

Пояснительная записка к решению задачи дедуктивной верификации

Определим постусловие функции с помощью lisp-функции `element-equality`, в которую достаточно передать длину списка `n` и список `a`. Результат этой функции совпадает с результатом Си-функции `element_equality`, поэтому постусловие зададим через оператор `=`:

```
/* (= (element-equality n a) result) */
```

lisp-функцию `element-equality` определим через вспомогательную функцию `element-equality-bool`, которая возвращает булево значение (`t` или `nil`):

```
(defun element-equality(n a)
  (if (element-equality-bool (- n 1) a) 1 0)
)
```

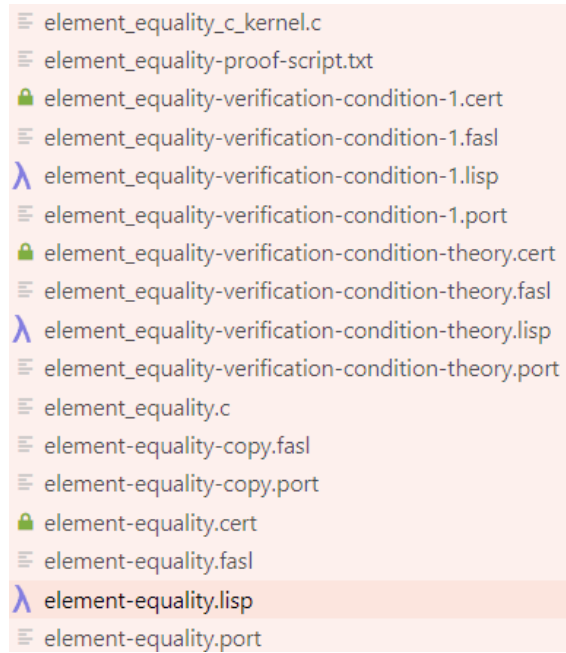
Это необходимо для упрощения рекурсивного задания функции. Функция `element-equality-bool` принимает индекс, до которого (включительно) необходимо проверить равенство элементов списка `a`. Таким образом, если передать `element-equality-bool (- n 1)`, то будет возвращён признак равенства всех элементов списка.

Функция `element-equality-bool` определена рекурсивно:

```
(defun element-equality-bool(j a)
  (if (not (natp j)) nil
      (if (= 0 j) t
          (and (= (nth (- j 1) a) (nth j a))
               (element-equality-bool (- j 1) a))
          )
      )
  )
)
```

С помощью оператора `if` проверяется, что `j` принадлежит множеству натуральных чисел. Если это не так, функция возвращает `nil`, что позволяет доказать завершимость функции. В основной ветви базой рекурсии является случай равенства `j` нулю. Условие `j = 0` соответствует проверке равенства единственного элемента самому себе, то есть всегда истинно. Поэтому в этом случае возвращается `t`. В случае `j /= 0` проверяется равенство текущего элемента (`(nth j a)`) и предыдущего элемента списка (`(nth (- j 1) a)`). Также рекурсивно проверяется равенство всех элементов до `(j - 1)` включительно с помощью вызова (`element-equality-bool (- j 1) a`). Если оба условия истинны, возвращается истина, то есть необходимо использовать оператор `and`.

В результате запуска верификации генерируются соответствующие .cert-файлы (файлы с пиктограммой зелёного замка):



A screenshot of a file explorer window showing a list of generated files. The files are organized into a list with icons on the left. The files are:

- element_equality_c_kernel.c
- element_equality-proof-script.txt
- element_equality-verification-condition-1.cert (green lock icon)
- element_equality-verification-condition-1.fasl
- element_equality-verification-condition-1.lisp (blue lambda icon)
- element_equality-verification-condition-1.port
- element_equality-verification-condition-theory.cert (green lock icon)
- element_equality-verification-condition-theory.fasl
- element_equality-verification-condition-theory.lisp (blue lambda icon)
- element_equality-verification-condition-theory.port
- element_equality.c
- element-equality-copy.fasl
- element-equality-copy.port
- element-equality.cert (green lock icon)
- element-equality.fasl
- element-equality.lisp (blue lambda icon)
- element-equality.port

Таким образом, верификация завершается успешно.

В случае, если задать условие $\text{and } (= (\text{nth } (- j 1) a) (\text{nth } j a))$ через равенство всех элементов первому элементу списка, то верификация не завершается успешно. Полагаю, для верификации такого условия необходимо задать инвариант цикла.