Schlussprüfung Finite-State-Methoden in der Sprachtechnologie FS 11

Aufgabenstellung: Simon Clematide

Prüfung vom 6. Juni 2011
Institut für Computerlinguistik
Universität Zürich

Vorname _____ Matrikelnummer _____

Nachname _____

Für Studierende der folgenden Studiengänge:

☐ BA - Studiengang Computerlinguistik (Phil. Fakultät)

☐ BA - Studiengang Computerlinguistik und Sprachtechnologie (Phil. Fakultät)

☐ BA-Studierende (Wirtschaftswissenschaftliche Fakultät)

☐ Studierende des Nebenfachs Informatik mit Studienbeginn ab WS 04/05

☐ Multidisziplinfach (ETH)

☐ Andere:

Nur für Lizentiatsstudierende der Computerlinguistik als ein Fach aus der Phil. Fakultät:

Strasse: _____ Hauptfach: _____

PLZ/Ort: _____ E-Mail: _____

Aufgabe Nr.:	1	2	3	4	5	6	**Summe**
Punktzahl:	10	10	10	8	12	10	60
Davon erreicht:							

Note SU: _____ Note SP: _____

Endnote: _____ Bestanden: ☐ Ja ☐ Nein

Auf jedes zusätzliches separates Blatt mit Lösungen den Nachnamen schreiben! Viel Erfolg!

Wichtige Hinweise

Punkte-Maximum: 60 (pro Minute 1 Punkt)
Hinweis: Bitte schreiben Sie in einem überlegten und knappen, aber verbalen Stil (keine Stichwortsammlungen). Bei inhaltlichen Auswahlsendungen, wo einfach mal alles spontan hingeschrieben wird und Falsches wie Korrektes munter vermischt sind, behalte ich mir Abzüge vor. Das Zeitbudget ist so berechnet, dass man vor dem Schreiben kurz überlegen kann.

10 **1. Beschreibungsebenen und Fachbegriffe der Morphologie (10 Punkte)** Zeigen Sie am Beispiel der Wortform „Gartenzwergleins“, dass Sie die morphologische Fachterminologie virtuos beherrschen. Geben Sie eine Musteranalyse, bei der Sie die gelernte Analysebegrifflichkeit aus der Morphologie für diese Wortform anwenden. Merken Sie auch an, was dabei eher (un-)typisch oder (un)produktiv ist.

- Bezüglich Wortbildung handelt es sich hier um eine Komposition und eine Derivation. Für sich genommen lässt sich schwer entscheiden, ob die beiden Stämme „garten“ und „zwerglein“ komponiert wurden, oder ob „garten“ und „zwerg“ komponiert wurden und als Ganzes einer Derivation mit „lein“ unterzogen sind. In einer hierarchischen Analyse wie bei Canoo muss dies entschieden werden, in einer reinen Segmentanalyse wie bei GERTWOL kann das unspezifiziert bleiben. Abhängig von der Entscheidung besteht die Derivationsbasis aus „zwerg“ oder „gartenzwerg“ und wird mit dem Derivationssuffix „-lein“ konkateniert. Derivationssuffixe tragen wie in diesem Fall gewisse lexikalisch Eigenschaften: „-lein“ ist ein nominales, sächliches Suffix. Semantisch hat das Suffix „-lein“ eine einheitliche Funktion als Verkleinerungsoperator. Er steht dabei in Konkurrenz zum Verkleinerungssuffix „-chen“, welcher viel breiter und produktiver anwendbar ist. Der Ausdruck „Zwerglein“ ist lexikalisiert und blockiert die Verwendung von „-chen“, welches vielleicht noch als Spontanbildung denkbar ist. Bei der Komposition tritt kein Fugenelement auf. Semantisch handelt es sich in beiden Fällen um eine Determinativkomposition.
- Bezüglich Flexion handelt es sich um Genitiv Singular. Das Genus der Wortform ist Neutrum. Das Flexionparadigma von „gartenzwerglein“ ist morphologisch sehr reduziert. Als einziges sichtbares Flexionsmorph tritt „-s“ auf im Genitiv Singular. Im Gegensatz etwa zum Wort „Schiff“ ist keine e-Einfügung möglich (veralteter Dativ: *„dem Gartenzwergleine“ oder Genitiv: *„wegen des Gartenzwergleines“).

10 **2. Linguistik vs. Sprachtechnologie (10 Punkte)** In Carstensen et al. 2009 findet sich Folgendes:

> Im Fall der Umlautung in *Väter* lässt sich ein zugrundeliegendes Pluralsuffix *-I* annehmen, das Umlautung auslöst, dann aber durch eine weitere phonologische Regel gelöscht wird:
>
> | Repräsentation | phonologische Regel | |
> |---|---|---|
> | Vater-I | | |
> | ⇓ | a → ä /__ I | (*a* wird vor *I* zu *ä*) |
> | Väter-I | | |
> | ⇓ | I → ∅ | (*I* wird gelöscht) |
> | Väter | | |

Kommentieren Sie die linguistischen und sprachtechnologischen Aspekte dieser Lösungsidee.

Die linguistische Idee ist vom aktuellen Sprachstand nur wenig motiviert. Warum hier ein abstraktes I angesetzt werden soll, um aus einem a ein ä zu machen, erscheint umständlich. Das Pluralsuffix I muss auf jeden Fall lexemspezifisch sein, da es andere deutsche Nomen gibt, wo das a nicht umgelautet wird im Plural. Vom sprachtechnologischen Standpunkt handelt es sich um ein kaskadiertes Anwenden von Ersetzungsregeln. Vom sprachtechnologischen Standpunkt ist die phonologische Regel zu ungenau formuliert, um als solche umgesetzt werden zu können. Zwischen dem a und dem I taucht ja noch Wortmaterial auf, andrerseits kann ja nur ein a des Stammes umgelautet werden. Obwohl die

linguistische Motivation der Lösungsidee nur noch wenig zugänglich ist, ist sie doch technisch gesehen eine mögliche vernünftige. D.h. sie entspricht dem Ansatz von GERTWOL einen Umlaut-Trigger zu setzen und dann durch allgemeine Regeln die Umlautung in der Wortform durchzuführen.

3. Nicht-konkatenative Morphologie (10 Punkte)

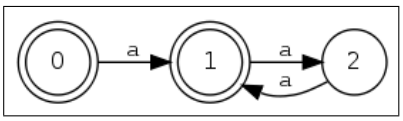
5 (a) Was versteht man unter nicht-konkatenativer Morphologie? Geben Sie Beispiele und eine möglichst gute abstrakte Charakterisierung.

Grundsätzlich bedeutet nicht-konkatenative Morphologie eine Abgrenzung zur rein konkatenativen Morphologie. Diese wiederum beinhaltet grob gesagt, dass Morphe aneinandergefügt werden. Typischerweise passieren auch an den Stellen, wo Morphe zusammengefügt werden, phonologische Anpassungen (Weglassen, Hinzufügen). Aber grundsätzlich handelt es sich um Morphkombinationen. Dinge wie Ablaut, Umlaut, Infigierung, Transfigierung oder alle nichtsegementalen Prozesse insbesondere Suppletion oder Reduplikation

5 (b) Inwiefern sind nicht-konkatenative Phänomene ein Problem für Finite-State-Ansätze?

Vollreduplikation ist ein grundsätzliches Problem für Finite-State-Ansätze, da es keinen endlichen Transducer geben kann, der beliebigen (!) Text duplizieren kann. Mithilfe der sekundären Kompilation ist es jedoch möglich, den wichtigen Fall von Stammvollreduplikation (typischerweise gibt es eben eine endliche und vorgegebene Anzahl von Stämmen).

4. Reguläre Ausdrücke und Übergangsdiagramme (8 Punkte)



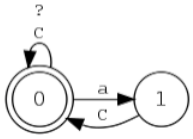
3 (a) Schreiben Sie einen regulären Ausdruck, der den obig abgebildeten Automaten ergibt:

Einige mögliche Lösungen mit xfst-Operatoren:

```
read regex a* - {aa}+;
read regex (a [a^2]*);
read regex (a {aa}*);
```

5 (b) Zeichnen Sie das Zustandsdiagramm eines minimalen (!) Automaten, der die Sprache eines mittels `read regex a+ => _ C;` definierten Netzwerks akzeptiert.

Da nach mindestens einem a immer ein C folgen muss, können gar nicht mehrere a einander folgen. Somit ist dieser reguläre Ausdruck äquivalent zu `read regex a => _ C;` ,



5. i-Wörter implementieren (12 Punkte)

Betrachten Sie folgendes Phänomen:

Langform	Kurzform
[Ka][tha][ri][na]	[Ka][thi]
[A][bi][tur]	[A][bi]
[Stu][dent]	[Stu][di]

2 (a) Welcher Typ von morphologischer Prozess liegt hier vor? Geben Sie eine präzise linguistische Beschreibung der zugrundeliegenden Wortbildungsgesetzmässigkeit.

Es handelt sich um einen (der selteneren) subtraktiven morphologischen Prozesse. Dieser Prozess operiert auf Silbeneinheiten. Jeweils in der 2. Silbe wird ab dem Silbenvokal abgetrennt und der Silbenvokal „i“ eingefügt.

10 (b) Implementieren Sie einen Transduktor, welcher eine Langform wie `[Ka][tha][ri][na]` (inklusive Klammerung) auf der oberen Seite auf eine Kurzform wie `[Ka][thi]` (inklusive Klammerung) auf der unteren Seite abbildet. Die Klammerung ist bereits im Input gegeben. Versuchen Sie eine übersichtliche Dekomposition des Problems.

Zwei Lösungsmöglichkeiten, bei denen die Verwendung von `@->` die Sache stark vereinfacht:

```
define Vokal [a|e|i|o|u];
define Silbe %[ [\ %]]+ %];
define SilbenAnfang %[ [\ [%]|Vokal]]+;

# Explizite Modellierung von Silbenanfängen
read regex [ Vokal ?* @-> i %] || .#. Silbe SilbenAnfang _ ];

# Oder etwas einfacher
read regex [ Vokal ?* @-> i %] || .#. Silbe ?* _ ];
```

10 6. Cola-Machine^{++} (10 Punkte)

Nach dem unerwarteten Ableben des amerikanischen Erfinders P. Epsi (vermutete Todesursache „Koffeinüberdosis“) im Jahr 1951 wurde auf einem Notizblatt folgender kryptischer XFST-Kode gefunden. Sie haben nun als XFST-ForensikerIn den Auftrag erhalten, eine möglichst genaue Beschreibung der Funktionalität dieses Kode-Stücks abzuliefern. Die Ermittlungsbehörden haben Ihnen noch mitgeteilt, dass N als Abkürzung für die Nickelmünze (5 Cents), D als Abkürzung für den Dime (10 Cents) und Q für den Quarter (25 Cents) steht.

```
define ColaMachine
   [ D -> N^2, Q -> N^5 ]
.o.
   [ N* ]
.o.
   [ N^5 @-> "COLA" ]
.o.
   [ N^2 @-> D ];
```

Keine Musterlösung hier, nur die Bemerkung, dass das Cola auch bei dieser Maschine 25 Cents kostet und das Rückgeld rausgegeben wird und zwar in Form von maximal 1 N und maximal 2 D.