Uniform Candy Distribution

Reynald Affeldt

National Institute of Advanced Industrial Science and Technology (AIST)

First time online: November 17, 2011

ucd

SSReflect formalization of the Uniform Candy Distribution [1] puzzle following [2]. Using the SSReflect extension [4] of the Coq proof-assistant [3].

1 Problem Description and Basic Properties

The child on the right and related properties:

```
Definition right n i := if i < n.-1 then i.+1 else O.
```

```
Lemma right_O n : right n n.-1 = O.
```

```
Lemma lt_right n i : i < n → right n i < n.
```

```
Lemma sum_rotate_right : ∀ l,
  \zsum_(0 ≤ i < size l) l`_(right (size l) i) ^^ 2 = \zsum_(0 ≤ i < size l) l`_i ^^ 2.
```

The "Increment if odd" operation:

```
Definition incr_odd x := if Zodd_bool x then x + 1 else x.
```

"Increment if odd" always increases the number of candies...

```
Lemma incr_odd_inc n l P : n = size l →
  \big[Zplus/0]_(0 ≤ i < n | P i) l`_i ≤ \big[Zplus/0]_(0 ≤ i < n | P i) (map incr_odd l)`_i.
```

...and increasing is strict when not all candies' numbers are even:

```
Lemma incr_odd_strict_inc l : ~~ all Zeven_bool l →
  \zsum_(0 ≤ i < size l) l`_i < \zsum_(0 ≤ i < size l) (map incr_odd l)`_i.
```

Compute the next number of candy for the ithe child:

```
Definition m_next i l := l`_i / 2 + l`_(right (size l) i) / 2.
```

Distribute the candies among the children:

```
Definition distribute l := map (m_next^~ l) (iota 0 (size l)).
```

Distribution preserves the total number of candies:

```
Lemma distribute_preserves_total : ∀ l, all Zeven_bool l →
```

$\backslash zsum\_(0 \le i < size\ l)\ (distribute\ l)`\_i = \backslash zsum\_(0 \le i < size\ l)\ l`\_i.$

The candies are uniformly distributed:

```
Definition uniform l := all (fun x ⇒ x == l`_O) l.
```

The main algorithm:

```
Fixpoint ucd k l := if k is k'.+1 then
                      if uniform l then l else ucd k' (map incr_odd (distribute l))
                    else l.
```

After each round, the list of numbers of candies is even:

```
Lemma ucd_even : ∀ k l, all Zeven_bool l → all Zeven_bool (ucd k l).
```

```
Lemma ucd_inc : ∀ d k l, all Zeven_bool l →
  \zsum_(0 ≤ i < size l) (ucd k l)`_i ≤ \zsum_(0 ≤ i < size l) (ucd (k + d) l)`_i.
```

2 The Main Properties

The Total Number of Candies is Bounded

After one round, the number of candies of one child is no more than what it or its neighbor had before:

```
Lemma one_round_one_child_bound l : all Zeven_bool l → ∀ i, (i < size l)%nat →
  (map incr_odd (distribute l))`_i ≤ Zmax l`_i l`_(right (size l) i).
```

The total sum of candies is never more than the n times the initial maximum for one child, where n is the total number of children:

```
Lemma ucd_bound k l : all Zeven_bool l →
  \zsum_(0 ≤ i < size l) (ucd k l)`_i ≤ Z_of_nat (size l) × Z_max l.
```

The Total Number of Candies Becomes Constant

After a while, the total number of candies cannot grow anymore (this proof uses the axiom of choice—a provable version in Coq) :

```
Lemma sum_stable l : poslst l → all Zeven_bool l → { k | ∀ d,
  \zsum_(0 ≤ i < size l) (ucd (k + d) l)`_i = \zsum_(0 ≤ i < size l) (ucd k l)`_i}.
```

Therefore, after a while, either the list is uniform, or the number of candies are all even after the distribution step (in other words, no need for "Increment if odd" anymore):

```
Lemma even_distribute l : poslst l → all Zeven_bool l → { k | ∀ d, uniform (ucd (k + d) l) ||
  ~~ uniform (ucd (k + d) l) && all Zeven_bool (distribute (ucd (k + d) l)) }.
```

After a While, The Sum of Squares Decreases at Each Round

Development of the sum of squares of differences
(i.e., $\sum_0^{n-2}(l_i - l_{i+1})^2 + (l_0 - l_{n-1})^2 = 2\,(\sum_0^{n-1} {l_i}^2 - \sum_0^{n-2} l_{i+1}\,l_i - l_0\,l_{n-1})$):

```
Lemma sum_squares_develop : ∀ n l, size l = n →
  \zsum_(0 ≤ i < n.-1) (l`_i - l`_i.+1) ^^ 2 + (l`_O - l`_n.-1) ^^ 2 =
```

$2 \times (\backslash zsum\_(0 \leq i < n)\ (l`\_i\ \hat{}\hat{}\ 2)$ - $\backslash zsum\_(0 \leq i < n.\text{-}1)\ (l`\_i.+1 \times l`\_i)$ - $l`\_O \times l`\_n.\text{-}1)$.

Therefore, after a while, the sum of squares of the number of candies strictly decreases at each distribution step:

```
Lemma one_iteration_decreases l : ~~ uniform l → all Zeven_bool l →
  \zsum_(0 ≤ i < size l) (distribute l)`_i ^^ 2 < \zsum_(0 ≤ i < size l) l`_i ^^ 2.
```

3 The Termination Proof

The type of outputs of the ucd algorithm starting from l and after at least k_0 steps:

```
Definition ucd_seq k0 l := sigT (fun s ⇒ { k | (k0 ≤ k)%nat × (ucd k l = s) } )%type.
```

The list inside an object of type *ucd_seq k0 l* (projection):

```
Definition lst {k0 l} (ks : ucd_seq k0 l) := match ks with existT v _ ⇒ v end.
Notation "# l" := (lst l) (at level 50).
```

Order relation between lists (parameterized by a minimum number of steps and the input of the ucd algorithm), defined by comparing the sum of squares:

```
Definition ss_lt k0 l (x y : ucd_seq k0 l ) :=
  \zsum_(0 ≤ i < size (# x)) (# x)`_i ^^ 2 < \zsum_(0 ≤ i < size (# y)) (# y)`_i ^^ 2.
```

This is a well-founded order:

```
Lemma well_founded_ss_lt k0 l : well_founded (ss_lt k0 l).
```

Starting with a list of positive, even integers, the ucd algorithm converges (by well-founded induction using the preceding order):

```
Theorem ucd_terminates l : poslst l → all Zeven_bool l →
  ∃ v, ∃ k1, ∀ k, (k1 ≤ k)%nat → ucd k l = v.
```

References

[1] Tom Bohman, Oleg Pikhurko, Alan Frieze, Danny Sleator. The Puzzle Toad. Puzzle 6: Uniform Candy Distribution. Carnegie Mellon, School of Computer Science. http://www.cs.cmu.edu/puzzle/puzzle6.html

[2] Solution to [1]. http://www.cs.cmu.edu/puzzle/puzzle6_answer.pdf

[3] INRIA. The Coq Proof Assistant. 1985–2011. http://coq.inria.fr/

[4] Georges Gonthier, Assia Mahboubi. An introduction to small scale reflection in Coq. *Journal of Formalized Reasoning* 3(2):95–152. 2010.