Übung 7

Alexander Mattick Kennung: qi69dube

Kapitel 1

16. Juli 2020

```
prepend::List Sprite->Animation->Animation
sprite (prepend Nil anim) =sprite anim
sprite (prepend s anim) = s

advance (prepend Nil anim) = anim
advance (prepend s:xs anim) = prepend xs anim

transition::Animation->Animation->Animation
sprite (transition a1 a2) = sprite a1
advance (transition a1 a2) = if compatible (sprite a1) (sprite a2)
 then advance(a2)
 else advance a1
```

3.

a) wir müssen nur den fall untersuchen, dass die prämisse gilt (sonst Ex falso sequitur quodlibet)

Im folgenden gehen wir also davon aus, dass die prämisse gilt.

Auswertung beider Seiten für sprite/advance:

1. sprite (transition (loop [s]) (loop (Cons t ts))) $\overset{def\ transition}{=}$ sprite(loop [s]) $\overset{def\ sprite}{=}$ s = sprite (loop [s])

2. advance (transition (loop [s]) (loop (Cons t ts))) $\overset{def\ transition}{=}$ advance(advance(loop [s])) $\overset{def\ advance}{=}$advance(loop(snoc Nil s))$\overset{def_snoc}{=}$ advance(loop([s]))

Was zu beweisen war. Der Satz folgt also direkt aus der Definition der funktionen. (er ist auf syntaktischer ebene korrekt)

b) Auch hier: nur der Fall mit wahren prämissen ist relevant. (Ex falso)

1. sprite (transition (loop Cons s ss) (loop (Cons t ts))) $\overset{def\ transition}{=}$ sprite (loop (Cons s ss)) $\overset{def\ sprite}{=}$ s

sprite (prepend [s] (loop (snoc ts t))) $\overset{def\ prepend}{=}$ s.

gilt

2.

advance (transition (loop Cons s ss) (loop (Cons t ts))) $\overset{def\ transition}{=}$ advance(loop (Cons t ts)) $\overset{def\_loop}{=}$ loop (snoc (ts t).

weiterhin gilt

advance (prepend [s] (loop (snoc ts t))) $\overset{def\ prepend}{=}$ advance (prepend Nil (loop (snoc ts t))) $\overset{def\ prepend}{=}$ loop (snoc ts t)

Also gilt dies auch.

(wir brauchen also wieder keine Bisimulation, mann kann aber natürlich eine Triviale einführen, wenn man will)

1

```
data ITree a where
        inner: ITree a->a
        left: ITree a-> ITree a
        right:ITree a-> ITree a
```

1.

$$G = A \times id \times id$$

2.

```
itadd::ITree Nat->ITree Nat->ITree Nat
inner (itadd a b) = (inner a)+(inner b)
left (itadd a b) = itadd (left a) (left b)
right (itadd a b) = itadd (right a) (right b)
```

3.

```
flip::Stream Bool->Stream Bool
hd (flip b) = not (hd b)
tl (flip b) = flip (tl b)
choose::Stream Bool->ITree a->Stream a
hd(choose a b)= inner b
tl(choose a b)=if hd a then choose (tl a) (left b) else choose (tl a) (right b)
mirror::ITree a->ITree a
inner (mirror a) = inner a
left (mirror a) = mirror(right a)
```

```
right (mirror a) = mirror(left a)
```

4.

Die Bisimulation muss hier für alle "Richtungen" gelten (bzw Relation muss für alle G-Terme gelten) für sRt gilt

$$inner\ s = inner\ t$$

$$(left\ s) R (left\ t)$$

$$(right\ s) R (right\ t)$$

5.

a) hier natürlich Koinduktion über unendliche Bäume:

Sei $R = \{(itadd\ t1\ t2, itadd\ t2\ t1) | \forall t1, t2 \in ITree\}$

inner (itadd t1 t2) $\overset{def\ itadd}{=}$ (inner t1) + (inner t2) $\overset{def\ itadd}{=}$ inner (itadd t2 t1)

left (itadd t1 t2) $\overset{def\ itadd}{=}$ (itadd (left t1) (left t2))$\underbrace{R(itadd(left\ t2)(left\ t1))}_{IV}$

right (itadd t1 t2) $\overset{def\ itadd}{=}$ (itadd (right t1) (right t2))$\underbrace{R(itadd(right\ t2)(right\ t1))}_{IV}$

hierbei wird ausgenutzt, dass right/left eine instanz von Tree liefert.

b) Koinduktion über Streams (da das äußerste ein Stream ist)

Sei $R = \{(choose(flip\ s)(mirror\ t), choose\ s\ t), \forall s \in Stream\}$

hd (choose (flip s) (mirror t)) $\overset{def\ choose}{=}$ inner (mirror t) $\overset{def\ mirror}{=}$ inner t = inner(choose s t).

tl (choose (flip s) (mirror t)) $\overset{def\ choose}{=}$ (if hd (flip s) then (choose (tl(flip s)) (left(mirror t))) else (choose (tl(flip s)) (right(mirror t)))).

$\overset{def\ flip}{=}$ (if not (hd s) then (choose (flip(tl s)) (left(mirror t))) else (choose (flip(tl s)) (right(mirror t))))

$\overset{def\ mirror}{=}$ (if not (hd s) then (choose (flip(tl s)) (mirror (right t))) else (choose (flip(tl s)) (mirror (left t))))

Andere Seite der Bisimulation:

tl (choose s t) $\overset{def\ choose}{=}$ if (hd s) then (choose (tl s) (left t)) else (choose (tl s) (right b))

2 Fälle (hd s)=True und (hd s) = False (induktion über Bool).

1. $(hd\ s) = True \implies not(hd\ s) = False$

(choose (flip(tl s)) (mirror (left t)))) R (choose (tl s) (left t))

2. $(hd\ s) = False \implies not(hd\ s) = True$

(choose (flip(tl s)) (mirror (right t)))R(choose (tl s) (right b))

somit gilt auch diese Aussage.