

Unitex: Metazeichen in Lexikon, Reg. Ausdrücken und Graphen

Metazeichen existieren in Unitex auf drei Ebenen:

1. im Lexikon (DELAF) und innerhalb "< >" trennen sie die Felder Flexionsform, Grundform, Klasse und flektive Merkmale voneinander; die Merkmalskodierung ist lexikonspezifisch
2. in den Boxen von Graphen markieren sie Subgraphen, Transduktionen, Variablen und beeinflussen die Interpretation des Boxentextes
3. in regulären Ausdrücken kennzeichnen sie Konkatenationen, Alternationen und Rekursionen. Reguläre Ausdrücke werden von Unitex zu einem Graphen umgeformt (`regex.grf`). Deshalb sind alle Metazeichen, die in Graphen verwendet werden können, auch in regulären Ausdrücken gültig. Falls sie Zeichen enthalten, die innerhalb reg. Ausdrücke Metazeichen sind, müssen diese geschützt werden. Sind die Zeichen auch in den Graphen Metazeichen, ist doppelter Schutz notwendig ("`\\\\"` matcht '+').

Die beiden Graph-Matcher in Unitex unterscheiden sich darin, welche Metasymbole sie interpretieren:

§ kennzeichnet Symbole, die (derzeit) nur von `Fst2Text` nicht aber `Locate` interpretiert werden;

* Symbole (insbesondere lexikalische) die nur in `Locate` gültig sind.

| Symbol | Kontext | Bedeutung und Beispiel |
|-----------|---------|--|
| . | Re | Konkatenation (gleichbedeutend mit dem Leerzeichen) |
| + | Gr/Re | Alternation |
| * | Re | Kleene-Stern |
| () | Re | zum Gruppieren von Ausdrücken |
| : | Gr | leitet Subgraph ein |
| : | Gr | plattformunabhängiger Pfadtrenner in Dateinamen eines Subgraph: <code>:dir:graph</code> sucht Subgraphen <code>dir/graph.grf</code> (Unix) bzw. <code>dir\graph.grf</code> relativ zum aufrufenden Graphen |
| : | Gr | zwei Doppelpunkte zu Beginn eines Subgraphennamens: Subgraph wird in Graphenbibliothek gesucht (<code>::pers/pers</code>) |
| / | Gr | leitet Ersetzungstext ein (für Transduktor) |
| / | Lex | Kommentarzeichen (,ab hier bis Zeilenende: davorstehende Leerzeichen gelten als Teil des Lexikoneintrags!') |
| # | Gr | verhindert Match von Leerzeichen an dieser Position |
| \ | alle | dient zum Schutz von Sonderzeichen. Literales '\' durch '\\\'. |
| \$ | Gr | für Variablen: <code>\$name(... \$name)</code> zum Einfangen eines Teilgraphen, <code>\$name</code> zur Ausgabe im Ersetzungstext des Transduktors; in Variablenamen sind erlaubt: <code>[A-Za-z0-9_]</code> . Variablen sind global definiert, d.h. in Subgraphen definierte Variablen können auch im übergeordneten Graphen oder in anderen Subgraphen ausgegeben werden. |
| [\$ | Gr | Kontextoperatoren <code>[\$</code> (Beginn), <code>]</code> (Ende), <code>![\$</code> (Beginn negativer Kontext); Kontexte sind äquivalent zum Lookahead-Operator (<code>(?=...)</code> , <code>(?!...)</code>) in Perl-kompatiblen regulären Ausdrücken |
| ![\$ | | |
|] | | |
| <Token> | Gr | Token mit (Groß/Klein-)Varianten gemäß Alphabetdatei: <code>being</code> matcht <i>Being</i> , <i>BEING</i> usw. |
| "<Token>" | Gr | Token in exakt dieser Schreibung: <code>"being"</code> |
| <Text> | Gr | Folge von Token (zu " s.o.). Der Text wird von Unitex beim Kompilieren des Graphen tokenisiert |
| "<Text>" | | |
| {<Tag>} | Gr/Text | lexikalisch annotiertes Token (,Tag'): <code>{Hasen,Hase.N+anim:nmM}</code> . Die interne Syntax eines Tags entspricht der eines Lexikoneintrags. |
| {S} | Gr/Text | Trennt Sätze im Text; auf {S} kann in Graphen Bezug genommen werden |

| Symbol | Kontext | Bedeutung und Beispiel |
|-----------------------|-----------------|---|
| {STOP} | Text | Trennt Textchunks im Text; auf {STOP} kann in Graphen nicht Bezug genommen werden; über {STOP} ist kein Match eines Graphen möglich |
| < > | Gr | schließt Metaausdruck ein \Rightarrow die Semantik einiger Zeichen ändert sich innerhalb von < > (s.u.)! |
| <^> | Gr ^s | Zeilenumbruch |
| <E> | Gr | leeres Wort (ϵ) |
| <MOT> | Gr | Zeichenfolge, bestehend aus Buchstaben (siehe <code>Alphabet.txt</code>) |
| <MIN> | Gr | Zeichenfolge (nur) aus Kleinbuchstaben |
| <MAJ> | Gr | Zeichenfolge (nur) aus Großbuchstaben |
| <PRE> | Gr | Folge von Buchstaben, beginnend mit einem Großbuchstaben: <i>Oder, KPdSU, AG</i> |
| <L> | Gr ^s | einzelner Buchstabe |
| <DIC> | Gr* | Token oder Tokenfolge, im Wörterbuch enthalten |
| <SDIC> | Gr* | einfacher Wörterbucheintrag (bestehend aus einem Token) |
| <CDIC> | Gr* | komplexer Wörterbucheintrag (mehr als ein Token): <i>take place</i> |
| <NB> | Gr | Ziffernfolge ([0-9]+) |
| <PNC> | Gr ^s | Interpunktion |
| <TOKEN> | Gr* | beliebiges Token, d.h. eine Folge von Buchstaben (<MOT>), ein ‚Tag‘ (<code>{heute, .ADV}</code>), die Satzendmarkierung (<code>{S}</code>) oder ein beliebiges anderes Zeichen; nicht aber {STOP} |
| <<G>> | Gr* | alle Formen eines Lemmas mit der Grundform <G>: <code><be> = am, are, is, ...</code> Achtung: Groß- / Kleinschreibung muss der im Lexikon entsprechen! |
| <<K>> | Gr* | alle Token, die der Wortart <K> zugeordnet werden können bzw. das Merkmal <K> tragen: <code><N></code> , <code><ADV></code> , <code><Hum></code> |
| <.<K>> | Gr* | dito: <code><.N></code> , <code><.ADV></code> , <code><.Hum></code> , aber ohne Ambiguität zwischen <K> und <G> |
| <<G>.<K>> | Gr* | alle Formen eines Lemmas mit Grundform <G> und der Wortart / dem Merkmal <K> angehört (und ev. weitere Merkmale aufweist, s.u. + und -): <code><be.V></code> , <code><house.N></code> |
| <<K>:<F>> | Gr* | alle Token mit Wortart/Merkmal <K>, die die grammatischen (flektivischen) Merkmale <F> aufweisen: <code><N:s></code> matcht alle Singularformen eines jeden Nomens |
| <<G>.<K>:<F>> | Gr* | Kombination der beiden oberen: <code><Haus.N:s></code> matcht alle Singularformen des Lemmas ‚Haus‘ |
| <<Form>, <G>.<K>:<F>> | Gr* | Kombinationen der oberen mit explizit angegebener Form: <code><Houses, Haus.N></code> Form <i>Houses</i> des Lemmas ‚Haus‘ (Nomen); äquivalent zu <Token> (hier: <i>Houses</i>) |
| ! | < > | Negation, Komplement: erlaubt nur in den Kombinationen <code><!MOT></code> (alle Token, die nicht aus Buchstaben bestehen, Leerzeichen und Satzendmarke ausgenommen), <code><!MIN></code> , <code><!MAJ></code> , <code><!PRE></code> , <code><!DIC></code> (alle Token vom Typ <MOT>, die nicht durch <MIN> erkannt werden), sowie allen Ausdrücken, die nur grammatische, semantische und flektivische Angaben enthalten (<code><!Haus.N:g></code> ist demnach nicht erlaubt!), z.B. findet <code><!N-Hum:s></code> alle Formen des Typs <MOT>, auf die <code><N-Hum:s></code> nicht zutrifft. |
| + | < > | zusätzliches Merkmal muss vorhanden sein: <code><.N+Hum></code> matcht Nomina mit Merkmal Hum |

| Symbol | Kontext | Bedeutung und Beispiel |
|--------------|---------|--|
| | Lex | Trennt im Lexikon klassifikatorische („semantische“) Merkmale voneinander (Alfreds, Alfred.N+Hum+Vorname:geM) |
| - | < > | Merkmal darf nicht vorhanden sein: <-Hum>, <.N-Hum> |
| : | < >/Lex | leitet Folge von flektivischen Merkmalen (jedes Zeichen steht für ein Merkmal) ein: Hauses.N:geN, Häuser.N:nmN:gmN:amN; die Reihenfolge spielt beim Match keine Rolle; alle Merkmale müssen, weitere dürfen vorhanden sein: <N:s> matcht alle Singularformen aller Nomina, d.h. N:ns, N+Hum:gs, etc. <N:ng> matcht vermutlich gar nichts, <N:n:g> Formen im Nominativ oder Genitiv |
| , | < >/Lex | trennt Wortform und Lemma |
| . | < >/Lex | trennt Lemma und Metainformation |
| = | Lex | Bindestrich oder Leerzeichen: Saudi=Arabien, .EN+Topon... -> |
| << >> | Gr* | schließt POSIX-konformen regulären Ausdruck ein, der die Matches des vorausgehenden Unitex-Patterns (fehlt dieses wird <TOKEN> angenommen) noch einmal filtert: <V><<^trav>> matcht alle Verben, die mit der Zeichenfolge <i>trav</i> beginnen, <<^..\$>> alle zweiseymboligen Token. |
| << >>_<opt>_ | Gr* | s. o., mit Optionen <opt>: <i>f</i> <i>case-sensitive</i> , d.h. regulärer Ausdruck wird nicht mit Alphabetdatei überladen: <V><<^[A-ZÄÖÜ]>>_f_ sucht nach großgeschriebenen Verbformen <i>b</i> <i>basic reg. expr.</i> Standard sind also erweiterte (<i>extended</i>) reg. Ausdrücke und Suche unabhängig von Groß- und Kleinschreibung |
| <.> | | |
| <!> | Elag | |
| <=> | | |