Technische Universität München
Institut für Informatik
Prof. Tobias Nipkow, Ph.D.
Dr. Peter Lammich

WS 2015/16
3. 11. 2015

Semantics of Programming Languages

Exercise Sheet 4

Exercise 4.1 Reflexive Transitive Closure

A binary relation is expressed by a predicate of type $R :: 's \Rightarrow 's \Rightarrow bool$. Intuitively, $R\ s\ t$ represents a single step from state s to state t.

The reflexive, transitive closure R^* of R is the relation that contains a step $R^*\ s\ t$, iff R can step from s to t in any number of steps (including zero steps).

Formalize the reflexive transitive closure as inductive predicate:

inductive *star* :: "$('a \Rightarrow 'a \Rightarrow bool) \Rightarrow 'a \Rightarrow 'a \Rightarrow bool$"

When doing so, you have the choice to append or prepend a step. In any case, the following two lemmas should hold for your definition:

lemma *star_prepend*: "$\llbracket r\ x\ y;\ star\ r\ y\ z \rrbracket \Longrightarrow star\ r\ x\ z$"
lemma *star_append*: "$\llbracket\ star\ r\ x\ y;\ r\ y\ z\ \rrbracket \Longrightarrow star\ r\ x\ z$"

Now, formalize the star predicate again, this time the other way round:

inductive *star'* :: "$('a \Rightarrow 'a \Rightarrow bool) \Rightarrow 'a \Rightarrow 'a \Rightarrow bool$"

Prove the equivalence of your two formalizations

lemma "$star\ r\ x\ y = star'\ r\ x\ y$"

Hint: The induction method expects the assumption about the inductive predicate to be first.

Exercise 4.2 Odd

The odd natural numbers can be specified by an inductive predicate:

inductive *odd* :: "$nat \Rightarrow bool$" **where**
  odd1: "*odd 1*"
| *oddSS*: "$odd\ n \Longrightarrow odd\ (n+2)$"

Prove, using Isar:

lemma
  assumes "$odd\ (n+2)$"
  shows "$odd\ n$"

Exercise 4.3 Binary Trees with the Same Shape

Consider this datatype of binary trees:

datatype *tree* = *Leaf int* | *Node tree tree*

Define an inductive binary predicate *sameshape* :: *tree* ⇒ *tree* ⇒ *bool*, where *sameshape* t_1 t_2 means that t_1 and t_2 have exactly the same overall size and shape. (The elements in the corresponding leaves may be different.)

inductive *sameshape* :: *"tree ⇒ tree ⇒ bool"* **where**

Now prove that the *sameshape* relation is transitive. Use the *inductive_cases* command to get customized elimination rules, and try to make an automatic proof. (Try to prove the lemma with induction and auto first, to figure out which cases you need.)

theorem *"⟦sameshape t_1 t_2; sameshape t_2 t_3⟧ ⟹ sameshape t_1 t_3"*

Homework 4.1 Counting Elements

Submission until Tuesday, November 10, 10:00am.

Give all your proofs in Isar, not apply style

Recall the count function, that counts how often a specified element occurs in a list:

fun *count* :: *"'a ⇒ 'a list ⇒ nat"* **where**
  "count x [] = 0"
| *"count x (y#ys) = (if x=y then Suc (count x ys) else count x ys)"*

Show that, if an element occurs in the list (its count is positive), the list can be split into a prefix not containing the element, the element itself, and a suffix containing the element one times less

lemma *"count x xs = Suc n ⟹ ∃ p s. xs = p@x#s ∧ count x p = 0 ∧ count x s = n"*

Homework 4.2 Counting Elements II

Submission until Tuesday, November 10, 10:00am.

Use Isar where appropriate

Use an inductive definition to specify the words accepted by the following context free grammar: $S \to aSbS \mid bSaS \mid \varepsilon$

datatype *character* = *a* | *b*

inductive *S* :: *"character list ⇒ bool"*

Show that every word accepted by the grammar has the same number of as and bs.

lemma *S_same_number*:
  assumes *"S xs"*
  shows *"count a xs = count b xs"*

The crucial lemma for the other direction, i.e., that every word with the same number of as and bs is accepted by the grammar, states that, in a sequence of numbers, such that the next number is one less or one more than the previous number, the first number is 0, and the last number is 1, we must find a 0 to 1 transition.

We first fix the situation described above in a context:

context
  fixes *d* :: *"nat ⇒ int"* — The sequence as a function
  fixes *n* :: *nat* — The maximum index into the sequence (length - 1)
  assumes *DIFF*: *"∀ i<n. abs (d i − d (i + 1)) = 1"* — The difference between adjacent elements is −1 or 1
  assumes *START*: *"d 0 = 0"* — The first element is 0
  assumes *END*: *"d n = 1"* — The last element is 1
begin

Your task is to prove the following theorem. You will need to generalize the theorem to allow arbitrary indexes with values ≤ 0 as starting point. Then, you may use the induction rule *inc_induct*.

Hint: If you have problems finding a proof, sketch the proof on paper first, and then try to translate your proof sketch to Isar!

  theorem *find_step*: *"∃ i<n. d i = 0 ∧ d (i+1) = 1"*

end

For **5 bonus points**, finish the proof of the other direction, i.e., prove:

theorem *"S l ⟷ count a l = count b l"*

Warning: This proof is hard. Do not attempt it unless you finished the rest of this homework. To succeed with this proof, you are strongly advised to sketch it on paper first, and then translate it to Isar.