MonaSearch – Baumsuche mit monadischer Logik zweiter Stufe

Hendrik Maryns

Sonderforschungsbereich 441
Universität Tübingen
hendrik@sfs.uni-tuebingen.de

14.7.2008

Eberhard Karls
Universität
Tübingen

Übersicht

Übersicht

Übersicht

1. Baumbanken
2. Monadische Logik zweiter Stufe
3. System
   - Bäume und deren Verzweigung
   - Knotenlabels
   - Guided Tree Automata
   - Mona

Ziel

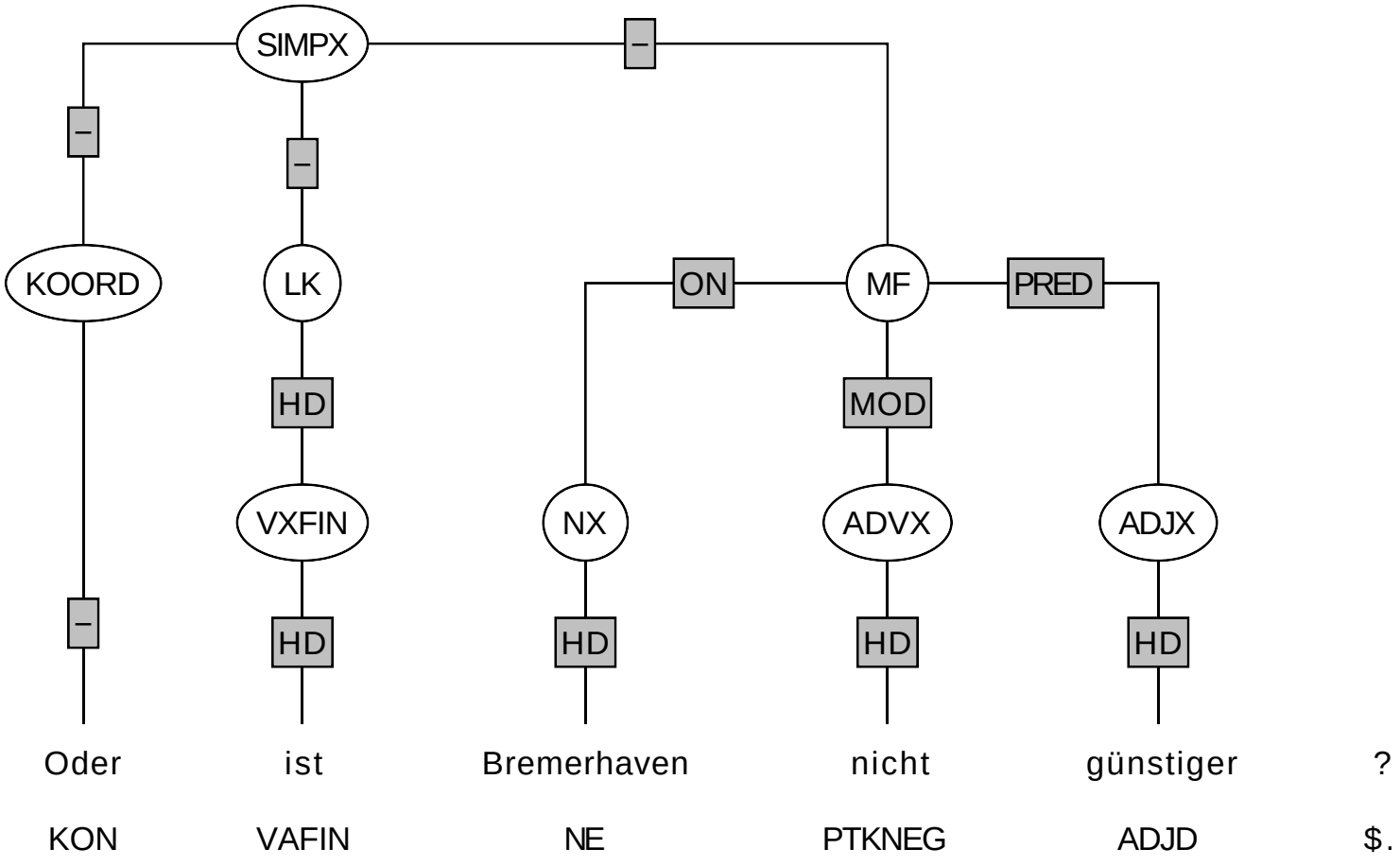
- Linguisten eine Möglichkeit bieten, in großen Datenbanken mit linguistischen Bäumen zu suchen.
- Im Idealfall: ausdrucksstark, schnell und einfach im Gebrauch.
- Benutzung eines Logikformalismus: klar definiert und einfach zu verstehen (aber...).
- Monadische Logik zweiter Stufe (MSO) gewählt, da oft gesehen als dass, was man braucht an Ausdrucksstärke in der Linguistik.
- Implementierung mit Hilfe von Baumautomaten.

Übersicht

Baumbanken

- Viele Linguistische Theorien modellieren syntaktische Struktur eines Satzes als einen Baum.
- Es sind inzwischen, u.a. in Tübingen, große Datenbanken mit solchen Bäumen aufgebaut worden, manche manuell annotiert, manche automatisch.
- Diese ‘Baumbanken’ bieten eine wertvolle Quelle für Belege für Theorien.
- Die Größe der Baumbanken macht es notwendig, Anwendungen zu entwickeln, die das Suchen nach spezifischen Phänomenen ermöglichen, von Hand suchen ist keine Option.

Beispielbaum aus TüBa-D/Z



Überlegungen

- Wesentlich ist, die Anzahl der Ergebnisse möglichst klein zu halten, also nur die Ergebnisse auszugeben, die einen auch wirklich interessieren.
- Deshalb ein Formalismus erwünscht, mit dem man genau das ausdrucken kann, was man will: hohe Ausdrucksstärke.
- Entscheidender Punkt ist die Datenkomplexität: die Größe der Baumbanken rechtfertigt Vorverarbeitung und eine großere Komplexität der Formelverarbeitung.

Übersicht

Monadische Logik zweiter Stufe

- Erweiterung der Logik erster Stufe.
- Hinzu kommen Variablen zweiter Stufe, die Knotenmengen darstellen.
- Quantifizieren ist erlaubt über Variablen erster un zweiter Stufe.

Beispiel:

$$\exists X(\neg\exists x(x \in X))$$

‘Es gibt eine leere Menge.’

Eigenschaften von MSO

- Erweiterung der Logik erster Stufe
- Echt höhere Ausdrucksstärke
- Transitivität binärer Relationen ausdrückbar
- Klare Semantik, die übliche Modelltheoretische

Übersetzungsprozess

- Ein wichtiges Ergebnis von Thatcher und Wright und Doner [**?**, **?**] beschreibt, wie man MSO-Formeln in Baumautomaten umwandeln kann.
- Dieses Verfahren ist leider EXPTIME-vollständig in der Länge der Formel.
- Die Evaluierung eines Baumautomaten auf einen Baum ist aber linear in der Größe des Baumes.
- Da die Datenkomplexität weitaus wichtiger ist, interessant für unsere Zwecke.
- Außerdem werden keine extrem komplexe Eingaben erwartet.

Implementierung

Ein erster Versuch, trotz allem die Automatenumwandlung selber zu implementieren, ist gescheitert.
Es gibt jedoch ein Programm, dass die Übersetzung implementiert: Mona [**?**]. Mona ist in erster Linie für Model Checking gedacht und produziert Automaten in ein eigenes Format. Es gibt jedoch eine kleine Bibliothek, die es unter anderem ermöglicht, Mona zu fragen einen Automat auf einen Baum zu evaluieren.
Mit Dank an Michael Jakl aus Wien, der uns auf diese Möglichkeit hingewiesen hat. (Die Gebrauchsanweisung ist leider ziemlich knapp.)

Übersicht

Abfragesystem

Der Grundlegende Ablauf ist also:

Abfragesystem

Der Grundlegende Ablauf ist also:

1. Der Benutzer führt eine MSO-Formel ein.

Abfragesystem

Der Grundlegende Ablauf ist also:

1. Der Benutzer führt eine MSO-Formel ein.
2. Die Formel wird von Mona in einen Baumautomaten übersetzt.

Abfragesystem

Der Grundlegende Ablauf ist also:

1. Der Benutzer führt eine MSO-Formel ein.
2. Die Formel wird von Mona in einen Baumautomaten übersetzt.
3. Jeder Baum der Baumbank wird dem Automaten übergeben.

Abfragesystem

Der Grundlegende Ablauf ist also:

1. Der Benutzer führt eine MSO-Formel ein.
2. Die Formel wird von Mona in einen Baumautomaten übersetzt.
3. Jeder Baum der Baumbank wird dem Automaten übergeben.
4. Die Bäume, die der Automat akzeptiert, erfüllen die Formel.

Probleme

- Grundlegendes:
  - Mona kann nur mit Binärbäume umgehen, während die Eingabebäume beliebige Verzweigung haben.
  - Mona erwartet, dass die Labels der Bäume nur aus $\{0, 1\}^*$ kommen.
  - Mona definiert eine eigene Variante von Baumautomaten, Guided Tree Automata.
- Technisch:
  - Mona muss extern aufgerufen werden.
  - Die Mona-Bibliothek ist geschrieben in kaum dokumentierter C-Kode, meine GUI in Java.

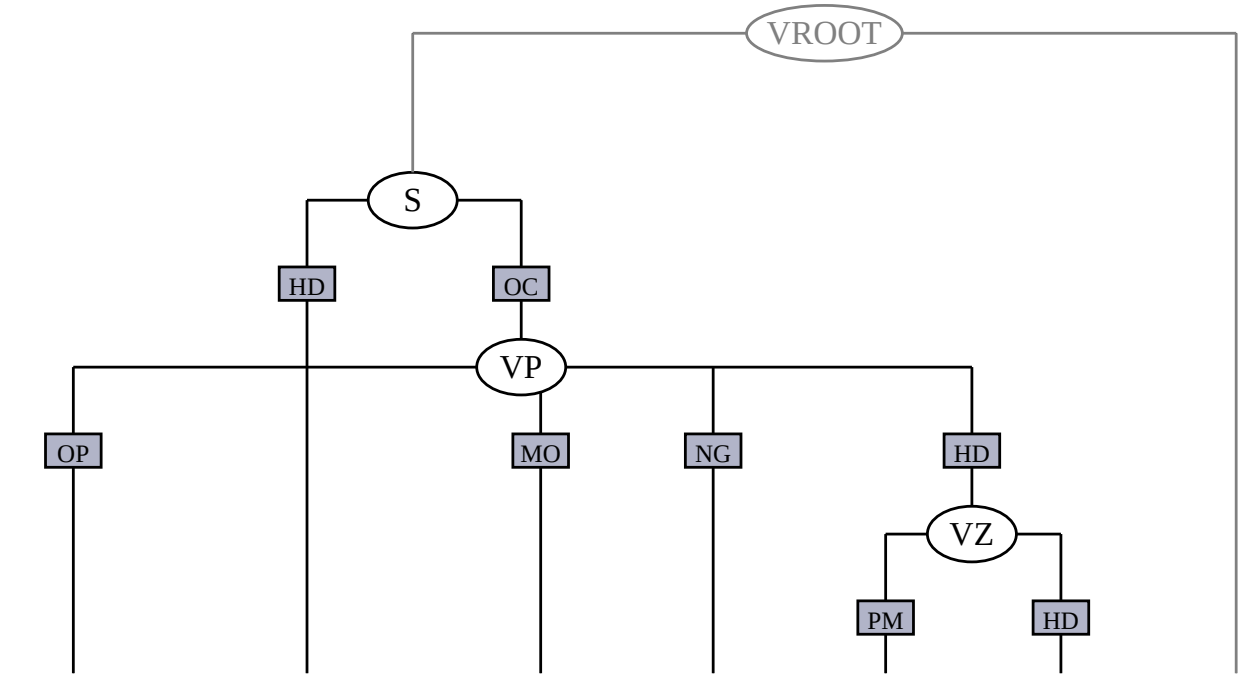
Übersicht

Probleme der baumbank

Der Name ‘Baumbank’ ist eigentlich ein Euphemismus:

- Manche Sätze besitzen mehrere nicht-verbundene Teilbaume (Satzzeichen, aber auch Teilsätze).
- Diese können manchmal übereinander stehen.
- Da die Reihenfolge der Blätter extern zum Baum festgelegt ist, kommen kreuzende Kanten vor.
- Letztlich kommen auch sekundäre Kanten vor, es sind also keine Bäume mehr, sondern DAGs.

Beispielbaum aus TIGER-Korpus mit kreuzenden Kanten



Damit	sei	jedoch	nicht	zu	rechnen	:
PROAV	VAFIN	ADV	PTKNEG	PTKZU	VVINF	$.
	3.Sg.Pres.Konj					
damit	sein	jedoch	nicht	zu	rechnen	:

Binarisierung

Es wird ein bekannter Trick verwendet, um beliebig verzweigende Bäume in Binärbäume um zu wandeln: “erste Tochter, nächste Schwester”:

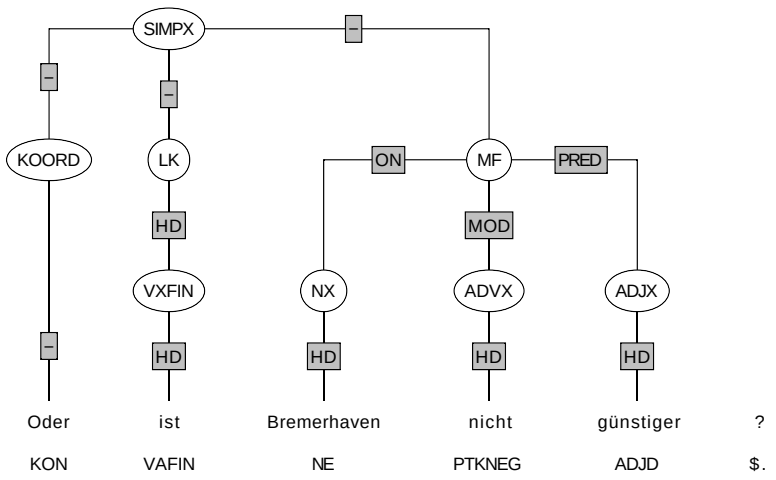
- Ein Knoten x im Originalbaum entspricht einen Knoten x' im Binärbaum.
- Wenn x Kinder hat, wird das am meisten links stehende das linke Kind von x', nach Transformation.
- Wenn x Schwester hat, wird die direkt rechtse Schwester das rechte Kind von x', nach Transformation.
- Die Knoten behalten ihre Label. Kantenlabel werden als zusätzliches Label im Knoten darunter gespeichert.

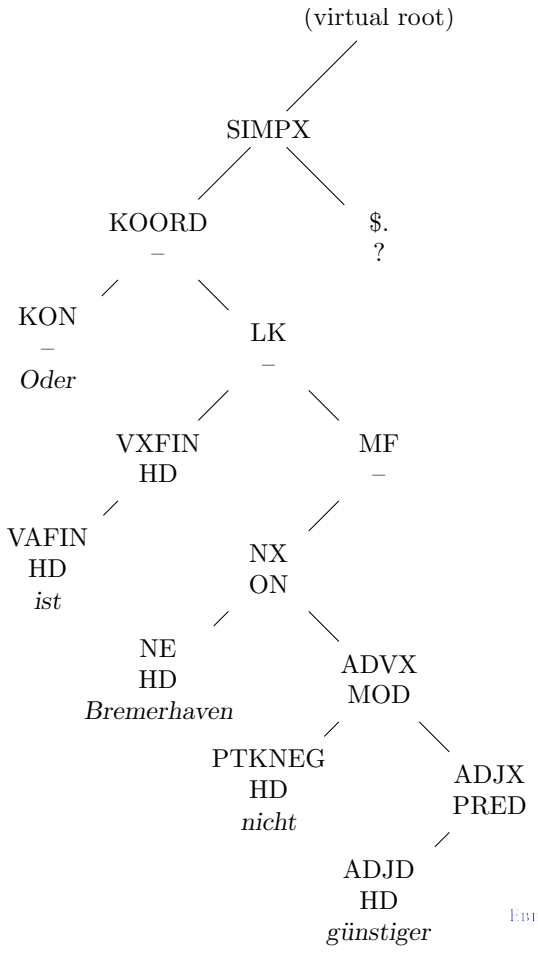
Binarisierung

- Es wird einen virtuellen Wurzelknoten eingeführt, an dem die unverbundenen Teile gehängt werden. Dabei kann Ordnungsinformation verloren gehen.
- Sekundäre Kanten werden ignoriert.
- Kreuzende Kanten werden aufgelöst, d.h. die reihenfolge der Töchter ist entscheidend.

Beispiel

Der Baum in Bild 6 wird umgewandelt in folgenden Binärbaum:





Formelumwandlung

Die Eingabeformeln müssen entsprechend umgewandelt werden, so dass Formel ϕ gilt in Baum b, genau dann wenn Formel ϕ' gilt in Binärbaum b'.
Es muss aber auch die Formel, die der Benutzer eingibt, in ein Format umgewandelt werden, dass Mona versteht. Diese beiden Schritte werden kombiniert, in dem die Konstrukte, die sich auf die Struktur des Baumes beziehen, in der Mona-Datei zu komplexeren Teilformeln umgewandelt werden.

Beispiel

Die Formel $x \triangleleft y$, 'x ist die Mutter von y', wird z.B. übersetzt durch

```
right_branch(x.0,y)
```

Mit `right_branch(a,b)` eine Formel, die ausdrückt, dass b auf dem rechten Zweig liegt, der von a heruntergeht:

```
pred right_branch (var1 x,y) =
  all2 X: (x in X &
    (all1 z, w: (z in X & w = z.1) => w in X)
  ) => y in X;
```

Übersicht

Labels in der Baumbank

Die Labels in den Baumbanken entsprechen grammatischen Funktionen der Teilbäume. Es gibt auch Kantenlabels und die Blätter haben mehrere Labels (Wort, Morphologie, Wortart, . . .).
Einfache Lösung, übernommen von der MSO-Umwandlung: Labels als Prädikate auffassen. Ein Label ist also die Menge aller Knoten, die dieses Label tragen.
Die Labels eines Knoten werden als Bitvektor kodiert: die Labels, die in der Formel vorkommen, kriegen der Reihe nach einen Index; einen Knoten im Binärbaum hat eine 1 am Index von X wenn er das Label X hat, sonst eine 0.
Für Mona sind Labels freie Variablen zweiter Stufe. Diese werden im Automaten mit jene Bitvektoren kodiert.

Schritte

Bemerke, dass die Kodierung der Labels abhängt von der Eingabeformel. Somit muss diese für jede Eingabe neu berechnet werden. Die Binarisierung der Bäume ist hingegen unabhängig von der Formel und kann deshalb einmalig gemacht werden.
Beim Registrieren einer neuen Baumbank findet eine Vorverarbeitung statt, die alle Bäume in Binärbäume umwandelt, jedoch die Labels beibehält.
Bei jeder Anfrage wird der Binärbaum gelesen, die Labels in Bitvektoren umgewandelt, dann dem Automaten übergeben.

Übersicht

Unterschied zu gewöhnlichen Baumautomaten

Guided Tree Automata sind Baumautomaten mit einer Führung. Die Führung kann Wissen über das Problem, das Modell, enthalten. Dies hilft die Größe der Automaten zu beschränken. Wir wissen aber nichts über die Bäume, die wir durchsuchen.

Glücklicher Zufall

Zum Glück kann Mona eine Dummy-Führung generieren. Dies hat zur Folge, dass der Wurzelknoten und sein rechter Teilbaum als Dummy angesehen werden, der eigentliche Baum also als linker Teilbaum des Wurzelknotens erwartet wird.
Hier kommt uns unser Trick aus der Vorverarbeitung zur Hilfe: wir hatten einen virtuellen Wurzelknoten eingeführt, der die unzusammenhängenden Teile verbindet. Bei der Binarisierung wird dieser den Wurzelknoten und der Rest des Baumes kommt tatsächlich als linker Teilbaum.

Übersicht

Mona aufrufen

Die Formel des Benutzers wird in ein Format umgewandelt, das Mona versteht, und in eine Datei geschrieben. Mona wird einfach mit einem Systemaufruf gestartet und schreibt den Automaten in eine Textdatei.
Nachteil: Mona muss installiert sein auf den Rechner. (Und im Pfad auffindbar sein.)

Automat auswerten

Die Mona-Bibliothek in C wird aufgerufen mit JNI. Um diesen Prozess zu erleichtern, wird SWIG benutzt. Dieses generiert Wrapper-Dateien für die entscheidenden C-Strukturen.
Der Automat wird mit eine Bibliotheksfunktion eingelesen. Auf der Java-Seite werden die Bitvektoren der Binärbäume berechnet und mithilfe der Wrappers ein entsprechender Binärbaum in C konstruiert. Dieser wird vom Automaten ausgewertet.
Nachteil: JNI erfordert systemspezifische Bibliotheken, Systemunabhängigkeit von Java geht verloren.

Vorzeige

`http://tcl.sfs.uni-tuebingen.de/MonaSearch`

Vorschau

Anstehende Erweiterungen und Verbesserungen:

- Eine Visualisierungskomponente, die die Ergebnisse als Bäume zeigt.
- Einfachere Installation mit Java Web Start.
- Komplexere Strukturen in der Baumbank bewältigen.
- Benutzeroberfläche erweitern, mehr Optionen anbieten.

FINIS

Literatur