Und oder

Die Syntax eines regulären Ausdrucks lässt sich im Grunde als eine *und*-Verkettung umschreiben. Der bereits bekannte Ausdruck *l<d.+* kodiert eine Zeichenfolge, bei der ein *d* am Wortanfang steht und von mindestens einem beliebigen Zeichen gefolgt wird. Anders gesagt matcht der Ausdruck auf eine Zeichenfolge in einem Text, wenn sie sich am Wortanfang befindet UND mit einem *d* startet UND diesem mindestens ein weiteres Zeichen folgt. Mit wachsender Erfahrung wird man jedoch auch Ausdrücke schreiben wollen, die mit *oder*-Operatoren arbeiten. Hierzu ein Beispiel aus dem Französischen:

In einem Korpus der gesprochenen Sprache sollen alle Vorkommen eines verneinten "c' est" gefunden werden. Nun ist im mündlichen Sprachgebrauch insbesondere mit folgender Variation zu rechnen:

1) ce n' est pas.

2) c' est pas.

Ohne einen *oder*-Operator müsste man hier zwei Suchanfragen formulieren, was in diesem Beispiel noch durchaus möglich wäre. Nun müssen wir aber auch damit rechnen, dass mit der Transkription der Audiodateien mehrere übermüdete studentische Hilfskräfte beschäftigt waren. Von diesen ließen einige eine Leerstelle auf das Apostroph folgen, andere nicht. Dies bewirkt, dass wir sogar 4 Suchanfragen starten müssten, könnten wir unsere Unsicherheit nicht durch ein *?* ausdrücken. Meine dadurch auf *c'_?est pas* auf *ce n'_?est pas* reduzierbare Ausdrücke ließen sich umschreiben als: SUCHE (*c'* gefolgt von einem Leerzeichen ODER nicht, gefolgt von einem *est pas*) UND SUCHE (*ce* gefolgt von einem *n'*, gefolgt von einem Leerzeichen ODER nicht, gefolgt von einem *est pas)*. Die beiden Suchanfragen können aber auch zusammengefasst werden unter: SUCHE (*c* gefolgt von ((*'* gefolgt von (einem Leerzeichen ODER nichts)) ODER *(e* gefolgt von einem Leerzeichen)), gefolgt von ((*n'* gefolgt von (einem Leerzeichen ODER nichts)) ODER nichts), gefolgt von *est pas*). Hä was? Die Schritt-für-Schritt-Anleitung folgt später.

Einige der bekannten Metazeichen stellen eine Möglichkeit dar, ein "oder" zum Ausdruck zu bringen. So sucht *fatt[oa]* zum Beispiel nach der dem Wortsegment *fatt* gefolgt von einem *o* ODER einem *a*. Schwieriger wird es, will man ein ODER zwischen zwei unterschiedliche Zeichenfolgen setzen. Nehmen wir an, wir bräuchten einen Ausdruck, der sowohl mit *fame* als auch mit *faro* matcht. Wir könnten zu diesem Zweck *fa[mr][eo]* verwenden, allerdings würden wir Gefahr laufen, auch *fare* (und *famo*) in unser Ergebnis einzuschließen. Um dies zu vermeiden, brauchen wir ein paar in diesem Manual noch nicht eingeführte Metazeichen:

```
() |
```

Die Klammern werden verwendet, um Zeichenfolgen zu gruppieren, was sich als wirklich praktisch erweist, will man Wiederholungsoperatoren auf eine Zeichenfolge und nicht lediglich auf einzelne Zeichen beziehen. Will man die Folge ABABAB in ABCABABABAD matchen, so lässt sich dies am einfachsten erzielen, wenn man mithilfe der runden Klammern *AB* als Gruppe definiert und ihr den Wiederholungsoperator *{3}* zuweist: *(AB){3}.* Man sei daran erinnert, dass einige Standards erfordern, dass jeder runden Klammer ein Backslash vorgeschaltet wird, falls sie als Metazeichen verwendet werden. Dies gilt ebenfalls für den senkrechten Strich

Der senkrechte Strich liefert uns schließlich den entscheidenden Operator: das *ODER*. Mit diesem lassen sich einfache Ausdrücke, wie *fatto|a* als Alternative zu *fatt[oa]*, formulieren, doch der große Mehrwert tritt erst in Kombination mit den runden Klammern auf. So lässt sich mit diesen Metazeichen auch der Ausdruck formulieren, der mit *fame* und *faro*, nicht jedoch mit *fare* (oder *famo*) matcht: fa(me|ro). Es ist wichtig, die Anzahl und Position der Klammern zu reflektieren. Befindet sich der senkrechte Strich außerhalb runder Klammern so wird er alles, was sich links von ihm befindet mit allem, was sich rechts von ihm befindet in eine *oder*-Relation. Grenzen können ihm dadurch gesetzt werden, dass er, die Zeichenfolge links und die Zeichenfolge rechts in runde Klammern gesetzt werden. Wie im Schulunterricht beim Eintippen von Rechenoperationen mit Brüchen, sollte man im Zweifelsfall lieber mehr Klammern verwenden, da sich so besser überwachen lässt, welche Zeichenfolgen von dem Operator ge*oder*t werden. Die folgende Darstellung dient der Veranschaulichung der Funktionsweise des senkrechten Striches in Kombination mit den gruppierenden Klammern. Ob Ausdruck a oder b zu verwenden ist, hängt von dem Standard des Editors/ des verwendeten Programms ab. Sollte ein Ausdruck nicht funktionieren, kann das Problem möglicherweise gelöst werden, indem Backslashes hinzugefügt oder getilgt werden.

Ausdruck a	Ausdruck b	matcht...
fame|ro	fame|ro	ENTWEDER (fame) ODER (ro)
fa(me)|(ro)	fa\(me\)|\(ro\)	ENTWEDER (fame) ODER (ro)
fa(me|ro)	fa\(me|ro\)	ENTWEDER (fame) ODER (faro)
fa((me)|(ro))	fa\(\(me\)|\(ro\)\)	ENTWEDER (fame) ODER (faro)
fame|faro	fame|faro	ENTWEDER (fame) ODER (faro)
(fame)|(faro)	\(fame\)|\(faro\)	ENTWEDER (fame) ODER (faro)

Mithilfe dieser beiden Metazeichen können komplexere Ausdrücke formuliert werden, wie zum Beispiel derjenige, der auf *ce n' est pas*, *c' est pas*, *ce n'est pas* und *c'est pas* passt. Hierfür macht es strategisch Sinn, Zeichenfolgen, deren Position sich in allen gewünschten Ergebnissen entspricht, außerhalb der runden Klammern, welche *oder*-Operatoren enthalten werden, zu setzen. Der erste Schritt sähe somit folgendermaßen aus:

c()est pas

Innerhalb der gesetzten Klammer sehen die Möglichkeiten nun wie folgt aus:

					Ergebnis
1a)	e	_	n'		*c(e n'_)est pas*
1b)	e	_	n'	_	*c(e n')est pas*

2a)	'				*c()est pas*
2b)	'	_			*c(_)est pas*

Alle Unterschiede in den Ergebnissen lassen sich nun auch kleinschrittig beschreiben. Auch hier werden gemeinsame Zeichen(folgen) außerhalb der Klammer mit dem *oder*-Operator notiert.

Der Unterschied zwischen 1a) und 1b) betrifft das Leerzeichen. Alles andere sollte sich folglich außerhalb der Klammer befinden (Gemeinsamkeiten sind in der Tabelle blau unterlegt, Unterschiede rot): *(e_n'(|_))*

Die Menge an Zeichen auf einer Seite des senkrechten Striches kann wie in diesem Beispiel auch leer sein. *(|_)* ist gleichbedeutend mit "ENTWEDER nichts ODER Leerstelle".

Der Unterschied zwischen 2a) und 2b) liegt ebenfalls im Leerzeichen. Die Herangehensweise entspricht der vorausgehenden (Gemeinsamkeiten sind in der Tabelle grün unterlegt, Unterschiede rot): *('(|_))*

Die beiden Teilausdrücke lassen sich nun mithilfe des *oder*-Operators verbinden und in *c()est pas* einfügen: *c((e_n'(|_))|('(|_)))est pas*

Bei der Formulierung komplexerer regulärer Ausdruck mit *oder*-Operatoren und Gruppierungsklammern muss bedacht werden:

- Ein senkrechter Strich setzt die sich links von ihm befindende Zeichenkette mit der sich rechts von ihm befindende in eine *oder*-Relation. Sind keine Klammern gesetzt, so wird der gesamte Ausdruck einbezogen.
- Um die Reichweite des oder-Operators einzuschränken, muss er mit den entsprechenden Zeichenketten (links und rechts) in runde Klammern gesetzt werden. Der oder-Operator reicht in dem Fall von der äußersten öffnenden (nicht links von ihm schließenden) Klammer links von ihm bis zur äußersten schließenden (nicht rechts von ihm öffnenden) Klammer rechts von ihm.
- Mithilfe der Klammern lässt sich präzise definieren, was in Alternative wozu steht.
- Eine Alternative kann auch kein einziges Zeichen enthalten.

Previous Escape | Up Reguläre Ausdrücke in der Arbeit mit Textdaten | Ersatz gefunden Next