## 5.6 Case Study: Integer Square Root

**5.6.1 Introduction.** In this section we will be the problem of computing the integer square root $\lfloor\sqrt{x}\rfloor$ of a natural number $x$. The function $\lfloor\sqrt{x}\rfloor$ can be introduced into PA by the following contextual definition:

$$\lfloor\sqrt{x}\rfloor = y \leftrightarrow y^2 \leq x < (y+1)^2.$$

We also intend to demonstrate that simple recursion such as primitive recursion does not always lead to a definition which, when used as rewriting rules, is efficient. A computationally optimal definition usually needs a more complex recursion/discrimination.

**5.6.2 Top-down program by primitive recursion.** We can find a primitive recursive derivation of the integer square root function by assuming as IH that

$$\lfloor\sqrt{x}\rfloor^2 \leq x < (\lfloor\sqrt{x}\rfloor + 1)^2$$

holds and then considering the relation between $x+1$ and $(\lfloor\sqrt{x}\rfloor+1)^2$. If

$$\lfloor\sqrt{x}\rfloor^2 \leq x < x+1 < (\lfloor\sqrt{x}\rfloor + 1)^2$$

then it clearly suffices to set $\lfloor\sqrt{x+1}\rfloor := \lfloor\sqrt{x}\rfloor$. Otherwise we must have

$$\lfloor\sqrt{x}\rfloor^2 \leq x < (\lfloor\sqrt{x}\rfloor + 1)^2 = x+1 < (\lfloor\sqrt{x}\rfloor + 2)^2$$

and it suffices to set $\lfloor\sqrt{x+1}\rfloor := \lfloor\sqrt{x}\rfloor + 1$. The idea is expressed by the following regular recursive definition:

$$\begin{array}{l}\lfloor\sqrt{0}\rfloor = 0\\ \lfloor\sqrt{x+1}\rfloor = \lfloor\sqrt{x}\rfloor \leftarrow x+1 < (\lfloor\sqrt{x}\rfloor+1)^2\\ \lfloor\sqrt{x+1}\rfloor = \lfloor\sqrt{x}\rfloor + 1 \leftarrow x+1 \geq (\lfloor\sqrt{x}\rfloor+1)^2.\end{array}$$

**5.6.3 Top-down program with assignment.** The program for $\lfloor\sqrt{x}\rfloor$ from Par. 5.6.2, though quite pleasant mathematically, is infeasible in practice for many reasons. One of them is the twofold occurrence of $\lfloor\sqrt{x}\rfloor$ in the recursive clauses: once in the test and once in the result. This leads to the exponential explosion of computation time. The explosion can be prevented by an *assignment* $\lfloor\sqrt{x}\rfloor = r$ in the following definition:

$$\begin{array}{l}\lfloor\sqrt{0}\rfloor = 0\\ \lfloor\sqrt{x+1}\rfloor = y \leftarrow \lfloor\sqrt{x}\rfloor = y \wedge x+1 < (y+1)^2\\ \lfloor\sqrt{x+1}\rfloor = y+1 \leftarrow \lfloor\sqrt{x}\rfloor = y \wedge x+1 \leq (y+1)^2.\end{array}$$

Note that only $x$ tests are needed.

**5.6.4 Bottom-up program by backward recursion.** The last program for $\lfloor\sqrt{x}\rfloor$ which uses primitive recursive and assignments can be improved as follows. Primitive recursion is an example of top-down approach for solving problems; in this case the computation of $\lfloor\sqrt{x}\rfloor$ goes down from $x$ to $x-1$ until it reaches 0 and then it does the comparisons and the incrementation by one on the way back. We will shorten the computation of $\lfloor\sqrt{x}\rfloor$ by using a bottom-up approach in which we search (starting from 0) the smallest number $y$ such that $x < (y+1)^2$.

This is done with the help the binary function $f(x,r)$ defined by

$$f(x,y) = y \leftarrow x < (y+1)^2$$
$$f(x,y) = f(x,y+1) \leftarrow x \geq (y+1)^2.$$

This is an example of definition by *backward recursion*, where $y^2$ grows towards $x$, i.e. by recursion with measure $x \dot{-} y^2$. Its condition of regularity

$$\vdash_{\mathrm{PA}} \; x \geq (y+1)^2 \rightarrow x \dot{-} (y+1)^2 < x \dot{-} y^2$$

is trivially satisfied.

The auxiliary function satisfies

$$\vdash_{\mathrm{PA}} \; y^2 \leq x \rightarrow f(x,y)^2 \leq x < (f(x,y)+1)^2 \tag{1}$$

and thus we can take the following identity

$$\vdash_{\mathrm{PA}} \; \lfloor\sqrt{x}\rfloor = f(x,0)$$

as an alternative program for computing the integer square root function. Note that computation of $\lfloor\sqrt{x}\rfloor$ now takes order $\lfloor\sqrt{x}\rfloor$ time.

It remains to show that the auxiliary function satisfies the invariant (1). The property is proved by backward induction on the difference $x \dot{-} y^2$. So take any $x, y$ such that $y^2 \leq x$ and consider two cases. If $x < (y+1)^2$ then the claim follows from

$$f(x,y)^2 = y^2 \leq x < (y+1)^2 = \big(f(x,y)+1\big)^2.$$

If $x \geq (y+1)^2$ then, by IH applied to $x$ and $y+1$, the claim follows from

$$f(x,y)^2 = f(x,y+1)^2 \overset{\text{IH}}{\leq} x \overset{\text{IH}}{<} \big(f(x,y+1)+1\big)^2 = \big(f(x,y)+1\big)^2.$$

**5.6.5 Bottom-up program with accumulator.** We can improve the bottom-up program for $\lfloor\sqrt{x}\rfloor$ by saving the squaring operation $(y+1)^2$ in the test $x < (y+1)^2$ which is repeatedly done for every recursive call. This can be done with the help of a ternary function $f(x,y,s)$ with the additional accumulator $s$ such that we have $f(x,y,y^2) = \lfloor\sqrt{x}\rfloor$ provided $y^2 \leq x$. As the second argument goes from $y$ to $y+1$ the accumulator goes from $s = y^2$ to $s_1 = (y+1)^2 = s + 2y + 1$. This arrangement reduces the squaring operation

to the increment $2y+1$ which is very fast in binary representation of natural numbers.

The auxiliary accumulator function $f(x,y,s)$ is defined by backward recursion on the difference $x \dot{-} y^2$:

$$f(x,y,s) = y \leftarrow s + 2y + 1 = s_1 \wedge x < s_1$$
$$f(x,y,s) = f(x,y+1,s_1) \leftarrow s + 2y + 1 = s_1 \wedge x \geq s_1.$$

Its condition of regularity

$$\vdash_{\mathrm{PA}} \; s + 2y + 1 = s_1 \wedge x \geq s_1 \rightarrow x \dot{-} (y+1)^2 < x \dot{-} y^2$$

is trivially satisfied. The function satisfies

$$\vdash_{\mathrm{PA}} \; y^2 \leq x \rightarrow f(x,y,y^2)^2 \leq x < \big(f(x,y,y^2)+1\big)^2 \tag{1}$$

and thus we can take the following identity

$$\vdash_{\mathrm{PA}} \; \lfloor \sqrt{x} \rfloor = f(x,0,0)$$

as an alternative program for computing the integer square root function.

It remains to show that the auxiliary function satisfies the invariant (1). The property is proved by backward induction on the difference $x \dot{-} y^2$. So take any $x, y$ such that $y^2 \leq x$ and consider two cases. If $x < (y+1)^2$ then the claim follows from

$$f(x,y,y^2)^2 = y^2 \leq x < (y+1)^2 = \big(f(x,y,y^2)+1\big)^2.$$

If $x \geq (y+1)^2$ then, by IH applied to $x$ and $y+1$, the claim follows from

$$\begin{aligned} f(x,y,y^2)^2 = f\big(x,y+1,(y+1)^2\big)^2 &\overset{\text{IH}}{\leq} x \overset{\text{IH}}{\leq} \\ &< \Big(f\big(x,y+1,(y+1)^2\big)+1\Big)^2 = \big(f(x,y,y^2)+1\big)^2. \end{aligned}$$

**5.6.6 Fast program by recursion on notation.** All programs considering so far share the same shortcomings: recursion goes exponentially longer that it should. We can obtain a fast $\mathcal{O}(\lg(x))$ program for $\lfloor \sqrt{x} \rfloor$ by recalling a high school algorithm working with the decimal, or rather centennial, notation because it considers two decimal digits at a time. The recursion does not work well in the decimal notation because we have $\lfloor \sqrt{10x} \rfloor \approx \lfloor \sqrt{10} \rfloor \lfloor \sqrt{x} \rfloor$ while it works well in the centennial notation because $\lfloor \sqrt{100x} \rfloor \approx 10 \lfloor \sqrt{x} \rfloor$. The same works in the 4-*ary representation* of natural numbers, where for every $x$ there are unique numbers $y$ and $z$ such that $x = 4y + z$ and $z < 4$ holds. Note that we then have $\lfloor \sqrt{4x} \rfloor \approx 2 \lfloor \sqrt{x} \rfloor$.

Thus assume as IH that we have for $y \neq 0$

$$\lfloor \sqrt{y} \rfloor^2 \leq y < (\lfloor \sqrt{y} \rfloor + 1)^2.$$

From the last we obtain $y \leq \left\lfloor \sqrt{y} \right\rfloor^2 + 2\left\lfloor \sqrt{y} \right\rfloor$ and so we get for $z < 4$:

$$\begin{aligned}(2\left\lfloor \sqrt{y} \right\rfloor)^2 \leq 4y \leq 4y + z \leq 4\left\lfloor \sqrt{y} \right\rfloor^2 + 8\left\lfloor \sqrt{y} \right\rfloor + z < \\ < 4\left\lfloor \sqrt{y} \right\rfloor^2 + 8\left\lfloor \sqrt{y} \right\rfloor + 4 = (2\left\lfloor \sqrt{y} \right\rfloor + 2)^2.\end{aligned}$$

This means that we have

$$2\left\lfloor \sqrt{y} \right\rfloor \leq \left\lfloor \sqrt{4y + z} \right\rfloor \leq 2\left\lfloor \sqrt{y} \right\rfloor + 1$$

and so the following identity is a fast program for $\left\lfloor \sqrt{x} \right\rfloor$:

$$\begin{aligned}&\left\lfloor \sqrt{x} \right\rfloor = (z \neq_* 0) \leftarrow x = 4y + z \wedge z < 4 \wedge y = 0 \\ &\left\lfloor \sqrt{x} \right\rfloor = r \leftarrow x = 4y + z \wedge z < 4 \wedge y \neq 0 \wedge 2\left\lfloor \sqrt{y} \right\rfloor = r \wedge x < (r+1)^2 \\ &\left\lfloor \sqrt{x} \right\rfloor = r + 1 \leftarrow x = 4y + z \wedge z < 4 \wedge y \neq 0 \wedge 2\left\lfloor \sqrt{y} \right\rfloor = r \wedge x \geq (r+1)^2.\end{aligned}$$

The following is its condition of regularity

$$\vdash_{\mathrm{PA}} \; x = 4y + z \wedge z < 4 \wedge y \neq 0 \rightarrow y < x,$$

which is trivially satisfied. Note that the expression on the right-hand side of the defining equation applies two assignments and two conditionals.

The first construct is assignment and uses the fast pattern matching with the *numeric pattern* $x = 4y + z \wedge z < 4$. The pattern is satisfied for every $x$ since there exist (unique) numbers $y, z$ satisfying the identity. It can be viewed as a generalization of *let*-constructs from functional programming languages. Note that the *local* variables $y, z$ are then referred later on. The pattern matching is fast, just consider the last two binary digits of the binary representation of the number $x$.

The reader will also note that the second assignment $2\left\lfloor \sqrt{y} \right\rfloor = r$ is crucial to the speed of the program because without it we would have two recursive invocations: once in the test and once in the result. Consequently, as the depth of recursion is $\mathcal{O}(\lg(x))$, we would have $\mathcal{O}(2^{\lg(x)}) = \mathcal{O}(x)$ recursive invocations and the definition would not be any faster than the one by primitive recursion.