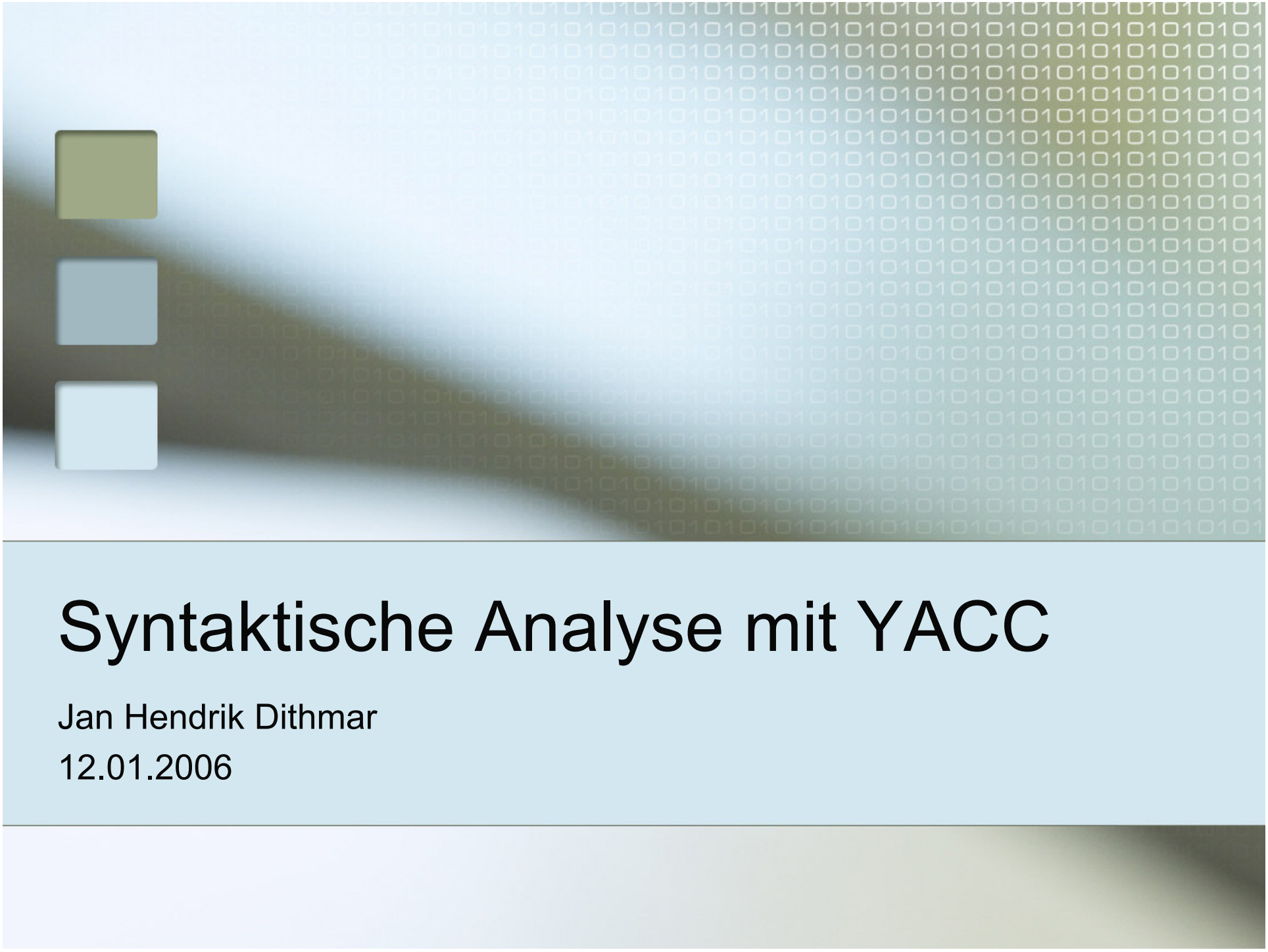


Syntaktische Analyse mit YACC

Jan Hendrik Dithmar

12.01.2006

Motivation

- Datenstrukturen einlesen
  - Erkennen von Baumstrukturen beim Einlesen aus einer Datei.
  - Wiederherstellen von Daten mit Hilfe der erkannten Struktur.
- Symbolische Ausdrücke erkennen
  - Verarbeiten von symbolischen Ausdrücken wie z.B. Terme
- Konfigurationsdateien bearbeiten

Beispiel

```
%{
#include <stdio.h>
#define YYSTYPE double
%}

%token ZAHL
%start eingabe

%%
eingabe         :       /* leere Eingabe */
                |       eingabe zeile
                ;

zeile           :       '\n'            /* Zeilenende */
                |       ausdruck '\n'   {printf("%g\n", $1);}
                ;

ausdruck        :       summe           { $$ = $1; }
                ;
```

Beispiel (Fortsetzung)

```
summe   :       summe '+' term          { $$ = $1 + $3; }
        |       summe '-' term          { $$ = $1 - $3; }
        |       term                    { $$ = $1; }
        ;

term    :       term '*' faktor         { $$ = $1 * $3; }
        |       term '/' faktor         { $$ = $1 / $3; }
        |       faktor                  { $$ = $1; }
        ;

faktor  :       ZAHL                    { $$ = $1; }
        |       '(' ausdruck ')'        { $$ = $2; }

%%
```

Beispiel

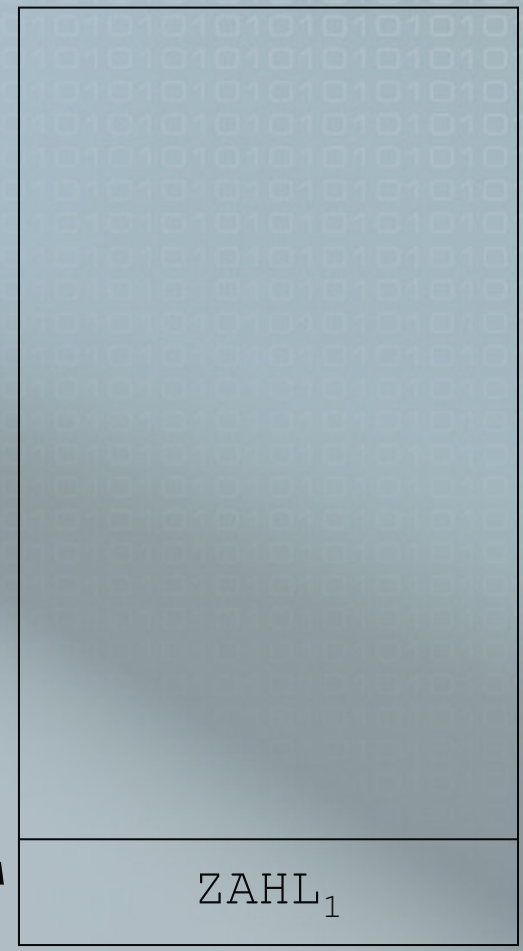
Eingabe:	Aktion:
`1+2*3\n`	

Stack:

Beispiel

Eingabe:	Aktion:
`1+2*3\n`	S

Stack:



Beispiel

Eingabe:	Aktion:
1+2*3\n	$R_{\texttt{faktor : ZAHL}}$

Reduzieren mit:

```
faktor : ZAHL
```

Semantische Aktion:

```
{ $$ = $1; }
```

Stack:

Beispiel

Eingabe:	Aktion:
1+2*3\n	$R_{\texttt{term : faktor}}$

Stack:

Reduzieren mit:

```
term : faktor
```

Semantische Aktion:

```
{ $$ = $1; }
```

Beispiel

Eingabe:	Aktion:
1+2*3\n	$R_{\texttt{summe : term}}$

Stack:

Reduzieren mit:

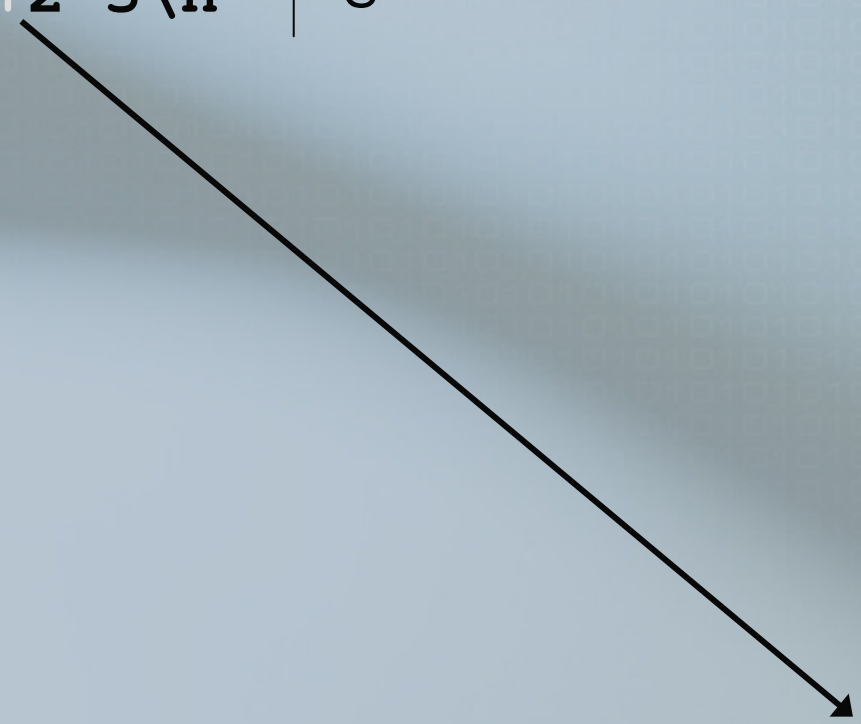
```
summe : term
```

Semantische Aktion:

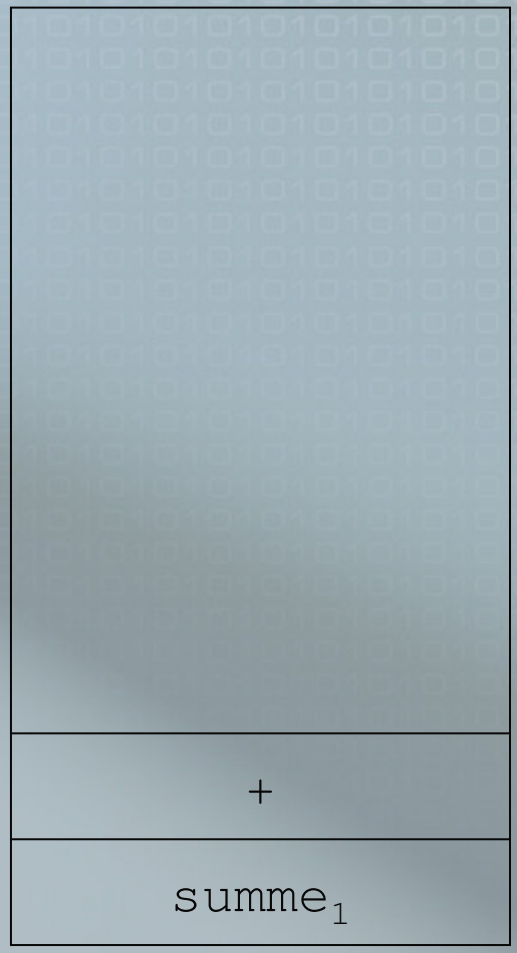
```
{ $$ = $1; }
```

Beispiel

Eingabe:	Aktion:
`1+2*3\n`	S



Stack:



Beispiel

Eingabe:	Aktion:
1+2*3\n	S

Beispiel

Eingabe:	Aktion:
1+2*3\n	$R_{\texttt{faktor : ZAHL}}$

Reduzieren mit:

```
faktor : ZAHL
```

Semantische Aktion:

```
{ $$ = $1; }
```

Stack:

Beispiel

Eingabe:	Aktion:
1+2*3\n	$R_{\texttt{term : faktor}}$

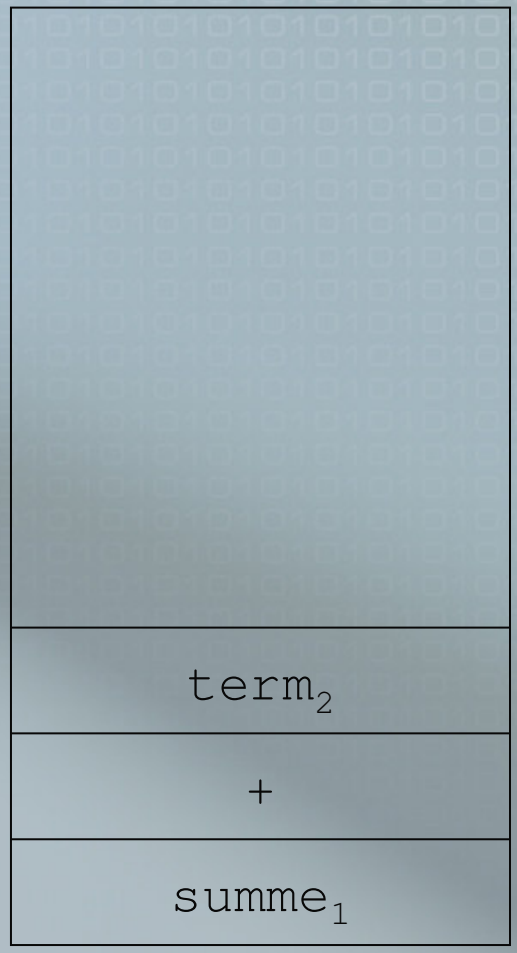
Reduzieren mit:

```
term : faktor
```

Semantische Aktion:

```
{ $$ = $1; }
```

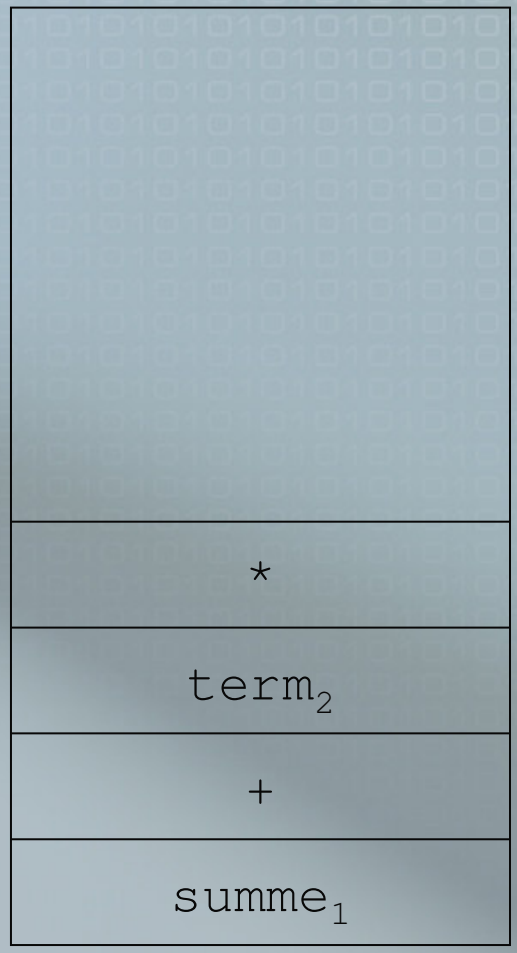
Stack:



Beispiel

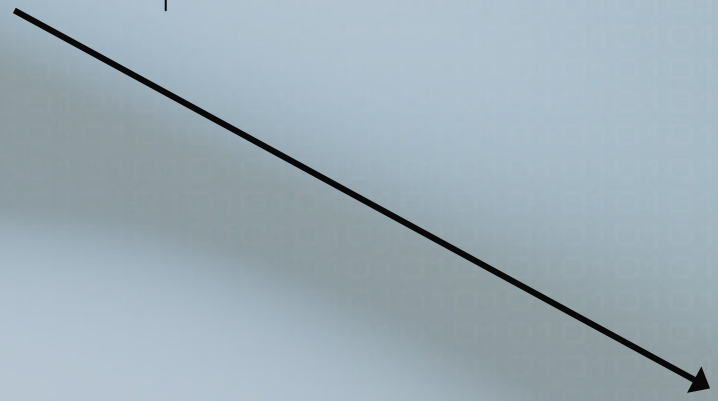
Eingabe:	Aktion:
`1+2*3\n`	S

Stack:

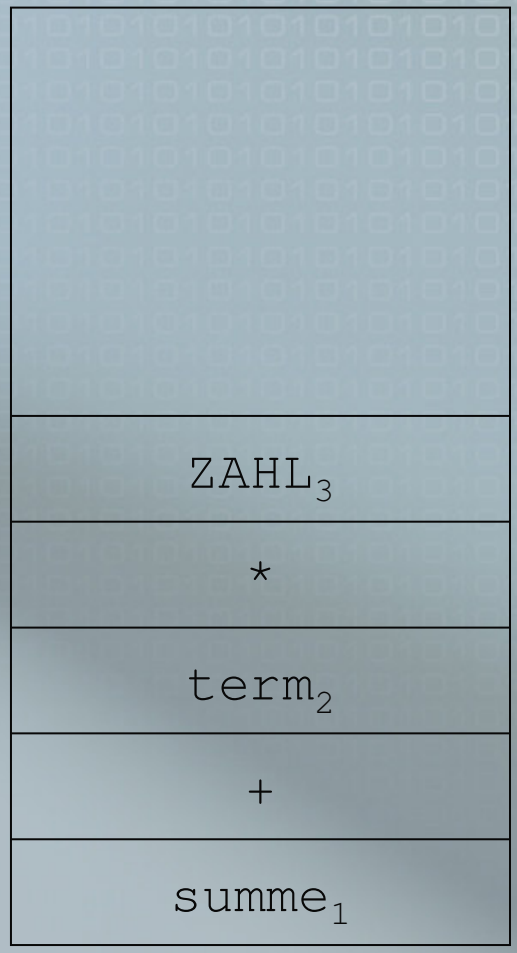


Beispiel

Eingabe:	Aktion:
1+2*3\n	S



Stack:



Beispiel

Eingabe:	Aktion:
`1+2*3\n`	$R_{\texttt{faktor : ZAHL}}$

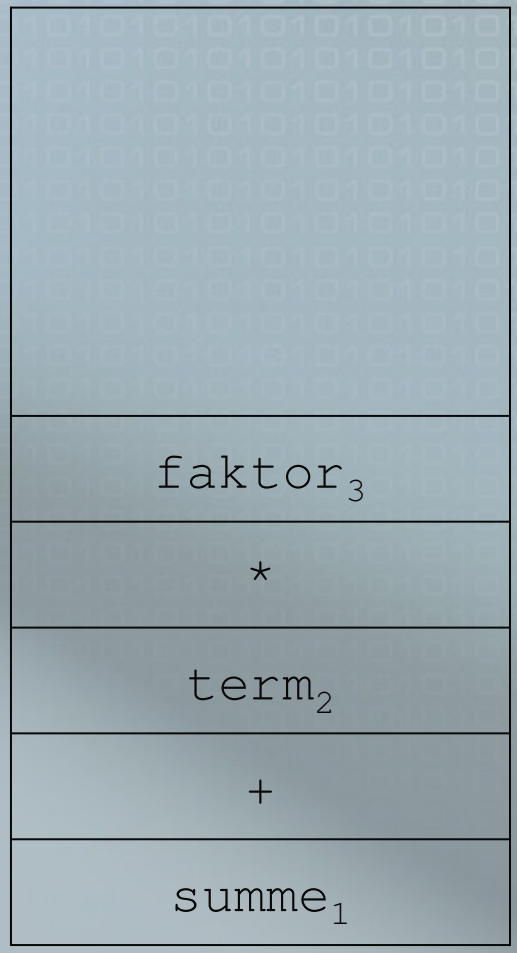
Reduzieren mit:

```
faktor : ZAHL
```

Semantische Aktion:

```
{ $$ = $1; }
```

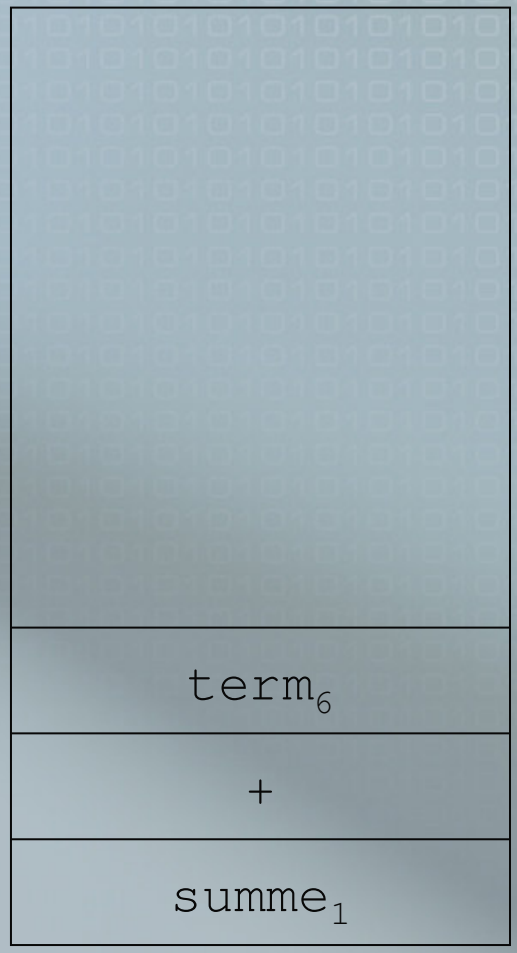
Stack:



Beispiel

Eingabe:	Aktion:
`1+2*3\n`	$R_{\texttt{term : term '*' faktor}}$

Stack:



Reduzieren mit:

```
term : term '*' faktor
```

Semantische Aktion:

```
{ $$ = $1 * $3; }
```

Beispiel

Eingabe:	Aktion:
`1+2*3\n`	$R_{\texttt{summe : summe '+' term}}$

Stack:

Reduzieren mit:

```
summe : summe '+' term
```

Semantische Aktion:

```
{ $$ = $1 + $3; }
```

Beispiel

Eingabe:	Aktion:
`1+2*3\n`	$R_{\texttt{ausdruck : summe}}$

Stack:

Reduzieren mit:

```
ausdruck : summe
```

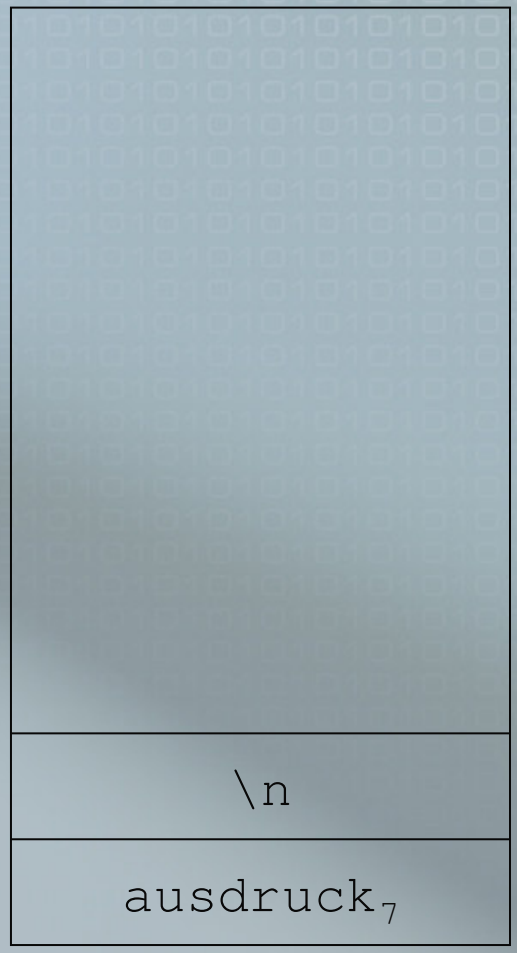
Semantische Aktion:

```
{ $$ = $1; }
```

Beispiel

Eingabe:	Aktion:
`1+2*3\n`	S

Stack:



Beispiel

Eingabe:	Aktion:
`1+2*3\n`	

Stack: