

# Theoretische Informatik, Praktikum

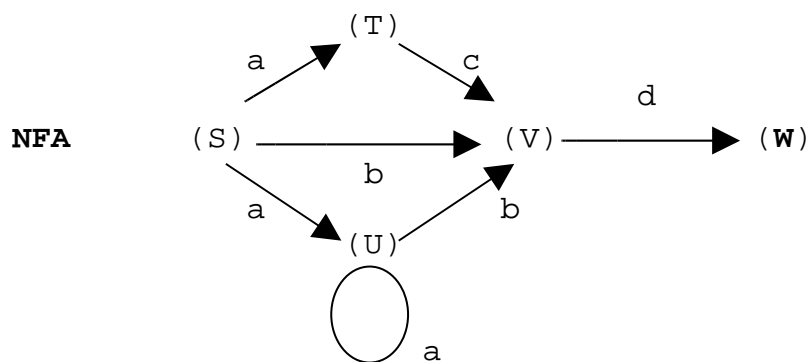
## Themen: Mustererkennung, Stringsuche

### Teil 1: Erkennen des Musters $(a^*b+ac)d$

Zur Erläuterung:

Der String `aaabd` passt zu dem obigen Muster,  
der String `aaacd` passt *nicht* zu diesem Muster.

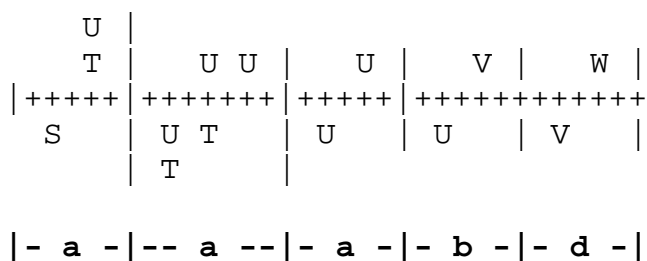
Bei der Mustererkennung ist der folgende (als Graph dargestellte) nichtdeterministische endliche Automat nützlich:



Wie man mittels dieses Automaten Muster erkennt, zeigt für den String `aaabd` diese Sequenz:

`(S)--a-->(TU)--a-->(U)--a-->(U)--b-->(V)--d-->(W)` akzeptiert

oder ausführlicher:



Die letzte Figur gibt Hinweise, wie man ein Programm zur Mustererkennung organisieren könnte. Es gibt eine zentrale Klasse `MATCHER` mit drei wesentlichen Features:

- ein Attribut vom Typ `MATRIX`, das den obigen Graphen in Form einer Matrix speichert
- ein argumentloser Konstruktor, der die Matrix konstruiert
- eine Funktion `is_matching(text: STRING): BOOLEAN`, die entscheidet, ob der String `text` zum Muster  $(a^*b+ac)d$  passt.

Bei der Funktion *is\_matching* hat man es im Verlauf der Iteration mit zwei unterschiedlichen Sorten von Zuständen zu tun: Bei Abarbeitung eines Zustandes werden die möglichen Nachfolgerzustände "auf die hohe Kante" gelegt. Diese unterschiedlichen Zustandssorten sollten sinnvollerweise getrennt verwaltet werden. In der Musterlösung

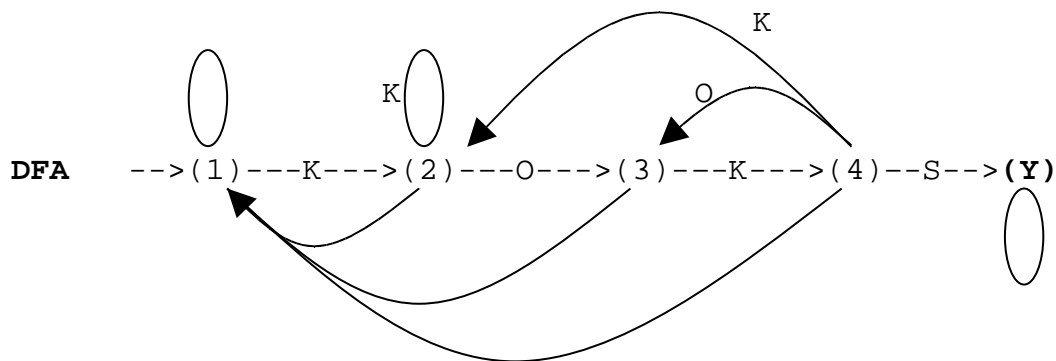
<http://www.fbi.fh-darmstadt.de/~meyer/skripte/ti/progs/>

werden dazu zwei Stack-Objekte benutzt.

**Hinweis:** Die automatische Konstruktion des nichtdeterministischen Automaten aus dem gegebenen Muster ist durchaus keine Kleinigkeit. Hinweise, wie man das machen könnte, finden sich im Skript (Teil 2). Die Implementierung der dort beschriebenen Transformation ist jedoch *nicht* Teil des Praktikums.

## Teil 2: Stringsuche

Mit dem folgenden deterministischen Automaten kann in Strings nach KOKS gesucht werden:



Wie man diesen Automaten benutzt, ist recht offensichtlich:  
 KOKKAKOKSSVX -> Endzustand **Y**, String akzeptiert (KOKS drin)  
 KOKOK -> Endzustand 4, String abgelehnt (KOKS nicht drin)

Schreiben Sie ein Programm, das (auf Basis des obigen Automaten) in eingelesenen Strings nach KOKS sucht und gegebenenfalls dessen Startposition ausgibt.

Wie Sie Ihr Programm organisieren, ist im Prinzip Ihnen überlassen. Bei der Musterlösung ist es folgendermaßen gemacht.

Es gibt eine zentrale Klasse SEARCHER mit drei hauptsächlichen Features:

- ein Konstruktor, dem der obige Graph (in Form einer Matrix)

übergeben wird

- ein Attribut vom Typ `MATRIX`, das die Matrix speichert
- eine Prozedur `search(text: STRING)`, die in `text` nach dem Suchwort (also KOKS) sucht

**Zusatz:** Erweitern Sie Ihr Programm, so daß nicht nur nach KOKS, sondern nach beliebigen Suchworten gesucht werden kann. Der geeignete Ort für diese Erweiterung ist der Konstruktor, dem dann nicht die fertige Matrix, sondern nur das Suchwort übergeben wird, und der die Matrix selber daraus generiert. Wie man das realisiert, ist nicht ganz offensichtlich, nähere Hinweise werden in der Vorlesung gegeben!