TLA2B: A new validation tool for TLA$^+$

Dominik Hansen & Michael Leuschel

Institut für Informatik
Heinrich-Heine-Universität Düsseldorf

27.08.2012

HEINRICH HEINE
UNIVERSITÄT DÜSSELDORF

Overview

1. Approach
2. Translation to B
3. Experiments
4. Demo
5. Conclusion

HEINRICH HEINE
UNIVERSITÄT DÜSSELDORF

Approach & Motivation

Approach

Translating the non-temporal part of TLA$^+$ to B

$\Rightarrow$ B does not support temporal formulas

Motivation

- Validating TLA$^+$ specification with existing B tools
- in particular: using the model checker ProB

HEINRICH HEINE UNIVERSITÄT DÜSSELDORF

Approach & Motivation

Approach

Translating the non-temporal part of TLA$^+$ to B

$\Rightarrow$ B does not support temporal formulas

Motivation

- Validating TLA$^+$ specification with existing B tools
- in particular: using the model checker PROB

HEINRICH HEINE
UNIVERSITÄT DÜSSELDORF

Approach & Motivation

Approach

Translating the non-temporal part of TLA$^+$ to B

$\Rightarrow$ B does not support temporal formulas

Motivation

- Validating TLA$^+$ specification with existing B tools
- in particular: using the model checker PROB

HEINRICH HEINE
UNIVERSITÄT DÜSSELDORF

TLA$^+$ & B-Method

	TLA$^+$	B-Method
Invented by	Leslie Lamport	J.R. Abrial
State-based	$\surd$	$\surd$
Set theory	$\surd$	$\surd$
Predicate logic	$\surd$	$\surd$
Arithmetic	$\surd$	$\surd$
Temporal formulas	$\surd$	X
State transition	Before-after predicate	Generalised substitutions
Model checker	TLC	ProB
Prove support	TLAPS	AtelierB

MyHourClock

MODULE *MyHourClock*

EXTENDS *Naturals*

CONSTANTS *start*

VARIABLES *hr*

ASSUME $start \in 1 \,..\, 12$

$$Inv \triangleq hr \in 1 \,..\, 12$$
$$Init \triangleq hr = start$$
$$Inc \triangleq hr < 12 \land hr' = hr + 1$$
$$Reset \triangleq hr = 12 \land hr' = 1$$
$$Next \triangleq Inc \lor Reset$$

HEINRICH HEINE UNIVERSITÄT DÜSSELDORF

MyHourClock

MODULE *MyHourClock*

EXTENDS *Naturals*

CONSTANTS *start*

VARIABLES *hr*

ASSUME $start \in 1 \,..\, 12$

$$Inv \triangleq hr \in 1 \,..\, 12$$
$$Init \triangleq hr = start$$
$$Inc \triangleq hr < 12 \land hr' = hr + 1$$
$$Reset \triangleq hr = 12 \land hr' = 1$$
$$Next \triangleq Inc \lor Reset$$

Config file

INIT Init
NEXT Next
INVARIANT Inv

HEINRICH HEINE
UNIVERSITÄT DÜSSELDORF

MyHourClock

MODULE *MyHourClock*

EXTENDS *Naturals*

CONSTANTS *start*

VARIABLES *hr*

ASSUME $start \in 1 \,..\, 12$

$Inv \triangleq hr \in 1 \,..\, 12$

$Init \triangleq hr = start$

$Inc \triangleq hr < 12 \land hr' = hr + 1$

$Reset \triangleq hr = 12 \land hr' = 1$

$Next \triangleq Inc \lor Reset$

Config file

INIT Init
NEXT Next
INVARIANT Inv

HEINRICH HEINE UNIVERSITÄT DÜSSELDORF

MyHourClock

MODULE *MyHourClock*

EXTENDS *Naturals*

CONSTANTS *start*

VARIABLES *hr*

ASSUME $start \in 1 \, .. \, 12$

$$Inv \triangleq hr \in 1 \, .. \, 12$$

$$Init \triangleq hr = start$$

$$Inc \triangleq hr < 12 \land hr' = hr + 1$$

$$Reset \triangleq hr = 12 \land hr' = 1$$

$$Next \triangleq Inc \lor Reset$$

Config file

INIT Init
NEXT Next
INVARIANT Inv

HEINRICH HEINE UNIVERSITÄT DÜSSELDORF

MyHourClock

MODULE *MyHourClock*

EXTENDS *Naturals*

CONSTANTS *start*

VARIABLES *hr*

ASSUME $start \in 1 \,..\, 12$

$$Inv \triangleq hr \in 1 \,..\, 12$$
$$Init \triangleq hr = start$$
$$Inc \triangleq hr < 12 \land hr' = hr + 1$$
$$Reset \triangleq hr = 12 \land hr' = 1$$
$$Next \triangleq Inc \lor Reset$$

Config file

INIT Init

NEXT Next

INVARIANT Inv

HEINRICH HEINE UNIVERSITÄT DÜSSELDORF

Translation of the example

$$
\begin{array}{l}
\text{MODULE } \mathit{MyHourClock} \\
\text{EXTENDS } \mathit{Naturals} \\
\text{CONSTANTS } \mathit{start} \\
\text{VARIABLES } \mathit{hr} \\
\text{ASSUME } \mathit{start} \in 1 \,..\, 12 \\
\hline
\mathit{Inv} \triangleq \mathit{hr} \in 1 \,..\, 12 \\
\mathit{Init} \triangleq \mathit{hr} = \mathit{start} \\
\mathit{Inc} \triangleq \mathit{hr} < 12 \land \mathit{hr}' = \mathit{hr} + 1 \\
\mathit{Reset} \triangleq \mathit{hr} = 12 \land \mathit{hr}' = 1 \\
\mathit{Next} \triangleq \mathit{Inc} \lor \mathit{Reset}
\end{array}
$$

Translation of the example

MODULE *MyHourClock*

EXTENDS *Naturals*
CONSTANTS *start*
VARIABLES *hr*
ASSUME $start \in 1 \,..\, 12$

$$
\begin{aligned}
Inv &\triangleq hr \in 1 \,..\, 12 \\
Init &\triangleq hr = start \\
Inc &\triangleq hr < 12 \land hr' = hr + 1 \\
Reset &\triangleq hr = 12 \land hr' = 1 \\
Next &\triangleq Inc \lor Reset
\end{aligned}
$$

```
MACHINE MyHourClock

END
```

Translation of the example

MODULE *MyHourClock*

EXTENDS *Naturals*

CONSTANTS *start*

VARIABLES *hr*

ASSUME $start \in 1 \,..\, 12$

$$Inv \triangleq hr \in 1 \,..\, 12$$
$$Init \triangleq hr = start$$
$$Inc \triangleq hr < 12 \land hr' = hr + 1$$
$$Reset \triangleq hr = 12 \land hr' = 1$$
$$Next \triangleq Inc \lor Reset$$

MACHINE *MyHourClock*

END

HEINRICH HEINE UNIVERSITÄT DÜSSELDORF

Translation of the example

```
—— MODULE MyHourClock ——
EXTENDS Naturals
CONSTANTS start
VARIABLES hr
ASSUME start ∈ 1 .. 12
```

$Inv \triangleq hr \in 1 \,..\, 12$

$Init \triangleq hr = start$

$Inc \triangleq hr < 12 \land hr' = hr + 1$

$Reset \triangleq hr = 12 \land hr' = 1$

$Next \triangleq Inc \lor Reset$

```
MACHINE MyHourClock
CONSTANTS start

END
```

HEINRICH HEINE UNIVERSITÄT DÜSSELDORF

Translation of the example

MODULE *MyHourClock*
EXTENDS *Naturals*
CONSTANTS *start*
VARIABLES *hr*
ASSUME $start \in 1\,..\,12$

$$
\begin{aligned}
Inv &\triangleq hr \in 1\,..\,12\\
Init &\triangleq hr = start\\
Inc &\triangleq hr < 12 \land hr' = hr + 1\\
Reset &\triangleq hr = 12 \land hr' = 1\\
Next &\triangleq Inc \lor Reset
\end{aligned}
$$

MACHINE *MyHourClock*
CONSTANTS *start*
VARIABLES *hr*

END

HEINRICH HEINE
UNIVERSITÄT DÜSSELDORF

Translation of the example

MODULE *MyHourClock*

EXTENDS *Naturals*
CONSTANTS *start*
VARIABLES *hr*
ASSUME $start \in 1 \,..\, 12$

$Inv \triangleq hr \in 1 \,..\, 12$
$Init \triangleq hr = start$
$Inc \triangleq hr < 12 \land hr' = hr + 1$
$Reset \triangleq hr = 12 \land hr' = 1$
$Next \triangleq Inc \lor Reset$

MACHINE *MyHourClock*
CONSTANTS *start*
VARIABLES *hr*
PROPERTIES $start \in 1 \,..\, 12$

END

HEINRICH HEINE UNIVERSITÄT DÜSSELDORF

Translation of the example

```
---------------- MODULE MyHourClock ----------------
EXTENDS Naturals
CONSTANTS start
VARIABLES hr
ASSUME start ∈ 1 .. 12
-----------------------------------------------------
Inv   ≜ hr ∈ 1 .. 12
Init  ≜ hr = start
Inc   ≜ hr < 12 ∧ hr' = hr + 1
Reset ≜ hr = 12 ∧ hr' = 1
Next  ≜ Inc ∨ Reset
=====================================================
```

```
MACHINE MyHourClock
CONSTANTS start
VARIABLES hr
PROPERTIES start ∈ 1 .. 12
INVARIANT hr ∈ 1 .. 12



END
```

HEINRICH HEINE
UNIVERSITÄT DÜSSELDORF

Translation of the example

MODULE *MyHourClock*

EXTENDS *Naturals*

CONSTANTS *start*

VARIABLES *hr*

ASSUME $start \in 1\,..\,12$

$Inv \triangleq hr \in 1\,..\,12$

$Init \triangleq hr = start$

$Inc \triangleq hr < 12 \land hr' = hr + 1$

$Reset \triangleq hr = 12 \land hr' = 1$

$Next \triangleq Inc \lor Reset$

```
MACHINE MyHourClock
CONSTANTS start
VARIABLES hr
PROPERTIES start ∈ 1 .. 12
INVARIANT hr ∈ 1 .. 12
INITIALISATION hr : (hr = start)

END
```

HEINRICH HEINE
UNIVERSITÄT DÜSSELDORF

Translation of the example

MODULE *MyHourClock*

EXTENDS *Naturals*

CONSTANTS *start*

VARIABLES *hr*

ASSUME $start \in 1 \mathrel{..} 12$

$$
\begin{aligned}
Inv &\triangleq hr \in 1 \mathrel{..} 12 \\
Init &\triangleq hr = start \\
Inc &\triangleq hr < 12 \land hr' = hr + 1 \\
Reset &\triangleq hr = 12 \land hr' = 1 \\
Next &\triangleq Inc \lor Reset
\end{aligned}
$$

MACHINE *MyHourClock*

CONSTANTS *start*

VARIABLES *hr*

PROPERTIES $start \in 1 \mathrel{..} 12$

INVARIANT $hr \in 1 \mathrel{..} 12$

INITIALISATION $hr : (hr = start)$

END

Translation of the example

MODULE *MyHourClock*

EXTENDS *Naturals*
CONSTANTS *start*
VARIABLES *hr*
ASSUME $start \in 1 \,..\, 12$

$$Inv \triangleq hr \in 1 \,..\, 12$$
$$Init \triangleq hr = start$$
$$Inc \triangleq hr < 12 \land hr' = hr + 1$$
$$Reset \triangleq hr = 12 \land hr' = 1$$
$$Next \triangleq Inc \lor Reset$$

```
MACHINE MyHourClock
CONSTANTS start
VARIABLES hr
PROPERTIES start ∈ 1 .. 12
INVARIANT hr ∈ 1 .. 12
INITIALISATION hr : (hr = start)
OPERATIONS
  Inc_Op = ANY hr_n
     WHERE hr < 12 ∧ hr_n = hr + 1
     THEN hr := hr_n END

  Reset_Op = ANY hr_n
     WHERE hr = 12 ∧ hr_n = 1
     THEN hr := hr_n END
END
```

Translation of the example

MODULE *MyHourClock*

EXTENDS *Naturals*
CONSTANTS *start*
VARIABLES *hr*
ASSUME $start \in 1 \,..\, 12$

$$Inv \triangleq hr \in 1 \,..\, 12$$
$$Init \triangleq hr = start$$
$$Inc \triangleq hr < 12 \land hr' = hr + 1$$
$$Reset \triangleq hr = 12 \land hr' = 1$$
$$Next \triangleq Inc \lor Reset$$

MACHINE *MyHourClock*
CONSTANTS *start*
VARIABLES *hr*
PROPERTIES $start \in 1 \,..\, 12$
INVARIANT $hr \in 1 \,..\, 12$
INITIALISATION $hr : (hr = start)$
OPERATIONS
Inc_Op = ANY hr_n
WHERE $hr < 12 \land hr\_n = hr + 1$
THEN $hr := hr\_n$ END

$Reset\_Op$ = ANY hr_n
WHERE $hr = 12 \land hr\_n = 1$
THEN $hr := hr\_n$ END
END

Translation of the example

MODULE *MyHourClock*

EXTENDS *Naturals*
CONSTANTS *start*
VARIABLES *hr*
ASSUME $start \in 1\,..\,12$

$$
\begin{aligned}
Inv &\triangleq hr \in 1\,..\,12\\
Init &\triangleq hr = start\\
Inc &\triangleq hr < 12 \land hr' = hr + 1\\
Reset &\triangleq hr = 12 \land hr' = 1\\
Next &\triangleq Inc \lor Reset
\end{aligned}
$$

MACHINE *MyHourClock*
CONSTANTS *start*
VARIABLES *hr*
PROPERTIES $start \in 1\,..\,12$
INVARIANT $hr \in 1\,..\,12$
INITIALISATION $hr : (hr = start)$
OPERATIONS
  $Inc\_Op =$ ANY hr_n
    WHERE $hr < 12 \land hr\_n = hr + 1$
    THEN $hr := hr\_n$ END

  $Reset\_Op =$ ANY hr_n
    WHERE $hr = 12 \land hr\_n = 1$
    THEN $hr := hr\_n$ END
END

HEINRICH HEINE UNIVERSITÄT DÜSSELDORF

Translation of the example

MODULE *MyHourClock*

EXTENDS *Naturals*
CONSTANTS *start*
VARIABLES *hr*
ASSUME $start \in 1 \mathrel{..} 12$

$$
\begin{aligned}
Inv &\triangleq hr \in 1 \mathrel{..} 12 \\
Init &\triangleq hr = start \\
Inc &\triangleq hr < 12 \land hr' = hr + 1 \\
Reset &\triangleq hr = 12 \land hr' = 1 \\
Next &\triangleq Inc \lor Reset
\end{aligned}
$$

```
MACHINE MyHourClock
CONSTANTS start
VARIABLES hr
PROPERTIES start ∈ 1 .. 12
INVARIANT hr ∈ 1 .. 12
INITIALISATION hr : (hr = start)
OPERATIONS
  Inc_Op = ANY hr_n
     WHERE hr < 12 ∧ hr_n = hr + 1
     THEN hr := hr_n END

  Reset_Op = ANY hr_n
     WHERE hr = 12 ∧ hr_n = 1
     THEN hr := hr_n END
END
```

HEINRICH HEINE UNIVERSITÄT DÜSSELDORF

Translation of the example

MODULE *MyHourClock*

EXTENDS *Naturals*
CONSTANTS *start*
VARIABLES *hr*
ASSUME $start \in 1\,..\,12$

$$Inv \triangleq hr \in 1\,..\,12$$
$$Init \triangleq hr = start$$
$$Inc \triangleq hr < 12 \land hr' = hr + 1$$
$$Reset \triangleq hr = 12 \land hr' = 1$$
$$Next \triangleq Inc \lor Reset$$

```
MACHINE MyHourClock
CONSTANTS start
VARIABLES hr
PROPERTIES start ∈ 1 .. 12
INVARIANT hr ∈ 1 .. 12
INITIALISATION hr : (hr = start)
OPERATIONS
  Inc_Op = ANY hr_n
      WHERE hr < 12 ∧ hr_n = hr + 1
      THEN hr := hr_n END

  Reset_Op = ANY hr_n
      WHERE hr = 12 ∧ hr_n = 1
      THEN hr := hr_n END
END
```

Concepts of typing

- TLA$^+$ is untyped, while B is strongly typed
- Type informations are needed for the translation
  - B requires type declarations of symbols
  - the translation of some operators depends on them

Type inference algorithm (Hindley-Milner)

- Types of symbols can be inferred by considering the context the symbols are used
  - e.g. x must have the type Integer in the expression $x + 1$
- some resulting restriction for the translation
  - variables must have a fixed type
  - only values of the same type are comparable

HEINRICH HEINE UNIVERSITÄT DÜSSELDORF

Concepts of typing

- TLA$^+$ is untyped, while B is strongly typed
- Type informations are needed for the translation
  - B requires type declarations of symbols
  - the translation of some operators depends on them

Type inference algorithm (Hindley-Milner)

- Types of symbols can be inferred by considering the context the symbols are used
  - e.g. x must have the type Integer in the expression $x + 1$
- some resulting restriction for the translation
  - variables must have a fixed type
  - only values of the same type are comparable

HEINRICH HEINE UNIVERSITÄT DÜSSELDORF

Supported TLA$^+$ values

- The following kinds of values exist in both languages:
  - integers, boolean values, strings, sets, functions, sequences, records
- Restrictions caused by the B type system
  - all elements of a set must have the same type
  - functions and sequences are based on sets
- Model values are translated using enumerated sets

  $\Rightarrow$ the translation conserves symmetry properties

Supported TLA$^+$ operators

- Most TLA$^+$ operators can be mapped to B built-in operators
  - operators of the standard modules Naturals, Integers, Sequences, FiniteSets
- Other operators can be expressed by a combination of B operators
  - e.g. if-then-else
- User-defined operators are translated using B Definitions which are a kind of macro

HEINRICH HEINE UNIVERSITÄT DÜSSELDORF

CHOOSE operator

Its general functionality can not be expressed in B

Standard Module TLA2B

- contains useful operators such as minimum, maximum, sum and product of a set
- this operators can be mapped to B build-in operators

Extending B

- PROB is able to load externally defined (polymorhic) functions
- semantics of the CHOOSE operator is expressed in this way

HEINRICH HEINE UNIVERSITÄT DÜSSELDORF

CHOOSE operator

Its general functionality can not be expressed in B

Standard Module TLA2B

- contains useful operators such as minimum, maximum, sum and product of a set
- this operators can be mapped to B build-in operators

Extending B

- PROB is able to load externally defined (polymorhic) functions
- semantics of the CHOOSE operator is expressed in this way

HEINRICH HEINE
UNIVERSITÄT DÜSSELDORF

CHOOSE operator

Its general functionality can not be expressed in B

Standard Module TLA2B

- contains useful operators such as minimum, maximum, sum and product of a set
- this operators can be mapped to B build-in operators

Extending B

- PROB is able to load externally defined (polymorhic) functions
- semantics of the CHOOSE operator is expressed in this way

HEINRICH HEINE UNIVERSITÄT DÜSSELDORF

Translator: TLA2B

- Automatic translation
- The frontend is based on the parser SANY
- Reuse of TLC's configuration file parser
  - constant/operator assignment and replacement
- Stand-alone translator

Integration of TLA2B into ProB

Applying PROB to TLA$^+$ specification

PROB

- Model Checking
- Constraint-based checking
- Test generation
- Animation
- Visualization

HEINRICH HEINE UNIVERSITÄT DÜSSELDORF

SimpleAllocator case study (by Merz)

- System to manage a set of resources (Rs) that are shared among a number of client processes (Cs)
- The specification allows TLC and ProB to use symmetry

Cs	Rs	TLC		TLA2B + ProB	
		no symmetry	symmetry	no symmetry	symmetry
3	2	<1 s	<1 s	2 s	<1 s
4	3	28 s	2 s	678 s	8 s
5	3	450 s	29 s	-	28 s
6	3	>4200 s	573 s	-	90 s

- Without symmetry TLC is superior to ProB
- For larger set sizes ProB's symmetry outperforms TLC

HEINRICH HEINE UNIVERSITÄT DÜSSELDORF

SimpleAllocator case study (by Merz)

- System to manage a set of resources (Rs) that are shared among a number of client processes (Cs)
- The specification allows TLC and ProB to use symmetry

Cs	Rs	TLC		TLA2B + ProB	
		no symmetry	symmetry	no symmetry	symmetry
3	2	<1 s	<1 s	2 s	<1 s
4	3	28 s	2 s	678 s	8 s
5	3	450 s	29 s	-	28 s
6	3	>4200 s	573 s	-	90 s

- Without symmetry TLC is superior to ProB
- For larger set sizes ProB's symmetry outperforms TLC

HEINRICH HEINE UNIVERSITÄT DÜSSELDORF

SimpleAllocator case study (by Merz)

- System to manage a set of resources (Rs) that are shared among a number of client processes (Cs)
- The specification allows TLC and ProB to use symmetry

Cs	Rs	TLC		TLA2B + ProB	
		no symmetry	symmetry	no symmetry	symmetry
3	2	$<$1 s	$<$1 s	2 s	$<$1 s
4	3	28 s	2 s	678 s	8 s
5	3	450 s	29 s	-	28 s
6	3	$>$4200 s	573 s	-	90 s

- Without symmetry TLC is superior to ProB
- For larger set sizes ProB's symmetry outperforms TLC

Constraint solving

N-Queens:

- Searching for all valid placements of N queens on an N×N chessboard so that no two queens attack each other
- The solution is encoded in a declarative way

N	Solutions	TLC	TLA2B + ProB
6	4	1 s	<1 s
7	40	16 s	<1 s
8	92	375 s	<1 s
9	352	2970 s	<1 s
11	2,680	-	<1 s
13	73,712	-	41 s

- PROB is supierior to TLC
- searching for a single solution:
  PROB finds a solution for N = 50 in less than a second

HEINRICH HEINE UNIVERSITÄT DÜSSELDORF

Constraint solving

N-Queens:

- Searching for all valid placements of N queens on an N×N chessboard so that no two queens attack each other
- The solution is encoded in a declarative way

N	Solutions	TLC	TLA2B + ProB
6	4	1 s	<1 s
7	40	16 s	<1 s
8	92	375 s	<1 s
9	352	2970 s	<1 s
11	2,680	-	<1 s
13	73,712	-	41 s

- PROB is supierior to TLC
- searching for a single solution: PROB finds a solution for N = 50 in less than a second

Constraint solving

N-Queens:

- Searching for all valid placements of N queens on an N×N chessboard so that no two queens attack each other
- The solution is encoded in a declarative way

N	Solutions	TLC	TLA2B + ProB
6	4	1 s	<1 s
7	40	16 s	<1 s
8	92	375 s	<1 s
9	352	2970 s	<1 s
11	2,680	-	<1 s
13	73,712	-	41 s

- ProB is supierior to TLC
- searching for a single solution:
  ProB finds a solution for N = 50 in less than a second

TLA2B + ProB

Demo

- ProB: http://www.stups.uni-duesseldorf.de/ProB/
- TLA2B: http://nightly.cobra.cs.uni-duesseldorf.de/tla/
- Logic Calculator: http://cobra.cs.uni-duesseldorf.de/evalB/

Conclusion & Future work

- Translator TLA2B
  - type inference algorithm
  - supports a large subset of TLA$^+$
- Integration of TLA2B into ProB
  - gain a new tool for TLA$^+$
  - complementary to TLC
- all B tools can be apply to the translated B machine

Future work:

- Testing the correctness of the translation
  - comparing the state spaces generated by TLC and ProB
- Improving and extending the translation
  - support for recursive function

HEINRICH HEINE
UNIVERSITÄT DÜSSELDORF

Questions?

if-then-else

- If-then-else is an expression in TLA$^+$

  e.g. $\textsc{if}\ x = 0\ \textsc{then}\ 1\ \textsc{else}\ 1/x$

- Cannot use B's if-then-else substitution for translation

HEINRICH HEINE
UNIVERSITÄT DÜSSELDORF

if-then-else

- If-then-else is an expression in TLA$^+$

  e.g. IF $x = 0$ THEN 1 ELSE $1/x$

- Cannot use B's if-then-else substitution for translation

Translation of IF P THEN e_1 ELSE e_2

HEINRICH HEINE
UNIVERSITÄT DÜSSELDORF

if-then-else

- If-then-else is an expression in TLA$^+$

  e.g. $\text{IF } x = 0 \text{ THEN } 1 \text{ ELSE } 1/x$

- Cannot use B's if-then-else substitution for translation

Translation of $\text{IF } P \text{ THEN } e_1 \text{ ELSE } e_2$

$\Rightarrow$ e_1 and e_2 must have the same type

if-then-else

- If-then-else is an expression in TLA$^+$

  e.g. $\text{IF } x = 0 \text{ THEN } 1 \text{ ELSE } 1/x$

- Cannot use B's if-then-else substitution for translation

Translation of $\text{IF } P \text{ THEN } e_1 \text{ ELSE } e_2$

$\Rightarrow$ e_1 and e_2 must have the same type

1. in case of boolean type
   $(P \Rightarrow e_1) \wedge (\neg(P) \Rightarrow e_2)$

HEINRICH HEINE
UNIVERSITÄT DÜSSELDORF

if-then-else

- If-then-else is an expression in TLA$^+$

  e.g. $\text{IF } x = 0 \text{ THEN } 1 \text{ ELSE } 1/x$

- Cannot use B's if-then-else substitution for translation

Translation of $\text{IF } P \text{ THEN } e_1 \text{ ELSE } e_2$

$\Rightarrow$ e_1 and e_2 must have the same type

1. in case of boolean type
   $(P \Rightarrow e_1) \wedge (\neg(P) \Rightarrow e_2)$
2. other
   $(\lambda t.(t \in \{\text{TRUE}\} \wedge P | e_1) \cup \lambda t.(t \in \{\text{TRUE}\} \wedge \neg P | e_2))(\text{TRUE})$

HEINRICH HEINE UNIVERSITÄT DÜSSELDORF

if-then-else

- If-then-else is an expression in TLA$^+$

  e.g. $\text{IF } x = 0 \text{ THEN } 1 \text{ ELSE } 1/x$

- Cannot use B's if-then-else substitution for translation

Translation of $\text{IF } P \text{ THEN } e_1 \text{ ELSE } e_2$

$\Rightarrow$ e_1 and e_2 must have the same type

1. in case of boolean type
   $(P \Rightarrow e_1) \wedge (\neg(P) \Rightarrow e_2)$
2. other
   $(\lambda t.(t \in \{\text{TRUE}\} \wedge P | e_1) \cup \lambda t.(t \in \{\text{TRUE}\} \wedge \neg P | e_2))(\text{TRUE})$

HEINRICH HEINE
UNIVERSITÄT DÜSSELDORF

if-then-else

- If-then-else is an expression in TLA$^+$

  e.g. $\text{IF } x = 0 \text{ THEN } 1 \text{ ELSE } 1/x$

- Cannot use B's if-then-else substitution for translation

Translation of $\text{IF } P \text{ THEN } e_1 \text{ ELSE } e_2$

$\Rightarrow$ e_1 and e_2 must have the same type

1. in case of boolean type
   $(P \Rightarrow e_1) \land (\neg(P) \Rightarrow e_2)$
2. other
   $(\lambda t.(t \in \{\text{TRUE}\} \land P | e_1) \cup \lambda t.(t \in \{\text{TRUE}\} \land \neg P | e_2))(\text{TRUE})$

HEINRICH HEINE
UNIVERSITÄT DÜSSELDORF

if-then-else

- If-then-else is an expression in TLA$^+$

  e.g. $\text{IF } x = 0 \text{ THEN } 1 \text{ ELSE } 1/x$

- Cannot use B's if-then-else substitution for translation

Translation of $\text{IF } P \text{ THEN } e_1 \text{ ELSE } e_2$

$\Rightarrow$ e_1 and e_2 must have the same type

1. in case of boolean type
   $(P \Rightarrow e_1) \wedge (\neg(P) \Rightarrow e_2)$
2. other
   $(\lambda t.(t \in \{\text{TRUE}\} \wedge P | e_1) \cup \lambda t.(t \in \{\text{TRUE}\} \wedge \neg P | e_2))(\text{TRUE})$

HEINRICH HEINE
UNIVERSITÄT DÜSSELDORF

if-then-else

- If-then-else is an expression in TLA$^+$

  e.g. $\text{IF } x = 0 \text{ THEN } 1 \text{ ELSE } 1/x$

- Cannot use B's if-then-else substitution for translation

Translation of $\text{IF } P \text{ THEN } e_1 \text{ ELSE } e_2$

$\Rightarrow$ e_1 and e_2 must have the same type

1. in case of boolean type
   $(P \Rightarrow e_1) \wedge (\neg(P) \Rightarrow e_2)$
2. other
   $(\lambda t.(t \in \{\text{TRUE}\} \wedge P | e_1) \cup \lambda t.(t \in \{\text{TRUE}\} \wedge \neg P | e_2))(\text{TRUE})$

HEINRICH HEINE
UNIVERSITÄT DÜSSELDORF

Records

- In B the Type of a record depends on
  - the fields of the record
  - the order of the fields
  - the types of the fields
- How to translate $[a \mapsto 1] = [b \mapsto TRUE]$?

  $rec(a : 1) = rec(b : TRUE)$

  $rec(a : \quad , b : \quad) = rec(a : \quad , b : \quad)$
- get a field x of a record r (TLA$^+$: $r.x$)

  $r'x$

Records

- In B the Type of a record depends on
  - the fields of the record
  - the order of the fields
  - the types of the fields
- How to translate $[a \mapsto 1] = [b \mapsto TRUE]$?

  $rec(a : 1) = rec(b : TRUE)$

  $rec(a : \quad , b : \quad) = rec(a : \quad , b : \quad)$
- get a field x of a record r (TLA$^+$: $r.x$)

  $r'x$

HEINRICH HEINE
UNIVERSITÄT DÜSSELDORF

Records

- In B the Type of a record depends on
  - the fields of the record
  - the order of the fields
  - the types of the fields
- How to translate $[a \mapsto 1] = [b \mapsto TRUE]$?

  $rec(a : 1) = rec(b : TRUE)$

  $rec(a : \quad , b : \quad) = rec(a : \quad , b : \quad)$
- get a field x of a record r (TLA$^+$: $r.x$)

  $r'x$

HEINRICH HEINE UNIVERSITÄT DÜSSELDORF

Records

- In B the Type of a record depends on
  - the fields of the record
  - the order of the fields
  - the types of the fields
- How to translate $[a \mapsto 1] = [b \mapsto TRUE]$?

  $rec(a : 1) = rec(b : TRUE)$ Typeerror

  $rec(a : \quad , b : \quad) = rec(a : \quad , b : \quad)$
- get a field x of a record r (TLA$^+$: $r.x$)

  $r'x$

Records

- In B the Type of a record depends on
  - the fields of the record
  - the order of the fields
  - the types of the fields
- How to translate $[a \mapsto 1] = [b \mapsto TRUE]$?

  $rec(a : 1) = rec(b : TRUE)$ Typeerror

  $rec(a : \quad , b : \quad) = rec(a : \quad , b : \quad)$
- get a field x of a record r (TLA$^+$: $r.x$)

  $r'x$

Records

- In B the Type of a record depends on
  - the fields of the record
  - the order of the fields
  - the types of the fields
- How to translate $[a \mapsto 1] = [b \mapsto TRUE]$?

  $rec(a : 1) = rec(b : TRUE)$ Typeerror

  $rec(a : \{TRUE \mapsto 1\}, b : \{\}) = rec(a : \{\}, b : \{TRUE \mapsto TRUE\})$
- get a field x of a record r (TLA$^+$: $r.x$)

  $r'x$

HEINRICH HEINE
UNIVERSITÄT DÜSSELDORF

Records

- In B the Type of a record depends on
  - the fields of the record
  - the order of the fields
  - the types of the fields
- How to translate $[a \mapsto 1] = [b \mapsto TRUE]$?

  $rec(a:1) = rec(b:TRUE)$ Typeerror

  $rec(a:\{TRUE \mapsto 1\}, b:\{\}) = rec(a:\{\}, b:\{TRUE \mapsto TRUE\})$
- get a field x of a record r (TLA$^+$: $r.x$)

  $r'x$

HEINRICH HEINE
UNIVERSITÄT DÜSSELDORF

Records

- In B the Type of a record depends on
  - the fields of the record
  - the order of the fields
  - the types of the fields
- How to translate $[a \mapsto 1] = [b \mapsto TRUE]$?

  $rec(a:1) = rec(b:TRUE)$ Typeerror

  $rec(a:\{TRUE \mapsto 1\}, b:\{\}) = rec(a:\{\}, b:\{TRUE \mapsto TRUE\})$
- get a field x of a record r (TLA$^+$: $r.x$)

  $r'x$

HEINRICH HEINE UNIVERSITÄT DÜSSELDORF

Records

- In B the Type of a record depends on
  - the fields of the record
  - the order of the fields
  - the types of the fields
- How to translate $[a \mapsto 1] = [b \mapsto TRUE]$?

  $rec(a : 1) = rec(b : TRUE)$ Typeerror

  $rec(a : \{TRUE \mapsto 1\}, b : \{\}) = rec(a : \{\}, b : \{TRUE \mapsto TRUE\})$
- get a field x of a record r (TLA$^+$: $r.x$)

  $r'x(TRUE)$

HEINRICH HEINE UNIVERSITÄT DÜSSELDORF