COMPSCI 3MI3 : Final Exam

Fall 2021

Instructor: Nicholas Moore

Exam Start Time: 9:00 AM, Monday Dec. 13th, 2021

Exam End Time: 9:00 AM, Monday Dec. 14th, 2021

IMPORTANT INSTRUCTIONS, PLEASE READ

THIS EXAMINATION PAPER INCLUDES 5 QUESTIONS ON 4 PAGES. YOU ARE RESPONSIBLE FOR ENSURING YOUR COPY OF THE PAPER IS COMPLETE. BRING ANY DISCREPENCIES TO THE ATTENTION OF THE COURSE INSTRUCTOR.

- This is an open book exam in take-home format. **Permitted** aids include:
  - Course notes, lecture and tutorial recordings.
- **Prohibited** aids include:
  - Communication with anyone other than the instructor or TAs.
  - Tutoring websites such as Chegg and Course Hero
- If you have a question during the examination, or need clarification on a question, the instructor will be monitoring the class Piazza forum the day of the exam (`https://piazza.com/class/ksaqz5ph1zf3sm`).
  - Publically posted questions must not contain any amount of you solution to a problem.
  - Privately posted questions may contain portions of your solutions to problems.
  - Public or private posting of any portion of your solutions to any website other than Piazza is academic dishonesty.
  - Sharing your solutions publically on Piazza is also academic dishonesty.
- Submission will be made through the "Exam" Avenue dropbox.
- Students with SAS accomodations are awarded extra writing time according to their accomodation plan, with the extra time based on a 24 hour writing period.
- Failure to submit your exam by the end of your writing period will result in a grade of zero.
- Exam solutions created using the LaTeXdocument generation system are eligible for one (1) bonus point, so long as the source file is provided, and all code is included using the `lstlistings` environment.
- By writing this examination, you are agreeing to be bound by the Senate policy on Academic Integrity.
  - Absolutely no communication between students while writing the exam is permitted.

Good Luck!

1. (9 points) **Argument by Induction**

   Van Eck's sequence is defined as follows:

   - $E(0)$ and $E(1)$, the first two elements of the sequence, are both equal to zero.
   - To compute $E(i+1)$, take $E(i)$, and count how many numbers it has been since the last time $E(i)$ occurred in the sequence, relative to i. That is, where j is the index at which the most recent occurance of $E(i)$ occurs, the next number is $i - j$.

   The first 20 numbers in the Van Eck's sequence are:

   - 0, 0, 1, 0, 2, 0, 2, 2, 1, 6, 0, 5, 0, 2, 6, 5, 4, 0, 5, 3

   Argue, via induction, that, for any element in the sequence n, the sum of all elements in the sequence up to and including n is bounded by n^2.

2. **Breaking Theorems: Short Answer**

   Consider the small step semantics for the Boolean terms of UAE (Topic 4, slide 10), and answer the following questions. Please note, these questions do not require proofs, but you need to justify your answers.

   Here is a table of theorems we covered in Topic 4, with slide references.

Theorem	Slide Number
[Determinacy of One-Step Evaluation]	20
[If t is in Normal Form, t is a Value]	31
[Uniqueness of Normal Forms]	36

   (a) (5 points) Let's suppose we add a new, obviously incorrect evaluation rule to the small step semantics of UAE:

   $$\texttt{if true then } t_2 \texttt{ else } t_3 \rightarrow t_3 \qquad \text{(E–Funny1)}$$

   For each theorem above, indicate whether or not it is invalidated by E-Funny1. If a theorem is broken, briefly describe why, and provide an example.

   (b) (5 points) Now let's suppose we add the following rule:

   $$\frac{t_2 \rightarrow t_2'}{\texttt{if } t_1 \texttt{ then } t_2 \texttt{ else } t_3 \rightarrow \texttt{if } t_1 \texttt{ then } t_2' \texttt{ else } t_3} \qquad \text{(E-Funny2)}$$

   For each theorem above, indicate whether or not it is invalidated by E-Funny2. If a theorem is broken, briefly describe why, and provide an example.

3. **Trillian Semantics**

   In the far reaches of the galaxy, there is an advanced civillization refered to by human explorers as the "Trill". Humans developed Boolean values due to their bi-lateral symmetry. The Trill have a tri-laterally symmetric body plan, however, and so, their version of propositional logic is very different to ours. We refer to it as "Trillean Logic". Where Humans use `True` and `False`, Trillians use `Ploort`, `Xlenb` and `Zmifm`. The Trill also have two Trillean operators (`Lrykl` and `Groint`), with semantics given below.

   Note that the following language is not an extension of any language we studied in class, so addressing arithmetic expressions or lambda calculus in this question would be a mistake!

   $\langle t \rangle$ ::= Lrykl $\langle t \rangle$
   | Groint $\langle t \rangle$ $\langle t \rangle$
   | $\langle v \rangle$

$\langle v \rangle$ ::= Ploort
 | Xlenb
 | Zmifm

$\langle T \rangle$ ::= Trill

$$\frac{t_1 \to t_1'}{\texttt{Lrykl}\ t_1 \to \texttt{Lrykl}\ t_1'} \qquad \text{(E-Lrykl)}$$

$$\texttt{Lrykl Ploort} \to \texttt{Xlenb} \qquad \text{(E-LryklPloort)}$$

$$\texttt{Lrykl Xlenb} \to \texttt{Zmifm} \qquad \text{(E-LryklXlenb)}$$

$$\texttt{Lrykl Zmifm} \to \texttt{Ploort} \qquad \text{(E-LryklZmifm)}$$

$$\frac{t_1 \to t_1'}{\texttt{Groint}\ t_1\, t_2 \to \texttt{Groint}\ t_1'\, t_2} \qquad \text{(E-Groint1)}$$

$$\frac{t_2 \to t_2'}{\texttt{Groint}\ t_1\, t_2 \to \texttt{Groint}\ t_1\, t_2'} \qquad \text{(E-Groint2)}$$

$$\frac{t_1 = \texttt{Lrykl}\ v_2}{\texttt{Groint}\ t_1\, v_2 \to v_2} \qquad \text{(E-GrointLrykl)}$$

$$t \in \{\texttt{Ploort}, \texttt{Xlenb}, \texttt{Zmifm}\} : \mathit{Trill} \qquad \text{(T-Trill)}$$

$$\frac{t_1 : \mathit{Trill}}{\texttt{Lrykl}\ t_1 : \mathit{Trill}} \qquad \text{(T-Lrykl)}$$

$$\frac{t_1 : \mathit{Trill} \qquad t_2 : \mathit{Trill}}{\texttt{Groint}\ t_1\, t_2 : \mathit{Trill}} \qquad \text{(T-Groint)}$$

(a) (12 points) Construct a proof or disproof of determinacy for the above language.

(b) (12 points) Construct a proof or disproof of progress for the above language.

(c) (12 points) Construct a proof or disproof of preservation for the above language.

For all of the above, if you end up disproving one of the above theorems, you must indicate which parts of the language the theorem does hold for, and indicate how you would fix the language.

4. (10 points) **Deallocation Semantics**
While defining reference semantics in topic 10, we were very careful not to define a deallocation operation. Such operations are inherantly type-dangerous, for reasons stated in lecture. Let's ignore all that and define one anyways.

Propose a grammar, operational semantics, and typing rules for the deallocation operation `free`. Note that your equations need to carry both Γ and μ through in the appropriate places, but don't worry about Σ.

5. (9 points) **Implementing Exceptions** The following table gives the semantics for value-carrying exceptions and exception handling that we discussed in class.

$\rightarrow$ *exceptions*   *Extends* $\lambda_{\rightarrow}$ *(9-1)*

New syntactic forms

$\mathtt{t} ::=$	...	*terms:*
	$\mathtt{raise\ t}$	*raise exception*
	$\mathtt{try\ t\ with\ t}$	*handle exceptions*

New evaluation rules $\boxed{\mathtt{t} \longrightarrow \mathtt{t}'}$

$$(\mathtt{raise}\ \mathtt{v}_{11})\ \mathtt{t}_2 \longrightarrow \mathtt{raise}\ \mathtt{v}_{11} \qquad \text{(E-AppRaise1)}$$

$$\mathtt{v}_1\ (\mathtt{raise}\ \mathtt{v}_{21}) \longrightarrow \mathtt{raise}\ \mathtt{v}_{21} \qquad \text{(E-AppRaise2)}$$

$$\frac{\mathtt{t}_1 \longrightarrow \mathtt{t}_1'}{\mathtt{raise}\ \mathtt{t}_1 \longrightarrow \mathtt{raise}\ \mathtt{t}_1'} \qquad \text{(E-Raise)}$$

$$\mathtt{raise}\ (\mathtt{raise}\ \mathtt{v}_{11}) \longrightarrow \mathtt{raise}\ \mathtt{v}_{11} \qquad \text{(E-RaiseRaise)}$$

$$\mathtt{try}\ \mathtt{v}_1\ \mathtt{with}\ \mathtt{t}_2 \longrightarrow \mathtt{v}_1 \qquad \text{(E-TryV)}$$

$$\mathtt{try}\ \mathtt{raise}\ \mathtt{v}_{11}\ \mathtt{with}\ \mathtt{t}_2 \longrightarrow \mathtt{t}_2\ \mathtt{v}_{11} \qquad \text{(E-TryRaise)}$$

$$\frac{\mathtt{t}_1 \longrightarrow \mathtt{t}_1'}{\mathtt{try}\ \mathtt{t}_1\ \mathtt{with}\ \mathtt{t}_2 \longrightarrow \mathtt{try}\ \mathtt{t}_1'\ \mathtt{with}\ \mathtt{t}_2} \qquad \text{(E-Try)}$$

New typing rules $\boxed{\Gamma \vdash \mathtt{t} : \mathsf{T}}$

$$\frac{\Gamma \vdash \mathtt{t}_1 : \mathsf{T}_{exn}}{\Gamma \vdash \mathtt{raise}\ \mathtt{t}_1 : \mathsf{T}} \qquad \text{(T-Exn)}$$

$$\frac{\Gamma \vdash \mathtt{t}_1 : \mathsf{T} \qquad \Gamma \vdash \mathtt{t}_2 : \mathsf{T}_{exn} \rightarrow \mathsf{T}}{\Gamma \vdash \mathtt{try}\ \mathtt{t}_1\ \mathtt{with}\ \mathtt{t}_2 : \mathsf{T}} \qquad \text{(T-Try)}$$

Given the following Haskell code as a template, encode the evaluation semantics of error handling in Haskell. Note: You do not have to worry about Γ, Σ, μ, or any of the typing rules as they require type polymorphism.

```haskell
data T = App T T
       | Val V
       | Raise T
       | TryWith T T
  deriving (Show, Eq)

ssos :: T -> T
-- E-AppRaise1
-- E-AppRaise2
-- E-Raise
-- E-RaiseRaise
-- E-TryV
-- E-TryRaise
-- E-Try
```

– END OF EXAM –