CS 5860 Formalizing Arithmetic continued

Thur Oct 20, 2011

PLAN

1. Eliza Kosyri will discuss an interesting feature of the homework she graded and illustrate how the ML type inference algorithm can help you understand the computational meaning of first-order statements. Her solution is very interesting and illustrates computational content that is hidden in the Boolean Logic and in the standard way of making specifications programmable.

2. We will finish giving evidence for Kleene's axioms and focus on the only axiom with computational content, induction, Ax 13.

3. ~~We'll~~ We will illustrate the standard use of induction and then examine its computational meaning.

Evidence semantics for Kleene's axioms about numbers (Group B).

We have already shown how to account for 0 using an additional axiom.

$\exists x. Z(x)$ with ~~the~~ evidence $\langle 0, * \rangle$

and how to account for successor, $s(x)$, which is Kleene's x' notation, using another axiom

$\forall x \exists y\, Succ(x,y)$ with evidence

$\lambda(x. \langle s(x), * \rangle)$

Evidence for Kleene's axioms continued

Axioms 14 & 17 can be expressed as

$a' = b' \Leftrightarrow a = b$ or in our notation

$\forall x, y. (Eq(s(x), s(y)) \Leftrightarrow Eq(x, y))$

For each direction of $\Leftrightarrow$, the evidence is the same, e.g. for $\forall x, y. (Eq(s(x), s(y)) \Rightarrow Eq(x, y))$ we use $\lambda(x, y. \lambda(e.e))$.

By Axiom 15, $\neg a' = 0$, Kleene means that for any specific number, which he denotes as bold a, say $\underline{a}$,

$$s(\underline{a}) \neq 0$$

where he takes $s(\underline{a}) \neq 0$ as a primitive (or atomic) predicate, so for him the evidence is just $*$.

We interpret this axiom as $\forall x. (s(x) = 0) \Rightarrow False$, or $\forall x. (Eq(s(x), 0) \Rightarrow False)$ in the offical syntax. The evidence is simply $\lambda(x. \lambda(y.y))$. This is correct because there is never evidence for $s(x) = 0$ for any x, thus if we assume y is evidence, then it would be evidence for False as well.

Axioms such as $a + 0 = a$ (Ax 18) we write as

$\forall x. Add(0, x, x)$, the evidence is $\lambda(x. *)$.

For $a + b' = (a+b)'$ we use a different order of arguments and postulate

$\forall x, y, z. (Add(x, y, z) \Rightarrow Add(s(x), y, s(z)))$ the evidence is $\lambda(x, y, z. \lambda(p.p))$ or $\lambda(x, y, z. \lambda(p.*))$.

Formalizing Arithmetic cont.

The evidence for axioms 18, 19 shows how to treat 20, 21. Essentially the evidence here is trivial since we are just providing witnesses for atomic relations, as in the case of equality, $Eq(x,y)$.

The only axiom with computational content is induction. We state it as

Induction Axiom $\big(A(0) \ \&\ \forall x.(A(x) \Rightarrow A(s(x)))\big) \Rightarrow \forall x. A(x)$.

Writing out the dependent types and expressing the axiom using hypotheses we have

$$b : A(0),\ f : (x : D \to A(x) \Rightarrow A(s(x))),\ x : D \vdash A(x)$$

How can this be used? If we don't try to provide more computational strength, we can see how to prove instances of $A(x)$ for any specific number, say $s(s(s(0)))$ (up to 3, citing the joke of the "engineer's induction" – its true for 0, 1, 2 and 3 so it must be true).

$A(0)$ has evidence b

$A(s(0))$ has evidence $f(0)(b)$

$A(s(s(0)))$ has $f(s(0))(f(0)(b))$

$A(s(s(s(0))))$ has $f(s(s(0)))(f(s(0))(f(0)(b)))$

But how do we get evidence for $\forall x. A(x)$? We need $\lambda(x$ expression built from $b, f)$.

Computational evidence for induction.

For the first time we need a kind of recursive evidence. Here is an ML function that gives the idea. We call the function ind for "induction."

think of the input types as

$$b : A(0), \quad f : \forall x.(A(x) \Rightarrow A(s(x))), \quad x : D$$

the output type is $A(x)$.

```
letrec ind(b, f, x) =
    if x=0 then b
    else f(x-1)(ind(b, f, x-1))
```

In Nuprl we use a slightly different form, namely

$$ind(0; b; u,i.\, f(u,i)) = b$$

$$ind(s(x); b; u,i.\, f(u,i)) = f(x, ind(x; b; u,i.\, f(u,i)))$$

This comes from the rule format in Refinement Logic for induction

$$H, x:D \vdash A(x) \text{ by } ind(x; \_\_; u,i.\, \_\_)$$

$$\vdash A(0) \text{ by } b$$

$$x:D, u:D, i:A(u-1) \vdash A(u) \text{ by } f(u,i)$$

Formalizing Arithmetic continued

We can use induction to find the computational meaning of these theorems about addition and multiplication.

Theorem 1 $\forall x, y. \exists z. \ Add(x, y, z)$

Theorem 2. $\forall x, y. \exists z. \ Mult(x, y, z)$

Try one of these as an exercise. We will solve them next time. We will also prove

Theorem 3. $\forall x, y, z. (Add(x, y, z) \Rightarrow Add(y, x, z))$

"Addition is commutative."

From Theorems 1, 2 we can build functions add(x,y) and mult(x,y). The commutativity theorem is

Theorem 3' $\forall x, y. (add(x, y) = add(y, x))$

$\forall x, y. \ Eq(add(x, y), add(y, x))$

We prove this by induction.

Before examining proofs of these theorems, we look at an intuitively appealing application of induction in the Stamps Problem. See the supplementary notes.