TLA^{+}:

- ist eine Spezifikationssprache
- basiert auf der temporalen Logik TLA (Temporal Logik of Actions)
- ermöglicht eine modulare Schreibweise von Spezifikationen
- ermöglicht strukturierte formale Beweise
- kann mit Hilfe des Pakets `tla` mit LaTeX gesetzt werden

module *Die Struktur eines Moduls*

parameters

Deklaration der Variablen und Konstanten

import
include

Einbindung anderer Module

predicates

Initialisierung der Variablen,
ggf. Belegung von Konstanten

definitions

Definition von Mengen, Funktionen etc.

actions

Definition der Operationen

temporal

die eigentliche Spezifikation

module *Der Rahmen*

```
\begin{module}{Der Rahmen}
\midbar
\end{module}
```

module *Deklarationen*

parameters

$Const_1, \ldots, Const_N$: constant
$var_1, \ldots, var_N$: variable

```
\begin{module}{Deklarationen}
  \begin{decls}{parameters}
    Const_1, \dots, Const_N : \CONSTANT \\
    var_1, \dots, var_N : \VARIABLE \\
  \end{decls}
\end{module}
```

module *Einbindung anderer Module*

$$
\begin{array}{rl}
\textbf{import} & \mathit{Mod}_1, \ldots, \mathit{Mod}_N \\
\textbf{include} & \mathit{ModName}\ \textbf{as}\ \mathit{ModAlias} \\
\quad\textbf{with} & \mathit{varName}_1 \leftarrow \mathit{varAlias}_1 \\
 & \ldots \\
 & \mathit{varName}_N \leftarrow \mathit{varAlias}_N \\
\ldots &
\end{array}
$$

```
\begin{module}{Einbindung anderer Module}
  \begin{decls}{}
    \IMPORT\ Mod_1, \dots, Mod_N \\
    \INCLUDE\ ModName \AS\ ModAlias \\
      \quad \WITH\ \begin{noj}
                     varName_1 <- varAlias_1 \\
                     \dots \\
                     varName_N <- varAlias_N \\
                   \end{noj} \\
    \dots
  \end{decls}
\end{module}
```

module *Initialisierung*

predicates

$$Init \triangleq \begin{array}{l} \land\ Const_1 = ConstValue_1 \\ \land\ \ldots \\ \land\ Const_N = ConstValue_N \\ \land\ var_1 = InitValue_1 \\ \land\ \ldots \\ \land\ var_N = InitValue_N \end{array}$$

```
\begin{module}{Initialisierung}
  \begin{decls}{predicates}
    Init == \begin{conj}
              Const_1 = ConstValue_1 \\
              \dots \\
              Const_N = ConstValue_N \\
              var_1 = InitValue_1 \\
              \dots \\
              var_N = InitValue_N \\
            \end{conj} \\
  \end{decls}
\end{module}
```

module *Definitionen*

definitions

$$Vars \triangleq \langle var_1, \ldots, var_N \rangle$$

$$\ldots$$

```
\begin{module}{Definitionen}
  \begin{decls}{definitions}
    Vars == << var_1, \dots, var_N >> \\
    \dots \\
  \end{decls}
\end{module}
```

module *Operationen*

actions

$$
\begin{aligned}
Act_1 &\triangleq \land\ PreConditions_1 \\
&\ \land\ PostConditions_1 \\
&\ \land\ \textsf{unchanged}\ unchangedVars_1 \\
&\dots \\
Act_N &\triangleq \land\ PreConditions_N \\
&\ \land\ PostConditions_N \\
&\ \land\ \textsf{unchanged}\ unchangedVars_N
\end{aligned}
$$

```
\begin{module}{Operationen}
  \begin{decls}{actions}
    Act_1 == \begin{conj}
                 PreConditions_1 \\
                 PostConditions_1 \\
                 \UNCHANGED unchangedVars_1 \\
               \end{conj} \\
    \dots \\
    Act_N == \begin{conj}
                 PreConditions_N \\
                 PostConditions_N \\
                 \UNCHANGED unchangedVars_N \\
               \end{conj} \\
  \end{decls}
\end{module}
```

module *Die eigentliche Spezifikation*

temporal

$$
\begin{aligned}
SomeAction &\triangleq \lor\ Act_1 \\
&\quad\ \lor\ \ldots \\
&\quad\ \lor\ Act_N \\
Spec &\triangleq \land\ Init \\
&\quad\ \land\ \Box[SomeAction]_{Vars}
\end{aligned}
$$

```
\begin{module}{Die eigentliche Spezifikation}
  \begin{decls}{temporal}
    SomeAction == \begin{disj}
                    Act_1 \\
                    \dots \\
                    Act_N \\
                  \end{disj} \\
    Spec == \begin{conj}
              Init \\
              [][SomeAction]_{Vars} \\
            \end{conj}
  \end{decls}
\end{module}
```

Abbrevs in TLA$^+$:

$\triangleq$	`==`	equals by definition (`\deq`)
$\Rightarrow$	`=>`	`\Rightarrow`
$\setminus$	`:\:`	`\backslash`
$\Box$	`[]`	`\Box`
$\Diamond$	`<>`	`\Diamond`
$\leadsto$	`~>`	`\leadsto`
$\lnot$	`~`	`\lnot`
$\longleftarrow$	`<--`	`\longleftarrow`
$\leftarrow$	`<-`	`\leftarrow`
$\langle$	`<<`	`\langle`
$\rangle$	`>>`	`\rangle`
$\longrightarrow$	`-->`	`\longrightarrow`
$\rightarrow$	`->`	`\rightarrow`
$\vdash$	`|-`	`\vdash`
$\models$	`|=`	`\models`
$\mapsto$	`|->`	`\mapsto`
$\neq$	`#`	`\neq`
$\lor$	`\/`	`\lor`
$\land$	`/\␣`	`\land`
	`/\%`	...at the end of a line
“text”	`"text"`	`{\rm ‘‘{\sf text}’’}` (a TLA$^+$ string constant)
	`_.`	`\,` (a small space)
	`__`	Produces a ‘unit’ space
	`___`	Produces two ‘unit’ spaces
	`__..._`	Produces n ‘unit’ spaces

Aussagen- und Prädikatenlogik:

`\TRUE`	true	
`\FALSE`	false	
`/\␣` oder `/\%`	$\wedge$	
`\/`	$\vee$	
`\neg`	$\neg$	
`=>`	$\Rightarrow$	
`\equiv`	$\equiv$	Äquivalenz ($\Longleftrightarrow$)
`\A x : p`	$\forall x : p$	
`\E x : p`	$\exists x : p$	
`\A x \in S : p`	$\forall x \in S : p$	
`\E x \in S : p`	$\exists x \in S : p$	

Temporale Logik:

`\CHOOSE x : p`	choose $x : p$	equals some x satisfying p
`p'`	p'	p is true in final state of step
`[A]_e`	$[A]_e$	$A \vee (e' = e)$
`\UNCHANGED e`	unchanged e	$e' = e$
`[]F`	$\Box F$	F is always true

Schreibweise von Mengen:

`\{e_1,\dots,e_n\}`	$\{e_1, \ldots, e_n\}$
`\{x \in S : p\}`	$\{x \in S \,:\, p\}$
`\{e : x \in S\}`	$\{e \,:\, x \in S\}$

... Tupeln:

`<<e_1,\dots,e_n>>`	$\langle\, e_1, \ldots, e_n \,\rangle$	n-Tupel
`e[i]`	$e[i]$	i-te Komponente
`S_1 \X\dots\X S_n`	$S_1 \times \ldots \times S_n$	Produktmenge

... Funktionen:

<table>
<tr><td><code>f[e]</code></td><td>$f[e]$</td><td>Funktions-Anwendung</td></tr>
<tr><td><code>\DOMAIN f</code></td><td>Domain f</td><td>Definitionsmenge</td></tr>
<tr><td><code>[x \in S |-> e]</code></td><td>$[x \in S \mapsto e]$</td><td>Funktions-Definition</td></tr>
<tr><td><code>[S -> T]</code></td><td>$[S \rightarrow T]$</td><td>Menge von Funktionen</td></tr>
</table>

... und Records:

`e.h`	$e.h$	Record-Komponente

$[h_1 \mapsto e_1, \ldots, h_n \mapsto e_n]$ Record-Definition

`[h_1 |-> e_1, \dots, h_n |-> e_n]`

$\{[h_1 : S_1, \ldots, h_n : S_n]\}$ Menge von Records

`{[h_1 : S_1, \dots, h_n : S_n]}`

Bedingte Ausdrücke:

if p **then** e_1
else e_2
	`\IF p`
	`  \THEN e_1`
	`  \ELSE e_2`
	`\FI`

case $p_1 \rightarrow e_1$
$\cdots$
$p_n \rightarrow e_n$
	`\CASE \begin{noj}`
	`        p_1 -> e_1 \\`
	`        \dots \\`
	`        p_n -> e_n \\`
	`      \end{noj}`

Lokale Definition von x_1**, . . . ,** x_n**:**

let $x_1 \triangleq e_1$
$\cdots$
$x_n \triangleq e_n$
in e
	`\LET x_1 == e_1 \\`
	`  \dots \\`
	`  x_n == e_n`
	`\IN e \NI`