Konkrete Syntax

Eine **kontext-freie Grammatik** definiert die **Struktur der Programme** der Sprache. Hieraus wird der Parser generiert. Sie muß deshalb eindeutig sein (und die LALR(1)-Bedingung erfüllen).

Notation an Beispielen:

```
Prog     ::= Block .
Block    ::= '{' Decl* Stmt+ '}' .
Stmt     ::= Block / Assign ';' / Expr ';' .
Decl     ::= 'int' VarDecls ';' .
VarDecls::= VarDecl // ',' .
VarDecl  ::= Ident .
```

Benannte Nichtterminale:	`Prog Block Decl`
Benannte Terminale:	`Ident`
Literale Terminale:	`'{'  '}'  ';'`

EBNF-Konstrukte:

Folge:	`Decl*`
nicht-leere Folge:	`Stmt+`
nicht-leere Folge mit Trenner:	`VarDecl // ','`
Alternativen:	`Block / Assign ';'`
optional:	`[ Parameter ]`
Klammerung:	`( ':=' / ':-' )`

Hinweise zum Entwurf:

- Syntax einer vorgegebenen Sprache nicht unnötig ändern.
- Syntaktische Struktur soll der semantischen entsprechen.
- EBNF nur, wo keine Berechnungen im Baum erwartet werden.
- Namensanalyse erfordert syntaktische Unterscheidung von Bezeichnern: definierendes und angewandtes Auftreten.

Abstrakte Syntax

Eine **kontext-freie Grammatik** definiert **Strukturbäume** als Programmrepräsentation (**Baum-Grammatik**). Der Strukturbaum wird vom Parser aufgebaut.

Analyse und Transformation werden als **Berechnungen in** bestimmten **Strukturbaum-Kontexten** spezifiziert.

Konkrete Produktion:

```
Decl    ::= 'int' VarDecls ';' .
```

abstrakte Produktion, Baum-Kontext:

```
RULE: Decl ::= 'int' VarDecls ';' COMPUTE
      Berechnungen für diesen Kontext
END;
```

Die Knoten des Strukturbaumes repräsentieren Nichtterminale der Baum-Grammatik. Terminale stehen nicht im Baum. Literale Terminale dienen nur zur Identifikation der Produktion.

EBNF-Konstrukte gibt es in der Baum-Grammatik nur sehr eingeschränkt (`LISTOF`).

Es brauchen nur die Baum-Kontexte angegeben zu werden, zu denen es tatsächlich Berechnungen gibt. Eli-Werkzeuge vervollständigen die Baum-Grammatik automatisch.

Ebenso ergänzen sie Produktionen zur konkreten Syntax, die nur in der Baum-Grammatik stehen.

Der Aufbau des Strukturbaumes durch den Parser wird von Eli automatisch generiert.

Überflüssige Kettenproduktionen

Konkrete Produktionen mit gleicher Struktur und Bedeutung zu einem Baum-Kontext zusammenfassen.
Kettenproduktionen zwischen äquivalenten Symbolen entfallen.

Konkrete Ausdrucksgrammatik:

```
Expr    ::= Expr AddOpr Fact / Fact .
Fact    ::= Fact MulOpr Operand / Operand .
Operand ::= '(' Expr ')' .
Operand ::= Number .
AddOpr  ::= '+' / '-' .
MulOpr  ::= '*' / '/' .
```

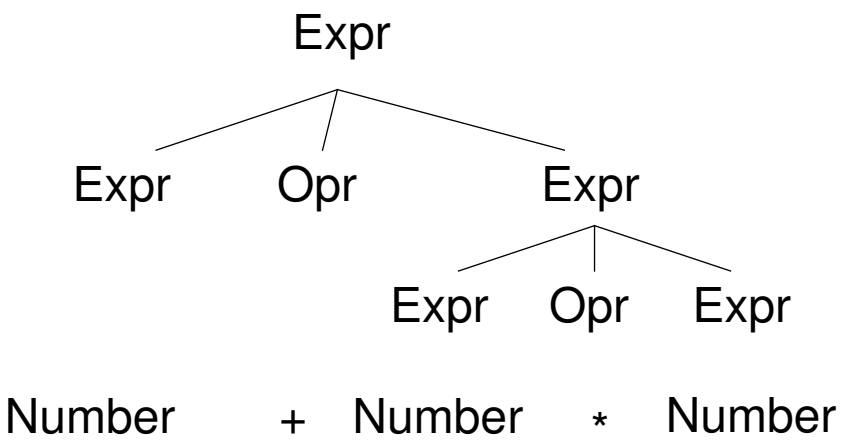
Äquivalente Symbole:

```
Expr ::= Fact Operand .
Opr  ::= AddOpr MulOpr .
```

Baum-Grammatik dazu:

```
RULE: Expr ::= Expr Opr Expr  END;
RULE: Expr ::= Number         END;
RULE: Opr  ::= '+'            END;
RULE: Opr  ::= '*'            END;
...
```

Strukturbaum dazu:



Ein Symbolknoten gehört zu zwei benachbarten Kontexten: oberer und unterer Kontext.