

## 1 Unitex – das Programm

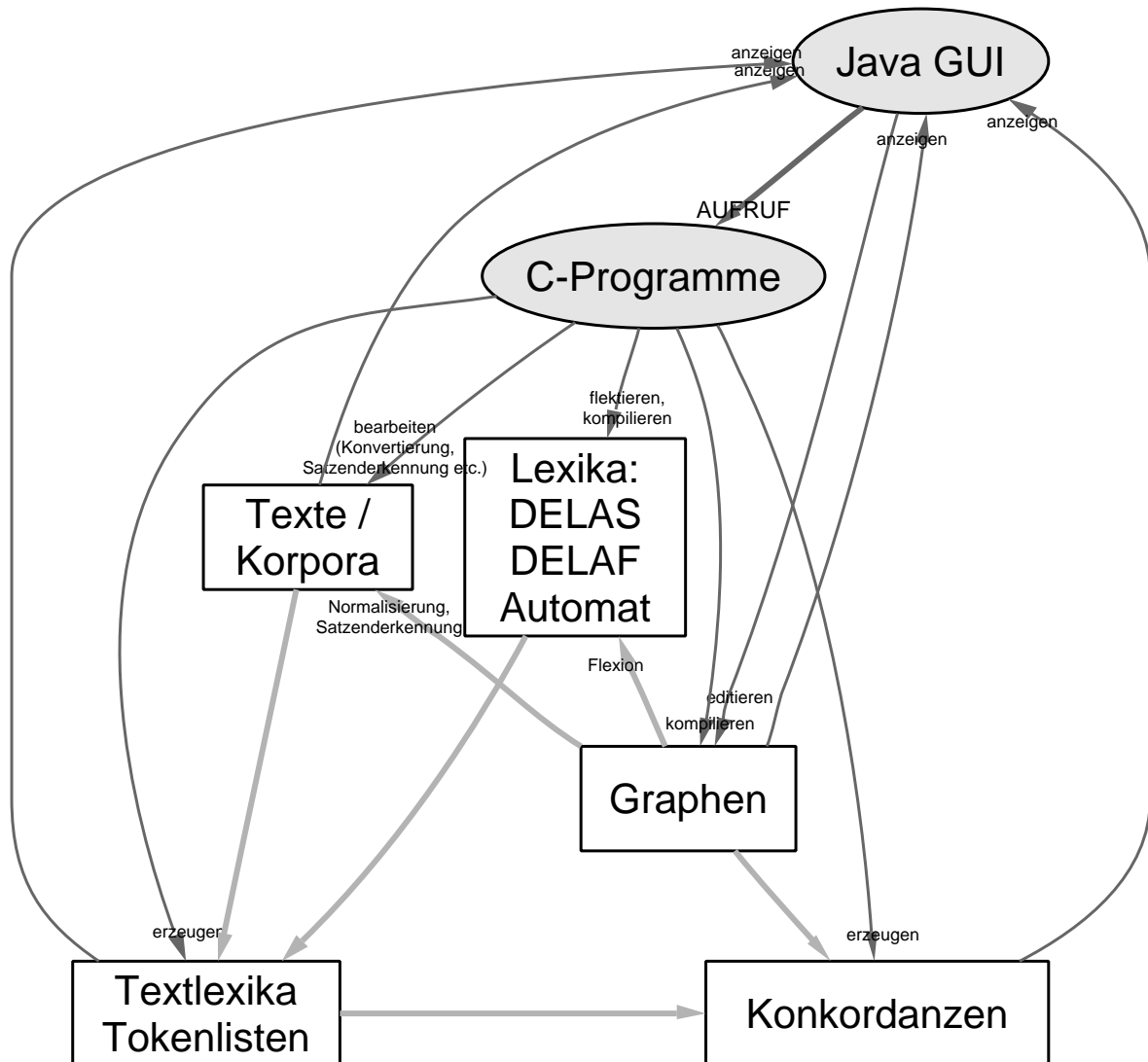


Abbildung 1: „Organigramm“ Unitex

- Programme
  - Java-Oberfläche
  - Sammlung von C-Programmen in  $\langle \text{Installationsverzeichnis} \rangle / \text{App} /$ , die auch einzeln aufgerufen werden können
- Ressourcen
  - Lexika für derzeit dreizehn Sprachen: alle Formen eines Lemmas, dessen Wortart, grammatische Kategorien, ev. semantische Merkmale
  - Graphen
  - Texte / Korpora

- Output
  - Token- und Frequenzlisten, Textlexika
  - Konkordanzen zu Graphen
  - Neuer Text mittels Transduktoren (segmentiert, annotiert etc.)

## 2 Vorverarbeitung

[ 0. Unitex arbeitet mit der Kodierung 'UTF-16 little endian'. Textdateien muss das Byteorder-mark U+FEFF vorangestellt sein. Zeilenumbrüche sollten als CR-LF kodiert werden. Zur Konvertierung empfiehlt sich das Tool `Convert` (bis v1.1 `Asc2Uni`). Möglich ist aber auch z.B. `recode l1..u6/21`]

### 1. `Normalize`:

- ersetzt Folge von Leerzeichen bzw. Zeilenumbrüchen durch ein Zeichen
- überprüft interne Syntax von lexikalisch annotierten Token (,Tags', z.B. {zu, .PRAEP})
- schreibt `<Text>.snt`

### 2. Manipulation des Textes (`<Text>.snt`) mittels Graphen in `Graphs/Preprocessing/`:

- a) `Fst2Txt -merge` mittels `Sentence/Sentence.fst2`: z.B. Satzenderkennung
- b) `Fst2Txt -replace` mittels `Replace/Replace.fst2`: z.B. Auflösung von Kontraktionen, Normalisierung von Anführungszeichen etc.

`<Text>.snt` wird dabei überschrieben!

### 3. `Tokenize`:

- tokenisiert Text aufgrund von `Alphabet.txt` (bei Option `-char_by_char` gilt jedes Zeichen – Tags und Leerzeichen ausgenommen – als separates Token)
- schreibt in Verzeichnis `<Text>_snt/`: `tokens.txt`, `text.cod` (Text als Liste von Pointern auf `tokens.txt`), `tok_by_alph.txt`, `tok_by_freq.txt`, `stats.n` (Anzahl der Token, Verteilung auf Typen), `enter.pos` (Liste von Pointern auf Zeilenumbrüche in `text.cod`)

### 4. lexikalische Analyse

- a) `Dico`: wendet Lexika auf Tokenliste (`tokens.txt`) bzw. `text.cod` (für Mehr-Token-Einträge) an; schreibt Textlexika `dlf` (einfache Formen), `dlc` (komplexe Formen), `err` (unbekannte Wörter) und Statistik `stat_dic.n`
- [ b) `PolyLex`: Programm zur Kompositaanalyse. Löscht erkannte Komposita aus `err`, schreibt sie in `dlf`. ]
- [ c) `Reconstrucao`: Programm zur Analyse portugiesischer Meso-/Enklitika ]

### 5. Sortierung der Textlexika:

- a) `SortTxt dlf`
- b) `SortTxt dlc`
- c) `SortTxt err`

### 3 Lexika

Unitex unterscheidet zwei Typen Lexika: (1.) das Lexikon mit flektierten Formen (DELAF) – Mehrwortlexeme eingeschlossen (DELACF),<sup>1</sup> und (2.) ein Grundformenwörterbuch (DELAS bzw. DELACS), das zur Verwendung mit Unitex in ein Vollformenwörterbuch expandiert werden muss.<sup>2</sup>

Lexika beider Typen können mit `CheckDic` auf Formatfehler überprüft werden. `CheckDic` liefert darüberhinaus eine Statistik über verwendete Codes, Zeichen etc.

#### 3.1 DELAF

##### Syntax eines Wörterbucheintrags

`<Vollform>, <Lemma> . <Wortart> [+ <klass. Merkmal>]* [ : <flekt. Merkmale>]* [ / <Kommentar> ]?`

also z.B.:

```
Andalusiens, Andalusien.N+GEO+Region:geN/ in Spanien
Männer, Mann.N+Hum:nmM:gmM:amM
heute,.ADV
```

Stimmen Grund- und Vollform überein kann letztere entfallen.

Metazeichen im Lexikon sind:

- , Trenner Form – Lemma
- . Trenner Lemma – Info
- + Trenner klassifikatorischer (semantischer) Merkmale
- : Trenner flektivischer (grammatischer) Merkmalskombinationen
- / Kommentar von hier bis zum Zeilenende
- = steht für Bindestrich oder Leerzeichen: `Mexiko=Stadt` entspricht *Mexiko-Stadt* und *Mexiko Stadt*

Kommen Metazeichen literal vor müssen sie durch ‘\’ geschützt werden: `E\=mc2,.Formel`, `teu\, teu\, teu,.INTJ` etc. Ansonsten sind beliebige Unicode-Zeichen in allen Positionen, d.h. auch zur Kennzeichnung von grammatischen Kategorien erlaubt.

**Groß- und Kleinschreibung** Unitex unterscheidet bei der Anwendung des Lexikons auf den Text zwischen Groß- und Kleinschreibung: klein matcht groß, aber nicht umgekehrt.

```
Schwarz,.N+FAM ⇒ Schwarz, SCHWARZ, nicht aber schwarz
schwarz,.A ⇒ schwarz, aber auch Schwarz, SCHWARZ, ...
```

Da so viele Fehlanalysen vermieden werden, ist es sinnvoll alle großgeschriebenen Wörter in der korrekten Schreibung ins Lexikon aufzunehmen.

Die Definition, welche Buchstaben als Groß-/Klein-Äquivalente zu gelten haben, erfolgt sprachspezifisch in `Alphabet.txt`. So lässt sich z.B. für das Französische *E* als Äquivalent zu *é* definieren.

---

<sup>1</sup> In der Anwendung besteht kein Unterschied zwischen beiden. Falls ein Mehrwortlexem ein Leerzeichen enthält, kann im Text stattdessen auch ein Zeilenumbruch stehen.

<sup>2</sup> Derzeit lassen sich Mehrwortlexeme nicht mit Unitex flektieren.

### 3.2 DELAS

Format:

$\langle \text{Grundform} \rangle, \langle \text{Flexionsklasse} \rangle [ + \langle \text{klass. Merkmal} \rangle ]^* [ / \langle \text{Kommentar} \rangle ] ?$

$\langle \text{Flexionsklasse} \rangle$  ist ein Verweis auf einen Transduktor, der die zur Flexion notwendigen Operationen beschreibt. Dabei bezeichnet 'L', dass das Wort um ein Zeichen von hinten gekürzt wird. Das entfernte Zeichen wird dabei auf einen Stack geschoben. 'R' entfernt das erste Zeichen vom Stack und fügt es dem Wort wieder an, 'C' kopiert es an das Wortende. Für alle anderen Zeichen wird dieses dem Wort angefügt und ein Zeichen vom Stack entfernt. **Inflect** flektiert die Einträge eines DELAS-Lexikons und schreibt ein DELAF-Lexikon.

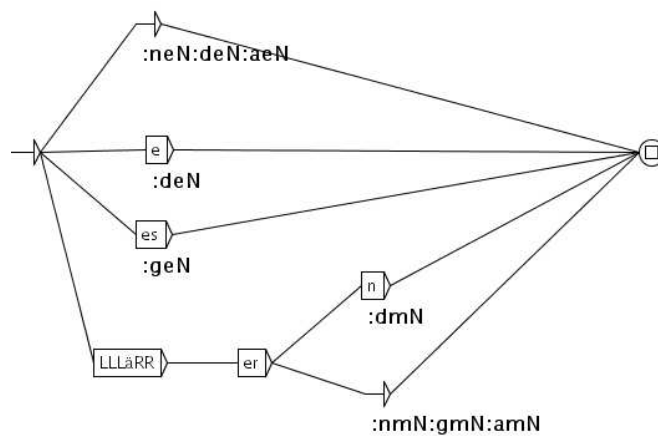


Abbildung 2: Flexionsgraph N2.grf

Haus, N2

wird durch **Inflect** mittels N2.grf (Abb. 2) expandiert zu:

Haus, Haus . N : neN : deN : aeN	
Hauses, Haus . N : geN	Häuser, Haus . N : nmN : gmN : amN
Hause, Haus . N : deN	Häusern, Haus . N : dmN

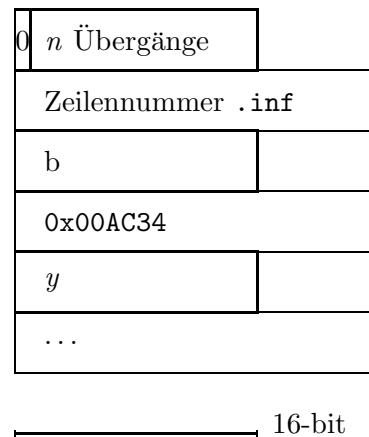
### 3.3 Kompression des Lexikons

Für die Anwendung mit Unitex wird mit **Compress** das Lexikon als DAWG (directed acyclic word graph) gepackt. Die Kompression ist abhängig von der Struktur des Lexikons (insbesondere der Sprache): englisch (10:1), deutsch/russisch (80:1).

nicht-terminaler Zustand:



terminaler Zustand:



Die Binärdatei `*.bin` enthält den DAWG, die Textdatei `*.inf` die klassifikatorischen und grammatischen Merkmale des Eintrags sowie – als Substringoperation kodiert – die Grundform des Lemmas.

### 3.4 Prioritäten bei der Anwendung der Lexika

Unitex unterscheidet drei Prioritäten bei der Anwendung der Lexika, falls der Dateiname eines Lexikons (ohne Endung `.bin`) auf `-` bzw. `+` endet:

1. `*-.bin`, 2. `*.bin`, 3. `*+.bin`

Token, die einem der Lexika einer Prioritätsebene gefunden wurden, werden in keinem Lexikon mit nachgeordneter Priorität mehr nachgeschlagen. So lassen sich z.B. bestimmte Lesarten für ein Token erzwingen. Innerhalb einer Prioritätsebene werden alle Lexika gleichrangig behandelt, d.h. verschiedene Lesarten eines Tokens aus unterschiedlichen Lexika werden ins Textlexikon geschrieben.

## 4 Suchen von Pattern und Konkordanzen

- siehe auch `Reg2Grf` und `Grf2Fst2` in Kap. 8
- `Locate` wendet Graphen auf Text an, erstellt Konkordanz (`concord.ind`). Wichtige Optionen:  
  - s/l/a kürzeste, längste, alle Treffer
  - i/m/r Verhalten, falls Graph Transduktor: Output bleibt unberücksichtigt (`ignore`), wird eingefügt (`merge`)<sup>3</sup>, ersetzt Treffer (`replace`)
- `Concord` gibt die Konkordanz in verschiedenen Formaten aus (HTML, Text, Glossanet). Weitere Optionen betreffen die Länge des Kontextes und die Sortierung der Treffer.  
 Das Ausgabeformat `'txt'` erzeugt eine `*.snt`-Datei, die durch die Transduktionen des gematchten Graphen (siehe Optionen von `Locate`) verändert wurde. Mittels ‚Tags‘ (z.B. `{letzte Woche, .ADV+temp}`), auf die im nächsten Durchlauf mit Metesymbolen referiert werden kann, lassen sich so Transduktoren kaskadieren.

<sup>3</sup> Transduktoren-Output wird links vom gematchten Text eingefügt

- `Extract` extrahiert alle Sätze (Separator ist '{S}'), die Treffer [nicht] enthalten.
- `ConcorDiff` vergleicht zwei Konkordanzen (`concord.ind`).

## 5 Textautomaten

Repräsentation des Textes als Automat, bestehend aus Teilautomaten mit jeweils einem Satz. Jedes Token ist in allen seinen ambigen Lesarten, die sich aus dem Lexikon ergeben, dargestellt.

Auf dem Textautomaten operiert das Programm ELAG (Kap. 6), auch eine manuelle Disambiguierung ist möglich.

- `Txt2Fst2` `<Text>` `<Alphabet>` erzeugt Textautomaten (`<Text>_snt/text.fst2`). Optionale Argumente sind:
  - `[-clean]` entfernt „schlechte“ Pfade aus dem Textautomat, d.h. Pfade, die unbekannte Token enthalten, wenn gleichzeitig ein alternativer Pfad mit ausschließlich bekannten Token (‚Lexikoneinträgen‘) existiert.
  - `[[normalization grammar]]` Der angegebene Automat wird zur Normalisierung verwendet (üblicherweise `<Sprache>/Graphs/Normalization/Norm.fst2`)
- `ImploseFst2` verkleinert den Textautomaten, indem Zustände, die sich nur durch grammatische (flektivische) Merkmale unterscheiden zusammengezogen werden:  
`Männer ,Mann .N+Hum :nmM + Männer ,Mann .N+Hum :gmM = Männer ,Mann .N+Hum :nmM :gmM`
- `ExploseFst2` analog s.o.
- `Fst2Grf` konvertiert Textautomat (kompilierter Graph `*.fst2`) in darstellbare Graphen (`*.grf`)
- `Evamb` berechnet durchschnittliche Ambiguitätsrate eines Textautomaten
- `MergeTextAutomaton` merged manuelle Änderungen am Textautomaten, d.h. an den zugehörigen darstellbaren Graphen (`*.grf`) in den Textautomaten

## 6 ELAG

Tool, um Ambiguitäten in einem Textautomaten zu eliminieren („Tagging“ mittels lokaler Grammatiken). Die Ergebnisse der Disambiguierung kommen `Locate` aber nicht zugute, da dieses nicht auf dem Textautomaten sucht! Siehe die Programme `Elag` und `ElagComp`, weitere Dokumentation auf der Unitex-Homepage.

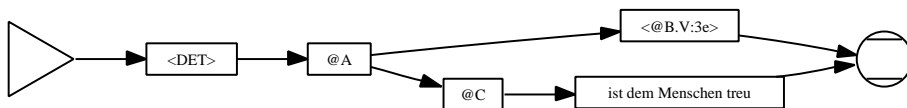
## 7 Lexikongrammatik

Eine Lexikongrammatik enthält syntaktisch-kombinatorische Merkmale einzelner Lemmata in tabellarischer Form. `Table2Grf` wandelt diese Tabelle mittels eines Referenzgraphen (Schablone) in eine Grammatik (Graph mit Subgraphen für jedes Wort) um.

- Die Tabelle muss als `<tab>`-getrennte Textdatei vorliegen, z.B.

A	B	C
Tier	Verb	treu
Hund	bellen	+
Katze	miauen	-
Fuchs	bellen	-

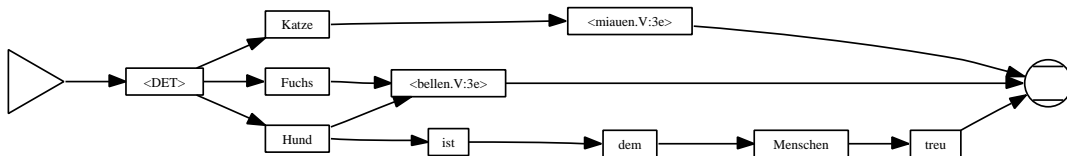
- Der Schablonen-Graph enthält Verweise auf die Tabellenspalten ('@A' etc.)



Dabei bedeutet '+', dass ein Pfad durch die angegebene Box möglich ist. Bei '-' wird der Pfad an der Position der Box gekappt. Andere Symbole oder Wörter werden in die entsprechende Box in der Position '@(id)' eingetragen.

'!@(id)' kehrt die Bedeutung von '+' und '-' um; '@%' gibt die Zeilennummer der Tabelle aus.

- Table2Grf generiert daraus für jede Tabellenzeile (die erste ausgenommen) einen Graphen. Alle generierten Graphen werden in einen Graphen „gelinkt“, der dann folgendem entspricht:



## 8 Tools für Graphen

- **Reg2Grf** wandelt reguläre Ausdrücke in äquivalente Graphen (\*.grf) um
- **Grf2Fst2** kompiliert einen Graphen (\*.grf in \*.fst2). Kompilierte Graphen sind:
  - tokenisiert: Boxen, die mehr als ein Token enthalten werden aufgeteilt (aus mehreren Token bestehender Text "in Deutschland" oder Alternationen "und+oder")
  - invertiert: Übergänge, nicht Zustände/Boxen sind mit Symbolen gelabelt
  - deterministisch: von jedem Zustand existiert nur ein Übergang gleichen Labels<sup>4</sup>
  - inkorporieren alle Subgraphen in einer Datei

<sup>4</sup> Prinzipiell sind Unitex-Graphen jedoch aus drei Gründen indeterministisch, was jedoch erst aufscheint, wenn die Label der Übergänge (=Text in den Boxen) interpretiert werden:

- wenn zwei Subgraphen mit dem gleichen Symbol beginnen und aus einem Zustand mit diesen Subgraphen gelabelte Übergänge bestehen
- Metasymbole (<N>, ADJ) sind mit Subgraphen vergleichbar: sie repräsentieren eine Menge von Terminalsymbolen, darunter auch komplexe
- zwei wörtlich verschiedene Transduktorboxen (bzw. -labels) können die gleichen Inputsymbole aufweisen: **the/foo** und **the/bar**. Unitex nimmt keine Umformung eines ‚ambiguen Transduktors‘ zur dessen Determinisierung vor.

- `Fst2List` gibt alle Pfade des Graphen als Liste aus
- Unitexgraphen gehören zur Klasse der ‚Recursive transition Networks‘ und unterscheiden sich von der Klasse der ‚Endlichen Automaten‘ dadurch, dass weitere Graphen (Automaten) aufgerufen werden können. Dies schließt rekursive Aufrufe ein. `Flatten` inkorporiert bis zu einer einstellbaren Rekursionstiefe alle Subgraphen. Als weitere Option stehen zur Auswahl:

FST [finite-state automaton] Bis zur eingestellten Rekursionstiefe werden alle Rekursion expandiert, danach alle weiteren Rekursionen „gekappt“.

RTN [recursive transition network] Subgraphenaufrufe bis zur eingestellten Tiefe werden expandiert; jenseits der eingestellten Tiefe wird – wie im Original – ein Subgraph aufgerufen.

Bei (rekursiv) verschachtelten Graphen wird `Locate` durch die Anwendung von `Flatten` erheblich beschleunigt.

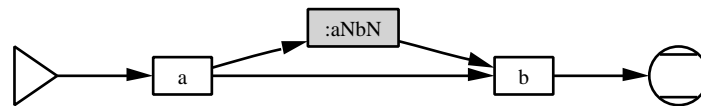


Abbildung 3: Graph aNbN

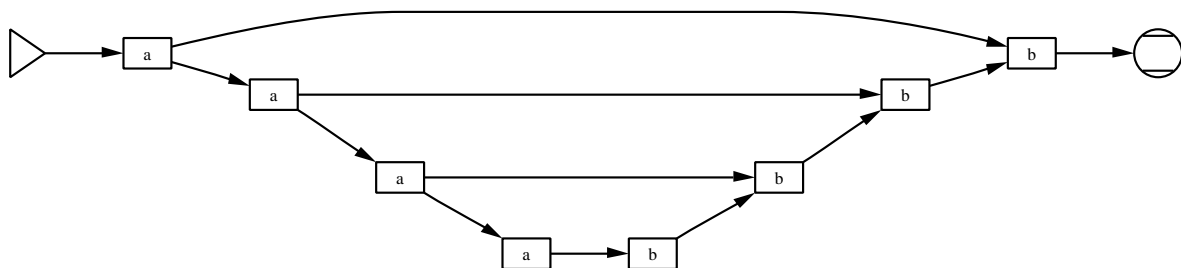


Abbildung 4: Graph aNbN nach `Flatten aNbN.fst2 FST 3`

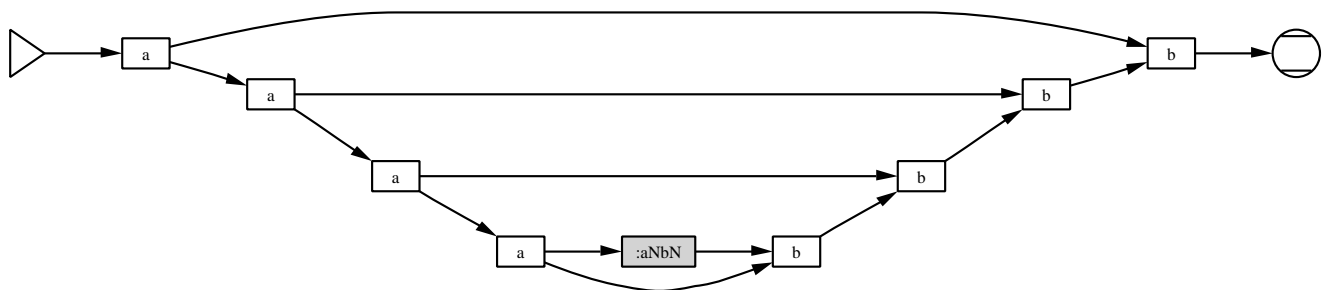


Abbildung 5: Graph aNbN nach `Flatten aNbN.fst2 RTN 3`

## 9 Bugs, Unzulänglichkeiten, häufige Fehler

- Flatten kann nicht äquivalente Automaten produzieren (known bug) – fixed!
- ‘<A>’ matcht nicht Lemma ‘A’
- relative Pfade der Subgraphen werden relativ zum obersten Graphen, nicht zum aufrufenden berechnet – fixed!
- Tokenisierung von Graphen beruht nicht auf `Alphabet.txt`, sondern auf fest implementierten (Unicode-)Tabellen – fixed: Tokenisierung mittels `Alphabet.txt` optional
- Output ambiguer Transduktoren ist nicht definiert. So ist gg. die beiden Graphen  
`graph: unfAmbTrDuc`                      `graph: infAmbTrDuc (with a loop over the subgraph)`

```

>>  - | "the"/ foo | -
    / ----- \
    \ ----- / [0]
    - | "the"/ bar | -
      -----

>>  - | the/ bla | --- | :unfAmbTrDuc | --- [0]
    / ----- \
    \ ----- /
    - | "the"/ bar | -
      -----

```

das Output des Graphen ‘infAmbTrDuc’ für die Eingabe ”the the the the”:

```

bla foo foo foo (Fst2Txt)
bla bar bar bar (Locate)
bla foo bar bar (Fst2Txt or Locate, fst2 flattened)

```

- teilweise beseitigt: Locate gibt jetzt alle möglichen Outputs aus. Wenn der Text verändert werden soll (replace-mode: Locate oder Grf2Fst2), bleibt es aber weiterhin zufällig.
- **Locate** matcht ‚greedy‘, d.h. links stehende Subgraphen „fressen“ soviel Text sie können (wichtig falls ambiguer Transduktor)
- innerhalb < > ist korrekte Groß-/Kleinschreibung verbindlich: <baum> matcht Formen des Lemmas BAUM nur, wenn dieses auch klein im Lexikon geführt wird
- beachte die Match-Konvention bzgl. Groß- und Kleinschreibung (Kap. 3.1)
- Subgraphenaufrufe müssen ohne Dateiendung `.grf` erfolgen
- **Grf2Fst2** ohne Textlexika (betrifft z.B. Satzenderkennung), und ohne reguläre Ausdrücke (innerhalb <<...>>)
- Tokenisierung: kein Zugriff auf Elemente unterhalb Tokenebene in **Locate**, z.B. Graph für Numeralkomposita wie *zweiundzwanzig*

## 10 Literatur und Links

Unitex-Homepage (<http://www-igm.univ-mlv.fr/~unitex/>)

Intex-Homepage (<http://intex.univ-fcomte.fr/>)