

## 4. Übung „Übersetzerbau“

Bearbeitung bis zum 13. Mai 2008

---

### Aufgabe 12

Wir erweitern die Programmiersprache Simple um folgende Konstrukte:

$Stm$	$\longrightarrow$	<code>if <math>BExp</math> then <math>Stm</math> else <math>Stm</math></code>	(IfStm)
$Stm$	$\longrightarrow$	<code>while <math>BExp</math> do <math>Stm</math></code>	(WhileStm)
$Stm$	$\longrightarrow$	<code>nop</code>	(NopStm)
$BExp$	$\longrightarrow$	<code><math>Exp</math> BinCmp <math>Exp</math></code>	(CmpBExp)
$BExp$	$\longrightarrow$	<code>tt</code>	(TTBExp)
$BExp$	$\longrightarrow$	<code>ff</code>	(FFBExp)
$BExp$	$\longrightarrow$	<code><math>BExp</math> BinBOp <math>BExp</math></code>	(BooleExp)
$BinCmp$	$\longrightarrow$	<code>=</code>	(Equal)
$BinCmp$	$\longrightarrow$	<code>/=</code>	(NotEqual)
$BinCmp$	$\longrightarrow$	<code>&lt;=</code>	(LessEqual)
$BinCmp$	$\longrightarrow$	<code>&gt;=</code>	(GreaterEqual)
$BinBOp$	$\longrightarrow$	<code>&amp;&amp;</code>	(And)
$BinBOp$	$\longrightarrow$	<code>  </code>	(Or)

- Erweitern Sie die zugehörigen algebraischen Haskell-Datenstrukturen.
- Erweitern Sie den Interpreter aus Aufgabe 5 um die neuen Konstrukte. Für die Interpretation der `while`-Schleife reicht ein einfacher rekursiver Abstieg in der Baumstruktur nicht aus. Überlegen Sie, wie Sie die Schleife im Interpreter abwickeln können, so dass Sie die Schleifen iterieren.

### Aufgabe 13

In der Vorlesung wurde die  $LL(k)$ - und die starke  $LL(k)$  ( $SLL(k)$ )-Eigenschaft definiert. Zeigen Sie anhand folgender Grammatik, dass  $SLL(2) \neq LL(2)$  gilt:

$$S \rightarrow aAab \mid bAbb \qquad A \rightarrow a \mid \varepsilon$$

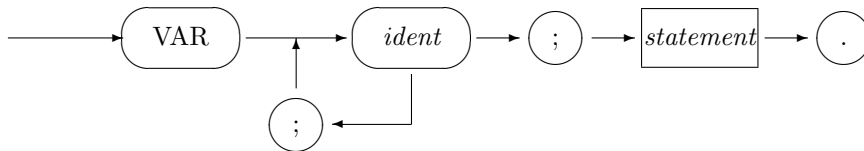
Bem.: Für  $k = 1$  stimmen beide Eigenschaften überein.

### Aufgabe 14

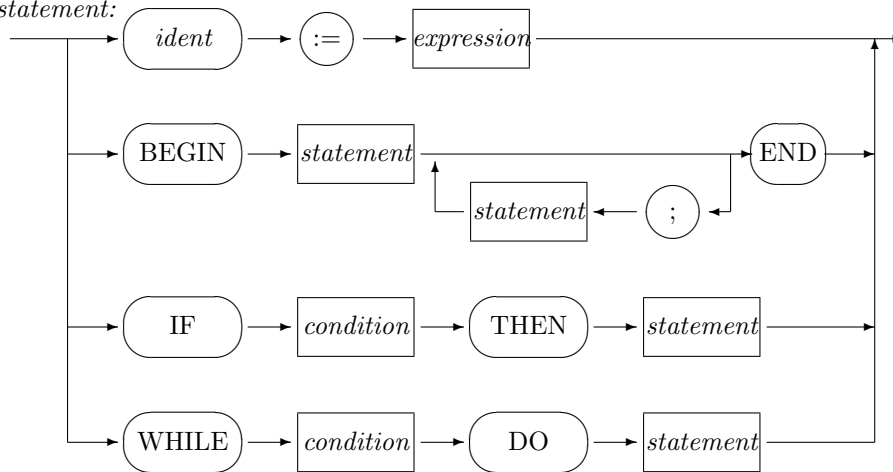
Die Syntax der Mini-Programmiersprache MPS sei durch die unten angegebenen Syntaxdiagramme definiert.

- Geben Sie die Syntax von MPS durch eine kontextfreie Grammatik  $G$  an.
- Berechnen Sie zu allen Nichtterminalsymbolen aus  $G$  die Mengen FIRST und FOLLOW.
- Prüfen Sie, ob es sich bei Ihrer Grammatik um eine LL(1)-Grammatik handelt. Wenn nicht, so geben Sie eine LL(1)-Grammatik  $G'$  für MPS an.
- Konstruieren Sie die zugehörige Parsing-Tabelle. Sollten Sie Aufgabenteil c) nicht gelöst haben, geben Sie dennoch eine (mehrdeutige) Parsingtabelle an.

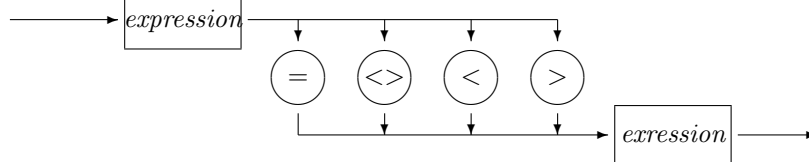
*program:*



*statement:*



*condition:*



*expression:*

