

Stofzuiger

By freekver

May 6, 2014

Contents

theory *Stofzuiger*
imports *Main Option*
begin

locale *equivalence-relation* =
 fixes *eq* :: 'a \Rightarrow 'a \Rightarrow bool (**infixr** \sim 60)
 and *dom* :: 'a set
 assumes *reflexive*: $\bigwedge a . a \sim a$
 and *symmetric*: $\bigwedge a\ b . a \sim b \longrightarrow b \sim a$
 and *transitive*: $\bigwedge a\ b\ c . a \sim b \wedge b \sim c \longrightarrow a \sim c$
begin

Let be $op \sim$ be an equivalence relation over domain dom .

definition *congruent* :: ('a \Rightarrow 'a) \Rightarrow bool
where *congruent* $f \equiv \forall a\ b . a \sim b \longrightarrow (f\ a) \sim (f\ b)$

definition *equivalence-class-of* :: 'a \Rightarrow 'a set ($\langle - \rangle$ 100)
where *equivalence-class-of* $a \equiv \{ a' \in dom . a \sim a' \}$

definition *quotient-set* :: 'a set \Rightarrow set ($A \backslash \sim$)
where *quotient-set* $\equiv equivalence-class-of\ dom$

Een strategy is *covering* with respect to relation $op \sim$ iff all equivalence classes c contain some element s in the strategy.

definition *covering* :: 'a list \Rightarrow bool
where *covering* $\sigma \equiv \forall c \in A \backslash \sim . \exists s \in c . s \in set\ \sigma$

theorem *covering-preserved*:
assumes *covering* σ
 and *congruent* f
 and *bij-betw* $f\ dom\ dom$
 shows *covering* ($map\ f\ \sigma$)
proof –

```

{
  have ecs-in-quotient-set:  $\forall s \in \text{dom}. \langle s \rangle \in A \setminus \sim$  by (metis (full-types) image-eqI
quotient-set-def)
  have range-is-dom:  $f \text{ ' dom} = \text{dom}$  by (metis assms(3) bij-betw-def)

  fix c
  assume c:  $c \in A \setminus \sim$ 
  from this assms(1) obtain s' where  $1: s' \in c \wedge s' \in \text{dom}$ 
    unfolding covering-def quotient-set-def equivalence-class-of-def image-def by
    auto

  hence  $\exists s \in \text{dom}. s \in \langle \text{inv-into dom } f s' \rangle$ 
    by (metis (lifting) assms(3) bij-betw-def equivalence-class-of-def inv-into-into
mem-Collect-eq reflexive)

  from this assms(1)
    obtain s where  $2: s \in \langle \text{inv-into dom } f s' \rangle \wedge s \in \text{set } \sigma$ 
    unfolding covering-def image-def
    using ecs-in-quotient-set[unfolded Ball-def, THEN spec, of inv-into dom f s']
      inv-into-into[of s' f dom] assms(3) range-is-dom 1
    by auto
  hence  $s \sim \text{inv-into dom } f s'$ 
    by (metis (lifting) equivalence-class-of-def mem-Collect-eq symmetric)
  hence  $f s \sim f (\text{inv-into dom } f s')$ 
    by (metis (lifting) assms(2) congruent-def)
  hence  $f s \sim s'$ 
    by (metis 1 range-is-dom f-inv-into-f)
  moreover
    have  $f s \in \text{dom}$ 
    by (metis (lifting) 2 range-is-dom equivalence-class-of-def image-eqI mem-Collect-eq)
  ultimately
    have  $f s \in c$ 
    using 1 c[unfolded quotient-set-def]
      symmetric[of f s s'] transitive[where  $b=s'$  and  $c=f s$ ]
    by (auto simp add: equivalence-class-of-def)
  moreover
    from 2 have  $f s \in f'(\text{set } \sigma)$ 
    by (metis imageI)
  ultimately
    have  $\exists s' \in c. s' \in f'(\text{set } \sigma)$  by auto
}
thus ?thesis unfolding covering-def by auto
qed
end

```

Kies een constant n. We beschouwen een robot die rond rijdt in een (n+1) x (n+1) grid.

```

axiomatization n :: nat
where non-empty:  $n > 0$ 

```

Het gedrag van de robot wordt beschreven door het volgende gelabelde transitie-systeem met acties *turn* en *forward*:

datatype $A = \textit{turn} \mid \textit{forward}$

En als state triples (x,y,d) :

datatype $\textit{dir} = N \mid E \mid S \mid W$

type-synonym $Q = \textit{nat} \times \textit{nat} \times \textit{dir}$

De transities worden gegeven door functie “*transition_relation*”. Dit is een partiele functie. Het levert geen resultaat op wanneer x of y out of bounds is. Er is bijv. geen overgang van $(0,2,N)$ met als actie ‘*forward*’.

definition $\textit{rotate} :: \textit{dir} \Rightarrow \textit{dir}$

where $\textit{rotate} \ d \equiv \textit{case} \ d \ \textit{of} \ N \Rightarrow E \mid E \Rightarrow S \mid S \Rightarrow W \mid W \Rightarrow N$

fun $\textit{transition_relation} :: Q \Rightarrow A \Rightarrow Q \ \textit{option}$

where $\textit{transition_relation} \ (x,y,d) \ \textit{turn} = \textit{Some} \ (x,y,\textit{rotate} \ d)$

$\mid \textit{transition_relation} \ (x,y,N) \ \textit{forward} = (\textit{if} \ y < n - 1 \ \textit{then} \ \textit{Some} \ (x, \ y + 1, N) \ \textit{else} \ \textit{None})$

$\mid \textit{transition_relation} \ (x,y,E) \ \textit{forward} = (\textit{if} \ x < n - 1 \ \textit{then} \ \textit{Some} \ (x + 1, \ y, E) \ \textit{else} \ \textit{None})$

$\mid \textit{transition_relation} \ (x,y,S) \ \textit{forward} = (\textit{if} \ y > 0 \ \textit{then} \ \textit{Some} \ (x, \ y - 1, S) \ \textit{else} \ \textit{None})$

$\mid \textit{transition_relation} \ (x,y,W) \ \textit{forward} = (\textit{if} \ x > 0 \ \textit{then} \ \textit{Some} \ (x - 1, \ y, W) \ \textit{else} \ \textit{None})$

Functie $R : S \rightarrow S$ is een functie die het hele grid 90 graden met de klok mee draait.

definition $R :: Q \Rightarrow Q$

where $R \ s \equiv \textit{case} \ s \ \textit{of} \ (x,y,d) \Rightarrow (y,(n - 1) - x,\textit{rotate} \ d)$

Functie $F : S \rightarrow S$ is een functie die het grid spiegelt in de as $x = 1/2 \ n$:

definition $\textit{flip} :: \textit{dir} \Rightarrow \textit{dir}$

where $\textit{flip} \ d \equiv \textit{case} \ d \ \textit{of} \ N \Rightarrow S \mid E \Rightarrow E \mid S \Rightarrow N \mid W \Rightarrow W$

definition $F :: Q \Rightarrow Q$

where $F \ s \equiv \textit{case} \ s \ \textit{of} \ (x,y,d) \Rightarrow ((n - 1) - x, \ y, \textit{flip} \ d)$

definition $\textit{option_equal} :: 'a \ \textit{option} \Rightarrow 'a \Rightarrow \textit{bool} \ (\textbf{infixr} \ \simeq \ 50)$

where $\textit{option_equal} \ \textit{ao} \ a \equiv \textit{case} \ \textit{ao} \ \textit{of} \ \textit{None} \Rightarrow \textit{False} \mid \textit{Some} \ a' \Rightarrow a' = a$

DEFINITIE Een *automorfisme* voor transitie-systeem L is een bijectie $f : S \rightarrow S$ zdd voor alle s, s' in S en voor alle a in A , $s \xrightarrow{a} s'$ iff $f(s) \xrightarrow{a} f(s')$.

definition $\textit{automorfism} :: ('s \Rightarrow \textit{bool}) \Rightarrow ('s \Rightarrow 'a \Rightarrow 's \ \textit{option}) \Rightarrow ('s \Rightarrow 's) \Rightarrow \textit{bool}$

where $\textit{automorfism} \ \textit{statep} \ L \ f \equiv \forall \ s \ s' \ a. \ \textit{statep} \ s \wedge \textit{statep} \ s' \longrightarrow L \ s \ a \simeq s' \longleftarrow L \ (f \ s) \ a \simeq (f \ s')$

DEFINITIE Een *autocontramorfisme* voor transitie-systeem L is een bijectie $f : S \rightarrow S$ zdd voor alle s, s' in S en voor alle a in A , $s \xrightarrow{a} s'$ iff $f(s') \xrightarrow{a} f(s)$.

$f(s)$. We gaan uit van geldige states. Welke states geldig zijn, wordt bepaald door functie *statep*, die als parameter meegegeven wordt.

definition *autocontramorfisme* :: $('s \Rightarrow \text{bool}) \Rightarrow ('s \Rightarrow 'a \Rightarrow 's \text{ option}) \Rightarrow ('s \Rightarrow 's) \Rightarrow \text{bool}$
where *autocontramorfisme statep* $L f \equiv \forall s s' a. \text{statep } s \wedge \text{statep } s' \longrightarrow L s a \simeq s' \longleftrightarrow L (f s') a \simeq (f s)$

CLAIM R is een automorfisme voor L.

abbreviation *statep* :: $Q \Rightarrow \text{bool}$
where *statep* $q \equiv \text{case } q \text{ of } (x,y,d) \Rightarrow x < n \wedge y < n$
lemma *R-is-automorfism*:
shows *automorfism statep transition-relation R*
proof–
{
 fix a
 fix $s s' :: Q$
 fix $x y d x' y' d'$
 assume $s = (x,y,d)$
 moreover assume $s' = (x',y',d')$
 moreover assume $x < n$
 moreover assume $x' < n$
 ultimately
 have $(\text{transition-relation } s a \simeq s') = (\text{transition-relation } (R s) a \simeq (R s'))$
 by (*cases (s,a) rule: transition-relation.cases,*
 auto simp add: option-equal-def R-def rotate-def split: dir.splits)
}
thus *?thesis*
 unfolding *automorfism-def*
 by (*auto split add: option.splits*)
qed

CLAIM F is een autocontramorfisme voor L.

lemma *F-is-autocontramorfisme*:
shows *autocontramorfisme statep transition-relation F*
proof–
{
 fix a
 fix $s s' :: Q$
 fix $x y d x' y' d'$
 assume $s = (x,y,d)$
 moreover assume $s' = (x',y',d')$
 moreover assume $x < n$
 moreover assume $x' < n$
 moreover assume $y < n$
 moreover assume $y' < n$
 ultimately
 have $(\text{transition-relation } s a \simeq s') = (\text{transition-relation } (F s') a \simeq F s)$
 by (*cases (s,a) rule: transition-relation.cases,*

```

      auto simp add: option-equal-def F-def flip-def rotate-def min-def split:
dir.splits)
}
thus ?thesis
  unfolding autocontramorfisme-def
  by (auto)
qed

```

Een strategy is een lijst van states. Een strategy is *covering* iff alle coördinaten (x,y) in een bepaalde richting voorkomen in de strategy.

definition $Q\text{-eq} :: Q \Rightarrow Q \Rightarrow \text{bool}$ (**infixr** \sim 50)
where $Q\text{-eq } a \ b \equiv \text{fst } a = \text{fst } b \wedge \text{fst } (\text{snd } a) = \text{fst } (\text{snd } b)$

interpretation $Q\text{-eq-int}$: equivalence-relation $op \sim \{(x,y,d) . x < n \wedge y < n\}$

```

proof (unfold-locales)
  fix a b c :: Q
  show  $a \sim a$  unfolding  $Q\text{-eq-def}$  by auto
  show  $a \sim b \longrightarrow b \sim a$  unfolding  $Q\text{-eq-def}$  by auto
  show  $a \sim b \wedge b \sim c \longrightarrow a \sim c$  unfolding  $Q\text{-eq-def}$  by metis
qed

```

lemma $R\text{-preserves-range}$:

assumes $x < n$

and $y < n$

shows $(x, y, b) \in R \text{ ' } (\{x. x < n\} \times \{(y, d). y < n\})$

proof–

have $\exists m < n. x = n - \text{Suc } m$

by (metis Nat.add-diff-inverse add-diff-cancel-right' assms(1) less-SucI monoid-add-class.add.right-neutral nat.exhaust nat-diff-split not-add-less1 not-less-eq)

moreover

have $\exists m < n. y = n - \text{Suc } m$

by (metis Nat.add-diff-inverse add-diff-cancel-right' assms(2) less-SucI monoid-add-class.add.right-neutral nat.exhaust nat-diff-split not-add-less1 not-less-eq)

ultimately

show ?thesis

using assms

unfolding $R\text{-def}$ image-def rotate-def

by (auto split add: dir.splits)

qed

lemma $F\text{-preserves-range}$:

assumes $x < n$

and $y < n$

shows $(x, y, b) \in F \text{ ' } (\{x. x < n\} \times \{(y, d). y < n\})$

proof–

have $\exists m < n. x = n - \text{Suc } m$

by (metis Nat.add-diff-inverse add-diff-cancel-right' assms(1) less-SucI monoid-add-class.add.right-neutral nat.exhaust nat-diff-split not-add-less1 not-less-eq)

```

thus ?thesis
  using assms
  unfolding F-def image-def flip-def
  by (auto split add: dir.splits)
qed

```

```

lemma map-R-preserves-covering :
assumes Q-eq-int.covering  $\sigma$ 
shows Q-eq-int.covering (map R  $\sigma$ )
proof–
  have Q-eq-int.congruent R
    unfolding Q-eq-int.congruent-def R-def rotate-def Q-eq-def
    by auto
  moreover
  have bij-betw R  $\{(x,y,d) . x < n \wedge y < n\} \{(x,y,d) . x < n \wedge y < n\}$ 
    unfolding bij-betw-def surj-def inj-on-def R-def rotate-def
    using R-preserves-range
    by (auto split add: dir.splits simp add: bij-betw-def surj-def inj-on-def R-def
rotate-def)
  ultimately
  show ?thesis
    using assms Q-eq-int.covering-preserved
    by auto
qed

```

```

lemma map-F-preserves-covering :
assumes Q-eq-int.covering  $\sigma$ 
shows Q-eq-int.covering (map F  $\sigma$ )
proof–
  have Q-eq-int.congruent F
    unfolding Q-eq-int.congruent-def F-def rotate-def Q-eq-def
    by auto
  moreover
  have bij-betw F  $\{(x,y,d) . x < n \wedge y < n\} \{(x,y,d) . x < n \wedge y < n\}$ 
    unfolding bij-betw-def surj-def inj-on-def R-def rotate-def
    using F-preserves-range
    by (auto split add: dir.splits simp add: bij-betw-def surj-def inj-on-def F-def
flip-def)
  ultimately
  show ?thesis
    using assms Q-eq-int.covering-preserved
    by auto
qed

```

end