J. Functional Programming **2** (3): 375–385, July 1992 © 1992 Cambridge University Press

FUNCTIONAL PEARLS

Unravelling greedy algorithms

RICHARD S. BIRD
Programming Research Group, Oxford University, UK

1 Introduction

In my previous *Functional Pearls* article (Bird, 1992), I proved a theorem giving conditions under which an optimization problem could be implemented by a greedy algorithm. A greedy algorithm is one that picks a 'best' element at each stage. Here, we return to this theorem and extend it in various ways. We then use the theory to solve an intriguing problem about unravelling sequences into a smallest number of ascending subsequences.

2 The greedy theorem

We begin by restating the theorem of Bird (1992). Suppose $F \in A \rightarrow \{[B]\}^+$, so that for each $a \in A$ the value Fa is a non-empty set of sequences over B. Suppose $f = \sqcap_C / \cdot F$, where $C \in [B] \rightarrow \mathrm{N}$. By definition, the binary operator $\sqcap_C$ returns the smaller of its arguments under some unspecified total ordering $\leqslant_C$ which respects C, i.e. $x \leqslant_C y$ implies $Cx \leqslant Cy$. Thus, fa is specified as some C-minimizing sequence in Fa. The theorem is that, under the conditions on C and F cited below, we can find an ordering $\leqslant_C$ for which f can be computed by a greedy algorithm

$$fa = \begin{cases} [] & \text{if } pa \\ [b] \mathbin{+\!\!+} f(a \ominus b) & \text{otherwise} \\ \text{where } b = \sqcap_B / Ha. \end{cases}$$

The conditions on C and F are as follows. First of all, C is assumed to be a *cost* function, meaning that C satisfies the two conditions

$$\begin{aligned} Cx = 0 &\equiv x = [] \\ Cx \leqslant Cy &\equiv C(u \mathbin{+\!\!+} x) \leqslant C(u \mathbin{+\!\!+} y) \end{aligned}$$

for all u, x and y. In particular, the length function $\#$ is a cost function, as is any function of the form $+/ \cdot w*$ provided w returns non-negative numbers.

Second, F admits a *decomposition* $(p, H, \ominus)$ in the sense that, for all a,

$$\begin{aligned} [] \in Fa &\equiv pa \\ ([b] \mathbin{+\!\!+} x) \in Fa &\equiv b \in Ha \wedge x \in F(a \ominus b). \end{aligned}$$

Equivalently, F is assumed to satisfy the equation

$$Fa = \{[] \mid pa\} \cup \{[b] \mathbin{+\!\!+} x \mid b \in Ha \wedge x \in F(a \ominus b)\}.$$

Third, F and C satisfy a certain *greedy* condition. The condition is that there exists an ordering $\leqslant_B$ on B with the property that, for all a with $\neg p\,a$, if

$$([b] +\!\!+ x) \in F\,a \wedge b \neq \sqcap_B/H\,a,$$

then there exists a c and y such that

$$([c] +\!\!+ y) \in F\,a \wedge c <_B b \wedge C([c] +\!\!+ y) \leqslant C([b] +\!\!+ x).$$

These are the conditions under which the greedy theorem holds. The proof of the theorem is in (Bird (1992), and we will not repeat it here.

There are a number of simple variations to the greedy theorem. For instance, we could phrase it as a maximization rather than minimization problem. Also, the first greedy condition, assigning minimum cost to the empty sequence, is not an important one. If we assigned maximum cost to the empty sequence, the result would be that f could be computed by the scheme

$$f\,a = \begin{cases} [b] +\!\!+ f(a \ominus b) & \text{if } H\,a \neq \{\} \\ [] & \text{otherwise} \\ \text{where } b = \sqcap_B/H\,a. \end{cases}$$

3 Right-reductions

Let us now look at two particular expressions for F for which a suitable decomposition exists. Both expressions involve right-reductions.

For an operator $\oplus \in B \times A \to A$ and $e \in A$, the right-reduction $\oplus \not\leftarrow e \in [B] \to A$ is defined informally by

$$(\oplus \not\leftarrow e)[b_1, b_2, \ldots, b_n] = b_1 \oplus (b_2 \oplus \ldots \oplus (b_n \oplus e)).$$

Thus, $\oplus \not\leftarrow e = \mathsf{foldr}\,(\oplus)\,e$ in functional programming. A number of computational problems can be specified as asking for some cost-minimizing sequence in the inverse image of a right-reduction:

$$f = \sqcap_C/ \cdot \mathrm{Inv}\,(\oplus \not\leftarrow e). \tag{1}$$

Here, $\mathrm{Inv}\,(f)$ is the *inverse image* of f, defined for $f \in X \to Y$ by

$$\mathrm{Inv}\,(f)\,y = \{x \in X \mid f\,x = y\}.$$

If f is surjective, then $\mathrm{Inv}\,(f)\,y$ is non-empty for all y. Setting $F = \mathrm{Inv}\,(\oplus \not\leftarrow e)$, and assuming $\oplus \not\leftarrow e$ is surjective, we have $F \in A \to \{[B]\}^+$. Furthermore,

$$\begin{aligned} & [] \in F\,a \\ \equiv\;& \{\text{definition of } F\} \\ & a = (\oplus \not\leftarrow e)\,[] \\ \equiv\;& \{\text{definition of } \not\leftarrow\} \\ & a = e \end{aligned}$$

and
$$
\begin{aligned}
&([b] \mathbin{+\!\!+} x) \in F a\\
\equiv\;& \{\text{definition } F\}\\
&a = (\oplus \not\leftarrow e)([b] \mathbin{+\!\!+} x)\\
\equiv\;& \{\text{definition of } \not\leftarrow\}\\
&(\exists c \in A : a = b \oplus c \wedge c = (\oplus \not\leftarrow e)\, x)\\
\equiv\;& \{\text{definition of } F\}\\
&(\exists c \in A : a = b \oplus c \wedge x \in F c).
\end{aligned}
$$

So we have

$$F a = \{[\,] \mid a = e\} \cup \{[b] \mathbin{+\!\!+} x \mid \exists c \in A : b \oplus c = a \wedge x \in F c\}.$$

This decomposition for F is almost, but not quite, what is required by the greedy theorem. To pass to that condition we need a property of $\oplus$, namely that there exist H and $\ominus$ so that

$$b \oplus c = a \equiv b \in H a \wedge c = a \ominus b.$$

The greedy theorem therefore requires that $\oplus$ be invertible, where $\oplus$ is *invertible* if there is a $\ominus$ such that

$$(b \oplus c) \ominus b = c.$$

Many operators are invertible; for example, each of $+$, $\mathbin{+\!\!+}$, $\uplus$ (bag union), and $\wedge\!\!\wedge$ (merge) are. However, many operators are not; for example, $\cup$ (set union), $\sqcup$ (maximum), and $\sqcap$ (minimum) are not invertible. If $\oplus$ is not assumed to be invertible, then at best we can find functions H and T so that

$$b \oplus c = a \equiv b \in H a \wedge c \in T(a, b).$$

The functions H and T can be defined in terms of $\operatorname{Inv}(\oplus) \in A \to \{B \times A\}$:

$$
\begin{aligned}
H a &= \pi_1 * \operatorname{Inv}(\oplus)\, a\\
T(a,b) &= \pi_2 * ((b =) \cdot \pi_1) \lhd \operatorname{Inv}(\oplus)\, a,
\end{aligned}
$$

where π_1, π_2 are the first and second projection functions on pairs. We then have

$$F a = \{[\,] \mid a = e\} \cup \{[b] \mathbin{+\!\!+} x \mid b \in H a \wedge x \in \cup / F * T(a,b)\}. \qquad (2)$$

With this decomposition for F, and with the other conditions of the greedy theorem, we get what we can call a semi-greedy theorem: there exists an ordering $\leqslant_C$ such that f can be computed by the scheme

$$
f a = \begin{cases}
[\,] & \text{if } a = e\\
[b] \mathbin{+\!\!+} \sqcap_C / f * T(a,b) & \text{otherwise}\\
\text{where } b = \sqcap_B / H a. &
\end{cases}
$$

The proof is an easy modification of the original one and will not be given.

The appearance of the above scheme for f can be improved. First, we construct an ordering $\leqslant_A$ so that

$$\sqcap_C / f * S = f(\sqcap_A / S) \tag{3}$$

for all non-empty sets $S \in \{A\}$. To define $\leqslant_A$, let $\leqslant_?$ be any ordering (which we assume to exist) on A. Define

$$a \leqslant_A b \equiv fa <_C fb \vee (fa = fb \wedge a \leqslant_? b).$$

Equation (3) follows, since $a \leqslant_A b$ implies $fa \leqslant_C fb$. Next, define yet another ordering:

$$(b1, c1) \leqslant_{BA} (b2, c2) \equiv b1 <_B b2 \vee (b1 = b2 \wedge c1 \leqslant_A c2).$$

Here, $\leqslant_B$ is the ordering given in the statement of the greedy theorem. The ordering $\leqslant_{BA}$ is just the lexicographic ordering on $B \times A$, using $\leqslant_B$ and $\leqslant_A$ as suborderings. The upshot of all this is that f can be computed by the scheme

$$fa = \begin{cases} [] & \text{if } a = c \\ [b] \mathbin{+\!\!+} fc & \text{otherwise} \\ \text{where } (b, c) = \sqcap_{BA} / \mathrm{Inv}(\oplus)\, a. \end{cases}$$

The proof is immediate from the fact that $(b, c) = \sqcap_{BA} / \mathrm{Inv}(\oplus)\, a$ if and only if $b = \sqcap_B / Ha$ and $c = \sqcap_A / T(a, b)$.

The revised scheme for f does not increase the efficiency of the algorithm. The given construction of $\leqslant_A$ means that evaluations of $\sqcap_{BA}$ involve additional computations of f. However, if we find some simpler definition of $\leqslant_A$ to satisfy (3), the new scheme will involve less work.

Finally, we mention an easy generalization of (1). Consider

$$f = \sqcap_C / \cdot \mathsf{all}\, p \triangleleft \cdot \mathrm{Inv}(\oplus \not\leftarrow e). \tag{4}$$

The difference with (1) is the presence of a filter operation which selects only those members of $\mathrm{Inv}(\oplus \not\leftarrow e)$ satisfying the predicate $\mathsf{all}\, p$. By definition, $\mathsf{all}\, p\, x$ holds just in the case that p holds for every element of x. An easy modification to the above theory establishes, again under the cost and greedy conditions on C and F, where $F = \mathsf{all}\, p \triangleleft \cdot \mathrm{Inv}(\oplus \not\leftarrow e)$, that f can be computed by the scheme

$$fa = \begin{cases} [] & \text{if } a = e \\ [b] \mathbin{+\!\!+} fc & \text{otherwise} \\ \text{where } (b, c) = \sqcap_{BA} / (p \cdot \pi_1) \triangleleft \mathrm{Inv}(\oplus)\, a. \end{cases}$$

4 Converse

There is a second form for F that posses a similar decomposition. The form is $F = \mu(\otimes \not\leftarrow E)$, where for $g \in X \to \{Y\}$ the converse $\mu G \in Y \to \{X\}$ is defined by

$$\mu G y = \{x \in X \mid y \in G x\}.$$

The function μG corresponds to the relational converse of G when G is interpreted as a relation R (so that xRy just when $y \in Gx$). Problems whose specifications involve relational converse arise in a number of areas, for example in language recognition and pattern matching. Another example appears below.

If $\mu(\otimes \not\leftarrow E)$ is to have type $A \to \{[B]\}^+$ for some A and B, we need at least $E \in \{A\}$ and $\otimes \in B \times \{A\} \to \{A\}$. It is sometimes the case that $\otimes$ can be defined in terms of another operator $\oplus \in B \times A \to \{A\}$ by the equation

$$b \otimes x = \cup / (b \oplus) * x.$$

If such an equation does hold, then we write $\otimes = \oplus^\circ$. A simple example is provided by the function subs, which returns the set of subsequences of a sequence. We have subs $= \oplus^\circ \not\leftarrow \{[\,]\}$, where $a \oplus x = \{[a] +\!\!+ x, x\}$. The function μsubs returns the set of *super* sequences of a sequence.

Example

Here is another example. Given sequences x and y, consider all the ways x and y can be shuffled together to give a single sequence. The set of shuffles $x \asymp y$ of x and y is defined by taking $x \asymp [\,] = [\,] \asymp x = \{x\}$ and

$$([a] +\!\!+ x) \asymp ([b] +\!\!+ y) = ([a] +\!\!+) * (x \asymp ([b] +\!\!+ y)) \cup ([b] +\!\!+ * (([a] +\!\!+ x) \asymp y).$$

Thus $\asymp$ has type $[A] \times [A] \to \{[A]\}$ for some A. The operator $\asymp^\circ$ has type $[A] \times [\{A]\} \to \{[A]\}$. The value $x \asymp^\circ xs$ shuffles x with sequences in xs in all possible ways. The function $R = \asymp^\circ \not\leftarrow \{[\,]\}$ takes a list of sequences (or, since the order in which the sequences are processed is not important, a *bag* of sequences) and shuffles them together in all possible ways. For example,

$$\begin{aligned} & R[[1],[2,3],[4]] \\ &= [1] \asymp^\circ ([2,3] \asymp^\circ ([4] \asymp^\circ \{[\,]\})) \\ &= [1] \asymp^\circ ([2,3] \asymp^\circ \{[4]\}) \\ &= [1] \asymp^\circ \{[4,2,3],[2,4,3],[2,3,4]\} \\ &= \{[1,4,2,3],[4,1,2,3],[4,2,1,3],[4,2,3,1],\ldots\}. \end{aligned}$$

The result is the set of twelve permutations of $\{1,2,3,4\}$ in which 2 precedes 3. Replacing the argument $[[1],[2,3],[4]]$ with the bag $\wr[1],[2,3],[4]\int$ gives exactly the same result. A right-reduction $\otimes \not\leftarrow e$ is defined on bags if $a \otimes (b \otimes c) = b \otimes (a \otimes c)$ for all a, b and c. We omit the proof that $\asymp^\circ$ has this property.

Finally, $\mu R x$ returns the set of lists (or bags) of sequences that when shuffled together can give back x. For example, supposing we exclude the empty list from each bag, the value of $\mu R[1,2,3]$ is the set of five bags

$$\{\wr[1,2,3]\int, \wr[1,2],[3]\int, \wr[1,3],[2]\int, \wr[1],[2,3]\int, \wr[1],[2],[3]\int\}.$$

To return to the general situation, let $F = \mu(\oplus^\circ \not\leftarrow E)$. We have

$$\begin{aligned} & [\,] \in F a \\ \equiv\ & \{\text{definition of } \mu\} \\ & a \in (\oplus^\circ \not\leftarrow E)[\,] \\ \equiv\ & \{\text{definition of } \not\leftarrow\} \\ & a \in E \end{aligned}$$

and
$$
\begin{aligned}
& [b] \mathbin{+\!\!+} x \in F\,a \\
\equiv\; & \{\text{definition of } F \text{ and } \mu\} \\
& a \in (\oplus^{\circ} \not\leftarrow E)([b] \mathbin{+\!\!+} x) \\
\equiv\; & \{\text{definition of } \not\leftarrow\} \\
& a \in b \oplus^{\circ} (\oplus^{\circ} \not\leftarrow E)\,x \\
\equiv\; & \{\text{definition of } \oplus^{\circ}\} \\
& a \in \cup/(b\oplus) * (\oplus^{\circ} \not\leftarrow E)\,x \\
\equiv\; & \{\text{set theory}\} \\
& (\exists c \in A : a \in b \oplus c \wedge c \in (\oplus^{\circ} \not\leftarrow E)\,x) \\
\equiv\; & \{\text{definition of } \mu \text{ and } F\} \\
& (\exists c \in A : (b, c) \in \mu(\oplus)\,a \wedge x \in F\,c) \\
\equiv\; & \{\text{introducing } H \text{ and } T\text{; see below}\} \\
& (\exists c \in A : b \in H\,a \wedge c \in T(a, b) \wedge x \in F\,c) \\
\equiv\; & \{\text{set theory}\} \\
& b \in H\,a \wedge x \in \cup/F * T(a, b).
\end{aligned}
$$

The definitions of H and T are

$$
\begin{aligned}
H\,a &= \pi_1 * \mu(\oplus)\,a, \\
T(a, b) &= \pi_2 * (b = \cdot\pi_2) \triangleleft \mu(\oplus)\,a.
\end{aligned}
$$

Hence, we again get decomposition (2). We summarize the above calculations as a theorem.

Theorem 1
Let $f = \sqcap_C/\cdot\mu(\oplus^{\circ} \not\leftarrow E)$. Suppose that C is a cost function and the greedy condition holds for C and $F = \mu(\oplus^{\circ} \not\leftarrow E)$, where we suppose also that F returns non-empty sets. Then we can find orderings $\leqslant_C$ and $\leqslant_{BA}$ for which f can be computed by the scheme

$$
f\,a = \begin{cases} [\,] & \text{if } a \in E \\ [b] \mathbin{+\!\!+} f\,c & \text{otherwise} \\ \text{where } (b, c) = \sqcap_{BA}/\mu(\oplus)\,a. \end{cases}
$$

5 Bags

The greedy theorem is stated in terms of a function $F \in A \to \{[B]\}^+$ but does not critically depend on F returning sets of sequences. Here, we recast the theorem in terms of bags, that is, we now suppose that $F \in A \to \{\wr B \int\}^+$. Except for the greedy condition, there is no real problem: we replace $[\,]$ by $\wr\int$, $[b]$ by $\wr b \int$, and $\mathbin{+\!\!+}$ by $\uplus$ (bag union) throughout the statements of the cost and decomposition conditions. In particular, the modified definition of a cost function gives that #, the size of a bag, is a cost function. Since the order of the elements of a bag is not determined, the second clause of the decomposition condition, namely

$$(\llbracket b \rrbracket \uplus x) \in F a \equiv b \in H a \wedge x \in F(a \ominus b),$$

has the consequence that $x \in F a$ implies $\mathsf{set}\, x \subseteq H a$, where $\mathsf{set}\, x$ is the set of distinct elements in x.

The greedy condition has to be reformulated as follows: there exists an ordering $\leqslant_B$ on B such that for all a satisfying $\neg p\, a$ (and so $\llbracket\, \rrbracket \notin F a$), if

$$x \in F a \wedge \sqcap_B / H a \notin x$$

there is a y such that

$$y \in F a \wedge \sqcap_B / y <_B \sqcap_B / x \wedge C y \leqslant C x.$$

With this change, the greedy theorem holds for bags (provided we also change the program for f in the obvious way). More precisely,

Theorem 2
Let $F \in A \rightarrow \{\llbracket B \rrbracket\}^+$ and $f = \sqcap_C / \cdot F$. Suppose the cost, decomposition, and greedy conditions hold for C and F. Then there exists an ordering $\leqslant_C$ such that f can be computed by a scheme

$$f a = \begin{cases} \llbracket\, \rrbracket & \text{if } p\, a \\ \llbracket b \rrbracket \uplus f(a \ominus b) & \text{otherwise} \\ \text{where } b = \sqcap_B / H a. \end{cases}$$

The proof is a straightforward modification of the one in Bird (1992).

6 The smallest upravel

We now apply the theory above to a problem about sequences. By definition, an *unravel* of a sequence x is a bag of subsequences of x that when shuffled together can give back x. For example, 'accompany' can be unravelled into three subsequences: 'acm', 'an' and 'copy'. The order of these lists is not important, but duplications do matter; for example, 'peptet' can be unravelled into two copies of 'pet'. Thus, an unravel is essentially a *bag* of sequences and not a list or set.

An unravel is called an *upravel* if all its member sequences are in ascending order. Since each of 'acm', 'an' and 'copy' are ascending sequences (assuming normal alphabetical order) they give an upravel of 'accompany'. Each non-empty sequence has at least one upravel, namely the upravel consisting of singleton sequences. However, of all possible upravels we want to determine one with the least number of elements.

The problem can be specified as one of computing $f = \sqcap_{\#} / \cdot F$, where

$$F = \mathsf{all\ up} \lhd \cdot \mu(\rtimes^{\circ} \not\leftarrow \{\llbracket\, \rrbracket\}).$$

The operators μ, $\rtimes$ and $\rtimes^{\circ}$ were defined above, and the predicate up is given by

$$\mathsf{up}\, x = (\forall i, j \in [0 \rightarrow \# x] : i < j \Rightarrow \mathsf{index}\, x\, i \leqslant \mathsf{index}\, x\, j),$$

where $[0 \rightarrow n]$ is the interval consisting of all natural j with $0 \leqslant j < n$, and $\mathsf{index}\, x j$ is the element of x at position j.

Can we apply the greedy theorem to this problem? Certainly, # (the size function on bags) is a cost function. Moreover, we know from the previous sections that F satisfies

$$F x = \{\, \lfloor\rfloor \mid x = [\,]\} \cup \{\, \lfloor y \rfloor \uplus ys \mid y \in H x \wedge ys \in \cup / F * T(x, y)\},$$

where
$$H x = \mathsf{up} \triangleleft \pi_1 * \mu(\bowtie) x$$

$$T(x, y) = \pi_2 * ((y =) \cdot \pi_1) \triangleleft \mu(\bowtie) x.$$

In fact, $H = \mathsf{up} \triangleleft \cdot \mathsf{subs}$, where $\mathsf{subs}\, x$ is the set of subsequences of x. We have $\pi_1 * \mu(\bowtie) x = \mathsf{subs}\, x$, and also $\pi_2 * \mu(\bowtie) x = \mathsf{subs}\, x$.

Setting $B = [A]$, it remains to find an ordering $\leqslant_B$ for which the greedy condition holds. One reasonable candidate for $\leqslant_B$ is some kind of lexicographical ordering. However, the standard lexicographical ordering on lists assigns $u \leqslant v$ when u is an initial segment of v; in particular, the empty sequence is the least element under $\leqslant$. It is likely that we get a smaller bag by choosing longer rather than shorter sequences at each stage; certainly, we never want to choose the empty sequence. So we shall reverse the standard convention and take $v \leqslant u$ when u is an initial segment of v. In the reversed lexicographical ordering, the empty sequence is the *last* entry in the dictionary. We define $u \leqslant_B [\,]$ for all u and

$$([a] \mathbin{+\!\!+} u) \leqslant_B ([b] \mathbin{+\!\!+} v) \equiv a < b \vee (a = b \wedge u \leqslant_B v).$$

With this choice of $\leqslant_B$ the greedy algorithm identifies $y = \mathsf{lus}\, x$ as the best sequence to choose, where

$$\mathsf{lus} = \sqcap_B / \mathsf{up} \triangleleft \cdot \mathsf{subs}.$$

To establish the greedy condition, we need some properties of lus, so we shall consider this function first.

6.1 The least upsequence

The first property is that $\mathsf{lus}\, x$ is the least subsequence of x:

$$\mathsf{lus} = \sqcap_B / \cdot \mathsf{subs}.$$

If the least subsequence y were not an upsequence, we could find a subsequence of y that was smaller than y.

Using this fact, and the definition of subs as a right-reduction, we can derive that $\mathsf{lus} = \oplus \not\leftarrow [\,]$, where $a \oplus [\,] = [a]$ and

$$a \oplus ([b] \mathbin{+\!\!+} x) = \begin{cases} [a, b] \mathbin{+\!\!+} x, & \text{if } a \leqslant b \\ [b] \mathbin{+\!\!+} x & \text{otherwise.} \end{cases}$$

For reasons of space, we shall leave this task as an exercise. It is also possible to derive the program by appealing to the greedy theorem, the only difference that the ordering is fixed and not specified by a cost function.

It follows from the above program that $\mathsf{lus}\, x$ is uniquely located as a subsequence of x. In other words, if $y = \mathsf{lus}\, x$, there is a unique z such that $(y, z) \in \mu(\asymp)\, x$. In fact, $z = x - y$, where $(-)$ is the list-difference operator defined by $x - [\,] = [\,]$ and

$$([a] \mathbin{+\!\!+} x) - ([b] \mathbin{+\!\!+} y) = \begin{cases} x - y & \text{if } a = b \\ [a] \mathbin{+\!\!+} (x - ([b] \mathbin{+\!\!+} y)) & \text{otherwise.} \end{cases}$$

Two other properties of lus follow from the program for computing it. Suppose $\mathsf{lus}\, x$ is located at position $[i_1, i_2, \ldots, i_n]$ in x. Then

$$i_{k-1} < j < i_k \Rightarrow \mathsf{index}\, x\, j > \mathsf{index}\, x\, i_k, \qquad (5)$$

$$i_k \leqslant j \Rightarrow \mathsf{index}\, x\, i_k \leqslant \mathsf{index}\, x\, j. \qquad (6)$$

Properties (5) and (6) are used below.

6.2 The greedy condition

Now for the greedy condition. Let $y = \mathsf{lus}\, x$ and suppose $us \in F x$ is such that $y \notin us$. We have to find a $vs \in F x$ so that $\sqcap_B / vs <_B \sqcap_B / us$ and $\# vs \leqslant \# us$.

Before starting, we should warn the reader that what follows is complicated by the fact that we have to refer to the *locations* of upsequences in x. Although the definition of $\leqslant_B$ does not depend on locations, the proof that it works does. This is because we are going to use a cut-and-paste argument, and we need to know that what we cut and paste remain subsequences of x.

Each $u \in us$ can be associated with some location $\mathsf{loc}\, u$ of u in x. The value $\mathsf{loc}\, u$ is a subsequence of $[0 \rightarrow \# x]$ satisfying

$$\mathsf{index}\, x * \mathsf{loc}\, u = u$$

and chosen so that the sequences $\mathsf{loc} * us$ partition $[0 \rightarrow \# x]$. As we have seen, the value $\mathsf{loc}\, y$ is fixed and independent of the way we choose loc for the bag us.

Let $u = \sqcap_B / us$. Since $y <_B u$, we know that y is not an initial segment of u, and so $\mathsf{loc}\, y$ is not an initial segment of $\mathsf{loc}\, u$. Let $\mathsf{loc}\, u1$ be the longest common initial segment of $\mathsf{loc}\, y$ and $\mathsf{loc}\, u$, and let $\mathsf{loc}\, y$ begin with $\mathsf{loc}\, u1 \mathbin{+\!\!+} [i]$. Let $\mathsf{index}\, x\, i = a$.

There are now two cases. Suppose firstly that $u1 = u$. Since $i \notin \mathsf{loc}\, u$, we have $i \in \mathsf{loc}\, v$ for some $v \in us$. Let $v = v1 \mathbin{+\!\!+} [a] \mathbin{+\!\!+} v2$, and define vs by

$$vs = us - \lbag u, v \rbag \uplus \lbag u \mathbin{+\!\!+} [a], v1 \mathbin{+\!\!+} v2 \rbag.$$

We have $\# vs = \# us$, and $u \mathbin{+\!\!+} [a]$ and $v1 \mathbin{+\!\!+} v2$ are upsequences of x. Since $u \mathbin{+\!\!+} [a] <_B u$, the greedy condition is established.

In the second case, $u1$ is a proper initial segment of u. Let $u = u1 \mathbin{+\!\!+} [b] \mathbin{+\!\!+} u2$, where $\mathsf{loc}\, u = \mathsf{loc}\, u1 \mathbin{+\!\!+} [j] \mathbin{+\!\!+} \mathsf{loc}\, u2$ and $b = \mathsf{index}\, x\, j$. We now argue that $i \notin \mathsf{loc}\, u2$, so $i \notin \mathsf{loc}\, u$. If $i \in \mathsf{loc}\, u2$, then $j < i$ and $a < b$ by property (5). But then u is not an upsequence.

So $i \notin \mathsf{loc}\, u$ and there is a $v \in us$ so that $i \in \mathsf{loc}\, v$. Let $v = v1 \mathbin{+\!\!+} [a] \mathbin{+\!\!+} v2$, where $\mathsf{loc}\, v = \mathsf{loc}\, v1 \mathbin{+\!\!+} [i] \mathbin{+\!\!+} \mathsf{loc}\, v2$. There are now two subsidiary cases, depending on whether $i < j$ or $i > j$. In both cases we need the fact that the last element l of $u1$, if it exists, satisfies $l < i \sqcap j$, since $\mathsf{loc}\, u1$ is the initial segment of y before i.

If $i < j$, then $a \leqslant b$ by property (6). We can therefore define vs by

$$vs = us - \lbag u, v \rbag \uplus \lbag u1 +\!\!+ [a,b] +\!\!+ u2, v1 +\!\!+ v2 \rbag .$$

Since $u1 +\!\!+ [a,b] +\!\!+ u2 <_B u1 +\!\!+ [b] +\!\!+ u2 = u$ the greedy condition is established.

If $i > j$, then $a < b$ by property (5). Let the last element of $\mathsf{loc}\, v1$, if it exists, be denoted by l'. We have $l' < i$, but also $l' < j$ because $l < j \leqslant l' < i$ implies $\mathsf{index}\, x\, l' > a$ by (5), contradicting the fact that v is an upsequence. Hence we can define vs by

$$vs = us - \lbag u, v \rbag \uplus \lbag u1 +\!\!+ [a] +\!\!+ v2, v1 +\!\!+ [b] +\!\!+ u2 \rbag .$$

Since $a < b$ we have $u1 +\!\!+ [a] +\!\!+ v2 <_B u1 +\!\!+ [b] +\!\!+ u2 = u$ and the greedy condition is again satisfied.

6.3 Optimization

The result is the following greedy algorithm for computing f:

$$f\,x = \begin{cases} \lbag\rbag, & \text{if } x = [\,] \\ \lbag y \rbag \uplus f(x - y), & \text{otherwise} \\ \text{where } y = \mathsf{lus}\, x. \end{cases}$$

With the given implementation of lus as a right-reduction, the greedy algorithm takes $O(n^2)$ steps, where $n = \#x$.

The greedy algorithm makes several passes through the input, computing a single component of the bag at each pass. To obtain a single-pass algorithm, we represent the bag $\lbag x1, x2, \ldots, xk \rbag$ by a sequence $[x1, x2, \ldots, xk]$, where $xi \leqslant_B xj$ if $i < j$. Then we have $x1 = \mathsf{lus}\, x$, $x2 = \mathsf{lus}\,(x - x1)$, and so on. Using this representation, we can compute f by a right-reduction $f = \odot \not\leftarrow [\,]$, where $a \odot [\,] = [[a]]$ and

$$a \odot ([x] +\!\!+ xs) = \begin{cases} [[a] +\!\!+ x] +\!\!+ xs & \text{if } a \leqslant \mathsf{hd}\, x \\ [x] +\!\!+ (a \odot xs), & \text{otherwise.} \end{cases}$$

For reasons of space we leave the derivation as another exercise. Implemented directly, evaluation of $a \odot xs$ takes time linear in the length of xs, but since the first elements of sequences in xs are in increasing order, we can use binary search to reduce this to logarithmic time. It follows that the smallest upravel can be computed in $O(n \log n)$ steps, where n is the length of the input.

7 Comments

The problem of the smallest upravel was first posed† by Kaldewaij (1985), who solved it by reducing it to the problem of computing a longest decreasing subsequence. Subsequently, Meertens (1985) gave a direct solution. The present treatment followed yet a third course, namely to reduce it to an instance of the greedy theorem.

There is a lot to be gained from the study of the example. First of all, there is the specification which uses the converse operator μ. Specifications using μ or Inv arise

† Added in proof: we have learnt subsequently that the problem is due to Meertens.

frequently in the formulation of optimization problems. Second, there is the idea of phrasing the problem in terms of bags rather than lists. Although the final form of the algorithm represents bags by ordered lists, the order of the lists is not arbitrary. Indeed, the ordering $\leqslant_B$ is crucial to the success of the algorithm and only emerges as a result of trying to satisfy the greedy condition. Third, there is the way the greedy algorithm appears. The idea of choosing the lexicographically least upsequence as the 'best' one at each stage is, though reasonable, not entirely obvious (the technical term is 'rabbit'), and the accompanying proof that it works is fairly subtle. I have tried without success to find a simpler proof.

Similar problems with rabbits arose in Kaldewaij (1985) and Meertens (1985). Kaldewaij used a combinatorial lemma, a dual of Dilworth's theorem, to reduce the problem to one of computing a longest decreasing subsequence. But the idea of a longest decreasing subsequence appears nowhere in the formulation of the problem. Meertens used an inductive strategy, essentially the idea of heading for a right-reduction, from the outset. First, the definition of F was massaged into the form of a right-reduction, and then this pattern of computation was promoted over the filter and the minimizing function. Unfortunately, this method involved inventing a non-obvious preorder. The only conclusion I can draw is that all approaches to the problem involve a rabbit, and perhaps the same is true for other greedy algorithms.

References

Bird, R. S. 1992. Two greedy algorithms. *J. Functional Programming*, 2 (1).
Kaldewaij, A. 1985. On the decomposition of sequences into ascending subsequences. *Infor. Processing Lett.*, 21, 69.
Meertens, L. 1985. Some more examples of algorithmic developments. *IFIP Wg2.1 Working Paper*, Pont à Mousson, France.