Lecture Notes on Representation Independence

15-814: Types and Programming Languages
Frank Pfenning

Lecture 18
Tuesday, November 5, 2019

1 Introduction

In this lecture we prove that we can replace the unary implementation of counters with the binary one without breaking any clients. This is a consequence of parametricity, and the definition of logical equality we developed in the previous two lectures. Since this lecture is a continuation of the previous one, we do not repeat the definitions here, but ask you to refer to them.

2 Defining a Relation Between Implementations

Recall that we are trying to show

$$\mathtt{BCtr} \sim \mathtt{NCtr} \in [\mathtt{CTR}]$$

which, by definition, comes down to defining a relation $R : \mathit{bin} \leftrightarrow \mathit{nat}$ such that

$$\langle E, \langle \mathit{inc}, \mathit{dec} \rangle\rangle \sim \langle Z, \langle S, \mathit{pred} \rangle\rangle \in [R \times (R \to R) \times (R \to 1 + R)]$$

The relation $R : \mathit{bin} \leftrightarrow \mathit{nat}$ we seek needs to relate natural numbers in two different representations. It is convenient and general to define such relations by using inference rules. In particular, this will allow us to prove properties by *rule induction*. An alternative approach would be to define such relations as functions, but because representations are often not unique this is not quite as general.

Once we have made this decision, the relation could be based on the structure of x : *bin* or on the structure of x : *nat*. The latter may run into difficulties because each number actually corresponds to infinitely many numbers in binary form: just add leading zeros that do not contribute to its value. Therefore, we define it based on the binary representation. In order to define it concisely, we use a representation function for (mathematical) natural numbers k into our language of values defined by

$$\begin{array}{lcl} \overline{0} & = & Z \\ \overline{n+1} & = & S\ \overline{n} \end{array}$$

We then define:

$$\frac{}{E\ R\ \overline{0}}\ R_e \qquad \frac{n\ R\ \overline{k}}{(B0\ x)\ R\ \overline{2k}}\ R_0 \qquad \frac{x\ R\ \overline{k}}{(B1\ x)\ R\ \overline{2k+1}}\ R_1$$

As usual, we consider $x\ R\ n$ to hold if and only if we can derive it using these rules.

3 Verifying the Relation

Because our signature exposes three constants, we now have to check three properties:

$$\begin{array}{l} E \sim Z \in [R] \\ \mathit{inc} \sim S \in [R \to R] \\ \mathit{dec} \sim \mathit{pred} \in [R \to 1 + R] \end{array}$$

We already have by definition that $v \sim v' \in [R]$ iff $v\ R\ v'$. For convenience, we also define the notation $e\ \mathbb{R}\ e'$ to stand for $e \approx e \in [\![R]\!]$

Lemma 1 $E \sim Z \in [R]$.

Proof: By definition $Z \sim E \in [R]$ is equivalent to $Z\ R\ E$, which follows immediately from rule R_e. □

Lemma 2 $\mathit{inc} \sim S \in [R \to R]$.

Proof: By definition of logical equality, this is equivalent to showing

For all values $x : \mathit{bin}$ and $n : \mathit{nat}$ with $x\ R\ n$ we have $(\mathit{inc}\ x)\ \mathbb{R}\ (S\ n)$.

Since R is defined inductively by a collection of inference rules, the natural attempt is to prove this by rule induction on the given relation, namely $x\ R\ n$.

Case: Rule

$$\frac{}{E\ R\ \overline{0}}\ R_e$$

with $x = E$ and $n = \overline{0}$. We have to show that $(\textit{inc}\ E)\ \mathbb{R}\ \overline{1}$.

$\textit{inc}\ E \mapsto^* B1\ E$	By defn. of *inc*
$B1\ E\ R\ \overline{1}$	By rules R_1 and R_e
$(\textit{inc}\ E)\ \mathbb{R}\ (S\ Z)$	Since $\overline{1} = S\ Z$

Case: Rule

$$\frac{x_1\ R\ \overline{k}}{(B0\ x_1)\ R\ \overline{2k}}\ R_0$$

where $x = B0\ x_1$ and $n = \overline{2k}$. To prove is $(\textit{inc}\ (B0\ x_1))\ \mathbb{R}\ \overline{2k+1}$.

$\textit{inc}\ (B0\ x_1) \mapsto^* B1\ x_1$	By defn. of *inc*
$x_1\ R\ \overline{k}$	Premise in this case
$B1\ x_1\ R\ \overline{2k+1}$	By rule R_1

Case: Rule

$$\frac{x_1\ R\ \overline{k}}{B1\ x_1\ R\ \overline{2k+1}}\ R_1$$

where $x = B1\ x_1$ and $n = \overline{2k+1}$. To show: $\textit{inc}\ (B1\ x_1)\ \mathbb{R}\ \overline{2k+2}$.

$\textit{inc}\ (B1\ x_1) \mapsto^* B0\ (\textit{inc}\ x_1)$	By defn. of *inc*
$x_1\ R\ \overline{k}$	Premise in this case
$(\textit{inc}\ x_1)\ \mathbb{R}\ \overline{k+1}$	By ind. hyp.
$\textit{inc}\ x_1 \mapsto^* x_2$ and $x_2\ R\ \overline{k+1}$	By defn. of $\mathbb{R}$
$(B0\ x_2)\ R\ \overline{2(k+1)}$	By rule R_1
$(B0\ (\textit{inc}\ x_1))\ \mathbb{R}\ \overline{2k+2}$	By defn. of $\mathbb{R}$

$\square$

In order to prove the relation between the implementation of the predecessor function we should write out the interpretation of the type $(\texttt{None} : 1) + (\texttt{Some} : R)$

$v \sim v' \in [(\texttt{None} : 1) + (\texttt{Some} : R)]$ iff ($v = \texttt{None} \cdot \langle\,\rangle$ and $v = \texttt{None} \cdot \langle\,\rangle$)
or ($v = \texttt{Some} \cdot v_1$ and $v' = \texttt{Some} \cdot v_1'$ and $v_1 \; R \; v_1'$.

Lemma 3 *dec* $\sim$ *pred* $\in [R \to ((\texttt{None} : 1) + (\texttt{Some} : R))]$

Proof: By definition of logical equality, this is equivalent to show

> *For all x : bin and n : nat with $x \; R \; n$ we have dec $x \approx$ pred $n \in$ $[(\texttt{None} : 1) + (\texttt{Some} : R)]$.*

We break this down into two properties, based on n.

(i) For all $x \; R \; \overline{0}$ we have *dec* $x \approx$ *pred* $\overline{0} \in [\![\texttt{None} : 1]\!]$.

(ii) For all $x \; R \; \overline{k+1}$ we have *dec* $x \approx$ *pred* $\overline{k+1} \in [\![\texttt{Some} : R]\!]$.

For Part (i), we note that *pred* $\overline{0} \mapsto^* \texttt{None} \cdot \langle\,\rangle$, so all that remains to show is that *dec* $x \mapsto^* \texttt{None} \cdot \langle\,\rangle$ for all $x \; R \; \overline{0}$. We prove this by rule induction on the derivation of $x \; R \; \overline{0}$.

Case(i):

$$\frac{}{E \; R \; \overline{0}} \; R_e$$

where $x = E$. Then *dec* x = *dec* $E \mapsto^* \texttt{None} \cdot \langle\,\rangle$.

Case(i):

$$\frac{x_1 \; R \; \overline{k}}{B0 \; x_1 \; R \; \overline{2k}} \; R_0$$

where $x = B0 \; x_1$ and $2k = 0$ and therefore also $k = 0$. Then

dec $(B0 \; x_1) \mapsto^*$ case (*dec* x_1) $(\texttt{None} \cdot \_ \Rightarrow \texttt{None} \cdot \langle\,\rangle \mid \texttt{Some} \; y \Rightarrow B1 \; y)$
dec $x_1 \mapsto^* \texttt{None} \cdot \langle\,\rangle$ By ind. hyp.
case (*dec* x_1) $(\texttt{None} \cdot \_ \Rightarrow \texttt{None} \cdot \langle\,\rangle \mid \texttt{Some} \; y \Rightarrow B1 \; y) \mapsto^* \texttt{None} \cdot \langle\,\rangle$

Case(i):

$$\frac{x_1 \; R \; \overline{k}}{B1 \; x_1 \; R \; \overline{2k+1}} \; R_1$$

This case is impossible since $2k + 1 \neq 0$.

Now we come to Part (ii). We note that *pred* $\overline{k+1} \mapsto^* \texttt{Some} \cdot \overline{k}$ so what we have to show is that

(ii)′ For all $x\ R\ \overline{k}$ with $k > 0$ we have *dec* $x \mapsto^* \texttt{Some}\ y$ with $y\ R\ \overline{k-1}$.

We prove this by rule induction on the derivation of $x\ R\ \overline{k+1}$.

Case(ii):

$$\frac{}{E\ R\ \overline{0}}\ R_e$$

for $x = E$ and $k = 0 > 0$, which is impossible.

Case(ii):

$$\frac{x_1\ R\ \overline{k}}{(B0\ x_1)\ R\ \overline{2k}}\ R_0$$

where $x = B0\ x_1$ and $n = \overline{2k}$ for $2k > 0$.

dec $x_1 \mapsto^* \texttt{Some} \cdot y_1$ for some $y_1\ R\ \overline{k-1}$	By ind. hyp. since $k > 0$
dec $(B0\ x_1) \mapsto^* \texttt{Some} \cdot (B1\ y_1)$	By defn. of *dec*
$(B1\ y_1)\ R\ \overline{2(k-1)+1}$	By rule R_1
$(B1\ y_1)\ R\ \overline{2k-1}$	By arithmetic

Case(ii):

$$\frac{x_1\ R\ \overline{k}}{(B1\ x_1)\ R\ \overline{2k+1}}\ R_1$$

for $x = B1\ x_1$ and $2k + 1 > 0$. Then

dec $(B1\ x_1) \mapsto^* \texttt{Some} \cdot (B0\ x_1)$	By defn. of *dec*
$x_1\ R\ \overline{k}$	Premise in this case
$(B0\ x_1)\ R\ \overline{2k}$	By rule R_0
$(B0\ x_1)\ R\ \overline{2k+1-1}$	By arithmetic

□

4 The Upshot

Because the two implementations are logically equal we can replace one implementation by the other without changing any client's behavior. This is because all clients are parametric, so their behavior does not depend on the library's implementation.

It may seem strange that this is possible because we have picked a particular relation to make this proof work. Let us reexamine the case/exists rule:

$$\frac{\Delta \,;\, \Gamma \vdash e : \exists \alpha.\, \tau \quad \Delta, \alpha\ \mathit{tp} \,;\, \Gamma, x : \tau \vdash e' : \tau'}{\Delta \,;\, \Gamma \vdash \mathbf{case}\ e\ (\langle \alpha, x \rangle \Rightarrow e') : \tau'}\ \textsf{case/exists}$$

In the second premise we see that the client e' is checked with a fresh type α and $x : \tau$ which may mention α. If we reify this into a function, we find

$$\Lambda \alpha.\, \lambda x.\, e' : \forall \alpha.\, \tau \to \tau'$$

where τ' does not depend on α.

By Reynolds's parametricity theorem we know that this function is parametric. This can now be applied for any σ and σ' and relation $R : \sigma \leftrightarrow \sigma'$ to conclude that if $v_0 \sim v_0' \in [[R/\alpha]\tau]$ then $(\Lambda \alpha \lambda x.\, e')[\sigma]\, v_0 \approx (\Lambda \alpha.\, \lambda x.\, e')[\sigma']\ v_0' \in [\![[R/\alpha]\tau']\!]$. But α does not occur in τ', so this is just saying that $[\sigma/\alpha, v_0/x]e' \approx [\sigma'/\alpha, v_0'/x]e' \in [\![\tau']\!]$. So the result of substituting the two different implementations is equivalent.

Exercises

Exercise 1 We can represent integers a as pairs $\langle x, y \rangle$ of natural numbers where $a = x - y$. We call this the *difference representation* and call the representation type *diff*.

$$\begin{array}{rcl} \mathit{nat} & \cong & (\mathtt{Z} : 1) + (\mathtt{S} : \mathit{nat}) \\ \mathit{diff} & = & \mathit{nat} \times \mathit{nat} \end{array}$$

In your answers below you may use *constructors* $Z : \mathit{nat}$ and $S : \mathit{nat} \to \mathit{nat}$ to construct as well as pattern-match subjects of type *nat*. If you need auxiliary functions on natural numbers, you should define them.

1. Define a function $\mathit{nat2diff} : \mathit{nat} \to \mathit{diff}$ that, when given a representation of the natural number n returns an integer representing n.

2. Define a constant $\mathit{d\_zero} : \mathit{diff}$ representing the integer 0 as well as a function $\mathit{dminus} : \mathit{diff} \to \mathit{diff} \to \mathit{diff}$ representing subtraction on integers.

3. Consider the type

$$ord = (\texttt{Lt} : 1) + (\texttt{Eq} : 1) + (\texttt{Gt} : 1)$$

   that represents the outcome of a comparison (`Lt` = "less than", `Eq` = "equal", `Gt` = "greater than"). Define a function *dcompare* : *diff* → *diff* → *ord* to compare the two integer arguments. Again, you may use *Lt*, *Eq* and *Gt* as constructors.

Exercise 2 We consider an alternative *signed representation* of integers where

$$sign = (\texttt{Pos} : nat) + (\texttt{Neg} : nat)$$

where $\texttt{Pos} \cdot x$ represents the integer x and $\texttt{Neg} \cdot x$ represents the integer $-x$. In your answers below you may use *Pos* and *Neg* as data constructors, both to construct elements of type *sign* and for pattern matching. Define the following functions in analogy with Exercise 1:

1. *nat2sign* : *nat* → *sign*
2. *s_zero* : *sign*
3. *s_minus* : *sign* → *sign* → *sign*
4. *s_compare* : *sign* → *sign* → *ord*

Exercise 3 In this exercise we pursue two different implementations of an integer counter, which can become negative (unlike the natural number counter in this lecture). The functions are simpler than the ones in Exercises 1 and 2 so that the logical equality argument is more manageable. We specify a signature

```
INTCTR = {
  type ictr
  zero : ictr
  inc : ictr -> ictr
  dec : ictr -> ictr
  is0 : ictr -> bool
}
```

where *zero*, *inc*, *dec* and *is0* have their obvious specification with respect to integers.

1. Write out the definition of *INTCTR* as an existential type.

2. Define the constants and functions *d_zero*, *d_inc*, *d_dec* and *d_is0* for the implementation where type $ictr = diff$ from Exercise 1.

3. Define the constants and functions *szero*, *s_inc*, *s_dec* and *s_is0* for the implementation where type $ictr = sign$ from Exercise 2.

Now consider the two definitions

$$\begin{aligned} &DCtr : INTCTR = \langle diff, \langle d\_zero, \langle d\_inc, \langle d\_dec, d\_is0 \rangle\rangle\rangle\rangle \\ &SCtr : INTCTR = \langle sign, \langle s\_zero, \langle s\_inc, \langle s\_dec, s\_is0 \rangle\rangle\rangle\rangle \end{aligned}$$

4. Prove that $DCtr \sim SCtr in [INTCTR]$ by defining a suitable relation $R : diff \leftrightarrow sign$ and proving that

$$\begin{aligned} \langle d\_zero, \langle d\_inc, \langle d\_dec, d\_is0 \rangle\rangle\rangle &\sim \langle s\_zero, \langle s\_inc, \langle s\_dec, s\_is0 \rangle\rangle\rangle \\ &\in [R \times (R \to R) \times (R \to R) \times (R \to bool)] \end{aligned}$$