COMPSCI 3MI3

Final Exam

1 Answers

1.1 Q1

We follow the definition of the Van Eck sequence as given in the exam paper, with $E(i)$ denoting the i-th number of the Van Eck sequence. We will prove that for any sequence up to the n-th element, the sum of all elements in the sequence up to an including n is bounded by n^2. Formally we prove:

$$P(n) = \sum_{i=0}^{n} E(i) \leq n^2$$

We will prove $P(n)$ holds for all $n \in \mathbb{N}$ by weak induction on n.

Base case: n = 0, 1, 2
It is trivial to show $P(n)$ holds for cases n = 0, 1, 2 by simple calculation.
$n = 0$: $0 \leq 0^2$ holds.
$n = 1$: $0 + 0 \leq 1^2 \equiv 0 \leq 1$ holds.
$n = 2$: $0 + 0 + 1 \leq 2^2 \equiv 1 \leq 4$ holds.

Thus, $P(0), P(1)$ and $P(2)$ hold and our base cases hold.

Induction step: $n > 2$. Assume $P(n)$ holds. We will show $P(n+1)$ holds.
By the formal definition of our property we must show:

$$\begin{aligned}
&\sum_{i=0}^{n+1} E(i) \leq (n+1)^2 && \langle \text{Definition of P(n+1)} \rangle \\
\iff &\sum_{i=0}^{n+1} E(i) \leq n^2 + 2n + 1 && \langle (a+b)^2 = a^2 + 2ab + b^2 \rangle \\
\iff &\sum_{i=0}^{n} E(i) + E(n+1) \leq n^2 + 2n + 1 && \langle \text{Split off top of } \Sigma \rangle
\end{aligned}$$

By our induction hypothesis $P(n)$ we know the sum of all elements up to n is bounded by n^2. Formally we know $\sum_{i=0}^{n} E(i) \leq n^2$.

$$\begin{aligned} & \sum_{i=0}^{n} E(i) + E(n+1) \leq n^2 + 2n + 1 \\ \iff & E(n+1) \leq 2n + 1 \qquad \langle \text{Ind. hyp. } P(n); \text{ Monotonicity of } \leq \rangle \end{aligned}$$

According to the definition of the Van Eck sequence, to compute $E(i+1)$ we count how many numbers it has been since $E(i)$ has occurred in the sequence relative to $i+1$. In a Van Eck sequence up to element $n+1$ we can count at most $n+1$ numbers before finding a previously occuring $E(i)$. Therefore we know $E(n+1) \leq n+1 \implies E(n+1) \leq 2n+1$. Thus our property holds for $P(n+1)$ and our inductive step holds.

Therefore, $P(n)$ holds for all $n \in \mathbb{N}$ by weak induction. We have shown that for any element in a Van Eck sequence of length n, the sum of all elements in the sequence up to an including the element is bounded by n^2. □

1.2 Q2

(a) If we add the evaluation rule E-Funny1 to our operational semantics:

- **Determinacy of One-Step Evalution** will be broken because we would introduce multiple evaluation rules that could apply to the same term at the same time. For example consider the term t with some subterms t_2, t_3:

  $$t = \textbf{if true then } t_2 \textbf{ else } t_3$$

  We can either apply E-Funny1 to t to obtain t_3 or we can apply E-IfTrue to obtain t_2. Therefore, we have $t \to t_3$ and $t \to t_2$ with $t_3 \neq t_2$, thus breaking the determinancy of our semantics.
- **If t is in Normal Form, t is a Value** will not be broken.
- **Uniqueness of Normal Forms** will be broken because we could evaluate a term t in multiple different ways, yiedling different normal forms. For example consider the term t below:

  $$t = \textbf{if true then true else false}$$

  Applying the E-Funny to t yields **false**, while applying E-IfTrue to t yields **true**. Thus we have $t \to^* \textbf{false}$ and $t \to^* \textbf{true}$ with both **false** and **true** being normal forms in our language, however **true** $\neq$ **false**. Therefore we have violated the theorem of uniqueness of normal forms.

(b) If we add the evaluation rule E-Funny2 to our operational semantics:

- **Determinacy of One-Step Evalution** will be broken because we would introduce multiple evaluation rules that could apply to the same terms at the same time. For example consider the term t with some subterms t_1, t_2 and t_3, with $t_1 \to t_1'$ and $t_2 \to t_2'$.

  $$t = \textbf{if } t_1 \textbf{ then } t_2 \textbf{ else } t_3$$

  We can either apply E-Funny2 to t with the premise $t_2 \to t_2'$ to obtain $t' = \textbf{if } t_1 \textbf{ then } t_2' \textbf{ else } t_3$, or we can apply E-If to t to obtain $t'' = \textbf{if } t_1' \textbf{ then } t_2 \textbf{ else } t_3$. Therefore, we have $t \to t'$ and $t \to t''$ but $t' \neq t''$, thus breaking the determinancy of our semantics.
- **If t is in Normal Form, t is a Value** will not be broken.
- **Uniqueness of Normal Forms** will not be broken.

1.3 Q3

(a) The property of determinacy does not hold for the given language and its semantics, since there are terms for which we can apply multiple evaluation rules at a time.

First, looking at E-Groint1 and E-Groint2 for a term $t = \textbf{Groint } t_1\ t_2$ with $t_1 \to t_1'$ and $t_2 \to t_2'$ we can apply either E-Groint1 to obtain $t' = \textbf{Groint } t_1'\ t_2$ or apply E-Groint2 to obtain $t'' = \textbf{Groint } t_1\ t_2'$. Since $t' \neq t''$ the semantics of the language are not determinate.

Secondly, if we examine E-Groint1 and E-GrointLrykl for a term $t = \textbf{Groint } (\textbf{Lrykl Zmifm})\ \textbf{Zmifm}$ we can apply either E-Groint1 to obtain $t' = \textbf{Groint Ploort Zmifm}$ or apply E-GrointLrykl to obtain $t'' = \textbf{Zmifm}$. Since $t' \neq t''$ the semantics of the language are not determinate.

One possible approach to fix the language and ensure determinancy is to replace E-Groint2 with a rule that has a value as a first term, and replace E-GrointLrykl with equivalent rules that works on two values, based on the semantics for **Lrykl**. These rules could work as follows:

$$\frac{t_2 \to t_2'}{\textbf{Groint } v_1\ t_2 \to \textbf{Groint } v_1\ t_2'} \qquad \text{(E-Groint2)}$$

$$\textbf{Groint Xlenb Ploort} \to \textbf{Ploort} \qquad \text{(E-GrointXP)}$$

Groint Zmifm Xlenb $\rightarrow$ **Xlenb** (E-GrointZX)

Groint Ploort Zmifm $\rightarrow$ **Zmifm** (E-GrointPZ)

(b) The property of progress does not hold for the given language and its semantics, since there exist well-typed terms in the language that are neither values nor does there exist an evaluation rule that can be applied to them.

We can consider any term $t =$ **Groint** $v_1\ v_2$ where v_1 and v_2 are values. We know we have $v_1 :$ *Trill* and $v_2 :$ *Trill* by T-Trill, which contains all values. Then by T-Groint, we can type **Groint** $v_1\ v_2 :$ *Trill*. While, t is well-typed, it is not a value nor can we apply an evaluation rule from the list given to t. Thus we have a well-typed term that is stuck not as a value, therefore the progress property does not hold.

One possible approach to fix the language and ensure the property of progress holds is to add evaluation rules of the form **Groint** $v_1\ v_2$ for *all* combinations of values v_1 and v_2. We could either complete the possible rules, adding onto the rules we defined in (a), or we could add the simple rule:

Groint $v_1\ v_2 \rightarrow v_1$ (E-GrointV)

Obviously, since humans do not understand the semantics of an alien language this additional rule may have an incorect meaning, but should ensure progress holds.

(c) We will prove that the property of preservation holds for the given language and its semantics. Formally we prove for any term t:

$$t : T \wedge t \rightarrow t' \implies t' : T$$

We will prove the above holds by induction on the typing derivation $t : T$. We proceed by cases.

Base case. Our base case denotes all typing derivations for which there do not exist any further sub-typing derivations. The possible typing derivations are of the following cases:

Case: T-Trill
For all cases where $t : T$ by T-Trill t is a value, therefore we cannot have $t \rightarrow t'$. Thus, the left side of our implication is false and our property is vacuously true.
Thus, our base case holds.

Induction step. We assume for all typing sub-derivations that the property of preservation holds. We prove for the remaining typing derivations the property of preservation holds. The typing derivation must be one of the following:

Case: T-Lrykl
If T-Lrykl was the last typing derivation then t : *Trill* and t has the form $\textbf{Lrykl}\, t_1$ with the premise t_1 : *Trill*.
If we can apply E-LryklPloort to t then $t = \textbf{Lrykl Ploort}$ and $t' = \textbf{Xlenb}$. By T-Trill we know t' : *Trill* and our property holds. We use the same logic to prove cases for E-LryklXlenb and E-LryklZmifm.
If we can apply E-Lrykl to t then $t = \textbf{Lrykl}\, t_1$, $t' = \textbf{Lrykl}\, t_1'$ and we have the premise $t_1 \to t_1'$. By our induction hypothesis on t_1 with the premises t_1 : *Trill* and $t_1 \to t_1'$ we know t_1' : *Trill*. Thus, by T-Lrykl, since we know t_1' : *Trill* we know $\textbf{Lrykl}\ t_1'$: *Trill*. Therefore, our property of preservation holds.

Case: T-Groint
If T-Groint was the last typing derivation then t : *Trill* and t has the form $\textbf{Groint}\, t_1\, t_2$ with the premises t_1 : *Trill* and t_2 : *Trill*.
If we can apply E-GrointLrykl to t then $t = \textbf{Groint}\, (\textbf{Lrykl}\, v_2)\, v_2$ and $t' = v_2$. Since t' is a value and by T-Trill we know all values are well-typed as *Trill*, we know t' : *Trill*. Thus, our property holds.
If we can apply E-Groint1 to t then $t = \textbf{Groint}\, t_1\, t_2$, $t' = \textbf{Groint}\, t_1'\, t_2$ and we have the premise $t_1 \to t_1'$. By our induction hypothesis on t_1 with the premises t_1 : *Trill* and $t_1 \to t_1'$ we know t_1' : *Trill*. Thus, by T-Groint with the premises t_1' : *Trill* and t_2 : *Trill* we know t' : *Trill*. Therefore, our property of preservation holds. We can make a symmetrical argument for proving E-Groint2.

Therefore, our induction step holds. We have shown for all possible typing derivations the property of preservation holds.
Thus, we've shown that preservation holds for the given language and its semantics.

1.4 Q4

We define the grammar, operational semantics and typing rules for a deallocation operation **free**, assuming all previously defined semantics from topic 10.

$\langle t \rangle$ ::= ...
  | **free** $\langle t \rangle$

$\langle v \rangle$ ::= ...

$\langle T \rangle$ ::= ...

$$\frac{t_1 \mid \mu \to t_1' \mid \mu'}{\textbf{free}\, t_1 \mid \mu \to \textbf{free}\, t_1' \mid \mu'} \quad \text{(E-Dealloc)}$$

$$\frac{l \in dom(\mu)}{\textbf{free}\, l \mid \mu \to unit \mid (\mu \setminus [l \mapsto v_1])} \quad \text{(E-DeallocL)}$$

Where $\mu \setminus [l \mapsto v_1]$ denotes the removal of the mapping of location l to value v_1 in the memory store, freeing up location l and removing v_1 from memory. Thus if $\mu' = \mu \setminus [l \mapsto v_1]$, then $dom(\mu') = dom(\mu) \setminus l$.

$$\frac{\Gamma \vdash t_1 : Ref\ T_1}{\Gamma \vdash \textbf{free}\, t_1 : Unit} \quad \text{(T-Dealloc)}$$

1.5 Q5

```
{-# LANGUAGE EmptyDataDecls, EmptyDataDeriving #-}
module Q5 where

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

data V deriving (Show, Eq)

-- Encodes the small-step semantics for error handling
ssos :: T -> T
ssos (App (Val _) (Raise v2@(Val _))) = Raise v2          -- E-AppRaise2
ssos (App (Raise v1@(Val _)) _) = Raise v1                -- E-AppRaise1
ssos (Raise (Raise v1@(Val _))) = Raise v1                -- E-RaiseRaise
ssos (Raise t) = Raise (ssos t)                           -- E-Raise
ssos (TryWith v1@(Val _) _) = v1                          -- E-TryV
ssos (TryWith (Raise v1@(Val _)) t2) = App t2 v1          -- E-TryRaise
ssos (TryWith t1 t2) = TryWith (ssos t1) t2               -- E-Try
ssos t = t              -- Reflexivity
```