Using Underspecification in the Derivation of
some Optimal Partition Algorithms

Maarten M Fokkinga

Centre for Mathematics and Computer Science, Amsterdam
University of Twente, Enschede

April 23, 1990

Abstract

Indeterminacy is inherent in the *specification* of optimal partition (and many more) algorithms, even though the algorithms themselves may be fully determinate. Indeterminacy is a notoriously hard phenomenon to deal with in a purely functional setting. In the paper "A Calculus Of Functions for Program Derivation" R. S. Bird tries to handle it by using *underspecified* functions. (Other authors have proposed to use 'indeterminate' functions, and to use relations instead of functions.)

In this paper we redo Bird's derivation of the Leery and Greedy algorithm while being very precise about underspecification, and still staying in the functional framework. It turns out that Bird's theorems are not exactly what one would like to have, and what one might understand from his wording of the theorems. We also give a derivation in the Bird-Meertens style of a (linear time) optimal partition algorithm that was originally found by J. C. S. P. van der Woude.

1   Introduction

We are considering problems of the following form: construct an efficiently computable expression (an algorithm) for the function that yields some optimal partition of its argument, or more precisely for the function that, for given predicate p and function f, yields some minimally f-valued partition (of its argument sequence) all of whose segments satisfy p. The specification does not prescribe which one to choose if there are several partitions that have an equal f-value, but the algorithm may (and in a functional setting *must*) be fully determinate.

The indeterminacy in the specification is notoriously hard to handle formally. Bird[5] employs a technique of *underspecification*: he uses functions whose definition is not fully given at the start of the derivation, but whose definition may be completed along the way. Unfortunately, Bird is not very precise about underspecification and some of his results are misleading or even wrong, and his Leery and Greedy Theorems are not what one would like to have (they suggest to provide an algorithm, but that is not yet the case). We set out to redo his derivation while clarifying the use of underspecification and avoiding the flaws. Intentionally, we stay as close as possible to Bird's concepts and terminology, just to see how far one can go in a purely functional framework.

Apart from a redo of the derivation of the Leery and the Greedy algorithms, we also present a further efficiency improvement of the Greedy algorithm, which we term the Gluttonous one.

This algorithm has originally been derived by Van der Woude[23] in the typical Eindhoven style of program derivation. He challenged us to give a derivation in the Bird-Meertens style.

It turns out that for each of the algorithms the driving heuristics can be described as the wish to "promote" (shift) some term of the expression to another place within the expression.

As regards to indeterminacy, other authors are working on possibly more general solutions than the use of underspecification. We mention two directions.

- Some researchers, like Backhouse[2] and De Moor[19], propose to employ the relational framework since, indeed, in a *relation* f one "input" x may be related by f to several "outputs" y, and hence indeterminacy is built in into the concepts. An implementation g of a relation f is not required to be extensionally the same as f but may be a *refinement* ($g \subseteq f$) and thus be more determinate than the initial, specified relation f. This approach looks very promising, but requires a redo of the theory of program development in a functional setting.

- Other people, like Meertens[16, 17] have proposed to develop the concept of *indeterminate function*, the basic one being the indeterminate choice selector $[\![$. The advantage is that we can continue to reason as we are used to do, viz. with functions, but there is the lurking danger of inconsistency: if all the desirable laws for functions (like the well known bêta and eta rule) are postulated to hold of indeterminate functions too, then the the laws together are inconsistent. So there is the need to choose which laws to keep and which ones to drop, the choice being guided by the ultimate goal of a *calculus* for program derivation. A definite solution has not been found.

In an imperative setting indeterminacy is handled by characterising the state changes by pre- and postconditions: state changes are effectively relations.

The style of program development that we shall follow is becoming known as the Bird-Meertens style, after its originators Meertens[16] and Bird[5]. Characteristic is its algebraic nature — one really calculates with *terms* (i.e., programs) exactly as in arithmetic where we calculate with terms that denote numbers (e.g., consider the short calculation $(a + b)(a - b) = a^2 + ab - ab - b^2 = a^2 - b^2$). In order that such a calculational method of program derivation is practically feasible several requirements have to be met: the notation should be compact and easy to manipulate, the concepts that are given shape syntactically should be general and have laws that are easy to apply, the number of laws should be not too large, and so on. In a series of papers such a calculus is being developed; see Meertens[16, 18], Bird[3, 5, 4, 6], Backhouse[1], Jeuring[10, 11], Malcolm[14, 13, 12], De Moor[19, 21, 20], Fokkinga[9, 8] to mention a few. We shall only briefly explain the notation and the laws that we need, and we shall not be very detailed in the actual calculation steps; for an introduction (and an exposition of detailed calculations) the reader is recommended to consult the above literature.

2 Preliminaries

We assume that the reader is familiar with the notions for sequences and sets.

α-set	$=$	the set of sets over α
$\alpha*$	$=$	the set of sequences over α

2

$$
\begin{array}{lll}
\alpha+ & = & \text{the set of non-empty sequences over } \alpha \\
\tau : \alpha \to \alpha+ & = & \text{singleton former} \\
{+\!\!+} : \alpha* \parallel \alpha* \to \alpha* & = & \text{associative ``join'' operation for sequences} \\
{+\!\!+}' : \alpha+ \parallel \alpha* \to \alpha* & = & \text{another ``join'' operation for sequences} \\
[\,] : \alpha* & = & \text{the empty sequence, an identity for } {+\!\!+} \text{ and a right unit for } {+\!\!+}' \quad .
\end{array}
$$

Important functions on sequences are reductions $\oplus/$, maps $f*$, and filters $p\triangleleft$. Appendix B contains the "standard knowledge" on these, as well as on the important Promotion Theorem and the notions of promotability and distributivity. In the sequel we let the verb "to promote" also mean "to shift right a term within an expression" like what is done within the Promotion Theorem with the promotable function.

There are two specific functions that occur throughout the paper: Inv and parts. $Inv(f)$ yields the set of originals under f of its argument, and parts yields the set of all partitions of its argument, a partition being a sequence of non-empty contiguous segments:

$$
\begin{array}{lll}
Inv(f) \cdot x & = & \{y \mid y \in Dom(f) \wedge fy = x\} \\
\textsf{parts} & : & \alpha* \to \alpha+*\text{-set} \\
\textsf{parts} & = & (\textsf{all}\,(\neq[\,]))\triangleleft \cdot Inv({+\!\!+}/) \\
& = & Inv({+\!\!+}' \not{+\!\!+} [\,]) \quad .
\end{array}
$$

The second equation for parts has the advantage that the test for non-emptiness does not need to be written explicitly, thus simplifying the formulas. De Moor[21] give various laws for Inv so that one can derive the following equations for parts from its specification.

$$
\begin{array}{llll}
(1) & \textsf{parts} \cdot [\,] & = & \{[\,]\} \\
(2) & \textsf{parts} \cdot x & = & \cup/ \cdot (({+\!\!+} \cdot \tau) \parallel_* \textsf{parts})* \cdot Inv({+\!\!+}') \cdot x \qquad \text{if } x \neq [\,] \quad .
\end{array}
$$

As an aid to the uninitiated reader, we transliterate the latter equation into more conventional notation. First rename x into $[a] {+\!\!+} x$ (since it is not empty). Then work out $Inv({+\!\!+}') \cdot [a] {+\!\!+} x$ a little bit, eliminating the implicit test for non-emptiness, and rewrite various terms one after the other. We get

$$
\begin{aligned}
& \textsf{parts} \cdot [a] {+\!\!+} x \\
={}& \cup/ \cdot (({+\!\!+} \cdot \tau) \parallel_* \textsf{parts})* \cdot ([a] {+\!\!+} \parallel id)* \cdot Inv({+\!\!+}) \cdot x \\
={}& \cup/ \cdot (({+\!\!+} \cdot \tau) \parallel_* \textsf{parts})* \cdot ([a] {+\!\!+} \parallel id)* \cdot \{(y,z) \mid y,z :: y {+\!\!+} z = x\} \\
={}& \cup/ \cdot (({+\!\!+} \cdot \tau) \parallel_* \textsf{parts})* \cdot \{([a] {+\!\!+} y, z) \mid y,z :: y {+\!\!+} z = x\} \\
={}& \cup/ \cdot \{({+\!\!+} \cdot \tau) \parallel_* \textsf{parts} \cdot ([a] {+\!\!+} y, z) \mid y,z :: y {+\!\!+} z = x\} \\
={}& \cup/ \cdot \{({+\!\!+} \cdot \tau \cdot [a] {+\!\!+} y)* (\textsf{parts}\,z) \mid y,z :: y {+\!\!+} z = x\} \\
={}& \cup/ \cdot \{([[a] {+\!\!+} y]{+\!\!+})* \textsf{parts}\,z \mid y,z :: y {+\!\!+} z = x\} \\
={}& \cup/ \cdot \{\{([[a] {+\!\!+} y]{+\!\!+})* u \mid u :: u \in \textsf{parts}\,z\} \mid y,z :: y {+\!\!+} z = x\} \\
={}& \{([[a] {+\!\!+} y]{+\!\!+}) \cdot u \mid u,y,z :: y {+\!\!+} z = x \wedge u \in \textsf{parts}\,z\} \\
={}& \{[[a] {+\!\!+} y] {+\!\!+} u \mid u,y,z :: y {+\!\!+} z = x \wedge u \in \textsf{parts}\,z\}
\end{aligned}
$$

The final expression is less suited for algebraic manipulation than expression (2), since in (2) all the semantically meaningful constituents are expressed by separate syntactical constituents, facilitating to manipulate them separately.

<div align="center">3</div>

If f is a function into a set with relation $\lozenge$, we define $\lozenge_f$ on the domain of f by

$$
\begin{aligned}
x \lozenge_f y &\equiv f\, x \lozenge f\, y \\
&\equiv \lozenge \cdot f \parallel f \cdot (x, y) \quad .
\end{aligned}
$$

In particular, we will make frequent use of $<_f$, $\le_f$ and $=_f$. All orders in this paper are total, i.e., well-defined for all arguments. If the range of f is linearly ordered, then $<_f$ is a transitive relation, and $\le_f$ is a pre-order; ($\le_f$ is not necessarily anti-symmetric, as for some $x \ne y$ it may hold that $x =_f y$). For any pre-order $\preceq$ we let $\prec$ mean the transitive relation defined by $x \prec y \equiv x \preceq y \land \lnot(y \preceq x)$. Similarly, $\simeq$ is the equivalence relation defined by $x \simeq y \equiv x \preceq y \land y \preceq x$. It then follows that $x \prec y \lor x \simeq y \lor y \prec x$ holds for all x and y; and these definitions hold also with $<_f$ and $=_f$ for $\prec$ and $\simeq$.

For functions g, h we set $g \le_f h$ if for all x, $g\,x \le_f h\,x$. As usual we say that a relation $\preceq$ *refines*, or *is a refinement of* another relation $\le$ if for all x and y, $x \preceq y \Rightarrow x \le y$. For any *linear* order $\le$ we define the function $\downarrow_\le$ by

$$
\begin{aligned}
x \downarrow_\le y &= x \quad \text{if } x \le y \\
&= y \quad \text{if } y \le x \quad .
\end{aligned}
$$

In the sequel we let α, β be types, and we use the following nomenclature consistently:

$$
\begin{array}{rcll}
a &:& \alpha & \text{element} \\
x, y, z &:& \alpha* & \text{list or segment, also ``main argument''} \\
u, v, w &:& \alpha** & \text{partition, or segment list} \\
r &:& \alpha**\text{-set} & \text{set of partitions.}
\end{array}
$$

For filtering on the first components we use a notational shorthand:

$$
p \triangleleft_1 = (p \cdot \pi_1) \triangleleft \quad .
$$

3 The specification

Throughout the paper we assume given a predicate p on $\alpha*$ such that p holds of all singletons, and a function f from $\alpha**$ into a set β linearly ordered by $\le$. We want to derive an algorithm that yields some minimally f-valued $(\mathsf{all}\,p)$-satisfying partition of its argument, if it exists. In order to avoid unnecessary complications we have assumed p to be true of all singleton sequences: it implies that $(\mathsf{all}\,p) \triangleleft \cdot \mathsf{parts} \cdot x$ is nonempty for all x (and conversely), so that such a partition exists for any argument. Thus the specification reads:

> Given f, $\le$ and p such that p is true of all singletons, find an efficiently computable $\mathcal{S}$ satisfying

$$
\text{(3)} \qquad \forall x: \quad \mathcal{S}\,x \in r \ \land \ \forall u \in r: \mathcal{S}\,x \le_f u
$$
$$
\text{where } r = (\mathsf{all}\,p) \triangleleft \cdot \mathsf{parts} \cdot x \quad .
$$

In a previous version of this paper, [7], we have shown how one can do without the precondition that p holds of all singletons. The solution is to introduce a fictitious value which plays the rôle of 'no such partition exists', and adjoin it to $\alpha**$. We do not choose this option here since it would distract the attention of the main topic.

A more manageable specification

Unfortunately we cannot begin *calculating* an efficiently computable solution for $\mathcal{S}$ in Specification (3). For this we need an initial expression which $\mathcal{S}$ is equal to (and which is possibly not yet efficient enough, or not at all computable). We shall now construct such an expression; we aim at a reduce. First we define for any pre-order $\preceq$ (like $\leq_f$)

$\mathsf{ACISM}(\preceq) =$ the set of all operations $\oplus$ on $\alpha**$ satisfying

Associativity	$(u \oplus v) \oplus w$	$=$	$u \oplus (v \oplus w)$
Commutativity	$u \oplus v$	$=$	$v \oplus u$
Idempotence	$u \oplus u$	$=$	u
Selectivity	$u \oplus v$	$\in$	$\{u, v\}$
Minimality	$u \oplus v$	$\preceq$	u, v .

In the sequel we will often make assertions about $\mathsf{ACISM}(\leq_f)$ where we could have given more general assertions by replacing the particular $\leq_f$ by an arbitary pre-order $\preceq$. In this way we can stay more closely to the terminology and notations of Bird[5].

Notice that these five properties are not independent of each other, in particular idempotency follows from selectivity. The first three properties allow us to use any $\oplus \in \mathsf{ACISM}(\leq_f)$ in a reduce over a set. Notice also that any two operations in $\mathsf{ACISM}(\leq_f)$ differ from each other only for arguments $u \neq v$ with $u =_f v$. Throughout the sequel we use $\downarrow_f$ as a *variable* ranging over $\mathsf{ACISM}(\leq_f)$; it is *not* the notation of a fixed function, like $\downarrow$, $+$, $+\!\!+$ and even f are. Bird[5] uses the symbol $\downarrow_f$ to denote an underspecified function whose definition he intends to complete along the way. This interpretation is confirmed by De Moor[19]. However, it is not at all clear where and how the definition of that function is supplemented. We formalise the underspecification by the set $\mathsf{ACISM}(\leq_f)$ and by having $\downarrow_f$ range over this set. (Using the symbol $\downarrow_f$ as a variable rather than as a fixed function is quite misleading and confusing, I think, but it is just the proper way to understand what Bird[5] is doing.)

Now we have that Specification (3) is equivalent to each of (5), (6), and (7) below; and these formulations provide us with an expression for the unknown $\mathcal{S}$ so that we can begin calculating a more efficient expression for it.

(4) Alternative Specifications Lemma   *Given that p holds of all singletons, specification (3) is equivalent to each of the following, where $R = (\mathsf{all}\, p) \triangleleft \cdot \, \mathsf{parts}$*

(5) $\qquad \exists \downarrow_f \in \mathsf{ACISM}(\leq_f): \quad \mathcal{S} =_f \downarrow_f / \cdot R \quad \wedge \quad \forall x: \mathcal{S}\, x \in R\, x$

(6) $\qquad \forall \downarrow_f \in \mathsf{ACISM}(\leq_f): \quad \mathcal{S} =_f \downarrow_f / \cdot R \quad \wedge \quad \forall x: \mathcal{S}\, x \in R\, x$

(7) $\qquad \exists \downarrow_f \in \mathsf{ACISM}(\leq_f): \quad \mathcal{S} = \downarrow_f / \cdot R$

To prove the equivalences we need the following two lemmas, of which some parts are mentioned for later use only.

(8) Least-f Characterisation Lemma   *For all f, all $\downarrow_f \in \mathsf{ACISM}(\leq_f)$ and all $r \neq \emptyset$:*

$\qquad \downarrow_f / r \in r \ \wedge \ \forall u \in r: \downarrow_f / r \leq_f u \quad .$

(9) ACISM Lemma   *For all f and $\leq$*

   1. if $\preceq$ is a linear order refining $\leq_f$, then function $\downarrow_\preceq \in \mathsf{ACISM}(\leq_f)$;

   2. there exists a linear order $\preceq$ that refines $\leq_f$;

   3. $\mathsf{ACISM}(\leq_f)$ is not empty;

   4. if f is injective, then $\mathsf{ACISM}(\leq_f)$ has just one member;

   5. for all $\downarrow_f \in \mathsf{ACISM}(\leq_f)$, the relation $\preceq$ defined by $u \preceq v \;\equiv\; (u{\downarrow_f}v = u)$, is a linear order refining $\leq_f$, and $\downarrow_\preceq = \downarrow_f$.

The proof of these two lemmas is easy; we give only some hints. For Lemma (8) use induction on the structure (or size) of r. For Lemma (9) we mention the following.

  9.1 Associativity of $\downarrow_\preceq$ follows from transitivity of $\preceq$,
      commutativity of $\downarrow_\preceq$ from anti-symmetry of $\preceq$,
      idempotence of $\downarrow_\preceq$ from reflexivity of $\preceq$,
      selectivity is immediate by the definition of $\downarrow_\preceq$, and
      minimality follows from $\preceq$ refining $\leq_f$.

  9.2 By the axiom of choice any set can be well-ordered; say the domain of f is well-ordered by $\sqsubseteq$. Then define $\preceq$ by $x \preceq y \;\equiv\; x <_f y \vee (x =_f y \wedge x \sqsubseteq y)$.

  9.3 Immediate from 9.1 and 9.2.

  9.4 The selectivity and minimality properties are sufficient to prove any two members of $\mathsf{ACISM}(\leq_f)$ equal.

  9.5 As in 9.1 the associativity, commutativity and idempotence of $\downarrow_f$ imply the transitivity, respectively anti-symmetry and reflexivity of $\preceq$.

Returning to the Alternative Specifications Lemma, the mutual equivalence of (3), (5), (6), and (7) is now easily shown; again we give some hints only.

(3) $\Rightarrow$ (5) By (9.3), let $\preceq$ be some linear ordering that refines $\leq_f$. Then by (9.1) $\downarrow_\preceq \in \mathsf{ACISM}(\leq_f)$, and it is easily verified using (8) that $\mathcal{S} =_f \downarrow_\preceq / R$.

(5) $\Rightarrow$ (6) Assume $\downarrow_f \in \mathsf{ACISM}(\leq_f)$ and $\mathcal{S} =_f \downarrow_f / R$; define $S = \downarrow_f / R$. Let $\oplus$ be arbitrary $\in \mathsf{ACISM}(\leq_f)$, and define $T = \oplus / R$. Let x be arbitrary and put $r = R\,x$. We know that $r \neq \emptyset$ since p holds of all singletons. Now, using Lemma (8) it follows that $S\,x \in r \wedge (\forall u \in r : S\,x \leq_f u)$ and that $T\,x \in r \wedge (\forall u \in r : T\,x \leq_f u)$. So $S\,x =_f S\,x \leq_f T\,x \leq_f S\,x =_f \mathcal{S}\,x$.

(6) $\Rightarrow$ (7) Define a *partial linear* order $\preceq$ by $u \preceq v \;\equiv\; \exists x : \mathcal{S}\,x = u \wedge u \in R\,x \wedge v \in R\,x$. (For transitivity and anti-symmetry it is relevant that the sets $R\,x = (\mathsf{all}\,p)\triangleleft \cdot \mathsf{parts} \cdot x$ are disjoint for distinct x.) Extend the partial linear order into a total linear one: $\preceq$. (In case α, hence $\alpha{*}{*}$, is countable, this is quite simple. In case $\alpha{*}{*}$ is uncountable, we need nonconstructive methods like transfinite induction and the axiom of choice. In view of the fact that we are dealing with computational problems, we might suppose that all the types we encounter are effectively enumerable.) Then $\mathcal{S} = \downarrow_\preceq / R$.

(7) $\Rightarrow$ (3) Immediate by the Least-f Characterisation (8).

Refinements

Bird[5] defines that a function g refines, or is a refinement of, f if $<_g$ refines $<_f$. However with this definition we get into trouble sooner or later, because we shall need that the ranges of g and f are different, and that the order for the g-values differs from the order for the f-values. What we need is the notion of a pre-order $\preceq$ (e.g., $\leq_g$) being a refinement of $\leq_f$; this notion of refinement has already been defined. The following lemma shows that the notion of refinement may be useful for the problem at hand.

(10) Refinement Lemma

$$\preceq_g \text{ is a refinement of } \leq_f \quad \equiv \quad \mathsf{ACISM}(\preceq_g) \subseteq \mathsf{ACISM}(\leq_f).$$

Proof The 'implies' part is easy and therefore omitted. For the 'follows from' part we argue as follows. Let x and y be arbitrary, and suppose $x \preceq_g y$; we need to show $x \leq_f y$. Let $\sqsubseteq$ be a linear order refining $\preceq_g$ such that $x \sqsubseteq y$. Now, $\downarrow_\sqsubseteq \in \mathsf{ACISM}(\preceq_g) \subseteq \mathsf{ACISM}(\leq_f)$ and also $x \downarrow_\sqsubseteq y = x$. Hence by the minimality property of members of $\mathsf{ACISM}(\leq_f)$ we find $x \leq_f y$, as desired. $\qquad\square$

Consequently, if g and $\preceq$ are such that $\preceq_g$ is a refinement of $\leq_f$ and S satisfies

$$\exists \downarrow_g \in \mathsf{ACISM}(\preceq_g)\colon\ S \ = \ \downarrow_g /\cdot (\mathsf{all}\,p) \triangleleft \cdot \mathsf{parts} \quad,$$

then S is a solution for $\mathcal{S}$ in (3).

4 The Leery Theorem

In view of the alternative specifications (5) and (7) we consider, for any $\downarrow_f \in \mathsf{ACISM}(\leq_f)$, a function $S(\downarrow_f)$ defined by

$$(11)\quad S(\downarrow_f) \ = \ \downarrow_f /\cdot (\mathsf{all}\,p) \triangleleft \cdot \mathsf{parts} \quad.$$

This definition will be valid throughout the remainder of the paper. Clearly, for any $\downarrow_f \in \mathsf{ACISM}(\leq_f)$ this particular $S(\downarrow_f)$ is a solution for $\mathcal{S}$ in (3). Therefore it suffices to find an efficiently computable expression for any or some $S(\downarrow_f)$. For the time being we fix the choice of $\downarrow_f$ arbitrarily, but sometimes we will need to make a particular choice. (Remember also that an efficiently computable expression for any T with $T =_f S(\downarrow_f)$ solves our task as well. Such a T pops up in the derivation of the Gluttonous algorithm.)

There is not much we can do but for working out parts. Suppose we remember Equations (1,2). We can substitute these equations in (11). What to do next? Let us *try and promote* $\downarrow_f /\cdot (\mathsf{all}\,p) \triangleleft$ to the right, to the recursive call of parts, so as to obtain a recursive equation for $S(\downarrow_f)$. This, then, is done in the proof of the next theorem.

(12) Bird's Leery Theorem   *Suppose $+\!\!\!+$ distributes backwards over $\downarrow_f$ and p holds of all singletons. Then*

$$
\begin{aligned}
S(\downarrow_f)\cdot [] \ &= \ [] \\
S(\downarrow_f)\cdot x \ &= \ \downarrow_f /\cdot (\tau \parallel_{+\!\!\!+} S(\downarrow_f)) *\cdot p\triangleleft_1 \cdot Inv(+\!\!\!+') \cdot x \qquad \text{if } x \neq [] \quad.
\end{aligned}
$$

We will refer to these equations as the Leery Equations for $S(\downarrow_f)$.

Proof Abbreviate $S(\downarrow_f)$ to S. Consider in turn the case $S\,[\,]$, and $S\,x$ where $x \neq [\,]$.

$$S \cdot [\,]$$
$$= \quad \text{unfold } S$$
$$\downarrow_f/ \cdot (\mathsf{all}\,p)\triangleleft \cdot \mathsf{parts} \cdot [\,]$$
$$= \quad \text{equation for } \mathsf{parts}$$
$$\downarrow_f/ \cdot (\mathsf{all}\,p)\triangleleft \cdot \{[\,]\}$$
$$= \quad \text{calculus}$$
$$[\,]$$

as desired. Next, for $x \neq [\,]$,

$$S \cdot x$$
$$= \quad \text{unfold } S\,, \text{ eqn for } \mathsf{parts}\,, \; x \neq [\,]$$
$$\downarrow_f/ \cdot (\mathsf{all}\,p)\triangleleft \cdot \cup/ \cdot ((+\!\!+ \cdot \tau) \,\|_* \mathsf{parts})* \cdot Inv(+\!\!+') \cdot x$$
$$= \quad \text{promoting } (\mathsf{all}\,p)\triangleleft \text{ as motivated above}$$
$$\downarrow_f/ \cdot \cup/ \cdot ((+\!\!+ \cdot \tau) \,\|_* ((\mathsf{all}\,p)\triangleleft \cdot \mathsf{parts}))* \cdot p\triangleleft_1 \cdot Inv(+\!\!+') \cdot x$$
$$= \quad \text{promoting } \downarrow_f/ \text{ as motivated above}$$
$$\downarrow_f/ \cdot (\downarrow_f/ \cdot (+\!\!+ \cdot \tau) \,\|_* ((\mathsf{all}\,p)\triangleleft \cdot \mathsf{parts}))* \cdot p\triangleleft_1 \cdot Inv(+\!\!+') \cdot x$$
$$= \quad \textbf{assume: } \downarrow_f/ \cdot +\!\!+ \,\|_* id \cdot (u,r) \; = \; id \,\|_{+\!\!+} (\downarrow_f/) \cdot (u,r)$$
$$\qquad \text{for all its arguments } u, r \; \text{ (see below)}$$
$$\downarrow_f/ \cdot (\tau \,\|_{+\!\!+} (\downarrow_f/ \cdot (\mathsf{all}\,p)\triangleleft \cdot \mathsf{parts}))* \cdot p\triangleleft_1 \cdot Inv(+\!\!+') \cdot x$$
$$= \quad \text{fold } S$$
$$\downarrow_f/ \cdot (\tau \,\|_{+\!\!+} S)* \cdot p\triangleleft_1 \cdot Inv(+\!\!+') \cdot x$$

as desired. The above assumed property of $\downarrow_f$ reads more conventionally written: for all its arguments u, r

$$\downarrow_f/ \cdot (u+\!\!+)* \cdot r \; = \; u +\!\!+ \cdot \downarrow_f/ \cdot r \quad .$$

It so happens that any r is nonempty, since it equals some $(\mathsf{all}\,p)\triangleleft \cdot \mathsf{parts} \cdot v$ (namely for some $v \in Inv(+\!\!+')\,x$). By the Promotion Theorem the equality follows from $u+\!\!+$ being $\downarrow_f$-promotable on nonempty domains, i.e.,

$$(13) \quad (u +\!\!+ v) \downarrow_f (u +\!\!+ w) \quad = \quad u +\!\!+ (v \downarrow_f w) \quad ,$$

for all u. This just asserts that $+\!\!+$ distributes backwards over $\downarrow_f$. $\qquad\square$

Remarks

(14) Distributivity of $u+\!\!+$ over $\downarrow_f$ is a bit stronger than necessary. Looking at the place where it is used, we see that it is sufficient if (13) holds for u, v, w that satisfy

$$u +\!\!+ v, \; u +\!\!+ w \; \in \; (\mathsf{all}\,p)\triangleleft \cdot \mathsf{parts} \cdot y$$

for some y. So, e.g., u, v, w containing empty members (segments) or satisfying $+\!\!+/ \cdot u +\!\!+ v \neq +\!\!+/ \cdot u +\!\!+ w$ need not satisfy the equation.

(15) We have proved the validity of the Leery Equations. It is a different task to show that these constitute an inductive definition of $S(\downarrow_f)$. This is obvious here, and will not be mentioned any more in the sequel. $\qquad\square$

So far for the *calculation* of a hopefully efficiently computable expression; the remainder of this section discusses the applicability of the theorem.

Using Bird's Leery Theorem

Bird's Leery Theorem is not quite the kind of statement that one would like to have: there is a condition on $\downarrow_f$ whereas operation $\downarrow_f$ is not given in the specification — only $\leq$, f and p are. In order to use the theorem (and to use the Leery Equations to define a solution for $\mathcal{S}$ in the specification) one has to choose (and implement) some operation $\downarrow_f \in \mathsf{ACISM}(\leq_f)$ and to verify that $+\!\!\!+$ distributes backwards over $\downarrow_f$. So we look for a condition on f that guarantees all or some $\downarrow_f$ to have the required distributivity property. Following Bird we define for a pre-order $\preceq$ (like $\leq_f$)

- $\preceq$ is *prefix-stable* if $v \preceq w \;\equiv\; u +\!\!\!+ v \preceq u +\!\!\!+ w$.

- $\preceq$ is *weakly prefix-stable* if $v \preceq w \;\Rightarrow\; u +\!\!\!+ v \preceq u +\!\!\!+ w$.

(Bird says that the function f is prefix-stable instead of the pre-order $\leq_f$; this causes trouble since the order $\leq$ is then implicit.) It would be nice if prefix stability of $\leq_f$ would imply that $+\!\!\!+$ distributes backwards over $\downarrow_f$ for all $\downarrow_f \in \mathsf{ACISM}(\leq_f)$. Although one might understand so from Bird's wording of the definition of prefix stability [5, Definition 1], it is not true. A counterexample is given by De Moor[19]:

> Take α to be the natural numbers and $\sqsubseteq$ the lexicographic order on $\alpha*$. Let f be $\#$ and $\leq$ be the conventional order on the naturals (so that $\leq_f$ is prefix-stable). Let $\preceq$ be the refinement of $\leq_f$ into the linear order on $\alpha**$ given by
>
> $$\begin{aligned} v \preceq w \quad\equiv\quad & v <_f w \;\vee \\ & (v =_f w \wedge v <_{\downarrow/\,\cdot\,+\!\!\!+/} w) \;\vee \\ & (v =_f w \wedge v =_{\downarrow/\,\cdot\,+\!\!\!+/} w \;\wedge v \sqsubseteq_{+\!\!\!+/} w) \quad. \end{aligned}$$
>
> Then by Lemma (9.1) $\downarrow_\preceq \in \mathsf{ACISM}(\leq_f)$ and, taking $u = [[0]]$, $v = [[1],[2]]$, $w = [[2],[0]]$ we find that $u+\!\!\!+$ does not distribute over $\downarrow_\preceq$:
>
> $$(u +\!\!\!+ v) \downarrow_\preceq (u +\!\!\!+ w) = u +\!\!\!+ v \neq u +\!\!\!+ w = u +\!\!\!+ (v\downarrow_\preceq w) \quad.$$

Yet one can guarantee the existence of a suitable $\downarrow_f$.

(16) Lemma *Suppose α is countable, $f : \alpha** \to \mathsf{Rationals}$, and $\leq_f$ is weakly prefix-stable. Then there exists (effectively) a $\downarrow_f \in \mathsf{ACISM}(\leq_f)$ such that $+\!\!\!+$ distributes backwards over $\downarrow_f$, and so the Leery Equations are valid for $S(\downarrow_f)$.*

Proof In the proof of his Lemma 1 Bird[5] constructs (by means of an auxiliary injective function $g \in \alpha** \to \mathsf{Rationals}$) a linear order $\preceq = \leq_g$ that refines $\leq_f$. By Lemma (9.1) $\downarrow_\preceq \in \mathsf{ACISM}(\leq_f)$. It is easily shown that $+\!\!\!+$ distributes backwards over $\downarrow_\preceq$. (This has been observed by De Moor[19].) $\qquad\square$

The lemma is not very useful for actual computational purposes, because the $\downarrow_f$ mentioned in the equations isn't efficiently computable. Fortunately there are two ways out.

Theorems

(17) *Suggested by Richard Bird. Suppose $\leq_f$ is prefix-stable and p holds of all singletons. Define a pre-order $\preceq$ on $\alpha**$ by*

$$v \preceq w \;\equiv\; v <_f w \vee (v =_f w \wedge v \sqsubseteq_{\#*} w)$$

where $\sqsubseteq$ is the lexicographic order on sequences of numbers. Let $\downarrow_f$ be arbitrary in $\mathsf{ACISM}(\preceq)$. *Then $\downarrow_f \in \mathsf{ACISM}(\leq_f)$ and the Leery Equations are valid for $S(\downarrow_f)$.*

(18) *From De Moor[21]. Suppose $f = \#$, $\leq$ is the usual order on numbers, p holds of all singletons, and α is linearly ordered (inducing the linear lexicographic order $\sqsubseteq$ on $\alpha*$). Define the linear order $\preceq$ on $\alpha**$ by*

$$v \preceq w \;\equiv\; v <_f w \vee (v =_f w \wedge v \sqsubseteq_{+\!/} w) \quad .$$

Then $\downarrow_{\preceq} \in \mathsf{ACISM}(\leq_f)$ and the Leery Equations are valid for $S(\downarrow_{\preceq})$.

Proofs

Ad (17). Observe that $\preceq$ refines $\leq_f$, so by the Refinement Lemma (10) we have $\downarrow_f \in \mathsf{ACISM}(\leq_f)$. Moreover, for any y we have that $\preceq$ is a linear order **on** $(\mathsf{all}\,p)\triangleleft \cdot \mathsf{parts} \cdot y$ and hence, together with the prefix stability of $\leq_f$, operator $+\!\!\!+$ distributes backwards over $\downarrow_f$ **on** $(\mathsf{all}\,p)\triangleleft \cdot \mathsf{parts} \cdot y$. By Remark (14) following the proof of the Leery Theorem this weaker distributivity property is sufficient for the Equations to hold.

Ad (18). By Lemma (9.1) $\downarrow_{\preceq} \in \mathsf{ACISM}(\leq_f)$. It is easily verified that $+\!\!\!+$ distributes backwards over $\downarrow_{\preceq}$. $\qquad\qquad\qquad\square$

There are now at least two ways to proceed in an attempt to improve the efficiency already provided by the Leery Theorem. One way leads to the Greedy (and Modest) algorithm; it is discussed in the next section. Another way leads to Van der Woude's algorithm (which we have called Gluttonous); it is discussed thereafter.

5   The Greedy Theorem

Recall the second Leery Equation for $S(\downarrow_f)$:

$$S(\downarrow_f) \cdot x \;=\; \downarrow_f/ \cdot (\tau \parallel_{+\!\!\!+} S(\downarrow_f))* \cdot p\triangleleft_1 \cdot Inv(+\!\!\!+') \cdot x \qquad \text{if } x \neq [] \quad .$$

In order to improve the efficiency of the right-hand side, we *try and promote* $\downarrow_f/$ to the right, just through $(\tau \parallel_{+\!\!\!+} S(\downarrow_f))*$, so that the presumably costly $S(\downarrow_f)$-computation is invoked only once. That is, we will

replace $\quad \downarrow_f/ \cdot (\tau \parallel_{+\!\!\!+} S(\downarrow_f))* \quad$ by $\quad \tau \parallel_{+\!\!\!+} S(\downarrow_f) \cdot \downarrow_g/$

where as usual $\downarrow_g \in \mathsf{ACISM}(\preceq_g)$ for some suitable g and $\preceq$. The arguments to $\downarrow_g$ will be pairs (y, z) satisfying $y +\!\!\!+ z = x$. The obvious choices for g are therefore $\# \cdot \pi_1$ and $\# \cdot \pi_2$, taking the normal $\leq$ for $\preceq$. So not only the goal of the current derivation is described

by the wish for a single (in this case simple) promotion, but also the resulting expression is suggested as well.

When worked out, the choice $g := \# \cdot \pi_1$ will give the equation $S(\downarrow_f) = \tau*$; see Fokkinga[7, the Modest Theorem]. This is too poor a result to be discussed here. So let us take $g := \# \cdot \pi_2$. In the current context we have that $\mathsf{ACISM}(\leq_{\# \cdot \pi_2})$ equals $\mathsf{ACISM}(\geq_{\# \cdot \pi_1})$ so that we may write $\uparrow_{\# \cdot \pi_1}$ instead of $\downarrow_{\# \cdot \pi_2}$.

(19) Bird's Greedy Theorem   *Suppose the Leery Equations are valid for $S(\downarrow_f)$ and, in addition, suppose $\leq_f$ is weakly prefix-stable and greedy and p is suffix-closed (conditions that arise in the proof, see below). Then*

$$
\begin{aligned}
S(\downarrow_f) \cdot [] \;\; &= \;\; [] \\
S(\downarrow_f) \cdot x \;\; &= \;\; \tau \parallel_{\#} S(\downarrow_f) \cdot \uparrow_{\# \cdot \pi_1} / \cdot p \triangleleft_1 \cdot Inv(\#') \cdot x \qquad \text{if } x \neq [] \quad .
\end{aligned}
$$

We refer to these equations as the Greedy Equations for $S(\downarrow_f)$.

Proof   Abbreviate $S(\downarrow_f)$ to S. In order to prove the second Greedy Equation from the Leery one, we have to show:

$$
\downarrow_f / \cdot (\tau \parallel_{\#} S)* \cdot r \;=\; \tau \parallel_{\#} S \cdot \uparrow_{\# \cdot \pi_1} / \cdot r
$$

for all $r = p \triangleleft_1 \cdot Inv(\#') \cdot x_0$ (with $x_0 \neq []$). By the Promotion Theorem this follows from $\tau \parallel_{\#} S$ being $\uparrow_{\# \cdot \pi_1} \to \downarrow_f$ distributive **on** r. (There is no requirement as regards to the identities of $\uparrow_{\# \cdot \pi_1}$ and $\downarrow_f$, since $r \neq \emptyset$.)

To prove the distributivity, observe that any two elements from $r \subseteq Inv(\#') \cdot x_0$ can be written as $(x, y \# z)$ and $(x \# y, z)$. Now,

$$
\begin{aligned}
&\quad\;\; \text{required distributivity} \\
&\equiv \qquad \text{unfolding} \\
&\quad\;\; (\tau \parallel_{\#} S \cdot (x, y \# z)) \downarrow_f (\tau \parallel_{\#} S \cdot (x \# y, z)) \;= \\
&\quad\;\; \tau \parallel_{\#} S \cdot (x, y \# z) \uparrow_{\# \cdot \pi_1} (x \# y, z) \\
&\equiv \qquad \text{calculus (for } any \; \uparrow_{\# \cdot \pi_1} \in \mathsf{ACISM}(\geq_{\# \cdot \pi_1})\text{)} \\
&\quad\;\; [x] \# (S \cdot y \# z) \downarrow_f [x \# y] \# Sz \;=\; [x \# y] \# Sz \\
&\Leftarrow \qquad \text{minimality and selectivity of } \downarrow_f \\
&\quad\;\; [x \# y] \# Sz <_f [x] \# (S \cdot y \# z) \quad \vee \quad [x] \# (S \cdot y \# z) = [x \# y] \# Sz \\
&\equiv \qquad \text{calculus} \\
&\quad\;\; [x \# y] \# Sz <_f [x] \# (S \cdot y \# z) \quad \vee \quad y = [] \quad \vee \\
&\quad\;\; S \cdot y \# z = Sz = \text{ a right zero of } \# \\
&\equiv \qquad \text{thus far no right zero of } \# \text{ has been introduced} \\
&\quad\;\; [x \# y] \# Sz <_f [x] \# (S \cdot y \# z) \quad \vee \quad y = []
\end{aligned}
$$

This is exactly the condition at which Bird *starts* to prove the theorem. (A right zero $\omega = \downarrow_f / \emptyset$ of $\#$ would have popped up when p would not hold of all singletons; see Fokkinga[7].) We will not repeat the details of the proof; let it suffice to say that Bird's proof goes through if p is suffix-closed and $\leq_f$ is weakly prefix-stable and greedy. These notions are defined thus (weak greediness is for later use only):

- p is *suffix-closed* if $p \cdot x \# y \Rightarrow py$.

- $\leq_f$ is *greedy* if
    1. $[x \mathbin{+\!\!+} y] \mathbin{+\!\!+} u \; <_f \; [x] \mathbin{+\!\!+} [y] \mathbin{+\!\!+} u$
    2. $[x \mathbin{+\!\!+} y] \mathbin{+\!\!+} [z] \mathbin{+\!\!+} u \; <_f \; [x] \mathbin{+\!\!+} [y \mathbin{+\!\!+} z] \mathbin{+\!\!+} u \quad$ for $y \neq []$.
- f is *weakly greedy* if
    1. $[x \mathbin{+\!\!+} y] \mathbin{+\!\!+} u \; \leq_f \; [x] \mathbin{+\!\!+} [y] \mathbin{+\!\!+} u$
    2. $[x \mathbin{+\!\!+} y] \mathbin{+\!\!+} [z] \mathbin{+\!\!+} u \; \leq_f \; [x] \mathbin{+\!\!+} [y \mathbin{+\!\!+} z] \mathbin{+\!\!+} u$.

Actually, the definitions may be weakened by requiring that the arguments to $<_f$ and $\leq_f$ are partitions of some sequence. $\qquad\qquad\square$

Using Bird's Greedy Theorem

Again there is the objection to the theorem that $\downarrow_f$ is not given, but is nevertheless mentioned (implicitly) in the conditions. This is even more unsatisfactory, as $\downarrow_f$ does not occur any more in the resulting expression for $S(\downarrow_f)$. Bird[5, Lemma 2] gives a construction of a greedy refinement out of any weakly greedy $\leq_f$, but unfortunately the construction does not preserve weak or real prefix stability. There are, again, two ways out.

Theorems

(20) *Suppose $\leq_f$ is prefix-stable and greedy, and p holds of all singletons and is suffix-closed. Define a pre-order $\preceq$ by*

$$v \preceq w \;\equiv\; v <_f w (v =_f w \wedge v \sqsubseteq_{\#*} w)$$

where $\sqsubseteq$ is the lexicographic order on sequences of numbers. Let $g = id$, and $\downarrow_g$ be arbitrary in $\mathsf{ACISM}(\preceq_g)$. Then $\downarrow_g \in \mathsf{ACISM}(\leq_f)$ and the Greedy Equations are valid for $S(\downarrow_g)$.

(21) *From De Moor[21]. Suppose $f = \#$ and $\leq$ is the usual order on numbers, and p is suffix-closed and holds of all singletons. Then the Greedy Equations are valid for $S(\downarrow_f)$, for some $\downarrow_f$ (which does not occur any more in the Equations!).*

Proofs

Ad (20). From the prefix stability of $\leq_f$ and the fact that, for any y, $\sqsubseteq_{\#*}$ is a linear order on $(\mathsf{all}\,p) \triangleleft \cdot \mathsf{parts} \cdot y$, one can conclude that $u \mathbin{+\!\!+} (v \downarrow_g w) = (u \mathbin{+\!\!+} v) \downarrow_g (u \mathbin{+\!\!+} w)$ for all $u \mathbin{+\!\!+} v,\, u \mathbin{+\!\!+} w \in \mathsf{parts}\, y$, and so the Leery Equations are valid for $S(\downarrow_g)$. Further, $\preceq_g$ is prefix-stable (hence weakly prefix-stable) since $\leq_f$ and $\sqsubseteq_{\#*}$ are so, and $\preceq_g$ is greedy since $\leq_f$ is so. Hence by Bird's Greedy Theorem the Greedy Equations are valid for $S(\downarrow_g)$. Finally, by construction $\preceq_g$ refines $\leq_f$, so by the Refinement Lemma (10) $\downarrow_g \in \mathsf{ACISM}(\leq_f)$.

Ad (21). Let $\leq'$ be the lexicographic order on $\alpha*$ induced by some linear order on α. Let $\sqsupseteq$ be the lexicographic order on sequences of numbers induced by $\geq$. Define $\preceq$ by

$$
\begin{aligned}
v \preceq w \quad\equiv\quad & v <_f w \;\vee \\
& (v =_f w \wedge v \sqsupseteq_{\#*} w) \;\vee \\
& (v =_f w \wedge v =_{\#*} w \;\wedge v \leq'_{+\!\!/} w) \quad.
\end{aligned}
$$

By construction $\preceq$ is a linear order that refines $\leq_f$, so by Lemma (9.1) $\downarrow_\preceq \in \mathsf{ACISM}(\leq_f)$. It is easily verified that $\mathbin{+\!\!+}$ distributes backwards over $\downarrow_\preceq$ (the lexicographic order $\leq'$ is relevant here), hence by the Leery Theorem the Leery Equations are valid for $S(\downarrow_\preceq)$. Further we have

that $\preceq$ is prefix stable (since $\leq_f$ and $\sqsupset_{\#*}$ are so) and is by construction (in particular $\sqsupset_{\#*}$) greedy, so taking $g = id$ we find by the Greedy Theorem (with $f, \leq, p$ instantiated by $g, \preceq, p$) that the Greedy Equations are valid for $S(\downarrow_{\preceq})$. $\qquad\qquad\qquad\qquad\square$

Notice that in the proofs the need to talk about g ($= id$) disappears completely if the theorems are formulated in terms of an arbitrary pre-order $\preceq$ rather than in terms of f, $\leq$ and $\leq_f$; cf. De Moor[21].) Notice also that none of the auxiliary (pre-)orders mentioned in the proofs occurs in the Equations.

6   The Gluttonous Theorem

In this section we derive the algorithm that Van der Woude[23] has derived in the Eindhoven style for the case $f = \#$, with the challenge[22] to redo it fully in the Bird-Meertens style. Our proof of the Replacement Lemma borrows quite some ideas of his proof, although probably no one will recognize his reasoning in it. Van der Woude handles the indeterminacy by using state-changes characterised by pre- and postconditions. We will continue using under-specification (in the sense we have formalised it in the previous sections). Throughout the following discussion we abbreviate $S(\downarrow_f)$ by S.

Recall, again, the second equation of the Leery Theorem:

$$S \cdot x \;=\; \downarrow_f / \cdot (\tau \parallel_{+\!\!\!+} S)* \cdot p\triangleleft_1 \cdot Inv(+\!\!\!+') \cdot x \qquad \text{if } x \neq [] \quad .$$

In the previous section we have promoted $\downarrow_f /$ to the right. Now we are going to promote S. For the sake of simplicity of the resulting expressions, let us first rename x into $[a] +\!\!\!+ x$ and eliminate the implicit test for nonemptiness in $Inv(+\!\!\!+')$:

$$S \cdot [a] +\!\!\!+ x \;=\; \downarrow_f / \cdot (\tau \parallel_{+\!\!\!+} S)* \cdot p\triangleleft_1 \cdot ([a] +\!\!\!+ \parallel id)* \cdot Inv(+\!\!\!+) \cdot x \quad .$$

The goal of the current approach can be described in the following two ways:

- Promote the recursive call of S to the right, through $p\triangleleft_1$ and $([a] +\!\!\!+ \parallel id)*$ (which is easy), and even through $Inv(+\!\!\!+)$ so that $S \cdot [a] +\!\!\!+ x$ is expressed in terms of a and Sx, and we have obtained a right reduce for S.

- Try to replace $Inv(+\!\!\!+) \cdot x$ by $(+\!\!\!+ / \parallel +\!\!\!+ /)* \cdot Inv(+\!\!\!+) \cdot S x$, meaning that *not all* prefixes of x have to be inspected in search of the outcome of the final $\downarrow_f /$, *but only* those prefixes that are a flattened prefix of the already optimal partition $S x$ of x. This might improve the efficiency of the algorithm drastically, as $(+\!\!\!+ / \parallel +\!\!\!+ /)* \cdot Inv(+\!\!\!+) \cdot S x$ yields a much smaller set that $Inv(+\!\!\!+) \cdot x$.

Van der Woude gives his motivation in the latter wording (he is really after an optimally efficient algorithm). Now that I have squiggolized his arguments and programs, I definitely prefer the former approach. Let us see how the wish to promote also suggests the proper replacement:

$\qquad S \cdot [a] +\!\!\!+ x$
$=\qquad$ above equation, i.e., Leery Theorem
$\qquad \downarrow_f / \cdot (\tau \parallel_{+\!\!\!+} S)* \cdot p\triangleleft_1 \cdot ([a] +\!\!\!+ \parallel id)* \cdot Inv(+\!\!\!+) \cdot x$
$=\qquad$ **goal**: promote S to the right;

calculus

$$\downarrow_f/ \cdot (\tau \parallel_{+\!\!\!+} id)* \cdot p\lhd_1 \cdot ([a] +\!\!\!+ \parallel id)* \cdot (id \parallel S)* \cdot Inv(+\!\!\!+) \cdot x$$

$=$ **goal**: promote S even further;

        assume: g **satisfies** $(\star)$ **(see below)**

$$\downarrow_f/ \cdot (\tau \parallel_{+\!\!\!+} id)* \cdot p\lhd_1 \cdot ([a] +\!\!\!+ \parallel id)* \cdot g \cdot S \cdot x$$

as desired, for now $S \cdot [a] +\!\!\!+ x$ has been expressed in terms of a and Sx. The property assumed of g reads

$(\star)$ $(id \parallel S)* \cdot Inv(+\!\!\!+) \;=\; g \cdot S$.

A trivial choice for g is "left hand side" $\cdot\, S^{-1}$ (note that $S^{-1} = +\!\!\!+/$), but this is not of great help. For a non-trivial g, type considerations suggest to try and choose $g = (+\!\!\!+/ \parallel id)* \cdot Inv(+\!\!\!+)$, or slightly more careful: $g = (+\!\!\!+/ \parallel (S \cdot +\!\!\!+/))* \cdot Inv(+\!\!\!+)$. (In the left-hand side of $(\star)$ the results have second components that are legal partitions, hence the suggestion for $S \cdot +\!\!\!+/$ in g.) With the latter choice we can "promote" S backwards to its original place; what remains is the suggestion to try and replace $Inv(+\!\!\!+)$ by $(+\!\!\!+/ \parallel +\!\!\!+/)* \cdot Inv(+\!\!\!+) \cdot S$. Q.E.D.

As it happens, this suggested replacement does not preserve $=$ but only $=_f$. The Replacement Lemma (23) presents the formalisation; it needs the following simple auxiliary result (which also has $=_f$ instead of $=$).

(22) Lemma   *Suppose $\leq_f$ is prefix-stable and p holds of all singletons. Abbreviate $S(\downarrow_f)$ to S. Then for all x*

$$id \parallel (S \cdot +\!\!\!+/) \;=_f\; id \quad \textbf{on the domain } Inv(+\!\!\!+)\, S\, x$$

i.e., for $v \in \mathsf{tails}\, S\, x$, $S \cdot +\!\!\!+/ \cdot v =_f v$.

Proof   Let x be arbitrary and $(u, v) \in Inv(+\!\!\!+)\, S\, x$. Then, writing $S +\!\!\!+ /\, v$ for $S \cdot +\!\!\!+/ \cdot v$,

     $(id \parallel (S \cdot +\!\!\!+/)) \cdot (u, v) \;=_f\; (u, v)$

$\equiv$ calculus

     $S +\!\!\!+ /\, v \;=_f\; v$

$\equiv$ pre-order property

     $v \leq_f S +\!\!\!+ /\, v \quad \wedge \quad S +\!\!\!+ /\, v \leq_f v$

$\equiv$ Least-f Characterisation (8) applied to the right conjunct

        noticing that $v \in (\mathsf{all}\, p)\lhd \cdot \mathsf{parts} \cdot +\!\!\!+/ \cdot v$ (from the defn of v and S)

     $v \;\leq_f\; S +\!\!\!+ /\, v$

$\equiv$ prefix stability of $\leq_f$

     $u +\!\!\!+ v \;\leq_f\; u +\!\!\!+ (S +\!\!\!+ /\, v)$

$\equiv$ assumption $u +\!\!\!+ v = Sx$

     $Sx \;\leq_f\; u +\!\!\!+ (S +\!\!\!+ /\, v)$

$\Leftarrow$ Lemma (8)

     $u +\!\!\!+ (S +\!\!\!+ /\, v) \;\in\; (\mathsf{all}\, p)\lhd \cdot \mathsf{parts} \cdot x$

$\equiv$ assumption $u +\!\!\!+ v = Sx$, properties of S and parts

true   .

<div align="right">□</div>

(23) Replacement Lemma (Van der Woude)   *Suppose $\leq_f$ is gluttonous (to be defined in the proof), and p is prefix-closed and true of all singletons. Then*

$$F \cdot Inv(\+) \ =_f \ F \cdot (\+/ \parallel \+/)* \cdot Inv(\+) \cdot S$$

where $F = \downarrow_f/ \cdot (\tau \parallel_\+ S)* \cdot p\vartriangleleft_1 \cdot ([a] \+ \parallel id)* .$

Proof   We show both $\leq_f$ and $\geq_f$ by establishing

$$\forall u \in \text{argument to } \downarrow_f/ \text{ in the rhs:} \quad \exists v \in \text{argument to } \downarrow_f/ \text{ in the lhs:} \quad v \leq_f u$$

which is easy, and

$$\forall u \in \text{argument to } \downarrow_f/ \text{ in the lhs:} \quad \exists v \in \text{argument to } \downarrow_f/ \text{ in the rhs:} \quad v \leq_f u$$

which is quite complicated. The details are given in the Appendix. The properties that we need to assume of $\leq_f$ and p turn out to be

- $\leq_f$ is *gluttonous*, meaning that
    1. $u <_\# v \ \Rightarrow \ u <_f v$
    2. $u =_f v \ \wedge \ x <_\# y \ \Rightarrow \ [x] \+ u <_f [y] \+ v$.

    (Actually these conditions may be weakened by requiring the extra premiss $u =_{\+/} v$, respectively $[x] \+ u \ =_{\+/} \ [y] \+ v$.)
- p is *prefix-closed*, meaning that $p \cdot x \+ y \Rightarrow px$.

<div align="right">□</div>

(24) The Gluttonous Theorem   *Suppose $\leq_f$ is prefix-stable and gluttonous, and $\+$ distributes backwards over $\downarrow_f$. Let p be prefix-closed and true on singletons. Define T as a right reduce with seed $[]$:*

$$
\begin{aligned}
T \quad &= \quad \oplus \not{+}_{[]} \\
a \oplus u \quad &= \quad \tau \parallel_\+ id \cdot \uparrow_{\# \cdot \pi_1}/ \cdot p\vartriangleleft_1 \cdot (([a] \+ \cdot \+ /) \parallel id)* \cdot Inv(\+) \cdot u \quad .
\end{aligned}
$$

Then

$$T \ =_f \ S(\downarrow_f) \quad .$$

Proof   Abbreviate $S(\downarrow_f)$ to S. By induction on the cons-structure of x we show $T x =_f S x$. For $x = []$ the claim is trivially true: $T[] = [] = S[]$. For $x = [a] \+ x_0$ we calculate as follows (note the three $=_f$-steps):

$$
\begin{aligned}
&S \cdot [a] \+ x_0 \\
=\quad &\text{Leery Theorem} \\
&\text{(assumed: } \+ \text{ distributes backwards over } \downarrow_f)
\end{aligned}
$$

<div align="center">15</div>

$$\downarrow_f / \cdot (\tau \parallel_{+\!\!+} S)* \cdot p\triangleleft_1 \cdot ([a] +\!\!+ \parallel id)* \cdot Inv(+\!\!+) \cdot x_0$$

$=_f \quad$ Replacement Lemma;

$\qquad$ (assumed: $\leq_f$ is gluttonous and p is prefix-closed and true of singletons)

$$\downarrow_f / \cdot (\tau \parallel_{+\!\!+} S)* \cdot p\triangleleft_1 \cdot ([a] +\!\!+ \parallel id)* \cdot (+\!\!+/ \parallel +\!\!+/)* \cdot Inv(+\!\!+) \cdot S \cdot x_0$$

$= \quad$ calculus (promoting S to the right and combining two consecutive $\parallel$'s)

$$\downarrow_f / \cdot (\tau \parallel_{+\!\!+} id)* \cdot p\triangleleft_1 \cdot (([a] +\!\!+ \cdot +\!\!+ /) \parallel (S \cdot +\!\!+/))* \cdot Inv(+\!\!+) \cdot S \cdot x_0$$

$=_f \quad$ Lemma (22) (assumed: $\leq_f$ is prefix-stable)

$$\downarrow_f / \cdot (\tau \parallel_{+\!\!+} id)* \cdot p\triangleleft_1 \cdot (([a] +\!\!+ \cdot +\!\!+ /) \parallel id)* \cdot Inv(+\!\!+) \cdot S \cdot x_0$$

$= \quad$ Promotion Theorem (for non-empty arguments);

$\qquad$ **proved below:** $(\tau \parallel_{+\!\!+} id)$ **is** $\uparrow_{\#\cdot\pi_1} \rightarrow \downarrow_f$ **distributive**

$$\tau \parallel_{+\!\!+} id \cdot \uparrow_{\#\cdot\pi_1}/ \cdot p\triangleleft_1 \cdot (([a] +\!\!+ \cdot +\!\!+ /) \parallel id)* \cdot Inv(+\!\!+) \cdot S \cdot x_0$$

$= \quad$ folding

$$a \oplus S x_0$$

$=_f \quad$ induction hypothesis;

$\qquad$ **proved below:** $u, v \in \mathsf{parts}\, x_0 \ \wedge\ u =_f v \quad \Rightarrow \quad a \oplus u =_f a \oplus v$

$$a \oplus T\, x_0$$

$= \quad$ definition T

$$T \cdot [a] +\!\!+ x_0$$

as desired. There is still two proof obligations. We discuss these in turn.

For the desired distributivity we proceed exactly analogous to the proof of the Greedy Theorem; (notice the difference between the two distributivities). First observe that, indeed, the special case $\downarrow_f / \cdot (\tau \parallel_{+\!\!+} id)* \cdot \emptyset \ =\ (\tau \parallel_{+\!\!+} id) \cdot \uparrow_{\#\cdot\pi_1}/ \cdot \emptyset$ need not be considered (as p is supposed to hold for singletons and $Inv(+\!\!+)$ does not yield $\emptyset$). Next, any two elements from $p\triangleleft_1 \cdot (([a] +\!\!+ \cdot +\!\!+ /) \parallel id)* \cdot Inv(+\!\!+) \cdot S \cdot x_0$ can be written as $(([a] +\!\!+ \cdot +\!\!+ / \cdot u), v +\!\!+ w)$ respectively $(([a] +\!\!+ \cdot +\!\!+ / \cdot u +\!\!+ v), w)$. So

$\qquad$ required distributivity

$\equiv \quad$ unfolding

$\qquad ([[a] +\!\!+ \cdot +\!\!+ / \cdot u] +\!\!+ v +\!\!+ w) \downarrow_f ([[a] +\!\!+ \cdot +\!\!+ / \cdot u +\!\!+ v] +\!\!+ w) \quad =$

$\qquad (\tau \parallel_{+\!\!+} id) \cdot ([a] +\!\!+ \cdot +\!\!+ / \cdot u, v +\!\!+ w) \uparrow_{\#\cdot\pi_1} ([a] +\!\!+ \cdot +\!\!+ / \cdot u +\!\!+ v, w)$

$\equiv \quad$ properties of $\#$, selectivity and maximality of $\uparrow_{\#\cdot\pi_1}$

$\qquad ([[a] +\!\!+ \cdot +\!\!+ / \cdot u] +\!\!+ v +\!\!+ w) \downarrow_f ([[a] +\!\!+ \cdot +\!\!+ / \cdot u +\!\!+ v] +\!\!+ w) \quad =$

$\qquad [[a] +\!\!+ \cdot +\!\!+ / \cdot u +\!\!+ v] +\!\!+ w$

$\Leftarrow \quad$ selectivity and minimality of $\downarrow_f$

$\qquad [[a] +\!\!+ \cdot +\!\!+ / \cdot u] +\!\!+ v +\!\!+ w >_f [[a] +\!\!+ \cdot +\!\!+ / \cdot u +\!\!+ v] +\!\!+ w \quad \vee \quad v = []$

$\Leftarrow \quad$ gluttonousness of $\leq_f$ (the first greedy condition already suffices)

$\qquad$ true .

The second proof obligation is to show

$$u, v \in \mathsf{parts}\, x_0 \ \wedge\ u =_f v \quad \Rightarrow \quad a \oplus u =_f a \oplus v \quad .$$

This follows from the slightly more general claim:

$(\star)$ $u, v \in \mathsf{parts}\, x_0 \;\wedge\; u =_f v \quad \Rightarrow \quad x{\otimes}u =_f x{\otimes}v$

where $[a]\otimes$ is a generalisation of $a\oplus$:

$$x{\otimes}u = \tau \parallel_{+\!\!\!+} id \cdot \uparrow_{\#\cdot\pi_1} / \cdot p\lhd_1 \cdot ((x +\!\!\!+ \cdot +\!\!\!+ /) \parallel id)* \cdot Inv(+\!\!\!+) \cdot u \quad .$$

First we observe

(a) $u =_f v \qquad\qquad\qquad \Rightarrow \quad u =_\# v$

(b) $[x] +\!\!\!+ u =_f [y] +\!\!\!+ v \quad \Rightarrow \quad x =_\# y \wedge u =_f v \quad .$

This is immediate from the gluttonousness of $\leq_f$. Now we prove $(\star)$ by induction on $\#u$. For $\#u = 0$ we have

$\quad\quad x \otimes u =_f x \otimes v$
$\equiv \qquad$ 2nd premiss of $(\star)$, observation (a), $\#u = 0$
$\quad\quad x \otimes [\,] =_f x \otimes [\,]$
$\equiv \qquad$ Leibniz
$\quad\quad$ true   .

For $\#u > 0$ we have, by observations (a) and (b) and premiss $u, v \in \mathsf{parts}\, x_0$, that for some y, u', v'

$\quad\quad u \quad = \quad [y] +\!\!\!+ u'$
$\quad\quad v \quad = \quad [y] +\!\!\!+ v'$
$\quad\quad y \quad \neq \quad [\,]$
$\quad\quad u' \quad =_f \quad v' \quad .$

Now we calculate

$\quad\quad x \otimes u \;=_f\; x \otimes v$
$\equiv \qquad$ unfolding u, v, $\otimes$
$\quad\quad \tau \parallel_{+\!\!\!+} id \cdot \uparrow_{\#\cdot\pi_1} / \cdot p\lhd_1 \cdot ((x +\!\!\!+ \cdot +\!\!\!+ /) \parallel id)* \cdot Inv(+\!\!\!+) \cdot [y] +\!\!\!+ u' \quad =_f$
$\quad\quad \tau \parallel_{+\!\!\!+} id \cdot \uparrow_{\#\cdot\pi_1} / \cdot p\lhd_1 \cdot ((x +\!\!\!+ \cdot +\!\!\!+ /) \parallel id)* \cdot Inv(+\!\!\!+) \cdot [y] +\!\!\!+ v'$
$\equiv \qquad$ equation for $Inv(+\!\!\!+)$;
$\qquad\qquad$ abbreviating the part up to the filter by dots;
$\qquad\qquad$ calculus
$\quad\quad \ldots\ldots \cdot \{(x, [y] +\!\!\!+ u')\} \cup (((x +\!\!\!+ y +\!\!\!+ \cdot +\!\!\!+ /) \parallel id)* \cdot Inv(+\!\!\!+) \cdot u') \quad =_f$
$\quad\quad \ldots\ldots \cdot \{(x, [y] +\!\!\!+ v')\} \cup (((x +\!\!\!+ y +\!\!\!+ \cdot +\!\!\!+ /) \parallel id)* \cdot Inv(+\!\!\!+) \cdot v')$
$\Leftarrow \qquad$ calculus, $y \neq [\,]$;
$\qquad\qquad$ folding into $\otimes$
$\quad\quad [x] +\!\!\!+ [y] +\!\!\!+ u' =_f [x] +\!\!\!+ [y] +\!\!\!+ v' \;\wedge\; (x +\!\!\!+ y) \otimes u' =_f (x +\!\!\!+ y) \otimes v'$
$\equiv \qquad$ induction hypothesis
$\quad\quad [x] +\!\!\!+ [y] +\!\!\!+ u' =_f [x] +\!\!\!+ [y] +\!\!\!+ v'$
$\Leftarrow \qquad$ weak prefix stability of $\leq_f$

$$u' =_f v'$$

$$\equiv \qquad \text{observed above}$$

$$\text{true} \quad .$$

$\square$

For us, it came as a surprise that the various equalities didn't hold true, and that we had to consider the weaker equivalence relations $=_f$. In retrospect it is obvious, and in view of the precision about the specification it is rather a surprise that in the Leery and Greedy Theorems there are real equalities $=$ and not equivalences $=_f$. After all, the particular function S was quite arbitrarily chosen.

Again, there is the objection to the theorem that it mentions $\downarrow_f$ in its conditions, whereas only f, $\leq$ and p are given in the specification.

Theorems

(25) *Suppose $\leq_f$ is prefix-stable and gluttonous, and p holds of all singletons and is prefix-closed. Then the T as defined in the Gluttonous Theorem (24) is a solution for $\mathcal{S}$ in Specification (3).*

(26) *Suppose $f = \#$ and p is prefix-closed and true of all singletons. Then the T as defined in the Gluttonous Theorem (24) is a solution for $\mathcal{S}$ in Specification (3).*

Proofs

Ad (25). Define a pre-order $\preceq$ by

$$v \preceq w \;\equiv\; v <_f w (v =_f w \wedge v \sqsubseteq_{\#*} w)$$

where $\sqsubseteq$ is the lexicographic order on sequences of numbers, and take $g = id$. Let $\downarrow_g$ be some member of $\mathsf{ACISM}(\preceq_g)$; the existence of such a $\downarrow_g$ is guaranteed by Lemma (9.3). As in the proof of Theorem (20) one can show that $+\!\!\!+$ distributes backwards over $\downarrow_g$, $\preceq_g$ is prefix-stable, and $\preceq_g$ is gluttonous. Hence by the Gluttonous Theorem $T \simeq_g S(\downarrow_g)$ (recall $\simeq_g$ is defined by $x \simeq_g y \equiv x \preceq_g y \wedge y \preceq_g x$). It follows that $T =_f S(\downarrow_g)$, and since $\preceq_g$ refines $\leq_f$ that $\downarrow_g \in \mathsf{ACISM}(\leq_f)$. So T is a solution for $\mathcal{S}$ in the Specification by the Alternative Specifications Lemma (4).

Ad (26). (Quite similar to the proof of Theorem (21)!) Let $\leq'$ be the lexicographic order on $\alpha*$ induced by some linear order on α. Let $\sqsubseteq$ be the lexicographic order on sequences of numbers. Define $\preceq$ by

$$
\begin{aligned}
v \preceq w \quad\equiv\quad & v <_f w \;\vee \\
& (v =_f w \wedge v \sqsubseteq_{\#*} w) \;\vee \\
& (v =_f w \wedge v =_{\#*} w \;\wedge v \leq'_{+\!\!\!+/} w) \quad.
\end{aligned}
$$

By construction $\preceq$ is a linear order that refines $\leq_f$, so by Lemma (9.1) $\downarrow_\preceq \in \mathsf{ACISM}(\leq_f)$. It is easily verified that $+\!\!\!+$ distributes backwards over $\downarrow_\preceq$ (the lexicographic order $\leq'$ is relevant here), hence by the Leery Theorem the Leery Equations are valid for $S(\downarrow_\preceq)$. Further we have that $\preceq$ inherits the prefix stability of $\leq_f$ and is by construction gluttonous, so taking $g = id$ we find by the Gluttonous Theorem (with $f, \leq, p$ instantiated by $g, \preceq, p$) that the $T =_f S(\downarrow_\preceq)$. Now the claim follows by the Alternative Specifications Lemma (4). $\square$

7 Conclusions

We have shown that the technique of underspecification may be employed succesfully in the derivation of algorithms in a purely functional setting. The "price" to be paid is the necessity of explicit quantifications on "the" underspecified function, and specifying the set of functions over which "the" underspecified function is intended to range. The explicit quantifications on $\downarrow_f$ have shown that Bird's[5] Leery and Greedy Theorem are not quite the kind of statements that one wants to have (but the defect has been repaired). The explicit quantifications have also prompted us to be aware of the fact that sometimes not just real equality ($=$) had to be shown, but only equality under f ($=_f$). This was quite important in our derivation of the Gluttonous Algorithm of Van der Woude[23].

We also like to mention that promotion (i.e., right shift of a term within an expression) has been the driving force in the derivation of the algorithms. A comparison with Bird[5] may be illuminating. Given the specification, Bird presents first the Greedy algorithm, then the Leery one (not indicating how they have been found), and finally proceeds to *verify* their correctness. In contrast with this we have *derived* first the Leery and then the Greedy algorithm in quite a natural way from the specification: promotion has been the driving force for the main derivation steps. A major cause of the difference is, we believe, the presence of bound variables in Bird's formulation of the algorithms: it is just too difficult to recognize that only a large scale promotion is the means to go from the one to the other.

The same can be said with respect to Van der Woude's derivation. Working in an traditional imperative setting, the concept of promotion can hardly be expressed, let alone that it can be a driving force in the derivation. The original incentive to this paper was the challenge to redo his derivation in the Bird-Meertens style. We found it rather surprising that we could provide a derivation where promotion is so clearly recognized as a driving force.

We have been able to avoid bound variables by using Inv and suitable function *combining forms*, notably the parallel combination. Thanks to this notation much of the algorithm derivation can be performed as an algebraic calculation.

References

[1] R.C. Backhouse. An exploration of the Bird-Meertens formalism. Technical Report CS8810, Dept of Math and Comp Sc, University of Groningen, 1988.

[2] R.C. Backhouse. Naturality of homomorphisms. In *International Summerschool on Constructive Algorithmics*, 1989. Held on Ameland, The Netherlands, September 1989.

[3] R.S. Bird. An Introduction to the Theory of Lists. In M. Broy, editor, *Logic of Programming and Calculi of Discrete Design*, pages 3–42. Springer Verlag, 1987. Also Technical Monograph PRG–56, Oxford University, Computing Laboratory, Programming Research Group, October 1986.

[4] R.S. Bird. Lecture notes on constructive functional programming. In M. Broy, editor, *Constructive Methods in Computing Science*. International Summer School directed by F.L. Bauer [et al.], Springer Verlag, 1989. NATO Advanced Science Institute Series (Series F: Computer and System Sciences Vol. 55).

[5] R.S. Bird. A calculus of functions for program derivation. In D.A. Turner, editor, *Research Topics in Functional Programming*, pages 287–308. Addison-Wesley, 1990. Also: Tech Report PRG–64 Oxford University (dec 1987).

[6] R.S. Bird, J. Gibbons, and G. Jones. Formal derivation of a pattern matching algorithm. *Science of Computer Programming*, 12:93–104, 1989.

[7] M.M. Fokkinga. Squiggolish derivations for some optimal partition algorithms — promotion as the driving force. CWI, Amsterdam. 25 pages, December 1989.

[8] M.M. Fokkinga. A BM style derivation of the lexico least upperbound of pattern e in subs x. CWI, Amsterdam, March 1990.

[9] M.M. Fokkinga. Homo- and catamorphisms, reductions and maps — an overview. CWI, Amsterdam, February 1990.

[10] J.T. Jeuring. Deriving algorithms on binary labelled trees. In P.G.M. Apers, D. Bosman, and J. van Leeuwen, editors, *Proceedings SION Computing Science in the Netherlands*, pages 229–249, 1989.

[11] J.T. Jeuring. Algorithms from theorems. In M. Broy and C.B. Jones, editors, *Programming concepts and Methods*, pages 247–266. North-Holland, 1990.

[12] G. Malcolm. An algebraic approach to infinite data structures. In *International Summer School on Constructive Algorithmics*, 1989. Held on Ameland, The Netherlands, September 1989.

[13] G. Malcolm. Factoring homomorphisms. In *International Summer School on Constructive Algorithmics*, 1989. Held on Ameland, The Netherlands, September 1989. Also Technical Report Computer Science Notes CS 8908, Dept of Math and Comp Sc, University of Groningen, 1989.

[14] G. Malcolm. Homomorphisms and promotability. In J.L.A. van de Snepscheut, editor, *Mathematics of Program Construction*, volume 375 of *Lect. Notes in Comp. Sc.*, pages 335–347. Springer Verlag, 1989.

[15] G. Malcolm. Squiggoling in context. *The Squiggolist*, 1(3):14th – 19th, 1990.

[16] L. Meertens. Algorithmics — towards programming as a mathematical activity. In J.W. de Bakker and J.C. van Vliet, editors, *Proceedings of the CWI Symposium on Mathematics and Computer Science*, pages 289–334. North-Holland, 1986.

[17] L. Meertens. Constructing a calculus of programs. In J.L.A. van de Snepscheut, editor, *Mathematics of Program Construction*, volume 375 of *Lect. Notes in Comp. Sc.*, pages 66–90. Springer Verlag, 1989.

[18] L. Meertens. Paramorphisms. *Formal Aspects of Computing*, 4(5):413–424, 1990.

[19] O. de Moor. Indeterminacy in optimization problems. In *International Summer School on Constructive Algorithmics*, 1989. Held on Ameland, The Netherlands, September 1989.

[20] O. de Moor. Inverses in program synthesis. In *International Summer School on Constructive Algorithmics*, 1989. Held on Ameland, The Netherlands, September 1989.

[21] O. de Moor. List partitions. In *International Summer School on Constructive Algorithmics*, 1989. Held on Ameland, The Netherlands, September 1989.

[22] J.C.S.P. van der Woude. Optimal segmentation. Technical Report Computing Science Notes 89/15, Eindhoven University of Technology, 1989.

[23] J.C.S.P. van der Woude. Private communication. Dutch Intercity Algorithmics Club, 1989.

A   Proof of the Replacement Lemma

Abbreviate $S(\downarrow_f)$ by S. The conclusion of Lemma (23) reads: for all x_0

$(\star)\quad F \cdot Inv(+\!\!+) \cdot x_0 \ =_f \ F \cdot (+\!\!+/ \parallel +\!\!+/)\ast \cdot Inv(+\!\!+) \cdot S \cdot x_0$

where $F = \downarrow_f/ \cdot (\tau \parallel_{+\!\!+} S)\ast \cdot p\triangleleft_1 \cdot ([a] +\!\!+ \parallel id)\ast$. First we show $\leq_f$ by establishing

$\forall u \in$ arg to $\downarrow_f/$ in the rhs of $(\star)$: $\quad \exists u' \in$ arg to $\downarrow_f/$ in the lhs of $(\star)$: $\quad u' \leq_f u \quad$.

To this end let u be such a value in the rhs, i.e., for some v, w

$$
\begin{aligned}
u \quad &= \quad [[a] +\!\!+ \cdot +\!\!+ / \cdot v] +\!\!+ (S \cdot +\!\!+ / \cdot w)\\
v +\!\!+ w \quad &= \quad S\,x_0 \quad .
\end{aligned}
$$

Now define $x = +\!\!+/\,v$, $y = +\!\!+/\,w$ and

$$ u' \ = \ [[a] +\!\!+ x] +\!\!+ Sy \quad . $$

Clearly $u = u'$, so $u' \leq_f u$. Further, $u' \in$ argument to $\downarrow_f/$ in the lhs if $x +\!\!+ y = x_0$; this is almost trivial:

$$
\begin{aligned}
&\quad x +\!\!+ y \ = \ x_0\\
\equiv \quad &\text{definition } x, y\\
&\quad (+\!\!+/\,v) +\!\!+ (+\!\!+/\,w) \ = \ x_0\\
\equiv \quad &\text{calculus}\\
&\quad +\!\!+/\,(v +\!\!+ w) \ = \ x_0\\
\equiv \quad &\text{introduction of } v \text{ and } w\\
&\quad +\!\!+/\,S\,x_0 \ = \ x_0\\
\equiv \quad &\text{Least-}f \text{ Characterisation (8)}\\
&\quad \text{true} \quad .
\end{aligned}
$$

Secondly, we need to show $\geq_f$ in $(\star)$. In case $x_0 = []$ this is easy. So assume $x_0 \neq []$. We will establish

$\forall u \in$ arg to $\downarrow_f/$ in the lhs of $(\star)$: $\quad \exists u' \in$ arg to $\downarrow_f/$ in the rhs of $(\star)$: $\quad u' \leq_f u \quad$.

Let u be such a value, i.e., for some x, y

(a) $\quad u \qquad\qquad = \quad [[a] +\!\!+ x] +\!\!+ Sy$

(b) $\quad x +\!\!+ y \qquad\, = \quad x_0$

(c) $\quad p \cdot [a] +\!\!+ x \quad = \quad \mathsf{true}$.

To construct u', observe that $S\,x_0$ has at least one nonempty segment in it (since $x_0 \neq [\,]$), and name the constituents of $S\,x_0$ by v, z, z', w such that

(d) $\quad v +\!\!+ [z +\!\!+ z'] +\!\!+ w \quad = \quad S\,x_0$

(e) $\quad x \qquad\qquad\qquad\; = \quad (+\!\!+/v) +\!\!+ z$

(f) $\quad y \qquad\qquad\qquad\; = \quad z' +\!\!+ (+\!\!+/w)$

(g) $\quad z +\!\!+ z' \qquad\qquad\;\; \neq \quad [\,]$.

Define

(h) $\quad u' \quad = \quad [[a] +\!\!+ (+\!\!+/v)] +\!\!+ (S \cdot +\!\!+/ \cdot [z +\!\!+ z'] +\!\!+ w)$.

It remains to show

(i) $u' \in$ argument to $\downarrow_f/$ in the rhs of $(\star)$,

(ii) $u' \leq_f u$.

Part (i) is evident, provided that $p \cdot [a] +\!\!+ (+\!\!+/v) = \mathsf{true}$. This follows from (e), (c) and prefix closedness of p. For part (ii) we argue as follows. First consider the case $z = [\,]$.

$\quad u' \leq_f u$

$\Leftarrow \qquad$ pre-order property

$\quad u' =_f u$

$\equiv \qquad$ unfoldings

$\quad [[a] +\!\!+ (+\!\!+/v)] +\!\!+ (S \cdot +\!\!+/ \cdot [z +\!\!+ z'] +\!\!+ w) =_f [[a] +\!\!+ (+\!\!+/v) +\!\!+ z] +\!\!+ (S \cdot z' +\!\!+ (+\!\!+/w))$

$\equiv \qquad$ case assumption $z = [\,]$

$\quad \mathsf{true}$.

It remains to prove (ii) for $z \neq [\,]$. As a stepping stone we first force $u' \leq_\# u$ by imposing a suitable condition on $\leq_f$ (the first gluttonous condition):

$\quad u' \leq_\# u$

$\equiv \qquad$ unfoldings

$\quad [[a] +\!\!+ \cdot +\!\!+ / \cdot v] +\!\!+ (S \cdot +\!\!+/ \cdot [z +\!\!+ z'] +\!\!+ w) \leq_\# [[a] +\!\!+ (+\!\!+/v) +\!\!+ z] +\!\!+ Sy$

$\Leftarrow \qquad$ arithmetic, property $\#$

$\quad [[a] +\!\!+ \cdot +\!\!+ / \cdot v] +\!\!+ (S \cdot +\!\!+/ \cdot [z +\!\!+ z'] +\!\!+ w) \leq_\# [[a] +\!\!+ \cdot +\!\!+ / \cdot v] +\!\!+ [z] +\!\!+ Sy$

$\Leftarrow \qquad$ **assume**: f gluttonous-1 (see below; in contraposition)

$\quad [[a] +\!\!+ \cdot +\!\!+ / \cdot v] +\!\!+ (S \cdot +\!\!+/ \cdot [z +\!\!+ z'] +\!\!+ w) \leq_f [[a] +\!\!+ \cdot +\!\!+ / \cdot v] +\!\!+ [z] +\!\!+ Sy$

$\Leftarrow \qquad$ weak prefix stability of $\leq_f$

$\quad S \cdot +\!\!+/ \cdot [z +\!\!+ z'] +\!\!+ w \leq_f [z] +\!\!+ Sy$

$\Leftarrow \qquad$ Least-f Characterisation (8), definition S

$$[z] \mathbin{+\!\!+} Sy \ \in \ (\text{all } p)\triangleleft \cdot \text{parts} \cdot z \mathbin{+\!\!+} z' \mathbin{+\!\!+} (\mathbin{+\!\!+}/w)$$

$\equiv$ equation (f), Least-f Characterisation, definition of S, $z \neq []$

$$p\,z \ = \ \text{true}$$

$\Leftarrow$ prefix closedness of p

$$p \cdot z \mathbin{+\!\!+} z' \ = \ \text{true}$$

$\Leftarrow$ Least-f Characterisation, definition S, (d)

$$\text{true} \quad .$$

Finally we can force $u' \leq_f u$ (even stronger: $u' <_f u$) by imposing another condition on $\leq_f$:

$$u' \ \leq_f \ u$$

$\Leftarrow$ pre-order property

$$u' \ <_f \ u$$

$\Leftarrow$ first gluttonous condition on $\leq_f$ (again)

$$u' <_\# u \ \vee \ (u' =_\# u \ \wedge \ u' <_f u)$$

$=$ predicate calculus

$$u' \leq_\# u \ \wedge \ (u' =_\# u \ \Rightarrow \ u' <_f u)$$

$=$ above calculation

$$u' =_\# u \ \Rightarrow \ u' <_f u$$

$\Leftarrow$ **assume**: $\leq_f$ gluttonous-2 (see below,

taking $u, v, x, y := u', u, \text{hd}\,u', \text{hd}\,u$), $z \neq []$

$$\text{true} \quad .$$

This completes the proof. The assumptions (called *gluttonousness* of $\leq_f$) read:

1. $u <_\# v \ \Rightarrow \ u <_f v$,

2. $u =_\# v \ \wedge \ x <_\# y \ \Rightarrow \ [x] \mathbin{+\!\!+} u <_f [y] \mathbin{+\!\!+} v$

and (1) may be weakened by requiring $u =_{+\!\!+/} v$ and (2) by requiring $[x] \mathbin{+\!\!+} u =_{+\!\!+/} [y] \mathbin{+\!\!+} v$.

B Notational Conventions and Summary of The Calculus

We use the dot $\cdot$ as a *separation mark* between subterms; this is easier for the eye than the use of parentheses. Thus $f\,g \cdot h\,i\,j \cdot k\,l = (f\,g)\,(h\,i\,j)\,(k\,l)$. (It is sometimes a matter of taste whether we write a dot or not, and most dots take the place of function composition.) Juxtaposition associates *to the right* and denotes both composition and application; in this paper this syntactic ambiguity does not lead to semantic ambiguity. For example, $f\,(g\,x) = f\,g\,x = f \cdot g \cdot x$. We use $\parallel$ and , as combinators between functions (called *parallel combination*, respectively *tupling*). These *combinators* have lower priority than any *operator*. The notational device of *sectioning* is used frequently: a binary operator provided with only one or even none operands is considered to be a function of the missing arguments. Thus

$$x \oplus y \ = \ x\oplus \cdot y \quad = \quad \oplus y \cdot x \ = \ \oplus \cdot (x, y)$$

$$
\begin{aligned}
(f, g) \cdot x &= (fx, gx) \\
f \parallel g \cdot (x, y) &= (fx, gy) \\
f \parallel_\oplus g \cdot (x, y) &= fx \oplus gy \\
&= \oplus \cdot f \parallel g \cdot (x, y) \quad .
\end{aligned}
$$

The Calculus

Here is a brief summary of the "standard" facts and concepts that we use (often tacitly).

Tree, sequence, bag, set Let α be a type (a set of values). Then the algebra that is initial for the signature

$$
\emptyset : \alpha\varpi, \quad \tau : \alpha \to \alpha\varpi, \quad \mathsf{join} : \alpha\varpi \parallel \alpha\varpi \to \alpha\varpi \quad ,
$$

where $\alpha\varpi$ denotes the carrier, is called a type of so-called *structures*. Constant $\emptyset$ is postulated to be the identity of join. Specific structures are obtained by postulating additional laws (equations). Without any further law, we call a structure a *tree*. When join is postulated to be associative, a structure is called a *sequence*, and we use the specific notation $\alpha*$, $[]$, $+\!\!+$, and $[x]$ for $\alpha\varpi$, $\emptyset$, join, and τx. When join is in addition postulated to be commutative, we call a structure a *bag*. When join is postulated to be idempotent as well, we call a structure a *set*, and we use the specific notation $\alpha\text{-set}$, $\cup$ for $\alpha\varpi$, join. The hierarchy tree - sequence - bag - set is known as the Boom hiearchy, after Hendrik Boom, who was the first to mention it explicitly.

Initiality is a formal concept; roughly said it means that any element of $\alpha\varpi$ can be denoted by a finite term built from the operations (and denotations for elements of α), and that any two terms denote different elements of $\alpha\varpi$ unless they can be proved equal by the laws that have been postulated. From the initiality property one can prove that functions may be defined "by induction on the structure"; see the equations for reduce, map and filter below. Also, it is the initiality property that forms the crux of the proof of the Promotion Theorem below.

Reduce, map, filter Suppose $\oplus : \alpha \parallel \alpha \to \alpha$, $f : \alpha \to \beta$ and $p : \alpha \to \mathsf{Bool}$ are given. Then there exist functions

$$
\begin{aligned}
\text{reduce} \quad &\oplus/ &: \alpha\varpi \to \alpha \\
\text{map} \quad &f* &: \alpha\varpi \to \beta\varpi \\
\text{filter} \quad &p\triangleleft &: \alpha\varpi \to \alpha\varpi
\end{aligned}
$$

satisfying the following equations (which constitute a definition by induction on the structure):

The Empty Rules

$$
\begin{aligned}
\oplus/ \cdot \emptyset &= 1_\oplus \quad \text{the identity of } \oplus \\
f* \cdot \emptyset &= \emptyset \\
p\triangleleft \cdot \emptyset &= \emptyset
\end{aligned}
$$

The One Point Rules

$$
\begin{aligned}
\oplus/ \cdot \tau x &= x \\
f* \cdot \tau x &= \tau \cdot fx \\
p\triangleleft \cdot \tau x &= \text{if } px \text{ then } \tau x \text{ else } \emptyset
\end{aligned}
$$

The Join Rules

$$\oplus/ \cdot x \,\mathsf{join}\, y \quad = \quad (\oplus/\,x) \oplus (\oplus/\,y)$$
$$f* \cdot x \,\mathsf{join}\, y \quad = \quad (f*x) \bowtie (f*y)$$
$$p\triangleleft \cdot x \,\mathsf{join}\, y \quad = \quad (p\triangleleft x) \bowtie (p\triangleleft y)$$

Whenever there occurs a reduce $\oplus/$ we shall take care that $\oplus$ satisfies at least the same laws as the join operation of its argument, for otherwise there would arise inconsistencies (since one can *derive* from the equations that $\oplus$ satisfies those laws).

Non-empty structures When we take $\emptyset$ out of the signature, and omit the Empty Rules, we get non-empty structures; in the case of sequences denoted by $\alpha+$. The advantage of doing so is that the identities $1_\oplus$ need not to exist.

Contextual information The qualified equality $f = g$ **on** D means $fx = gx$ for all x in D. In the calculations this is dealt with only informally. One may consult Malcolm[15] for an approach to do so formally.

Distribution, Promotion A function f is called $\oplus \to \otimes$ *distributive* (**on** D) if for all x, y (in D), $f \cdot x\oplus y = fx \otimes fy$. In case $\oplus$ is the same as $\otimes$ we say just "$\oplus$ distributive". When for all x function $x\odot$ is $\oplus$ distributive, we say $\odot$ *distributes backwards over* $\oplus$.

A function f is called $\oplus \to \otimes$ *promotable* (**on** D) if it is $\oplus \to \otimes$ distributive and, in addition, (D is empty or) $f\,1_\oplus = 1_\otimes$. A very important theorem is the so-called **Promotion Theorem**:

$$f \quad \text{is} \quad \oplus \to \otimes \text{ promotable} \qquad \Rightarrow \qquad f \cdot \oplus/ \ = \ \otimes/ \cdot f*$$
and
$$f \quad \text{is} \quad \oplus \to \otimes \text{ promotable on } D \quad \Rightarrow \quad f \cdot \oplus/ \cdot D \ = \ \otimes/ \cdot f* \cdot D \quad .$$

(The converse implications are easily derived by choosing suitable arguments.) The usefulness of the theorem is enhanced by the many, very many promotability properties that are immediate from the syntactic form of the expressions; in particular promotability chains nicely:

$$\left.\begin{array}{l} f \text{ is } \oplus \to \otimes \text{ promotable} \\ g \text{ is } \otimes \to \odot \text{ promotable} \end{array}\right\} \quad \Rightarrow \quad g \cdot f \text{ is } \oplus \to \odot \text{ promotable}$$

(and similarly for distributivity), and

$$\begin{array}{lll} f* & \text{is} & \mathsf{join} \to \mathsf{join} \text{ promotable} \\ p\triangleleft & \text{is} & \mathsf{join} \to \mathsf{join} \text{ promotable} \\ \oplus/ & \text{is} & \mathsf{join} \to \oplus \text{ promotable} \\ * & \text{is} & \circ \to \circ \text{ promotable} \quad . \end{array}$$

In the last assertion, $\circ$ denotes function composition; fully worked out the assertion gives the **Map Distributivity** and **Map Preserves Identity** law:

$$(f \cdot g)* \quad = \quad f* \cdot g*$$
$$id* \qquad = \quad id \quad .$$

Whenever we say "promote $<:\ someterm\ :>$ to the right" in a calculation step, the formal justification can be given in terms of these promotability properties.

Further Facts and Functions Function all is defined by $\mathsf{all}\,p = \wedge/\cdot p*$. Functions π_1 and π_2 select the left and right component of a tuple, e.g., $\pi_1\,(x,y) = x$. There are various straightforward laws that can be derived on the fly if the need arises; in particular so for the parallel combinator $\|$. Let us mention one that is less obvious:

$$(\mathsf{all}\,p)\triangleleft \cdot (\mathbin{+\!\!+} \cdot \tau) \,\|_*\, f \quad = \quad (\mathbin{+\!\!+} \cdot \tau) \,\|_*\, ((\mathsf{all}\,p)\triangleleft \cdot f) \cdot (p \cdot \pi_1)\triangleleft \quad .$$

Directed Reduce Consider the equations (for given e and $\oplus$ and unknown f)

$$
\begin{aligned}
f \cdot [] \quad &= \quad e \\
f \cdot [a] \mathbin{+\!\!+} x \quad &= \quad a \oplus fx \quad .
\end{aligned}
$$

It can be shown that these *define* a function, which we denote by $\oplus \mathbin{\not\!\!\leftarrow} e$ and which we call a *right reduce*. The advantage over a reduce is that there is more freedom for $\oplus$: operation $\oplus$ need not be associative, and may have type $\alpha \,\|\, \beta \to \beta$ (with $e \in \beta$). We omit a discussion of the laws that are valid for right reduces.

When implemented in an imperative program notation, $(\oplus \mathbin{\not\!\!\leftarrow} e)\,x$ reads:

$$
\begin{aligned}
&\|[\qquad \mathbf{var}\ f \\
&\bullet \qquad f := e; \\
&\qquad\qquad \mathbf{for}\ a\ \mathbf{from\ right\ to\ left\ over}\ x\ \mathbf{do} \\
&\qquad\qquad\qquad f := a \oplus f; \\
&\qquad\qquad \mathbf{result\ is}\ f \\
&\,]| \quad .
\end{aligned}
$$

Similarly we have left reduces.