Progress and Preservation Proofs for the Expressions "`iszero`" and "`pred`" in the Arith Language

Zayd Hammoudeh
(zayd.hammoudeh@sjsu.edu)

April 13, 2016

Contents

List of Figures

1 Arith Language

Arith is a basic language; its expressions, values, and types are enumerated in figure 1. Arith's small-step, evaluation order semantics are defined in figure 2 while Arith's type rules are enumerated in figure 3.

$e ::=$		*Expressions*
	`true`	Boolean True
	`false`	Boolean False
	`0`	Integer Value 0
	`succ` (e)	Successor Expressions
	`pred` (e)	Predecessor Expressions
	`iszero` (e)	Zero Value Check Expressions
	`if` (e) `then` (e) `else` (e)	Conditional Expressions
$v ::=$		*Values*
	i	integer values
	b	boolean values
$T ::=$		*Types*
	`Bool`	Boolean Type
	`Int`	Integer Type

Figure 1: The Arith language

Evaluation Rules: $\boxed{e \to e'}$

[E-Succ-Ctxt] $$\frac{e_1 \to e_1'}{\texttt{succ}\ (e_1) \to \texttt{succ}\ (e_1')}$$

[E-Succ] $$\frac{i' = i + 1}{\texttt{succ}\ (i) \to i'}$$

[E-Pred-Ctxt] $$\frac{e_1 \to e_1'}{\texttt{pred}\ (e_1) \to \texttt{pred}\ (e_1')}$$

[E-Pred] $$\frac{i' = i - 1}{\texttt{pred}\ (i) \to i'}$$

[E-IsZero-Ctxt] $$\frac{e_1 \to e_1'}{\texttt{iszero}\ (e_1) \to \texttt{iszero}\ (e_1')}$$

[E-IsZero-Z] $$\texttt{iszero}\ (\texttt{0}) \to \texttt{true}$$

[E-IsZero-NZ] $$\frac{i \neq 0}{\texttt{iszero}\ (i) \to \texttt{false}}$$

[E-If-Ctxt] $$\frac{e_1 \to e_1'}{\texttt{if}\ (e_1)\ \texttt{then}\ (e_2)\ \texttt{else}\ (e_3) \to \texttt{if}\ (e_1')\ \texttt{then}\ (e_2)\ \texttt{else}\ (e_3)}$$

[E-If-True] $$\texttt{if}\ (\texttt{true})\ \texttt{then}\ (e_2)\ \texttt{else}\ (e_3) \to e_2$$

[E-If-False] $$\texttt{if}\ (\texttt{false})\ \texttt{then}\ (e_2)\ \texttt{else}\ (e_3) \to e_3$$

Figure 2: Small-Step, Evaluation Order Semantics Semantics for the Arith Language

Type Rules: $\boxed{e : T}$

$$[\text{T-True}] \quad \texttt{true} : \texttt{Bool}$$

$$[\text{T-False}] \quad \texttt{false} : \texttt{Bool}$$

$$[\text{T-Int}] \quad i : \texttt{Int}$$

$$[\text{T-Succ}] \quad \frac{e_1 : \texttt{Int}}{\texttt{succ}\ (e)_1 : \texttt{Int}}$$

$$[\text{T-Pred}] \quad \frac{e_1 : \texttt{Int}}{\texttt{pred}\ (e)_1 : \texttt{Int}}$$

$$[\text{T-IsZero}] \quad \frac{e_1 : \texttt{Int}}{\texttt{iszero}\ (e)_1 : \texttt{Bool}}$$

$$[\text{T-If}] \quad \frac{e_1 : \texttt{Bool}, \quad e_2 : T, \quad e_3 : T}{\texttt{if}\ (e_1)\ \texttt{then}\ (e_2)\ \texttt{else}\ (e_3) : T}$$

Figure 3: Type Rules for the Arith Language

2 Progress

In semantics context, "progress" entails that a well-type expression will not "get stuck." Figure 4 is the formal, theoretical definition of progress.

Given $e : T$, then either:

1. e is a value.
2. There exists an e' such that: $e \to e'$.

Figure 4: Formal Definition of the Progress Theorem

The following subsections are the formal proofs of progress for the type rules in figure 3.

2.1 Proving Progress for Boolean and Integer Values

Figure 4 establish that progress is achieved if an expression e is a value. Hence, by this criterion, type rules [T-TRUE] , [T-FALSE] , and [T-INT] all hold.

2.2 Proving Progress for the If Expression

Figure 5 is the proof of progress for the If expression in the Arith Language.

Given:

$$e = \texttt{if}\ (e_1)\ \texttt{then}\ (e_2)\ \texttt{else}\ (e_3)\ e_1 : \texttt{Bool},\quad e_2 : T,\quad e_3 : T$$

Then:

$$e : T$$

By induction, an expression e_1 must be a value or $\exists e_1$ such that:

1. If $e_1 = i$ where "i", then $e \rightarrow i'$ where:

$$i' = i + 1$$

2. Otherwise, $e_1 \rightarrow e_1'$ which means that [E-IF-CTXT] applies (as shown below) which by induction holds.

$$\succ e_1 \rightarrow \texttt{if}\ (e_1')\ \texttt{then}\ (e_2)\ \texttt{else}\ (e_3)$$

Figure 5: Proof of Progress for the If Expression

2.3 Proving Progress for the Successor Expression

Figure 3 shows the type rule for the successor expression. The proof of progress for this expression is shown in figure 6.

Given:

$$e = \texttt{succ}\ (e_1)$$
$$e_1 : \texttt{Int}$$

Then:

$$T = \texttt{Int}$$

By induction, e_1 must either be a value or $\exists e_1'$ such that:

1. If e_1 is a value, then either [E-If-True] or [E-If-False] applies since $e_1 : \texttt{Bool}$.
2. Otherwise [E-If-Ctxt] applies (as shown below) which by induction holds.

$$e \rightarrow \texttt{if}\ (e_1')\ \texttt{then}\ (e_2)\ \texttt{else}\ (e_3)$$

Figure 6: Proof of Progress for the Successor Expression