

Tutorial: Desenvolvendo Componentes de Software com o Método B

David Déharbe

ForAll — Formal Methods and Research Laboratory
DIMAp — Departamento de Informática e Matemática Aplicada
UFRN — Universidade Federal do Rio Grande do Norte
Natal, RN, Brazil
INES — INC&T para Engenharia de Software

Congresso Brasileiro de Software, 2014




Programa de Pós-graduação em Sistemas e Computação
Departamento de Informática e Matemática Aplicada
Universidade Federal do Rio Grande do Norte — Natal

- ▶ Mestrado, Doutorado (fluxo contínuo, bolsas disponíveis)
- ▶ Algoritmos Experimentais, Engenharia de Software, Fundamentos da Computação, Linguagens de Programação e Métodos Formais, Processamento Gráfico e Inteligência Computacional, Sistemas Integrados e Distribuídos.
- ▶ Conceito 5
- ▶ <http://www.posgraduacao.ufrn.br/ppgsc>

Programa de Pós-graduação em Engenharia de Software
Instituto Metrópole Digital
Universidade Federal do Rio Grande do Norte — Natal

- ▶ Mestrado Profissional
- ▶ Desenvolvimento de Jogos Digitais, Engenharia de Sistemas Web.

Principais referências

-  J.-R. Abrial.
The B-Book: Assigning programs to meanings.
Oxford University Press, 1996.
-  S. Schneider
the b-method — an introduction
Palgrave McMillan 2001
-  J.B. Wordsworth.
Software Engineering with B.
Addison-Wesley, 1996.

Roteiro

Introdução

Contextualização

Visão global do método B

(tira-gosto)

Especificação dos requisitos funcionais

Uma notação para especificar

Verificação da especificação

Substituições generalizadas e obrigações de prova

Refinamento

Uma notação para refinar

Verificação de refinamentos

Implementação

Introdução

Uma linguagem para implementar

Ferramentas

Ferramentas industriais

Complementos

Outline

Introdução

Contextualização

Visão global do método B

(tira-gosto)

Especificação dos requisitos funcionais

Uma notação para especificar

Verificação da especificação

Substituições generalizadas e obrigações de prova

Refinamento

Uma notação para refinar

Verificação de refinamentos

Implementação

Introdução

Uma linguagem para implementar

Ferramentas

Ferramentas industriais

Complementos

Qual problemas o método B se propõe a solucionar?

B is a method for specifying, designing, and coding software systems [1].

- ▶ Objetivo: desenvolver componentes de software;
- ▶ Abordagem *formal*:
 - ▶ sintaxe
 - ▶ semântica precisa
 - ▶ raciocínio matemático para garantir corretude
- ▶ Presuposto: requisitos funcionais são definidos;
- ▶ Plataformas alvo: modelo de programação imperativo, sequencial, sem recursão, sem alocação dinâmica de memória.
- ▶ Existem ferramentas comerciais para aplicar o método B;
- ▶ Existem empresas que usam o método B.
- ▶ Existem sistemas que executam software desenvolvido com o método B.

Histórico breve

- ▶ Conceitos de base:
 - ▶ Cálculo de pré-condição mais fraca (Dijkstra);
 - ▶ Linguagens de especificação formal Z e VDM;
 - ▶ Conceitos de refinamento desenvolvido por Hoare e Jones.
- ▶ 85–88: Criação no Oxford University Programming Research Group;
- ▶ Pesquisador principal: Jean-Raymond Abrial.
Contribuidores D. Gries, J. Prinz, C.C. Morgan, P. Gardiner, I.H. Sorensen;
- ▶ 88–94: Ferramentas comerciais (B-Toolkit).
- ▶ 2014: Ferramenta comercial Atelier-B 4.2.

Casos de sucesso

- ▶ Financiamento do projeto fundador: BP;
- ▶ Desenvolvimento do B-Toolkit: colaboração com GEC Alsthom;
- ▶ Desenvolvimento de sub-sistemas de controle do metrô: MATRA Transport, Siemens;
- ▶ Projeto de software na área de aeronáutica: GEC-Marconi, Praxis.

Outline

Introdução

- Contextualização

- Visão global do método B**

- (tira-gosto)

- Especificação dos requisitos funcionais

 - Uma notação para especificar

 - Verificação da especificação

 - Substituições generalizadas e obrigações de prova

- Refinamento

 - Uma notação para refinar

 - Verificação de refinamentos

- Implementação

 - Introdução

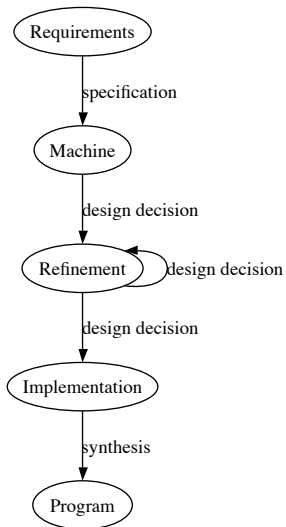
 - Uma linguagem para implementar

- Ferramentas

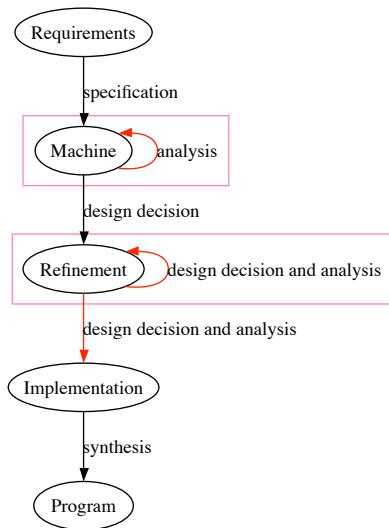
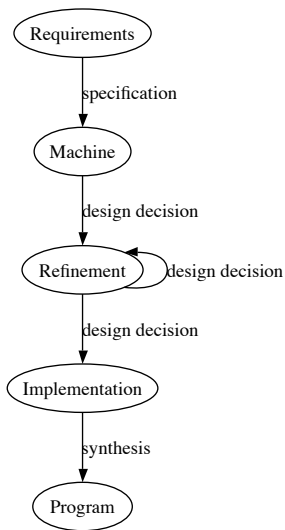
 - Ferramentas industriais

- Complementos

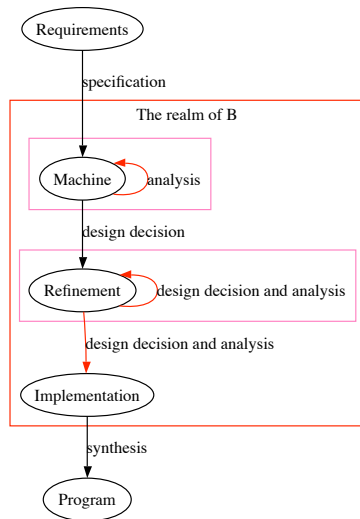
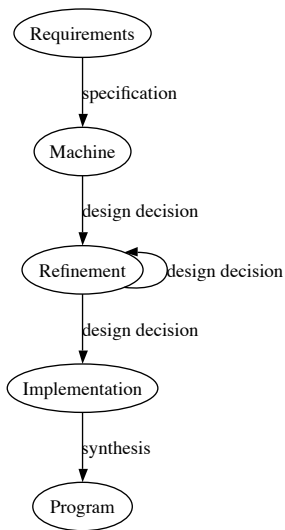
Uma visão global



Uma visão global



Uma visão global



Uma visão global

- ▶ análise da máquina: validação, verificação de viabilidade e de coerência.
 - ▶ animação/simulação
 - ▶ provas
- ▶ decisões de projeto: escolha de estruturas de dados, implementação das funcionalidades com algoritmos.
 - ▶ decisão humana
 - ▶ leis de refinamento
- ▶ análise de refinamentos: conformidade com artefatos mais abstratos.
 - ▶ provas

Uma visão global

- ▶ análise da máquina: validação, verificação de viabilidade e de coerência.
 - ▶ animação/simulação
 - ▶ provas
- ▶ decisões de projeto: escolha de estruturas de dados, implementação das funcionalidades com algoritmos.
 - ▶ decisão humana
 - ▶ leis de refinamento
- ▶ análise de refinamentos: conformidade com artefatos mais abstratos.
 - ▶ provas

processo tipo cascata + projeto dirigido por modelo

(tira-gosto)

Outline

Introdução

Contextualização

Visão global do método B

(tira-gosto)

Especificação dos requisitos funcionais

Uma notação para especificar

Verificação da especificação

Substituições generalizadas e obrigações de prova

Refinamento

Uma notação para refinar

Verificação de refinamentos

Implementação

Introdução

Uma linguagem para implementar

Ferramentas

Ferramentas industriais

Complementos

Modelo semântico

- ▶ Grafo dirigido
- ▶ Vértice = Estado
 - ▶ valoração das variáveis do modelo
- ▶ Transições:
 - ▶ Conecta cada estado com sucessores possíveis
- ▶ Estados iniciais
- ▶ Estados legais

Modelo semântico

- ▶ Grafo dirigido
- ▶ Vértice = Estado
 - ▶ valoração das variáveis do modelo
- ▶ Transições:
 - ▶ Conecta cada estado com sucessores possíveis
- ▶ Estados iniciais
- ▶ Estados legais

TAMANHO \rightsquigarrow representação implícita

Modelo semântico: exemplo

- ▶ Variáveis de estado: $DATA, m$
- ▶ Um estado possível $DATA = \{2, 3, -1\}, m = 3$
- ▶ Estados e lógica dos predicados:
 - ▶ estado: $DATA = \{2, 3, -1\} \wedge m = 3$
 - ▶ conjunto de estados: $\mathbf{card}(DATA) > 0 \implies m \in DATA$
- ▶ Estados, lógica e substituição:
 - ▶ $DATA, m := \{2, 3, -1\}, 3$
- ▶ Estados iniciais
 - ▶ $DATA = \{\}$ ou $DATA := \{\}$
- ▶ Estados legais
 - ▶ $DATA \neq \{\} \implies m = \mathbf{max}(DATA)$

Modelo semântico: exemplo

- ▶ $DATA = \{2, 3, -1\} \wedge m = 3 \rightsquigarrow DATA = \{2, -1\} \wedge m = 2$
- ▶ Transições e lógica dos predicados
 - ▶ $DATA = \{2, 3, -1\} \wedge m = 3 \wedge$
 $DATA' = \{2, -1\} \wedge m' = 2$
- ▶ Transições, lógica e substituição:
 - ▶ $[DATA, m := DATA - \{3\}, \mathbf{max}(DATA - \{3\})]$
 $DATA = \{2, 3, -1\} \wedge m = 3$

Especificação de componente em B

- ▶ Módulo básico de especificação: **máquina**.
- ▶ Máquina de estados e transições.
- ▶ Relação das variáveis.
- ▶ Especificação dos estados autorizados
- ▶ Especificação dos estados iniciais
- ▶ Especificação das transições
- ▶ Extras:
 - ▶ parâmetros, constantes, módulos de definições, sub-componentes, propriedades adicionais.

Estrutura das máquinas

MACHINE *Name (Parameters)*

CONSTRAINTS *specifies parameters*

SEES *accesses external definitions*

USES *lists sub-components*

CONSTANTS *names useful concepts*

PROPERTIES *specifies useful concepts*

VARIABLES *list of variables*

INVARIANT

invariant predicate

INITIALISATION

initialization substitution

OPERATIONS

outputs \leftarrow name(inputs) $\hat{=}$ substitution

END

Obs. Alguns tipos de cláusulas foram omitidas.

Estrutura das máquinas

MACHINE *Name (Parameters)*

CONSTRAINTS *specifies parameters*

SEES *accesses external definitions*

USES *lists sub-components*

CONSTANTS *names useful concepts*

PROPERTIES *specifies useful concepts*

VARIABLES *list of variables*

INVARIANT

invariant predicate

INITIALISATION

initialization substitution

OPERATIONS

outputs \leftarrow name(inputs) $\hat{=}$ substitution

END

Obs. Alguns tipos de cláusulas foram omitidas.

Estrutura das máquinas

MACHINE *Name (Parameters)*

CONSTRAINTS *specifies parameters*

SEES *accesses external definitions*

USES *lists sub-components*

CONSTANTS *names useful concepts*

PROPERTIES *specifies useful concepts*

VARIABLES *list of variables*

INVARIANT

invariant predicate

INITIALISATION

initialization substitution

OPERATIONS

outputs \leftarrow name(inputs) $\hat{=}$ substitution

END

Obs. Alguns tipos de cláusulas foram omitidas.

Estrutura das máquinas

MACHINE *Name (Parameters)*

CONSTRAINTS *specifies parameters*

SEES *accesses external definitions*

USES *lists sub-components*

CONSTANTS *names useful concepts*

PROPERTIES *specifies useful concepts*

VARIABLES *list of variables*

INVARIANT

invariant predicate

INITIALISATION

initialization substitution

OPERATIONS

outputs \leftarrow name(inputs) $\hat{=}$ substitution

END

Obs. Alguns tipos de cláusulas foram omitidas.

Estrutura das máquinas

MACHINE *Name (Parameters)*

CONSTRAINTS *specifies parameters*

SEES *accesses external definitions*

USES *lists sub-components*

CONSTANTS *names useful concepts*

PROPERTIES *specifies useful concepts*

VARIABLES *list of variables*

INVARIANT

invariant predicate

INITIALISATION

initialization substitution

OPERATIONS

outputs \leftarrow name(inputs) $\hat{=}$ substitution

END

Obs. Alguns tipos de cláusulas foram omitidas.

Estrutura das máquinas

MACHINE *Name (Parameters)*

CONSTRAINTS *specifies parameters*

SEES *accesses external definitions*

USES *lists sub-components*

CONSTANTS *names useful concepts*

PROPERTIES *specifies useful concepts*

VARIABLES *list of variables*

INVARIANT

invariant predicate

INITIALISATION

initialization substitution

OPERATIONS

outputs \leftarrow name(inputs) $\hat{=}$ substitution

END

Obs. Alguns tipos de cláusulas foram omitidas.

Estrutura das máquinas

MACHINE *Name (Parameters)*

CONSTRAINTS *specifies parameters*

SEES *accesses external definitions*

USES *lists sub-components*

CONSTANTS *names useful concepts*

PROPERTIES *specifies useful concepts*

VARIABLES *list of variables*

INVARIANT

invariant predicate

INITIALISATION

initialization substitution

OPERATIONS

outputs \leftarrow name(inputs) $\hat{=}$ substitution

END

Obs. Alguns tipos de cláusulas foram omitidas.

Estrutura das máquinas

MACHINE *Name (Parameters)*

CONSTRAINTS *specifies parameters*

SEES *accesses external definitions*

USES *lists sub-components*

CONSTANTS *names useful concepts*

PROPERTIES *specifies useful concepts*

VARIABLES *list of variables*

INVARIANT

invariant predicate

INITIALISATION

initialization substitution

OPERATIONS

outputs \leftarrow name(inputs) $\hat{=}$ substitution

END

Obs. Alguns tipos de cláusulas foram omitidas.

Tipos de dados e expressões

- ▶ tipos
 - ▶ tipos inteiros, booleanos, tipos enumerados, “tipos abstratos”;
 - ▶ produto cartesiano, registros, conjunto das partes;
- ▶ expressões:
 - ▶ expressões aritméticas e booleanas,
 - ▶ pares, tuplas,
 - ▶ conjuntos definidos por extenso, conjuntos definidos em intenção,
 - ▶ relações, funções, expressões lambda.

Alguns exemplos de substituições

- ▶ substituição simples
 $v := E$
- ▶ substituição múltipla
 $v_1 \cdots v_n := E_1 \cdots E_n$
- ▶ substituição múltipla
 $v_1, \dots, v_n := E_1, \dots, E_n$
- ▶ substituição simultânea
 $S_1 \parallel \cdots \parallel S_n$
- ▶ faça elemento de
 $v \in E$
- ▶ faça tal que
 $v : (P)$

Escolha limitada

Sintaxe:

CHOICE

S_1

OR

S_2

OR

...

END

Example

CHOICE

$X := -X$

OR

SKIP

END

Escolha ilimitada

Sintaxe:

ANY

$v_1 \cdots v_n$

WHERE

$P(v_1, \cdots, v_n)$

THEN

S

END

Example

ANY v **WHERE**

$n_1 \bmod v = 0 \wedge n_2 \bmod v = 0 \wedge$

$\forall w \cdot w > v \Rightarrow n_1 \bmod w \neq 0 \vee n_2 \bmod w \neq 0$

THEN

$val := v$

END

Escolha limitada condicional

Sintaxe:

```
SELECT  $P_1$   
THEN  $S_1$   
WHEN  $P_2$   
THEN  $S_2$   
...  
ELSE  $S_n$   
END
```

Example

```
SELECT  $v1 \geq v2$   
THEN  $res := v1$   
WHEN  $v2 \geq v1$   
THEN  $res := v2$   
END
```

Exemplo de máquina (1)

LowBound (1/2)

MACHINE *LowBound*

VARIABLES *DATA*, *smallest*

INVARIANT

$DATA \in \mathbb{P}(INT) \wedge smallest \in INT \wedge$
 $(DATA \neq \emptyset \Rightarrow smallest = \mathbf{min}(DATA))$

INITIALISATION $DATA := \emptyset \parallel greatest := \in INT$

OPERATIONS

drop $\hat{=}$

PRE $DATA \neq \emptyset$ **THEN**

ANY *value* **WHERE** $VALUE \in DATA$ **THEN**

$DATA := DATA - \{value\} \parallel$

SELECT $\mathbf{card}(DATA) \geq 2$ **THEN**

$smallest := \mathbf{min}(DATA - \{value\})$

END

END

END;

Exemplo de máquina (1)

LowBound (2/2)

```
res  $\leftarrow$  get  $\hat{=}$   
PRE DATA  $\neq \emptyset$  THEN  
  res := min(DATA)  
END  
END;  
put(value)  $\hat{=}$   
PRE value  $\in$  INT THEN  
  DATA := DATA  $\cup$  {value} || smallest := min(DATA  $\cup$  {value})  
END  
END
```

Exemplo de máquina (2)

Caderno de aniversário (parte I)

MACHINE *BirthdayAgenda* (*NAME*, *DATE*)

VARIABLES *known*, *birthday*

INVARIANT

$known \subseteq NAME \wedge birthday \in known \rightarrow DATE$

INITIALISATION $known, birthday := \emptyset, \emptyset$

OPERATIONS

Register (*n*, *d*) $\hat{=}$

PRE $n \in NAME \wedge d \in DATE \wedge n \notin known$

THEN

$known := known \cup \{n\} \parallel$

$birthday := birthday \cup \{n \mapsto d\}$

END

Exemplo de máquina (2)

Caderno de aniversário (parte II)

```
 $d \leftarrow \text{FindBirthday } (n) \hat{=} \\ \mathbf{PRE} \\ n \in \text{NAME} \wedge n \in \text{known} \\ \mathbf{THEN} \\ d := \text{birthday}(n) \\ \mathbf{END}; \\ a \leftarrow \text{FindParty } (d) \hat{=} \\ \mathbf{PRE} \\ d \in \text{DATE} \\ \mathbf{THEN} \\ a := \text{birthday}^{-1} (d) \\ \mathbf{END} \\ \mathbf{END}$ 
```

Exemplo de máquina (2)

Caderno de aniversário (parte II)

```
 $d \leftarrow \text{FindBirthday} (n) \hat{=} \\ \text{PRE} \\ n \in \text{NAME} \wedge n \in \text{known} \\ \text{THEN} \\ d := \text{birthday}(n) \\ \text{END}; \\ a \leftarrow \text{FindParty} (d) \hat{=} \\ \text{PRE} \\ d \in \text{DATE} \\ \text{THEN} \\ a := \text{birthday}^{-1} (d) \\ \text{END} \\ \text{END}$ 
```


Exemplo de máquina (2)

Caderno de aniversário (parte II)

$d \leftarrow \text{FindBirthday } (n) \hat{=}$

PRE

$n \in \text{known}$

THEN

$d := \text{birthday}(n)$

END;

$a \leftarrow \text{FindParty } (d) \hat{=}$

PRE

$d \in \text{DATE}$

THEN

$a := \text{birthday}^{-1} (d)$

END

END

Outline

Introdução

Contextualização

Visão global do método B

(tira-gosto)

Especificação dos requisitos funcionais

Uma notação para especificar

Verificação da especificação

Substituições generalizadas e obrigações de prova

Refinamento

Uma notação para refinar

Verificação de refinamentos

Implementação

Introdução

Uma linguagem para implementar

Ferramentas

Ferramentas industriais

Complementos

Verificação da especificação

- ▶ Todas as expressões constando da especificação devem ser bem definidas:
 - ▶ $\dots \mathbf{min}(S) \dots : \vdash S \neq \emptyset?$

Verificação da especificação

- ▶ Todas as expressões constando da especificação devem ser bem definidas:
 - ▶ $\dots \mathbf{min}(S) \dots : \vdash S \neq \emptyset?$
- ▶ As restrições sobre parâmetros, constantes, variáveis devem ser satisfatíveis.

Verificação da especificação

- ▶ Todas as expressões constando da especificação devem ser bem definidas:
 - ▶ $\dots \mathbf{min}(S) \dots : \vdash S \neq \emptyset?$
- ▶ As restrições sobre parâmetros, constantes, variáveis devem ser satisfatíveis.
- ▶ A máquina deve iniciar em um estado autorizado:

A inicialização deve estabelecer o invariante.

Verificação da especificação

- ▶ Todas as expressões constando da especificação devem ser bem definidas:
 - ▶ $\dots \text{min}(S) \dots : \vdash S \neq \emptyset?$
- ▶ As restrições sobre parâmetros, constantes, variáveis devem ser satisfatíveis.
- ▶ A máquina deve iniciar em um estado autorizado:

A inicialização deve estabelecer o invariante.

- ▶ As operações, quando suas pré-condições são satisfeitas, não devem levar a máquina de um estado autorizado para um estado não autorizada máquina:

As operações devem preservar o invariante.

Verificação da especificação

- ▶ Todas as expressões constando da especificação devem ser bem definidas:
 - ▶ $\dots \text{min}(S) \dots : \vdash S \neq \emptyset?$
- ▶ As restrições sobre parâmetros, constantes, variáveis devem ser satisfatíveis.
- ▶ A máquina deve iniciar em um estado autorizado:

A inicialização deve estabelecer o invariante.

- ▶ As operações, quando suas pré-condições são satisfeitas, não devem levar a máquina de um estado autorizado para um estado não autorizada máquina:

As operações devem preservar o invariante.

- ▶ Geração de **obrigações de prova** e verificação das mesmas.

Formatação das obrigações de prova

- ▶ Fórmula lógica:
 - ▶ $H_1 \wedge H_2 \wedge \cdots H_n \Rightarrow G$
 - ▶ é válida?
- ▶ Sequente:
 - ▶ $H_1, H_2, \cdots H_n \vdash G$

Obrigações de prova para a inicialização

A inicialização deve estabelecer o invariante:

$$H_p, H_i, H_c \vdash [S]Inv,$$

onde:

- ▶ H_p são as restrições sobre os parâmetros da máquina.
- ▶ H_i são as definições importadas.
- ▶ H_c são as restrições sobre as constantes da máquina
- ▶ S é a substituição da inicialização, e
- ▶ Inv é o invariante.

Obrigações de prova para a inicialização

A inicialização deve estabelecer o invariante:

$$H_p, H_i, H_c \vdash [S]Inv,$$

onde:

- ▶ H_p são as restrições sobre os parâmetros da máquina.
- ▶ H_i são as definições importadas.
- ▶ H_c são as restrições sobre as constantes da máquina
- ▶ S é a substituição da inicialização, e
- ▶ Inv é o invariante.

Example

Máquina *BirthdayAgenda*:

$$\vdash [known, birthday := \emptyset, \emptyset] \\ known \subseteq NAME \wedge birthday \in known \rightarrow DATE$$

Obrigações de prova para a inicialização

A inicialização deve estabelecer o invariante:

$$H_p, H_i, H_c \vdash [S]Inv,$$

onde:

- ▶ H_p são as restrições sobre os parâmetros da máquina.
- ▶ H_i são as definições importadas.
- ▶ H_c são as restrições sobre as constantes da máquina
- ▶ S é a substituição da inicialização, e
- ▶ Inv é o invariante.

Example

Máquina *BirthdayAgenda*:

$$\begin{aligned} &\vdash [known, birthday := \emptyset, \emptyset] \\ &\quad known \subseteq NAME \wedge birthday \in known \rightarrow DATE \\ \equiv &\vdash (\emptyset \subseteq NAME) \wedge (\emptyset \in \emptyset \rightarrow DATE) \end{aligned}$$

Obrigações de prova para a inicialização

A inicialização deve estabelecer o invariante:

$$H_p, H_i, H_c \vdash [S]Inv,$$

onde:

- ▶ H_p são as restrições sobre os parâmetros da máquina.
- ▶ H_i são as definições importadas.
- ▶ H_c são as restrições sobre as constantes da máquina
- ▶ S é a substituição da inicialização, e
- ▶ Inv é o invariante.

Example

Máquina *BirthdayAgenda*:

$$\begin{aligned} & \vdash [known, birthday := \emptyset, \emptyset] \\ & \quad known \subseteq NAME \wedge birthday \in known \rightarrow DATE \\ \equiv & \vdash (\emptyset \subseteq NAME) \wedge (\emptyset \in \emptyset \rightarrow DATE) \\ \equiv & \vdash \emptyset \subseteq NAME, \\ & \vdash \emptyset \in \emptyset \rightarrow DATE \end{aligned}$$

Obrigações de prova para as operações

As operações devem **preservar** o invariante:

$$H_p, H_i, H_c, Inv, P \vdash [S]Inv,$$

onde

- ▶ P é a pré-condição da operação, e
- ▶ S é a substituição da operação.

Example

INVARIANT $known \subseteq NAME \wedge birthday \in known \rightarrow DATE$

$Register(n, d) \hat{=}$

PRE $n \in NAME \wedge d \in DATE \wedge n \notin known$

THEN

$known := known \cup \{n\} \parallel birthday := birthday \cup \{n \mapsto d\}$

END

$Inv,$
 P
 $\vdash [S]$
 Inv

Example

INVARIANT $known \subseteq NAME \wedge birthday \in known \rightarrow DATE$

Register (n, d) $\hat{=}$

PRE $n \in NAME \wedge d \in DATE \wedge n \notin known$

THEN

$known := known \cup \{n\} \parallel birthday := birthday \cup \{n \mapsto d\}$

END

$known \subseteq NAME, birthday \in known \rightarrow DATE,$

P

$\vdash [S]$

Inv

Example

INVARIANT $known \subseteq NAME \wedge birthday \in known \rightarrow DATE$

Register (n, d) $\hat{=}$

PRE $n \in NAME \wedge d \in DATE \wedge n \notin known$

THEN

$known := known \cup \{n\} \parallel birthday := birthday \cup \{n \mapsto d\}$

END

$known \subseteq NAME, birthday \in known \rightarrow DATE,$

$n \in NAME, d \in DATE, n \notin known$

$\vdash [S]$

Inv

Example

INVARIANT $known \subseteq NAME \wedge birthday \in known \rightarrow DATE$

Register (n, d) $\hat{=}$

PRE $n \in NAME \wedge d \in DATE \wedge n \notin known$

THEN

$known := known \cup \{n\} \parallel birthday := birthday \cup \{n \mapsto d\}$

END

$known \subseteq NAME, birthday \in known \rightarrow DATE,$

$n \in NAME, d \in DATE, n \notin known$

$\vdash [known := known \cup \{n\} \parallel birthday := birthday \cup \{n \mapsto d\}]$

Inv

Example

INVARIANT $known \subseteq NAME \wedge birthday \in known \rightarrow DATE$

Register (n, d) $\hat{=}$

PRE $n \in NAME \wedge d \in DATE \wedge n \notin known$

THEN

$known := known \cup \{n\} \parallel birthday := birthday \cup \{n \mapsto d\}$

END

$known \subseteq NAME, birthday \in known \rightarrow DATE,$

$n \in NAME, d \in DATE, n \notin known$

$\vdash [known := known \cup \{n\} \parallel birthday := birthday \cup \{n \mapsto d\}]$

$known \subseteq NAME \wedge birthday \in known \rightarrow DATE$

Example

INVARIANT $known \subseteq NAME \wedge birthday \in known \rightarrow DATE$

Register (n, d) $\hat{=}$

PRE $n \in NAME \wedge d \in DATE \wedge n \notin known$

THEN

$known := known \cup \{n\} \parallel birthday := birthday \cup \{n \mapsto d\}$

END

$known \subseteq NAME, birthday \in known \rightarrow DATE,$

$n \in NAME, d \in DATE, n \notin known$

$\vdash [known := known \cup \{n\} \parallel birthday := birthday \cup \{n \mapsto d\}]$

$known \subseteq NAME \wedge birthday \in known \rightarrow DATE$

Example

INVARIANT $known \subseteq NAME \wedge birthday \in known \rightarrow DATE$

$Register(n, d) \hat{=}$

PRE $n \in NAME \wedge d \in DATE \wedge n \notin known$

THEN

$known := known \cup \{n\} \parallel birthday := birthday \cup \{n \mapsto d\}$

END

$known \subseteq NAME, birthday \in known \rightarrow DATE,$
 $n \in NAME, d \in DATE, n \notin known$

$\vdash [known := known \cup \{n\} \parallel birthday := birthday \cup \{n \mapsto d\}]$
 $known \subseteq NAME \wedge birthday \in known \rightarrow DATE$

\Leftrightarrow $known \subseteq NAME, birthday \in known \rightarrow DATE,$
 $n \in NAME, d \in DATE, n \notin known$

$\vdash known \cup \{n\} \subseteq NAME \wedge$
 $birthday \cup \{n \mapsto d\} \in known \cup \{n\} \rightarrow DATE$

Example (Proof sketch)

$known \subseteq NAME,$

$birthday \in known \rightarrow DATE,$

$n \in NAME,$

$d \in DATE,$

$n \notin known \vdash$

$known \cup \{n\} \subseteq NAME \quad \wedge$

$birthday \cup \{n \mapsto d\} \in known \cup \{n\} \rightarrow DATE$

Example (Proof sketch)

$known \subseteq NAME,$

$birthday \in known \rightarrow DATE,$

$n \in NAME,$

$d \in DATE,$

$n \notin known \vdash$

$known \cup \{n\} \subseteq NAME \quad \wedge$

$birthday \cup \{n \mapsto d\} \in known \cup \{n\} \rightarrow DATE$

Example (Proof sketch)

$known \subseteq NAME,$
 $birthday \in known \rightarrow DATE,$
 $n \in NAME,$
 $d \in DATE,$
 $n \notin known$

$\vdash known \cup \{n\} \subseteq NAME$

$known \subseteq NAME,$
 $birthday \in known \rightarrow DATE,$
 $n \in NAME,$
 $d \in DATE,$
 $n \notin known$

$\vdash birthday \cup \{n \mapsto d\} \in known \cup \{n\} \rightarrow DATE$

Example (Proof sketch)

$known \subseteq NAME,$
 $birthday \in known \rightarrow DATE,$
 $n \in NAME,$
 $d \in DATE,$
 $n \notin known$

$\vdash known \cup \{n\} \subseteq NAME$

$known \subseteq NAME,$
 $birthday \in known \rightarrow DATE,$
 $n \in NAME,$
 $d \in DATE,$
 $n \notin known$

$\vdash birthday \cup \{n \mapsto d\} \in known \cup \{n\} \rightarrow DATE$

Example (Proof sketch)

$known \subseteq NAME,$
 $birthday \in known \rightarrow DATE,$
 $n \in NAME,$
 $d \in DATE,$
 $n \notin known$
 $\vdash known \cup \{n\} \subseteq NAME$

$known \subseteq NAME,$
 $birthday \in known \rightarrow DATE,$
 $n \in NAME,$
 $d \in DATE,$
 $n \notin known$
 $\vdash birthday \cup \{n \mapsto d\} \in known \cup \{n\} \rightarrow DATE$

Outline

Introdução

Contextualização

Visão global do método B

(tira-gosto)

Especificação dos requisitos funcionais

Uma notação para especificar

Verificação da especificação

Substituições generalizadas e obrigações de prova

Refinamento

Uma notação para refinar

Verificação de refinamentos

Implementação

Introdução

Uma linguagem para implementar

Ferramentas

Ferramentas industriais

Complementos

Componentes da notação B

- ▶ Lógica de primeira ordem ;
- ▶ Teoria dos conjuntos ;
- ▶ Aritmética inteira ;
- ▶ Substituições generalizadas .

Componentes da notação B

- ▶ Lógica de primeira ordem (parecido com Z);
- ▶ Teoria dos conjuntos (parecido com Z);
- ▶ Aritmética inteira (parecido com Z);
- ▶ Substituições generalizadas (**específico**).

Substituições generalizadas

O que são elas?

- ▶ Meio de descrever mudanças de estado;
- ▶ Algumas são sintaticamente similares a construções de programação imperativas;
- ▶ **Transformador de predicados** (condições, fórmulas)

Substituições generalizadas

O que são elas?

- ▶ Meio de descrever mudanças de estado;
- ▶ Algumas são sintaticamente similares a construções de programação imperativas;
- ▶ **Transformador de predicados** (condições, fórmulas)

Example

A substituição $[V := V+1]$ realiza a seguinte transformação: todas as ocorrências de V são substituídas pela expressão $V+1$, por exemplo

$$[V := V+1] V > 0 \equiv V+1 > 0$$

A condição $V > 0$ foi **transformado** em outra condição: $V+1 > 0$.

- ▶ Edsger W. Dijkstra, Guarded commands, nondeterminacy and formal derivation of program. Communications of the ACM, 18(8):453–457, August 1975.

Substituições generalizadas

Semântica

- ▶ Semântica de pré-condição mais fraca
- ▶ A cada tipo de substituição é associado um transformador de predicado
- ▶ A transformação da condição P pela substituição S é denotada $[S]P$
- ▶ É a pré-condição mais geral para que, uma vez que a substituição S é aplicada a um estado que satisfaz esta condição, o estado resultante satisfaz P .
- ▶ Ler $[S]P$ como “a condição mais geral tal que S estabelece P ”.

Substituição simples

$$[V := E]P \equiv P[E/V]$$

Example

$$[v := v + 1]v > 0 \equiv (v + 1) > 0 \equiv v \geq 0$$

Substituição múltipla

$$[V, W := E, F] P \equiv [tmp := F][V := E][W := tmp] P$$

onde *tmp* é uma nova variável.

Example

$$[V, W := W, V] V > W \equiv [tmp := V][V := W][W := tmp] V > W$$

Substituição múltipla

$$[V, W := E, F] P \equiv [tmp := F][V := E][W := tmp]P$$

onde tmp é uma nova variável.

Example

$$[V, W := W, V] V > W \equiv [tmp := V][V := W][W := tmp] V > W$$

Substituição múltipla

$$[V, W := E, F] P \equiv [tmp := F][V := E][W := tmp]P$$

onde tmp é uma nova variável.

Example

$$\begin{aligned} [V, W := W, V] V > W &\equiv [tmp := V][V := W][W := tmp] V > W \\ &\equiv [tmp := V][V := W] V > tmp \end{aligned}$$

Substituição múltipla

$$[V, W := E, F] P \equiv [tmp := F][V := E][W := tmp]P$$

onde tmp é uma nova variável.

Example

$$\begin{aligned} [V, W := W, V] V > W &\equiv [tmp := V][V := W][W := tmp] V > W \\ &\equiv [tmp := V][V := W] V > tmp \end{aligned}$$

Substituição múltipla

$$[V, W := E, F] P \equiv [tmp := F][V := E][W := tmp]P$$

onde tmp é uma nova variável.

Example

$$\begin{aligned} [V, W := W, V] V > W &\equiv [tmp := V][V := W][W := tmp] V > W \\ &\equiv [tmp := V][V := W] V > tmp \\ &\equiv [tmp := V] W > tmp \end{aligned}$$

Substituição múltipla

$$[V, W := E, F] P \equiv [tmp := F][V := E][W := tmp]P$$

onde *tmp* é uma nova variável.

Example

$$\begin{aligned} [V, W := W, V] V > W &\equiv [tmp := V][V := W][W := tmp] V > W \\ &\equiv [tmp := V][V := W] V > tmp \\ &\equiv [tmp := V] W > tmp \end{aligned}$$

Substituição múltipla

$$[V, W := E, F] P \equiv [tmp := F][V := E][W := tmp] P$$

onde tmp é uma nova variável.

Example

$$\begin{aligned} [V, W := W, V] V > W &\equiv [tmp := V][V := W][W := tmp] V > W \\ &\equiv [tmp := V][V := W] V > tmp \\ &\equiv [tmp := V] W > tmp \\ &\equiv W > V \end{aligned}$$

Substituição: faça elemento

$$\begin{aligned} & [x : \in S]P \\ \equiv & (\forall x \cdot x \in S \Rightarrow P) \end{aligned}$$

Example

$$[val : \in \{3, 5, 7\}] 0 \leq val \wedge val \leq 10$$

Substituição: faça elemento

$$\begin{aligned} & [x : \in S]P \\ \equiv & (\forall x \cdot x \in S \Rightarrow P) \end{aligned}$$

Example

$$\begin{aligned} & [val : \in \{3, 5, 7\}]0 \leq val \wedge val \leq 10 \\ \equiv & \forall val \cdot val \in \{3, 5, 7\} \Rightarrow 0 \leq val \wedge val \leq 10 \end{aligned}$$

Substituição: escolha limitada

$$\begin{aligned} & [\mathbf{CHOICE } S_1 \mathbf{ OR } S_2 \mathbf{ END}]P \\ \equiv & [S_1]P \wedge [S_2]P \end{aligned}$$

Example

$$[\mathbf{CHOICE } xx := 0 \mathbf{ OR } yy := 0 \mathbf{ END}]xx \times yy = 0$$

Substituição: escolha limitada

$$\begin{aligned} & [\mathbf{CHOICE } S_1 \mathbf{ OR } S_2 \mathbf{ END}]P \\ \equiv & [S_1]P \wedge [S_2]P \end{aligned}$$

Example

$$\begin{aligned} & [\mathbf{CHOICE } xx := 0 \mathbf{ OR } yy := 0 \mathbf{ END}]xx \times yy = 0 \\ \equiv & [xx := 0]xx \times yy = 0 \wedge [yy := 0]xx \times yy = 0 \end{aligned}$$

Substituição: escolha limitada

$$\begin{aligned} & [\mathbf{CHOICE } S_1 \mathbf{ OR } S_2 \mathbf{ END}]P \\ \equiv & [S_1]P \wedge [S_2]P \end{aligned}$$

Example

$$\begin{aligned} & [\mathbf{CHOICE } xx := 0 \mathbf{ OR } yy := 0 \mathbf{ END}]xx \times yy = 0 \\ \equiv & [xx := 0]xx \times yy = 0 \wedge [yy := 0]xx \times yy = 0 \\ \equiv & 0 \times yy = 0 \wedge xx \times 0 = 0 \end{aligned}$$

Substituição: escolha ilimitada

$$\begin{aligned} & [\mathbf{ANY } v \mathbf{ WHERE } G \mathbf{ THEN } S \mathbf{ END}]P \\ \equiv & \forall v \cdot G \Rightarrow [S]P \end{aligned}$$

Example

$$\begin{aligned} & [\mathbf{ANY } inc \mathbf{ WHERE } 0 < inc \wedge inc + val \leq MAXINT \\ & \quad \mathbf{THEN } val := inc + val \mathbf{ END}] \\ & \quad 0 \leq val \wedge val \leq MAXINT \\ \equiv & \forall inc \cdot 0 < inc \wedge inc + val \leq MAXINT \\ & \quad \Rightarrow [val := inc + val] 0 \leq val \wedge val \leq MAXINT \\ \equiv & \forall inc \cdot 0 < inc \wedge inc + val \leq MAXINT \\ & \quad \Rightarrow 0 \leq inc + val \wedge inc + val \leq MAXINT \end{aligned}$$

Substituição vácuca

$$[\mathbf{SKIP}]P \equiv P$$

Substituição com pré-condição

$$[\mathbf{PRE} \ C \ \mathbf{THEN} \ S \ \mathbf{END}]P \equiv C \wedge ([S]P)$$

Outline

Introdução

Contextualização

Visão global do método B

(tira-gosto)

Especificação dos requisitos funcionais

Uma notação para especificar

Verificação da especificação

Substituições generalizadas e obrigações de prova

Refinamento

Uma notação para refinar

Verificação de refinamentos

Implementação

Introdução

Uma linguagem para implementar

Ferramentas

Ferramentas industriais

Complementos

Refinamento

- ▶ é uma etapa na construção de uma implementação executável da especificação;
 - horizontal** enriquece a especificação com requisitos funcionais adicionais;
 - vertical** reflete uma decisão de projeto;
- ▶ versão concreta das variáveis, inicialização e operações da especificação;
- ▶ **relação de refinamento**: relaciona valores das variáveis concretas e abstratas.
- ▶ operações: mesma interface, mesmo comportamento observável.

Os refinamentos verticais

- ▶ Refinamento de dados – codificação;
- ▶ Refinamento de operações – algoritmos.
 - ▶ + entradas (enfraquece a pré-condição);
 - ▶ - não determinismo (fortalece a pós-condição).

Estrutura dos refinamentos

- ▶ **REFINEMENT**;
- ▶ **REFINES** uma máquina ou outro refinamento;
- ▶ **INVARIANT** tipo das variáveis e a relação de refinamento (invariante de colagem);
- ▶ mesmas operações, com a mesma assinatura.

Exemplo: a máquina [1]

MACHINE *ExampleM*

VARIABLES y

INVARIANT $y \in \mathbb{F}(\mathbb{N}_1)$

INITIALISATION $y := \emptyset$

OPERATIONS

$enter(n) = \mathbf{PRE} \ n \in \mathbb{N}_1 \ \mathbf{THEN} \ y := y \cup \{n\} \ \mathbf{END};$

$m \leftarrow getmax = \mathbf{PRE} \ y \neq \emptyset \ \mathbf{THEN} \ m := \max(y) \ \mathbf{END}$

END

Exemplo: o refinamento [1]

REFINEMENT *ExampleR*

REFINES *ExampleM*

VARIABLES z

INVARIANT $z \in \mathbb{N} \wedge z = \max(y \cup \{0\})$

INITIALISATION $z := 0$

OPERATIONS

$enter(n) = \mathbf{PRE} \ n \in \mathbb{N}_1 \ \mathbf{THEN} \ z := \max(\{z, n\}) \ \mathbf{END};$

$m \leftarrow getmax = \mathbf{PRE} \ z \neq 0 \ \mathbf{THEN} \ m := z \ \mathbf{END}$

END

Substituições para refinamentos

- ▶ sequenciamento ;
- ▶ bloco **BEGIN . . . END**
- ▶ condicional **IF**
- ▶ condicional **CASE**
- ▶ variável local
- ▶ chamada de operações

Semântica (sequenciamento)

$$[S1; S2]P \equiv [S1]([S2]P)$$

Semântica (sequenciamento)

$$[S1; S2]P \equiv [S1]([S2]P)$$

Example

$$[x := x + 1; x := x * 2]x = 2$$

Semântica (sequenciamento)

$$[S1; S2]P \equiv [S1]([S2]P)$$

Example

$$\begin{aligned} & [x := x + 1; x := x * 2]x = 2 \\ \equiv & [x := x + 1][x := x * 2]x = 2 \end{aligned}$$

Semântica (sequenciamento)

$$[S1; S2]P \equiv [S1]([S2]P)$$

Example

$$\begin{aligned} & [x := x + 1; x := x * 2]x = 2 \\ \equiv & [x := x + 1][x := x * 2]x = 2 \\ \equiv & [x := x + 1]x * 2 = 2 \end{aligned}$$

Semântica (sequenciamento)

$$[S1; S2]P \equiv [S1]([S2]P)$$

Example

$$\begin{aligned} & [x := x + 1; x := x * 2]x = 2 \\ \equiv & [x := x + 1][x := x * 2]x = 2 \\ \equiv & [x := x + 1]x * 2 = 2 \\ \equiv & [x := x + 1]x = 1 \end{aligned}$$

Semântica (sequenciamento)

$$[S1; S2]P \equiv [S1]([S2]P)$$

Example

$$\begin{aligned} & [x := x + 1; x := x * 2]x = 2 \\ \equiv & [x := x + 1][x := x * 2]x = 2 \\ \equiv & [x := x + 1]x * 2 = 2 \\ \equiv & [x := x + 1]x = 1 \\ \equiv & x + 1 = 1 \end{aligned}$$

Semântica (sequenciamento)

$$[S1; S2]P \equiv [S1]([S2]P)$$

Example

$$\begin{aligned} & [x := x + 1; x := x * 2]x = 2 \\ \equiv & [x := x + 1][x := x * 2]x = 2 \\ \equiv & [x := x + 1]x * 2 = 2 \\ \equiv & [x := x + 1]x = 1 \\ \equiv & x + 1 = 1 \\ \equiv & x = 0 \end{aligned}$$

Substituição condicional

$$\begin{aligned} & \mathbf{[IF } C \mathbf{ THEN } S_1 \mathbf{ ELSE } S_2 \mathbf{ END]}P \\ & \equiv (C \Rightarrow [S_1]P) \wedge (\neg C \Rightarrow [S_2]P) \end{aligned}$$

Example

$$\begin{aligned} & \mathbf{[IF } V \bmod 2 = 0 \mathbf{ THEN } V := V/2 \mathbf{ ELSE } V := 3 * V + 1 \mathbf{ END]} \\ & V \geq 0 \\ & \equiv (V \bmod 2 = 0) \Rightarrow ([V := V/2]V \geq 0) \\ & \wedge (\neg(V \bmod 2 = 0) \Rightarrow [V := 3 * V + 1]V \geq 0) \end{aligned}$$

Substituição condicional

$$\begin{aligned} & \mathbf{[IF } C \mathbf{ THEN } S_1 \mathbf{ ELSE } S_2 \mathbf{ END]}P \\ & \equiv (C \Rightarrow [S_1]P) \wedge (\neg C \Rightarrow [S_2]P) \end{aligned}$$

Example

$$\begin{aligned} & \mathbf{[IF } V \bmod 2 = 0 \mathbf{ THEN } V := V/2 \mathbf{ ELSE } V := 3 * V + 1 \mathbf{ END]} \\ & V \geq 0 \\ & \equiv (V \bmod 2 = 0) \Rightarrow ([V := V/2]V \geq 0) \\ & \quad \wedge (\neg(V \bmod 2 = 0) \Rightarrow [V := 3 * V + 1]V \geq 0) \end{aligned}$$

Substituição condicional

$$\begin{aligned} & \mathbf{[IF } C \mathbf{ THEN } S_1 \mathbf{ ELSE } S_2 \mathbf{ END]}P \\ & \equiv (C \Rightarrow [S_1]P) \wedge (\neg C \Rightarrow [S_2]P) \end{aligned}$$

Example

$$\begin{aligned} & \mathbf{[IF } V \bmod 2 = 0 \mathbf{ THEN } V := V/2 \mathbf{ ELSE } V := 3 * V + 1 \mathbf{ END]} \\ & V \geq 0 \\ & \equiv (V \bmod 2 = 0) \Rightarrow (V/2 \geq 0) \\ & \wedge (\neg(V \bmod 2 = 0) \Rightarrow [V := 3 * V + 1]V \geq 0) \end{aligned}$$

Substituição condicional

$$\begin{aligned} & \mathbf{[IF } C \mathbf{ THEN } S_1 \mathbf{ ELSE } S_2 \mathbf{ END]}P \\ & \equiv (C \Rightarrow [S_1]P) \wedge (\neg C \Rightarrow [S_2]P) \end{aligned}$$

Example

$$\begin{aligned} & \mathbf{[IF } V \bmod 2 = 0 \mathbf{ THEN } V := V/2 \mathbf{ ELSE } V := 3 * V + 1 \mathbf{ END]} \\ & V \geq 0 \\ & \equiv (V \bmod 2 = 0) \Rightarrow (V/2 \geq 0) \\ & \wedge (\neg(V \bmod 2 = 0) \Rightarrow [V := 3 * V + 1]V \geq 0) \end{aligned}$$

Substituição condicional

$$\begin{aligned} & \mathbf{[IF } C \mathbf{ THEN } S_1 \mathbf{ ELSE } S_2 \mathbf{ END]}P \\ & \equiv (C \Rightarrow [S_1]P) \wedge (\neg C \Rightarrow [S_2]P) \end{aligned}$$

Example

$$\begin{aligned} & \mathbf{[IF } V \bmod 2 = 0 \mathbf{ THEN } V := V/2 \mathbf{ ELSE } V := 3 * V + 1 \mathbf{ END]} \\ & V \geq 0 \\ & \equiv (V \bmod 2 = 0) \Rightarrow (V/2 \geq 0) \\ & \quad \wedge (\neg(V \bmod 2 = 0) \Rightarrow (3 * V + 1 \geq 0)) \end{aligned}$$

Outline

Introdução

Contextualização

Visão global do método B

(tira-gosto)

Especificação dos requisitos funcionais

Uma notação para especificar

Verificação da especificação

Substituições generalizadas e obrigações de prova

Refinamento

Uma notação para refinar

Verificação de refinamentos

Implementação

Introdução

Uma linguagem para implementar

Ferramentas

Ferramentas industriais

Complementos

Obrigações de prova

- ▶ O refinamento tem que ser “compatível” com a especificação;
 - ▶ refinamento de execuções (*trace refinement*)
- ▶ **inicialização**: $\mathbf{INIT}_R \rightsquigarrow \mathbf{INIT}_M$;
- ▶ **operações**: $\mathbf{OP}_R \rightsquigarrow \mathbf{OP}_M$.
- ▶ “compatível”:
 - ▶ **pré-condição** abstrata \Rightarrow concreta;
 - ▶ **substituição** resulte em valores concretos compatíveis com os abstratos, como definido na relação \mathbf{INV}_R .
 - ▶ **pós-condição** concreta \Rightarrow abstrata.

Refinamento, verificação e inicialização

- ▶ Especificação
 - ▶ INV_M : invariante da especificação;
 - ▶ $INIT_M$: inicialização;
 - ▶ consistência: $[INIT_M]INV_M$.
- ▶ INV_R : relação de refinamento;
 - ▶ $\neg INV_R$: relação de incompatibilidade entre estados abstratos e concretos;
- ▶ $[INIT_M]\neg INV_R$: estados iniciais concretos incompatíveis com estado inicial abstrato.
- ▶ Obrigação de prova:

$$[INIT_R]\neg[INIT_M]\neg INV_R.$$

Nenhum estado inicial concreto é incompatível com um estado inicial abstrato.

Refinamento, verificação e inicialização: um exemplo

$[\text{INIT}_R] \neg [\text{INIT}_M] \neg \text{INV}_R$

$\Leftrightarrow [z := 0] \neg [y := \emptyset] \neg (z \in \mathbb{N} \wedge z = \mathbf{max}(y \cup \{0\}))$

Refinamento, verificação e inicialização: um exemplo

$[\text{INIT}_R] \neg [\text{INIT}_M] \neg \text{INV}_R$

$\Leftrightarrow [z := 0] \neg [y := \emptyset] \neg (z \in \mathbb{N} \wedge z = \mathbf{max}(y \cup \{0\}))$

Refinamento, verificação e inicialização: um exemplo

$$[\mathbf{INIT}_R] \neg [\mathbf{INIT}_M] \neg \mathbf{INV}_R$$

$$\Leftrightarrow [z := 0] \neg [y := \emptyset] \neg (z \in \mathbb{N} \wedge z = \mathbf{max}(y \cup \{0\}))$$

$$\Leftrightarrow [z := 0] \neg \neg (z \in \mathbb{N} \wedge z = \mathbf{max}(\emptyset \cup \{0\}))$$

Refinamento, verificação e inicialização: um exemplo

$$[\text{INIT}_R] \neg [\text{INIT}_M] \neg \text{INV}_R$$

$$\Leftrightarrow [z := 0] \neg [y := \emptyset] \neg (z \in \mathbb{N} \wedge z = \mathbf{max}(y \cup \{0\}))$$

$$\Leftrightarrow [z := 0] \neg \neg (z \in \mathbb{N} \wedge z = \mathbf{max}(\emptyset \cup \{0\}))$$

Refinamento, verificação e inicialização: um exemplo

$$[\mathbf{INIT}_R] \neg [\mathbf{INIT}_M] \neg \mathbf{INV}_R$$

$$\Leftrightarrow [z := 0] \neg [y := \emptyset] \neg (z \in \mathbb{N} \wedge z = \mathbf{max}(y \cup \{0\}))$$

$$\Leftrightarrow [z := 0] \neg \neg (z \in \mathbb{N} \wedge z = \mathbf{max}(\emptyset \cup \{0\}))$$

$$\Leftrightarrow [z := 0] z \in \mathbb{N} \wedge z = \mathbf{max}(\{0\})$$

Refinamento, verificação e inicialização: um exemplo

$$[\mathbf{INIT}_R] \neg [\mathbf{INIT}_M] \neg \mathbf{INV}_R$$

$$\Leftrightarrow [z := 0] \neg [y := \emptyset] \neg (z \in \mathbb{N} \wedge z = \mathbf{max}(y \cup \{0\}))$$

$$\Leftrightarrow [z := 0] \neg \neg (z \in \mathbb{N} \wedge z = \mathbf{max}(\emptyset \cup \{0\}))$$

$$\Leftrightarrow [z := 0] z \in \mathbb{N} \wedge z = \mathbf{max}(\{0\})$$

Refinamento, verificação e inicialização: um exemplo

$$[\text{INIT}_R] \neg [\text{INIT}_M] \neg \text{INV}_R$$

$$\Leftrightarrow [z := 0] \neg [y := \emptyset] \neg (z \in \mathbb{N} \wedge z = \mathbf{max}(y \cup \{0\}))$$

$$\Leftrightarrow [z := 0] \neg \neg (z \in \mathbb{N} \wedge z = \mathbf{max}(\emptyset \cup \{0\}))$$

$$\Leftrightarrow [z := 0] z \in \mathbb{N} \wedge z = \mathbf{max}(\{0\})$$

$$\Leftrightarrow 0 \in \mathbb{N} \wedge 0 = \mathbf{max}(\{0\})$$

Refinamento, verificação e inicialização: um exemplo

$$[\text{INIT}_R] \neg [\text{INIT}_M] \neg \text{INV}_R$$

$$\Leftrightarrow [z := 0] \neg [y := \emptyset] \neg (z \in \mathbb{N} \wedge z = \mathbf{max}(y \cup \{0\}))$$

$$\Leftrightarrow [z := 0] \neg \neg (z \in \mathbb{N} \wedge z = \mathbf{max}(\emptyset \cup \{0\}))$$

$$\Leftrightarrow [z := 0] z \in \mathbb{N} \wedge z = \mathbf{max}(\{0\})$$

$$\Leftrightarrow 0 \in \mathbb{N} \wedge \mathbf{0} = \mathbf{max}(\{0\})$$

Using: $\mathbf{max}(\{n\}) = n$.

Refinamento, verificação e inicialização: um exemplo

$$[\text{INIT}_R] \neg [\text{INIT}_M] \neg \text{INV}_R$$

$$\Leftrightarrow [z := 0] \neg [y := \emptyset] \neg (z \in \mathbb{N} \wedge z = \mathbf{max}(y \cup \{0\}))$$

$$\Leftrightarrow [z := 0] \neg \neg (z \in \mathbb{N} \wedge z = \mathbf{max}(\emptyset \cup \{0\}))$$

$$\Leftrightarrow [z := 0] z \in \mathbb{N} \wedge z = \mathbf{max}(\{0\})$$

$$\Leftrightarrow 0 \in \mathbb{N} \wedge 0 = \mathbf{max}(\{0\})$$

$$\Leftrightarrow 0 \in \mathbb{N} \wedge 0 = 0$$

Refinamento, verificação e inicialização: um exemplo

$$[\text{INIT}_R] \neg [\text{INIT}_M] \neg \text{INV}_R$$

$$\Leftrightarrow [z := 0] \neg [y := \emptyset] \neg (z \in \mathbb{N} \wedge z = \mathbf{max}(y \cup \{0\}))$$

$$\Leftrightarrow [z := 0] \neg \neg (z \in \mathbb{N} \wedge z = \mathbf{max}(\emptyset \cup \{0\}))$$

$$\Leftrightarrow [z := 0] z \in \mathbb{N} \wedge z = \mathbf{max}(\{0\})$$

$$\Leftrightarrow 0 \in \mathbb{N} \wedge 0 = \mathbf{max}(\{0\})$$

$$\Leftrightarrow 0 \in \mathbb{N} \wedge 0 = 0$$

$$\Leftrightarrow \top$$

Refinamento, verificação, operações sem saídas

$$\begin{aligned} \mathbf{INV}_M, \mathbf{INV}_R, \mathbf{PRE}_M &\vdash \mathbf{PRE}_R \\ \mathbf{INV}_M, \mathbf{INV}_R, \mathbf{PRE}_M &\vdash [\mathbf{OP}_R] \neg [\mathbf{OP}_M] \neg \mathbf{INV}_R \end{aligned}$$

onde

- ▶ **PRE_M** e **OP_M** pré-condição e substituição da especificação da operação
- ▶ **PRE_R** and **OP_R** pré-condição e substituição do refinamento da operação

Refinamento, verificação, operações sem saídas: exemplo

$INV_M, INV_R, PRE_M \vdash PRE_R$

$\iff y \in \mathbb{F}(\mathbb{N}_1), z \in \mathbb{N}, z = \mathbf{max}(y \cup \{0\}), n \in \mathbb{N}_1 \vdash n \in \mathbb{N}_1$

Refinamento, verificação, operações sem saídas: exemplo

$INV_M, INV_R, PRE_M \vdash PRE_R$

$\iff y \in \mathbb{F}(\mathbb{N}_1), z \in \mathbb{N}, z = \mathbf{max}(y \cup \{0\}), n \in \mathbb{N}_1 \vdash n \in \mathbb{N}_1$

$\iff \top$

Refinamento, verificação, operações sem saídas: exemplo

$INV_M, INV_R, PRE_M \vdash [OP_R] \neg [OP_M] \neg INV_R$

$$\begin{aligned} \iff & y \in \mathbb{F}(\mathbb{N}_1), z \in \mathbb{N}, z = \mathbf{max}(y \cup \{0\}), n \in \mathbb{N}_1 \vdash \\ & [z := \mathbf{max}(\{z, n\})] \neg [y := y \cup \{n\}] \\ & \neg(z \in \mathbb{N} \wedge z = \mathbf{max}(y \cup \{0\})) \end{aligned}$$

Refinamento, verificação, operações sem saídas: exemplo

$INV_M, INV_R, PRE_M \vdash [OP_R] \neg [OP_M] \neg INV_R$

$\iff y \in \mathbb{F}(\mathbb{N}_1), z \in \mathbb{N}, z = \mathbf{max}(y \cup \{0\}), n \in \mathbb{N}_1 \vdash$
 $[z := \mathbf{max}(\{z, n\})] \neg [y := y \cup \{n\}]$
 $\neg(z \in \mathbb{N} \wedge z = \mathbf{max}(y \cup \{0\}))$

Refinamento, verificação, operações sem saídas: exemplo

$INV_M, INV_R, PRE_M \Rightarrow [OP_R] \neg [OP_M] \neg INV_R$

$$\begin{aligned} \iff & y \in \mathbb{F}(\mathbb{N}_1), z \in \mathbb{N}, z = \mathbf{max}(y \cup \{0\}), n \in \mathbb{N}_1 \\ \vdash & [z := \mathbf{max}(\{z, n\})] \\ & \neg \neg (z \in \mathbb{N} \wedge z = \mathbf{max}(y \cup \{n\} \cup \{0\})) \end{aligned}$$

Refinamento, verificação, operações sem saídas: exemplo

$$\mathbf{INV}_M, \mathbf{INV}_R, \mathbf{PRE}_M \Rightarrow [\mathbf{OP}_R] \neg [\mathbf{OP}_M] \neg \mathbf{INV}_R$$

$$\begin{aligned} \iff & y \in \mathbb{F}(\mathbb{N}_1), z \in \mathbb{N}, z = \mathbf{max}(y \cup \{0\}), n \in \mathbb{N}_1 \\ \vdash & [z := \mathbf{max}(\{z, n\})] \\ & \neg \neg (z \in \mathbb{N} \wedge z = \mathbf{max}(y \cup \{n\} \cup \{0\})) \end{aligned}$$

Refinamento, verificação, operações sem saídas: exemplo

$\mathbf{INV}_M, \mathbf{INV}_R, \mathbf{PRE}_M \Rightarrow [\mathbf{OP}_R] \neg [\mathbf{OP}_M] \neg \mathbf{INV}_R$

$\iff y \in \mathbb{F}(\mathbb{N}_1), z \in \mathbb{N}, z = \mathbf{max}(y \cup \{0\}), n \in \mathbb{N}_1$
 $\vdash [z := \mathbf{max}(\{z, n\})]$
 $(z \in \mathbb{N} \wedge z = \mathbf{max}(y \cup \{n\} \cup \{0\}))$

Refinamento, verificação, operações sem saídas: exemplo

$INV_M, INV_R, PRE_M \Rightarrow [OP_R] \neg [OP_M] \neg INV_R$

$\iff y \in \mathbb{F}(\mathbb{N}_1), z \in \mathbb{N}, z = \mathbf{max}(y \cup \{0\}), n \in \mathbb{N}_1$
 $\vdash [z := \mathbf{max}(\{z, n\})]$
 $(z \in \mathbb{N} \wedge z = \mathbf{max}(y \cup \{n\} \cup \{0\}))$

Refinamento, verificação, operações sem saídas: exemplo

$\mathbf{INV}_M, \mathbf{INV}_R, \mathbf{PRE}_M \vdash [\mathbf{OP}_R] \neg [\mathbf{OP}_M] \neg \mathbf{INV}_R$

$$\begin{aligned} \iff & y \in \mathbb{F}(\mathbb{N}_1), z \in \mathbb{N}, z = \mathbf{max}(y \cup \{0\}), n \in \mathbb{N}_1 \\ & \vdash \mathbf{max}(\{z, n\}) \in \mathbb{N} \wedge \\ & \mathbf{max}(\{z, n\}) = \mathbf{max}(y \cup \{n\} \cup \{0\}) \end{aligned}$$

Refinamento, verificação, operações sem saídas: exemplo

$INV_M, INV_R, PRE_M \vdash [OP_R] \neg [OP_M] \neg INV_R$

$$\begin{aligned} \iff & y \in \mathbb{F}(\mathbb{N}_1), z \in \mathbb{N}, z = \max(y \cup \{0\}), n \in \mathbb{N}_1 \\ & \vdash \max(\{z, n\}) \in \mathbb{N} \wedge \\ & \max(\{z, n\}) = \max(y \cup \{n\} \cup \{0\}) \end{aligned}$$

Refinamento, verificação, operações sem saídas: exemplo

$$\begin{aligned} & \mathbf{INV}_M, \mathbf{INV}_R, \mathbf{PRE}_M \Rightarrow [\mathbf{OP}_R] \neg [\mathbf{OP}_M] \neg \mathbf{INV}_R \\ \iff & \quad z \in \mathbb{N}, z = \mathbf{max}(y \cup \{0\}), n \in \mathbb{N}_1 \\ & \vdash \mathbf{max}(\{z, n\}) = \mathbf{max}(y \cup \{n\} \cup \{0\}) \end{aligned}$$

Refinamento, verificação, operações sem saídas: exemplo

$$\begin{aligned} \text{INV}_M, \text{INV}_R, \text{PRE}_M &\Rightarrow [\text{OP}_R] \neg [\text{OP}_M] \neg \text{INV}_R \\ \iff & z \in \mathbb{N}, z = \mathbf{\max}(y \cup \{0\}), n \in \mathbb{N}_1 \\ &\vdash \mathbf{\max}(\{z, n\}) = \mathbf{\max}(y \cup \{n\} \cup \{0\}) \end{aligned}$$

Utilizando: $\mathbf{\max}(\mathbf{\max}(S_1 \cup S_2) \cup S_3) = \mathbf{\max}(S_1 \cup S_2 \cup S_3)$.

Refinamento, verificação, operações sem saídas: exemplo

$\mathbf{INV_M, INV_R, PRE_M} \Rightarrow [\mathbf{OP_R}] \neg [\mathbf{OP_M}] \neg \mathbf{INV_R}$

$\iff z \in \mathbb{N}, z = \mathbf{max}(y \cup \{0\}), n \in \mathbb{N}_1$
 $\quad \vdash \mathbf{max}(\{z, n\}) = \mathbf{max}(y \cup \{n\} \cup \{0\})$

$\iff \top$

Refinamento, verificação, operações com saídas

Obrigações de prova:

$$\mathbf{INV_M, INV_R, PRE_M \vdash PRE_R}$$
$$\mathbf{INV_M, INV_R, PRE_M \vdash [[o := o']OP_R] \neg [OP_M] \neg (INV_R \wedge o = o')}$$

onde:

- ▶ \mathbf{o} é a saída, e \mathbf{o}' um novo identificador.

Refinamento, verificação, operações com saída

Exemplo

$$\mathbf{INV_M, INV_R, PRE_M \vdash PRE_R}$$

Refinamento, verificação, operações com saída

Exemplo

$$\begin{aligned} & \mathbf{INV_M, INV_R, PRE_M \vdash PRE_R} \\ \iff & y \in \mathbb{F}(\mathbb{N}_1), z \in \mathbb{N}, z = \mathbf{max}(y \cup \{0\}), y \neq \emptyset \vdash z \neq 0 \end{aligned}$$

Refinamento, verificação, operações com saída

Exemplo

$INV_M, INV_R, PRE_M \vdash PRE_R$

$\iff y \in \mathbb{F}(\mathbb{N}_1), z \in \mathbb{N}, z = \mathbf{max}(y \cup \{0\}), y \neq \emptyset \vdash z \neq 0$

$\iff \top$

Refinamento, verificação, operações com saída

Exemplo

$$\begin{aligned} & \mathbf{INV}_M, \mathbf{INV}_R, \mathbf{PRE}_M \vdash [[\mathbf{o} := \mathbf{o}']\mathbf{OP}_R] \neg [\mathbf{OP}_M] \neg \mathbf{INV}_R \wedge \mathbf{o} = \mathbf{o}' \\ \iff & y \in \mathbb{F}(\mathbb{N}_1), z \in \mathbb{N}, z = \mathbf{max}(y \cup \{0\}), y \neq \emptyset \\ & \vdash [[m := m]m := z] \neg ([m := \mathbf{max}(y)]) \\ & \quad \neg (z \in \mathbb{N} \wedge z = \mathbf{max}(y \cup \{0\}) \wedge m = m') \end{aligned}$$

Refinamento, verificação, operações com saída

Exemplo

$$\begin{aligned} & \mathbf{INV}_M, \mathbf{INV}_R, \mathbf{PRE}_M \vdash [[\mathbf{o} := \mathbf{o}']\mathbf{OP}_R] \neg [\mathbf{OP}_M] \neg \mathbf{INV}_R \wedge \mathbf{o} = \mathbf{o}' \\ \iff & y \in \mathbb{F}(\mathbb{N}_1), z \in \mathbb{N}, z = \mathbf{max}(y \cup \{0\}), y \neq \emptyset \\ & \vdash [[m := m]m := z] \neg ([m := \mathbf{max}(y)]) \\ & \quad \neg (z \in \mathbb{N} \wedge z = \mathbf{max}(y \cup \{0\}) \wedge m = m') \end{aligned}$$

Refinamento, verificação, operações com saída

Exemplo

$$\begin{aligned} & \mathbf{INV}_M, \mathbf{INV}_R, \mathbf{PRE}_M \vdash [[\mathbf{o} := \mathbf{o}']\mathbf{OP}_R] \neg [\mathbf{OP}_M] \neg \mathbf{INV}_R \wedge \mathbf{o} = \mathbf{o}' \\ \iff & y \in \mathbb{F}(\mathbb{N}_1), z \in \mathbb{N}, z = \mathbf{max}(y \cup \{0\}), y \neq \emptyset \\ & \vdash [[m := m']m := z] \\ & \neg \neg (z \in \mathbb{N} \wedge z = \mathbf{max}(y \cup \{0\}) \wedge \mathbf{max}(y) = m') \end{aligned}$$

Refinamento, verificação, operações com saída

Exemplo

$$\begin{aligned} & \mathbf{INV}_M, \mathbf{INV}_R, \mathbf{PRE}_M \vdash [[\mathbf{o} := \mathbf{o}']\mathbf{OP}_R] \neg [\mathbf{OP}_M] \neg \mathbf{INV}_R \wedge \mathbf{o} = \mathbf{o}' \\ \iff & y \in \mathbb{F}(\mathbb{N}_1), z \in \mathbb{N}, z = \mathbf{max}(y \cup \{0\}), y \neq \emptyset \\ & \vdash [[m := m']m := z] \\ & \neg \neg (z \in \mathbb{N} \wedge z = \mathbf{max}(y \cup \{0\}) \wedge \mathbf{max}(y) = m') \end{aligned}$$

Refinamento, verificação, operações com saída

Exemplo

$$\begin{aligned} & \mathbf{INV}_M, \mathbf{INV}_R, \mathbf{PRE}_M \vdash [[\mathbf{o} := \mathbf{o}']\mathbf{OP}_R] \neg [\mathbf{OP}_M] \neg \mathbf{INV}_R \wedge \mathbf{o} = \mathbf{o}' \\ \iff & y \in \mathbb{F}(\mathbb{N}_1), z \in \mathbb{N}, z = \mathbf{max}(y \cup \{0\}), y \neq \emptyset \\ & \vdash [[m := m']m := z] \\ & z \in \mathbb{N} \wedge z = \mathbf{max}(y \cup \{0\}) \wedge \mathbf{max}(y) = m' \end{aligned}$$

Refinamento, verificação, operações com saída

Exemplo

$$\begin{aligned} & \mathbf{INV}_M, \mathbf{INV}_R, \mathbf{PRE}_M \vdash [[\mathbf{o} := \mathbf{o}']\mathbf{OP}_R] \neg [\mathbf{OP}_M] \neg \mathbf{INV}_R \wedge \mathbf{o} = \mathbf{o}' \\ \iff & y \in \mathbb{F}(\mathbb{N}_1), z \in \mathbb{N}, z = \mathbf{max}(y \cup \{0\}), y \neq \emptyset \\ & \vdash [[m := m]m := z] \\ & z \in \mathbb{N} \wedge z = \mathbf{max}(y \cup \{0\}) \wedge \mathbf{max}(y) = m' \end{aligned}$$

Refinamento, verificação, operações com saída

Exemplo

$$\begin{aligned} & \mathbf{INV}_M, \mathbf{INV}_R, \mathbf{PRE}_M \vdash [[\mathbf{o} := \mathbf{o}']\mathbf{OP}_R] \neg [\mathbf{OP}_M] \neg \mathbf{INV}_R \wedge \mathbf{o} = \mathbf{o}' \\ \iff & y \in \mathbb{F}(\mathbb{N}_1), z \in \mathbb{N}, z = \mathbf{max}(y \cup \{0\}), y \neq \emptyset \\ & \vdash [m' := z]z \in \mathbb{N} \wedge z = \mathbf{max}(y \cup \{0\}) \wedge \mathbf{max}(y) = m' \end{aligned}$$

Refinamento, verificação, operações com saída

Exemplo

$$\begin{aligned} & \mathbf{INV}_M, \mathbf{INV}_R, \mathbf{PRE}_M \vdash [[\mathbf{o} := \mathbf{o}']\mathbf{OP}_R] \neg [\mathbf{OP}_M] \neg \mathbf{INV}_R \wedge \mathbf{o} = \mathbf{o}' \\ \iff & y \in \mathbb{F}(\mathbb{N}_1), z \in \mathbb{N}, z = \mathbf{max}(y \cup \{0\}), y \neq \emptyset \\ & \vdash [m' := z] z \in \mathbb{N} \wedge z = \mathbf{max}(y \cup \{0\}) \wedge \mathbf{max}(y) = m' \end{aligned}$$

Refinamento, verificação, operações com saída

Exemplo

$$\begin{aligned} & \mathbf{INV}_M, \mathbf{INV}_R, \mathbf{PRE}_M \vdash [[\mathbf{o} := \mathbf{o}']\mathbf{OP}_R] \neg [\mathbf{OP}_M] \neg \mathbf{INV}_R \wedge \mathbf{o} = \mathbf{o}' \\ \iff & y \in \mathbb{F}(\mathbb{N}_1), z \in \mathbb{N}, z = \mathbf{max}(y \cup \{0\}), y \neq \emptyset \\ & \vdash z \in \mathbb{N} \wedge z = \mathbf{max}(y \cup \{0\}) \wedge \mathbf{max}(y) = z \end{aligned}$$

Refinamento, verificação, operações com saída

Exemplo

$$\begin{aligned} & \mathbf{INV}_M, \mathbf{INV}_R, \mathbf{PRE}_M \vdash [[\mathbf{o} := \mathbf{o}']\mathbf{OP}_R] \neg [\mathbf{OP}_M] \neg \mathbf{INV}_R \wedge \mathbf{o} = \mathbf{o}' \\ \iff & y \in \mathbb{F}(\mathbb{N}_1), z \in \mathbb{N}, z = \mathbf{max}(y \cup \{0\}), y \neq \emptyset \\ & \vdash z \in \mathbb{N} \wedge z = \mathbf{max}(y \cup \{0\}) \wedge \mathbf{max}(y) = z \end{aligned}$$

Refinamento, verificação, operações com saída

Exemplo

$$\begin{aligned} & \mathbf{INV}_M, \mathbf{INV}_R, \mathbf{PRE}_M \vdash [[\mathbf{o} := \mathbf{o}']\mathbf{OP}_R] \neg [\mathbf{OP}_M] \neg \mathbf{INV}_R \wedge \mathbf{o} = \mathbf{o}' \\ \iff & y \in \mathbb{F}(\mathbb{N}_1), z \in \mathbb{N}, z = \mathbf{max}(y \cup \{0\}), y \neq \emptyset \vdash z \in \mathbb{N} \\ & y \in \mathbb{F}(\mathbb{N}_1), z \in \mathbb{N}, z = \mathbf{max}(y \cup \{0\}), y \neq \emptyset \vdash z = \mathbf{max}(y \cup \{0\}) \\ & y \in \mathbb{F}(\mathbb{N}_1), z \in \mathbb{N}, z = \mathbf{max}(y \cup \{0\}), y \neq \emptyset \vdash \mathbf{max}(y) = z \end{aligned}$$

Refinamento, verificação, operações com saída

Exemplo

$$\begin{aligned} & \mathbf{INV}_M, \mathbf{INV}_R, \mathbf{PRE}_M \vdash [[\mathbf{o} := \mathbf{o}']\mathbf{OP}_R] \neg [\mathbf{OP}_M] \neg \mathbf{INV}_R \wedge \mathbf{o} = \mathbf{o}' \\ \iff & y \in \mathbb{F}(\mathbb{N}_1), z \in \mathbb{N}, z = \mathbf{max}(y \cup \{0\}), y \neq \emptyset \vdash z \in \mathbb{N} \\ & y \in \mathbb{F}(\mathbb{N}_1), z \in \mathbb{N}, z = \mathbf{max}(y \cup \{0\}), y \neq \emptyset \vdash z = \mathbf{max}(y \cup \{0\}) \\ & y \in \mathbb{F}(\mathbb{N}_1), z \in \mathbb{N}, z = \mathbf{max}(y \cup \{0\}), y \neq \emptyset \vdash \mathbf{max}(y) = z \end{aligned}$$

Refinamento, verificação, operações com saída

Exemplo

$$\begin{aligned} & \mathbf{INV}_M, \mathbf{INV}_R, \mathbf{PRE}_M \vdash [[\mathbf{o} := \mathbf{o}']\mathbf{OP}_R] \neg [\mathbf{OP}_M] \neg \mathbf{INV}_R \wedge \mathbf{o} = \mathbf{o}' \\ \iff & y \in \mathbb{F}(\mathbb{N}_1), z \in \mathbb{N}, z = \mathbf{max}(y \cup \{0\}), y \neq \emptyset \vdash \mathbf{max}(y) = z \end{aligned}$$

Refinamento, verificação, operações com saída

Exemplo

$$\begin{aligned} & \mathbf{INV}_M, \mathbf{INV}_R, \mathbf{PRE}_M \vdash [[\mathbf{o} := \mathbf{o}']\mathbf{OP}_R] \neg [\mathbf{OP}_M] \neg \mathbf{INV}_R \wedge \mathbf{o} = \mathbf{o}' \\ \iff & y \in \mathbb{F}(\mathbb{N}_1), z \in \mathbb{N}, z = \mathbf{max}(y \cup \{0\}), y \neq \emptyset \vdash \mathbf{max}(y) = z \end{aligned}$$

Refinamento, verificação, operações com saída

Exemplo

$$\Leftrightarrow \text{INV}_M, \text{INV}_R, \text{PRE}_M \vdash [[\mathbf{o} := \mathbf{o}']\text{OP}_R] \neg [\text{OP}_M] \neg \text{INV}_R \wedge \mathbf{o} = \mathbf{o}'$$

Outline

Introdução

Contextualização

Visão global do método B

(tira-gosto)

Especificação dos requisitos funcionais

Uma notação para especificar

Verificação da especificação

Substituições generalizadas e obrigações de prova

Refinamento

Uma notação para refinar

Verificação de refinamentos

Implementação

Introdução

Uma linguagem para implementar

Ferramentas

Ferramentas industriais

Complementos

Implementação no método B

- ▶ Artefato “Implementation”
 - ▶ codificação dos com um sistema de tipos clássico;
 - ▶ realização das operações com o paradigma imperativo sequencial.
- ▶ É um tipo de refinamento (mesmas obrigações de prova).
- ▶ Propriedades verificadas:
 - ▶ Determinística.
 - ▶ Todos os dados são de um tipo diretamente codificável
 - ▶ variáveis concretas,
 - ▶ instâncias de máquinas de biblioteca.

Outline

Introdução

Contextualização

Visão global do método B

(tira-gosto)

Especificação dos requisitos funcionais

Uma notação para especificar

Verificação da especificação

Substituições generalizadas e obrigações de prova

Refinamento

Uma notação para refinar

Verificação de refinamentos

Implementação

Introdução

Uma linguagem para implementar

Ferramentas

Ferramentas industriais

Complementos

Máquinas de biblioteca

- ▶ Módulos reutilizáveis
- ▶ Podem ser
 - ▶ desenvolvidas com o método B
 - ▶ básicas e desenvolvidas usando um outro método
- ▶ Exemplo:
 - ▶ Tipos: Booleanos, tipos inteiros, enumerações, arranjos, registros, *string*;
 - ▶ Chamadas de sistema: TCP/IP connections, file manipulations, etc.

Construção de laços

Laços são uma fonte de erros:

- ▶ não terminam;
- ▶ terminam cedo demais, ou tarde demais;
- ▶ as variáveis tem valores errados;

WHILE T : formula **DO** B : substitution
VARIANT V : expression **INVARIANT** I : formula
END

▶ **variante:**

- ▶ expressão inteira;
- ▶ limite superior da quantidade de repetições;
- ▶ prova do término;

▶ **invariante**

- ▶ fórmula lógica
- ▶ verdadeira cada vez que a guarda é avaliada
- ▶ caracteriza a relação entre os dados do algoritmo
- ▶ prova da corretude.

Exemplo de laço

Example

```
y := x; ctr := 0;  
WHILE ctr < 5 DO  
  x := x + 1; ctr := ctr + 1  
VARIANT 6 - ctr  
INVARIANT ctr ∈ 0..5 ∧ x = y + ctr  
END
```

Verificação de laço

$[INIT; \text{WHILE } T \text{ DO } B \text{ VARIANT } V \text{ INVARIANT } I \text{ END}]R$

Obrigações de prova:

I-rule O invariante do laço é verdadeiro a primeira vez:

$C \vdash [INIT]I$

C : condições correspondentes ao contexto de execução;

F-rule R é verdadeiro quando o laço termina: $I, \neg T \vdash R$;

P-rule preservação do invariante: $I, T \vdash [B]I$;

T1-rule o variante nunca é negativo:

$I \vdash V \in \mathbb{N}$;

T2-rule o variante é diminuído:

$I, T \vdash [V_i := V][B]V < V_i$.

Verificação de um laço: exemplo

Example

```
[y := x; ctr := 0;  
WHILE ctr < 5 DO  
x := x + 1; ctr := ctr + 1  
VARIANT 6 - ctr  
INVARIANT ctr ∈ 0..5 ∧ x = y + ctr  
END]x = y + 5
```

Example

$$[INIT]I \iff [x := y; ctr := 0] ctr \in 0..5 \wedge x = y + ctr$$

Example

$$[INIT]I \iff [x := y; ctr := 0] ctr \in 0..5 \wedge x = y + ctr$$

Example

$$\begin{aligned} [INIT]I &\iff [x := y; ctr := 0]ctr \in 0..5 \wedge x = y + ctr \\ &\iff [x := y][ctr := 0]ctr \in 0..5 \wedge x = y + ctr \end{aligned}$$

Example

$$\begin{aligned} [INIT]I &\iff [x := y; ctr := 0]ctr \in 0..5 \wedge x = y + ctr \\ &\iff [x := y][ctr := 0]ctr \in 0..5 \wedge x = y + ctr \end{aligned}$$

Example

$$\begin{aligned} [INIT]I &\iff [x := y; ctr := 0]ctr \in 0..5 \wedge x = y + ctr \\ &\iff [x := y][ctr := 0]ctr \in 0..5 \wedge x = y + ctr \\ &\iff [x := y]0 \in 0..5 \wedge x = y + 0 \end{aligned}$$

Example

$$\begin{aligned} [INIT]I &\iff [x := y; ctr := 0]ctr \in 0..5 \wedge x = y + ctr \\ &\iff [x := y][ctr := 0]ctr \in 0..5 \wedge x = y + ctr \\ &\iff [x := y]0 \in 0..5 \wedge x = y + 0 \end{aligned}$$

Example

$$\begin{aligned} [INIT]I &\iff [x := y; ctr := 0]ctr \in 0..5 \wedge x = y + ctr \\ &\iff [x := y][ctr := 0]ctr \in 0..5 \wedge x = y + ctr \\ &\iff [x := y]0 \in 0..5 \wedge x = y + 0 \\ &\iff 0 \in 0..5 \wedge y = y + 0 \end{aligned}$$

Example

$$\begin{aligned} [INIT]I &\iff [x := y; ctr := 0]ctr \in 0..5 \wedge x = y + ctr \\ &\iff [x := y][ctr := 0]ctr \in 0..5 \wedge x = y + ctr \\ &\iff [x := y]0 \in 0..5 \wedge x = y + 0 \\ &\iff 0 \in 0..5 \wedge y = y + 0 \\ &\iff \top \end{aligned}$$

Example

$$\begin{aligned} & I, \neg T \vdash R \\ \iff & ctr \in 0..5, x = y + ctr, \neg(ctr < 5) \vdash x = y + 5 \end{aligned}$$

Example

$$\begin{aligned} & I, \neg T \vdash R \\ \iff & ctr \in 0..5, x = y + ctr, \neg(ctr < 5) \vdash x = y + 5 \end{aligned}$$

Example

$$\begin{aligned} & I, \neg T \vdash R \\ \iff & ctr \in 0..5, x = y + ctr, \neg(ctr < 5) \vdash x = y + 5 \\ \iff & ctr \in 0..5, x = y + ctr, ctr \geq 5 \vdash x = y + 5 \end{aligned}$$

Example

$$\begin{aligned} & I, \neg T \vdash R \\ \iff & \mathit{ctr} \in 0..5, x = y + \mathit{ctr}, \neg(\mathit{ctr} < 5) \vdash x = y + 5 \\ \iff & \mathit{ctr} \in 0..5, x = y + \mathit{ctr}, \mathit{ctr} \geq 5 \vdash x = y + 5 \end{aligned}$$

Example

$$I, \neg T \vdash R$$

$$\iff ctr \in 0..5, x = y + ctr, \neg(ctr < 5) \vdash x = y + 5$$

$$\iff ctr \in 0..5, x = y + ctr, ctr \geq 5 \vdash x = y + 5$$

$$\iff ctr = 5, x = y + ctr \vdash x = y + 5$$

Example

$$I, \neg T \vdash R$$

$$\iff ctr \in 0..5, x = y + ctr, \neg(ctr < 5) \vdash x = y + 5$$

$$\iff ctr \in 0..5, x = y + ctr, ctr \geq 5 \vdash x = y + 5$$

$$\iff ctr = 5, x = y + ctr \vdash x = y + 5$$

Example

$$I, \neg T \vdash R$$

$$\iff ctr \in 0..5, x = y + ctr, \neg(ctr < 5) \vdash x = y + 5$$

$$\iff ctr \in 0..5, x = y + ctr, ctr \geq 5 \vdash x = y + 5$$

$$\iff ctr = 5, x = y + ctr \vdash x = y + 5$$

$$\iff \vdash \top$$

T1-rule

Example

$$\begin{aligned} & I \vdash V \in \mathbb{N} \\ \iff & \text{ctr} \in 0..5, x = y + \text{ctr} \vdash 6 - \text{ctr} \in \mathbb{N} \end{aligned}$$

T1-rule

Example

$$\begin{aligned} & I \vdash V \in \mathbb{N} \\ \iff & \mathit{ctr} \in 0..5, x = y + \mathit{ctr} \vdash 6 - \mathit{ctr} \in \mathbb{N} \\ \iff & \mathit{ctr} \in 0..5 \vdash 6 - \mathit{ctr} \in \mathbb{N} \end{aligned}$$

Example

$$I \vdash V \in \mathbb{N}$$

$$\iff ctr \in 0..5, x = y + ctr \vdash 6 - ctr \in \mathbb{N}$$

$$\iff ctr \in 0..5 \vdash 6 - ctr \in \mathbb{N}$$

$$\iff ctr = 0 \vdash 6 - ctr \in \mathbb{N}$$

$$ctr = 1 \vdash 6 - ctr \in \mathbb{N}$$

$$ctr = 2 \vdash 6 - ctr \in \mathbb{N}$$

$$ctr = 3 \vdash 6 - ctr \in \mathbb{N}$$

$$ctr = 4 \vdash 6 - ctr \in \mathbb{N}$$

$$ctr = 5 \vdash 6 - ctr \in \mathbb{N}$$

Example

$$\begin{aligned} & I \vdash V \in \mathbb{N} \\ \iff & \text{ctr} \in 0..5, x = y + \text{ctr} \vdash 6 - \text{ctr} \in \mathbb{N} \\ \iff & \text{ctr} \in 0..5 \vdash 6 - \text{ctr} \in \mathbb{N} \\ \iff & \text{ctr} = 0 \vdash 6 - \text{ctr} \in \mathbb{N} \\ & \text{ctr} = 1 \vdash 6 - \text{ctr} \in \mathbb{N} \\ & \text{ctr} = 2 \vdash 6 - \text{ctr} \in \mathbb{N} \\ & \text{ctr} = 3 \vdash 6 - \text{ctr} \in \mathbb{N} \\ & \text{ctr} = 4 \vdash 6 - \text{ctr} \in \mathbb{N} \\ & \text{ctr} = 5 \vdash 6 - \text{ctr} \in \mathbb{N} \\ \iff & \top \end{aligned}$$

Example

$$I, T \vdash [V_i := V][B] V < V_i$$
$$\iff ctr \in 0..5, x = y + ctr, ctr < 5 \vdash$$
$$[V_i := 6 - ctr][x := x + 1; ctr := ctr + 1] 6 - ctr < V_i$$

T2-rule

Example

$I, T \vdash [V_i := V][B]V < V_i$

$\iff ctr \in 0..5, x = y + ctr, ctr < 5 \vdash$

$[V_i := 6 - ctr][x := x + 1; ctr := ctr + 1]6 - ctr < V_i$

Example

$I, T \vdash [V_i := V][B] V < V_i$

$\iff ctr \in 0..5, x = y + ctr, ctr < 5 \vdash$
 $[V_i := 6 - ctr][x := x + 1; ctr := ctr + 1] 6 - ctr < V_i$

$\iff ctr \in 0..5, \wedge x = y + ctr, ctr < 5 \vdash$
 $[V_i := 6 - ctr][x := x + 1][ctr := ctr + 1] 6 - ctr < V_i$

T2-rule

Example

$I, T \vdash [V_i := V][B] V < V_i$

$\iff ctr \in 0..5, x = y + ctr, ctr < 5 \vdash$
 $[V_i := 6 - ctr][x := x + 1; ctr := ctr + 1] 6 - ctr < V_i$

$\iff ctr \in 0..5, \wedge x = y + ctr, ctr < 5 \vdash$
 $[V_i := 6 - ctr][x := x + 1][ctr := ctr + 1] 6 - ctr < V_i$

T2-rule

Example

$I, T \vdash [V_i := V][B] V < V_i$

$\iff ctr \in 0..5, x = y + ctr, ctr < 5 \vdash$
 $[V_i := 6 - ctr][x := x + 1; ctr := ctr + 1]6 - ctr < V_i$

$\iff ctr \in 0..5, \wedge x = y + ctr, ctr < 5 \vdash$
 $[V_i := 6 - ctr][x := x + 1][ctr := ctr + 1]6 - ctr < V_i$

$\iff \dots \vdash [V_i := 6 - ctr][x := x + 1]6 - (ctr + 1) < V_i$

T2-rule

Example

$I, T \vdash [V_i := V][B] V < V_i$

$\iff ctr \in 0..5, x = y + ctr, ctr < 5 \vdash$
 $[V_i := 6 - ctr][x := x + 1; ctr := ctr + 1]6 - ctr < V_i$

$\iff ctr \in 0..5, \wedge x = y + ctr, ctr < 5 \vdash$
 $[V_i := 6 - ctr][x := x + 1][ctr := ctr + 1]6 - ctr < V_i$

$\iff \dots \vdash [V_i := 6 - ctr][x := x + 1]6 - (ctr + 1) < V_i$

T2-rule

Example

$I, T \vdash [V_i := V][B] V < V_i$

$\iff ctr \in 0..5, x = y + ctr, ctr < 5 \vdash$
 $[V_i := 6 - ctr][x := x + 1; ctr := ctr + 1] 6 - ctr < V_i$

$\iff ctr \in 0..5, \wedge x = y + ctr, ctr < 5 \vdash$
 $[V_i := 6 - ctr][x := x + 1][ctr := ctr + 1] 6 - ctr < V_i$

$\iff \dots \vdash [V_i := 6 - ctr][x := x + 1] 6 - (ctr + 1) < V_i$

$\iff \dots \vdash [V_i := 6 - ctr] 6 - (ctr + 1) < V_i$

T2-rule

Example

$I, T \vdash [V_i := V][B] V < V_i$

$\iff ctr \in 0..5, x = y + ctr, ctr < 5 \vdash$
 $[V_i := 6 - ctr][x := x + 1; ctr := ctr + 1]6 - ctr < V_i$

$\iff ctr \in 0..5, \wedge x = y + ctr, ctr < 5 \vdash$
 $[V_i := 6 - ctr][x := x + 1][ctr := ctr + 1]6 - ctr < V_i$

$\iff \dots \vdash [V_i := 6 - ctr][x := x + 1]6 - (ctr + 1) < V_i$

$\iff \dots \vdash [V_i := 6 - ctr]6 - (ctr + 1) < V_i$

T2-rule

Example

$$I, T \vdash [V_i := V][B] V < V_i$$

$$\begin{aligned} \iff & \quad ctr \in 0..5, x = y + ctr, ctr < 5 \vdash \\ & \quad [V_i := 6 - ctr][x := x + 1; ctr := ctr + 1] 6 - ctr < V_i \end{aligned}$$

$$\begin{aligned} \iff & \quad ctr \in 0..5, \wedge x = y + ctr, ctr < 5 \vdash \\ & \quad [V_i := 6 - ctr][x := x + 1][ctr := ctr + 1] 6 - ctr < V_i \end{aligned}$$

$$\iff \dots \vdash [V_i := 6 - ctr][x := x + 1] 6 - (ctr + 1) < V_i$$

$$\iff \dots \vdash [V_i := 6 - ctr] 6 - (ctr + 1) < V_i$$

$$\iff \dots \vdash 5 - ctr < 6 - ctr$$

T2-rule

Example

$$I, T \vdash [V_i := V][B] V < V_i$$

$$\begin{aligned} \iff & \quad ctr \in 0..5, x = y + ctr, ctr < 5 \vdash \\ & \quad [V_i := 6 - ctr][x := x + 1; ctr := ctr + 1] 6 - ctr < V_i \end{aligned}$$

$$\begin{aligned} \iff & \quad ctr \in 0..5, \wedge x = y + ctr, ctr < 5 \vdash \\ & \quad [V_i := 6 - ctr][x := x + 1][ctr := ctr + 1] 6 - ctr < V_i \end{aligned}$$

$$\iff \dots \vdash [V_i := 6 - ctr][x := x + 1] 6 - (ctr + 1) < V_i$$

$$\iff \dots \vdash [V_i := 6 - ctr] 6 - (ctr + 1) < V_i$$

$$\iff \dots \vdash 5 - ctr < 6 - ctr$$

$$\iff T$$

Example

$$\begin{aligned} & I, T \vdash [B]I \\ \iff & \text{ctr} \in 0..5, x = y + \text{ctr} \text{ctr} < 5 \vdash \\ & [x := x + 1; \text{ctr} := \text{ctr} + 1] \text{ctr} \in 0..5 \wedge x = y + \text{ctr} \end{aligned}$$

Example

$$\begin{aligned} & I, T \vdash [B]I \\ \iff & \text{ctr} \in 0..5, x = y + \text{ctr} \text{ctr} < 5 \vdash \\ & [x := x + 1; \text{ctr} := \text{ctr} + 1] \text{ctr} \in 0..5 \wedge x = y + \text{ctr} \end{aligned}$$

Example

$I, T \vdash [B]I$

$\iff \text{ctr} \in 0..5, x = y + \text{ctr}, \text{ctr} < 5 \vdash$
 $[x := x + 1; \text{ctr} := \text{ctr} + 1] \text{ctr} \in 0..5 \wedge x = y + \text{ctr}$

$\iff \text{ctr} \in 0..5, x = y + \text{ctr}, \text{ctr} < 5 \vdash$
 $[x := x + 1][\text{ctr} := \text{ctr} + 1]$
 $\text{ctr} \in 0..5 \wedge x = y + \text{ctr}$

Example

$I, T \vdash [B]I$

$\iff ctr \in 0..5, x = y + ctr, ctr < 5 \vdash$
 $[x := x + 1; ctr := ctr + 1] ctr \in 0..5 \wedge x = y + ctr$

$\iff ctr \in 0..5, x = y + ctr, ctr < 5 \vdash$
 $[x := x + 1][ctr := ctr + 1]$
 $ctr \in 0..5 \wedge x = y + ctr$

Example

$I, T \vdash [B]I$

$\iff \text{ctr} \in 0..5, x = y + \text{ctr}, \text{ctr} < 5 \vdash$
 $[x := x + 1; \text{ctr} := \text{ctr} + 1] \text{ctr} \in 0..5 \wedge x = y + \text{ctr}$

$\iff \text{ctr} \in 0..5, x = y + \text{ctr}, \text{ctr} < 5 \vdash$
 $[x := x + 1][\text{ctr} := \text{ctr} + 1]$
 $\text{ctr} \in 0..5 \wedge x = y + \text{ctr}$

$\iff \text{ctr} \in 0..5, x = y + \text{ctr}, \text{ctr} < 5 \vdash$
 $[x := x + 1]$
 $\text{ctr} + 1 \in 0..5 \wedge x = y + \text{ctr} + 1$

Example

$I, T \vdash [B]I$

$\iff \text{ctr} \in 0..5, x = y + \text{ctr}, \text{ctr} < 5 \vdash$
 $[x := x + 1; \text{ctr} := \text{ctr} + 1] \text{ctr} \in 0..5 \wedge x = y + \text{ctr}$

$\iff \text{ctr} \in 0..5, x = y + \text{ctr}, \text{ctr} < 5 \vdash$
 $[x := x + 1][\text{ctr} := \text{ctr} + 1]$
 $\text{ctr} \in 0..5 \wedge x = y + \text{ctr}$

$\iff \text{ctr} \in 0..5, x = y + \text{ctr}, \text{ctr} < 5 \vdash$
 $[x := x + 1]$
 $\text{ctr} + 1 \in 0..5 \wedge x = y + \text{ctr} + 1$

Example

$I, T \vdash [B]I$

$\iff ctr \in 0..5, x = y + ctr, ctr < 5 \vdash$
 $[x := x + 1; ctr := ctr + 1] ctr \in 0..5 \wedge x = y + ctr$

$\iff ctr \in 0..5, x = y + ctr, ctr < 5 \vdash$
 $[x := x + 1][ctr := ctr + 1]$
 $ctr \in 0..5 \wedge x = y + ctr$

$\iff ctr \in 0..5, x = y + ctr, ctr < 5 \vdash$
 $[x := x + 1]$
 $ctr + 1 \in 0..5 \wedge x = y + ctr + 1$

$\iff ctr \in 0..5, x = y + ctr, ctr < 5 \vdash ctr + 1 \in 0..5$
 $ctr \in 0..5, x = y + ctr, ctr < 5 \vdash x + 1 = y + ctr + 1$

Example

$I, T \vdash [B]I$

$\iff ctr \in 0..5, x = y + ctr, ctr < 5 \vdash$
 $[x := x + 1; ctr := ctr + 1] ctr \in 0..5 \wedge x = y + ctr$

$\iff ctr \in 0..5, x = y + ctr, ctr < 5 \vdash$
 $[x := x + 1][ctr := ctr + 1]$
 $ctr \in 0..5 \wedge x = y + ctr$

$\iff ctr \in 0..5, x = y + ctr, ctr < 5 \vdash$
 $[x := x + 1]$
 $ctr + 1 \in 0..5 \wedge x = y + ctr + 1$

$\iff ctr \in 0..5, x = y + ctr, ctr < 5 \vdash ctr + 1 \in 0..5$
 $ctr \in 0..5, x = y + ctr, ctr < 5 \vdash x + 1 = y + ctr + 1$

Example

$I, T \vdash [B]I$

$\iff ctr \in 0..5, x = y + ctr, ctr < 5 \vdash$
 $[x := x + 1; ctr := ctr + 1] ctr \in 0..5 \wedge x = y + ctr$

$\iff ctr \in 0..5, x = y + ctr, ctr < 5 \vdash$
 $[x := x + 1][ctr := ctr + 1]$
 $ctr \in 0..5 \wedge x = y + ctr$

$\iff ctr \in 0..5, x = y + ctr, ctr < 5 \vdash$
 $[x := x + 1]$
 $ctr + 1 \in 0..5 \wedge x = y + ctr + 1$

$\iff T$
 $ctr \in 0..5, x = y + ctr, ctr < 5 \vdash x + 1 = y + ctr + 1$

Example

$I, T \vdash [B]I$

$\iff ctr \in 0..5, x = y + ctr, ctr < 5 \vdash$
 $[x := x + 1; ctr := ctr + 1] ctr \in 0..5 \wedge x = y + ctr$

$\iff ctr \in 0..5, x = y + ctr, ctr < 5 \vdash$
 $[x := x + 1][ctr := ctr + 1]$
 $ctr \in 0..5 \wedge x = y + ctr$

$\iff ctr \in 0..5, x = y + ctr, ctr < 5 \vdash$
 $[x := x + 1]$
 $ctr + 1 \in 0..5 \wedge x = y + ctr + 1$

$\iff T$
 $ctr \in 0..5, x = y + ctr, ctr < 5 \vdash x + 1 = y + ctr + 1$

Example

$I, T \vdash [B]I$

$\iff ctr \in 0..5, x = y + ctr, ctr < 5 \vdash$
 $[x := x + 1; ctr := ctr + 1]ctr \in 0..5 \wedge x = y + ctr$

$\iff ctr \in 0..5, x = y + ctr, ctr < 5 \vdash$
 $[x := x + 1][ctr := ctr + 1]$
 $ctr \in 0..5 \wedge x = y + ctr$

$\iff ctr \in 0..5, x = y + ctr, ctr < 5 \vdash$
 $[x := x + 1]$
 $ctr + 1 \in 0..5 \wedge x = y + ctr + 1$

$\iff T$

Outline

Introdução

Contextualização

Visão global do método B

(tira-gosto)

Especificação dos requisitos funcionais

Uma notação para especificar

Verificação da especificação

Substituições generalizadas e obrigações de prova

Refinamento

Uma notação para refinar

Verificação de refinamentos

Implementação

Introdução

Uma linguagem para implementar

Ferramentas

Ferramentas industriais

Complementos

- ▶ Verificação de sintaxe;
- ▶ Verificação de tipos;
- ▶ Prova automática e semi-automática;
- ▶ Gerenciamento do processo;
- ▶ Síntese de código C, ADA;
- ▶ Linha de comandos ou interação gráfica;
- ▶ Animação (B-RAMA);
- ▶ Automação de refinamentos (BART);
- ▶ Gratuito, código parcialmente aberto, formato aberto (XML): extensível.
 - ▶ Animação, model-checking (ProB)
 - ▶ Verificação automática (BEval)
 - ▶ Geração de código (b2llvm)

Aspectos que este tutorial não abordou

- ▶ Açúcar sintático;
- ▶ Construções de modularização.

- ▶ Integração com outras tecnologias de engenharia de software (UML);
 - ▶ Integração com outros formalismos: CSP, CCS.
- ▶ Síntese de código;
- ▶ Automação das atividades de verificação;
- ▶ Extensão do domínio de modelagem: probabilidade, aspectos temporais, domínios contínuos, etc.