Finite-State Technology

Teil II:
Formale Sprachen

Programm

- Erster größerer Teil des Seminars mit einem Überblick über die formalen Grundlagen der Finite-State Technology
- Einstieg: Wo ordnen sich reguläre Sprachen innerhalb der formalen Sprachen ein? (heute)
- FSA's als eine Beschreibung regulärer Sprachen (08.05.2007)
- FST's als eine Beschreibung regulärer Relationen (15.05.2007)

Wiederholung

- Ausgangspunkt: Das Problem von Analyse und Generierung natürlichsprachlicher Phonologien und Morphologien.
- Das formale Problem: Phonologische und morphologische Regeln werden in der theoretischen Linguistik durch (sequentiell geordnete) kontextsensitive Regeln beschrieben (z.B. n -> m / _ p).
- Kontextsensitive Regeln sind komputationell nicht effizient zu behandeln (Komplexitätsprobleme).
- Durchbruch: Johnson (1972) und Kaplan/Kay (um 1980) entdecken unabhängig voneinander eine Eigenschaft phonologischer Regeln: sie lassen sich durch reguläre Relationen beschreiben.
- Was ist das eigentlich und warum ist das ein Durchbruch?

Überblick für die heutige Vorlesung

- Was ist eine formale Sprache?
- Wie lassen sich formale Sprachen beschreiben?
- Was ist die einfachste formale Sprache?
- Welche anderen formalen Sprachen gibt es?
- Was ist der Zusammenhang zwischen formalen Sprachen und natürlichen Sprachen?

Strings, Sprachen und Grammatiken

- Ganz allgemein können Sprachen einfach als Mengen von Strings über einem Alphabet A aufgefaßt werden.
- Einige Mengen von Strings können durch formale Grammatiken beschrieben werden.

Strings

- Für die Bildung von Strings braucht man eine endliche Menge A von Symbolen – dies ist das Alphabet einer Sprache (bei manchen Autoren wird immer ∑ verwendet).
- Definition: Ein String ist eine endliche Aneinanderkettung (Konkatenation) von Symbolen aus einem Alphabet A.
- Die Konkatenation hat entweder kein eigenes Symbol oder •.
- Jede Menge A (von Strings) in Verbindung mit der Operation der Konkatenation wird (unter bestimmten weiteren Bedingungen) auch als Monoid A* bezeichnet.

Strings - Beispiel

- Alphabet A = {0, a, b}
- Monoid A* = {0, a, b, aa, ab, bb, aaa, bbb, aaaa, bbbb, aabbba, bbbbbbaaabbbbbbb etc.}
- In FSM besteht das Alphabet aus *atomic expressions*.
- Frage: Ist dieses Alphabet endlich?
- Was ist der Unterschied zwischen:

  a) read regex [s t r i n g]; und

  b) read regex [string];?

Sprachen

- Eine Sprache L über A ist nun jede beliebige Teilmenge M von A*.
- Achtung: Strings sind Ketten von Elementen, Sprachen sind Mengen von Ketten von Elementen
- Die Menge aller Strings eines Alphabets, die eine bestimmte Länge haben, wird durch Exponentialnotation dargestellt.
- A^1 = die Menge aller Strings der Länge 1 ({a, b, c, ...})
- A^2 = die Menge aller Strings der Länge 2 ({aa, ab, ba, bb, ...} = {a•a, a•b, b•a, b•b, ...} })
- Komplexe Sprachen können aus einfachen durch bestimmte Boolesche Operationen gebildet werden.
- Die Art und Anzahl der möglichen Operationen wird durch den Typ der Sprache beschränkt.

Sprachen - Beispiel

- Sprache LV: {o, s, t, mus, tis, nt} (=lateinische Verbalsuffixe)
- Sprache VS: {ama, vide} (=latenische Verbalstämme)
- Sprache LAT = VS • LV = jedes Element aus VS konkateniert mit jedem Element aus LV =

  {amao, amas, amat, amamus, amatis, amant, video, vides, videt, videmus, videtis, vident}
- Frage: Ist LAT eine Teilsprache der natürlichen Sprache Latein?

- Einige Abfolgen von Strings sind in einer Sprache enthalten (die grammatischen Strings), andere sind es nicht (die ungrammatischen Strings). Die latenische Sprache enthält nicht den String {amao}.

Grammatiken

- Key concept Formal Language (Jurafsky/Martin 2000:39): “A model which can both generate and recognize all and only the strings of a formal language acts as a *definition* of the formal language.“
- Es gibt zwei Klassen von Darstellung, die als formale Grammatik benutzt werden können: Automaten (abstrakte Computationsmittel) und *String rewriting systems* (eigentliche formale Grammatik).
- Beide definieren Sprachen.
- Allgemein gesagt: Formale Grammatiken sind deduktive Systeme mit Ableitungsregeln der Art A -> B.
- Manche Sprachen haben keine Grammatik.

Reguläre Sprachen

- These: Ein großer Teil natürlichsprachlicher Phänomene der Phonologie und Morphologie kann durch die einfachsten formalen Sprachen und Automaten beschrieben werden.
- Die einfachsten formalen Sprachen sind die regulären Sprachen.

EINE Definition regulärer Sprachen

- Partee (1993:463):

  Given an alphabet $\sum$

  1) 0 is a regular language

  2) for any string x ε $\sum$*, {x} is a regular language

  3) if A and B are regular languages, so is A $\cup$ B

  4) if A and B are regular languages, so is AB

  5) if A is a regular language, so is A*

  Nothing else is a regular language unless its being so follows from 1-5.

Eigenschaften regulärer Sprachen

- “A set operation such as union has a corresponding operation on finite-state networks only if the set of regular relations and languages is closed under that operation. Closure means that if the sets to which the operation is applied are regular, the result is also regular, that is, encodable as a finite-state network.“ (Beesley/Karttunen 2003:54)
- Reguläre Sprachen sind geschlossen unter: Vereinigung, Konkatenation, Iteration, Reversion, Schnitt, Subtraktion, Komplement.
- Der Vorteil dieser Eigenschaften wird später im Vergleich mit regulären Relationen und anderen Grammatiken klar.
- Allgemein: Solche Eigenschaften müssen jeweils für jeden Typ von Sprache bewiesen werden (Ein wenig machen wir das bei den FSA‘s).

Eigenschaften - Beispiele

- Vereinigung:
  Die Vereinigung L1 ∪ L2 = {a | a ε L1 oder a ε L2}
- Komplement:
  \L = {a | a nicht ε L}
- Iteration:
  Der Schnitt L1 ∩ L2 = {a | a ε L1 und a ε L2}

Eigenschaften regulärer Sprachen - Beispiel

- Ziel: Wir wollen eine Syntax für das Englische entwerfen.
- Angenommen, wir haben schon ein paar Strings aus dem Englischen extrahiert: Menge S = {he, hopes, that, this, works}.
- Dies erlaubt folgende grammatikalische Teilsprache des Englischen:
  Sprache L1 = {he • hopes • that • this • works}.
- Mittels eines Lexikons wird nun jeder dieser Strings mit seiner syntaktischen Kategorie konkateniert. Dabei stellt man Homonymie fest (durch Disjunktionen angegeben):
- Sprache L2 = {{he_Pro} {hopes_N | hopes_V} {that_Pro | that_Conj | that_Det} {this_Det | this_Pro} {works_N | works_V}}

Eigenschaften regulärer Sprachen - Beispiel

- Diese erweiterte Sprache erlaubt nun mehrere Kombinationen:
  {
  he_Pro • hopes_V • that_Conj • this_Pro • works_V (☺) |
  he_Pro • hopes_N • that_Pro • this_Det • works_N (☹) |
  he_Pro • hopes_V • that_Det • this_Det • works_N (☹) |
  ... }
- Nur einige dieser Kombinationen sind im Englischen aber grammatikalisch (☺).
- Frage: Wie kann man die grammatischen (syntaktisch zulässigen) Kombinationen extrahieren?

Eigenschaften regulärer Sprachen - Beispiel

- EINE Lösung arbeitet mit *constraints*, z.B.: *constraint1* = {that_Det this_Det} (diese Kombination soll nicht erlaubt sein).
- Die ungrammatikalischen Kombinationen werden dann einfach durch Subtraktion der *constraints* aus L herausgenommen:
- Sprache L – *constraint*1 =
  {
  he_Pro • hopes_V • that_Conj • this_Pro • works_V (☺) |
  ~~he_Pro • hopes_N • that_Pro • this_Det • works_N (☹) |~~
  he_Pro • hopes_V • that_Pro • this_Det • works_N (☹) |
  ... }
- Weitere *constraints* sind noch nötig (Roche/Schabes 1997:I)

Beschreibung regulärer Sprachen I: Reguläre Ausdrücke

- Reguläre Sprachen können durch Grammatiken mit Ableitungsregeln beschrieben werden.
- Für reguläre Sprachen gibt es jedoch zusätzlich die algebraische Notation der regulären Ausdrücke.
- In der Notation von FST können reguläre Ausdrücke so aussehen: a, (a), a+, a*, ab, a-b, a^2b^5...
- Komplexe reguläre Ausdrücke können aus einfachen zusammengesetzt werden - sie haben keine erweiterte Ausdruckskraft, vereinfachen aber die Arbeit.
- Für Linguisten richtet sich die Wahl der Notation (reguläre Ausdrücke oder Grammatik mit Ableitungsregeln) nach der Art der zu beschreibenden Daten. Eine Kombination ist ebenfalls möglich.

Komplexe reguläre Ausdrücke

- Beispiel aus Beesley/Karttunen (2003:64): *Restriction operator*.
- a => b_c bedeutet: Wenn irgendwo ein a auftaucht, dann darf es nur direkt nach einem b und direkt vor einem c stehen.
- a => b_c = ~[[~[?* b] a ?*] | [?* a ~[c ?*]]]
  Zu lesen als: Es ist nicht der Fall, daß der erste Teil der Disjunktion ([~[?* b] a ?*]) wahr ist oder daß der zweite Teil der Disjunktion ([?* a ~[c ?*]]) wahr ist oder daß beide Teile wahr sind.
  Erster Teil der Disjunktion bedeutet: Es steht kein b direkt vor a.
  Zweiter Teil der Disjunktion bedeutet: Es steht kein c direkt nach a.
- Das „?“ kann in diesem Beispiel sowohl als *any symbol* als auch als *unknown symbol* interpretiert werden. (Beesley/Karttunen 2003:2.3.4)

Komplexe reguläre Ausdrücke

- Fazit: Eine kleine Sprache mit mehreren dieser Restriktionen ist mit einfachen regulären Ausdrücken schon nicht mehr zu lesen.
- Andere komplexe Ausdrücke wie Regeln (*replacements*) der Art a -> b sind genauso aus einfachen regulären Ausdrücken aufgebaut, werden aber erst später bei den regulären Relationen behandelt.

Beschreibung regulärer Sprachen II: Reguläre Grammatiken

- Formale Grammatiken mit Ableitungsregeln sind eine Art der Definition formaler Sprachen.
- Grammatiken sind Quadrupel G <S, P, T, N> aus einem Axiom S (Startsymbol), Ableitungsregeln bzw. Produktionen P, einer Menge T von terminalen Symbolen und einer Menge N von nichtterminalen Symbolen.
- Frage: Woraus können N und T in der Linguistik bestehen?

  Beispiel Syntax: T enthält die Wörter, N enthält die syntaktischen Kategorien (V, VP, NP etc.).

  Beispiel Morphologie: T enthält die Morpheme, N enthält die morphologischen Kategorien (Kasus, Numerus, Genus etc.).

Beschreibung regulärer Sprachen II: Reguläre Grammatiken

- Das generelle Schema einer Regel ist A → B. Wenn A,B = (N ∪ T), dann ist diese Schreibweise äquivalent zu (N ∪ T)* x (N ∪ T)*.
- Diese Regeln erlauben prinzipiell unendliche Rekursion.
- Jeder Typ von formaler Grammatik beschränkt diese Regel etwas anders.
- Reguläre Sprachen werden durch sogenannte reguläre Grammatiken beschrieben.

Andere Grammatiken - die Chomsky-Hierarchie

- Eine bekannte Art der Klassifikation von formalen Grammatiken beruht auf Noam Chomsky.
- Reguläre Grammatiken:

  Jede Regel hat die Form N x (T ∪ TN).
- Kontextfreie Grammatiken:

  Jede Regel hat die Form N x (N ∪ T).
- (Kontextsensitive Sprachen haben einen noch höheren Typ, spielen aber in unserem Seminar keine Rolle.)
- Achtung: Nicht jede natürliche Sprache, die mit einer kontextfreien Grammatik beschrieben wird, muß auch kontextfrei sein...

Formale Grammatiken in der Linguistik

- Wenn man eine formale Grammatik wie z.B. eine Programmiersprache definiert hat, kennt man alle in ihr erlaubten Ableitungen (z.B. die erlaubten Programme).
- In der Linguistik kennt man zwar alle erlaubten Ableitungen (Daten) einer natürlichen Sprache, muß aus ihnen eine formale Grammatik aber erst noch ableiten.
- Die Linguistik gliedert sich nun traditionell in verschiedene Teilwissenschaften: Phonetik, Phonologie, Morphologie, Syntax etc.
- Jede dieser Teilwissenschaften befaßt sich mit einer Teilmenge der natürlichen Sprache und kann diese mit einer eigenen formalen Grammatik beschreiben.

Formale Grammatiken in der Linguistik

- Die Wahl der formalen Grammatik wir dabei durch die komplexen Eigenschaften der natürlichen Sprachen und durch praktische und theoretische Anforderungen bestimmt.
- Es kann bewiesen werden, daß sie einige syntaktische Konstruktionen nicht mehr durch eine reguläre Grammatik beschreiben lassen.
- Phonologien und Morphologien lassen sich dagegen durch reguläre Grammatiken beschreiben.
- Auf der anderen Seite werden in der theoretischen Linguistik Phonologien traditionell durch kontextsensitive Grammatiken beschrieben. Morphologien wiederum werden auch mit kontextfreien Grammatiken beschrieben.
- Was hat das für Gründe?

Formale Grammatiken in der Linguistik

- Ein Grund: Die Daten lassen sich einfacher beschreiben. (Siehe das Beispiel mit den komplexen regulären Ausdrücken.)
- Ein anderer Grund: Die natürlichen Sprachen sind vielleicht durch einen höheren Typ von formaler Grammatik charakterisiert.
- Nachteil: Je höher der Typ der Grammatik in der Hierarchie, desto schwieriger ist die Komputation. Praktische Anwendungen werden also erschwert.

Reguläre Grammatiken - Beispiel

- Wir wollen eine reguläre Grammatik schreiben, die folgenden Satz generiert : *He thinks that this works*.

  $S \rightarrow$ he V_t
  $V_t \rightarrow$ thinks Conj
  Conj $\rightarrow$ that Det
  Det $\rightarrow$ this V_i
  $V_i \rightarrow$ works

- Dies ist EIN Beispiel für eine (rechtslineare) reguläre Grammatik.
- Frage: Woraus bestehen N und T?

Beispiel mit kontextfreier Grammatik

- Es gibt immer mehrere Möglichkeiten, eine Grammatik für eine Sprache zu schreiben: *He thinks that this works*.

  S $\rightarrow$ NP VP
  NP $\rightarrow$ N | Pro
  VP $\rightarrow$ VI | VI Conj S
  N $\rightarrow$ PN | Pro | Det
  PN $\rightarrow$ works
  Pro $\rightarrow$ he
  ...

- Dies ist EIN Beispiel für eine kontextfreie Grammatik.

Anderes Beispiel mit kontextfreier Grammatik

- Eine Teilsprache des Russischen für Verben wie *perepisyvatsja*.

  WORD → PREF STEM (SUFF1) SUFF2 SUFF3 (POSTFIX)
  PREF → pere | na | u | vy...
  STEM → pis | risov
  SUFF1 → yv | ov | ev
  SUFF2 → a
  SUFF3 → tj
  POSTFIX → sja

- Dies ist eine weitere (übergenerierende) kontextfreie Grammatik.

Mehrere Grammatiken?

- Ein und dieselbe Menge von Daten kann mit mehreren Grammatiken beschrieben werden. Dies hat zwei Bedeutungen.

  A) Man kann z.B. N und T jeweils unterschiedlich festlegen. Beispiel: In der generativen Grammatik ist die IP Teil von N. In der *Lexical Functional Grammar* gibt es in N keine IP.

  B) Man kann manche Daten mit Grammatiken verschiedenen Typs beschreiben. Dabei gilt: Alle Sprachen niedrigeren Typs sind in Sprachen höheren Typs enthalten, aber nicht umgekehrt. Beispiel: Phonologien lassen sich durch reguläre Relationen und kontextsensitive Grammatiken beschreiben.

Welche Sprachen haben welchen Typ?

- Woher weiß man, von welchem Typ eine bestimmte Sprache ist?
- Kann man das beweisen?

Das Pumping-Lemma

- Beispiel: Woher weiß man, daß eine Sprache regulär ist?
- Positiver Beweis: Man kann eine Grammatik für sie schreiben (reguläre Grammatik oder Automat).
- Falls das nicht gelingt, ist nicht bewiesen, daß die Sprache nicht regulär ist (vielleicht muß man noch etwas länger nachdenken).
- Man kann jedoch beweisen, daß eine Sprache nicht regulär ist.
- Dies macht man mit dem Pumping-Lemma.
- Das Pumping-Lemma wird auch für Sprachen höheren Typs benutzt.
- Diese Beweise sind immer negativ.
- Frage: Wie können Kaplan/Kay (1994) zeigen, daß phonologische Regeln durch reguläre Relationen beschreibbar sind?

Limitation regulärer Sprachen

- Ganz allgemein: Reguläre Grammatiken können keine Sprachen beschreiben, bei denen man ein Gedächtnis braucht.
- Klassisches Beispiel (bei regulären Ausdrücken): 0^n1^n.
- Für solche Beispiele braucht man (mindestens) kontextfreie Grammatiken.
- Analoges Beispiel aus der Informatik: Computerprogramme mit gleicher Anzahl von „(„ und „)“.
- Analoges Beispiel aus natürlichen Sprachen: „wenn...dann“, „entweder... oder“.
- Frage: Wo finden sich solche Beispiele in der Linguistik? Unterscheiden sich Syntax und Semantik von Phonologie und Morphologie?

Beispiel *center embedding*

- *The cat died. The cat the dog chased died. The cat the dog the rat bit chased died. ...*
- RE: (the + common noun)^n (transitive verb)$^{n-1}$ intransitive verb
- Kontextfreie Grammatik mit Ableitungsregeln (aus Christiansen 1994):

  S → NP VP
  NP → N rel
  NP → N
  VP → V
  rel → NP V(trans)

Beispiel *nested dependencies*

- “Anyone$_1$ who feels that if$_2$ so-many$_3$ more$_4$ students$_5$ whom we$_6$ haven‘t$_6$ actually admired are$_5$ sitting in on the course than$_4$ ones we have that$_3$ the room had to be changed, then$_2$ probably auditors will have to be excluded, is$_1$ likely to agree that the curriculum needs revision.“ (Chomsky and Miller 1963, zitiert aus Partee et al. 1993)
- Struktur dieser Abhängigkeiten: $\{xx^R \mid x \in \{a, b^*\}\}$
- Intersektion mit regulärer Sprache aa*bbaa* ergibt $\{a^nb^2a^n \mid n \geq 1\}$
  Achtung: Diese Sprache ist nicht regulär (☹).
- Frage: War dieser Satz einfach zu parsen?
- Wieviele Abhängigkeiten kann man noch verarbeiten?

Reguläre Sprachen als Modell natürlicher Sprachen?

- Ausgangspunkt: Rekursion ist prinzipiell unendlich.
- Reguläre Grammatiken können iterative Rekursion darstellen, aber keine strukturelle Rekursion (Begriffe von Christiansen 1994).
  (Bsp. iterative Rekursion: *Ich glaube, daß sie denkt, daß...*)
  (Bsp. strukturelle Rekursion: *The cat the dog saw left.*)
- Gibt es aber unendliche Rekursion in natürlichen Sprachen?
- Es scheint eine kognitive Grenze in der Aufnahmefähigkeit zu geben.
- Mit dieser Einschränkung wären alle Bereiche natürlicher Sprache prinzipiell durch reguläre Sprachen zu beschreiben (Christiansen 1994).
- Anderes Beispiel: Anwendung dieses Prinzips in der LFG.

Beispiele für Anwendungen von regulären Sprachen

- Compilerbau
- Pattern matching
- Pattern recognition
- Speech processing (z.B. FSM)
- ...

Ausblick: Grammatiken und Automaten

- Automaten sind eine andere Form der Beschreibung von Sprachen.
- “A set of strings is a finite automaton language if and only if it is a regular language.“ (Partee et al. 1993:464)
- Entsprechung von Sprachen und Automaten:

Reguläre Sprachen - FSAs
Kontextfreie Sprachen - Kellerautomaten (Pushdown Automata)
Kontextsensitive Sprachen - Turingmaschinen

- FSA‘s werden in der nächsten Sitzung besprochen.

Literatur

- Beesley/Karttunen (2003)
- Christiansen (1994)
- Hopcroft et al. (2001)
- Jurafsky/Martin (2000)
- Partee et al. (1993)
- Roche/Schabes (1997)

- Genaue Referenzen auf dem Seminarplan.