Semântica do PVS: Tipos Dependentes e Aplicações

André Luiz Galdino[1]

Universidade de Brasília
Departamento de Matemática
Grupo Teoria da Computação

[1]Liberado para cursar doutorado pelo
Departamento de Matemática
Campus de Catalão
Universidade Federal de Goiás

28 de Outubro de 2004

O que é PVS?

O que é PVS?

O PVS (Prototype Verification System) é um ambiente para especificação e verificação semi-automática.

O que é PVS?

O PVS (Prototype Verification System) é um ambiente para especificação e verificação semi-automática.

Desenvolvido na década de 80 pelo SRI International Computer Science Laboratory, o PVS é baseado sobre uma lógica fortemente tipada de ordem-superior, e possui um verificador de provas baseado sobre cálculos sequentes que combina decisões de procedimento e estratégias.

Universo Básico U

Para definir a semântica precisamos de um universo que contém os conjuntos **2** e **R** e é fechado com respeito a produtos cartesianos, $X \times Y$, e potências de conjuntos, $\wp(A)$.

Universo Básico U

Para definir a semântica precisamos de um universo que contém os conjuntos **2** e **R** e é fechado com respeito a produtos cartesianos, $X \times Y$, e potências de conjuntos, $\wp(A)$.

$$
\begin{aligned}
U_0 &= \{\mathbf{2}, \mathbf{R}\} \\
U_{i+1} &= U_i \cup \{X \times Y | X, Y \in U_i\} \cup \{X^Y | X, Y \in U_i\} \cup \bigcup_{X \in U_i} \wp(X) \\
U_\omega &= \bigcup_{i<\omega} U_i \\
U &= U_\omega
\end{aligned}
$$

Pré-tipos

Os pré-tipos da teoria de tipos simples incluem:

- os tipos *básicos* bool e real;

Pré-tipos

Os pré-tipos da teoria de tipos simples incluem:

- os tipos *básicos* bool e real;
- uma função pré-tipo que é construída como $[A \to B]$, com A e B pré-tipos;

Pré-tipos

Os pré-tipos da teoria de tipos simples incluem:

- os tipos *básicos* bool e real;
- uma função pré-tipo que é construída como $[A \to B]$, com A e B pré-tipos;
- um produto de pré-tipos que é construído como $[A_1, A_2]$, com A_1 e A_2 pré-tipos.

Pré-termos

Os pretermos da linguagem consistem de:

- constantes: c, $TRUE$, $FALSE$

Pré-termos

Os pretermos da linguagem consistem de:

- constantes: c, $TRUE$, $FALSE$
- variáveis: $x : VAR \quad T$ onde T é um pré-tipo

Pré-termos

Os pretermos da linguagem consistem de:

- constantes: c, $TRUE$, $FALSE$
- variáveis: $x : VAR \quad T$ onde T é um pré-tipo
- pares: $\langle a_1, a_2 \rangle$ onde cada a_i é um pré-termo

Pré-termos

Os pretermos da linguagem consistem de:

- constantes: c, $TRUE$, $FALSE$
- variáveis: $x : VAR \quad T$ onde T é um pré-tipo
- pares: $\langle a_1, a_2 \rangle$ onde cada a_i é um pré-termo
- projeções: $p_i \ a$ onde $i \in \{1, 2\}$ e a é um par

Pré-termos

Os pretermos da linguagem consistem de:

- constantes: c, $TRUE$, $FALSE$
- variáveis: $x : VAR \quad T$ onde T é um pré-tipo
- pares: $\langle a_1, a_2 \rangle$ onde cada a_i é um pré-termo
- projeções: $p_i \ a$ onde $i \in \{1, 2\}$ e a é um par
- aplicações: $f \ a$ onde f e a são pré-termos

Pré-termos

Os pretermos da linguagem consistem de:

- constantes: c, $TRUE$, $FALSE$
- variáveis: $x : VAR \quad T$ onde T é um pré-tipo
- pares: $\langle a_1, a_2 \rangle$ onde cada a_i é um pré-termo
- projeções: $p_i \ a$ onde $i \in \{1, 2\}$ e a é um par
- aplicações: $f \ a$ onde f e a são pré-termos
- abstrações: $\lambda(x : T) : a$ onde T é um pré-tipo e a é um pré-termo

Exemplo de Contexto, $kind(\Gamma(s))$ e $type(\Gamma(s))$

A sequência de declarações abaixo é um contexto denotado por Γ.

$bool : TYPE, TRUE : bool, FALSE : bool, x : VAR\ [[bool, bool] \rightarrow bool]$

Exemplo de Contexto, $kind(\Gamma(s))$ e $type(\Gamma(s))$

A sequência de declarações abaixo é um contexto denotado por Γ.

$bool : TYPE, TRUE : bool, FALSE : bool, x : VAR\ [[bool, bool] \rightarrow bool]$

Neste contexto temos:

- $kind(\Gamma(bool)) = TYPE$

Exemplo de Contexto, $kind(\Gamma(s))$ e $type(\Gamma(s))$

A sequência de declarações abaixo é um contexto denotado por Γ.

$bool : TYPE, TRUE : bool, FALSE : bool, x : VAR\ [[bool, bool] \rightarrow bool]$

Neste contexto temos:

- $kind(\Gamma(bool)) = TYPE$
- $kind(\Gamma(TRUE)) = CONSTANT$

Exemplo de Contexto, $kind(\Gamma(s))$ e $type(\Gamma(s))$

A sequência de declarações abaixo é um contexto denotado por Γ.

$bool : TYPE, TRUE : bool, FALSE : bool, x : VAR\ [[bool, bool] \rightarrow bool]$

Neste contexto temos:

- $kind(\Gamma(bool)) = TYPE$
- $kind(\Gamma(TRUE)) = CONSTANT$
- $kind(\Gamma(x)) = VARIABLE$

Exemplo de Contexto, $kind(\Gamma(s))$ e $type(\Gamma(s))$

A sequência de declarações abaixo é um contexto denotado por Γ.

$bool : TYPE, TRUE : bool, FALSE : bool, x : VAR\ [[bool, bool] \rightarrow bool]$

Neste contexto temos:

- $kind(\Gamma(bool)) = TYPE$
- $kind(\Gamma(TRUE)) = CONSTANT$
- $kind(\Gamma(x)) = VARIABLE$
- $type(\Gamma(x)) = [[bool, bool] \rightarrow bool]$

Observações

- O contexto vazio é representado como $\{\}$

Observações

- O contexto vazio é representado como $\{\}$
- Seja Γ um contexto e $s : D$. Então:

Observações

- O contexto vazio é representado como $\{\}$
- Seja Γ um contexto e $s : D$. Então:
  - $(\Gamma, s : D)(s) = D$

Observações

- O contexto vazio é representado como $\{\}$
- Seja Γ um contexto e $s : D$. Então:
  - $(\Gamma, s : D)(s) = D$
  - $(\Gamma, s : D)(r) = \Gamma(r)$ para $r \neq s$.

Observações

- O contexto vazio é representado como $\{\}$
- Seja Γ um contexto e $s : D$. Então:
  - $(\Gamma, s : D)(s) = D$
  - $(\Gamma, s : D)(r) = \Gamma(r)$ para $r \neq s$.
- Se s não está declarado em Γ, então $\Gamma(s)$ está indefinido.

Regra de Tipos: Contexto

$$\tau()(\{\}) = \mathit{CONTEXT}$$

$$\tau()(\Gamma, s : \mathit{TYPE}) = \mathit{CONTEXT}, \text{ se } \Gamma(s) \text{ é indefinido e } \tau()(\Gamma) = \mathit{CONTEXT}$$

$$\tau()(\Gamma, c : T) = \mathit{CONTEXT}, \text{ se } \Gamma(c) \text{ é indefinido, } \tau(\Gamma)(T) = \mathit{TYPE} \text{ e } \tau()(\Gamma) = \mathit{CONTEXT}$$

$$\tau()(\Gamma, x : \mathit{VAR}\ T) = \mathit{CONTEXT}, \text{ se } \Gamma(x) \text{ é indefinido, } \tau(\Gamma)(T) = \mathit{TYPE} \text{ e } \tau()(\Gamma) = \mathit{CONTEXT}$$

Regra de Tipos: Pré-tipos

$$\tau(\Gamma)(s) = TYPE \text{ se } kind(\Gamma(s)) = TYPE$$

$$\tau(\Gamma)([A \to B]) = TYPE \text{ se } \tau(\Gamma)(A) = \tau(\Gamma)(B) = TYPE$$

$$\tau(\Gamma)([A_1, A_2]) = TYPE \text{ se } \tau(\Gamma)(A_i) = TYPE \text{ para } 1 \leq i \leq 2$$

Regra de Tipos: Pré-termos

$$
\begin{aligned}
\tau(\Gamma)(s) &= type(\Gamma(s)) \\
&\quad \text{se } kind(\Gamma(s)) \in \{CONSTANT, VARIABLE\} \\
\tau(\Gamma)(f\ a) &= B \text{ se } \tau(\Gamma)(f) = [A \to B] \text{ e } \tau(\Gamma)(a) = A \\
\tau(\Gamma)(\lambda(x : T) : a) &= [T \to \tau(\Gamma, x : VAR\ T)(a)] \\
&\quad \text{se } \Gamma(x) \text{ é indefinido e } \tau(\Gamma)(T) = TYPE \\
\tau(\Gamma)((a_1, a_2)) &= [\tau(\Gamma)(a_1), \tau(\Gamma)(a_2)] \\
\tau(\Gamma)(p_i\ a) &= T_i \text{ onde} \\
&\quad \tau(\Gamma)(a) = [T_1, T_2]
\end{aligned}
$$

Exemplo: Regra de Tipos

Seja Ω classificado como o contexto $bool : TYPE$, $TRUE : bool$, $FALSE : bool$.

$$\begin{aligned}
\tau()(\{\}) &= CONTEXT \\
\tau()(\Omega) &= CONTEXT \\
\tau(\Omega)([[bool, bool] \rightarrow bool]) &= TYPE \\
\tau(\Omega)((TRUE, FALSE)) &= [bool, bool] \\
\tau(\Omega)(p_2\ (TRUE, FALSE)) &= bool \\
\tau(\Omega)(\lambda(x : bool) : TRUE) &= [bool \rightarrow bool]
\end{aligned}$$

Semântica do PVS

Uma designação γ é uma lista de ligações da forma $\{s_1 \leftarrow t_1\} \cdots \{s_n \leftarrow t_n\}$. A aplicação de uma designação γ para um símbolo s é tal que $\gamma\{s \leftarrow t\}(s)$ é t, enquanto que $\gamma\{r \leftarrow t\}(s)$ é $\gamma(s)$ quando $r \neq s$.

Semântica do PVS

Uma designação γ é uma lista de ligações da forma $\{s_1 \leftarrow t_1\} \cdots \{s_n \leftarrow t_n\}$. A aplicação de uma designação γ para um símbolo s é tal que $\gamma\{s \leftarrow t\}(s)$ é t, enquanto que $\gamma\{r \leftarrow t\}(s)$ é $\gamma(s)$ quando $r \neq s$.

A semântica da teoria de tipos simples do PVS é dada aplicando um tipo T para um conjunto (possivelmente vazio) $\mathcal{M}(\Gamma \mid \gamma)(T)$, e um termo a com tipo designado T para um elemento do conjunto $\mathcal{M}(\Gamma \mid \gamma)(T)$ no universo U.

Significado da Função

$$\mathcal{M}(\Gamma \mid \gamma)(s) = \gamma(s) \text{ se } \mathit{kind}(\Gamma(s)) \text{ está no conjunto, } \{\mathit{TYPE}, \mathit{CONSTANT}, \mathit{VARIABLE}\}$$

$$\mathcal{M}(\Gamma \mid \gamma)([A \to B]) = \mathcal{M}(\Gamma \mid \gamma)(B)^{\mathcal{M}(\Gamma \mid \gamma)(A)}$$

$$\mathcal{M}(\Gamma \mid \gamma)([T_1, T_2]) = \mathcal{M}(\Gamma \mid \gamma)(T_1) \times \mathcal{M}(\Gamma \mid \gamma)(T_2)$$

$$\mathcal{M}(\Gamma \mid \gamma)(f\ a) = (\mathcal{M}(\Gamma \mid \gamma)(f))(\mathcal{M}(\Gamma \mid \gamma)(a))$$

$$\mathcal{M}(\Gamma \mid \gamma)(\lambda(x : T) : a) = \{\langle y, z\rangle \mid y \in \mathcal{M}(\Gamma \mid \gamma)(T), z = \mathcal{M}(\Gamma, x : \mathit{VAR}\ \ T \mid \gamma\{x \leftarrow y\})(a)\}$$

$$\mathcal{M}(\Gamma \mid \gamma)((a_1, a_2)) = \langle \mathcal{M}(\Gamma \mid \gamma)(a_1), \mathcal{M}(\Gamma \mid \gamma)(a_2)\rangle$$

$$\mathcal{M}(\Gamma \mid \gamma)(p_i\ \ a) = t_i, \text{ onde } \mathcal{M}(\Gamma \mid \gamma)(a) = \langle t_1, t_2\rangle$$

Exemplo: Significado da Função

Seja ω uma designação para o contexto Ω do Exemplo 2.4, da forma

$$\{bool \leftarrow \mathbf{2}\}\{TRUE \leftarrow \mathbf{1}\}\{FALSE \leftarrow \mathbf{0}\}$$

então

$$\begin{aligned}
\mathcal{M}(\Omega \mid \omega)([bool, bool]) &= \mathbf{2} \times \mathbf{2} \\
\mathcal{M}(\Omega \mid \omega)((TRUE, FALSE)) &= \langle \mathbf{1}, \mathbf{0} \rangle \\
\mathcal{M}(\Omega \mid \omega)(\lambda(x : bool) : TRUE)) &= \{\langle \mathbf{0},\ \mathbf{1} \rangle, \langle \mathbf{1}, \mathbf{1} \rangle\}
\end{aligned}$$

Satisfação

Uma designação de contexto γ é dita satisfazer um contexto Γ (em símbolos $\gamma \models \Gamma$) se, e somente se,

- $\gamma(bool) = \mathbf{2}$,
- $\gamma(TRUE) = \mathbf{1}$,
- $\gamma(FALSE) = \mathbf{0}$,
- $\gamma(s) \in U$ sempre que $kind(\Gamma(s)) = TYPE$, e
- $\gamma(s) \in \mathcal{M}(\Gamma \mid \gamma)(type(\Gamma(s))$ sempre que $kind(\Gamma(s)) \in \{CONSTANT, VARIABLE\}$.

Correção de Tipos e Termos

Correção de Tipos e Termos

Proposição

Se $\tau()(\Gamma) = \tau()(\Gamma') = \mathit{CONTEXT}$ e Γ é um prefixo de Γ', então para todo pré-tipo A, $\tau(\Gamma)(A) = \mathit{TYPE}$ implica $\tau(\Gamma')(A) = \mathit{TYPE}$, e para todo pré-termo a, $\tau(\Gamma)(a) = A$ implica $\tau(\Gamma')(a) = A$.

Correção de Tipos e Termos

Proposição

Se $\tau()(\Gamma) = \tau()(\Gamma') = \mathit{CONTEXT}$ e Γ é um prefixo de Γ', então para todo pré-tipo A, $\tau(\Gamma)(A) = \mathit{TYPE}$ implica $\tau(\Gamma')(A) = \mathit{TYPE}$, e para todo pré-termo a, $\tau(\Gamma)(a) = A$ implica $\tau(\Gamma')(a) = A$.

Teoremas

- Se $\tau()(\Gamma) = \mathit{CONTEXT}$ e $\tau(\Gamma)(a) = A$, então $\tau(\Gamma)(A) = \mathit{TYPE}$.

Correção de Tipos e Termos

Proposição

Se $\tau()(\Gamma) = \tau()(\Gamma') = \mathit{CONTEXT}$ e Γ é um prefixo de Γ', então para todo pré-tipo A, $\tau(\Gamma)(A) = \mathit{TYPE}$ implica $\tau(\Gamma')(A) = \mathit{TYPE}$, e para todo pré-termo a, $\tau(\Gamma)(a) = A$ implica $\tau(\Gamma')(a) = A$.

Teoremas

- Se $\tau()(\Gamma) = \mathit{CONTEXT}$ e $\tau(\Gamma)(a) = A$, então $\tau(\Gamma)(A) = \mathit{TYPE}$.
- (Correção de Tipos)Se $\tau()(\Gamma) = \mathit{CONTEXT}$, γ satisfaz Γ, e $\tau(\Gamma)(A) = \mathit{TYPE}$, então $\mathcal{M}(\Gamma \mid \gamma)(A) \in U$.

Correção de Tipos e Termos

Proposição

Se $\tau()(\Gamma) = \tau()(\Gamma') = \mathit{CONTEXT}$ e Γ é um prefixo de Γ', então para todo pré-tipo A, $\tau(\Gamma)(A) = \mathit{TYPE}$ implica $\tau(\Gamma')(A) = \mathit{TYPE}$, e para todo pré-termo a, $\tau(\Gamma)(a) = A$ implica $\tau(\Gamma')(a) = A$.

Teoremas

- Se $\tau()(\Gamma) = \mathit{CONTEXT}$ e $\tau(\Gamma)(a) = A$, então $\tau(\Gamma)(A) = \mathit{TYPE}$.
- (Correção de Tipos)Se $\tau()(\Gamma) = \mathit{CONTEXT}$, γ satisfaz Γ, e $\tau(\Gamma)(A) = \mathit{TYPE}$, então $\mathcal{M}(\Gamma \mid \gamma)(A) \in U$.
- (Correção de Termos) Se $\tau()(\Gamma) = \mathit{CONTEXT}$, γ satisfaz Γ, e $\tau(\Gamma)(a)$ está definido e igual a A, então $\mathcal{M}(\Gamma \mid \gamma)(a) \in \mathcal{M}(\Gamma \mid \gamma)(A)$.

Adicionando Subtipos

Adicionando Subtipos

- O subtipo de elementos de um tipo T satisfazendo o predicado p é escrito como $\{x : T \mid p(x)\}$.

Adicionando Subtipos

- O subtipo de elementos de um tipo T satisfazendo o predicado p é escrito como $\{x : T \mid p(x)\}$.
- Por exemplo, se T é o tipo real então o subtipo dos números reais não nulos é dado como $\{x : real \mid x \neq 0\}$.

Supertipo Maximal e Supertipo Direto

Supertipo Maximal e Supertipo Direto

$$
\begin{aligned}
\mu(s) &= s \\
\mu(\{x : T \mid a\}) &= \mu(T) \\
\mu([A \to B]) &= [A \to \mu(B)] \\
\mu([A_1, A_2]) &= [\mu(A_1), \mu(A_2)]
\end{aligned}
$$

Supertipo Maximal e Supertipo Direto

$$
\begin{aligned}
\mu(s) &= s \\
\mu(\{x : T \mid a\}) &= \mu(T) \\
\mu([A \to B]) &= [A \to \mu(B)] \\
\mu([A_1, A_2]) &= [\mu(A_1), \mu(A_2)]
\end{aligned}
$$

$$
\begin{aligned}
\mu_0(\{x : T \mid a\}) &= \mu_0(T) \\
\mu_0(T) &= T, \text{ caso contrário}
\end{aligned}
$$

Exemplo: Supertipo Maximal e Direto

Dado um contexto contendo as declarações

$int : TYPE$

$0 : int$

$\leq : [[int, int] \rightarrow bool]$

$nat : TYPE = \{i : int \mid 0 \leq i\}$

$$natinjection : TYPE = \{f : [nat \rightarrow nat] \mid \forall (i, j : nat) : f(i) = f(j) \supset i = j\}$$

Exemplo: Supertipo Maximal e Direto

Dado um contexto contendo as declarações

$int : TYPE$

$0 : int$

$\leq : [[int, int] \rightarrow bool]$

$nat : TYPE = \{i : int \mid 0 \leq i\}$

$$natinjection : TYPE = \{f : [nat \rightarrow nat] \mid \forall (i, j : nat) : f(i) = f(j) \supset i = j\}$$

temos que

$$\begin{aligned} \mu(natinjection) &= \mu([nat \rightarrow nat]) \\ &= [nat \rightarrow \mu(nat)] \\ &= [nat \rightarrow int] \end{aligned}$$

$$\mu_0(natinjection) = [nat \rightarrow nat]$$

Restrição de Subtipo

Restrição de Subtipo

Definição

$$
\begin{aligned}
\pi(s) &= \lambda(x : s) : TRUE \\
\pi(\{y : T \mid a\}) &= \lambda(x : \mu(T)) : (\pi(T)(x) \wedge a[x/y]) \\
\pi([A \to B]) &= \lambda(x : [A \to \mu(B)]) : (\forall(y : A) : \pi(B)(x(y))) \\
\pi([A_1, A_2]) &= \lambda(x : [\mu(A_1), \mu(A_2)]) : (\pi(A_1)(p_1\ x) \wedge \pi(A_2)(p_2\ x))
\end{aligned}
$$

Exemplo: Restrição de Subtipo

$$\pi(nat) = \lambda(j : int) : 0 \leq j$$

$$\pi([nat \rightarrow nat])$$

$$= \lambda(g : [nat \rightarrow int]) : \forall(i : nat) : (\lambda(j : int) : 0 \leq j)(g(i))$$

$$\pi(natinjection)$$

$$= \lambda(f : [nat \rightarrow int]) : \quad \pi([nat \rightarrow nat])(f)$$
$$\wedge \quad (\forall(i, j : nat) : f(i) = f(j) \supset i = j)$$

$$= \lambda(f : [nat \rightarrow int]) :$$
$$(\lambda(g : [nat \rightarrow int]) : \forall(i : nat) : (\lambda(j : int) : 0 \leq j)(g(i)))(f)$$
$$\wedge \quad (\forall(i, j : nat) : f(i) = f(j) \supset i = j)$$

Obrigações de Prova para Equivalência de Tipos (Maximais)

Obrigações de Prova para Equivalência de Tipos (Maximais)

- $(s \simeq s) = TRUE$

Obrigações de Prova para Equivalência de Tipos (Maximais)

- $(s \simeq s) = TRUE$
- $([A \to B] \simeq [A' \to B']) =$
  $((\mu(A) \simeq \mu(A')); (\pi(A) = \pi(A')); (B \simeq B'))$

Obrigações de Prova para Equivalência de Tipos (Maximais)

- $(s \simeq s) = TRUE$
- $([A \to B] \simeq [A' \to B']) =$
  $((\mu(A) \simeq \mu(A')); (\pi(A) = \pi(A')); (B \simeq B'))$
- $([A_1, A_2] \simeq [B_1, B_2]) = ((A_1 \simeq B_1); (A_2 \simeq B_2))$

Obrigações de Prova para Equivalência de Tipos (Maximais)

- $(s \simeq s) = \mathit{TRUE}$
- $([A \to B] \simeq [A' \to B']) =$
  $((\mu(A) \simeq \mu(A')); (\pi(A) = \pi(A')); (B \simeq B'))$
- $([A_1, A_2] \simeq [B_1, B_2]) = ((A_1 \simeq B_1); (A_2 \simeq B_2))$
- $(A \simeq B) = \mathit{FALSE}$, caso contrário.

Exemplo: Tipos Equivalentes

Seja o contexto contendo as declarações

$int : TYPE$

$0 : int$

$\leq : [[int, int] \rightarrow bool]$

$nat : TYPE = \{i : int \mid 0 \leq i\}$

$$natinjection : TYPE = \{f : [nat \rightarrow nat] \mid \forall (i, j : nat) : f(i) = f(j) \supset i = j\}$$

$NAT : TYPE = \{i : int \mid i \leq 0 \supset i = 0\}$

$$NATinjection : TYPE = \{f : [NAT \rightarrow NAT] \mid \forall (i, j : NAT) : f(i) = f(j) \supset i = j\}$$

Continuação do exemplo: Tipos Equivalentes

$$\mu([natinjection \rightarrow natinjection]) = [natinjection \rightarrow [nat \rightarrow int]]$$

$$\mu([NATinjection \rightarrow NATinjection]) = [NATinjection \rightarrow [NAT \rightarrow int]]$$

Continuação do exemplo: Tipos Equivalentes

$$\mu([natinjection \rightarrow natinjection]) = [natinjection \rightarrow [nat \rightarrow int]]$$

$$\mu([NATinjection \rightarrow NATinjection]) = [NATinjection \rightarrow [NAT \rightarrow int]]$$

Então

$$\mu([natinjection \rightarrow natinjection]) \simeq \mu([NATinjection \rightarrow NATinjection])$$

Continuação do exemplo: Tipos Equivalentes

$$\mu([natinjection \rightarrow natinjection]) = [natinjection \rightarrow [nat \rightarrow int]]$$

$$\mu([NATinjection \rightarrow NATinjection]) = [NATinjection \rightarrow [NAT \rightarrow int]]$$

Então

$$\mu([natinjection \rightarrow natinjection]) \simeq \mu([NATinjection \rightarrow NATinjection])$$

se, e somente se,

$$[natinjection \rightarrow [nat \rightarrow int]] \simeq [NATinjection \rightarrow [NAT \rightarrow int]]$$

$$\begin{aligned} = \quad & (\mu(natinjection) \simeq \mu(NATinjection)); \\ & (\pi(natinjection) = \pi(NATinjection)); \\ & ([nat \rightarrow int]) \simeq ([NAT \rightarrow int]) \end{aligned}$$

Continuação do exemplo: Tipos Equivalentes

- $(\pi(\mathit{natinjection}) = \pi(\mathit{NATinjection}))$

Continuação do exemplo: Tipos Equivalentes

- $(\pi(natinjection) = \pi(NATinjection))$

se, e somente se,

$$(\lambda(f : [nat \to int]) : (\lambda(g : [nat \to int]) : \forall(i : nat) : 0 \leq g(i))(f) \\ \wedge \quad (\forall(i, j : nat) : f(i) = f(j) \supset i = j)$$

$=$

$$\lambda(f : [NAT \to int]) : \\ (\lambda(g : [NAT \to int]) : \forall(i : NAT) : g(i) \leq 0 \ \supset \ g(i) = 0)(f)\ ) \\ \wedge \quad (\forall(i, j : NAT) : f(i) = f(j) \supset i = j)$$

Continuação do exemplo: Tipos Equivalentes

- $(\pi(natinjection) = \pi(NATinjection))$

se, e somente se,

$$(\lambda(f : [nat \to int]) : (\lambda(g : [nat \to int]) : \forall(i : nat) : 0 \leq g(i))(f) \\ \wedge \quad (\forall(i, j : nat) : f(i) = f(j) \supset i = j)$$

$=$

$$\lambda(f : [NAT \to int]) : \\ (\lambda(g : [NAT \to int]) : \forall(i : NAT) : g(i) \leq 0 \ \supset \ g(i) = 0)(f)\ ) \\ \wedge \quad (\forall(i, j : NAT) : f(i) = f(j) \supset i = j)$$

- $([nat \to int]) \simeq ([NAT \to int])$

$$= \quad (int \simeq int); \\ (\lambda(i : int) : 0 \leq i) = (\lambda(i : int) : i \leq 0 \ \supset \ i = 0); \\ (int \simeq int)$$

Compatibilidade entre Tipos

Definição

Dois tipos A e B são ditos ser compatíveis no contexto Γ (em notação, $(A \sim B)_\Gamma$) se $\vdash_\Gamma a$, para cada a em $(\mu(A) \simeq \mu(B))$.

Exemplo: Verificando Tipo

$$\begin{array}{lcl} \tau(\Gamma)(f\ \ a) & = & B, \text{ onde } \mu_0(\tau(\Gamma)(f)) = [A \to B], \\ & & \tau(\Gamma)(a) = A', \\ & & (A \sim A')_\Gamma, \\ & & \vdash_\Gamma \pi(A)(a) \end{array}$$

Exemplo: Verificando Tipo

Seja $g : \{f : [nat \to nat] \mid f(0) = 0\}$, e $x : int$. Então

$$\begin{aligned}
\mu_0(\tau(\Gamma)(g)) &= [nat \to nat] \\
\tau(\Gamma)(x) &= int, \qquad \text{e ainda} \\
int &\sim nat, \qquad \text{desde que } \mu(int) = \mu(nat) = number \\
\tau(\Gamma)(g(x)) &= nat, \qquad \text{com a obrigação de prova} \\
\pi(nat)(x) &= (x \geq 0).
\end{aligned}$$

Tipos Dependentes: Motivação

Tipos Dependentes: Motivação

- Definição do coeficiente binomial $C(n, k) = \binom{n}{k}$

Tipos Dependentes: Motivação

- Definição do coeficiente binomial $C(n,k) = \binom{n}{k}$
- Primeiro, definimos a operação fatorial recursivamente:

$n : VAR \quad nat$

$fatorial(n) :$ RECURSIVE posnat $=$

$(IF \quad n > 0$ THEN $n * fatorial(n-1)$ ELSE 1 ENDIF)

MEASURE n

Tipos Dependentes: Motivação

- Definição do coeficiente binomial $C(n,k) = \binom{n}{k}$
- Primeiro, definimos a operação fatorial recursivamente:

  $n : VAR \quad nat$

  $fatorial(n)$: RECURSIVE posnat =

  $(IF \quad n > 0$ THEN $n * fatorial(n-1)$ ELSE 1 ENDIF)

  MEASURE n
- $C(n,(k : upto(n)))$: rat =

  $fatorial(n)/(fatorial(k) * fatorial(n-k))$

onde $upto(n) = \{s : nat \mid s <= n\}$

Tipos Dependentes

- Um produto de tipos com dependência é escrito como $[x : A, B]$

Tipos Dependentes

- Um produto de tipos com dependência é escrito como $[x : A, B]$
- Uma função tipo com dependência é escrita como $[x : A \rightarrow B]$.

Exemplo: Tipos Dependentes

Exemplo: Tipos Dependentes

- $[i : nat, \{j : nat \mid j \leq i\}]$

Exemplo: Tipos Dependentes

- $[i : nat, \{j : nat \mid j \leq i\}]$
- $[i : nat, [\{j : nat \mid j \leq i\} \rightarrow bool]]$

Exemplo: Tipos Dependentes

- $[i : nat, \{j : nat \mid j \leq i\}]$
- $[i : nat, [\{j : nat \mid j \leq i\} \rightarrow bool]]$
- $[i : int \rightarrow \{j : int \mid i \leq j\}]$

Exemplo: Tipos Dependentes

- $[i : nat, \{j : nat \mid j \leq i\}]$
- $[i : nat, [\{j : nat \mid j \leq i\} \rightarrow bool]]$
- $[i : int \rightarrow \{j : int \mid i \leq j\}]$
- $[nat, d : \{n : nat \mid n \neq 0\} \rightarrow \{r : nat \mid r < d\}]$

Teoria de Prova do PVS

Teoria de Prova do PVS

A teoria de prova do PVS é apresentada em termos de cálculos sequente.

Teoria de Prova do PVS

A teoria de prova do PVS é apresentada em termos de cálculos sequente.

Um sequente é da forma:

$$\Sigma \vdash_{\Gamma} \Lambda$$

onde Γ é um contexto, Σ é o conjunto de fórmulas antecedentes e Λ é o conjunto de fórmulas consequentes.

Teoria de Prova do PVS

A teoria de prova do PVS é apresentada em termos de cálculos sequente.

Um sequente é da forma:

$$\Sigma \vdash_{\Gamma} \Lambda$$

onde Γ é um contexto, Σ é o conjunto de fórmulas antecedentes e Λ é o conjunto de fórmulas consequentes.

As regras de inferência são apresentadas na forma:

$$\frac{\texttt{premissa(s)}}{\texttt{conclusão}}\,\texttt{nomeladocondição}$$

Regras de Provas do PVS

- Enfraquecimento **(W)**

$$\frac{\Sigma_1 \vdash_\Gamma \Lambda_1}{\Sigma_2 \vdash_\Gamma \Lambda_2}(\mathbf{W})$$

se $\Sigma_1 \subseteq \Sigma_2$ e $\Lambda_1 \subseteq \Lambda_2$

Regras de Provas do PVS

- Enfraquecimento **(W)**

$$\frac{\Sigma_1 \vdash_\Gamma \Lambda_1}{\Sigma_2 \vdash_\Gamma \Lambda_2}(\mathbf{W})$$

se $\Sigma_1 \subseteq \Sigma_2$ e $\Lambda_1 \subseteq \Lambda_2$

- Contração **(C)**

Direita

$$\frac{a, a, \Sigma \vdash_\Gamma \Lambda}{a, \Sigma \vdash_\Gamma \Lambda}(\mathbf{C} \vdash)$$

Esquerda

$$\frac{\Sigma \vdash_\Gamma a, a, \Lambda}{\Sigma \vdash_\Gamma a, \Lambda}(\vdash \mathbf{C})$$

Regras de Provas do PVS

- Comuta **(X)**

Direita

$$\frac{\Sigma_1, a, b, \Sigma_2 \vdash_\Gamma \Lambda}{\Sigma_1, b, a, \Sigma_2 \vdash_\Gamma \Lambda}(\mathbf{X} \vdash)$$

Esquerda

$$\frac{\Sigma \vdash_\Gamma \Lambda_1, a, b, \Lambda_2}{\Sigma \vdash_\Gamma \Lambda_1, b, a, \Lambda_2}(\mathbf{X} \vdash)$$

Regras de Provas do PVS

- Comuta **(X)**

Direita

$$\frac{\Sigma_1, a, b, \Sigma_2 \vdash_\Gamma \Lambda}{\Sigma_1, b, a, \Sigma_2 \vdash_\Gamma \Lambda}(\mathbf{X} \vdash)$$

Esquerda

$$\frac{\Sigma \vdash_\Gamma \Lambda_1, a, b, \Lambda_2}{\Sigma \vdash_\Gamma \Lambda_1, b, a, \Lambda_2}(\mathbf{X} \vdash)$$

- Corte **(Corte)**

$$\frac{(\tau(\Gamma)(a) \sim bool)_\Gamma \qquad \Sigma, a \vdash_\Gamma \Lambda \qquad \Sigma \vdash_\Gamma a, \Lambda}{\Sigma \vdash_\Gamma \Lambda}(\mathbf{Corte})$$

Regras de Provas do PVS

- Axiomas Proposicionais **(Ax)**

$$\frac{}{\Sigma, a \vdash_{\Gamma} a, \Lambda}(\mathbf{Ax} \vdash)$$

Regras de Provas do PVS

- Axiomas Proposicionais **(Ax)**

$$\frac{}{\Sigma, a \vdash_\Gamma a, \Lambda}(\mathbf{Ax} \vdash)$$

- TRUE, FALSE

Direita

$$\frac{}{\Sigma, \mathit{FALSE} \vdash_\Gamma \Lambda}(\mathbf{FALSE} \vdash)$$

Esquerda

$$\frac{}{\Sigma \vdash_\Gamma \mathit{TRUE}, \Lambda}(\vdash \mathbf{TRUE})$$

Regras de Provas do PVS

- Contexto

$$\frac{}{\vdash_{\Gamma} a}(\textbf{Fórmula-Contexto}) \quad \text{se } a \text{ é uma fórmula em } \Gamma.$$

Regras de Provas do PVS

- Contexto

$$\frac{}{\vdash_\Gamma a}(\textbf{Fórmula-Contexto}) \quad \text{se } a \text{ é uma fórmula em } \Gamma.$$

$$\frac{}{\vdash_\Gamma s = a}(\textbf{Definição-Contexto}) \quad \text{se } s : T = a \text{ é uma definição de constante em } \Gamma.$$

Regras de Provas do PVS

- Contexto

$$\frac{}{\vdash_\Gamma a}(\textbf{Fórmula-Contexto}) \quad \text{se } a \text{ é uma fórmula em } \Gamma.$$

$$\frac{}{\vdash_\Gamma s = a}(\textbf{Definição-Contexto}) \quad \text{se } s : T = a \text{ é uma definição de constante em } \Gamma.$$

Direita

$$\frac{\Sigma, a \vdash_{\Gamma, a} \Lambda}{\Sigma, a \vdash_\Gamma \Lambda}(\textbf{Contexto} \vdash)$$

Esquerda

$$\frac{\Sigma \vdash_{\Gamma, \neg a} a, \Lambda}{\Sigma \vdash_\Gamma a, \Lambda}(\vdash \textbf{Contexto})$$

Regras de Provas do PVS

- Contexto

$$\frac{}{\vdash_\Gamma a}(\textbf{Fórmula-Contexto}) \quad \text{se } a \text{ é uma fórmula em } \Gamma.$$

$$\frac{}{\vdash_\Gamma s = a}(\textbf{Definição-Contexto}) \quad \text{se } s : T = a \text{ é uma definição de constante em } \Gamma.$$

Direita

$$\frac{\Sigma, a \vdash_{\Gamma, a} \Lambda}{\Sigma, a \vdash_\Gamma \Lambda}(\textbf{Contexto} \vdash)$$

Esquerda

$$\frac{\Sigma \vdash_{\Gamma, \neg a} a, \Lambda}{\Sigma \vdash_\Gamma a, \Lambda}(\vdash \textbf{Contexto})$$

$$\frac{\Sigma \vdash_\Gamma \Lambda}{\Sigma \vdash_{\Gamma'} \Lambda}(\textbf{Contexto-W}) \quad \textit{se } \Gamma \textit{ é um prefixo de } \Gamma'.$$

Regras de Provas do PVS

- Condicional **(IF)**

$$\frac{\Sigma, a, b \vdash_{\Gamma, a} \Lambda \qquad \Sigma, c \vdash_{\Gamma, \neg a} a, \Lambda}{\Sigma, IF(a, b, c) \vdash_{\Gamma} \Lambda}(\mathbf{IF} \vdash)$$

Regras de Provas do PVS

- Condicional **(IF)**

$$\frac{\Sigma, a, b \vdash_{\Gamma, a} \Lambda \qquad \Sigma, c \vdash_{\Gamma, \neg a} a, \Lambda}{\Sigma, IF(a, b, c) \vdash_{\Gamma} \Lambda}(\mathbf{IF} \vdash)$$

$$\frac{\Sigma, a \vdash_{\Gamma, a} b, \Lambda \qquad \Sigma \vdash_{\Gamma, \neg a} a, c, \Lambda}{\Sigma \vdash_{\Gamma} IF(a, b, c), \Lambda}(\vdash \mathbf{IF})$$

Regras de Provas do PVS

- Igualdade

Reflexiva

$$\frac{}{\Sigma \vdash_{\Gamma} a = a, \Lambda}(\textbf{Refl})$$

Substituição

$$\frac{a = b, \Sigma[b] \vdash_{\Gamma} \Lambda[b]}{a = b, \Sigma[a] \vdash_{\Gamma} \Lambda[a]}(\textbf{Subs})$$

Regras de Provas do PVS

- Igualdade Booleana

$$\frac{\Sigma[TRUE], a \vdash_{\Gamma} \Lambda[TRUE]}{\Sigma[a], a \vdash_{\Gamma} \Lambda[a]}(\textbf{Subs TRUE})$$

Regras de Provas do PVS

- Igualdade Booleana

$$\frac{\Sigma[TRUE], a \vdash_\Gamma \Lambda[TRUE]}{\Sigma[a], a \vdash_\Gamma \Lambda[a]}(\textbf{Subs TRUE})$$

$$\frac{\Sigma[FALSE], a \vdash_\Gamma \Lambda[FALSE]}{\Sigma[a], a \vdash_\Gamma \Lambda[a]}(\textbf{Subs FALSE})$$

Regras de Provas do PVS

- Igualdade Booleana

$$\frac{\Sigma[TRUE], a \vdash_\Gamma \Lambda[TRUE]}{\Sigma[a], a \vdash_\Gamma \Lambda[a]}(\textbf{Subs TRUE})$$

$$\frac{\Sigma[FALSE], a \vdash_\Gamma \Lambda[FALSE]}{\Sigma[a], a \vdash_\Gamma \Lambda[a]}(\textbf{Subs FALSE})$$

$$\frac{}{\Sigma, TRUE = FALSE \vdash_\Gamma \Lambda}(\textbf{TRUE-FALSE})$$

Regras de Provas do PVS

- Redução
- β redução

$$\frac{}{\vdash_\Gamma (\lambda(x : T) : a)(b) = a[b/x]}(\beta)$$

Regras de Provas do PVS

- Redução
- β redução

$$\frac{}{\vdash_\Gamma (\lambda(x : T) : a)(b) = a[b/x]}(\beta)$$

- π redução

$$\frac{}{\vdash_\Gamma p_i(a_1, a_2) = a_i}(\pi)$$

Regras de Provas do PVS

- Extensionalidade

$$\frac{\Sigma \vdash_{\Gamma, s:A} (f \quad s) =_{B[s/x]} (g \quad s), \Lambda}{\Sigma \vdash_{\Gamma} f =_{[x:A \to B]} g, \Lambda} (\textbf{FunExt}) \;\; \Gamma(s) \;\; \textit{indefinido}$$

Regras de Provas do PVS

- Extensionalidade

$$\frac{\Sigma \vdash_{\Gamma, s:A} (f \quad s) =_{B[s/x]} (g \quad s), \Lambda}{\Sigma \vdash_{\Gamma} f =_{[x:A \to B]} g, \Lambda} (\textbf{FunExt}) \;\; \Gamma(s) \;\; \textit{indefinido}$$

$$\frac{\Sigma \vdash_{\Gamma} p_1(a) =_{T_1} p_1(b), \Lambda \quad \Sigma \vdash_{\Gamma} p_2(a) =_{T_2[(p_1 \; a)/x]} p_2(b), \Lambda}{\Sigma \vdash_{\Gamma} a =_{[x:T_1 T_2]} b, \Lambda} (\textbf{TunExt})$$

Regras de Provas do PVS

- Extensionalidade

$$\frac{\Sigma \vdash_{\Gamma, s:A} (f \quad s) =_{B[s/x]} (g \quad s), \Lambda}{\Sigma \vdash_{\Gamma} f =_{[x:A \to B]} g, \Lambda}(\textbf{FunExt}) \;\; \Gamma(s) \;\; \textit{indefinido}$$

$$\frac{\Sigma \vdash_{\Gamma} p_1(a) =_{T_1} p_1(b), \Lambda \quad \Sigma \vdash_{\Gamma} p_2(a) =_{T_2[(p_1 \; a)/x]} p_2(b), \Lambda}{\Sigma \vdash_{\Gamma} a =_{[x:T_1 T_2]} b, \Lambda}(\textbf{TunExt})$$

- Restrição de Tipo

$$\frac{\tau(\Gamma)(a) = A \qquad \pi(A)(a), \Sigma \vdash_{\Gamma} \Lambda}{\Sigma \vdash_{\Gamma} \Lambda}(\textbf{TipoPred})$$

Aplicações

O PVS já foi aplicado:

- para demonstrar teoremas clássicos em análise matemática;
- para sistemas críticos, por exemplo, para desenvolvimento de software para aeronaves;
- para verificação de sistemas operacionais, algoritmos distribuídos.

Conclusão

A essência semântica da linguagem é um lambda cálculo tipado com funções simples, subtipos, tipos dependentes, teorias e expressões condicionais.

Os subtipos e tipos com dependência são uma característica crucial da linguagem.

Referências

- Sam Owre and Natarajan Shankar. *The formal semantics of PVS*. Technical Report SRI-CSL-97-2, Computer Science Laboratory, SRI International, Menlo Park, CA, August 1997.
- N. Shankar, S. Owre, J. M. Rushby, and D. W. J. Stringer-Calvert. *PVS Prover Guide*. Computer Science Laboratory, SRI International, Menlo Park, CA, September 1999.
- S. Owre, N. Shankar, J. M. Rushby, and D. W. J. Stringer-Calvert. *PVS System Guide*. Computer Science Laboratory, SRI International, Menlo Park, CA, September 1999.
- V. A. Fernandes, *Detecção e Resolução de Conflitos de Tráfego Aéreo*, Dissertação de Mestrado, Departamento de Matemática, Universidade de Brasília, 2004.
- G. Dowek, C. Muñoz, and A. Geser, *Tactical Conflict Detection and Resolution in a 3-D Airspace*, Proceedings of the Fourth International Air Traffic Management RD Seminar ATM 2001. Extended version available as ICASE Report 2001-7, 2001.

Referências

- V. A. Fernandes, *Detecção e Resolução de Conflitos de Tráfego Aéreo*, Dissertação de Mestrado, Departamento de Matemática, Universidade de Brasília, 2004.
- G. Dowek, C. Muñoz, and A. Geser, *Tactical Conflict Detection and Resolution in a 3-D Airspace*, Proceedings of the Fourth International Air Traffic Management RD Seminar ATM 2001. Extended version available as ICASE Report 2001-7, 2001.
- J. Maddalon, R. Butler, A. Geser and C. Muñoz, *Formal Verification of a Conflict Resolution and Recovery Algorithm*, NASA/TP-2004-213015, 2004.
- C. Muñoz, R. Butler, V. Carreño, and G. Dowek, *Formal verification of conflict detection algorithms, Software Tools for Technology Transfer*, 2003. Extended version available as NASA/TM-2001-210864,2001.
- N. Shankar, S. Owre, and J. M. Rushby. *PVS Tutorial*. Computer Science Laboratory, SRI International, Menlo Park, CA, February 1993.

Referências

- S. Owre, J. M. Rushby, and N. Shankar. *PVS: A prototype verification system*. In Deepak Kapur, editor, *11th International Conference on Automated Deduction (CADE)*, volume 607 of *Lecture Notes in Artificial Intelligence*, pages 748–752, Saratoga, NY, June 1992. Springer-Verlag.
- S. Owre, N. Shankar, J. M. Rushby, and D. W. J. Stringer-Calvert. *PVS Language Reference* Computer Science Laboratory, SRI International, Menlo Park, CA, September 1999.
- N. Shankar, S. Owre, J. M. Rushby, and D. W. J. Stringer-Calvert. *PVS Prover Guide*. Computer Science Laboratory, SRI International, Menlo Park, CA, September 1999.
- S. Owre, N. Shankar, J. M. Rushby, and D. W. J. Stringer-Calvert. *PVS System Guide*. Computer Science Laboratory, SRI International, Menlo Park, CA, September 1999.
- S. Owre and N. Shankar. *The formal semantics of PVS*. Technical Report SRI-CSL-97-2, Computer Science Laboratory, SRI International, Menlo Park, CA, August 1997.

Referências

- J. Rushby, S. Owre, and N. Shankar. *Subtypes for specifications: Predicate subtyping in PVS*. IEEE Transactions on Software Engineering, 24(9):709–720, September 1998.
- N. Shankar and S. Owre *Principles and pragmatics of subtyping in PVS*. In Didier Bert, Christine Choppy, and Peter Mosses, editors, Recent Trends in Algebraic Development Techniques, WADT '99, volume 1827 of Lecture Notes in Computer Science, pages 37–52, Toulouse, France, September 1999. Springer-Verlag.
- J. Crow, S. Owre, J. Rushby, N. Shankar, and Mandayam Srivas. *A tutorial introduction to PVS*. Presented at WIFT '95: Workshop on Industrial-Strength Formal Specification Techniques, Boca Raton, Florida, April 1995. Available, with specification files, at `http://www.csl.sri.com/wift-tutorial.html`.