

Visualisierung regulärer Ausdrücke

Ein Vortrag von Martin Häcker und Felix Schwarz

Was uns beschäftigte...

```
^\(IBOutlet[ ]*\)*\([\_a-zA-Z]\[_a-zA-Z0-9]*\)[ ]*\(\**\)[ ]*\([\ ]*\)\([a-zA-Z]\)\([\_a-zA-Z0-9]*\)[ ]*;/\2/  
^\(IBOutlet[ ]*\)*\([\_a-zA-Z]\[_a-zA-Z0-9]*\)[ ]*\(\**\)[ ]*\([\ ]*\)\([a-zA-Z]\)\([\_a-zA-Z0-9]*\)[ ]*;/\3/  
^\(IBOutlet[ ]*\)*\([\_a-zA-Z]\[_a-zA-Z0-9]*\)[ ]*\(\**\)[ ]*\([\ ]*\)\([a-zA-Z]\)\([\_a-zA-Z0-9]*\)[ ]*;/\4/  
^\(IBOutlet[ ]*\)*\([\_a-zA-Z]\[_a-zA-Z0-9]*\)[ ]*\(\**\)[ ]*\([\ ]*\)\([a-zA-Z]\)\([\_a-zA-Z0-9]*\)[ ]*;/\5/  
^\(IBOutlet[ ]*\)*\([\_a-zA-Z]\[_a-zA-Z0-9]*\)[ ]*\(\**\)[ ]*\([\ ]*\)\([a-zA-Z]\)\([\_a-zA-Z0-9]*\)[ ]*;/\6/
```

Findet die Variablendeklarationen in
Objective-C Programmen

Das Problem

- „Some people, when confronted with a problem, think »I know, I'll use regular expressions.« Now they have two problems.“
Jamie Zawinski, in comp.lang.emacs (from the fortune file)

Was sind reguläre Ausdrücke?

- flexibles Werkzeug zum Parsen von Text
- Syntax nicht ganz einheitlich, hier
Verwendung der Syntax von Perl 5
- lange in Gebrauch, erste Ansätze 1941
- selbst komplexe Muster werden in einer
Zeile kodiert

kleine Syntaxkunde (I)

.	ein beliebiges Zeichen
\d	eine Ziffer (also 0 bis 9)
\w	ein alphanumerisches Zeichen
\s	ein Leerzeichen oder ein Tabulator

kleine Syntaxkunde (II)

<code>\d+</code>	ein oder mehrere Ziffern
<code>\d*</code>	keine oder beliebig viele Ziffern
<code>\d{3}</code>	genau drei Ziffern
<code>\d{2,5}</code>	2 bis 5 Ziffern

kleine Syntaxkunde (III)

(a-z)	erkennt die Zeichen 'a' bis 'z'
(Uni Schule)	erkennt die Strings 'Uni' und 'Schule' (Alternative)
[abc]	erkennt die Zeichen 'a', 'b', 'c' Sonderzeichen werden maskiert, z.B: \.

kleine Syntaxkunde (IV)

`(\d\d),(\d\d) Euro`

Cent-Betrag in \$2, Euro-Betrag in \$1

`(\d\d),\1 Euro`

Beträge, bei denen Cent=Euro sind

Darstellungsprobleme,
kurz analysiert

Existierende Werkzeuge

Kate



Komodo

The screenshot shows the 'Regular Expression Toolkit' window in Komodo. The window title is 'Regular Expression Toolkit'. The main area is titled 'Rx Toolkit' and contains a text input field for a regular expression, which contains `(.*) (\d+)`. Below the input field are several checkboxes for modifiers: Global, Multi-line Mode, Ignore Case, Single-line Mode, and Extended. Below the modifiers is another text input field for a string to match against, which contains `testing123`. At the bottom of the main area, there is a status bar that says 'Match succeeded: 2 groups' and several navigation buttons. Below the status bar is a table titled 'Group Match Variables' with two columns: 'Variable' and 'Value'. The table contains two rows: one for '\$1' with the value 'testing12' and one for '\$2' with the value '3'.

Regular Expression Toolkit

Rx Toolkit

Enter a regular expression

`(.*) (\d+)`

Modifiers: Global Multi-line Mode Ignore Case Single-line Mode Extended

Enter a string to match against

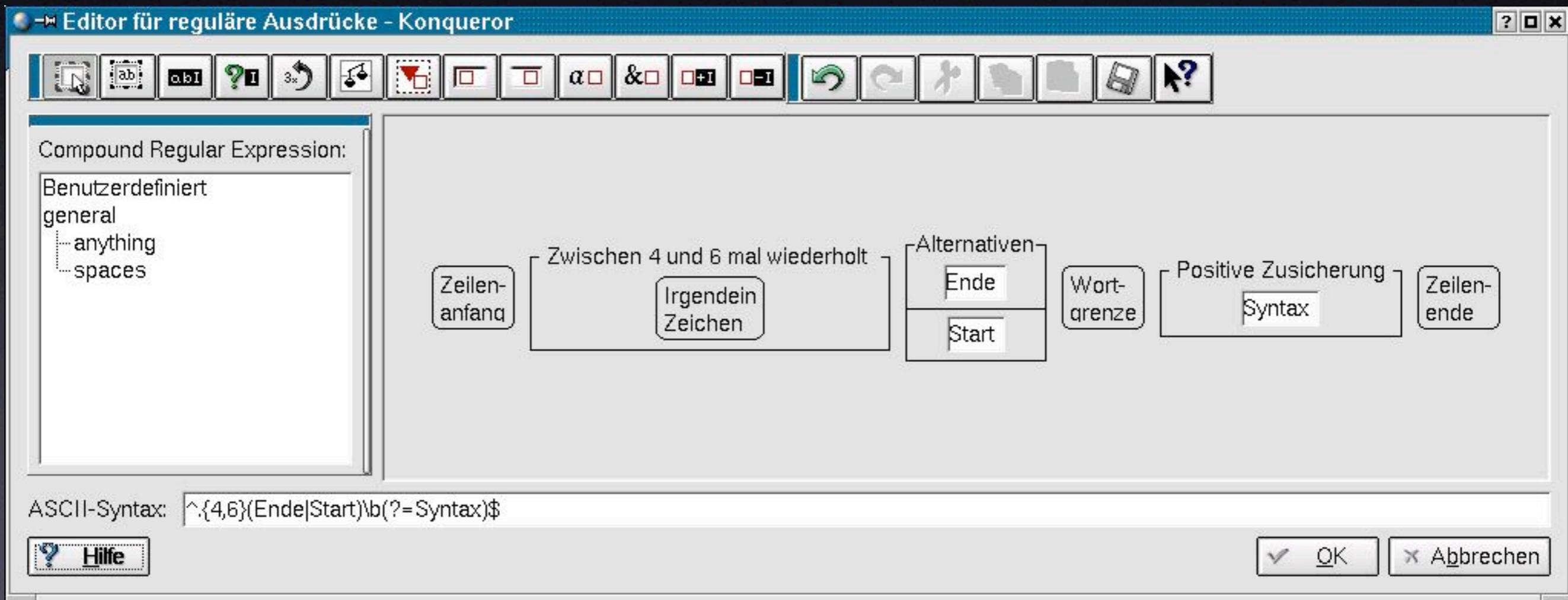
`testing123`

Match succeeded: 2 groups

Group Match Variables

Variable	Value
\$1	testing12
\$2	3

kregexpeditor



Visual Regexp

The screenshot shows the Visual REGEXP 3.0 application window. The title bar reads "Visual REGEXP 3.0". The menu bar includes "File", "Edit", "View", "Select mode", "Insert regexp", and "Help".

The main text area contains the following regular expression pattern: `(?:^|") (http|ftp|mailto):(?://)? (\w+(?:[\.\:\@]\w+)*?) (?:/|@) ([^"\?]*?) (?:\?([^"\?]*?)?)?(?:$|")`. The pattern is color-coded: blue for the protocol, green for the host, yellow for the path, and pink for the query string.

Below the text area are several checkboxes: nocase, all, line, lineanchor (k), linestop (m), and inline.

There are "Go" and "Clear [z]" buttons. A "Select:" dropdown menu is set to "match", followed by buttons for capturing groups 1 through 9.

A "Replace" button is located to the right of a large empty text input field.

The bottom section shows a text area with the URL `http://kroupware.kde.org/architecture-1.1/index.html` highlighted in pink, corresponding to the query string part of the regex. Below this, it displays "1 matches" and "Previous" and "Next" navigation buttons.

Überblick über die Visualisierungstechniken

- Code-Visualisierung
- Ablauf-Visualisierung
- Ergebnis-Visualisierung
- Synergie-Effekte
- Profiling

Syntax-Highlighting

(\\$(?:\d{2}pm)*|(?:sunday)*?.\d?)

(\\$(?:\d{2}pm)*|(?:sunday)*?.\d?)

Syntax-Styling

$(\backslash\$ (?:\d{2}pm)^* | (? :sunday)^* ? \backslash.\backslash\$)$

$(\backslash\$ \underline{(?:\d{2}pm)^*} | \underline{(?: :sunday)^*} ? \backslash.\backslash\$)$

Visualisierung über Muster

$abc(\backslash d+(?:\backslash w+\backslash s))-(?:\backslash ?\backslash .\backslash ^\backslash \$)|(?:\backslash d\backslash d?)(?:\backslash w+)-\backslash 2\backslash \$$

$abc(\backslash d+(\backslash w+\backslash s))-(?:\backslash ?\backslash .\backslash ^\backslash \$)|(\backslash d\backslash d?)(?:\backslash w+)-\backslash 2\backslash \$$

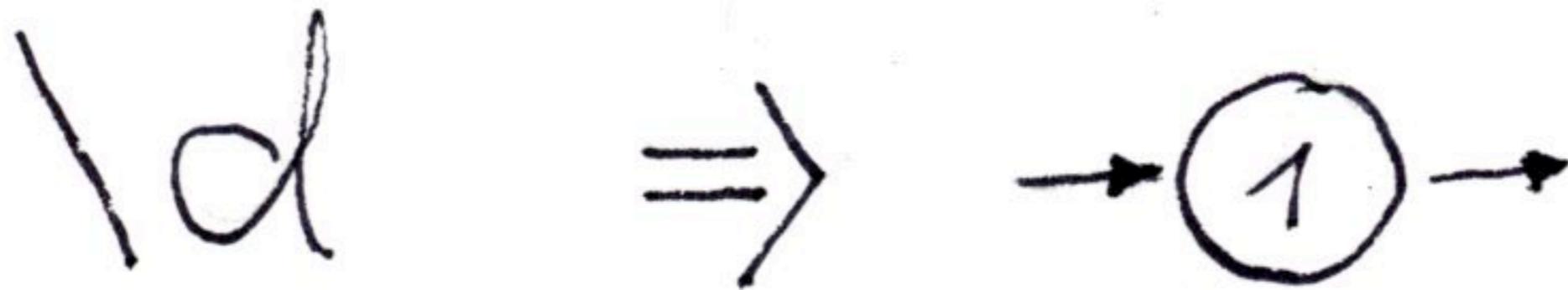
The diagram illustrates the decomposition of the original regex into three parts:

- Part 1: $(\backslash d+(\backslash w+\backslash s))$ (highlighted in blue)
- Part 2: $(?:\backslash ?\backslash .\backslash ^\backslash \$)$ (highlighted in blue)
- Part 3: $(\backslash d\backslash d?)(?:\backslash w+)-\backslash 2\backslash \$$ (highlighted in blue)

A red arrow points from the first part of the original regex to the first part of the decomposed version. A red bracket groups the second and third parts of the decomposed version, with an arrow pointing to the second and third parts of the original regex.

Alien-Highlighting

Alien-Highlighting



Alien-Highlighting



Alien-Highlighting

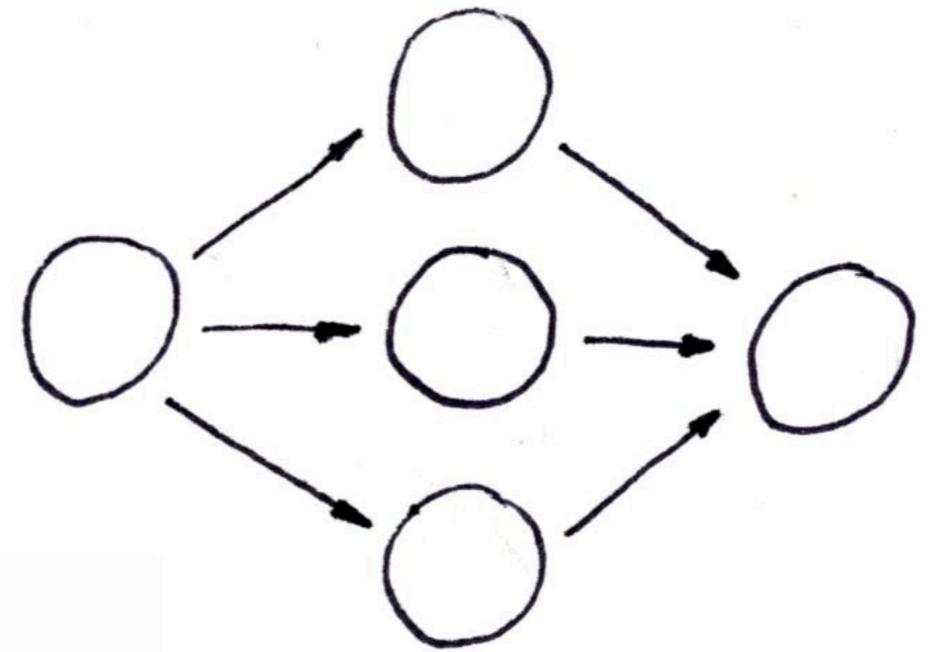
$\backslash d\{, 5\}$

\Rightarrow

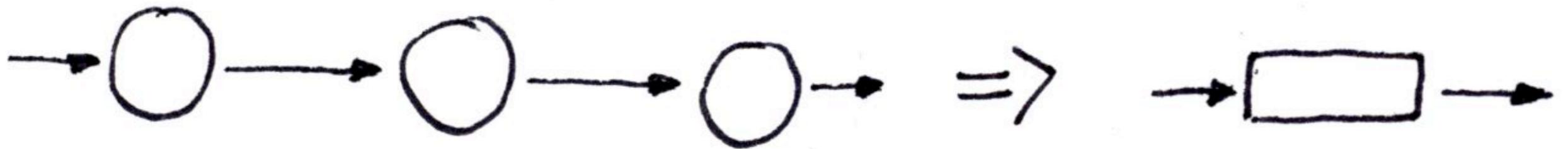


Alien-Highlighting

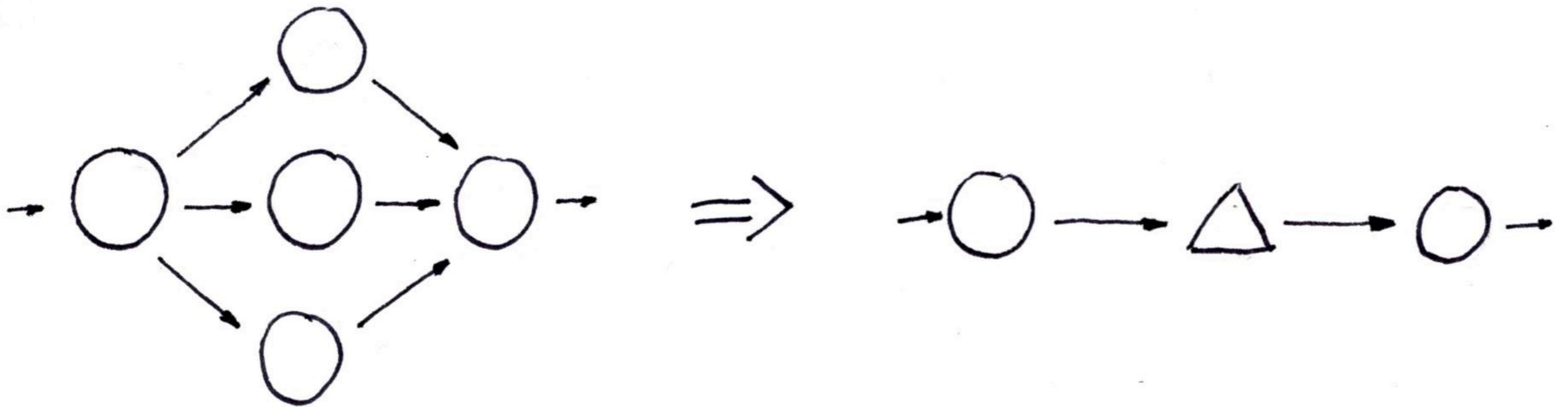
$\setminus d (a/b/c) \setminus d \Rightarrow$



Alien-Highlighting



Alien-Highlighting



Alien-Highlighting

$\setminus d *$ \Rightarrow \rightarrow \rightarrow

Übersetzung in natürliche Sprachen

- `\d` \Rightarrow 0 1 2 3 4 5 6 7 8 9
- `\x21-\x25` \Rightarrow ! " # \$ %
- `^\d{3,5}$` \Rightarrow Eine Zeile die nur eine drei- bis fünfstellige Zahl enthält

Visuelle Programmierung

Ablauf-Visualisierung

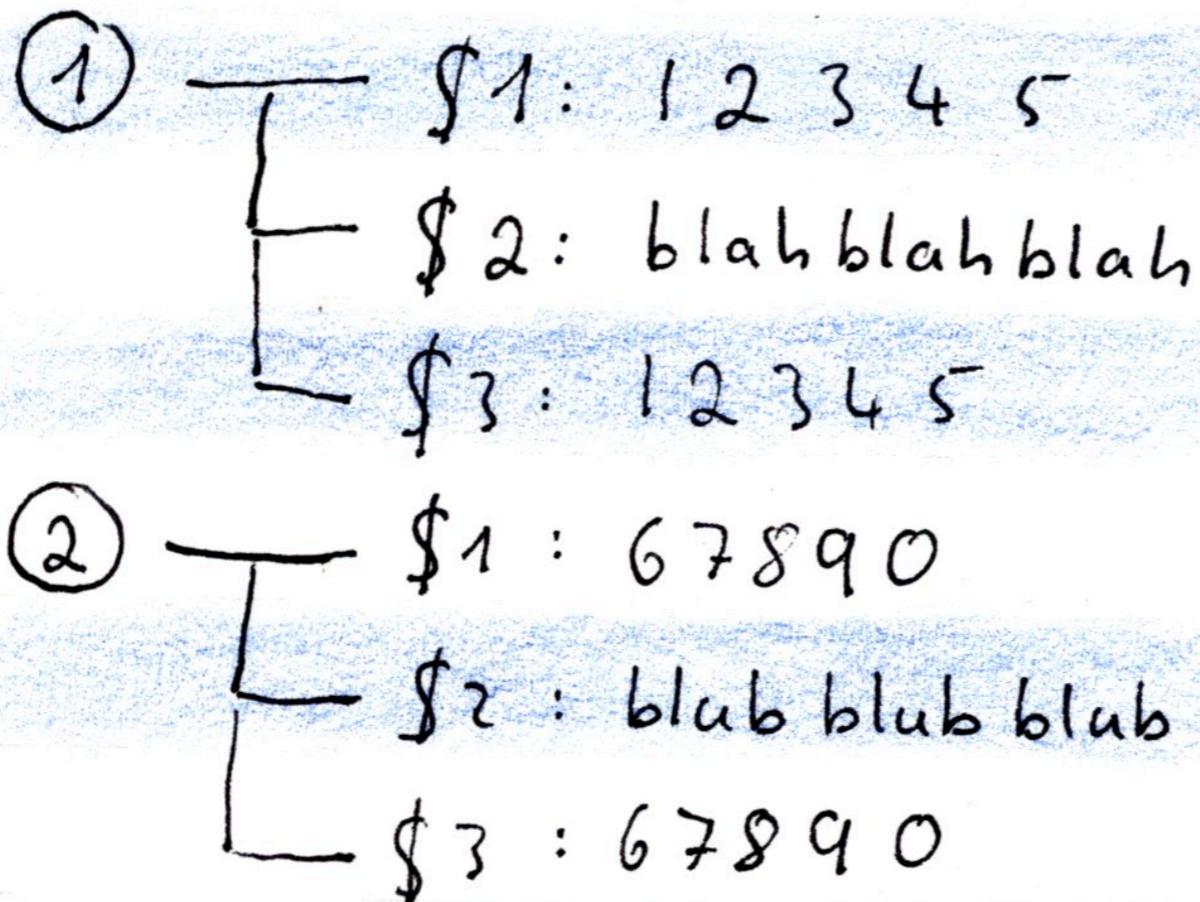
$\wedge (\backslash d\{5\}) \cdot * (\backslash d\{5\}) \$$

1 2 3 4 5 blah blah blah blah 1 2 3 4 5



Ablauf-Visualisierung

Treffer (Global)



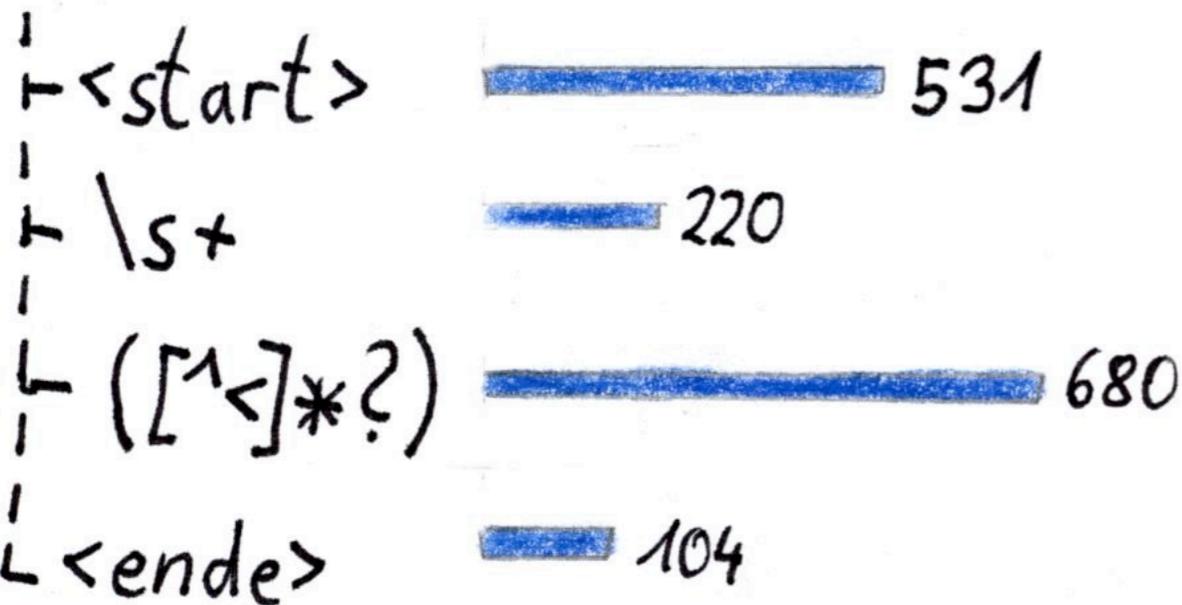
Ergebnis-Visualisierung

Synergie-Effekte

Profiling

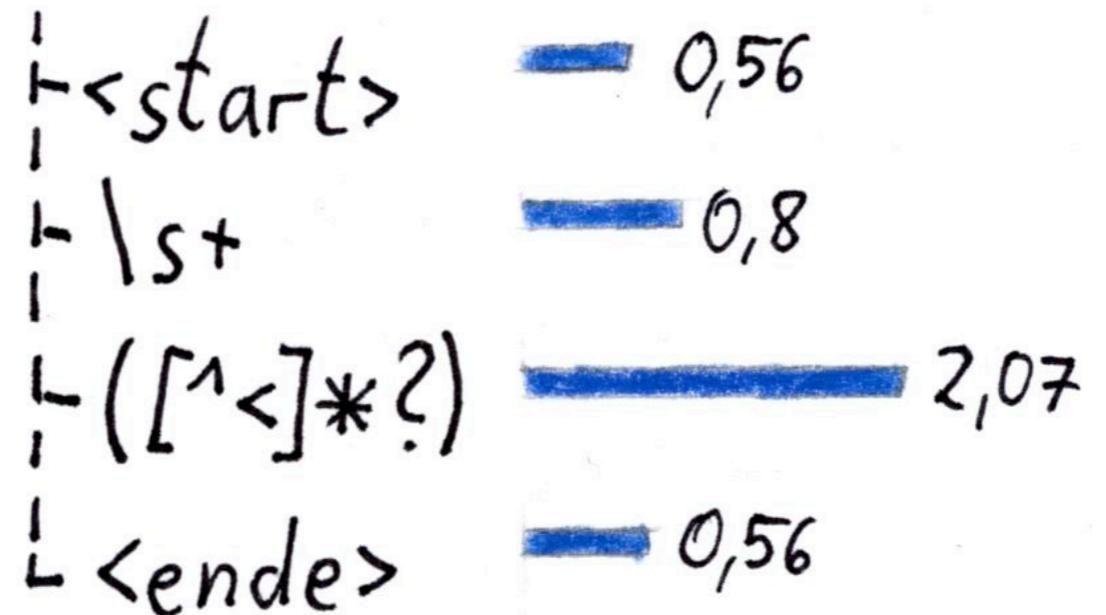
untersuchte Zeichen

<start> \s+ ([^<]*?) <ende>



Zeit pro Aufruf

<start> \s+ ([^<]*?) <ende>



Bewertung existierender Werkzeuge

Kate



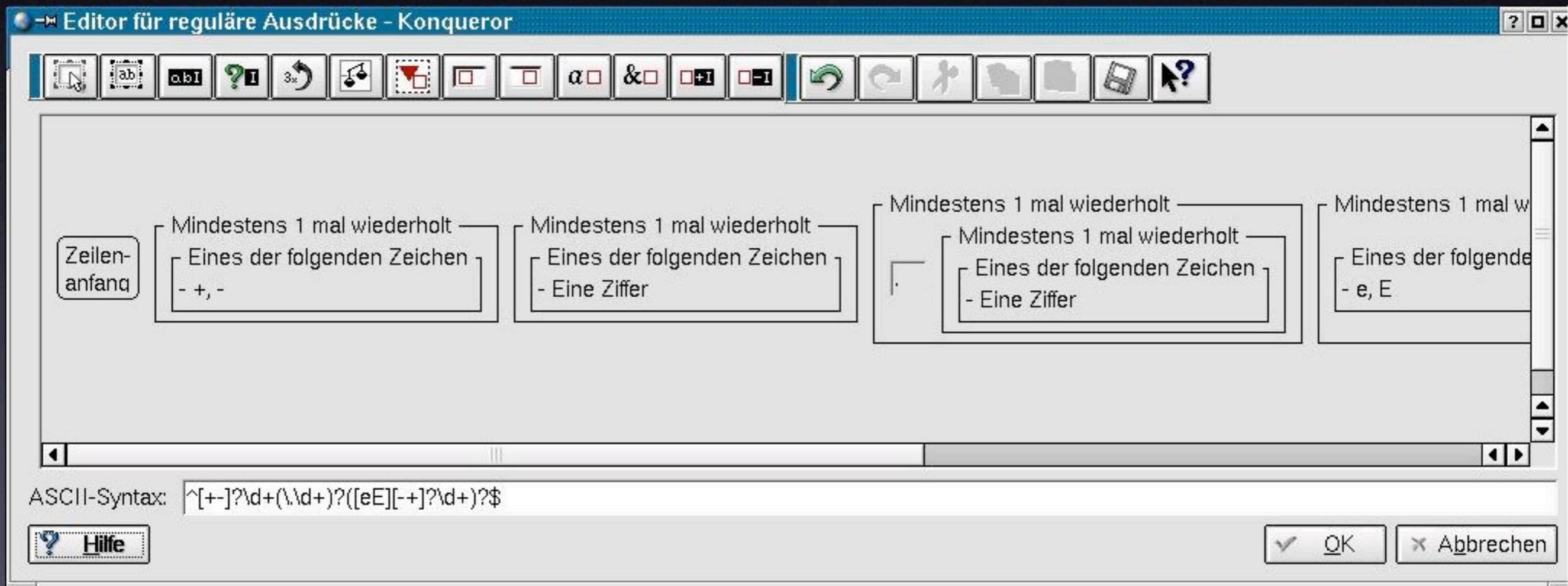
Komodo

The screenshot shows the 'Regular Expression Toolkit' window in Komodo. The window title is 'Regular Expression Toolkit'. The main area is titled 'Rx Toolkit' and contains the following elements:

- A text input field for the regular expression: `(.*) (\d+)`
- A 'Modifiers' section with the following options:
 - Global
 - Multi-line Mode
 - Ignore Case
 - Single-line Mode
 - Extended
- A text input field for the string to match against: `testing123`
- A status bar showing a green circle, a refresh icon, and the text 'Match succeeded: 2 groups'. To the right of the status bar are four navigation buttons: <<, <, >, and >>.
- A section titled 'Group Match Variables' containing a table with two columns: 'Variable' and 'Value'.

Variable	Value
\$1	testing12
\$2	3

kregexpeditor



Visual Regexp

The screenshot shows the Visual REGEXP 3.0 application window. The title bar reads "Visual REGEXP 3.0". The menu bar includes "File", "Edit", "View", "Select mode", "Insert regexp", and "Help".

The main text area contains the following regular expression pattern: `(?:^|") (http|ftp|mailto):(?://)? (\w+(?:[\.\:\@]\w+)*?) (?:/|@) ([^"\?]*?) (?:\?([^"\?]*?)?)?(?:$|")`. The pattern is color-coded: `(?:^|")` is blue, `(http|ftp|mailto):` is red, `(?://)?` is yellow, `(\w+(?:[\.\:\@]\w+)*?)` is green, `(?:/|@)` is yellow, `(?:\?([^"\?]*?)?)` is pink, and `(?:$|")` is yellow.

Below the pattern are several checkboxes: nocase, all, line, lineanchor (k), linestop (m), and inline.

There are "Go" and "Clear [z]" buttons. A "Select:" dropdown is set to "match", followed by buttons for groups 1 through 9.

A "Replace" button is located to the right of a large empty text input field.

The bottom text area shows a single match: `http://kroupware.kde.org/architecture-1.1/index.html`. The match is color-coded to match the pattern above: `http://` is red, `kroupware.kde.org/` is green, `architecture-1.1/` is pink, and `index.html` is pink.

At the bottom left, it says "1 matches". To the right are "Previous" and "Next" buttons.

Fazit

Was hier keinen Platz
mehr hatte...

Herzlichen Dank!