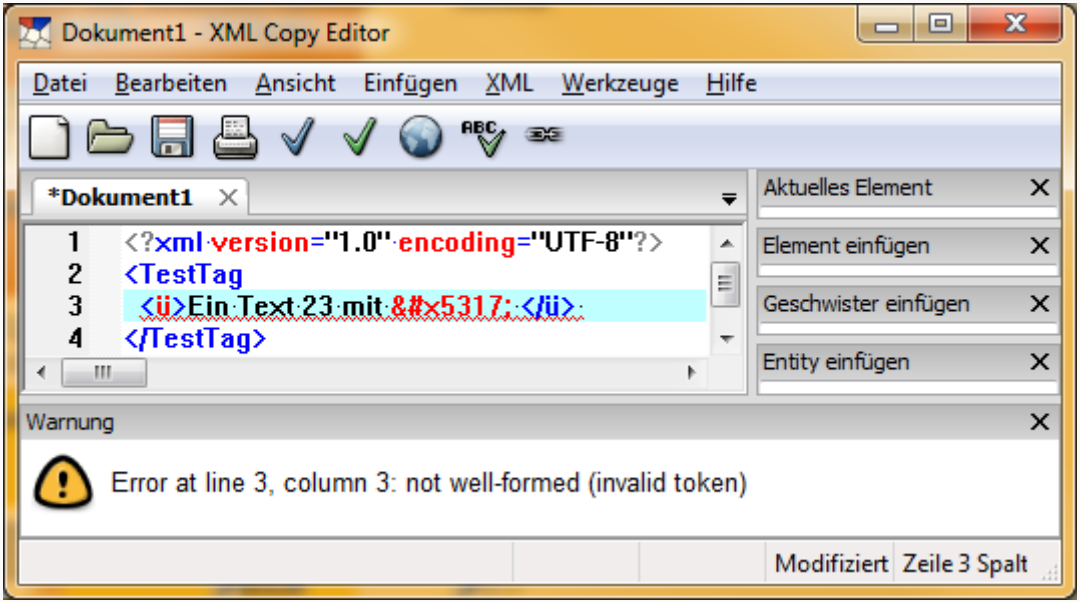
Error-Recovery

Carsten Gips (FH Bielefeld)

Fehler beim Parsen



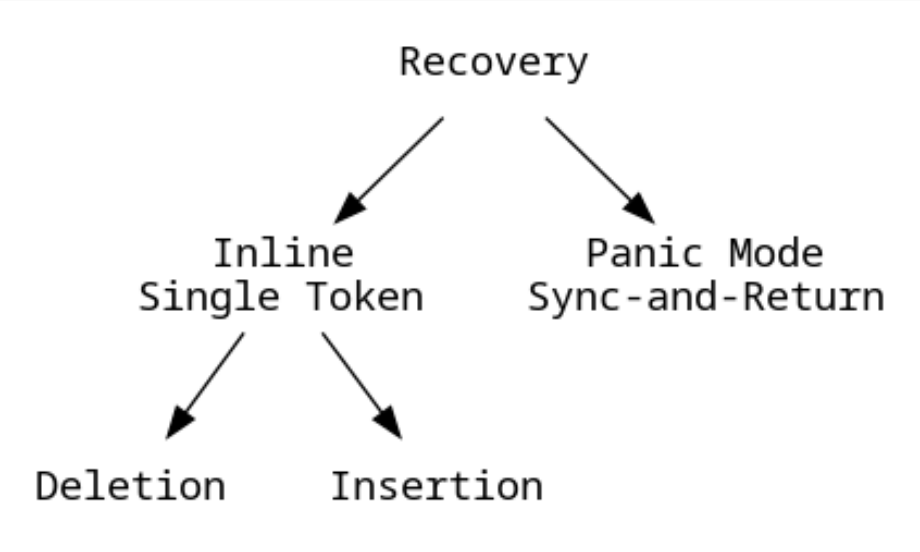
Typische Fehler beim Parsing

```
grammar VarDef;

alt   : stmt | stmt2 ;
stmt  : 'int' ID ';' ;
stmt2 : 'int' ID '=' ID ';'  ;
```

ANTLR4: VarDef.g4, Beispiele: VarDef.txt

Überblick Recovery bei Parser-Fehlern



Skizze: Generierte Parser-Regeln (ANTLR)

```
stmt  : 'int' ID ';' ;
```

```
def stmt():
    try: match("int"); match(ID); match(";")
    catch (RecognitionException re):
        _errHandler.reportError(self)              # let's report it
        _errHandler.recover(self)                  # Panic-Mode

def match(x):
    if lookahead == x: consume()
    else: _errHandler.recoverInline(self)          # Inline-Mode
}
```

Inline-Recovery bei Token-Mismatch (Skizze)

```
def recoverInline(parser):
    # SINGLE TOKEN DELETION
    if singleTokenDeletion(parser):
        return getMatchedSymbol(parser)

    # SINGLE TOKEN INSERTION
    if singleTokenInsertion(parser):
        return getMissingSymbol(parser)

    # that didn't work, throw a new exception
    throw new InputMismatchException(parser)
}
```

Panic Mode: Sync-and-Return (Skizze)

```
def rule():
    try: ... rule-body ...
    catch (RecognitionException re):
        _errHandler.reportError(self)        # let's report it
        _errHandler.recover(self)            # Panic-Mode
}
```

=> Entferne solange Token, bis aktuelles Token im "*Resynchronization Set*"

ANTLR: Einsatz des "*Resynchronization Set*"

```
stmt : 'if' expr ':' stmt           // Following Set für "expr": {':'}
     | 'while' '(' expr ')' stmt ;  // Following Set für "expr": {')'}
expr : term '+' INT ;               // Following Set für "term": {'+'}
```

- Eingabe: `if :`
- Aufruf-Stack nach Bearbeitung von `if`: `[stmt, expr, term]`
- **Resynchronization Set**: `{'+', ':'}`

Hinweis: *FOLLOW* $\neq$ *Following*

Fehlerproduktionen

```
stmt : 'int' ID ';'
     : 'int' ID             {notifyErrorListeners("Missing ';'");}
     : 'int' ID ';' ';'     {notifyErrorListeners("Too many ';'");}
     ;
```

::: notes ## Anmerkung: Nicht eindeutige Grammatiken

```
stat: expr ';' | ID '+' ID ';' ;
expr: ID '+' ID | INT ;
```

=> Was passiert bei der Eingabe: `a+b` ??! Welche Regel/Alternative soll jetzt matchen, d.h. welcher AST soll am Ende erzeugt werden?!

ANTLR4

Nicht eindeutige Grammatiken führen **nicht** zu einer Fehlermeldung, da nicht der Nutzer mit seiner Eingabe Schuld ist, sondern das Problem in der Grammatik selbst steckt.

Während des Debuggings von Grammatiken lohnt es sich aber, diese Warnungen zu aktivieren. Dies kann entweder mit der Option "`-diagnostics`" beim Aufruf des `grun`-Tools geschehen oder über das Setzen

Wrap-Up

- Fehler bei `match()`: *single token deletion* oder *single token insertion*
- Panic Mode: *sync-and-return* bis Token in *Resynchronization Set* (ANTLR4) oder `error`-Token shiftbar (Bison)
  - ANTLR4: Sonderbehandlung bei Start von Sub-Regeln und in Schleifen
  - ANTLR4: Fail-Save zur Vermeidung von Endlosschleifen
- Fehler-Alternativen in Grammatik einbauen

LICENSE