

---

**Compilerbau**

---

## Blatt 3

Abgabe: 7.11.2001

1. [5 Punkte] Geben Sie eine Funktion `regexp_accepts_empty` : 'a regexp -> bool an, so daß `regexp_accepts_empty r` genau dann `true` liefert, wenn  $\epsilon$  zur Sprache des regulären Ausdrucks  $r$  gehört.
2. [7 Punkte]
  - Beweisen Sie die Korrektheit der in der Vorlesung angegebenen Funktion `after_symbol`. Benutzen Sie dazu strukturelle Induktion über den `regexp`-Datentyp.
  - Funktioniert `matches` immer noch korrekt, wenn in `after_symbol` die Funktion `concat` durch den Konstruktor `Concat` sowie `alternate` durch `Alternate` ersetzt wird?
3. [18 Punkte] Geben Sie mit Hilfe des in der Vorlesung angegebenen Codes die Definition eines Scanners für Caml gemäß der Abschnitte „Lexical Conventions“ und „Names“ (dort nur bis einschl. *label-name*) in der Caml-Dokumentation an. Die Beschreibung soll Literale richtig interpretieren, also z.B. den Wert von Integer-Literalen berechnen und Escape-Sequenzen in Zeichenketten und Zeichen auflösen. Ignorieren Sie Fließkomma-Konstanten, Kommentare und Line-Number-Directives.

Anleitung:

- (a) Checken Sie aus dem CVS das Modul `compilerbau` aus. Die Anleitung dazu finden Sie auf der Homepage. Fügen Sie Ihre Definition von `accepts_empty` in der Datei `regexp.ml` anstelle des Platzhalters an. Im Verzeichnis `mini-caml` finden Sie im Modul `Camllex` den Datentyp `caml_token` für die Tokens von Caml und im Modul `Camlsyn` den Datentyp `syntax`, den Sie für die Attribute verwenden sollen.

- (b) Definieren Sie reguläre Ausdrücke für die verschiedenen Tokens. Geben Sie dazu die regulären Ausdrücke gemäß der Dokumentation an. Eine binäre Ziffer wird dabei z.B. durch

```
let bindigit = alternate (symbol '0') (symbol '1')
```

und ihr Präfix durch

```
let binprefix = alternate (concat (symbol '0') (symbol 'b'))  
                        (concat (symbol '0') (symbol 'B'))
```

repräsentiert.

Ergänzen Sie für Schlüsselwörter die folgende Assoziationsliste

```
let keyword_associations =  
  [ ("and", TAnd);  
    ("as", TAs);  
    ...]
```

und wandeln Sie `keyword_associations` in eine Liste aus regulären Ausdrücken und einer Aktion um. Die Aktion soll in diesem Fall `Camlsyn.Nil` als Attribut zurückgeben.

- (c) Schreiben Sie reguläre Ausdrücke für die Bestandteile von Strings (Normale Zeichen, Escape-Sequenzen, Dezimal-Kodierung) und fügen Sie diese mit entsprechenden Aktionen zu einem Scanner für Strings zusammen.
- (d) Definieren Sie den Scanner für Caml, indem Sie die regulären Ausdrücke mit entsprechenden Aktionen paaren und an `Lex.scan_one` übergeben.