CIS 500 — Software Foundations

Midterm I

Answer key

October 8, 2003

Inductive Definitions

Review: Recall that the function *size*, which calculates the total number of nodes in the abstract syntax tree of a term in the language of arithmetic and boolean expressions, can be written either in standard recursive function notation

$$
\begin{array}{lcl}
\mathit{size}(\texttt{true}) & = & 1 \\
\mathit{size}(\texttt{false}) & = & 1 \\
\mathit{size}(\texttt{0}) & = & 1 \\
\mathit{size}(\texttt{succ}\ \texttt{t}_1) & = & \mathit{size}(\texttt{t}_1) + 1 \\
\mathit{size}(\texttt{pred}\ \texttt{t}_1) & = & \mathit{size}(\texttt{t}_1) + 1 \\
\mathit{size}(\texttt{iszero}\ \texttt{t}_1) & = & \mathit{size}(\texttt{t}_1) + 1 \\
\mathit{size}(\texttt{if}\ \texttt{t}_1\ \texttt{then}\ \texttt{t}_2\ \texttt{else}\ \texttt{t}_3) & = & \mathit{size}(\texttt{t}_1) + \mathit{size}(\texttt{t}_2) + \mathit{size}(\texttt{t}_3) + 1
\end{array}
$$

or, equivalently, as the least relation closed under the following inference rules:

$$(\texttt{true}, 1) \in \mathit{size}$$

$$(\texttt{false}, 1) \in \mathit{size}$$

$$(\texttt{0}, 1) \in \mathit{size}$$

$$\frac{(\texttt{t}_1, n) \in \mathit{size}}{((\texttt{succ}\ \texttt{t}_1), (n+1)) \in \mathit{size}}$$

$$\frac{(\texttt{t}_1, n) \in \mathit{size}}{((\texttt{pred}\ \texttt{t}_1), (n+1))) \in \mathit{size}}$$

$$\frac{(\texttt{t}_1, n) \in \mathit{size}}{((\texttt{iszero}\ \texttt{t}_1), (n+1)) \in \mathit{size}}$$

$$\frac{(\texttt{t}_1, n_1) \in \mathit{size} \qquad (\texttt{t}_2, n_2) \in \mathit{size} \qquad (\texttt{t}_3, n_3) \in \mathit{size}}{((\texttt{if}\ \texttt{t}_1\ \texttt{then}\ \texttt{t}_2\ \texttt{else}\ \texttt{t}_3), (n_1 + n_2 + n_3 + 1)) \in \mathit{size}}$$

1. (7 points) Suppose that we define another relation, *weird*, as the least relation closed under the following rules:

$$(\texttt{true}, 1) \in \textit{weird} \tag{W1}$$

$$(\texttt{0}, 1) \in \textit{weird} \tag{W2}$$

$$(\texttt{0}, 8) \in \textit{weird} \tag{W3}$$

$$\frac{(\texttt{t}_1, n) \in \textit{weird}}{((\texttt{succ}\ \texttt{t}_1), (n+1)) \in \textit{weird}} \tag{W4}$$

$$\frac{(\texttt{t}_1, n) \in \textit{weird}}{((\texttt{succ (succ}\ \texttt{t}_1\texttt{)}), (n+2)) \in \textit{weird}} \tag{W5}$$

$$\frac{((\texttt{succ (succ}\ \texttt{t}_1\texttt{)}), n) \in \textit{weird}}{((\texttt{pred}\ \texttt{t}_1), n) \in \textit{weird}} \tag{W6}$$

$$\frac{(\texttt{iszero (iszero}\ \texttt{t}_1\texttt{)}), n) \in \textit{weird}}{((\texttt{iszero}\ \texttt{t}_1), (n+1)) \in \textit{weird}} \tag{W7}$$

$$\frac{(\texttt{t}_1, n_1) \in \textit{weird} \qquad (\texttt{t}_2, n_2) \in \textit{weird} \qquad (\texttt{t}_3, n_3) \in \textit{weird}}{((\texttt{if}\ \texttt{t}_1\ \texttt{then}\ \texttt{t}_2\ \texttt{else}\ \texttt{t}_3), (n_1 + n_2 + n_3 + 1)) \in \textit{weird}} \tag{W8}$$

Which of the following pairs are related by *weird*? Write *Yes* (if related) or *No* (if not) next to each pair.

(a) $(\texttt{true}, 1)$ *Answer: Yes*

(b) $((\texttt{if true then 0 else 0}), 18)$ *Answer: Yes*

(c) $((\texttt{if true then 0 else 0}), 11)$ *Answer: Yes*

(d) $((\texttt{pred 0}), 3)$ *Answer: Yes*

(e) $((\texttt{succ (succ true)}), 3)$ *Answer: Yes*

(f) $((\texttt{iszero 0}), 3)$ *Answer: No*

(g) $((\texttt{pred false}), 3)$ *Answer: No*

Grading scheme: One point for each item.

2. (5 points) Here are the same rules again:

$$(\texttt{true},\ 1) \in \textit{weird} \tag{W1}$$

$$(\texttt{0},\ 1) \in \textit{weird} \tag{W2}$$

$$(\texttt{0},\ 8) \in \textit{weird} \tag{W3}$$

$$\frac{(\texttt{t}_1,\ n) \in \textit{weird}}{((\texttt{succ}\ \texttt{t}_1),\ (n+1)) \in \textit{weird}} \tag{W4}$$

$$\frac{(\texttt{t}_1,\ n) \in \textit{weird}}{((\texttt{succ}\ \texttt{(succ}\ \texttt{t}_1\texttt{)}),\ (n+2)) \in \textit{weird}} \tag{W5}$$

$$\frac{((\texttt{succ}\ \texttt{(succ}\ \texttt{t}_1\texttt{)}),\ n) \in \textit{weird}}{((\texttt{pred}\ \texttt{t}_1),\ n) \in \textit{weird}} \tag{W6}$$

$$\frac{(\texttt{iszero}\ \texttt{(iszero}\ \texttt{t}_1\texttt{)}),\ n) \in \textit{weird}}{((\texttt{iszero}\ \texttt{t}_1),\ (n+1)) \in \textit{weird}} \tag{W7}$$

$$\frac{(\texttt{t}_1,\ n_1) \in \textit{weird} \qquad (\texttt{t}_2,\ n_2) \in \textit{weird} \qquad (\texttt{t}_3,\ n_3) \in \textit{weird}}{((\texttt{if}\ \texttt{t}_1\ \texttt{then}\ \texttt{t}_2\ \texttt{else}\ \texttt{t}_3),\ (n_1+n_2+n_3+1)) \in \textit{weird}} \tag{W8}$$

Which of these rules can be *dropped* without changing the relation that they define? (I.e., what is the smallest subset of the above rules such that the least relation closed under this subset is the same as the least relation closed under all the rules?)

Write the name(s) of the unnecessary rule(s) here:

Answer: W5 and W7

gradingschemeTwo points off for each incorrect rule; two points off for each missing rule.

Typed Arithmetic Expressions

The full definition of the language of typed arithmetic and boolean expressions is reproduced, for your reference, at the end of the exam. Some properties enjoyed by this language are listed at the bottom of this page.

3. (5 points) Suppose we add a new rule

$$\texttt{if true then } \texttt{t}_2 \texttt{ else } \texttt{t}_3 \longrightarrow \texttt{t}_3 \qquad \text{(E-FUNNY1)}$$

to the ones given at the end of the exam. Do these properties of the original system continue to hold in the presence of this rule?

For each property that *becomes false* when the proposed rule is added to the system, state the name of the property and give a brief counter-example demonstrating that it does not hold in the presence of the new rule.

Answer:

Determinism: *`if true then 0 else succ(0)` can now evaluate in one step to either `0` or `succ 0`.*

Uniqueness: *Same counter-example.*

Grading scheme:

- *One point off for an "almost correct" but slightly confused counter-example.*
- *Two points off for completely mangled or incomprehensible counter-example.*
- *Two points off for missing a property that becomes false.*
- *Two points off for each property that is incorrectly identified as becoming false.*
- *-3 for correct answer but no counter-examples*
- *No credit for no answer.*

Properties:

Determinism (of one-step evaluation): if $\texttt{t} \longrightarrow \texttt{t}'$ and $\texttt{t} \longrightarrow \texttt{t}''$, then $\texttt{t}' = \texttt{t}''$.

Uniqueness (of normal forms): If $\texttt{t} \longrightarrow^* \texttt{u}$ and $\texttt{t} \longrightarrow^* \texttt{u}'$, where $\texttt{u}$ and $\texttt{u}'$ are both normal forms, then $\texttt{u} = \texttt{u}'$.

Termination (of evaluation): For every term $\texttt{t}$ there is some normal form $\texttt{t}'$ such that $\texttt{t} \longrightarrow^* \texttt{t}'$.

Progress: If $\texttt{t} : \texttt{T}$, then either $\texttt{t}$ is a value or else there is some $\texttt{t}'$ with $\texttt{t} \longrightarrow \texttt{t}'$.

Preservation: If $\texttt{t} : \texttt{T}$ and $\texttt{t} \longrightarrow \texttt{t}'$, then $\texttt{t}' : \texttt{T}$.

4. (5 points) Suppose instead that we add this rule:

$$\frac{\mathtt{t}_2 \longrightarrow \mathtt{t}_2'}{\mathtt{if}\ \mathtt{t}_1\ \mathtt{then}\ \mathtt{t}_2\ \mathtt{else}\ \mathtt{t}_3 \longrightarrow \mathtt{if}\ \mathtt{t}_1\ \mathtt{then}\ \mathtt{t}_2'\ \mathtt{else}\ \mathtt{t}_3} \qquad \text{(E-FUNNY2)}$$

Answer in the same format as problem 3: For each property that becomes false when the proposed rule is added, write its name and give a brief counter-example. The properties are listed again at the bottom of this page for easy reference.

(N.b.: In this problem, we are considering the effect of adding the rule E-FUNNY2 to the original language of typed arithmetic expressions, *not* including the rule E-FUNNY1 proposed in problem 3.)

Answer:

> **Determinism***: `if false then (pred 0) else (succ 0)` can now evaluate in one step to either `succ 0` or `if false then 0 else (succ 0)`. (There were several other correct sorts of counter-examples for this one.)*

Grading scheme: -3 for not identifying determinism as becoming false; -2 for not providing a counter-example (or for a mangled counter-example); -1 for a somewhat-right counter-example. -2 each for incorrectly identifying other properties as becoming false.

Properties:

> **Determinism** (of one-step evaluation): if $\mathtt{t} \longrightarrow \mathtt{t}'$ and $\mathtt{t} \longrightarrow \mathtt{t}''$, then $\mathtt{t}' = \mathtt{t}''$.
>
> **Uniqueness** (of normal forms): If $\mathtt{t} \longrightarrow^* \mathtt{u}$ and $\mathtt{t} \longrightarrow^* \mathtt{u}'$, where $\mathtt{u}$ and $\mathtt{u}'$ are both normal forms, then $\mathtt{u} = \mathtt{u}'$.
>
> **Termination** (of evaluation): For every term $\mathtt{t}$ there is some normal form $\mathtt{t}'$ such that $\mathtt{t} \longrightarrow^* \mathtt{t}'$.
>
> **Progress:** If $\mathtt{t} : \mathtt{T}$, then either $\mathtt{t}$ is a value or else there is some $\mathtt{t}'$ with $\mathtt{t} \longrightarrow \mathtt{t}'$.
>
> **Preservation:** If $\mathtt{t} : \mathtt{T}$ and $\mathtt{t} \longrightarrow \mathtt{t}'$, then $\mathtt{t}' : \mathtt{T}$.

5. (5 points) Suppose instead that we add this rule to the original languge of typed arithmetic expressions:

$$\texttt{pred false} \longrightarrow \texttt{pred (pred false)} \qquad \text{(E-FUNNY3)}$$

Do the properties of the original system continue to hold in the presence of this rule?

Answer in the same format as the previous two problems.

Answer:

Termination: `pred false` *diverges.*

Grading scheme: -2 for each property wrong (to a maximum of five); -2 for giving the correct property that changed but not providing a counter-example.

Properties:

Determinism (of one-step evaluation): if $\texttt{t} \longrightarrow \texttt{t}'$ and $\texttt{t} \longrightarrow \texttt{t}''$, then $\texttt{t}' = \texttt{t}''$.

Uniqueness (of normal forms): If $\texttt{t} \longrightarrow^* \texttt{u}$ and $\texttt{t} \longrightarrow^* \texttt{u}'$, where $\texttt{u}$ and $\texttt{u}'$ are both normal forms, then $\texttt{u} = \texttt{u}'$.

Termination (of evaluation): For every term $\texttt{t}$ there is some normal form $\texttt{t}'$ such that $\texttt{t} \longrightarrow^* \texttt{t}'$.

Progress: If $\texttt{t} : \texttt{T}$, then either $\texttt{t}$ is a value or else there is some $\texttt{t}'$ with $\texttt{t} \longrightarrow \texttt{t}'$.

Preservation: If $\texttt{t} : \texttt{T}$ and $\texttt{t} \longrightarrow \texttt{t}'$, then $\texttt{t}' : \texttt{T}$.

6. (5 points) Suppose instead that we add this rule to the original languge of typed arithmetic expressions:

$$\texttt{0} : \texttt{Bool} \qquad \text{(T-FUNNY4)}$$

Do the properties of the original system continue to hold in the presence of this rule?

Answer in the same format as the previous three problems.

Answer:

Progress: *`if 0 then true else true` has type `Bool`, is a normal form, and is not a value.*

Grading scheme: -3 for saying Preservation fails and Progress is preserved; -2 for for claiming other properies become false; -4 for claiming that no properties become false.

Properties:

Determinism (of one-step evaluation): if $\texttt{t} \longrightarrow \texttt{t}'$ and $\texttt{t} \longrightarrow \texttt{t}''$, then $\texttt{t}' = \texttt{t}''$.

Uniqueness (of normal forms): If $\texttt{t} \longrightarrow^* \texttt{u}$ and $\texttt{t} \longrightarrow^* \texttt{u}'$, where $\texttt{u}$ and $\texttt{u}'$ are both normal forms, then $\texttt{u} = \texttt{u}'$.

Termination (of evaluation): For every term $\texttt{t}$ there is some normal form $\texttt{t}'$ such that $\texttt{t} \longrightarrow^* \texttt{t}'$.

Progress: If $\texttt{t} : \texttt{T}$, then either $\texttt{t}$ is a value or else there is some $\texttt{t}'$ with $\texttt{t} \longrightarrow \texttt{t}'$.

Preservation: If $\texttt{t} : \texttt{T}$ and $\texttt{t} \longrightarrow \texttt{t}'$, then $\texttt{t}' : \texttt{T}$.

7. (5 points) Suppose instead that we add this rule to the original languge of typed arithmetic expressions:

$$\texttt{pred}\ \texttt{0} : \texttt{Bool} \qquad \text{(T-FUNNY5)}$$

Do the properties of the original system continue to hold in the presence of this rule?

Answer in the same format as the previous three problems.

Answer:

Preservation*: `pred 0` has type `Bool` and evaluates in one step to `0`, which does not have type `Bool`.*

Properties:

Determinism (of one-step evaluation): if $t \longrightarrow t'$ and $t \longrightarrow t''$, then $t' = t''$.

Uniqueness (of normal forms): If $t \longrightarrow^* u$ and $t \longrightarrow^* u'$, where u and u' are both normal forms, then $u = u'$.

Termination (of evaluation): For every term t there is some normal form t' such that $t \longrightarrow^* t'$.

Progress: If $t : T$, then either t is a value or else there is some t' with $t \longrightarrow t'$.

Preservation: If $t : T$ and $t \longrightarrow t'$, then $t' : T$.

Untyped lambda-calculus

8. (12 points) Write down the normal forms of the following λ-terms, or “none” if a term has no normal form:

   (a) `(λs. λz. s (s z)) (λx. x)`
   Answer: `λz. (λx. x) ((λx. x) z)`

   (b) `(λx. x) (λx. x) (λx. x) (λx. x x)`
   Answer: `λx. x x`

   (c) `(λt. λf. t) (λt. λf. f) (λt. λf. t) (λt. λf. f)`
   Answer: `λf. f`

   (d) `(λx. x x) (λx. x x) (λx. x)`
   Answer: None

   (e) `λx. (λx. x x) (λx. x x)`
   Answer: `λx. (λx. x x) (λx. x x)`

   (f) `(λf. (λx. f (λy. x x y)) (λx. f (λy. x x y))) (λs. λz. z)`
   Answer: `λz. z`

   Grading scheme: Binary. 2 points each.

9. (6 points) Here are the definitions of the Church numerals and the basic operations over them from Chapter 5 of TAPL:

```
c₀ = λs. λz. z;
c₁ = λs. λz. s z;
c₂ = λs. λz. s (s z);
c₃ = λs. λz. s (s (s z));

scc = λn. λs. λz. s (n s z);
plus = λm. λn. λs. λz. m s (n s z);
iszro = λm. m (λx. fls) tru;

tru = λt. λf. t
fls = λt. λf. f
and = λb. λc. b c fls;
not = λb. b fls tru

pair = λf.λs.λb. b f s;
fst = λp. p tru;
snd = λp. p fls;

zz = pair c₀ c₀;
ss = λp. pair (snd p) (plus c₁ (snd p));
prd = λm. fst (m ss zz);
```

Use these combinators to fill in the blank in the following definition to yield a function `less` that, when applied to two church numerals, c_m and c_n, returns `tru` if $m < n$ and otherwise returns `fls`. For example, `less` c_2 c_4 should evaluate to `tru`, while `less` c_3 c_1 and `less` c_3 c_3 should both evaluate to `fls`.

Your answer should use only the combinators defined above (plus applications and variables). Do not write any explicit λ-abstractions.

Answer:

```
less = λm. λn. (not (iszro (m prd n)))
```

Grading scheme: One point off for missing parenthesis or wrong order of evaluation; One point off for swapping m and n; Two points off for neglecting the case of m = n; Three points off for violating the rules (No lambda abstractions).

10. (6 points) Recall the Church encoding of lists from the solution to homework 4.

```
nil = λc. λn. n;
cons = λh. λt. λc. λn. c h (t c n);
head = λl. l (λh.λt.h) fls;
tail = λl.
         fst (l (λx. λp. pair (snd p) (cons x (snd p)))
                (pair nil nil));
isnil = λl. l (λh.λt.fls) tru;
```

Fill in the blanks in the following definition to yield a lambda term `map` that takes a term `l` representing a list and a function `f`, applies `f` to each element of `l`, and yields a list of the results (just like the `List.map` function in OCaml). For example,

map scc (cons c_2 (cons c_0 (cons c_1 nil)))

should be equivalent to

(cons c_3 (cons c_1 (cons c_2 nil))).

Your answer should consist entirely of variables and applications—no lambda-abstractions and no uses of any of the combinators defined above.

Answer:

```
map = λl. λf. λc. λn. l (λh. λt. c (f h) t) n
```

Grading scheme: One point off for missing parenthesis or wrong order of evaluation; One point off for each slight mistake if the answer is almost right; Partial credits are generously given if the answer is structurally similar to the correct answer.

Nameless representation of terms

11. (2 points) Suppose we have defined the naming context $\Gamma = \mathtt{a}, \mathtt{b}, \mathtt{c}, \mathtt{d}$. Then one possible "named representation" of the deBruijn term `λ. 1 0 (λ. 1)` would be `λx. d x (λy. x)`.

    Write down a possible named representation for each of the following deBruijn terms.

    (a) `λ. λ. 1 3 0`

    Answer: `λx. λy. x c y`

    (b) `λ. 3 (λ. 2 1 1) 1`

    Answer: `λx. b (λy. d x x) d`

    Grading scheme: Binary

12. (3 points) Write down (in deBruijn notation) the normal form of the following deBruijn term.

    `(λ. λ. 1 (λ. 1)) (λ. 0)`

    Answer: `λ. (λ. 0) (λ.1)`

    Grading scheme: Binary

Behavioral Equivalence

13. (14 points) Recall the definitions of observational and behavioral equivalence from the lecture notes:

- Two terms `s` and `t` are *observationally equivalent* iff either both are normalizable (i.e., they reach a normal form after a finite number of evaluation steps) or both are divergent.
- Terms `s` and `t` are *behaviorally equivalent* iff, for every finite sequence of values $v_1, v_2, \ldots, v_n$, the applications

$$s\ v_1\ v_2\ \ldots\ v_n$$

and

$$t\ v_1\ v_2\ \ldots\ v_n$$

are observationally equivalent.

Recall, also, the following definitions of lambda-terms from the text:

```
c₀ = λs. λz. z;
c₁ = λs. λz. s z;
c₂ = λs. λz. s (s z);
c₃ = λs. λz. s (s (s z));
plus = λm. λn. λs. λz. m s (n s z);
fix = λf. (λx. f (λy. x x y)) (λx. f (λy. x x y));
```

For each of the following pairs of terms, write *Yes* if the terms are behaviorally equivalent and *no* if they are not.

(a) `tru`
`λx. λy. (λz. z) x`
Answer: Yes

(b) `λx. λy. x y`
`λx. x`
Answer: Yes

(c) `plus` c_2 c_1
c_3
Answer: Yes

(d) `λx. λy. x y`
`λx. λy. x (λz. z) y`
Answer: No

(e) `(λx. x x) (λx. x x)`
`(λx. x x x) (λx. x x x)`
Answer: Yes

(f) `(λx. x x) (λx. x x)`
`λx. (λx. x x) (λx. x x)`
Answer: No

(g) `λf. (λx. f (x x)) (λx. f (x x))`
`fix`
Answer: No

Grading scheme: Binary; each item worth two points.

For reference: Boolean and arithmetic expressions

Syntax

`t`	::=		*terms*
		`true`	*constant true*
		`false`	*constant false*
		`if t then t else t`	*conditional*
		`0`	*constant zero*
		`succ t`	*successor*
		`pred t`	*predecessor*
		`iszero t`	*zero test*
`v`	::=		*values*
		`true`	*true value*
		`false`	*false value*
		`nv`	*numeric value*
`nv`	::=		*numeric values*
		`0`	*zero value*
		`succ nv`	*successor value*
`T`	::=		*types*
		`Bool`	*type of booleans*
		`Nat`	*type of numbers*

Evaluation

$$\texttt{if true then } t_2 \texttt{ else } t_3 \longrightarrow t_2 \qquad \text{(E-IfTrue)}$$

$$\texttt{if false then } t_2 \texttt{ else } t_3 \longrightarrow t_3 \qquad \text{(E-IfFalse)}$$

$$\frac{t_1 \longrightarrow t_1'}{\texttt{if } t_1 \texttt{ then } t_2 \texttt{ else } t_3 \longrightarrow \texttt{if } t_1' \texttt{ then } t_2 \texttt{ else } t_3} \qquad \text{(E-If)}$$

$$\frac{t_1 \longrightarrow t_1'}{\texttt{succ } t_1 \longrightarrow \texttt{succ } t_1'} \qquad \text{(E-Succ)}$$

$$\texttt{pred 0} \longrightarrow \texttt{0} \qquad \text{(E-PredZero)}$$

$$\texttt{pred (succ } nv_1\texttt{)} \longrightarrow nv_1 \qquad \text{(E-PredSucc)}$$

$$\frac{t_1 \longrightarrow t_1'}{\texttt{pred } t_1 \longrightarrow \texttt{pred } t_1'} \qquad \text{(E-Pred)}$$

$$\texttt{iszero 0} \longrightarrow \texttt{true} \qquad \text{(E-IszeroZero)}$$

$$\texttt{iszero (succ } nv_1\texttt{)} \longrightarrow \texttt{false} \qquad \text{(E-IszeroSucc)}$$

$$\frac{t_1 \longrightarrow t_1'}{\texttt{iszero } t_1 \longrightarrow \texttt{iszero } t_1'} \qquad \text{(E-IsZero)}$$

continued on next page...

Typing

$$\texttt{true} : \texttt{Bool} \tag{T-True}$$

$$\texttt{false} : \texttt{Bool} \tag{T-False}$$

$$\frac{\texttt{t}_1 : \texttt{Bool} \quad \texttt{t}_2 : \texttt{T} \quad \texttt{t}_3 : \texttt{T}}{\texttt{if}\ \texttt{t}_1\ \texttt{then}\ \texttt{t}_2\ \texttt{else}\ \texttt{t}_3 : \texttt{T}} \tag{T-If}$$

$$\texttt{0} : \texttt{Nat} \tag{T-Zero}$$

$$\frac{\texttt{t}_1 : \texttt{Nat}}{\texttt{succ}\ \texttt{t}_1 : \texttt{Nat}} \tag{T-Succ}$$

$$\frac{\texttt{t}_1 : \texttt{Nat}}{\texttt{pred}\ \texttt{t}_1 : \texttt{Nat}} \tag{T-Pred}$$

$$\frac{\texttt{t}_1 : \texttt{Nat}}{\texttt{iszero}\ \texttt{t}_1 : \texttt{Bool}} \tag{T-IsZero}$$