

Theory exercises

Program verification with types and logic (NWI-IMC060)

Week 10 and 11

1. In this exercise, we consider the representation predicate for linked lists and nested linked lists. Linked lists are given by the following inductively-defined data type of our imperative object language:

$$l \in \text{llist } A ::= \text{lnil} \mid \text{lcons } hd \ tl \quad \text{where } hd \in A \text{ and } tl \in \text{ref } (\text{llist } A)$$

We define representation predicates $\text{is_list } l \ \vec{x}$ and $\text{is_nested_list } l \ \vec{\vec{x}}$ that relate linked lists $l \in \text{llist } A$ to Coq lists $\vec{x} \in \text{list } A$, respectively nested linked lists $l \in \text{llist } (\text{llist } A)$ to nested Coq lists $\vec{\vec{x}} \in \text{list } (\text{list } A)$:

```

is_list l  $\vec{x}$   $\triangleq$  match  $\vec{x}$  with
| []       $\Rightarrow l = \text{lnil}$ 
| hd  $\vec{x}_2 \Rightarrow \exists tl \ l_2. l = \text{lcons } hd \ tl \ * \ tl \mapsto l_2 \ * \ \text{is\_list } l_2 \ \vec{x}_2$ 
end

is_nested_list l  $\vec{\vec{x}}$   $\triangleq$  match  $\vec{\vec{x}}$  with
| []       $\Rightarrow l = \text{lnil}$ 
|  $\vec{x}_1 \ \vec{x}_2 \Rightarrow \exists hd \ tl \ l_2. l = \text{lcons } hd \ tl \ * \ \text{is\_list } hd \ \vec{x}_1 \ * \$ 
 $tl \mapsto l_2 \ * \ \text{is\_nested\_list } l_2 \ \vec{\vec{x}}_2$ 
end

```

Consider the following program that deallocates a nested linked list and returns the number of nested nodes that have been deallocated:

```

free_nested_list l  $\triangleq$  match l with
| lnil       $\Rightarrow 0$ 
| lcons hd tl  $\Rightarrow$  let  $n_1 = \text{free\_list } hd$  in
 $\text{let } n_2 = \text{free\_nested\_list } (\text{free } tl)$  in
 $n_1 + n_2$ 
end

```

Here, we use the function `free_list` whose specification is as follows:

$$[\text{is_list } l \ \vec{x}] \text{free_list } l \ [n. n = \text{length } \vec{x}]$$

- (a) Give a precise specification of `free_nested_list l` in terms of a Hoare triple.
- (b) Give a proof outline for your precise specification of `free_nested_list l`.

2. In this exercise, we consider the representation predicate for linked lists. Linked lists are given by the following inductively-defined data type in our imperative object language:

$$l \in \text{llist } A ::= \text{lnil} \mid \text{lcons } hd \ tl \quad \text{where } hd \in A \text{ and } tl \in \text{ref } (\text{llist } A)$$

We define the representation predicates $\text{is_list } v \ \vec{x}$ and $\text{is_ref_list } l \ \vec{x}$ that relate linked lists $v \in \text{llist } A$ and references to linked lists $l \in \text{ref } (\text{llist } A)$ of our imperative language to a mathematical list $\vec{x} \in \text{list } A$ as follows:

$$\begin{aligned} \text{is_list } v \ \vec{x} &\triangleq \text{match } \vec{x} \text{ with} \\ &\quad | [] \quad \Rightarrow v = \text{lnil} \\ &\quad | hd \ \vec{x}_2 \Rightarrow \exists tl. v = \text{lcons } hd \ tl \ * \ \text{is_ref_list } tl \ \vec{x}_2 \\ &\quad \text{end} \\ \text{is_ref_list } l \ \vec{x} &\triangleq \exists v. l \mapsto v \ * \ \text{is_list } v \ \vec{x} \end{aligned}$$

Consider the program of type $\text{ref } (\text{llist } A) * \text{ref } (\text{llist } A) \rightarrow ()$ that appends references to linked lists:

$$\begin{aligned} \text{app_list } x \ y &\triangleq \text{match } !x \text{ with} \\ &\quad | \text{lnil} \quad \Rightarrow x \leftarrow \text{free } y; () \\ &\quad | \text{lcons } hd \ tl \Rightarrow \text{app_list } tl \ y \\ &\quad \text{end} \end{aligned}$$

- (a) Give a specification of the append function $\text{app_list } x \ y$.
- (b) Give a proof outline for your specification of the append function.