

theory *Problem-1*

imports *Main*

begin

Jacek has n cards, labelled with consecutive numbers $1, \dots, n$. He places them in a row on the table, in any order he chooses. He will be taking the cards off the table in the ordering given by the cards' labels he will first take card number 1, then number 2, and so on. Before Jacek begins taking the cards off the table, Placek will color each card red, blue or yellow. Prove that Placek may color the cards in such a manner, that while Jacek is taking them off the table, at all times the following property is maintained: between any two cards of the same color, there is at least one card of a different color.

datatype *color* = *Red* | *Blue* | *Yellow*

definition *property* :: *color list* \Rightarrow *bool* **where**

property *clrs* $\longleftrightarrow (\forall a < \text{length } \text{clrs}. \forall b < \text{length } \text{clrs}.$

$a < b \wedge \text{clrs} ! a = \text{clrs} ! b \longrightarrow (\exists m \in \{a..b\}. \text{clrs} ! m \neq \text{clrs} ! a))$

lemma *successively-conv-nth*: *successively* *R* *xs* $\longleftrightarrow (\forall i < \text{length } \text{xs} - 1. R (\text{xs} ! i) (\text{xs} ! \text{Suc } i))$

by (*induction* *xs* *rule*: *induct-list012*; *force simp*: *nth-Cons split*: *nat.splits*)

lemma *propertyD*: *property* *xs* $\Longrightarrow i < j \Longrightarrow j < \text{length } \text{xs} \Longrightarrow \text{xs} ! i = \text{xs} ! j \Longrightarrow \exists m \in \{i..j\}.$
 $\text{xs} ! m \neq \text{xs} ! i$

unfolding *property-def* **by** *auto*

lemma *property-altdef*[*simp*]: *property* *xs* $\longleftrightarrow \text{distinct-adj } \text{xs}$

proof (*intro iffI*)

assume *p*: *property* *xs*

show *distinct-adj* *xs*

unfolding *distinct-adj-def* *successively-conv-nth*

proof (*intro allI impI*)

fix *i* **assume** *i*: $i < \text{length } \text{xs} - 1$

show $\text{xs} ! i \neq \text{xs} ! \text{Suc } i$

proof

assume *: $\text{xs} ! i = \text{xs} ! \text{Suc } i$

with *p i* **have** $\exists m \in \{i.. \text{Suc } i\}. \text{xs} ! m \neq \text{xs} ! i$

by (*intro propertyD*) *auto*

thus *False* **using** * **by** (*auto simp*: *le-Suc-eq*)

qed

qed

next

assume *distinct-adj* *xs*

show *property* *xs* **unfolding** *property-def*

proof *safe*

fix *i j* **assume** *ij*: $i < j < \text{length } \text{xs} \text{xs} ! i = \text{xs} ! j$

from (*distinct-adj* *xs*) **have** $\text{xs} ! i \neq \text{xs} ! \text{Suc } i$

using *ij* **by** (*auto simp*: *successively-conv-nth distinct-adj-def*)

thus $\exists m \in \{i..j\}. \text{xs} ! m \neq \text{xs} ! i$

using *ij* **by** (*intro bexI*[*of* - *Suc i*]) *auto*

qed

qed

lemma *property-insert*:

assumes *property* (*l* @ *r*)

obtains *clr* **where** *property* (*l* @ *clr* # *r*)

proof –

obtain *clr* **where** *clr*: $\text{clr} \neq \text{last } l \text{ clr} \neq \text{hd } r$

by (*metis* *color.distinct*(1,3,5))

let *?list* = *l* @ [*clr*] @ *r*

have *distinct-adj* *l* *distinct-adj* *r*

using *assms* **by** *auto*

hence *property* *?list*

unfolding *property-altdef distinct-adj-append-iff*
 using *clr* by *auto*
 thus *?thesis* using *that* by *simp*
 qed

definition *remove-smallest* :: *nat list* \Rightarrow *nat list* **where**
remove-smallest *xs* = *remove1* (*Min* (*set xs*)) *xs*

lemma *remove1-split*:
 assumes *a* \in *set xs*
 shows $\exists l r. xs = l @ a \# r \wedge remove1\ a\ xs = l @ r$
 using *assms* **proof** (*induction xs*)
 case (*Cons x xs*)
 show *?case*
proof *cases*
 assume *x* = *a*
 show *?thesis*
 apply (*rule exI[of - []]*)
 using $\langle x = a \rangle$ by *simp*
 next
 assume *x* \neq *a*
 then have *a* \in *set xs*
 using $\langle a \in set\ (x \# xs) \rangle$
 by *simp*
 then obtain *l r* **where** *: *xs* = *l* @ *a* # *r* $\wedge remove1\ a\ xs = l @ r$
 using *Cons.IH* by *auto*
 show *?thesis*
 apply (*rule exI[of - x # l]*)
 apply (*rule exI[of - r]*)
 using $\langle x \neq a \rangle$ * by *auto*
 qed
 qed *simp*

lemma *remove-smallest-distinct*:
distinct xs \implies *distinct* (*remove-smallest xs*)
 unfolding *remove-smallest-def* by *simp*

lemma *remove-smallest-subset*:
set (*remove-smallest xs*) \subseteq *set xs*
 unfolding *remove-smallest-def* by (*rule set-remove1-subset*)

lemma *remove-smallest-length[simp]*: *xs* $\neq [] \implies length\ (remove-smallest\ xs) < length\ xs$
 by (*simp add: remove-smallest-def length-remove1*)

lemma *remove-smallest-map*:
 assumes *map f xs* = *map g xs*
 shows *map f* (*remove-smallest xs*) = *map g* (*remove-smallest xs*)
proof *cases*
 assume *xs* = []
 then show *?thesis* by (*simp add: remove-smallest-def*)
 next
 assume *xs* $\neq []$
 then have *Min* (*set xs*) \in *set xs*
 by *auto*
 then obtain *l r* **where** *: *xs* = *l* @ *Min* (*set xs*) # *r* **and** **: *remove-smallest xs* = *l* @ *r*
 unfolding *remove-smallest-def*
 using *remove1-split* by *fast*
 from *assms* have $\forall x \in set\ xs. f\ x = g\ x$ by *auto*
 hence $(\forall x \in set\ l. f\ x = g\ x) \wedge (\forall x \in set\ r. f\ x = g\ x)$
 apply (*subst* (*asm*) *)
 by *auto*
 then show *?thesis*
 unfolding **

```

    by simp
qed

lemma removal-induction:
  assumes  $P \ []$ 
  assumes  $\bigwedge xs. xs \neq [] \implies P \ (\text{remove-smallest } xs) \implies P \ xs$ 
  shows  $P \ ys$ 
  using assms
  apply induction-schema
  apply auto[1]
  by lexicographic-order

theorem problem1:
  fixes order :: nat list
  assumes distinct order
  shows  $\exists \text{ colors} :: \text{nat} \Rightarrow \text{color}. \forall n. \text{property} \ (\text{map colors} \ ((\text{remove-smallest} \sim n) \ \text{order}))$ 
    ( $\text{is } \exists \text{ colors}. \forall n. ?P \ \text{colors } n \ \text{order}$ )
  using assms proof (induction rule: removal-induction)
  case 1
  have remove-smallest [] = []
    by (simp add: remove-smallest-def)
  hence  $(\text{remove-smallest} \sim n) \ [] = []$  for  $n$ 
    by (induction n) auto
  then show ?case by simp
next
  case (2 order)
  then have distinct: distinct (remove-smallest order)
    using remove-smallest-distinct by auto
  note 2.IH[OF distinct]
  then obtain colors where IH':  $?P \ \text{colors } n \ (\text{remove-smallest order})$  for  $n$ 
    by auto
  let ?m = Min (set order)
  obtain xs ys where xmys:  $\text{order} = xs @ ?m \# ys$  and xsys:  $\text{remove-smallest order} = xs @$ 
     $ys$ 
    unfolding remove-smallest-def
    using  $\langle \text{order} \neq [] \rangle$  by atomize-elim (auto intro: remove1-split)
  let ?l = map colors xs and ?r = map colors ys
  have property  $(?l @ ?r)$ 
    using xsys IH'[of 0] by simp
  then obtain clr where property-clr:  $\text{property} \ (?l @ \text{clr} \# ?r)$ 
    by (auto intro: property-insert)
  let ?colors' = colors(?m := clr)
  have  $?m \notin \text{set } xs \ \ ?m \notin \text{set } ys$ 
    by (metis  $\langle \text{distinct order} \rangle$  xmys distinct-append not-distinct-conv-prefix)
    (metis  $\langle \text{distinct order} \rangle$  xmys distinct.simps(2) distinct-append)
  hence property0:  $\text{property} \ (\text{map } ?\text{colors}' \ \text{order})$ 
    using property-clr by (subst (2) xmys) simp
  have  $?m \notin \text{set} \ (\text{remove-smallest order})$ 
    unfolding remove-smallest-def
    using  $\langle \text{distinct order} \rangle$  by simp
  hence  $?m \notin \text{set} \ ((\text{remove-smallest} \sim n) \ (\text{remove-smallest order}))$  ( $\text{is } ?m \notin \text{set} \ (xs \ n)$ )
    for  $n$ 
    by (induction n) (use remove-smallest-subset in auto)
  hence map-colors:  $\text{map } ?\text{colors}' \ (xs \ n) = \text{map colors} \ (xs \ n)$  for  $n$ 
    by simp
  show ?case
    apply (intro exI[ $\text{of } - ?\text{colors}'$ ] allI)
    subgoal for  $n$ 
      apply (cases n)
      apply (simp only: funpow-0, rule property0)
      by (auto simp only: funpow-Suc-right o-apply map-fun-upd map-colors IH')
    done
  qed

```

end