University of Cambridge
Computer Laboratory
Dr Tjark Weber

Easter Term 2010/11
Exercises 3: Solutions
May 27, 2011

Interactive Formal Verification (L21)

1 Sums of Powers, Polynomials

This assignment *will be assessed* to determine 50% of your final mark. Please complete the indicated tasks and write a brief document explaining your work. You may prepare this document using Isabelle's theory presentation facility, but this is not required. (A very simple way to print a theory file legibly is to use the Proof General command Isabelle > Commands > Display draft. You can combine the resulting output with a document produced using your favourite word processing package.) A clear write-up describing elegant, clearly structured proofs of all tasks will receive maximum credit.

You must work on this assignment as an individual. Collaboration is not permitted.

1.1 Sums of Powers

We consider sums of consecutive powers: $S_p(n) = \sum_{k=1}^{n} k^p$.

▷ Define a corresponding function `S p n`.

```
definition S :: "nat ⇒ nat ⇒ nat" where
  "S p n ≡ ∑k=1..n. k^p"
```

Hint: exponentiation and summation functions are already available in Isabelle/HOL.

Clearly, $S_0(n) = n$. It is also well-known that $S_1(n) = \frac{n^2+n}{2}$.

▷ Prove these identities.

```
lemma "S 0 n = n"
by (simp add: S_def)

lemma "2 * S 1 n = n^2 + n"
by (induct n) (auto simp add: S_def power2_eq_square)
```

At this point, we might suspect that $S_p(n)$ is a polynomial in n with rational coefficients.

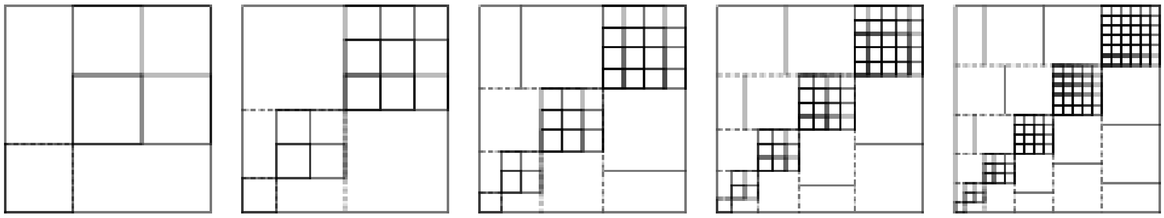

Figure 1: Visualization of Nicomachus's theorem

▷ Verify this conjecture for $p = 2$, i.e., find $k > 0$ and a polynomial *poly* in n so that $k \cdot S_2(n) = poly$. Prove the resulting identity.

```
lemma "6 * S 2 n = 2*n^3 + 3*n^2 + n"     — replace k and poly
by (induct n) (auto simp add: S_def algebra_simps power2_eq_square
power3_eq_cube)
```

Hint: useful simplification rules for addition and multiplication are available as `algebra_simps`. The *Find theorems* command can be used to discover further lemmas.

For $p = 3$, our conjecture follows from the astonishing identity $\sum_{k=1}^{n} k^3 = (\sum_{k=1}^{n} k)^2$, which is known as *Nicomachus's theorem.*

▷ Prove Nicomachus's theorem.

```
lemma l1: "4 * S 3 n = (n^2 + n)^2"
by (induct n) (auto simp add: S_def algebra_simps power2_eq_square
power3_eq_cube)

lemma l2: "4 * (m::nat) = (2 * n)^2 ⟹ m = n^2"
by (simp add: power2_eq_square)

theorem "S 3 n = (S 1 n)^2"
by (simp only: l1 l2 gauss)
```

Before we could prove our conjecture for arbitrary p (which we will not do as part of this assignment, but search for *Faulhaber's formula* if you want to know more), we need to define polynomials.

1.2 Polynomials

A polynomial in one variable can be given by the list of its coefficients: e.g., $[0, \frac{1}{6}, \frac{1}{3}, \frac{1}{2}]$ represents the polynomial $\frac{1}{2}x^3 + \frac{1}{3}x^2 + \frac{1}{6}x + 0$. (We list coefficients in reverse order, i.e., from lower to higher degree.)

Coefficients may be integers, rationals, reals, etc. In general, we require coefficients to be elements of a commutative ring (cf. `Rings.thy`).

To every polynomial in one variable we can associate a *polynomial function* on the ring of coefficients. This function's value is obtained by substituting its argument for the polynomial's variable, i.e., by evaluating the polynomial.

▷ Define a function `poly cs x` so that *poly* $[c_0, c_1, \ldots, c_n]\ x = c_n \underbrace{x \cdot \ldots \cdot x}_{n \text{ factors}} + \ldots + c_1 x + c_0$.

```
fun poly :: "'a::comm_ring list ⇒ 'a::comm_ring ⇒ 'a::comm_ring" where
  "poly [] _ = 0"
| "poly (c#cs) x = c + x * poly cs x"
```

▷ Define a function `poly_plus p q` that computes the sum of two polynomials.

```
fun poly_plus :: "'a::comm_ring list ⇒ 'a::comm_ring list ⇒ 'a::comm_ring list"
where
  "poly_plus p [] = p"
| "poly_plus [] q = q"
| "poly_plus (p#ps) (q#qs) = (p + q) # poly_plus ps qs"
```

▷ Prove correctness of `poly_plus`.

```
lemma poly_plus_correct: "poly (poly_plus p q) x = poly p x + poly q x"
by (induct p q rule: poly_plus.induct) (auto simp add: algebra_simps)
```

Hint: Isabelle provides customized induction rules for recursive functions, e.g., `poly_plus.induct`. See the *Tutorial on Function Definitions* for details.

▷ Define a function `poly_times p q` that computes the product of two polynomials.

```
fun poly_times :: "'a::comm_ring list ⇒ 'a::comm_ring list ⇒ 'a::comm_ring
list" where
  "poly_times [] _ = []"
| "poly_times (p#ps) q = poly_plus (map (op * p) q) (poly_times ps (0 # q))"
```

▷ Prove correctness of `poly_times`.

```
lemma poly_map_times: "poly (map (op * c) p) x = c * poly p x"
by (induct p) (auto simp add: algebra_simps)

lemma "poly (poly_times p q) x = poly p x * poly q x"
by (induct p q rule: poly_times.induct) (auto simp add: algebra_simps
poly_plus_correct poly_map_times)
```

▷ **Solutions are due on Friday, May 27, 2011, at 12 noon.** Please deliver a printed copy of the completed assignment to student administration by that deadline, and also send the corresponding Isabelle theory file to `tw333@cam.ac.uk`.