Semantic Models for a Logic of Partial Functions

Matthew Lovert

Joint Work With Cliff Jones
Newcastle University

September 13th 2010

Introduction

- Terms that involve the application of partial functions and operators can fail to denote
- Classical (two-valued) logic has no meaning for non-denoting logical values
- The Logic of Partial Functions (LPF) is used to reason about propositions that include terms that can fail to denote
- Interested in providing a mechanisation of LPF
- Semantic formalisations for LPF:
    - Structural Operational Semantics
    - Denotational Semantics

Outline

Outline

Partial Functions

- **Total Function**: A function which produces a result for every argument within its domain
- **Partial Function**: A function which may not produce a result for some argument(s) within its domain:
  - The application of a partial function may lead to a non-denoting term
- Partial functions and operators arise frequently in program specifications:
  - Division
  - Taking the head of a list
  - Recursive function definitions
  - ...

Partial Functions Examples

The *zero* Function

$$zero : \mathbb{Z} \rightarrow \mathbb{Z}$$

$$zero(i) \quad \triangleq \quad \textbf{if } i = 0 \textbf{ then } 0 \textbf{ else } zero(i - 1)$$

Partial Functions Examples

The *zero* Function

$$zero : \mathbb{Z} \rightarrow \mathbb{Z}$$

$$zero(i) \quad \triangleq \quad \textbf{if } i = 0 \textbf{ then } 0 \textbf{ else } zero(i - 1)$$

Property 1

$$\forall i \in \mathbb{Z} \cdot i \geq 0 \;\Rightarrow\; zero(i) = 0$$

Partial Functions Examples

The *zero* Function

$zero : \mathbb{Z} \rightarrow \mathbb{Z}$

$zero(i) \quad \triangleq \quad \textbf{if } i = 0 \textbf{ then } 0 \textbf{ else } zero(i - 1)$

Property 1

$\forall i \in \mathbb{Z} \cdot i \geq 0 \;\Rightarrow\; \textbf{zero(i)} = 0$

Possible Non-denoting Term

Partial Functions Examples

The *zero* Function

$zero : \mathbb{Z} \rightarrow \mathbb{Z}$

$zero(i) \quad \triangleq \quad \textbf{if } i = 0 \textbf{ then } 0 \textbf{ else } zero(i - 1)$

Property 1

$\forall i \in \mathbb{Z} \cdot i \geq 0 \;\Rightarrow\; \textbf{zero(i) = 0}$

Which Could Lead to a Possible Non-denoting Logical Value

Partial Functions Examples

The *zero* Function

$$zero : \mathbb{Z} \rightarrow \mathbb{Z}$$

$$zero(i) \triangleq \textbf{if } i = 0 \textbf{ then } 0 \textbf{ else } zero(i-1)$$

Property 1

$$\forall i \in \mathbb{Z} \cdot i \geq 0 \Rightarrow zero(i) = 0$$

$$1 \geq 0 \Rightarrow zero(1) = 0$$
$$\rightarrow \textbf{true} \Rightarrow 0 = 0$$
$$\rightarrow \textbf{true} \Rightarrow \textbf{true}$$
$$\rightarrow \textbf{true}$$

Partial Functions Examples

The *zero* Function

$$zero : \mathbb{Z} \to \mathbb{Z}$$

$$zero(i) \triangleq \textbf{if } i = 0 \textbf{ then } 0 \textbf{ else } zero(i-1)$$

Property 1

$$\forall i \in \mathbb{Z} \cdot i \geq 0 \Rightarrow zero(i) = 0$$

$$-1 \geq 0 \Rightarrow zero(-1) = 0$$
$$\to \textbf{false} \Rightarrow \bot_{\mathbb{Z}} = 0$$
$$\to \textbf{false} \Rightarrow \bot_{\mathbb{B}}$$
$$\to \bot_{\mathbb{B}}$$

Partial Functions Examples

The *zero* Function

$$zero : \mathbb{Z} \rightarrow \mathbb{Z}$$

$$zero(i) \quad \triangleq \quad \textbf{if } i = 0 \textbf{ then } 0 \textbf{ else } zero(i - 1)$$

Property 2

$$\forall i \in \mathbb{Z} \cdot zero(i) = 0 \lor zero(-i) = 0$$

Partial Functions Examples

The *zero* Function

$$zero : \mathbb{Z} \rightarrow \mathbb{Z}$$

$$zero(i) \quad \triangleq \quad \textbf{if } i = 0 \textbf{ then } 0 \textbf{ else } zero(i - 1)$$

Property 2

$$\forall i \in \mathbb{Z} \cdot zero(i) = 0 \vee zero(-i) = 0$$

$zero(1) = 0 \vee zero(-1) = 0$
$\rightarrow 0 = 0 \vee \bot_{\mathbb{Z}} = 0$
$\rightarrow \textbf{true} \vee \bot_{\mathbb{B}}$
$\rightarrow \bot_{\mathbb{B}}$

Coping with Non-denoting Terms

- First-Order Predicate Calculus (FOPC):
  - The logical operators and quantifiers have no meaning for non-denoting logical values
- We need some way of coping with non-denoting terms
- John Harrison: Four main approaches to coping with non-denoting terms:
  - Return a value for input outside of the domain
  - Return an arbitrary value
  - Type error
  - Logic of partial terms

Outline

The Logic of Partial Functions

- First-Order Predicate Logic
- Extend the meaning of the logical operators so they can handle non-denoting logical values
- Three-valued logic:
  - **true**
  - **false**
  - **undefined** ($\perp$)
- Blamey's notion of "gaps" in the value space

The Logic of Partial Functions Continued...

- The truth tables are the strongest extension of their classical interpretations

$\vee$	**true**	$\perp_{\mathbb{B}}$	**false**
true	**true**	**true**	**true**
$\perp_{\mathbb{B}}$	**true**	$\perp_{\mathbb{B}}$	$\perp_{\mathbb{B}}$
false	**true**	$\perp_{\mathbb{B}}$	**false**

$\Rightarrow$	**true**	$\perp_{\mathbb{B}}$	**false**
true	**true**	$\perp_{\mathbb{B}}$	**false**
$\perp_{\mathbb{B}}$	**true**	$\perp_{\mathbb{B}}$	$\perp_{\mathbb{B}}$
false	**true**	**true**	**true**

- Parallel evaluation of the operands
- Return a result as soon as enough information becomes available:
  - No contradiction
  - **true** $\vee \perp_{\mathbb{B}}$, $\perp_{\mathbb{B}} \vee$ **true**

The Logic of Partial Functions Continued...

- Equivalences with classical logic:
  - Contrapositive of implication
  - Commutativity of disjunction
  - ...
- Quantifiers
- No law of the excluded middle ($e \vee \neg e$):
  - $zero(-1) = 0 \vee \neg(zero(-1) = 0)$
- Definedness operator (δ):
  - $\delta(e) = e \vee \neg e$

$$\frac{e_1 \vdash e_2}{e_1 \;\Rightarrow\; e_2}$$

The Logic of Partial Functions Continued...

- Equivalences with classical logic:
  - Contrapositive of implication
  - Commutativity of disjunction
  - ...
- Quantifiers
- No law of the excluded middle ($e \vee \neg e$):
  - $zero(-1) = 0 \vee \neg(zero(-1) = 0)$
- Definedness operator (δ):
  - $\delta(e) = e \vee \neg e$

$$\boxed{\Rightarrow\text{-}I}\ \frac{\delta(e_1); e_1 \vdash e_2}{e_1 \Rightarrow e_2}$$

The Logic of Partial Functions Continued...

The *zero* Function

$$zero : \mathbb{Z} \rightarrow \mathbb{Z}$$

$$zero(i) \quad \triangleq \quad \textbf{if } i = 0 \textbf{ then } 0 \textbf{ else } zero(i - 1)$$

Property 1

$$\forall i \in \mathbb{Z} \cdot i \geq 0 \;\Rightarrow\; zero(i) = 0$$

$$\begin{aligned}
& -1 \geq 0 \;\Rightarrow\; zero(-1) = 0 \\
& \rightarrow \textbf{false} \;\Rightarrow\; \bot_{\mathbb{Z}} = 0 \\
& \rightarrow \textbf{false} \;\Rightarrow\; \bot_{\mathbb{B}} \\
& \rightarrow \textbf{true}
\end{aligned}$$

The Logic of Partial Functions Continued...

The *zero* Function

$$zero : \mathbb{Z} \to \mathbb{Z}$$

$$zero(i) \triangleq \textbf{if } i = 0 \textbf{ then } 0 \textbf{ else } zero(i-1)$$

Property 2

$$\forall i \in \mathbb{Z} \cdot zero(i) = 0 \vee zero(-i) = 0$$

$$zero(1) = 0 \vee zero(-1) = 0$$
$$\to 0 = 0 \vee \bot_{\mathbb{Z}} = 0$$
$$\to \textbf{true} \vee \bot_{\mathbb{B}}$$
$$\to \textbf{true}$$

Outline

Expression Constructs

- All expressions must be of the type BOOL or INT
- Quantification only over the integers

$$\mathit{Expr} = \mathit{Value} \mid \mathit{Id} \mid \mathit{Equality} \mid \mathit{Or} \mid \mathit{Exists} \mid \mathit{FuncCall}$$

$$\mathit{Value} = \mathbb{B} \mid \mathbb{Z}$$

$$\mathit{Equality} :: \begin{array}{l} a \colon \mathit{Expr} \\ b \colon \mathit{Expr} \end{array}$$

$$\mathit{Or} :: \begin{array}{l} a \colon \mathit{Expr} \\ b \colon \mathit{Expr} \end{array}$$

$$\mathit{Exists} :: \begin{array}{r} a \colon \mathit{Id} \\ b \colon \mathit{Expr} \end{array}$$

Functions

- Single integer argument
- Return an integer result
- No free variables

$$
\begin{aligned}
\mathit{FuncCall} \;::\; & \mathit{func}: \mathit{Id} \\
& \mathit{arg}: \mathit{Expr}
\end{aligned}
$$

$$
\begin{aligned}
\mathit{Func} \;::\; & \mathit{param}: \mathit{Id} \\
& \mathit{result}: \mathit{Expr}
\end{aligned}
$$

$$\Gamma = \mathit{Id} \xrightarrow{m} \mathit{Func}$$

Context Conditions

- Remove ill-formed expressions and function definitions from consideration in our semantics

$$Type = \textsc{Bool} \mid \textsc{Int}$$

$$Types = Id \xrightarrow{m} Type$$

$$wf\text{-}Func : Func \times Types \times \Gamma \to \mathbb{B}$$

$$wf\text{-}Func(mk\text{-}Func(p, r), vars, \gamma) \quad \triangleq$$
$$\quad wf\text{-}Expr(r, \{p \mapsto \textsc{Int}\}, \gamma) = \textsc{Int}$$

Context Conditions Continued...

$$wf\text{-}Expr : Expr \times Types \times \Gamma \to (Type \mid \textsc{Error})$$

$$wf\text{-}Expr(e, vars, \gamma) \quad \triangleq$$

cases e **of**

$$\ldots \to \ldots$$

$$e \in Id \wedge e \in \mathbf{dom}\ vars \to vars(e)$$

$$mk\text{-}Or(a, b) \to \mathbf{let}\ l = wf\text{-}Expr(a, vars, \gamma)\ \mathbf{in}$$

$$\mathbf{if}\ l = \textsc{Bool} \wedge l = wf\text{-}Expr(b, vars, \gamma)$$

$$\mathbf{then}\ \textsc{Bool}$$

$$\mathbf{else}\ \textsc{Error}$$

$$\ldots \to \ldots$$

others ERROR

end

Outline

Structural Operational Semantics

- Memory store

$$\Sigma = \mathit{Id} \xrightarrow{m} \mathit{Value}$$

- Semantic Relation

$$\xrightarrow{e} : \mathcal{P}((\mathit{Expr} \times \Sigma \times \Gamma) \times \mathit{Expr})$$

- Identifiers

$$\boxed{\mathit{Id\text{-}E}}\ \frac{\mathit{id} \in \mathit{Id}}{(\mathit{id}, \sigma, \gamma) \xrightarrow{e} \sigma(\mathit{id})}$$

Structural Operational Semantics Continued...

$$\boxed{\textit{Equality-L}}\ \frac{(a, \sigma, \gamma) \xrightarrow{e} a'}{(\textit{mk-Equality}(a, b), \sigma, \gamma) \xrightarrow{e} \textit{mk-Equality}(a', b)}$$

$$\boxed{\textit{Equality-R}}\ \frac{(b, \sigma, \gamma) \xrightarrow{e} b'}{(\textit{mk-Equality}(a, b), \sigma, \gamma) \xrightarrow{e} \textit{mk-Equality}(a, b')}$$

Structural Operational Semantics Continued...

$$\text{Equality-L}\ \frac{(a, \sigma, \gamma) \xrightarrow{e} a'}{(mk\text{-}Equality(a, b), \sigma, \gamma) \xrightarrow{e} mk\text{-}Equality(a', b)}$$

$$\text{Equality-R}\ \frac{(b, \sigma, \gamma) \xrightarrow{e} b'}{(mk\text{-}Equality(a, b), \sigma, \gamma) \xrightarrow{e} mk\text{-}Equality(a, b')}$$

$$\text{Equality-E}\ \frac{a \in \mathbb{Z}; b \in \mathbb{Z}}{(mk\text{-}Equality(a, b), \sigma, \gamma) \xrightarrow{e} [\![=]\!](a, b)}$$

Structural Operational Semantics Continued...

$$\text{Or-L}\ \frac{(a, \sigma, \gamma) \xrightarrow{e} a'}{(mk\text{-}Or(a, b), \sigma, \gamma) \xrightarrow{e} mk\text{-}Or(a', b)}$$

$$\text{Or-R}\ \frac{(b, \sigma, \gamma) \xrightarrow{e} b'}{(mk\text{-}Or(a, b), \sigma, \gamma) \xrightarrow{e} mk\text{-}Or(a, b')}$$

Structural Operational Semantics Continued...

$$\boxed{\textit{Or-L}}\ \frac{(a, \sigma, \gamma) \xrightarrow{e} a'}{(\textit{mk-Or}(a, b), \sigma, \gamma) \xrightarrow{e} \textit{mk-Or}(a', b)}$$

$$\boxed{\textit{Or-R}}\ \frac{(b, \sigma, \gamma) \xrightarrow{e} b'}{(\textit{mk-Or}(a, b), \sigma, \gamma) \xrightarrow{e} \textit{mk-Or}(a, b')}$$

$$\boxed{\textit{Or-E1}}\ \frac{}{(\textit{mk-Or}(\mathbf{true}, b), \sigma, \gamma) \xrightarrow{e} \mathbf{true}}$$

$$\boxed{\textit{Or-E2}}\ \frac{}{(\textit{mk-Or}(a, \mathbf{true}), \sigma, \gamma) \xrightarrow{e} \mathbf{true}}$$

$$\boxed{\textit{Or-E3}}\ \frac{}{(\textit{mk-Or}(\mathbf{false}, \mathbf{false}), \sigma, \gamma) \xrightarrow{e} \mathbf{false}}$$

- “copes with gaps”

Structural Operational Semantics Continued...

$$\boxed{\textit{Exists-T}}\ \frac{\exists i \in \mathbb{Z} \cdot (e, \sigma \dagger \{t \mapsto i\}, \gamma) \xrightarrow{e}{}^{*} \textbf{true}}{(\textit{mk-Exists}(t, e), \sigma, \gamma) \xrightarrow{e} \textbf{true}}$$

$$\boxed{\textit{Exists-F}}\ \frac{\forall i \in \mathbb{Z} \cdot (e, \sigma \dagger \{t \mapsto i\}, \gamma) \xrightarrow{e}{}^{*} \textbf{false}}{(\textit{mk-Exists}(t, e), \sigma, \gamma) \xrightarrow{e} \textbf{false}}$$

Structural Operational Semantics Continued...

$$\boxed{\textit{Exists-T}}\ \frac{\exists i \in \mathbb{Z} \cdot (e, \sigma \dagger \{t \mapsto i\}, \gamma) \xrightarrow{e}\!\!* \ \textbf{true}}{(\textit{mk-Exists}(t, e), \sigma, \gamma) \xrightarrow{e} \textbf{true}}$$

$$\boxed{\textit{Exists-F}}\ \frac{\forall i \in \mathbb{Z} \cdot (e, \sigma \dagger \{t \mapsto i\}, \gamma) \xrightarrow{e}\!\!* \ \textbf{false}}{(\textit{mk-Exists}(t, e), \sigma, \gamma) \xrightarrow{e} \textbf{false}}$$

$$\begin{array}{l} \ldots \vee (e, \sigma \dagger \{t \mapsto -1\}, \gamma) \xrightarrow{e} \textbf{false} \vee \\ (e, \sigma \dagger \{t \mapsto 0\}, \gamma) \xrightarrow{e} \textbf{false} \vee \\ (e, \sigma \dagger \{t \mapsto 1\}, \gamma) \xrightarrow{e} \textbf{false} \vee \ldots \end{array}$$

Structural Operational Semantics Continued...

$$\boxed{\mathit{FuncCall\text{-}A}}\ \frac{(\mathit{arg}, \sigma, \gamma) \xrightarrow{e} \mathit{arg}'}{(\mathit{mk\text{-}FuncCall}(\mathit{id}, \mathit{arg}), \sigma, \gamma) \xrightarrow{e} \mathit{mk\text{-}FuncCall}(\mathit{id}, \mathit{arg}')}$$

Structural Operational Semantics Continued...

$$\boxed{\textit{FuncCall-A}}\ \frac{(\textit{arg}, \sigma, \gamma) \xrightarrow{e} \textit{arg}'}{(\textit{mk-FuncCall}(\textit{id}, \textit{arg}), \sigma, \gamma) \xrightarrow{e} \textit{mk-FuncCall}(\textit{id}, \textit{arg}')}$$

$$\boxed{\textit{FuncCall-E}}\ \frac{\textit{arg} \in \mathbb{Z}}{(\textit{mk-FuncCall}(\textit{id}, \textit{arg}), \sigma, \gamma) \xrightarrow{e} \textit{mk-FuncInter}(\gamma(\textit{id}).\textit{result}, \gamma(\textit{id}).\textit{param}, \textit{arg})}$$

Structural Operational Semantics Continued...

$$\textit{FuncCall-A}\ \frac{(arg, \sigma, \gamma) \xrightarrow{e} arg'}{(\textit{mk-FuncCall}(id, arg), \sigma, \gamma) \xrightarrow{e} \textit{mk-FuncCall}(id, arg')}$$

$$\textit{FuncCall-E}\ \frac{arg \in \mathbb{Z}}{(\textit{mk-FuncCall}(id, arg), \sigma, \gamma) \xrightarrow{e} \textit{mk-FuncInter}(\gamma(id).result, \gamma(id).param, arg)}$$

$$\textit{FuncInter-A}\ \frac{(res, \sigma \dagger \{paramid \mapsto param\}, \gamma) \xrightarrow{e} res'}{(\textit{mk-FuncInter}(res, paramid, param), \sigma, \gamma) \xrightarrow{e} \textit{mk-FuncInter}(res', paramid, param)}$$

Structural Operational Semantics Continued...

$$\boxed{\textit{FuncCall-A}}\ \frac{(\textit{arg}, \sigma, \gamma) \xrightarrow{e} \textit{arg}'}{(\textit{mk-FuncCall}(\textit{id}, \textit{arg}), \sigma, \gamma) \xrightarrow{e} \textit{mk-FuncCall}(\textit{id}, \textit{arg}')}$$

$$\boxed{\textit{FuncCall-E}}\ \frac{\textit{arg} \in \mathbb{Z}}{\begin{array}{c}(\textit{mk-FuncCall}(\textit{id}, \textit{arg}), \sigma, \gamma) \xrightarrow{e} \\ \textit{mk-FuncInter}(\gamma(\textit{id}).\textit{result}, \gamma(\textit{id}).\textit{param}, \textit{arg})\end{array}}$$

$$\boxed{\textit{FuncInter-A}}\ \frac{(\textit{res}, \sigma \dagger \{\textit{paramid} \mapsto \textit{param}\}, \gamma) \xrightarrow{e} \textit{res}'}{\begin{array}{c}(\textit{mk-FuncInter}(\textit{res}, \textit{paramid}, \textit{param}), \sigma, \gamma) \xrightarrow{e} \\ \textit{mk-FuncInter}(\textit{res}', \textit{paramid}, \textit{param})\end{array}}$$

$$\boxed{\textit{FuncInter-E}}\ \frac{\textit{res} \in \mathbb{Z}}{(\textit{mk-FuncInter}(\textit{res}, \textit{paramid}, \textit{param}), \sigma, \gamma) \xrightarrow{e} \textit{res}}$$

Denotational Semantics

- Set theoretic definition of the values denoted by expressions

$$\mathcal{E} : \mathcal{P}((\mathit{Expr} \times \Sigma \times \Gamma) \times \mathit{Value})$$

- Defined in parts as

$$\mathcal{E} = \mathcal{E}\mathit{exists} \cup \mathcal{E}\mathit{funccall}$$

Denotational Semantics Continued...

$$
\begin{aligned}
&\mathcal{E}\,exists = \\
&\quad \{((\textit{mk-Exists}(t, e), \sigma, \gamma), \textbf{true}) \mid \\
&\qquad\qquad)\} \cup \\
&\quad \{((\textit{mk-Exists}(t, e), \sigma, \gamma), \textbf{false}) \mid \\
&\qquad\qquad \}
\end{aligned}
$$

Denotational Semantics Continued...

$$
\begin{aligned}
&\mathcal{E}\mathit{exists} = \\
&\quad \{((\mathit{mk\text{-}Exists}(t, e), \sigma, \gamma), \mathbf{true}) \mid \\
&\qquad\qquad (\{(e, \sigma \dagger \{t \mapsto i\}, \gamma) \mid i \in \mathbb{Z}\})\} \cup \\
&\quad \{((\mathit{mk\text{-}Exists}(t, e), \sigma, \gamma), \mathbf{false}) \mid \\
&\qquad\qquad \}
\end{aligned}
$$

Denotational Semantics Continued...

$$
\begin{aligned}
&\mathcal{E}\,exists = \\
&\quad \{((mk\text{-}Exists(t, e), \sigma, \gamma), \mathbf{true}) \mid \\
&\qquad\qquad (\{(e, \sigma \dagger \{t \mapsto i\}, \gamma) \mid i \in \mathbb{Z}\} \lhd \mathcal{E})\} \cup \\
&\quad \{((mk\text{-}Exists(t, e), \sigma, \gamma), \mathbf{false}) \mid \\
&\qquad\qquad \}
\end{aligned}
$$

Denotational Semantics Continued...

$$
\begin{aligned}
&\mathcal{E}\mathit{exists} = \\
&\quad \{((\mathit{mk\text{-}Exists}(t, e), \sigma, \gamma), \mathbf{true}) \mid \\
&\qquad\qquad \mathbf{rng}\,(\{(e, \sigma \dagger \{t \mapsto i\}, \gamma) \mid i \in \mathbb{Z}\} \lhd \mathcal{E})\} \cup \\
&\quad \{((\mathit{mk\text{-}Exists}(t, e), \sigma, \gamma), \mathbf{false}) \mid \\
&\qquad\qquad \}
\end{aligned}
$$

Denotational Semantics Continued...

$$
\begin{array}{l}
\mathcal{E}\mathit{exists} = \\
\quad \{((\mathit{mk\text{-}Exists}(t, e), \sigma, \gamma), \mathbf{true}) \mid \\
\qquad\qquad \mathbf{true} \in \mathbf{rng}\,(\{(e, \sigma \dagger \{t \mapsto i\}, \gamma) \mid i \in \mathbb{Z}\} \lhd \mathcal{E})\} \cup \\
\quad \{((\mathit{mk\text{-}Exists}(t, e), \sigma, \gamma), \mathbf{false}) \mid \\
\qquad\qquad \}
\end{array}
$$

Denotational Semantics Continued...

$$
\begin{aligned}
&\mathcal{E}\,\mathit{exists} = \\
&\quad \{((\mathit{mk\text{-}Exists}(t, e), \sigma, \gamma), \mathbf{true}) \mid \\
&\qquad\qquad \mathbf{true} \in \mathbf{rng}\,(\{(e, \sigma \dagger \{t \mapsto i\}, \gamma) \mid i \in \mathbb{Z}\} \lhd \mathcal{E})\} \cup \\
&\quad \{((\mathit{mk\text{-}Exists}(t, e), \sigma, \gamma), \mathbf{false}) \mid \\
&\qquad\qquad \{((e, \sigma \dagger \{t \mapsto i\}, \gamma), \mathbf{false}) \mid i \in \mathbb{Z}\}\}
\end{aligned}
$$

Denotational Semantics Continued...

$$
\begin{aligned}
&\mathcal{E}\,\mathit{exists} = \\
&\quad \{((\mathit{mk\text{-}Exists}(t,e),\sigma,\gamma),\mathbf{true}) \mid \\
&\qquad\qquad \mathbf{true} \in \mathbf{rng}\,(\{(e,\sigma \dagger \{t \mapsto i\},\gamma) \mid i \in \mathbb{Z}\} \lhd \mathcal{E})\} \cup \\
&\quad \{((\mathit{mk\text{-}Exists}(t,e),\sigma,\gamma),\mathbf{false}) \mid \\
&\qquad\qquad \{((e,\sigma \dagger \{t \mapsto i\},\gamma),\mathbf{false}) \mid i \in \mathbb{Z}\} \subseteq \mathcal{E}\}
\end{aligned}
$$

Denotational Semantics Continued...

$$((\textit{mk-FuncCall}(\textit{zero}, 1), \sigma, \gamma), 0) \in \mathcal{E}$$

$$(\textit{mk-FuncCall}(\textit{zero}, -1), \sigma, \gamma) \notin \textbf{dom}\, \mathcal{E}$$

$$\mathcal{E}\,\textit{funccall} = \{((\textit{mk-FuncCall}(f, \textit{arg}), \sigma, \gamma), \textit{res}) \mid \}$$

- Proofs can be based upon this definition

Denotational Semantics Continued...

$$((\textit{mk-FuncCall}(\textit{zero}, 1), \sigma, \gamma), 0) \in \mathcal{E}$$

$$(\textit{mk-FuncCall}(\textit{zero}, -1), \sigma, \gamma) \notin \mathbf{dom}\, \mathcal{E}$$

$$\begin{aligned} &\mathcal{E}\, \textit{funccall} = \\ &\quad \{((\textit{mk-FuncCall}(f, \textit{arg}), \sigma, \gamma), \textit{res}) \mid \\ &\qquad\qquad ((\textit{arg}, \sigma, \gamma), \textit{arg}') \in \mathcal{E}\} \end{aligned}$$

- Proofs can be based upon this definition

Denotational Semantics Continued...

$$((mk\text{-}FuncCall(zero, 1), \sigma, \gamma), 0) \in \mathcal{E}$$

$$(mk\text{-}FuncCall(zero, -1), \sigma, \gamma) \notin \mathbf{dom}\,\mathcal{E}$$

$$\begin{array}{l} \mathcal{E}\,funccall = \\ \quad \{((mk\text{-}FuncCall(f, arg), \sigma, \gamma), res) \mid \\ \qquad\qquad ((arg, \sigma, \gamma), arg') \in \mathcal{E} \wedge \\ \qquad\qquad ((\gamma(f).result, \sigma \dagger \{\gamma(f).param \mapsto arg'\}, \gamma), res) \in \mathcal{E}\} \end{array}$$

- Proofs can be based upon this definition

References

J. H. Cheng and C. B. Jones
On the usability of logics which handle partial functions.
In C. Morgan and J. C. P. Woodcock, editors, 3rd Refinement Workshop, 51–69, 1991.

C. B. Jones.
Reasoning about partial functions in the formal development of programs.
Electronic Notes in Theoretical Computer Science, 145:3–25, 2006.

S. C. Kleene
Introduction to Metamathematics.
Van Nostrad, 1952

Thank you.

Any Questions?