DATR: Eine flexible Sprache für Lexikalische Semantik

Eric Auer <eric@coli.uni-sb.de>

27. Mai 2002

Einleitung

Was ist DATR

- Abkürzung für *D*efault *AT*ribute *R*epresentation
- „Einfache Sprache um nichtmonotone Vererbungsnetze mit Pfad/Wert-Gleichungen zu definieren"

Vorteile

- Einfach zu implementieren („1 Seite Prolog")
- Inferenz und deklarative Semantik klar definiert
- Praktisch um (z.B. in LexSem) Generalisierungen kompakt auszudrücken

Motivation: Ein Beispiel

So könnten zwei Lexikoneinträge aussehen:

```
Word1:
  <syn cat>   =  verb  ← = ist extensional
  <syn type>  =  main
  <syn form>  =  present participle
  <mor form>  =  love ing.
Word2:
  <syn cat>   =  verb  ← wie oben
  <syn type>  =  main  ← wie oben
  <syn form>  =  passive participle
  <mor form>  =  love ing.
```

Motivation: Erste Verbesserung

```
VERB:
  <syn cat>   ==  verb  ← == ist Definition
  <syn type>  ==  main.
Word1:
  <syn cat>   ==  VERB:<syn cat>  ← Verweis zum Allgemeinen
  <syn type>  ==  VERB:<syn type>  ← d.h. lokale Vererbung)
  <syn form>  ==  present participle
  <mor form>  ==  love ing.
Word1:
  <syn cat>   ==  VERB:<syn cat>  ← wir wiederholen uns!
  <syn type>  ==  VERB:<syn type>
  <syn form>  ==  passive participle
  <mor form>  ==  love ed.
```

Motivation: Zweite Verbesserung

```
VERB:
  <syn cat>    ==  verb
  <syn type>   ==  main.
Love:
  <>           ==  VERB  ← default für alle Pfade
  <mor root>   ==  love.
Word2:
  <>           ==  Love
  <syn form>   ==  present participle
  <mor form>   ==  <mor root> ing.
Word1:
  <>           ==  Love
  <syn form>   ==  passive participle
  <mor form>   ==  <mor root> ed.  ← immernoch nicht optimal
```

Auswertbare Pfade

```
VERB:
  <syn cat>                          ==  verb
  <syn type>                         ==  main
  <mor form>                         ==  "<mor "<syn form>">"   ← doppelter ep
  <mor past>                         ==  "<mor root>" ed        ← Trick: evaluable paths
  <mor passive>                      ==  "<mor past>"           ← (für globale Vererbung)
  <mor present>                      ==  "<mor root>"           ← default für das Präfix
  <mor present participle>           ==  "<mor root>" ing
  <mor present tense sing three>     ==  "<mor root>" s.
Love:                                                           ← gem. mor root
  <>                                 ==  VERB
  <mor root>                         ==  love.
Word1:
  <>                                 ==  Love
  <syn form>                         ==  present participle.    ← via doppelten ep...
Word2:
  <>                                 ==  Love
  <syn form>                         ==  passive participle.
```

Kompliziertere Subregularitäten

```
EN_VERB:
  <>                                 ==  VERB  ← subreguläre Variante von VERB
  <mor past participle>              ==  "<mor root>" en.
Mow:
  <>                                 ==  EN_VERB
  <mor root>                         ==  mow.
Sew:
  <>                                 ==  EN_VERB
  <mor root>                         ==  sew.
Be:
  <>                                 ==  EN_VERB
  <mor root>                         ==  be
  <mor present tense sing one>       ==  am
  <mor present tense sing three>     ==  is
  <mor present tense plur>           ==  are
  <mor past tense sing one>          ==  ...
```

Pfadverlängerungen

```
VERB:
  <mor present tense>  ==  “<mor root>”
  <mor form>           ==  <mor “<syn form>”>.
```

Das längste passende Präfix legt die Antwort fest, diese ist für längere Pfade also implizit:

```
ableitbar:
  <mor present tense sing>  ==  “<mor root sing>”
  <mor present tense plur>  ==  “<mor root plur>”
  <mor form present>        ==  <mor “<syn form present>” present>
  <mor form passive>        ==  <mor “<syn form passive>” passive>
```

Parameterverwendung

```
VERB:
  <syn form>     ==  “<syn tense>” “<syn number>” “<syn person>”
Word4:
  <>             ==  Sew
  <syn tense>    ==  present
  <syn person>   ==  third  ← Reihenfolge ist natürlich egal
  <syn number>   ==  sing.
```

UPN Boolesche Algebra

```
Boolean:
  <>                  ==  false  ← allgemeinster default
  <or>                ==  true   ← default für or
  <if>                ==  true
  <not false>         ==  true
  <and true true>     ==  true   ← Ausnahme vom allg. default
  <if true false>     ==  false
  <or false false>    ==  false. ← Ausnahme vom or default
```

Exkurs: Syntax

Was gibt es?

- Reservierte Symbole: `: ” < > = == . ’ % #`
- Knoten, Atome und Variablen

Rvalues (rekursiv definiert!)

- Atom oder Variable: `atom1 $variable1`
- Lokaler Vererbungsdeskriptor: Knoten und/oder Pfad, `knoten:<deskriptor1 desk2...>`
- Globaler Vererbungsdeskriptor: dasselbe mit Quotes

Exkurs: Fortsetzung

Sätze

- „Alles bis zum Punkt"
- Extensional mit =, definitional mit ==
- Definition impliziert Extension
- Schreibweisen: Jedes Statement als ein Satz oder Knotennamen am Anfang und dann mehrere Statements.
- Vererbung: Lokal und Global – hatten wir schon...
- Vererbung definiert bequem defaults, longest/best match wins

Mehr als nur Lexikon: Endliche Automaten

```
# vars $abc: a b c d e f g h i j k l m n o p q r s t u v w x y z.
# vars $vowel: a e i o u.                    ← eine Compilerdirektive
# vars $consonant: $abc - $vowel.                ← Differenzmenge
# vars $var.                                     ← beliebiger Wert
# vars $foo: $abc - b a r.
BUGGY:                              ← Beispiel für versteckte Probleme:
  <e>                 ==   e i <>
  <$vowel>            ==   $vowel e <>.  ← zwei Regeln für e!
SPELL:
  <$abc>              ==   $abc <>
  <e + $vowel>        ==   $vowel <>.

ableitbar:
  <l o v e>           =    l o v e
  <l o v e + s>       =    l o v e s
  <l o v e + e d>     =    l o v e d
  <l o v e + i n g>   =    l o v i n g
```

Endliche Transducer: Swahili „Ni ta ku penda”

```
S1:
     <subj 1 sg>   ==   ni S2:<>
     <subj 2 sg>   ==   u S2:<>
     <subj 3 sg>   ==   a S2:<>
     <subj 1 pl>   ==   tu S2:<>
     <subj 2 pl>   ==   m S2:<>
     <subj 3 pl>   ==   wa S2:<>.
S2:
     <past>        ==   li S3:<>.
     <futr>        ==   ta S3:<>.
S3:
     <obj 1 sg>    ==   ni S2:<>
     <obj 2 sg>    ==   ku S2:<>
     <obj 3 sg>    ==   m S2:<>
     <obj 1 pl>    ==   tu S2:<>
     <obj 2 pl>    ==   wa S2:<>
     <obj 3 pl>    ==   wa S2:<>.
S4:  <like>        ==   penda.
```

Ein Schritt weiter

Regelmässigkeiten in Regeln

- Bisher: Regelmässigkeiten im Lexikon, endliche Automaten und Transducer
- Neu: Über Variablen/Parameter, evaluable Paths. . . Regelmässigkeiten lexikalischer Regeln selbst wieder in DATR darstellen!
- Beispiel: Subkategorisierungslisten (folgende Folie, zeigt auch wie Listen dargestellt werden können)
- Problem: Nur begrenzte Behandlung von Listen und Disjunktion (übernächste Folie)

Listen: Subkategorisierung (1)

```
NIL:
  <>                              ==  nil  ← Ende einer Liste
  <rest>                          ==  UNDEF
  <first>                         ==  UNDEF.
NP_ARG:                                 ← eine Abstraktionshilfe
  <first syn cat>                 ==  np
  <first syn case>                ==  accusative
  <rest>                          ==  NIL:<>.
VERB:                                       ← intransitives Verb
  <syn cat>                       ==  verb
  <syn args>                      ==  NP_ARG:<>
  <syn args first syn case>       ==  nominative.
...
```

Listen: Subkategorisierung (2)

```
NP_ARG:
  <first syn cat>              ==  np
  <first syn case>             ==  accusative
  <rest>                       ==  NIL:<>.
VERB:                                          ← intransitives Verb
  <syn cat>                    ==  verb
  <syn args>                   ==  NP_ARG:<>
  <syn args first syn case>    ==  nominative.
TR_VERB:
  <>                           ==  VERB
  <syn args rest>              ==  NP_ARG:<>.
DI_VERB:
  <>                           ==  TR_VERB  ← erste zwei Argumente wie bisher
  <syn args rest rest>         ==  PP_ARG:<>.  ← hier nicht gezeigt
```

Problem: Disjunktive Effekte

```
Bank1:
  <>             ==  NOUN
  <mor root>     ==  bank
  <sem gloss>    ==  side of river.
Bank2:
  <>             ==  NOUN
  <mor root>     ==  bank
  <sem gloss>    ==  financial institution.
```

Regelmässige Polysemie

```
Cherry:
  <>                 ==  NOUN
  <mor root>         ==  cherry
  <sem gloss 1>      ==  sweet red berry  ← Aufzählung mit Nummern
  <sem gloss 2>      ==  tree bearing <sem gloss 1>
  <sem gloss 3>      ==  wood from <sem gloss 2>.
```

Lässt sich in eine Regel für „Obstbaumwörter" abstrahieren

Disjunktion: Ansätze für Aufzählungen

```
Word71:
  <syn number>           ==  plural
  <mor form>             ==  hoof s
  <mor form alternate>   ==  hoove s.   ← ähnlich wie oben
Word72:
  <syn number>           ==  plural
  <mor forms>            ==  hoof s | hoove s.
Word72:
  <syn number>           ==  plural
  <mor forms>            ==  { hoof s , hoove s }.
```

Problem: `{ foo , bar }` und `foo | bar` sind für DATR nur Daten. „Verständnis" für Aufzählungen muss beim Verarbeiten der DATR Ausgabe dazukommen.

Konsistenz: Ideen

Funktionalität fordern

- DATR Beschreibungen sind meistens definitional
- Normal maximal eine Aussage pro Knoten/Pfad Paar
- Meist partielle Funktionen von Pfaden zu Werten
- Inkonsitenz doch möglich, also Forderung: Funktionalität
- Inkonsistenz kann als „neu korrigiert alt" interpretiert werden.

Konsistenz: Details

Definition von Funktionalität

- Eine DATR Beschreibung ist funktional gdw. sie nur definitionale Aussagen enthält und diese eine (partielle) Funktion von Knoten/Pfad Paaren zu Deskriptorsequenzen beschreiben.
- Jede funktionale DATR Beschreibung ist konsistent
- Auch andere DATR Beschr. können konsistent sein, aber selten nützlich (nächste Folie)

Konsistenz: Beispiele

```
INCONSISTENT:
  <syn cat>  ==  verb
  <syn cat>  ==  noun.   ← Syntax erlaubt das!
NONFUNC1:
  <a>        ==  UNDEF  ← bewirkt keine constraints
  <a>        ==  1.     ← also dennoch konsistent
NONFUNC2:
  <a>        ==  b
  <a>        ==  1
  <b>        ==  1.     ← zufällig konsistent
NONFUNC3:
  <a>        ==  b
  <b>        ==  a      ← gegenseitige Anhängigkeit
  <a>        ==  1.
```

Vererbungshierarchien

Klassen von Vererbung

- Maximal eine Quelle: Baumförmiges Netzwerk
- Disjunkte Quellen: Orthogonales Netzwerk (Orthogonal Multiple Inheritance, OMI)
- OMI ist normal in funktionalen DATRs (konsistent, längster Subpfad gewinnt)
- PMI – Prioritized Multiple Inheritance: In DATR darstellbar (Evans et al. 1993), laut Evans und Gazdar wenig Vorteile

Anwendungmodi

Drei Seiten

- Anwendung: Lexika erzeugen, warten, benutzen
- Theorie, Abfrage und Wert
- Wert gesucht: Inferenz
  `Love:<mor past participle>` liefert `love ed`
- Frage gesucht: Inverse Abfrage
  Welche Form von welchem Verb ist `love ed`?
- Theorie gesucht (Theorieerzeugung): Wäre genial mit genug perfekt annotierten Daten! Brauchbar für Erweiterung, Umwandlung, groben Entwurf von Lexika, Morphologie

Anwendungen und Implementationen

Viele Möglichkeiten!

- Hierarchien verfeinern (Light et al.)
- Lexikon für PATR Parser, volle Abfrage (Duda & Gebhardi)
- DATR mit erweiterter Abfragesprache EDQL (Gibbon)
- DATR von Evans und Gazdar, in Prolog
- DDATR (Scheme), NODE (Prolog) von Gibbon
- QDATR (Prolog)
- Versionen mit reverse query (Langer)
- DATRs von extensionalen Daten ableiten (Barg)

Zusammenfassung

Was ist DATR?

- einfache, allgemeine „Programmiersprache" für LexSem, vergleichbar mit HPSG für Syntax
- nicht für eine bestimmte Theorie
- Idee von nichtmonotonen Vererbungsnetzen: Defaults und Sonderfälle
- Inferenz und deklarative Semantik klar definiert
- für LTAG, PATR, UCG uvm. schon verwendet

Zusammenfassung: Sprachen

Flexibel einsetzbar

- versucht für alle Sprachen anwendbar zu sein (Bsp.: Arabisch, Tschechisch, Englisch, Spanisch, Russisch, Französisch, Latein, auch (kleiner) für Sanskrit, Japanisch etc.)
- nicht auf eine Ebene festgelegt: Phonologie, Orthographie, Morphologie, Syntax, Semantik. . .
- Anwendungsvorschlag: DRT für (Diskurs-) Semantik, HPSG für Syntax, weitere Module für Morphologie und Ortographie, alle in DATR darstellen
- Viele Beispielfragmente und Informationen: *www.datr.org*