Calculating (Haskell) programs from Galois connections

J.N. Oliveira [1] Shin-Cheng Mu [2]

[1]CCTC, University of Minho, Portugal

[2]IIS, Academia Sinica, Taiwan, Taiwan

HASLab Seminar Series

3rd October 2010

Motivation

Questions:

- Why is computer **programming** "hard" in general?
- Is there a **generic** skill, or competence, that one such acquire to become a "good programmer"?

Need for abstraction:

- **Technology** (mess) — don't fall in the trap: simply **abstract** from it!
- **Requirements** — again abstract from these as much as possible — write formal models or specs

One is led to formal models, or specifications. But, once again,

- What is it that makes the specification hard to fulfill?

Problems = Easy + Hard

Superlatives in problem statements, eg.

- *"... the smallest such number"*
- *"... the longest such list"*
- *"... the best approximation"*

suggest two layers in specifications:

- the **easy** layer — **broad** class of solutions (eg. a *prefix* of a list)
- the **difficult** layer — requires one **particular** such solution regarded as **optimal** in some sense (eg. "shortest with maximal density").

Example

Requirements for **whole division** $x \div y$:

- Write a program which computes number z which, multiplied by y, approximates x.
- Check your program with the following test data:
  $x, y, z = 7, 2, 1$
  $x, y, z = 7, 2, 2$
- Ups! Forgot to tell that I want the **largest** such number (sorry!):
  $x, y, z = 7, 2, 3$

Deriving the algorithm... from what?

... where is the formal specification of $x \div y$?

Example

Requirements for **whole division** $x \div y$:

- Write a program which computes number z which, multiplied by y, approximates x.
- Check your program with the following test data:
  $x, y, z = 7, 2, 1$
  $x, y, z = 7, 2, 2$
- Ups! Forgot to tell that I want the **largest** such number (sorry!):
  $x, y, z = 7, 2, 3$

Deriving the algorithm... from what?

... where is the formal specification of $x \div y$*?*

Example — writing a spec

First version (literal):

$$x \div y = \langle \bigvee z \ :: \ z \times y \leq x \rangle \tag{1}$$

Second version (involved):

$$z = x \div y \ \Leftrightarrow \ \langle \exists \ r \ : \ 0 \leq r < y : \ x = z \times y + r \rangle \tag{2}$$

Third version (clever!):

$$z \times y \leq x \ \Leftrightarrow \ z \leq x \div y \qquad (y > 0) \tag{3}$$

— a **Galois connection**.

Why (3) is better than (1,2)

It captures the requirements:

- It is a solution: $x \div y$ multiplied by y approximates x

$$(x \div y) \times y \leq x$$

(let $z := x \div y$ in (3) and simplify)

- It is the best solution because it provides the **largest** such number:

$$z \times y \leq x \;\Rightarrow\; z \leq x \div y \qquad (y > 0)$$

(the $\Rightarrow$ part of $\Leftrightarrow$).

Advantages:

"Generous" and highly calculational!

"Generosity" of GCs

"GCs as specs" offer the possibility of reasoning about the operation one whishes to implement prior to the actual implementation. For instance, the following facts about whole division stem directly from (3):

$$\begin{aligned} (n \div m) \div d &= n \div (d \times m) \\ n \div 1 &= n \\ n \div d \geq 1 &\Leftrightarrow d \leq n \end{aligned}$$

From another GC, specifying the *take* function on lists,

$$\mathsf{len}\ ys \leq n \ \wedge \ ys \sqsubseteq xs \quad \Leftrightarrow \quad ys \sqsubseteq take(n, xs) \tag{4}$$

($\sqsubseteq$ denoting prefix) one immediately infers properties such as eg.

$$take(n, take(m, xs)) = take(min(n, m), xs) \tag{5}$$

etc — see detials in Oliveira (2010).

Dissecting GCs

- Elsewhere, Silva and Oliveira (2008) follow the "GCs as specs" motto and show how to derive $x \div y$ from its defining GC.

- Today I would like to focus on a particular class of GCs in which the **easy+hard** split is particularly apparent.

- Such GCs will be handled in the relational pointfree style, eventually leading to specs elegantly captured by a binary combinator of shape

$$E \upharpoonright H$$

where E (=easy) provides the broad class of solutions and H (=hard) provides the criterion for optimizing E so as to obtain the "*superlative* effect".

$GC = E \upharpoonright H$

Let us PF-transform one of the several alternative definitions of a Galois connection $f \vdash g$ — Theorem 5.29 of (Aarts et al., 1992):

- f is monotonic
- $(f \cdot g)x \leq x$
- $(f\ x) \leq y \Rightarrow x \sqsubseteq (g\ y)$

Two bullets in a single line, in PF-notation:

$$f \cdot g \ \subseteq \ \leq \quad \wedge \quad f^\circ \cdot \leq \ \subseteq \ \sqsubseteq \cdot g$$

These are equivalent to (shunting, converses)

$$\underbrace{g \ \subseteq \ f^\circ \cdot \leq}_{\text{“easy”}} \quad \wedge \quad \underbrace{g \cdot (f^\circ \cdot \leq)^\circ \ \subseteq \ \sqsupseteq}_{\text{“hard”}}$$

We will use a special notation for this, to be explained shortly:

$$g \ \subseteq \ (f^\circ \cdot \leq) \upharpoonright \sqsupseteq \tag{6}$$

Intuition about $R \upharpoonright S$

Combinator $R \upharpoonright S$ (read: *"R optimized by S"*) was proposed by Ferreira and Oliveira (2010) for induction-less reasoning about **Alloy** sequences. Example of $R \upharpoonright S$ in **data-processing** ("choose the best mark"):

$$\left(\begin{array}{c|c} \textit{Mark} & \textit{Student} \\ \hline 10 & \textit{John} \\ 11 & \textit{Mary} \\ 12 & \textit{John} \\ 15 & \textit{Arthur} \end{array}\right) \upharpoonright \geq \quad = \quad \begin{array}{c|c} \textit{Mark} & \textit{Student} \\ \hline 11 & \textit{Mary} \\ 12 & \textit{John} \\ 15 & \textit{Arthur} \end{array}$$

Example of $R \upharpoonright S$ in **list-processing**: given a sequence $A \xleftarrow{S} \mathbb{N}$,

$$A \xleftarrow{nub\ S} \mathbb{N} \quad \triangleq \quad (S^{\circ} \upharpoonright \leq)^{\circ}$$

removes all duplicates while keeping the first instances. (Data in $\mathbb{N}$ could be regarded as "time stamps".)

$GC = E(easy) \upharpoonright H(ard)$

Thanks to the algebra of $R \upharpoonright S$ (to be discussed shortly) (6) can be strengthened to equality and therefore

$$g = (f^\circ \cdot \leq) \upharpoonright \sqsupseteq \tag{7}$$

This provides us with a method for calculating algorithmic versions of the adjoints of a GC, by exploiting the properties of the "shrink" combinator $\upharpoonright$. Example: GC

$$k \times b \leq a \;\Leftrightarrow\; k \leq a \div b \qquad (b > 0)$$

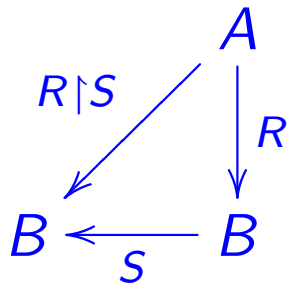
(whole division) PF-transforms into

$$\div b \;=\; ((\times b)^\circ \cdot \leq) \upharpoonright \geq \qquad (b > 0)$$

Definition of $R \upharpoonright S$

Given relation $B \xleftarrow{R} A$ and

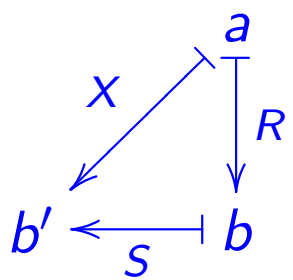
optimization criterion $B \xleftarrow{S} B$ on its outputs,



define $R \upharpoonright S$ by universal property:

$$X \subseteq R \upharpoonright S \quad \Leftrightarrow \quad X \subseteq R \ \wedge \ X \cdot R^{\circ} \subseteq S \tag{8}$$

This ensures $R \upharpoonright S$ as the largest sub-relation X of R such that, for all $b', b \in B$, if there exists $a \in A$ such that $b' X a \wedge b R a$, then $b' S b$ holds ("b' better than b").



Definition of $R \upharpoonright S$

The same in a closed formula,

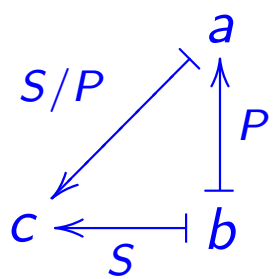
$$R \upharpoonright S \quad = \quad \underbrace{R}_{easy} \cap \underbrace{S/R^{\circ}}_{hard} \tag{9}$$

thanks to the GC of relational division (compare with **integer** division):

$$X \cdot R \subseteq S \quad \Leftrightarrow \quad X \subseteq S \,/\, R \tag{10}$$

With points:

$$c(S \,/\, P)a \;\Leftrightarrow\; \langle \forall\ b \;:\; a\ P\ b :\ c\ S\ b \rangle$$



Definition of $R \upharpoonright S$

Thus, $b'(R \upharpoonright S)a$ means

$$b' \ R \ a \ \wedge \ \langle \forall \ b \ : \ b \ R \ a : \ b' \ S \ b \rangle$$

Comments:

- Reasoning with such quantified expressions would mean "going one century back".
- Instead, we resort on the algebra of relational division — see eg. next slide.

Role of division ("hard" part)

From GC $X \cdot R \subseteq S \Leftrightarrow X \subseteq S / R$ infer:

- (Right) cancellation:

$$(S/R) \cdot R \subseteq S \tag{11}$$

- Upper-adjoint distribution:

$$(S \cap P)/R = (S/R) \cap (P/R) \tag{12}$$

- Lower-adjoint distribution:

$$(X \cup Y) \cdot R = X \cdot R \cup Y \cdot R \tag{13}$$

etc

Basic properties

Chaotic optimization:

$$R \upharpoonright \top \quad = \quad R \tag{14}$$

Impossible optimization:

$$R \upharpoonright \bot \quad = \quad \bot \tag{15}$$

Force determinism:

$$R \upharpoonright id \quad = \quad \text{largest deterministic fragment of } R \tag{16}$$

Basic properties

Function fusion (where R_f abbreviates $f^\circ \cdot R \cdot f$):

$$(R \restriction S) \cdot f = (R \cdot f) \restriction S \quad (17)$$

$$(f \cdot S) \restriction R = f \cdot (S \restriction R_f) \quad (18)$$

Ensure simplicity (determinism):

$$R \restriction S \text{ is simple} \Leftarrow S \text{ is anti-symmetric} \quad (19)$$

Deterministic (simple) = already optimized: for R simple,

$$R \restriction S = R \quad \Leftrightarrow \quad \text{img}\, R \subseteq S \quad (20)$$

Thus (functions)

$$f \restriction S = f \quad \Leftarrow \quad S \text{ is reflexive} \quad (21)$$

Basic properties

Union:

$$(R \cup S) \upharpoonright Q = (R \upharpoonright Q) \cap Q/S^{\circ} \cup (S \upharpoonright Q) \cap Q/R^{\circ} \quad (22)$$

This has a number of corollaries, namely conditionals:

$$(P \rightarrow R \ , \ T) \upharpoonright S \ = \ P \rightarrow (R \upharpoonright S) \ , \ (T \upharpoonright S) \quad (23)$$

Disjoint union:

$$[R, S] \upharpoonright U \ = \ [R \upharpoonright U, S \upharpoonright U] \quad (24)$$

where the *junc* operator

$$[R, S] \ \triangleq \ R \cdot i_1^{\circ} \ \cup \ S \cdot i_2^{\circ} \quad (25)$$

is associated to relational coproducts.

The "function competition" rule

Finally, a corollary of the union rule,

$$(f \cup g) \upharpoonright S \quad = \quad (f \cap S \cdot g) \cup (g \cap S \cdot f) \tag{26}$$

since $S/g^{\circ} = S \cdot g$. Comments:

- For S anti-symmetric, $(f \cup g) \upharpoonright S$ is always simple at the cost of not being entire.
- If furthermore one function (say g) "always wins" over the other with respect S — $(g\ x)S(f\ x)$ for all x — then $(f \cup g) \upharpoonright S = g$.

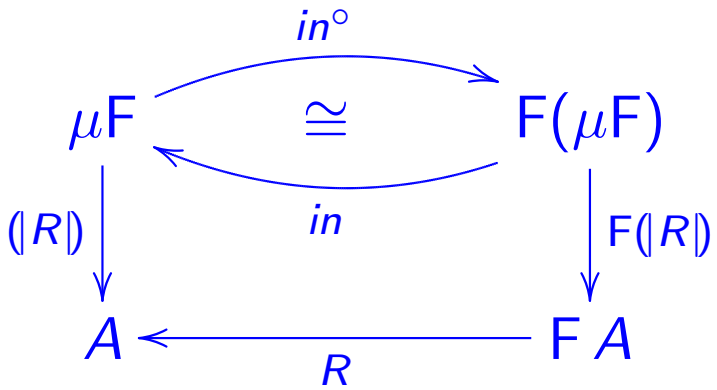
Optimizing inductive relations

Quite often, the orderings involved in optimization are **inductive** relations.

- Inductive orderings lead to recursive programs
- "Greedy algorithms" and "dynamic programming" studied in this way in the *Algebra of Programming* book (Bird and de Moor, 1997).
- Complexity of the approach puts many readers off (need for a tabular, power allegory; always transposing relations to powerset functions; ...)
- $R \upharpoonright S$ algebra **greatly simplifies** and generalizes the calculation of programs from such specifications. (Notably, there is no need for power transpose.)

Folds ($k\alpha\tau\alpha$s)

In general, for F a polynomial functor (relator) and initial $\mu\mathsf{F} \xleftarrow{in} \mathsf{F}(\mu\mathsf{F})$,



there is a unique solution to equation $X = R \cdot \mathsf{F}\, X \cdot in^\circ$ — thus universal property:

$$X = (\!|R|\!) \quad \Leftrightarrow \quad X = R \cdot \mathsf{F}\, X \cdot in^\circ \tag{27}$$

(Read $(\!|R|\!)$ as "fold R" or "$\kappa\alpha\tau\alpha$ R".)

Fold essentials

Therefore, by Knaster-Tarski: $(\!|R|\!)$ is both **the least** prefix point

$$(\!|R|\!) \subseteq X \quad \Leftarrow \quad R \cdot \mathsf{F}\, X \cdot in^{\circ} \subseteq X \tag{28}$$

and **the greatest** postfix point:

$$X \subseteq (\!|R|\!) \quad \Leftarrow \quad X \subseteq R \cdot \mathsf{F}\, X \cdot in^{\circ} \tag{29}$$

Corollaries include **reflexion**,

$$(\!|in|\!) \quad = \quad id \tag{30}$$

and two forms of $\kappa\alpha\tau\alpha$-**fusion**:

$$S \cdot (\!|R|\!) \subseteq (\!|T|\!) \quad \Leftarrow \quad S \cdot R \subseteq T \cdot \mathsf{F}\, S \tag{31}$$

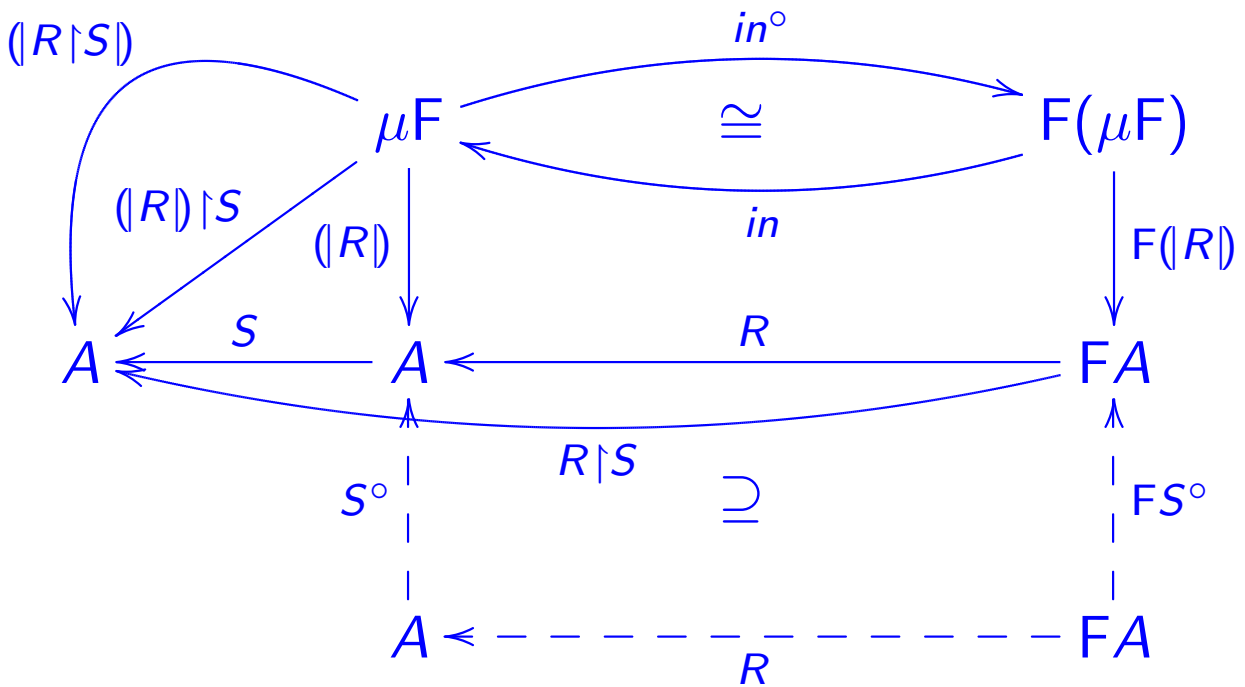
$$(\!|T|\!) \subseteq S \cdot (\!|R|\!) \quad \Leftarrow \quad T \cdot \mathsf{F}\, S \subseteq S \cdot R \tag{32}$$

"Greedy" theorem

Our version of theorem 7.2 by Bird and de Moor (1997):

$$(\!|R \upharpoonright S|\!) \subseteq (\!|R|\!) \upharpoonright S \quad \Leftarrow \quad S^\circ \xleftarrow{R} \mathsf{F}\, S^\circ \tag{33}$$

for S transitive. (**NB**: $R \xleftarrow{X} S$ means $X \cdot S \subseteq R \cdot X$) In a diagram, where the side condition is depicted in dashed arrows:



Calculational proof

$$
\begin{array}{ll}
 & (\!|R \upharpoonright S|\!) \subseteq (\!|R|\!) \upharpoonright S \\
\Leftrightarrow & \quad \{ \text{ universal property of } (\upharpoonright) \ (8) \ \} \\
 & (\!|R \upharpoonright S|\!) \subseteq (\!|R|\!) \ \wedge \ (\!|R \upharpoonright S|\!) \cdot (\!|R|\!)^{\circ} \subseteq S \\
\Leftrightarrow & \quad \{ \text{ monotonicity, since } X \upharpoonright Y \subseteq X \text{ in general } \} \\
 & (\!|R \upharpoonright S|\!) \cdot (\!|R|\!)^{\circ} \subseteq S \\
\Leftrightarrow & \quad \{ \text{ hylomorphisms: } (\!|S|\!) \cdot (\!|R|\!)^{\circ} = \langle \mu \ X \ :: \ S \cdot \mathsf{F}\, X \cdot R^{\circ} \rangle \ \} \\
 & \langle \mu \ X \ :: \ (R \upharpoonright S) \cdot \mathsf{F}\, X \cdot R^{\circ} \rangle \subseteq S \\
\Leftarrow & \quad \{ \text{ least (pre)fixpoint } \} \\
 & (R \upharpoonright S) \cdot \mathsf{F}\, S \cdot R^{\circ} \subseteq S
\end{array}
$$

Calculational proof (closing)

$$
\begin{array}{ll}
 & (R \upharpoonright S) \cdot \mathsf{F}\, S \cdot R^\circ \subseteq S \\
\Leftarrow & \quad \{ \text{ side-condition } S^\circ \xleftarrow{R} \mathsf{F}\, S^\circ \text{ ; converses ; monotonicity } \} \\
 & (R \upharpoonright S) \cdot R^\circ \cdot S \subseteq S \\
\Leftarrow & \quad \{ \text{ since } R \upharpoonright S \subseteq S/R^\circ \ \} \\
 & (S/R^\circ) \cdot R^\circ \cdot S \subseteq S \\
\Leftarrow & \quad \{ \text{ division cancellation (11) } \} \\
 & S \cdot S \subseteq S \\
\Leftarrow & \quad \{ \ S \text{ assumed transitive } \} \\
 & \textsc{True}
\end{array}
$$

(Re-worked from (Bird and de Moor, 1997).)

Example of greedy programming

The *mps* problem,

> $mps :: [Int] \rightarrow [Int]$
> $y\ mps\ x = y$ *is a prefix of* x *that yields the maximum sum*

which translates straight into

$$y\ mps\ x \quad \Rightarrow \quad y \sqsubseteq x\ \wedge\ \langle \forall\ z\ :\ z \sqsubseteq x :\ sum\ y \geq sum\ z \rangle$$

(where $\sqsubseteq = (\!|\ nil, cons \cup nil\ |\!)$ is the prefix ordering) which in turn PF-transforms into

$$mps \quad \subseteq \quad \sqsubseteq \upharpoonright \geq_{sum}$$

(**NB:** not a GC, this is nevertheless a good example to warm up.)

Example of greedy programming

We calculate:

$$
\begin{array}{ll}
 & mps \subseteq \sqsubseteq \upharpoonright \geq_{sum} \\
\Leftrightarrow & \quad \{ \text{ definition of prefix ordering } \} \\
 & mps \subseteq (\!| nil, cons \cup nil |\!) \upharpoonright \geq_{sum} \\
\Leftarrow & \quad \{ \text{ greedy theorem for folds } \} \\
 & mps \subseteq (\!| [nil, cons \cup nil] \upharpoonright \geq_{sum} |\!) \\
\Leftrightarrow & \quad \{ \text{ junc-rule (24) ; determinism of } nil \text{ } \} \\
 & mps \subseteq (\!| nil, (cons \cup nil) \upharpoonright \geq_{sum} |\!) \\
\Leftrightarrow & \quad \{ \text{ function competition rule (26) } \} \\
 & mps \subseteq (\!| nil, (cons \cap \geq_{sum} \cdot nil) \cup (nil \cap \geq_{sum} \cdot cons) |\!)
\end{array}
$$

(Side condition ignored for brevity.)

Example of greedy programming

Let R abbreviate $(nil \cap \geq_{sum} \cdot cons) \cup (cons \cap \geq_{sum} \cdot nil)$. Then $y\ R\ (a : x)$ will mean

$$y = [\,] \wedge 0 \geq a + sum\ x \ \vee \ y = a : x \wedge a + sum\ x \geq 0$$

The case $a + sum\ x = 0$ being ambiguous, we still have a relational fold. Thus we need to further shrink what we started from,

$$mps \quad = \quad (\sqsubseteq \upharpoonright \geq_{sum}) \upharpoonright \sqsubseteq \tag{34}$$

to obtain a function. In Haskell, the overall outcome will be:

```
mps [] = []
mps(a:s) = let x = mps s
           in if sum x > -a then a:x else []
```

Dynamic programming (DP) in the $E \upharpoonright H$ style

Variant of theorem 9.1 of (Bird and de Moor, 1997,page 220):

Theorem

Given

- *algebra* $R \xleftarrow{h} \mathsf{F}\, R$ *for transitive* R*:*
- *hylomorphism* $(\!|h|\!) \cdot (\!|T|\!)^\circ = \mu g$ *(for* $g\ X = h \cdot \mathsf{F}\, X \cdot T^\circ$*) where* T *is entire.*

Then

$$\langle \mu\ X\ ::\ (g\ X) \upharpoonright R \rangle \quad \subseteq \quad \mu g \upharpoonright R \tag{35}$$

holds, that is,

$$\langle \mu\ X\ ::\ (h \cdot \mathsf{F}\, X \cdot T^\circ) \upharpoonright R \rangle \quad \subseteq \quad (\!|h|\!) \cdot (\!|T|\!)^\circ \upharpoonright R \tag{36}$$

□

Corollary for unfolds

NB: Unfolds are converses of folds, of shape $(\!|R|\!)^\circ$.

Corollary (Unfold shrinking)

Given transitive R *such that* $R \xleftarrow{in} \mathsf{F}\, R$ *and entire* T,

$$\langle \mu\, X \;::\; (in \cdot \mathsf{F}\, X \cdot T^\circ) \upharpoonright R \rangle \quad \subseteq \quad (\!|T|\!)^\circ \upharpoonright R \tag{37}$$

□

Provides a way of interweaving optimization $(\upharpoonright R)$ with the unfold recursion path.

Calculation of DP theorem

$$\langle \mu\, X \;::\; (g\; X) \upharpoonright R \rangle \;\subseteq\; \mu g \upharpoonright R$$

$\Leftarrow$ { fixpoint rule }

$$(g(\mu g \upharpoonright R)) \upharpoonright R \;\subseteq\; \mu g \upharpoonright R$$

$\Leftrightarrow$ { universal property of "shrink" }

$$(g(\mu g \upharpoonright R)) \upharpoonright R \subseteq \mu g \;\wedge\; ((g(\mu g \upharpoonright R)) \upharpoonright R) \cdot (\mu g)^\circ \subseteq R$$

$\Leftarrow$ { $X \upharpoonright R \subseteq X$ (twice) }

$$g(\mu g) \subseteq \mu g \;\wedge\; ((g(\mu g \upharpoonright R)) \upharpoonright R) \cdot (\mu g)^\circ \subseteq R$$

$\Leftrightarrow$ { least fixpoint is a prefixpoint }

$$((g(\mu g \upharpoonright R)) \upharpoonright R) \cdot (\mu g)^\circ \subseteq R$$

Calculation of DP theorem

We are left with $((g(\mu g \upharpoonright R)) \upharpoonright R) \cdot (\mu g)^\circ \subseteq R$ which rewrites to

$$((h \cdot \mathsf{F}\, M \cdot T^\circ) \upharpoonright R) \cdot T \cdot \mathsf{F}\, H^\circ \cdot h^\circ \subseteq R \tag{38}$$

by expanding $\mu g = g(\mu g)$ and using abbreviations $H = \mu g$ and $M = H \upharpoonright R$.

The calculation of (38) is given in the next slide. A property of the "shrink" combinator,

$$(S \cdot T) \upharpoonright R \;\subseteq\; (R \cdot S)/(T^\circ) \tag{39}$$

(for S entire, see calculation below) plays the main role.

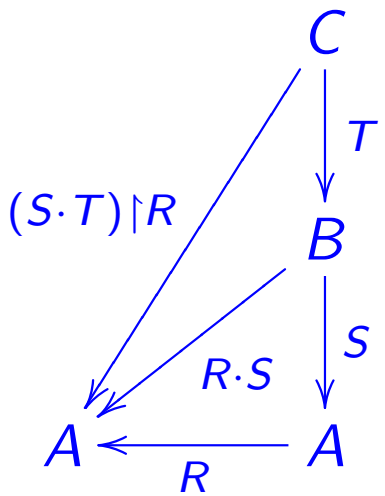
Calculation of DP theorem

$$
\begin{array}{ll}
 & ((h \cdot \mathsf{F}\, M \cdot T^\circ) \upharpoonright R) \cdot T \cdot \mathsf{F}\, H^\circ \cdot h^\circ \subseteq R \\
\Leftarrow & \quad \{ \text{ (39) for } S := h \cdot \mathsf{F}\, M,\ T := T^\circ \text{ etc } \} \\
 & R \cdot (h \cdot \mathsf{F}\, M) \cdot \mathsf{F}\, H^\circ \cdot h^\circ \subseteq R \\
\Leftrightarrow & \quad \{ \text{ relator F; substitutions } \} \\
 & R \cdot h \cdot \mathsf{F}((H \upharpoonright R) \cdot H^\circ) \cdot h^\circ \subseteq R \\
\Leftarrow & \quad \{ \text{ } (\upharpoonright)\text{-cancellation } \} \\
 & R \cdot h \cdot \mathsf{F}\, R \cdot h^\circ \subseteq R \\
\Leftarrow & \quad \{ \text{ } R \xleftarrow{h} \mathsf{F}\, R \text{ , that is, } h \cdot \mathsf{F}\, R \cdot h^\circ \subseteq R \text{ } \} \\
 & R \cdot R \subseteq R \\
\Leftrightarrow & \quad \{ \text{ } R \text{ transitive } \} \\
 & \textsc{True}
\end{array}
$$

Auxiliary

For entire S,

$$(S \cdot T) \upharpoonright R \;\subseteq\; (R \cdot S)/(T^\circ)$$



Calculation:

$$\begin{array}{ll} & (S \cdot T) \cap R/(S \cdot T)^\circ \;\subseteq\; (R \cdot S)/(T^\circ) \\ \Leftarrow & \quad \{\ \text{monotonicity}\ \} \\ & R/(T^\circ \cdot S^\circ) \;\subseteq\; (R \cdot S)/(T^\circ) \\ \Leftrightarrow & \quad \{\ (41)\ \text{below}\ \} \\ & \textsc{True} \end{array}$$

Auxiliary

Generalization of

$$R/(T \cdot f^\circ) = (R \cdot f)/T \tag{40}$$

to

$$R/(T \cdot S^\circ) \subseteq (R \cdot S)/T \quad \Leftarrow \quad S \text{ entire} \tag{41}$$

Calculation:

$$\begin{array}{ll} & X \subseteq R/(T \cdot S^\circ) \\ \Rightarrow & \quad \{ \text{ division ; monotonicity of composition } \} \\ & X \cdot T \cdot S^\circ \cdot S \subseteq R \cdot S \\ \Rightarrow & \quad \{ \ S \text{ is entire } \} \\ & X \cdot T \subseteq R \cdot S \\ \Leftrightarrow & \quad \{ \text{ division } \} \\ & X \subseteq (R \cdot S)/T \end{array}$$

Example: calculation of whole division

Recall

$$\div b \;=\; \underbrace{((\times b)^\circ \cdot \leq)}_{\text{“easy”}} \upharpoonright \underbrace{\geq}_{\text{“hard”}}$$

Strategy: appeal to corollary for unfolds of DP-theorem,

$$\langle \mu\, X \;::\; (in \cdot \mathsf{F}\, X \cdot T^\circ) \upharpoonright R \rangle \;\subseteq\; (\!|\, T \,|\!)^\circ \upharpoonright R$$

provided we fuse pair-algebra $(\times b)^\circ \cdot \leq$ into some unfold $(\!|\, T \,|\!)^\circ$:

$$\begin{array}{c} (\!|\, T \,|\!)^\circ = (\times b)^\circ \cdot \leq \\ \Leftrightarrow \qquad \{\ \text{converses}\ \} \\ (\!|\, T \,|\!) = \geq \cdot (\times b) \end{array}$$

where $\times b = (\!|\, 0, (+b) \,|\!)$ and $\geq = (\!|\, \top, suc \,|\!)$.

Example: calculation of whole division

Since $\geq \cdot \mathit{zero} = \top$ and $\geq \cdot (+b) = (+b) \cdot \geq$ — cf.

$$y \geq x + b \Leftrightarrow \langle \exists\ z\ ::\ y = z + b \wedge z \geq x \rangle$$

— by fold-fusion we obtain

$$T = [\top, +b]$$

Therefore, we can rely on:

$$\langle \mu\ X\ ::\ ([\mathit{zero}, \mathit{suc}] \cdot (\mathit{id} + X) \cdot [\top, +b]^\circ) \upharpoonright R \rangle \quad \subseteq \quad (\!|\top, +b|\!)^\circ \upharpoonright \geq$$

the lower side of which collapses into

$$\langle \mu\ X\ ::\ (\mathit{zero} \cup \mathit{suc} \cdot X \cdot (+b)^\circ) \upharpoonright \geq \rangle$$

Example: calculation of whole division

The final step is to remove the $\restriction$ combinator from

$$\langle \mu\ X \ ::\ (zero \cup suc \cdot X \cdot (+b)^\circ) \restriction \geq \rangle$$

using the **union rule**, recall

$$(R \cup S) \restriction Q = (R \restriction Q) \cap Q/S^\circ \cup (S \restriction Q) \cap Q/R^\circ$$

We obtain (with S abbreviating $suc \cdot X \cdot (+b)^\circ$ in the denominator)

$$\langle \mu\ X \ ::\ (zero \cap \geq / S^\circ) \ \cup \ suc \cdot X \cdot (+b)^\circ \rangle$$

since

- $zero \restriction \geq = zero$
- $\geq / zero^\circ = \top$ and $\leq \cdot zero = zero$
- S is simple (deterministic)

Example: calculation of whole division

We observe that

- $zero \cap \geq / S^\circ$ is the fragment of function $zero$ restricted to all points where S is undefined. (Details in the annex.)
- This happens outside the domain of $(+b)^\circ = (-b)$ — a partial function in $\mathbb{N}_0$ —, that is, for inputs below b.
- We are left with unfold

$$\div b \;=\; \langle \mu\, X \;::\; (< b) \to zero \;,\; suc \cdot X.(-b) \rangle$$

that is

$$\div b \;=\; (< b) \to zero \;,\; suc \cdot (\div b).(-b)$$

With points:

$$x \div b \;=\; \mathit{if}\ x < b\ \mathit{then}\ 0\ \mathit{else}\ 1 + ((x - b) \div b)$$

Last but not least

We stepped over side condition

$$\geq \xleftarrow{in=[zero,suc]} (id+ \geq)$$

— that is,

$$[zero, suc] \cdot (id+ \geq) \subseteq \geq \cdot [zero, suc]$$

— which can be dealt with in more generic terms by recalling that

$$\geq = (\!|\top, suc|\!)$$

and appealing to the generic result which follows.

Orderings which "extend" *in*

Let *in* be "the" initial F-algebra and $R = (\!|\, I \,|\!)$ where $in \subseteq I$. Then $R \xleftarrow{in} \mathsf{F}\, R$ always holds, cf:

$$
\begin{array}{ll}
 & in \cdot \mathsf{F}(\!|\, I \,|\!) \subseteq (\!|\, I \,|\!) \cdot in \\
\Leftrightarrow & \quad \{ \text{ fold-cancellation } \} \\
 & in \cdot \mathsf{F}(\!|\, I \,|\!) \subseteq I \cdot \mathsf{F}(\!|\, I \,|\!) \\
\Leftarrow & \quad \{ \text{ monotonicity } \} \\
 & in \subseteq I
\end{array}
$$

Thus, for instance, $\geq \xleftarrow{in=[zero,suc]} (id + \geq)$ holds in the natural numbers since $\geq = (\!|\, [\top, suc] \,|\!)$ and $zero \subseteq \top$. Similarly for $\leq \xleftarrow{in=[zero,suc]} (id + \leq)$ since $\leq = (\!|\, [zero, zero \cup suc] \,|\!)$ and $suc \subseteq zero \cup suc$.

Going generic

- What we have done for whole division generalizes to a broad class of algorithms which can be identified with **GC adjoints**, for instance *take* and *trim*, respectively

$$\mathsf{len}\, y \leq n \wedge y \sqsubseteq x \;\Leftrightarrow\; y \sqsubseteq \mathit{take}(n, x)$$
$$\mathit{sum}\; y \leq n \wedge y \sqsubseteq x \;\Leftrightarrow\; y \sqsubseteq \mathit{trim}(n, x)$$

  where $\sqsubseteq = (\!|\mathit{nil}, \mathit{nil} \cup \mathit{cons}|\!)$ is the prefix ordering.

- The process repeats itself, eg.

$$\mathit{trim} = (\langle \mathit{sum}, \mathit{id} \rangle^{\circ} \cdot (\leq \times \sqsubseteq)) \upharpoonright \sqsubseteq^{\circ}$$

  where $\langle \mathit{sum}, \mathit{id} \rangle$ becomes a fold via "banana-split", and so on and so forth (Details in a forthcoming paper.)

Winding up — related work

- The $R \upharpoonright S$ combinator corresponds to what Bird and de Moor (1997) write as $min\ S \cdot \Lambda R$ where $\mathcal{P}B \xleftarrow{\Lambda R} A$ (a function) is the power-transpose of relation $B \xleftarrow{R} A$ and $B \xleftarrow{min\ S} \mathcal{P}B$ computes the minimum of a set (if it exists) according to relation S.

- By exploiting the algebra of $R \upharpoonright S$ we show that there is no need for powersets (power allegory, in general).

- We hope our approach makes the last chapters of (Bird and de Moor, 1997) more accessible to a wider community, by simplifying both the notation and the main results.

Winding up — related work

- However, still a lot of relational machinery — need for auxiliary results.

- Currently studying the GC approach to algorithm design more deeply, recall a previous talk of one of us (Oliveira, 2010).

- Also trying to calculate more complex adjoints, for instance the *best Gantt chart (schedule)* induced by a task-dependency graph G, knowing tasks time spans, which is the upper adjoint of GC

$$sc * G \mathrel{\dot{\geq}} sp \quad \Leftrightarrow \quad sc \mathrel{\dot{\geq}} sp \div G \tag{42}$$

where $sc * G$ reads "the laziest span for sc allowed by G", and where $sp \div G$ reads "the best schedule for task span sp allowed by G".

Annex — Reasoning about $zero \cap \geq / S^\circ$

Recall that S abbreviates $suc \cdot X \cdot (+b)^\circ$. We first show that $zero \cap \geq / S^\circ$ and S are domain-disjoint:

$$
\begin{array}{ll}
 & (zero \cap \geq / S^\circ) \cdot S^\circ \subseteq \bot \\
\Leftarrow & \quad \{\ (R \cap S) \cdot U \subseteq R \cdot U \cap S \cdot U\ \} \\
 & zero \cdot S^\circ \cap (\geq / S^\circ) \cdot S^\circ \subseteq \bot \\
\Leftarrow & \quad \{\ \text{division (cancelation) ; Dedekind}\ \} \\
 & zero \cdot (S^\circ \cap zero^\circ \cdot \geq) \subseteq \bot \\
\Leftrightarrow & \quad \{\ \leq \cdot\, zero = zero\ \} \\
 & zero \cdot (S^\circ \cap zero^\circ) \subseteq \bot \\
\Leftrightarrow & \quad \{\ S \cap zero = \bot\ \} \\
 & zero \cdot \bot \subseteq \bot
\end{array}
$$

Reasoning about $zero \cap \geq / S^\circ$

This, together with the implicit GC yields

$$X \subseteq zero \cap \geq / S^\circ \quad \Leftrightarrow \quad X \subseteq zero \ \wedge \ X \cdot S^\circ \subseteq \bot$$

meaning that $zero \cap \geq / S^\circ$ is the largest fragment of $zero$ domain-disjoint of S. That is, using McCarthy's conditional notation:

$$zero \cap \geq / S^\circ \quad = \quad S \to \bot \ , \ zero$$

Since $S = S \to S \ , \ \bot$, we obtain

$$zero \cap \geq / S^\circ \ \cup \ S \quad = \quad S \to S \ , \ zero$$

References

C. Aarts, R.C. Backhouse, P. Hoogendijk, E.Voermans, and J. van der Woude. A relational theory of datatypes, December 1992. Available from `www.cs.nott.ac.uk/~rcb`.

R. Bird and O. de Moor. *Algebra of Programming*. Series in Computer Science. Prentice-Hall International, 1997. C.A.R. Hoare, series editor.

M.A. Ferreira and J.N. Oliveira. Variations on an Alloy-centric tool-chain in verifying a journaled file system model. Technical Report DI-CCTC-10-07, DI/CCTC, University of Minho, Gualtar Campus, Braga, January 2010. Available from the authors' websites.

J.N. Oliveira. A Look at Program "Galculation", January 2010. Presentation at the IFIP WG 2.1 #65 Meeting.

P.F. Silva and J.N. Oliveira. 'Galculator': functional prototype of a Galois-connection based proof assistant. In *PPDP '08: Proceedings of the 10th international ACM SIGPLAN conference on Principles and practice of declarative programming*, pages 44–55, New York, NY, USA, 2008. ACM. ISBN

978-1-60558-117-0. doi:
http://doi.acm.org/10.1145/1389449.1389456. .