

Allquantoren in der TIGER-Abfragesprache

Torsten Marek¹ Joakim Lundborg² Martin Volk³

¹Institut für Linguistik
Universität des Saarlandes

²Institut für Linguistik
Stockholms Universitet

³Institut für Computerlinguistik
Universität Zürich

30. September 2008



Übersicht

- Beschreibungssprache für partielle Syntax-Graphen
- Entwickelt von (König and Lezius, 2003)
- Erste Implementierung in TIGERSearch (Lezius, 2002)



Übersicht

- Beschreibungssprache für partielle Syntax-Graphen
- Entwickelt von (König and Lezius, 2003)
- Erste Implementierung in TIGERSearch (Lezius, 2002)

Ein Beispiel

*Finde alle NPs, die das Nomen „Haus“
enthalten.*

```
[cat="NP"] > #n & #n:[word="Haus"&  
pos="NN"]
```



Übersicht

- Beschreibungssprache für partielle Syntax-Graphen
- Entwickelt von (König and Lezius, 2003)
- Erste Implementierung in TIGERSearch (Lezius, 2002)

Ein Beispiel

Finde alle NPs, die das Nomen „Haus“ enthalten.

```
[cat="NP"] > #n & #n:[word="Haus"&pos="NN"]
```

... als Graph



Mächtigkeit von Abfrageformalismen

(Lai and Bird, 2004):

- 7 prototypische Korpusabfragen
- 5 Suchwerkzeuge



Mächtigkeit von Abfrageformalismen

(Lai and Bird, 2004):

- 7 prototypische Korpusabfragen
- 5 Suchwerkzeuge

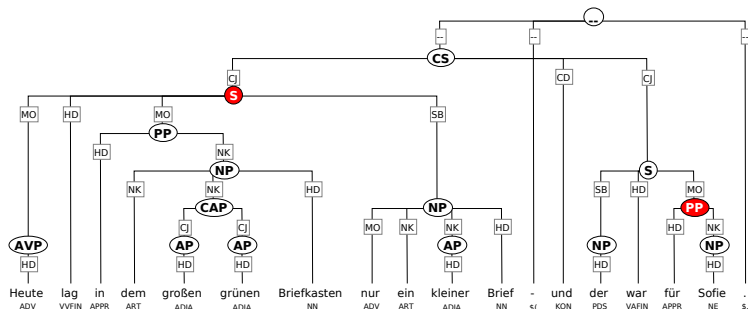
Problematische Abfragen

Finde alle Sätze, in denen keine PP vorkommt:

```
[cat="S"] !>* [cat="PP"]
```



Falsche Ergebnisse



- ein korrektes Ergebnis für die Abfrage:
Finde einen S-Knoten und einen PP-Knoten, sodass der S-Knoten den PP-Knoten nicht (transitiv) dominiert.



Quantifikatoren

- alle Knoten sind implizit existentiell quantifiziert
 $\exists \#np1 (\exists \#pp1 (\#np1 > \#pp1 \ \& \ \dots))$



Quantifikatoren

- alle Knoten sind implizit existentiell quantifiziert
 $\exists \#np1 (\exists \#pp1 (\#np1 > \#pp1 \ \& \ \dots))$
- daher sind keine Aussagen über Mengen von Knoten möglich



Quantifikatoren

- alle Knoten sind implizit existentiell quantifiziert
 $\exists \#np1 (\exists \#pp1 (\#np1 > \#pp1 \ \& \ \dots))$
- daher sind keine Aussagen über Mengen von Knoten möglich

Lösungsvorschläge in (Lezius, 2002)

$\forall \#pp ([cat="S"] > * \#pp) \Rightarrow \#pp: [cat!="PP"]$



Knotenmengen

- *Knotenmengen* als neuer Datentyp



Knotenmengen

- Knoten*mengen* als neuer Datentyp
- Bedingungen über Knotenmengen müssen für alle Elemente der Menge erfüllt sein



Knotenmengen

- Knoten*mengen* als neuer Datentyp
- Bedingungen über Knotenmengen müssen für alle Elemente der Menge erfüllt sein
- alle Variablen, die % als Präfix haben, sind Mengen



Knotenmengen

- *Knotenmengen* als neuer Datentyp
- Bedingungen über *Knotenmengen* müssen für alle Elemente der Menge erfüllt sein
- alle Variablen, die % als Präfix haben, sind Mengen

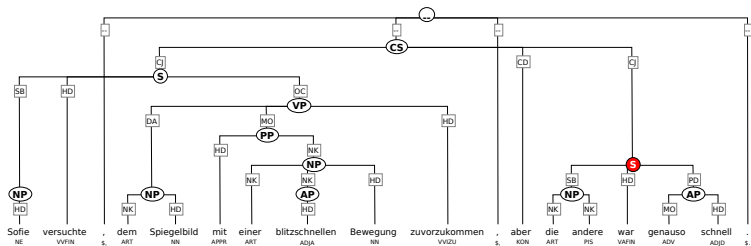
Erneuter Versuch

Finde alle Sätze, die keine PP enthalten:

```
[cat="S"] !>* %pp: [cat="PP"]
```



Korrekte Ergebnisse



Leere Mengen

Finde alle Graphen, die keine PP enthalten:

```
%pp: [cat="PP"] & empty(%pp)
```



Leere Mengen

Finde alle Graphen, die keine PP enthalten:

```
%pp: [cat="PP"] & empty(%pp)
```

Mengenprädikate

- *empty, nonempty*



Leere Mengen

Finde alle Graphen, die keine PP enthalten:

```
%pp: [cat="PP"] & empty(%pp)
```

Mengenprädikate

- *empty*, *nonempty*
- komplette Mengenalgebra wäre möglich



Leere Mengen

Finde alle Graphen, die keine PP enthalten:

```
%pp: [cat="PP"] & empty(%pp)
```

Mengenprädikate

- *empty*, *nonempty*
- komplette Mengenalgebra wäre möglich
- ... fügt aber ungewollte Komplexität hinzu



Limitierungen

- bisher können Knotenmengen nur anhand lokaler Merkmale erstellt werden



Limitierungen

- bisher können Knotenmengen nur anhand lokaler Merkmale erstellt werden
- Verwendung von Operatoren (Dominanz etc.) nicht möglich



Limitierungen

- bisher können Knotenmengen nur anhand lokaler Merkmale erstellt werden
- Verwendung von Operatoren (Dominanz etc.) nicht möglich

In Planung: geschachtelte Abfragen

Finde den tiefsten gemeinsamen Elternknoten einer Sequenz NP VP^a:

```
#np: [cat="NP"] .* #np: [cat="VP"] &  
#np !>* #vp & #vp !>* #np &  
#a: [NT] >* #np & #a >* #vp &  
%b: {#a >* %: [NT] & % >* #np} !>* #vp
```

^a(Lai and Bird, 2004)

Übersicht

- Werkzeug zur Erstellung paralleler Baumdatenbanken
- TIGERSearch-Implementation für Suche über Alignments
- Erste Anwendung: SMULTRON^a

^a<http://www.ling.su.se/dali/research/smultron/index.htm>



Umgebung

Korpus: TIGER 2.1 (50.470 Graphen)

Rechner: Core 2 Duo 2.2 GHz, 2 GiB RAM

Software: Linux 2.6.26, Python 2.5.2, SQLite 3.5.9



Umgebung

Korpus: TIGER 2.1 (50.470 Graphen)

Rechner: Core 2 Duo 2.2 GHz, 2 GiB RAM

Software: Linux 2.6.26, Python 2.5.2, SQLite 3.5.9

Zeiten

Abfrage	Zeit (s)
<code>[cat="S"] !>* %pp: [cat="PP"]</code>	5.58
<code>%pp: [cat="PP"] & empty(%pp)</code>	1.43
<code>[cat="S"] !>* %a: [pos="ART"]</code>	5.78
<code>#np: [cat="NP"] !>* %p: [cat="PP"] & #np >* [cat="NP"]</code>	7.74

Bezugsquellen

- implementiert in der Entwicklungsversion des Stockholm TreeAligner^a (Lundborg et al., 2007)
- Demnächst: Version 1.0
- Demnächst: Web-Demo
- Demnächst: Teil des NLTK^b

^a<http://dev.ling.su.se/treealigner>




^b<http://www.nltk.org>



Vielen Dank für Ihre Aufmerksamkeit.

Fragen?



-  König, E. and Lezius, W. (2003).
The TIGER language – a description language for syntax graphs,
formal definition.
Technical report, IMS, University of Stuttgart.
-  Lai, C. and Bird, S. (2004).
Querying and updating treebanks: A critical survey and requirements
analysis.
In *Proceedings of the Australasian Language Technology Workshop*.
-  Lezius, W. (2002).
Ein Suchwerkzeug für syntaktisch annotierte Korpora.
PhD thesis, IMS, University of Stuttgart.
-  Lundborg, J., Marek, T., Mettler, M., and Volk, M. (2007).
Using the Stockholm TreeAligner.
In *Proc. of The 6th Workshop on Treebanks and Linguistic Theories*
Bergen.

