoritm kan implementeras på olika sätt.)

3

Parsningsalgoritmer II

- Olika urval analyser: alla (om grammatiken tillåter ambiguitet), "bästa" (utifrån någon rangordning), en enda, etc.
- Många parsningsalgoritmer beskrivs i icke-deterministiska termer: de tillåter då olika vägar till olika analyser (vid ambiguitet) eller till en och samma analys. (Exempel: shift-reduce-algoritmerna som vi skall se på längre fram.)
- "Recognition" — när man bara får ett ja eller nej till frågan om grammatiken kan generera strängen.

5

Parsning: två problem/förtjänster

Viktigt för parsningskomponenter i tillämpningar (typiska önskemål):

- Man vill ha en analys:
Om man hanterar autentiskt språk stöter man på "överraskande" fenomen. **Robusthet:** systemet gör något vettigt med allt det hittar.
- Man vill inte ha många analyser:
Språk är genuint **flertydigt** och ofta artificiellt (och extremt) flertydigt givet en grammatik. **Disambiguering:** Systemet väljer en analys att gå vidare med (på något smart sätt).

7

Parsningsalgoritmer: inledning

- Vad är *parsning*?
- Vad är en *algoritm*?
- Allmänna egenskaper hos algoritmer för syntaktisk analys:
 - Slag av grammatik och representation.
 - Täckningsgrad.
 - "Strategier" för fraskonstruktion.
 - Önskvärda egenskaper
- Tillämpningar

2

Parsningalgoritmer I

- *Parsing* — engelska ordet för satslösning.
Teknisk bemärkelse (och i svenskan): syntaktisk analys i dator, både av naturligt och artificiellt språk.
- Parsningsalgoritm: En algoritm som givet en typ av grammatisk analys, räknar ut en mängd analyser (typiskt syntaxträd) av en given en sträng av symboler.
- Bygger ofta på en formell grammatik.

4

Tillämpningar av parsning

Många språkteknologiska tillämpningar inkluderar parsning:

- Språkgranskning (stavning, grammatik, stil etc.)
- Dokumentsökning
- Informationsextraktion
- Maskinöversättning
- Talsyntes

Syntaktisk information är nästan alltid värdefull. Frågan är om man vill betala priset i tid/datorkraft och kostnaden för själva parsern.

6

Grammatiker

Formella deklarativa grammatiker: en typ av regelsystem.

- Vanligt inom DL/ST: CFG. Ibland kompletterade med särdrag.
- En grammatik är ett (mycket komplext) villkor på hur grammatiska strukturer får se ut.
- Den grammatiska formalismen begränsar hur detta får villkor se ut.
- Indirekt tillåts/förbjuds strängar av terminalsymboler (ord).
- CFG är **kategoriskt** tillåtande/förbjudande.

8

Interpreterande parsningsalgoritmer



- Grammatiken: deklarativ och läses som den är som en resurs i en databas.
- Algoritmen är generell för en viss typ av grammatik.

9

Separation mellan grammatik och algoritm

- Både interpreterande och kompillerande parsning håller grammatik och algoritm åtskilda.
- Fördel: Grammatik och parsningmjukvara kan utvecklas, användas, köpas och säljas oberoende av varandra.
- Detta arbetssätt stödjer och förutsätter standardisering (i enlighet med valda grammatikformalismen).
- Grammatiken kan ansluta till annat språkvetenskapligt arbete.

11

Klassifikation/analys av parsningsalgoritmer

Parsningsalgoritmer kan beskrivas ur ett antal synvinklar.

- Principer för regeltillämpning.
- Djupet eller bredden först?
- Alternativhantering.
- Läsordning (i praktiken nästan alltid vänster-höger).
- Minnesutnyttjande.

Vi förutsätter fortsättningsvis att grammatiken är en CFG.

13

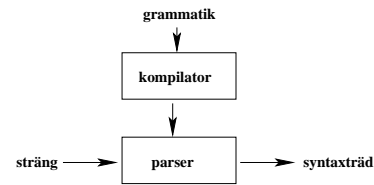
Regelapplikation "bottom-up"

Bottom-up, datadrivet.

- Vi bygger träden nedifrån; vi börjar med orden.
- Träden är lokalt fullständiga.
- Vi applicerar regler utifrån de konstituenten vi lyckats verifiera i inputsträngen. (Vi ser på reglernas högerled.)
- T.ex. om vi funnit en NP och en VP (i direkt följd) och har regeln $S \rightarrow NP VP$ så drar vi slutsatsen att vi har en S.

15

Kompilerande parsningsalgoritmer



- Grammatiken kompileras, och "ses" inte mer av parsern.
- Abstrakt algoritm och grammatik "sammanvävs".

10

Parsning utan grammatik

- Grammatiker vägleder i typiska fall parsningsalgoritmer.
- Parsningsalgoritmer kan vägledas på andra sätt, t.ex. av en klassificerare som bedömer vilken operation som är smartast att göra i varje givet läge (beskrivet av lämpliga särdrag). Bakomliggande resurser:
 - icke-deterministisk parser (transitionssystem)
 - data i form av parsade uttryck (trädbank)
 - maskininlärningsalgoritmer
- Kommande kurs: *Tekniker för storskalig parsning*.

12

Regelapplikation "top-down"

Top-down/prediktivt/förväntningsdrivet.

- Vi bygger träden uppifrån; vi börjar med startsymbolsnoden.
- När vi läser inputord så får de en bestämd (hypotetisk) plats i meningen.
- Vi tillämpar regler utifrån vilken kategori vi letar efter. (Vi ser på reglernas vänsterled.)
- T.ex. om vi söker efter en S och har regeln $S \rightarrow NP VP$, så letar vi utifrån den efter en NP och en VP (vid en given plats i input).

14

Djup eller bredd först?

- "Djupet-först": Bygga så mycket av trädet som går innan man går vidare i inputsträngen. "Djupet-först" blir (så gott som) automatiskt följd av en "top-down" strategi. (Man kan inte predicera förrän föregående konstituent är klar.)
- "Bredden-först": Bygga struktur på bredden. ("Bottom-up" mer flexibelt.)

16

Alternativhantering

Vid alternativa möjligheter, t.ex. när två regler kan tillämpas, eller två lexikoningångar finns:

- Deterministisk strategi: Gör ett val, och strunta i alternativen.
- "Backtracking": pröva "första bästa" alternativ, och gå tillbaka senare (om det behövs).
- "Parallellprocessning": Undersök alla "vägar", som jämbördiga möjligheter.
- "Look ahead": Välj genom att kika framåt.

17

"Värdering" av algoritmer

- Korrekthet: Analyserna bör stämma med grammatiken.
- Fullständighet: Återfinns alla möjliga analyser?
- Effektivitet, m.a.p. tid och minneskrav.
(Komplexitet: Hur påverkas tidsåtgången av längre input?)
- Grammatikens egenskaper (stöd för lingvistiskt utvecklingsarbete). Begriplighet. Möjlighet att utnyttja befintliga analyser, etc.
- Tillgång på implementationer och verktyg.

19

Systemets sätt att minnas vad det gjort

- Minimalt "minne": endast det träd som är under uppbyggnad hålls i minnet. T.ex. vid backtracking där "avvisade" delträd inte sparas.
Problem: risk för dubbelarbete.
- Delanalyser bokförs i s.k. (välformade) delsträngstabeller eller charts (diagram). "Chartparsning": alla tänkbara delanalyser lagras.
Problem: viss kostnad för lagring och åtkomst.

18

Parsning av artificiella språk

- Parsning är också viktigt för tillämpningar som programspråk, HTML-tolkning, T_EX-kompilering.
- Entydiga språk, anpassade för datorbearbetning.
- Typiskt utformade så att alternativa analyser elimineras snabbt. Gör dem snabbparsade.
- Krav på att stora "texter" skall bearbetas on-line.
- Andra typer av algoritmer används än för DL/ST.
Mycket snabba. Behöver ej hantera ambiguitet, t.ex.

20