Parametric Insertion Sort
Justin Raymond
December 1, 2015

1 Parametric Insertion Sort

Parametric insertion sort is a higher order algorithm which sorts a list using a comparison function which is passed to it as an argument. The running time of insertion sort is $\mathcal{O}(n^2)$. This characterization of the complexity of parametric insertion sort does not capture role of the comparison function in the running time. When sorting a list of integers, where comparison between any two integers takes constant time, this does not matter. However, when sorting a list of strings, where the complexity of comparison is order the length of the string, the length of the strings may influence the running time more than the length of the list when sorting small lists of large strings.

```
data list = Nil of unit | Cons of int × list

insert = λf.λx.λxs.rec(xs, Nil↦ Cons⟨x,Nil⟩,
                        Cons↦ ⟨y,⟨ys,r⟩⟩.rec(f x y, True ↦ Cons⟨x,Cons⟨y,ys⟩⟩,
                                                   False ↦ Cons⟨y,force(r)⟩))

sort = λf.λxs.rec(xs, Nil↦ Nil, Cons↦ ⟨y,⟨ys,r⟩⟩.insert f y force(r))
```

Figure 1: Parametric insertion sort in the source language

1.1 Insert

The function `sort` relies on the function `insert` to insert the head of the list into the result of recursively sorted tail of the list. We will begin with a translation and interpretation of `insert`.

1.1.1 Translation

The translation of `insert` is broken into chunks to make it more manageable. Figure 2 steps through the translation of the comparison function `f` applied to variables `x` and `y`.

$$
\begin{aligned}
&\|\mathtt{f\ \ x\ \ y}\| = (1 + \|\mathtt{f\ \ x}\|_c + \|\mathtt{y}\|_c) +_c \|\mathtt{f\ \ x}\|_p\|\mathtt{y}\|_p \\
&\quad = (1 + 1 + \|\mathtt{f}\|_c + \|\mathtt{x}\|_c + (\|\mathtt{f}\|_p\|\mathtt{x}\|_p)_c + \|\mathtt{y}\|_c) +_c \|\mathtt{f}\|_p\|\mathtt{x}\|_p\|\mathtt{y}\|_p \\
&\quad = (2 + \langle \mathtt{0\,,f}\rangle_c + \langle \mathtt{0\,,x}\rangle_c + \langle \mathtt{0\,,y}\rangle_c) +_c \langle \mathtt{0\,,f}\rangle_p\langle \mathtt{0\,,x}\rangle_p\langle \mathtt{0\,,y}\rangle_p \\
&\quad = (2 + 0 + 0 + 0) +_c \mathtt{(f\ \ x\ \ y)} \\
&\quad = \langle 2 + ((\mathtt{f\ \ x})_p\ \ \mathtt{y})_c, ((\mathtt{f\ \ x})_p\ \ \mathtt{y})_p\rangle
\end{aligned}
$$

Figure 2: Translation of `(f:Int→Int→Bool) (x:Int) (y:Int)`. We assume `f`, `x` and `y` are variables, so the cost of their translation is 0. Since `f` is a function of two arguments, the cost of `f` is 0 unless `f` is fully applied. Notice that `f` in $\|\mathtt{f}\|$ is a source variable while plain `f` is a potential variable.

The translation `true` and `false` branches are given in figures 3 and 4 respectively.

$$
\begin{aligned}
&\|\mathtt{True}{\mapsto}\mathtt{Cons}\langle\mathtt{x\,,Cons}\langle\mathtt{y\,,ys}\rangle\rangle\| \\
&\quad = \mathtt{True}{\mapsto}\ 1 +_c \|\mathtt{Cons}\langle\mathtt{x\,,Cons}\langle\mathtt{y\,,ys}\rangle\rangle\| \\
&\quad = \mathtt{True}{\mapsto}\ 1 +_c \langle\|\langle\mathtt{x\,,Cons}\langle\mathtt{y\,,ys}\rangle\rangle\|_c,\mathtt{Cons}\|\langle\mathtt{x\,,Cons}\langle\mathtt{y\,,ys}\rangle\rangle\|_p\rangle \\
&\quad = \mathtt{True}{\mapsto}\ 1 +_c \langle\langle\|\mathtt{x}\|_c + \|\mathtt{Cons}\langle\mathtt{y\,,ys}\rangle\|_c, \langle\|\mathtt{x}\|_p, \|\mathtt{Cons}\langle\mathtt{y\,,ys}\rangle\|_p\rangle\rangle_c\,, \\
&\qquad\qquad\quad \mathtt{Cons}\langle\|\mathtt{x}\|_c + \|\mathtt{Cons}\langle\mathtt{y\,,ys}\rangle\|_c, \langle\|\mathtt{x}\|_p, \|\mathtt{Cons}\langle\mathtt{y\,,ys}\rangle\|_p\rangle\rangle_p\rangle \\
&\quad = \mathtt{True}{\mapsto}\ 1 +_c \langle\|\mathtt{x}\|_c + \|\mathtt{Cons}\langle\mathtt{y\,,ys}\rangle\|_c,\mathtt{Cons}\langle\|\mathtt{x}\|_p, \|\mathtt{Cons}\langle\mathtt{y\,,ys}\rangle\|_p\rangle\rangle \\
&\quad = \mathtt{True}{\mapsto}\ 1 +_c \langle\langle\mathtt{0\,,x}\rangle_c + \langle\|\langle\mathtt{y\,,ys}\rangle\|_c,\mathtt{Cons}\|\langle\mathtt{y\,,ys}\rangle\|_p\rangle_c\,, \\
&\qquad\qquad\quad \mathtt{Cons}\langle\langle\mathtt{0\,,x}\rangle_p, \langle\|\langle\mathtt{y\,,ys}\rangle\|_c,\mathtt{Cons}\|\langle\mathtt{y\,,ys}\rangle\|_p\rangle_p\rangle\rangle \\
&\quad = \mathtt{True}{\mapsto}\ 1 +_c \langle 0 + \|\langle\mathtt{y\,,ys}\rangle\|_c\,,\mathtt{Cons}\langle\mathtt{x\,,Cons}\|\langle\mathtt{y\,,ys}\rangle\|_p\rangle\rangle \\
&\quad = \mathtt{True}{\mapsto}\ 1 +_c \langle\langle\|\mathtt{y}\|_c + \|\mathtt{ys}\|_c, \langle\|\mathtt{y}\|_p, \|\mathtt{ys}\|_p\rangle\rangle_c\,,\mathtt{Cons}\langle\mathtt{x\,,Cons}\langle\|\mathtt{y}\|_c + \|\mathtt{ys}\|_c, \langle\|\mathtt{y}\|_p, \|\mathtt{ys}\|_p\rangle\rangle_p\rangle\rangle \\
&\quad = \mathtt{True}{\mapsto}\ 1 +_c \langle\|\mathtt{y}\|_c + \|\mathtt{ys}\|_c\,,\mathtt{Cons}\langle\mathtt{x\,,Cons}\langle\|\mathtt{y}\|_p, \|\mathtt{ys}\|_p\rangle\rangle\rangle \\
&\quad = \mathtt{True}{\mapsto}\ 1 +_c \langle\langle\mathtt{0\,,y}\rangle_c + \langle\mathtt{0\,,ys}\rangle_c\,,\mathtt{Cons}\langle\mathtt{x\,,Cons}\langle\langle\mathtt{0\,,y}\rangle_p, \langle\mathtt{0\,,ys}\rangle_p\rangle\rangle\rangle \\
&\quad = \mathtt{True}{\mapsto}\ 1 +_c \langle 0,\mathtt{Cons}\langle\mathtt{x\,,Cons}\langle\mathtt{y\,,ys}\rangle\rangle\rangle \\
&\quad = \mathtt{True}{\mapsto}\ \langle 1,\mathtt{Cons}\langle\mathtt{x\,,Cons}\langle\mathtt{y\,,ys}\rangle\rangle\rangle
\end{aligned}
$$

Figure 3: Translation of `True→Cons⟨x,Cons⟨y,ys⟩⟩` in the inner `rec` of `insert`. In this case the element we are inserting into the list comes before the head of the list under the ordering given by `f`.

$$
\begin{aligned}
&\|\texttt{False}\mapsto\texttt{Cons}\langle\texttt{y},\texttt{force(r)}\rangle\| \\
&\quad= \texttt{False}\mapsto 1 +_c \|\texttt{Cons}\langle\texttt{y},\texttt{force(r)}\rangle\| \\
&\quad= \texttt{False}\mapsto 1 +_c \langle\|\langle\texttt{y},\texttt{force(r)}\rangle\|_c, \texttt{Cons}\|\langle\texttt{y},\texttt{force(r)}\rangle\|_p\rangle \\
&\quad= \texttt{False}\mapsto 1 +_c \langle\langle\|\texttt{y}\|_c + \|\texttt{force(r)}\|_c, \langle\|\texttt{y}\|_p, \|\texttt{force(r)}\|_p\rangle\rangle_c, \\
&\qquad\qquad\quad \texttt{Cons}\langle\|\texttt{y}\|_c + \|\texttt{force(r)}\|_c, \langle\|\texttt{y}\|_p, \|\texttt{force(r)}\|_p\rangle\rangle_p\rangle \\
&\quad= \texttt{False}\mapsto 1 +_c \langle\|\texttt{y}\|_c + \|\texttt{force(r)}\|_c, \texttt{Cons}\langle\|\texttt{y}\|_p, \|\texttt{force(r)}\|_p\rangle\rangle \\
&\quad= \texttt{False}\mapsto 1 +_c \langle\langle\texttt{0},\texttt{y}\rangle_c + (\|\texttt{r}\|_c +_c \|\texttt{r}\|_p)_c, \texttt{Cons}\langle\langle\texttt{0},\texttt{y}\rangle_p, (\|\texttt{r}\|_c +_c \|\texttt{r}\|_p)\rangle\rangle \\
&\quad= \texttt{False}\mapsto 1 +_c \langle 0+\texttt{r}_c, \texttt{Cons}\langle\texttt{y},\texttt{r}_p\rangle\rangle \\
&\quad= \texttt{False}\mapsto \langle 1+\texttt{r}_c, \texttt{Cons}\langle\texttt{y},\texttt{r}_p\rangle\rangle
\end{aligned}
$$

Figure 4: Translation of the `False` branch of the inner `rec` of `insert`. `r` stands for the recursive call, and has type `susp list`. In this case the element we are inserting into the list comes after the head of the list under the ordering given by `f`.

Figure 5 uses the translation of `f x y` and the `true` and `false` branches to construct the translation of the inner `rec` construct.

$$
\begin{aligned}
&\|\texttt{rec(f x y,True}\mapsto\texttt{Cons}\langle\texttt{x},\texttt{Cons}\langle\texttt{y},\texttt{ys}\rangle\rangle\texttt{,False}\mapsto\texttt{Cons}\langle\texttt{y},\texttt{force(r)}\rangle\texttt{)}\| \\
&\quad= \|\texttt{f x y}\|_c +_c \texttt{rec(}\|\texttt{f x y}\|_p\texttt{,True}\mapsto 1 +_c \|\texttt{Cons}\langle\texttt{x},\texttt{Cons}\langle\texttt{y},\texttt{ys}\rangle\rangle\|, \\
&\qquad\qquad\qquad\qquad\quad \texttt{False}\mapsto 1 +_c \|\texttt{Cons}\langle\texttt{y},\texttt{force(r)}\rangle\|\texttt{)} \\
&\quad= 2 + ((\|\texttt{f}\|_p\ \texttt{x})_p\ \texttt{y})_c +_c \texttt{rec(}((\|\texttt{f}\|_p\ \texttt{x})_p\ \texttt{y})_p, \\
&\qquad\qquad\qquad\qquad\quad \texttt{True}\mapsto 1 +_c \|\texttt{Cons}\langle\texttt{x},\texttt{Cons}\langle\texttt{y},\texttt{ys}\rangle\rangle\|, \\
&\qquad\qquad\qquad\qquad\quad \texttt{False}\mapsto 1 +_c \|\texttt{Cons}\langle\texttt{y},\texttt{force(r)}\rangle\|\texttt{)} \\
&\quad= 2 + ((\|\texttt{f}\|_p\ \texttt{x})_p\ \texttt{y})_c +_c \texttt{rec(}((\|\texttt{f}\|_p\ \texttt{x})_p\ \texttt{y})_p, \\
&\qquad\qquad\qquad\qquad\quad \texttt{True}\mapsto \langle 1,\texttt{Cons}\langle\texttt{x},\texttt{Cons}\langle\texttt{y},\texttt{ys}\rangle\rangle\rangle \\
&\qquad\qquad\qquad\qquad\quad \texttt{False}\mapsto 1 +_c \|\texttt{Cons}\langle\texttt{y},\texttt{force(r)}\rangle\|\texttt{)} \\
&\quad= 2 + ((\|\texttt{f}\|_p\ \texttt{x})_p\ \texttt{y})_c +_c \texttt{rec(}((\|\texttt{f}\|_p\ \texttt{x})_p\ \texttt{y})_p, \\
&\qquad\qquad\qquad\qquad\quad \texttt{True}\mapsto \langle 1,\texttt{Cons}\langle\texttt{x},\texttt{Cons}\langle\texttt{y},\texttt{ys}\rangle\rangle\rangle \\
&\qquad\qquad\qquad\qquad\quad \texttt{False}\mapsto \langle 1+\texttt{r}_c,\texttt{Cons}\langle\texttt{y},\texttt{r}_p\rangle\rangle\texttt{)}
\end{aligned}
$$

Figure 5: Translation of the inner `rec` in `insert`. In the `True` case, we have found the place of `x` in the list and we so stop. In the `False` case, `x` comes after the head of list under the ordering given by `f` and we must recurse on the tail of the list.

The `Nil` and `Cons` branches of the outer `rec` construct are given in figures 6 and 7, respectively.

$$\begin{aligned}
&\|\texttt{Nil}\mapsto\texttt{Cons}\langle\texttt{x},\texttt{Nil}\rangle\|\\
&\quad= \texttt{Nil}\mapsto 1 +_c \|\texttt{Cons}\langle\texttt{x},\texttt{Nil}\rangle\|\\
&\quad= \texttt{Nil}\mapsto 1 +_c \langle\|\langle\texttt{x},\texttt{Nil}\rangle\|_c, \texttt{Cons}\|\langle\texttt{x},\texttt{Nil}\rangle\|_p\rangle\\
&\quad= \texttt{Nil}\mapsto 1 +_c \langle\langle\|\texttt{x}\|_c + \|\texttt{Nil}\|_c, \langle\|\texttt{x}\|_p, \|\texttt{Nil}\|_p\rangle\rangle_c, \texttt{Cons}\langle\|\texttt{x}\|_c + \|\texttt{Nil}\|_c, \langle\|\texttt{x}\|_p, \|\texttt{Nil}\|_p\rangle\rangle_p\rangle\\
&\quad= \texttt{Nil}\mapsto 1 +_c \langle\|\texttt{x}\|_c + \|\texttt{Nil}\|_c, \texttt{Cons}\langle\|\texttt{x}\|_p, \|\texttt{Nil}\|_p\rangle\rangle\\
&\quad= \texttt{Nil}\mapsto 1 +_c \langle\langle\texttt{0},\texttt{x}\rangle_c + \langle\texttt{0},\texttt{Nil}\rangle_c, \texttt{Cons}\langle\langle\texttt{0},\texttt{x}\rangle_p, \langle\texttt{0},\texttt{Nil}\rangle_p\rangle\rangle\\
&\quad= \texttt{Nil}\mapsto 1 +_c \langle 0+0, \texttt{Cons}\langle\texttt{x},\texttt{Nil}\rangle\rangle\\
&\quad= \texttt{Nil}\mapsto \langle 1, \texttt{Cons}\langle\texttt{x},\texttt{Nil}\rangle\rangle
\end{aligned}$$

Figure 6: Translation of the `Nil` branch of the outer `rec` in `insert`. The insertion of an element into an empty list results in a singleton list containing only the element. This branch is also reached when the ordering given by `f` dictates `x` comes after than everything in the list, and should be placed at the back of the list.

$$\begin{aligned}
&\|\texttt{Cons}\mapsto \langle\texttt{y},\langle\texttt{ys},\texttt{r}\rangle\rangle.\texttt{rec(f x y, True} \mapsto \texttt{Cons}\langle\texttt{x},\texttt{Cons}\langle\texttt{y},\texttt{ys}\rangle\rangle,\\
&\qquad\qquad\qquad\qquad\qquad\quad \texttt{False} \mapsto \texttt{Cons}\langle\texttt{y},\texttt{force(r)}\rangle\texttt{)}\|\\
&\quad= \texttt{Cons}\mapsto \langle\texttt{y},\langle\texttt{ys},\texttt{r}\rangle\rangle.1 +_c \|\texttt{rec(f x y, True} \mapsto \texttt{Cons}\langle\texttt{x},\texttt{Cons}\langle\texttt{y},\texttt{ys}\rangle\rangle,\\
&\qquad\qquad\qquad\qquad\qquad\quad \texttt{False} \mapsto \texttt{Cons}\langle\texttt{y},\texttt{force(r)}\rangle\texttt{)}\|\\
&\quad= \texttt{Cons}\mapsto \langle\texttt{y},\langle\texttt{ys},\texttt{r}\rangle\rangle.1 +_c (2 + ((\texttt{f x})_p\ \texttt{y})_c) +_c \texttt{rec}(((\texttt{f x})_p\ \texttt{y})_p,\\
&\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad \texttt{True}\mapsto \langle 1, \texttt{Cons}\langle\texttt{x},\texttt{Cons}\langle\texttt{y},\texttt{ys}\rangle\rangle\rangle\\
&\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad \texttt{False}\mapsto \langle 1+\texttt{r}_c, \texttt{Cons}\langle\texttt{y},\texttt{r}_p\rangle\rangle))\\
&\quad= \texttt{Cons}\mapsto \langle\texttt{y},\langle\texttt{ys},\texttt{r}\rangle\rangle.(3 + ((\texttt{f x})_p\ \texttt{y})_c) +_c \texttt{rec}(((\texttt{f x})_p\ \texttt{y})_p,\\
&\qquad\qquad\qquad\qquad\qquad\qquad\qquad \texttt{True}\mapsto \langle 1, \texttt{Cons}\langle\texttt{x},\texttt{Cons}\langle\texttt{y},\texttt{ys}\rangle\rangle\rangle\\
&\qquad\qquad\qquad\qquad\qquad\qquad\qquad \texttt{False}\mapsto \langle 1+\texttt{r}_c, \texttt{Cons}\langle\texttt{y},\texttt{r}_p\rangle\rangle))
\end{aligned}$$

Figure 7: Translation of the `Cons` branch of the outer `rec` in `insert`. In this branch we recurse on a nonempty list. We check if `x` is comes before the head of the list under the ordering given by `f`, in which case we are done, otherwise we recurse on the tail of the list.

We put these together to give the translation of `insert`.

```
‖insert‖ = ‖λf.λx.λxs.rec(xs, Nil↦ Cons⟨x,Nil⟩,
                    Cons↦ ⟨y,⟨ys,r⟩⟩.rec(f x y, True ↦ Cons⟨x,Cons⟨y,ys⟩⟩,
                                              False ↦ Cons⟨y,force(r)⟩))‖

  = ⟨0,λf.⟨0,λx.⟨0,λxs.‖rec(xs, Nil↦ Cons⟨x,Nil⟩,
                         Cons↦ ⟨y,⟨ys,r⟩⟩.rec(f x y, True ↦ Cons⟨x,Cons⟨y,ys⟩⟩,
                                                   False ↦ Cons⟨y,force(r)⟩))‖⟩⟩⟩

  = ⟨0,λf.⟨0,λx.⟨0,λxs.⟨0,xs⟩_c+_c
         rec(⟨0,xs⟩_p,
             Nil↦ ⟨1,Cons⟨x,Nil⟩⟩
             Cons↦ ⟨y,⟨ys,r⟩⟩.(3 + (((⟨0,f⟩_p x)_p y)_c)+_c rec(((⟨0,f⟩_p x)_p y)_p,
                                                   True↦ ⟨1,Cons⟨x,Cons⟨y,ys⟩⟩⟩
                                                   False↦ ⟨1+r_c,Cons⟨y,r_p⟩⟩))

  = ⟨0,λf.⟨0,λx.⟨0,λxs.
         rec(xs,
             Nil↦ ⟨1,Cons⟨x,Nil⟩⟩
             Cons↦ ⟨y,⟨ys,r⟩⟩.(3 + ((f x)_p y)_c)+_c rec((f x)_p y)_p,
                                                 True↦ ⟨1,Cons⟨x,Cons⟨y,ys⟩⟩⟩
                                                 False↦ ⟨1+r_c,Cons⟨y,r_p⟩⟩))
```

Figure 8: Translation of `insert`

Finally we give a translation of `insert f x xs` in figure 9 because this is the term we will interpret in a size-based semantics.

```
  ‖insert f x xs‖ = (1 + ‖insert f x‖_c + ‖xs‖_c) +_c ‖insert f x‖_p‖xs‖_p

     = (1 + ‖insert f x‖_c + ‖xs‖_c) +_c ‖insert f x‖_p‖xs‖_p

     = (2 + ‖insert f‖_c + ‖x‖_c + (‖insert f‖_p‖x‖_p)_c + ‖xs‖_c) +_c ‖insert f‖_p‖x‖_p‖xs‖_p

     = (2 + ‖insert f‖_c + ‖x‖_c + ‖xs‖_c) +_c ‖insert f‖_p‖x‖_p‖xs‖_p

     = (3 + ‖insert‖_c + ‖f‖_c + (‖insert‖_p‖f‖_p‖x‖_p)_c + ‖x‖_c + ‖xs‖_c) +_c ‖insert‖_p‖f‖_p‖x‖_p‖xs‖_p

     = (3 + ‖f‖_c + ‖x‖_c + ‖xs‖_c) +_c ‖insert‖_p‖f‖_p‖x‖_p‖xs‖_p

     = (3 + ‖f‖_c + ‖x‖_c + ‖xs‖_c)+_c rec(‖xs‖_p,
             Nil↦ ⟨1,Cons⟨x,Nil⟩⟩
             Cons↦ ⟨y,⟨ys,r⟩⟩.(3 + ((‖f‖_p‖x‖_p)_p y)_c)+_c rec((‖f‖_p‖x‖_p)_p y)_p,
                                                 True↦ ⟨1,Cons⟨‖x‖_p,Cons⟨y,ys⟩⟩⟩
                                                 False↦ ⟨1+r_c,Cons⟨y,r_p⟩⟩))
```

Figure 9: The translation of `insert f x xs`. Unlike before, we do not assume that `f`, `x`, `xs` are variables. They may be expressions with non-zero costs.

The result is:

```
‖insert f x xs‖ = (3 + ‖f‖_c + ‖x‖_c + ‖xs‖_c)
```

```
+_c rec(‖xs‖_p,
        Nil↦ ⟨1,Cons⟨x,Nil⟩⟩
        Cons↦ ⟨y,⟨ys,r⟩⟩.(3 + ((‖f‖_p‖x‖_p)_p  y)_c)+_c rec((‖f‖_p‖x‖_p)_p  y)_p,
                                                True↦ ⟨1,Cons⟨‖x‖_p,Cons⟨y,ys⟩⟩⟩
                                                False↦ ⟨1+r_c,Cons⟨y,r_p⟩⟩))
```

1.1.2 Interpretation

We well use an interpretation of lists as a pair of their greatest element and their length. Figure 10 formalizes this interpretation.

$$\begin{aligned}
[\![list]\!] &= \mathbb{Z} \times \mathbb{N}^{\infty} \\
D^{list} &= \{*\} + \{\mathbb{Z}\} \times \mathbb{N}^{\infty} \\
size_{list}(Nil) &= (-\infty, 0) \\
size_{list}(Cons(i,(j,n))) &= (max\{i,j\}, 1+n)
\end{aligned}$$

Figure 10: Interpretation of lists as lengths

We use the mutual ordering on pairs. That is, $(s,n) \leq (s',n')$ if $n \leq n'$ and $s < s'$ or $n < n'$ and $s \leq s'$.

First we interpret the `rec`, which drives of the cost of `insert`. As in the translation, we break the interpretation up to make it more manageable. We will write map, λ and $+_c$ in the semantics, which stand for the semantic equivalents of the syntactic `map`, λ and $+_c$. The definitions of these semantic functions mirror the definitions of their syntactic equivalents. Figures 11 and 12 walk through the interpretation.

$$\begin{aligned}
&[\![\texttt{rec((f x)}_p\texttt{ y)}_p\texttt{,}\\
&\quad\texttt{True}\mapsto \langle 1,\texttt{Cons}\langle\texttt{x,Cons}\langle\texttt{y,ys}\rangle\rangle\rangle\\
&\quad\texttt{False}\mapsto \langle 1+\texttt{r}_c,\texttt{Cons}\langle\texttt{y,r}_p\rangle\rangle\texttt{))}\ ]\!]\xi\{\texttt{f}\mapsto f,\texttt{x}\mapsto x,\texttt{y}\mapsto y,\texttt{ys}\mapsto (i,n),\texttt{r}\mapsto r\}
\end{aligned}$$

$$f_{True}(\langle\rangle) = [\![\langle\texttt{1,Cons}\langle\texttt{x,Cons}\langle\texttt{y,ys}\rangle\rangle\rangle]\!]\xi\{\texttt{f}\mapsto f,\texttt{x}\mapsto x,\texttt{y}\mapsto y,\texttt{ys}\mapsto (i,n),\texttt{r}\mapsto r\}$$

$$= (1,(max\{x,y,i\},2+n))$$

$$f_{False}(\langle\rangle) = [\![\langle 1+\texttt{r}_c,\texttt{Cons}\langle\texttt{y,r}_p\rangle\rangle\texttt{))}\ ]\!]\xi\{\texttt{f}\mapsto f,\texttt{x}\mapsto x,\texttt{y}\mapsto y,\texttt{ys}\mapsto (i,n),\texttt{r}\mapsto r\}$$

$$= (1+\pi_0 r,(max\{y,\pi_0\pi_1 r\},1+\pi_1\pi_1 r))$$

$$= \bigvee_{size(w)\leq\pi_1(\pi_1(f\ \ x)\ \ y)} case(w,(f_{True},f_{False}))$$

$$= \bigvee_{size(w)\leq\pi_1(\pi_1(f\ \ x)\ \ y)} case(w,(\lambda\langle\rangle.(1,(max\{x,y,i\},2+n)),\lambda\langle\rangle.(1+\pi_0 r,(max\{y,\pi_0\pi_1 r\},1+\pi_1\pi_1 r))$$

$$= (1,(max\{x,y,i\},2+n))\vee(1+\pi_0 r,(max\{y,\pi_0\pi_1 r\},1+\pi_1\pi_1 r))$$

Figure 11: Interpretation of the inner `rec` of `insert` with lists abstracted to sizes

$$\begin{aligned}
g(i,n) = [\![&\texttt{rec(xs,}\\
&\quad\texttt{Nil}\mapsto\langle\texttt{1,Cons}\langle\texttt{x,Nil}\rangle\rangle\\
&\quad\texttt{Cons}\mapsto\langle\texttt{y,}\langle\texttt{ys,r}\rangle\rangle\texttt{.}(3+(\texttt{(f x)}_p\texttt{ y)}_c)+_c\texttt{rec((f x)}_p\texttt{ y)}_p\texttt{,}\\
&\qquad\qquad\texttt{True}\mapsto\langle 1,\texttt{Cons}\langle\texttt{x,Cons}\langle\texttt{y,ys}\rangle\rangle\rangle\\
&\qquad\qquad\texttt{False}\mapsto\langle 1+\texttt{r}_c,\texttt{Cons}\langle\texttt{y,r}_p\rangle\rangle\texttt{)))}]\!]\xi\{\texttt{f}\mapsto f,\texttt{x}\mapsto x,\texttt{xs}\mapsto(i,n)\}
\end{aligned}$$

$$f_{Nil}(\langle\rangle) = [\![\langle\texttt{1,Cons}\langle\texttt{x,Nil}\rangle\rangle]\!]\xi\{\texttt{f}\mapsto f,\texttt{x}\mapsto x,\texttt{xs}\mapsto(i,n)\}$$

$$f_{Nil}(\langle\rangle) = (1,(x,1))$$

$$\begin{aligned}
f_{Cons}((j,(j,m))) = [\![&(3+\texttt{((f x)}_p\texttt{ y)}_c)+_c\texttt{rec(}\ldots\texttt{)}]\!]\xi\\
&\{\texttt{f}\mapsto f,\texttt{x}\mapsto x,\texttt{xs}\mapsto(i,n),\langle\texttt{y,}\langle\texttt{ys,r}\rangle\rangle\mapsto(map^{\mathbb{Z}\times\mathbb{N}^\infty}(\lambda a.(a,[\![\texttt{rec(}w,\ldots\texttt{)}]\!]\xi\{w\mapsto a\}),(j,(j,m))))\}
\end{aligned}$$

$$= [\![\ldots]\!]\xi\{\ldots\langle\texttt{y,}\langle\texttt{ys,r}\rangle\rangle\mapsto(j,map^{\mathbb{N}^\infty}(\lambda a.(a,[\![\texttt{rec(}w,\ldots\texttt{)}]\!]\xi\{w\mapsto a\}),(j,m)))\}$$

$$= [\![\ldots]\!]\xi\{\ldots\langle\texttt{y,}\langle\texttt{ys,r}\rangle\rangle\mapsto(j,((j,m),[\![\texttt{rec(}w,\ldots\texttt{)}]\!]\xi\{w\mapsto(j,m)\}))\}$$

$$= [\![\ldots]\!]\xi\{\ldots\langle\texttt{y,}\langle\texttt{ys,r}\rangle\rangle\mapsto(j,((j,m),g(j,m)))\}$$

$$\begin{aligned}
= (3+\pi_0(\pi_1(f\ \ x)\ \ j))+_c&\\
&((1,(max\{x,j\},2+m)))\vee(1+\pi_0 g(j,m),(max\{j,\pi_0\pi_1 g(j,m)\},1+\pi_1\pi_1 g(j,m)))
\end{aligned}$$

$$\begin{aligned}
= (3+\pi_0(\pi_1(f\ \ x)\ \ j))+_c&\\
&(1\vee(1+\pi_0 g(j,m)),(max\{x,j,\pi_0\pi_1 g(j,m)\},2+m\vee 1+\pi_1\pi_1 g(j,m)))
\end{aligned}$$

$$= (4+\pi_0(\pi_1(f\ \ x)\ \ j)+\pi_0 g(j,m),(max\{x,j\pi_0\pi_1 g(j,m)\},2+m\vee 1+\pi_1\pi_1 g(j,m)))$$

$$g(i,n) = \bigvee_{size(z)\leq(i,n)} case(z,(f_{Nil},f_{Cons}))$$

Figure 12: Interpretation of `rec` in `insert`.

The initial result is given in equation 1.

$$\begin{aligned}
&f_{Nil}(\langle\rangle) = (1, (x, 1))\\
&f_{Cons}(j, (j, m)) = (4 + \pi_0(\pi_1(f\ x)\ j) + \pi_0 g(j, m),\\
&\quad (max\{x, j, \pi_0\pi_1 g(j, m)\}, 2 + m \vee 1 + \pi_1\pi_1 g(j, m)))\\
&g(i, n) = \bigvee_{size(z)\leq(i,n)} case(z, (f_{Nil}, f_{Cons})) \qquad (1)
\end{aligned}$$

This recurrence is difficult to work with. Specifically, we cannot apply traditional methods of solving it. We will manipulate it into a more usable form by eliminating the arbitrary maximum. We will separate the recurrence into a recurrence for the cost and a recurrence for the potential, and solve those independently.

Lemma 1.1. $g_c(i, n) \leq (4 + ((f\ x)_p\ i)_c n + 1$

Proof. We prove this by induction on n. Recall we use the mutual ordering on pairs.

case $n = 0$

$g_c(i, n) = (1, (x, 1))_c = 1$

case$n > 0$

$$\begin{aligned}
&= \bigvee_{size(z)\leq(i,n)} case(z, (f_{Nil}, f_{Cons}))\\
&= \bigvee_{j<i,m\leq n \text{ or } j\leq i,m<n} case((j, m), (f_{Nil}, f_{Cons}))\\
&= \bigvee_{j<i,m\leq n \text{ or } j\leq i,m<n} 4 + ((f\ x)_p\ j)_c + g_c(j, m')) \qquad \text{where } m' = m - 1\\
&= \bigvee_{j<i,m\leq n \text{ or } j\leq i,m<n} 4 + ((f\ x)_p\ j)_c + (4 + ((f\ x)_p\ j)_c)m' + 1 \quad \text{by the induction hypothesis}\\
&= \bigvee_{j<i,m\leq n \text{ or } j\leq i,m<n} (4 + ((f\ x)_p\ j)_c)(m' + 1) + 1\\
&= \bigvee_{j<i,m\leq n \text{ or } j\leq i,m<n} (4 + ((f\ x)_p\ j)_c)m + 1\\
&\leq \bigvee_{i<j,m\leq n \text{ or } i\leq j,m<n} (4 + ((f\ x)_p\ i)_c)n + 1\\
&\leq (4 + ((f\ x)_p\ i)_c)n + 1
\end{aligned}$$

$\square$

As expected, we find the cost of insert is bounded by the length of the list and the largest element.

Lemma 1.2. $g_p(i,n) \leq (max\{x,i\}, n+1)$

Proof. We prove this by induction on n.

case $n = 0$

$g_p(i,n) = (1,(x,1))_p = (x,1)$.

case $n > 0$

$$\begin{aligned}
&= \bigvee_{size(z)\leq(i,n)} case(z,(f_{Nil}, f_{Cons}) \\
&= \bigvee_{j<i,m\leq n \text{ or } j\leq i,m<n} (max\{x,j,\pi_0\pi_1 g(j,m')\}, 2+m' \vee 1+\pi_1\pi_1 g(j,m')) && \text{where } m' = m-1 \\
&\leq \bigvee_{j<i,m\leq n \text{ or } j\leq i,m<n} (max\{x,j\}, 2+m') && \text{by the induction hypoth} \\
&\leq \bigvee_{j<i,m\leq n \text{ or } j\leq i,m<n} (max\{x,i\}, 1+n) \\
&\leq (max\{x,i\}, 1+n)
\end{aligned}$$

$\square$

We find the length of the potential is bounded by one plus the length of the input, and the largest element in the output is bounded by the maximum of the element being inserted and the largest element in the input. This is somewhat unsatisfactory, since we would expect the relationship to be equality. What happens if we try to prove the equality?

Lemma 1.3. $g_p(i,n) = (max\{x,i\}, n+1)$

Proof. We attempt to prove this by induction on n. The first steps proceed similarly to 1.2.

case $n = 0$

$g_p(i,n) = (1,(x,1))_p = (x,1)$.

case $n > 0$

$$
\begin{aligned}
&= \bigvee_{size(z) \leq (i,n)} case(z, (f_{Nil}, f_{Cons}) \\
&= \bigvee_{j<i, m \leq n \text{ or } j \leq i, m<n} (max\{x, j, \pi_0\pi_1 g(j, m')\}, 2 + m' \vee 1 + \pi_1\pi_1 g(j, m')) && \text{where } m' = m - 1 \\
&= \bigvee_{j<i, m \leq n \text{ or } j \leq i, m<n} (max\{x, j\}, 2 + m') && \text{by the induction hypoth} \\
&= \bigvee_{j<i, m \leq n} (max\{x, j\}, 1 + m) \vee \bigvee_{j \leq i, m<n} (max\{x, j\}, 1 + m)
\end{aligned}
$$

□

We see that we get stuck. Because of the mutual ordering on pairs, our big maximum is over all z such that $size(z) < (i, n)$. This includes (j, m) such that $j < i^m \leq n$. We have no way of reasoning about the potential of $g(i-1, n) \vee g(i, n-1)$. So we cannot prove equality for 1.2. This indicates we may not have the optimal ordering on pairs.

Using lemmas 1.1 and 1.2, we can express the cost and potential of `insert` in terms of its arguments.

$$insert\ f\ x\ xs \leq (4 + ((f\ x)_p\ i)_c n + 1, (max\{x, i\}, n + 1)) \tag{2}$$

1.2 Sort

1.2.1 Translation

The translation of sort is shown in figure 15. The translation of the `Nil` and `Cons` branches in the `rec` are walked through in figures 13 and 14, respectively. The translation of `sort` applied to its arguments is given in figure 16.

$$\|\texttt{Nil}\mapsto\texttt{Nil}\|$$
$$=\texttt{Nil}\mapsto 1 +_c \|\texttt{Nil}\|$$
$$=\texttt{Nil}\mapsto 1 +_c \langle\texttt{0,Nil}\rangle$$
$$=\texttt{Nil}\mapsto \langle\texttt{1,Nil}\rangle$$

Figure 13: Translation of `Nil` branch of `sort`.

$$\|\texttt{Cons}\mapsto \langle\texttt{y},\langle\texttt{ys},\texttt{r}\rangle\rangle\texttt{.insert f y force(r)}$$
$$=\texttt{Cons}\mapsto 1 +_c \|\texttt{insert f y force(r)}\|$$
$$=\texttt{Cons}\mapsto \langle\texttt{y},\langle\texttt{ys},\texttt{r}\rangle\rangle.1 +_c (\|\texttt{force(r)}\|_c) +_c \|\texttt{insert f y}\|_p\|\texttt{force(r)}\|_p$$
$$=\texttt{Cons}\mapsto \langle\texttt{y},\langle\texttt{ys},\texttt{r}\rangle\rangle.1 +_c ((\|\texttt{r}\|_c +_c \|\texttt{r}\|_p)_c) +_c \|\texttt{insert f y}\|_p(\|\texttt{r}\|_c +_c \|\texttt{r}\|_p)_p$$
$$=\texttt{Cons}\mapsto \langle\texttt{y},\langle\texttt{ys},\texttt{r}\rangle\rangle.1 +_c \texttt{r}_c +_c \|\texttt{insert f y}\|_p\texttt{r}_p$$
$$=\texttt{Cons}\mapsto \langle\texttt{y},\langle\texttt{ys},\texttt{r}\rangle\rangle.1 +_c \texttt{r}_c +_c 3 +_c \|\texttt{insert}\|_p \texttt{ f y r}_p$$
$$=\texttt{Cons}\mapsto \langle\texttt{y},\langle\texttt{ys},\texttt{r}\rangle\rangle.(4 + \texttt{r}_c) +_c \|\texttt{insert}\|_p \texttt{ f y r}_p$$

Figure 14: Translation of `Cons` branch of `sort`.

$$\begin{aligned}\|\texttt{sort}\| &= \langle 0, \lambda\texttt{f}.\langle 0, \lambda\texttt{xs}.\|\texttt{rec(xs, Nil}\mapsto\texttt{ Nil,}\\ &\qquad\qquad \texttt{Cons}\mapsto \langle\texttt{y},\langle\texttt{ys},\texttt{r}\rangle\rangle\texttt{.insert f y force(r))}\|\rangle\rangle\\ &= \langle 0, \lambda\texttt{f}.\langle 0, \lambda\texttt{xs.rec(xs, Nil}\mapsto \langle\texttt{1,Nil}\rangle\\ &\qquad\qquad \texttt{Cons}\mapsto \langle\texttt{y},\langle\texttt{ys},\texttt{r}\rangle\rangle.4 +_c \texttt{r}_c +_c \|\texttt{insert}\|_p \texttt{ f y r}_p\end{aligned}$$

Figure 15: Translation of `sort`

$$\begin{aligned}\|\texttt{sort f xs}\| &= (1 + \|\texttt{sort f}\|_c + \|\texttt{xs}\|_c) +_c \|\texttt{sort f}\|_p\|\texttt{xs}\|_p\\ &= (1 + (1 + \|\texttt{sort}\|_c + \|\texttt{f}\|_c + \|\texttt{xs}\|_c)) +_c \|\texttt{sort}\|_p\|\texttt{f}\|_p\|\texttt{xs}\|_p\\ &= (1 + (1 + 0 + 0 + 0)) +_c \|\texttt{sort}\|_p\|\texttt{f}\|_p\|\texttt{xs}\|_p\\ &= 2 +_c \|\texttt{sort}\|_p\texttt{f}\|_p\|\texttt{xs}\|_p\\ &= 2 +_c \texttt{rec(}\|\texttt{xs}\|_p\texttt{, Nil}\mapsto \langle\texttt{1,Nil}\rangle\\ &\qquad\qquad \texttt{Cons}\mapsto \langle\texttt{y},\langle\texttt{ys},\texttt{r}\rangle\rangle.(4 + \texttt{r}_c) +_c \|\texttt{insert}\|_p \texttt{ f y r}_p\end{aligned}$$

Figure 16: Translation of `sort` applied to variables `f` and `xs`

1.2.2 Interpretation

The `rec` construct again drives the cost and potential of `sort`. The walk through of the interpretation of the `rec` is given in figure 17.

$$g(i,n) = \llbracket \texttt{rec(}\|\texttt{xs}\|_p\texttt{, Nil}\mapsto \langle \texttt{1,Nil}\rangle \; \texttt{Cons}\mapsto \langle \texttt{y},\langle \texttt{ys},\texttt{r}\rangle\rangle.(4+\texttt{r}_c) +_c \|\texttt{insert}\|_p \texttt{ f y r}_p\texttt{)}\rrbracket \xi\{xs \mapsto n\}$$

$$= \llbracket \texttt{rec(}\|\texttt{xs}\|_p\texttt{, Nil}\mapsto \langle \texttt{1,Nil}\rangle \; \texttt{Cons}\mapsto \langle \texttt{y},\langle \texttt{ys},\texttt{r}\rangle\rangle.(4+\texttt{r}_c) +_c \|\texttt{insert}\|_p \texttt{ f y r}_p\texttt{)}\rrbracket \xi\{xs \mapsto n\}$$

$$= \llbracket \texttt{rec(}\|\texttt{xs}\|_p\texttt{, Nil}\mapsto \langle \texttt{1,Nil}\rangle \; \texttt{Cons}\mapsto \langle \texttt{y},\langle \texttt{ys},\texttt{r}\rangle\rangle.(4+\texttt{r}_c) +_c \|\texttt{insert}\|_p \texttt{ f y r}_p\texttt{)}\rrbracket \xi\{xs \mapsto n\}$$

$$= \bigvee_{size(z)\leq n} case(z, (f_{Nil}, f_{Cons}))$$

$$f_{Nil}(\langle\rangle) = \llbracket \langle \texttt{1,Nil}\rangle \rrbracket \xi$$
$$= f_{Nil}(\langle\rangle) = (1, (\neg\infty, 0))$$

$$f_{Cons}((j,m)) = \llbracket \ldots \rrbracket \xi\{\langle \texttt{y},\langle \texttt{ys},\texttt{r}\rangle\rangle \mapsto map^{j\times \mathbb{N}^\infty}(\lambda a.(a, \llbracket \texttt{rec(}w,\ldots\texttt{)}\rrbracket \xi\{w\mapsto a\}), (j,m))\}$$

$$= \llbracket \ldots \rrbracket \xi\{\langle \texttt{y},\langle \texttt{ys},\texttt{r}\rangle\rangle \mapsto (map^{int}(\lambda a.(\ldots), j), map^{\mathbb{N}^\infty}(\lambda a.(a, \llbracket \texttt{rec(}w,\ldots\texttt{)}\rrbracket \xi\{w\mapsto a\}), m))\}$$

$$= \llbracket \ldots \rrbracket \xi\{\langle \texttt{y},\langle \texttt{ys},\texttt{r}\rangle\rangle \mapsto (j, (m, \llbracket \texttt{rec(}w,\ldots\texttt{)}\rrbracket \xi\{w\mapsto m\}))\}$$

$$= \llbracket (4+\texttt{r}_c) +_c \|\texttt{insert}\|_p \texttt{ f y r}_p \rrbracket \xi\{\langle \texttt{y},\langle \texttt{ys},\texttt{r}\rangle\rangle \mapsto (j, (m, g(j,m))\}$$

$$= (4+\pi_0 g(j,m)) +_c insert\ f\ j\ \pi_1 g(j,m)$$

$$g(i,n) = \bigvee_{size(z)\leq n} case(z, (\lambda(\langle\rangle).(1,(\neg\infty,0)), \lambda(j,m).(4+\pi_0 g(j,m)) +_c insert\ f\ j\ \pi_1 g(j,m)))$$

Figure 17: Interpretation of `rec` in `sort`.TO DO FIX THIS

Equation 3 shows the initial recurrence extracted.

$$g(i,n) = \bigvee_{size(z)\leq(i,n)} case(z, (\lambda(\langle\rangle).(1,(\neg\infty,0), \lambda(j,m).4+\pi_0 g(j,m))+_c(insert\ f\ j\ \pi_1 g(j,m))) \tag{3}$$

Observe that in equation 3, the cost is depends on the potential of the recursive call. Therefore we must solve the recurrence for the potential first.
Lemma 1.4. $\pi_1 g(n) \leq (j,n)$

Proof. We prove this by induction on n. We use equation 2 to determine the potential of the *insert* function.

case $n = 0$ $\pi_1 g(i,n) = (i,0)$

case $n > 0$

$$\begin{aligned}
\pi_1 g(i,n) &= \pi_1 \bigvee_{size(z)\leq n} case(z, (\lambda(\langle\rangle).(1, (\neg\infty, 0)), \lambda(j,m).4 + \pi_0 g(j,m)) +_c (insert\ f\ j\ \pi_1 g(j,m)) \\
&= \bigvee_{j\leq i, m<n \text{ or } j<i, m\leq n} \pi_1(insert\ f\ j\ \pi_1 g(j',m')) \qquad j' \leq j, m' = m-1 \\
&\leq \bigvee_{j\leq i, m<n \text{ or } j<i, m\leq n} \pi_1(insert\ f\ j\ (j',m')) \\
&\leq \bigvee_{j\leq i, m<n \text{ or } j<i, m\leq n} (max\{j,j'\}, m'+1 \\
&\leq \bigvee_{j\leq i, m<n \text{ or } j<i, m\leq n} (j,m) \\
&\leq \bigvee_{j\leq i, m<n \text{ or } j<i, m\leq n} (i,n) \\
&\leq (i,n)
\end{aligned}$$

□

As in the interpretation of `insert` we are left with a less than satisfactory bound on the potential of `sort`. It would grievous mistake to write a sorting function whose output was smaller than its input. Under the current interpretation of lists, this would mean either the length of the list decreased or the size of the largest element in the list decreased. Unfortunately we are stuck with an upper bound on the size of the output because or interpretation of `insert` only provides an upper bound on the potential of its output.

We may solve the recurrence for the cost of `sort`.
Lemma 1.5. $\pi_0 g(n) \leq (4 + \pi_0(\pi_1(f\ x)\ i)n^2 + 5n + 1$

Proof. We prove this by induction on n.

case $n = 0$ $\pi_0 g(i,n) = 1$

case $n > 0$

$$
\begin{aligned}
\pi_0 g(i,n) &= \pi_0 \bigvee_{size(z)\le(i,n)} case(z, (\lambda(\langle\rangle).(1,(\neg\infty,0)), \lambda(j,m).4+\pi_0 g(j,m)) +_c (insert\ f\ j\ \pi_1 g(j,m \\
&= \bigvee_{j<i,m\le n \text{ or } j\le i,m<n} 4+\pi_0 g(j,m-1)+\pi_0(insert\ f\ j\ \pi_1 g(j,m-1)) \\
&\le \bigvee_{j<i,m\le n \text{ or } j\le i,m<n} 4+\pi_0 g(j,m-1)+\pi_0(insert\ f\ j\ (j,m-1)) \\
&\le \bigvee_{j<i,m\le n \text{ or } j\le i,m<n} 4+\pi_0 g(j,m-1)+(4+\pi_0(\pi_1(f\ j)\ j))(m-1)+1 \\
&\text{let } c_1 = (4+\pi_0(\pi_1(f\ j)\ j)) \\
&\le \bigvee_{j<i,m\le n \text{ or } j\le i,m<n} 4+c_1(m-1)^2+5(m-1)+1+c_1(m-1)+1 \\
&\le \bigvee_{j<i,m\le n \text{ or } j\le i,m<n} 4+c_1m^2-2c_1m+c_1+5m-5+1+c_1m-c_1+1 \\
&\le \bigvee_{j<i,m\le n \text{ or } j\le i,m<n} c_1m^2-c_1m+5m+1 \\
&\le \bigvee_{j<i,m\le n \text{ or } j\le i,m<n} (4+\pi_0(\pi_1(f\ i)\ i))n^2+5n+1 \\
&\le (4+\pi_0(\pi_1(f\ i)\ i))n^2+5n+1
\end{aligned}
$$

□

As expected the cost of `sort` is $\mathcal{O}(n^2)$ where n is the length of the list. It is clear from the analysis how the cost of the comparison function determines the running time of `sort`. We can see that the comparison function is called order n^2 times.