Sequential algorithms as bistable maps

Pierre-Louis Curien (CNRS - Université Paris 7)

March 19, 2009

Abstract

We exhibit Cartwright-Curien-Felleisen's model of observably sequential algorithms as a full subcategory of Laird's bistable biorders, thereby reconciling two views of functions: functions-as-algorithms (or programs), and functions-as-relations. We then characterize affine sequential algorithms as affine bistable functions in the full subcategory of locally boolean orders.

1 En guise de prologue

Gilles Kahn, pour moi, ce fut d'abord un article, en français s'il vous plait, texte qui fut le point de départ de ma recherche:

G. Kahn et G. Plotkin, Domaines concrets, TR IRIA-Laboria 336 (1978), paru en version anglaise en 1993 – signe de son influence dans le temps – dans le volume d'hommage à Corrado Böhm [14].

On ne pouvait imaginer un meilleur appât pour le jeune homme que j'étais, arrivé à l'informatique par le fruit d'une hésitation entre mathématiques (intimidantes) et langues (les vraies). Un autre collègue trop tôt disparu, Maurice Gross, m'avait aidé à choisir une tierce voie et m'avait guidé vers le DEA d'Informatique Théorique de Paris 7. Les cours de Luc Boasson et de Dominique Perrin m'avaient déjà bien ferré, mais la rencontre des domaines concrets m'a définitivement "attrapé", et parce qu'il s'agissait de structures ressemblant aux treillis – rencontrés assez tôt dans ma scolarité grâce aux Leçons d'Algèbre Moderne de Paul Dubreil et Marie-Louise Dubreil Jacotin que m'avait conseillées mon professeur de mathématiques –, et parce que Gérard Berry qui m'avait mis ce travail entre les mains avait une riche idée pour bâtir sur cette pierre.

L'idée directrice de cet article était de donner une définiton générale de structure de données, comprenant les listes, les arbres, les enregistrements, les enregistrements avec variantes, etc..., et, comme l'on fait dans toute bonne mathématique, une bonne notion de morphisme entre ces structures: Cette définition était donnée sous deux facettes équivalentes et reliées par un *théorème de représentation*: l'une concrète, en termes de cellules (nœuds d'arbres, champs d'enrigistrements, . . .) et de valeurs, l'autre abstraite, en termes d'ordres partiels. Ce théorème, le premier du genre, a servi de modèle à des travaux ultérieurs (ceux de Winskel sur les structures d'événements en particulier).

S'agissant des morphismes, la contribution n'était pas moins importante. Après celles de Vuillemin et de Milner qui ne s'apppliquaient bien qu'au premier ordre, la définition de fonction séquentielle de Kahn-Ploktin était "la bonne". Cette notion, couplée au théorème de séquentialité de Berry obtenu à la même époque, ouvrait la voie à la construction d'un modèle séquentiel du λ-calcul, et c'est ici qu'intervint l'idée de Berry: passer des fonctions (séquentielles) à une notion plus concrète de morphisme.

Les algorithmes séquentiels [3], nés de cette intuition initiale, et que nous revisitons ici, sont, je l'espère, restés fidèles à l'esprit des domaines concrets, qui se voulaient formaliser des structures données *observables*. Dans un langage comme CAML, les types fonctionnels ne sont pas observables, mais dans le langage CDS (concrete data structure) que Berry a proposé et qui a été développé sur les fondements théoriques établis dans ma thèse, ils le sont [4]. Last but not least, la première sémantique opérationnelle du langage (décrite dans ma thèse d'Etat [9]) bien que séquentielle, s'inspire beaucoup du modèle des coroutines, présenté dans autre article fondateur [13].

Sur un plan plus pratique, c'est sur la table traçante du (mythique) bâtiment 8 de l'INRIA Rocquencourt que j'ai pu sortir les beaux transparents (je ne crois pas en avoir réalisé, sans effort particulier, de plus esthétiques depuis) de ma soutenance de Thèse d'Etat, écrits en FLIP.

Elégance, profondeur, et sens pratique, cela ne résume pas Gilles Kahn, mais c'est ce dont je puis témoigner ici. De sa voix aussi, joviale et impérieuse, voire impériale, qui m'a sauvé la mise lors d'une présentation de mes travaux dans un colloque franco-américain à Fontainebleau. Je m'apprêtais à écrire au feutre, non sur le transparent, mais sur la table du projecteur...

2 Introduction

This paper reconciles two views of functions: functions-as-algorithms (or programs), and functions-as-relations. Our functions-as-algorithms are the observably sequential algorithms on sequential data structures of Cartwright, Curien and Felleisen [5, 6, 7], and our functions-as-relations are Laird's bistable and pointwise monotonic functions on bistable biorders [15]. We exhibit a full embedding from the former to the latter.

Since the moment when the first version of these notes has circulated privately (2002), Laird has further improved the understanding of the connection by defining a full subcategory of bistable biorders, the category of locally boolean domains, whose intensional behaviour may be "synthesized" through decomposition theorems. A consequence of these theorems is that every locally boolean domain is isomorphic to a sequential data structure.

Hence the results presented here actually yield an equivalence of categories (and another one between the respective subcategories of affine morphisms).

In order to make the paper relatively self-contained, we provide the necessary definitions in sections 3 and 4. The main full embedding result is proved in section 5, and the affine case is addressed in section 6.

3 Sequential data structures

We define sequential data structures, which are concrete descriptions of partial orders whose elements are called strategies (there is indeed a game semantical reading of these structures, see e.g. [7]).

Definition 3.1 *A* sequential data structure *(sds) [1] is a triple* $\mathbf{S} = (C, V, P)$*, where*

- C *is a set of* cells*,*
- V *is a set of* values *(C and V disjoint),*
- P *is a prefix-closed set of non-empty alternating sequences* $c_1 v_1 c_2 \ldots$ *(with the* c_i*'s in* C *and the* v_i*'s in* V*), which are called* positions*,*

We denote by Q (resp. R) the subset of sequences of odd (resp. even) length of P, which are called queries *(resp.* responses*).*

One moreover assumes two special values $\perp$ and $\top$, not in V (nor in C) (in earlier work [6], $\top$ was called error, and in ludics [12] it is called demon). We let w range over $V \cup \{\perp, \top\}$.

Here are a few examples of sds's (described by their forest of positions):

$$\begin{array}{cc}
\mathit{bool} & \mathit{bool} \times \mathit{bool} \\[1em]
? \left\{ \begin{array}{l} \mathit{tt} \\ \mathit{ff} \end{array} \right. &
\left\{ \begin{array}{l} ?_1 \left\{ \begin{array}{l} \mathit{tt} \\ \mathit{ff} \end{array} \right. \\ ?_2 \left\{ \begin{array}{l} \mathit{tt} \\ \mathit{ff} \end{array} \right. \end{array} \right.
\end{array}$$

Thus, *bool* has one cell ? and two values *tt* and *ff*, while *bool* $\times$ *bool* is made of two copies of *bool*, tagged by 1 and 2, respectively.

Our last example is the sds of terms over a signature, say, $\Sigma = \{f^2, g^1, a^0\}$:

$$\epsilon \left\{ \begin{array}{l} f \left\{ \begin{array}{l} 1 \left\{ \begin{array}{l} f \left\{ \begin{array}{l} 11\cdots \\ 12 \left\{ \begin{array}{l} f\cdots \\ g \left\{ \ 121\cdots \right. \\ a \end{array} \right. \end{array} \right. \\ g \left\{ \ 11\cdots \right. \\ a \end{array} \right. \\ 2\cdots \end{array} \right. \\ g \left\{ \ 1\cdots \right. \\ a \end{array} \right.$$

Here, the cells are words over the alphabet $\{1, 2\}$ (occurrences!) and the values are the function symbols.

Definition 3.2 *An observable strategy (or strategy for short) is a set $x \subseteq R \cup \{q\top \mid q \in Q\} \cup \{q\bot \mid q \in Q\}$ which satisfies the following properties:*

- *x is closed under (even length) prefixes,*
- *whenever $qw_1, qw_2 \in x$, then $w_1 = w_2$,*
- *whenever $r \in x$, then for all c such that $rc \in Q$, there exists w such that $rcw \in x$.*

The second condition is the crucial one: it tells us that queries are answered uniquely in a strategy. The other conditions are there for technical convenience.

A typical strategy, represented as a forest, i.e., as a set of branches, looks like this:

$$\left\{ \begin{array}{l} \vdots \\ c_0\ v_0 \left\{ \begin{array}{l} c_1\ v_1 \left\{ \begin{array}{l} c_3\ \bot \\ \vdots \\ c_4\ \top \\ \vdots \\ c_5\ \{ \end{array} \right. \\ c_2\ v_2 \left\{ \ \vdots \right. \end{array} \right. \\ \vdots \end{array} \right.$$

Here are some concrete strategies (below each strategy, we give the corresponding boolean, pair of booleans, or term):

$$\begin{array}{lcccc}
\mathit{bool} & ?\bot & ?\mathit{tt} & ?\mathit{ff} & ?\top \\[2ex]
 & \bot & \mathit{tt} & \mathit{ff} & \top
\end{array}$$

$$\begin{array}{lcccc}
\mathit{bool} \times \mathit{bool} & \left\{\begin{array}{l} ?_1\bot \\ ?_2\bot \end{array}\right. & \left\{\begin{array}{l} ?_1\bot \\ ?_2\mathit{tt} \end{array}\right. & \left\{\begin{array}{l} ?_1\mathit{ff} \\ ?_2\bot \end{array}\right. & \cdots \\[2ex]
 & (\bot, \bot) & (\bot, \mathit{tt}) & (\mathit{ff}, \bot) & \cdots
\end{array}$$

$$\epsilon\; f \left\{\begin{array}{l} 1\; f \left\{\begin{array}{l} 11\; a \\ 12\; g \left\{\; 121\; a \right. \end{array}\right. \\[2ex] 2\; a \end{array}\right.$$

$$f(f(a, g(a)), a)$$

The above definition of strategy does no do justice to the arborescent structure of strategies, as in the examples just given. We now give a syntax of strategies-as-terms.

$$\frac{rc \in Q}{c\bot : \mathit{strat}(rc)} \qquad \frac{rc \in Q}{c\top : \mathit{strat}(rc)}$$

$$\frac{rc_0v_0 \in R \qquad x_c : \mathit{strat}(rc_0v_0c) \quad (rc_0v_0c \in Q)}{c_0v_0\dot{\{}x_c \mid rc_0v_0c \in Q\} : \mathit{strat}(rc_0)}$$

$$\frac{x_c : \mathit{strat}(c) \quad (c \in Q)}{\{x_c \mid c \in Q\} : \mathit{strat}}$$

A strategy is a set $x = \{x_c \mid c \in Q\} : \mathit{strat}$. The strategy-as-set-of-responses associated to a strategy-as-term is defined as $\{r \mid r \in x\}$, where $r \in x$ is defined by the following rules:

$$\frac{}{c\bot \in c\bot} \qquad \frac{}{c\top \in c\top}$$

$$\frac{r_1 \in x_{c_1}}{c_0 v_0 r_1 \in c_0 v_0 \dot{\{} x_c \mid r c_0 v_0 c \in Q\}} \qquad \frac{}{c_0 v_0 \in c_0 v_0 \dot{\{} x_c \mid r c_0 v_0 c \in Q\}}$$

$$\frac{r \in x_c}{r \in \{x_c \mid c \in Q\}}$$

This defines an injection, whose image is easily seen to be the set of strategies x that do not have any infinite branch, in the sense that there is no sequence

$$c_1\, v_1\, c_2\, v_2\, \dots\, c_n\, v_n\, \dots$$

whose even prefixes all belong to x. We shall call such strategies *finitary.* Hence we have two alternative definitions of finitary strategies: one by extension, and a "procedural" one.

We denote the set of strategies by $D^{\top}(\mathbf{S})$, and we use $D(\mathbf{S})$ for the set of strategies obtained by removing the rule introducing $c\top$. We write:

- $q \in A(x)$ if $q\bot \in x$,
- $q \in F(x)$ if $qv \in x$ for some $v \in V$ or if $q\top \in x$, and
- $x <_q y$ when $q \in A(x)$ and $q \in F(y)$.

If $q_1 = r_1 c_1, \dots, q_n = r_n c_n \in A(x)$, we denote by $x[q_1 \leftarrow x_1, \dots, q_n \leftarrow x_n]$ the strategy obtained by replacing $q_i\bot$ by $x_i : \mathit{strat}(q_i)$, for all i, in x. We shall abbreviate $q = rc \leftarrow c\bot$ as $q \leftarrow \bot$, and similarly with $\top$. We shall use the convention that writing

$$x = x[q_0 \leftarrow \bot, \dots, q_n \leftarrow \bot]$$

l means that $A(x) = \{q_0, \dots, q_n\}$.

Definition 3.3 *We define four orders $\leq^s$ (stable), $\leq^c$ (costable) , $\leq$ (bistable), and $\sqsubseteq$ (pointwise) as the congruence closures of, respectively:*

for $\leq^s$:

$$\frac{rc \in Q \quad x \in \mathit{strat}(rc)}{c\bot \leq^s x}$$

for $\leq^c$*:*

$$\frac{x \in \mathit{strat}(rc)}{x \leq^c c\top}$$

for $\leq$*:*

$$\frac{rc \in Q}{c\bot \leq c\top}$$

for $\sqsubseteq$*:*

$$\frac{rc \in Q \quad x \in \mathit{strat}(rc)}{c\bot \sqsubseteq x} \qquad \frac{rc \in Q \quad x \in \mathit{strat}(rc)}{x \sqsubseteq c\top}$$

By congruence closure, we mean, say for $\sqsubseteq$, the following rules:

$$\frac{rc_0v_0 \in R \qquad x_c \sqsubseteq y_c \quad (rc_0v_0c \in Q)}{c_0v_0\dot{\{}x_c \mid rc_0v_0c \in Q\} \sqsubseteq c_0v_0\dot{\{}y_c \mid rc_0v_0c \in Q\}}$$

$$\frac{x_c \sqsubseteq y_c \quad (c \in Q)}{\{x_c \mid c \in Q\} \sqsubseteq \{y_c \mid c \in Q\}}$$

In words, replacing a $\bot$ by a (correctly typed) tree results in a stable increase, while removing a subtree and replacing it by $\top$ results in a costable increase. The bistable order is the intersection of the stable and the costable order: an increase in this order consists only in changing $\bot$'s to $\top$'s. The bistable order turns out to play a crucial role in the axiomatization (see section 4). The pointwise order is the order generated by the union of the stable and the costable orders.

Streicher has remarked that $\sqsubseteq$ and $\leq$ make an sds a bistable biorder. This is an easy check left to the reader (after reading of section 4). Here, we extend this to a full and faithful embedding of the category of sds's and sequential algorithms into the category of bistable and $\sqsubseteq$-monotonic maps. Therefore, we now move on to define the relevant *categories* [1, 15].

Definition 3.4 *Given sets* $A, B \subseteq A$*, for any word* $w \in A^*$*, we define* $w\lceil_B$ *as follows:*

$$\epsilon\lceil_B = \epsilon \qquad (wm)\lceil_B = \begin{cases} w\lceil_B & \text{if } m \in A\backslash B \\ (w\lceil_B)m & \text{if } m \in B\ . \end{cases}$$

Definition 3.5 *Given two sds's* $\mathbf{S} = (C, V, P)$ *and* $\mathbf{S}' = (C', V', P')$*, we define* $\mathbf{S} \multimap \mathbf{S}' = (C'', V'', P'')$ *as follows. The sets* C'' *and* V'' *are disjoint unions:*

$$\begin{array}{rcl} C'' & = & C' \cup V \\ V'' & = & V' \cup C\ . \end{array}$$

P'' *consists of the alternating positions* s *starting with a* c'*, and which are such that:*

$$\begin{array}{l} s\lceil_{\mathbf{S}'} \in P', (s\lceil_{\mathbf{S}} = \epsilon \textit{ or } s\lceil_{\mathbf{S}} \in P), \textit{ and} \\ s \textit{ has no prefix of the form } scc'. \end{array}$$

The strategies of $\mathbf{S} \multimap \mathbf{S}'$ *are called observably affine sequential algorithms from* $\mathbf{S}$ *to* $\mathbf{S}'$.

A typical affine sequential algorithm looks like this:

$$\left\{\begin{array}{l}\vdots\\ c'_1\ c_1\left\{\begin{array}{l}\vdots\\ v_1\ c_2\left\{\begin{array}{l}\vdots\\ v_2\ v'_1\left\{\begin{array}{l}\vdots\\ c'_2\cdots\\ \vdots\end{array}\right.\\ \vdots\end{array}\right.\\ \vdots\end{array}\right.\\ \vdots\end{array}\right.$$

The initial query c'_1 of the output prompts a query c_1 to the input. If the answer to this query is v_1, then the strategy wants to further explore the input and asks the query $c_1v_1c_2$, and so on, until enough of the input is known to answer the query c'_1 with v'_1, and then the output may launch a new query c'_2, etc...

We next move on to define observably sequential algorithms (not necessarily affine). Our model follows linear logic's decomposition $\mathbf{S} \to \mathbf{S}' = (!\mathbf{S}) \multimap \mathbf{S}'$ [11]. The decomposition for sequential algorithms on sequential data structures was discovered by Lamarche [17, 7].

Definition 3.6 (exponential – sds) *Let* $\mathbf{S} = (C, V, P)$ *be a sds. The following recursive clauses define a set* $P_!$ *of alternating words over* $Q \cup R = P$*:*

$$\frac{(\mathbf{r} = \epsilon \textit{ or } \mathbf{r} \in P_!) \quad q \in A(\textit{state}(\mathbf{r}))}{\mathbf{r}q \in P_!} \qquad \frac{\mathbf{q} \in P_! \textit{ and } \textit{state}(\mathbf{q}r) \in D(\mathbf{S})}{\mathbf{q}r \in P_!}$$

where state is the following function mapping responses (or ϵ) of $P_!$ to strategies of $\mathbf{M}$*:*

$$\frac{}{\mathit{state}(\epsilon) = \{c\bot \mid c \textit{ initial}\}} \qquad \frac{q = r_1c_1 \quad r = qv_1}{\mathit{state}(\mathbf{r}qr) = \mathit{state}(\mathbf{r})[q \leftarrow c_1v_1\{c\bot \mid rc \in Q\}]}$$

The sds $(Q, R, P_!)$ is called $!\mathbf{S}$.

The above definition looks technical, but it just amounts to saying that the cells and values of $!\mathbf{S}$ are the queries and the responses of $\mathbf{S}$, and that the queries and responses of $!\mathbf{S}$ are sequences obtained from some tree traversal of the strategies of $\mathbf{S}$. This is in spirit similar to the coherence space semantics of linear logic [11].

Definition 3.7 *Given two sds's $\mathbf{S}$ and $\mathbf{S}'$, the strategies of $!\mathbf{S} \multimap \mathbf{S}'$ are called observably sequential algorithms from $\mathbf{S}$ to $\mathbf{S}'$.*

Despite their appearance, observably sequential algorithms are (in bijection with) functions: this key insight did not wait for Laird's work, and was indeed one of the contributions of Cartwright and Felleisen [5, 6, 1] to the older work of Berry and Curien on sequential algorithms on concrete data structures [3].

Definition 3.8 *Let f be an observably sequential algorithm from $\mathbf{S}$ to $\mathbf{S}'$ and x a strategy of $\mathbf{S}$. We define $f \cdot x$ (also written $f(x)$) as the normal form of $\langle\, f \mid x \,\rangle$ for the following rewriting system, which is easily seen to be confluent (no critical pairs) and terminating on finitary strategies.*

$$\begin{array}{llll}
\langle\, c'v'\{x''_i \mid i \in I\} \mid x \,\rangle & \rightarrow & c'v'\{\langle\, x''_i \mid x \,\rangle \mid i \in I\} & & \\
\langle\, c'\bot \mid x \,\rangle & \rightarrow & c'\bot & & \\
\langle\, c'\top \mid x \,\rangle & \rightarrow & c'\top & & \\
\langle\, c'q\{x''_r \mid r''c'qr \in Q''\} \mid x \,\rangle & \rightarrow & c'?(\langle\, x''_{r_0} \mid x \,\rangle) & (r_0 = qv_0 \in x) & (\mathit{in\ strat}(r''c')) \\
\langle\, c'q\{x''_r \mid r''c'qr \in Q''\} \mid x \,\rangle & \rightarrow & c'\bot & (q\bot \in x) & (\mathit{in\ strat}(r''c')) \\
\langle\, c'q\{x''_r \mid r''c'qr \in Q''\} \mid x \,\rangle & \rightarrow & c'\top & (q\top \in x) & (\mathit{in\ strat}(r''c')) \\
?\langle\, r_0q\{x''_r \mid r''r_0qr \in Q''\} \mid x \,\rangle & \rightarrow & ?\langle\, x''_{r_1} \mid x \,\rangle & (r_1 \in x) & (\mathit{in\ strat}(r''r_0)) \\
?(\langle\, r_0q\{x''_r \mid r''r_0qr \in Q''\} \mid x \,\rangle) & \rightarrow & \bot & (q\bot \in x) & (\mathit{in\ strat}(r''r_0)) \\
?(\langle\, r_0q\{x''_r \mid r''r_0qr \in Q''\} \mid x \,\rangle) & \rightarrow & \top & (q\top \in x) & (\mathit{in\ strat}(r''r_0)) \\
?(\langle\, rv'\{x''_i \mid i \in I\} \mid x \,\rangle) & \rightarrow & v'\{\langle\, x''_i \mid x \,\rangle \mid i \in I\} & & \\
?(\langle\, r\bot \mid x \,\rangle) & \rightarrow & \bot & & \\
?(\langle\, r\top \mid x \,\rangle) & \rightarrow & \top & &
\end{array}$$

For example, if $c'\,c_1\,v_1\,\ldots\,c_n\,v_n\,v'\,c'_1\,c\,v\,v'_1 \in f$ and $c_1\,v_1\,\ldots\,c_n\,v_n\,c\,v \in x$, then $c'\,v'\,c'_1\,v'_1 \in f \cdot x$. (The reader should have in mind here that potentially f (resp. x) branches after $c_1, \ldots, c_n, c$ (resp. after $v_1, \ldots, v_n$), and that the strategies choices determine alternately which branch to choose.)

Definition 3.9 *Let* $\mathbf{S}$ *and* $\mathbf{S}'$ *be two sds's. A* $\leq^s$*-monotonic function* $f : D^\top(\mathbf{S}) \to D^\top(\mathbf{S}')$ *is called* sequential *at* x *if for any* $q' \in A(f(x))$ *one of the following properties hold:*

1. $\forall\, y \geq^s x \;\; q' \notin F(f(y))$.
2. $\exists\, q \in A(x) \;\;\; \forall\, y > x \;\; (f(x) <_{q'} f(y) \;\Rightarrow\; x <_q y)$.

A query q *satisfying condition (2) is called a* sequentiality index *of* f *at* (x, q')*. The index is called strict if (1) does not hold. If (1) holds, then any* q *in* $A(x)$ *is a (vacuous) sequentiality index.*

If q *is a sequentiality index at* (x, q') *and if moreover* $q'\top \in f(x[q \leftarrow \top])$*, then we say that* f *is* observably sequential *at* x, q'*, with index* q *(note then that the index is strict and that there can be no other sequentiality index).*

The function f *is called sequential (resp. observably sequential) from* $\mathbf{S}$ *to* $\mathbf{S}'$ *if it is sequential (resp. observably sequential) at all points.*

With the notation introduced above and using the $\leq^s$ monotonicity, we can rephrase the definition of observably sequential function more symmetrically as follows, at $x = x[q_1 \leftarrow \bot, \ldots, q_n \leftarrow \bot]$:

$$\begin{array}{l} \forall x_1, \ldots, x_{i-1}, x_{i+1}, \ldots, x_n \\ \qquad q'\bot \in f(x[q_1 \leftarrow x_1, \ldots q_{i-1} \leftarrow x_{i-1}, q_i \leftarrow \bot, q_{i+1} \leftarrow x_{i+1}, \ldots, q_n \leftarrow x_n]) \end{array}$$

$$\begin{array}{l} \forall x_1, \ldots, x_{i-1}, x_{i+1}, \ldots, x_n \\ \qquad q'\top \in f(x[q_1 \leftarrow x_1, \ldots q_{i-1} \leftarrow x_{i-1}, q_i \leftarrow \top, q_{i+1} \leftarrow x_{i+1}, \ldots, q_n \leftarrow x_n]) \end{array}$$

Or, alternatively, using instead the $\sqsubseteq$-monotonicity (which is established below), we get the following quantifier-free definition:

$$q'\bot \in f(x[q_1 \leftarrow \top, \ldots q_{i-1} \leftarrow \top, q_i \leftarrow \bot, q_{i+1} \leftarrow \top, \ldots, q_n \leftarrow \top])$$

$$q'\top \in f(x[q_1 \leftarrow \bot, \ldots q_{i-1} \leftarrow \bot, q_i \leftarrow \top, q_{i+1} \leftarrow \bot, \ldots, q_n \leftarrow \bot])$$

Theorem 3.10 *Observably sequential algorithms and observably sequential functions are in one-to-one correspondence, and under the bijection, the pointwise (resp. the stable) (resp. the bistable) ordering defined above indeed becomes the pointwise ordering (resp. Berry's stable ordering [2]) (resp. Laird's bistable ordering [15]):*

$$\begin{array}{lcl} f \sqsubseteq g & \Leftrightarrow & (\forall\, x \;\; f(x) \;\sqsubseteq\; g(x)) \\ f \leq^s g & \Leftrightarrow & (f \sqsubseteq g \textit{ and } (\forall\, x, y \;\; (x \leq^s y \Rightarrow f(x) = f(y) \wedge^s g(x)))) \\ f \leq g & \Leftrightarrow & (f \sqsubseteq g \textit{ and } (\forall\, x, y \;\; (x \leq y \Rightarrow f(x) = f(y) \wedge g(x)) \textit{ and } g(y) = f(y) \vee g(x))) \end{array}$$

The notation $\wedge^s$ *denotes the greatest lower bound with respect to the stable ordering, while* $\wedge$ *and* $\vee$ *refer to the bistable ordering (all these bounds exist, because* $f(y), g(x) \sqsubseteq g(y)$*).*

Proof. Except for the last equivalence, the theorem is proved in [6]. We check here only that if f, g are observably sequential, $f \leq g$, and $x \leq y$, then $g(y) = g(x) \vee f(y)$. Suppose that $g(y) > g(x) \vee f(y)$. Let q' be filled in $g(y)$ and accessible from $g(x) \vee f(y)$. It follows that q' is accessible from $g(x)$ and from $f(y)$. The only way for g to make the difference w.r.t. f on input y is to use one of its additional $\top$'s, say, $q'\top$, in the interaction with y. More precisely, one of the $\top$'s of g is responsible for the filling of q'. But the interaction of g with y is the same as with x. Hence q' must be filled in $g(x)$, contrary to our assumption. □

As an illustration of what is going on, suppose that $f_1, f_2 : \mathbf{S} \to \mathbf{S}'$ are $\leq^s$-minimal observably sequential algorithms such that

$$\begin{array}{c} c'\ c_1\ (c_1v_1)\ c_2\ (c_2v_2)\ c \in f_1 \\ c'\ c_2\ (c_2v_2)\ c_1\ (c_1v_1)\ c \in f_2 \end{array}$$

It looks like f_1 and f_2 define the same function (and indeed they do define the same function from $D(\mathbf{S})$ to $D(\mathbf{S}')$), but on $x = \{c_1\bot, c_2\top\}$ we have:

$$\begin{array}{c} f_1 \cdot x = \{c'\bot\} \\ f_2 \cdot x = \{c'\top\} \end{array}$$

Here, both $\bot$ and $\top$ act very much like colours used in chemical experiments: they track the presence of a proper value in c_1 and c_2, respectively. The special values $\bot$ and $\top$ play entirely symmetric role as long as no infinite computations take place.

If we allow general, non finitary strategies, the abstract machine of definition 3.8 may well diverge (i.e. not terminate). Following Scott-Ershov's tradition, a non-terminnating computation receives $\bot$ as value, and then $\bot$ and $\top$ cease to be symmetrical, since $\bot$ has become overloaded: it is both a symbol for non-termination, and an explicit symbol for a special error value that could be called "stop by lack of information". The other error value $\top$ could be called "deliberate stop" (see [10, 12, 8]).

4 Bistable biorders

So far, so good: we have a charaterization of our notion of functions-as-algorithms as functions-as-relations. But still, the definition of sequential function (originally due to Kahn and Plotkin [14]) requires a rather concrete, "algorithmic" definition of domain (here, sequential data structures). We can get rid of this too, using Laird's notion of bistability, which can be defined more abstractly. In this section, we recall Laird's definitions [15].

Definition 4.1 *A bistable biorder is a set $(D, \leq, \sqsubseteq)$ equipped with two partial orders, such that whenever x, y have a lower or an upper bound for $\leq$ then $x \vee y$ and $x \wedge y$ exist (for*

$\leq$), and then $x \vee y = x \sqcup y$ and $x \wedge y = x \sqcap y$, hence the two orders coincide for $\leq$-connected components. Moreover $\wedge$ and $\vee$ distribute over each other in each component. One writes $x \updownarrow y$ when x, y have a $\leq$-upper bound (or equivalently a $\leq$-lower bound). A bistable function is a $\leq$-monotonic function such that for any $x \updownarrow y$:

$$\begin{array}{c} f(x \wedge y) = f(x) \wedge f(y) \\ f(x \vee y) = f(x) \vee f(y) \end{array}$$

(we refer to the two halves of this property as $\wedge$-bistability and $\vee$-bistability, respectively).

We observe (for the reader with some linear logic culture) that the preservation of $\vee$ in the above definition is not a requirement of linearity, as it is taken with respect to $\leq$, not $\leq^s$.

In [15], the following theorem is proved:

Proposition 4.2 *The category of bistable biorders and bistable and $\sqsubseteq$-monotonic functions is cartesian closed.*

5 Full embedding

In this section, we show that the category of observably sequential algorithms embeds fully in Laird's category of bistable biorders and bistable and $\sqsubseteq$-monotonic functions. This amounts to the following proposition:

Theorem 5.1 *Given two sds's* **S** *and* **S'**, *a function $f : D^\top(\mathbf{S})$ to $D^\top(\mathbf{S'})$ is observably sequential if and only if it is bistable and $\sqsubseteq$-monotonic.*

PROOF. The proof of (f $\sqsubseteq$-monotonic and bistable $\Rightarrow$ f sequential) follows the steps of (and generalizes) Laird's proof of this property at $x = \bot$ [15]. Suppose that q' is accessible from $f(x)$ (with $x = x[q_0 \leftarrow \bot, \ldots, q_n \leftarrow \bot]$), and suppose moreover that none of the q_i's is a sequentiality index. Then by $\sqsubseteq$ monotonicity we have that q' is filled in all of the $f(x[q_0 \leftarrow \top, \ldots, q_{i-1} \leftarrow \top, q_i \leftarrow \bot, q_{i+1} \leftarrow \top, \ldots, q_n \leftarrow \top])$. By $\leq$-monotonicity it is also filled in $f(x[q_0 \leftarrow \top, \ldots, q_n \leftarrow \top])$, and hence it is filled with the same value in all of the $f(x[q_0 \leftarrow \top, \ldots, q_{i-1} \leftarrow \top, q_i \leftarrow \bot, q_{i+1} \leftarrow \top, \ldots, q_n \leftarrow \top])$. Now, by $\wedge$-bistability, it should also be filled in $f(x)$ since

$$x = \bigwedge_{i=1\ldots n} x[q_0 \leftarrow \top, , \ldots, q_{i-1} \leftarrow \top, q_i \leftarrow \bot, q_{i+1} \leftarrow \top, \ldots, q_n \leftarrow \top]\ .$$

We show that if f is $\sqsubseteq$-monotonic and bistable, then it is observably sequential. That is, we have to show that if f has, say, q_0 as sequentiality index at (x, q'), then $q'\top \in$

$f(x[q_0 \leftarrow \top])$. Suppose not. Then, if y is such that $x \leq^s y$ and q' is filled in $f(y)$, we have $y \sqsubseteq y[q_0 \leftarrow \top]$ and $x[q_0 \leftarrow \top] \leq y[q_0 \leftarrow \top]$. Since by pointwise monotonicity q' is filled in $f(y[q_0 \leftarrow \top])$, there exists a sequentiality index q_1 for f at $(x[q_0 \leftarrow \top], q')$. Continuing in this way, we would exhaust all the $\bot$'s of x. So we get that q' is filled in, say, $f(x[q_0 \leftarrow \top, q_1 \leftarrow \top, \ldots, q_n \leftarrow \top])$, and hence by $\vee$-bistability in one of the $f(x[q_i \leftarrow \top])$. But it cannot be an $i > 0$ since q_0 is a sequentiality index at x, hence it is $i = 0$, which contradicts the assumption [1] .

Conversely, if f observably sequential, we first show that f is $\sqsubseteq$-monotonic. It is enough to check that if $x \leq^c y$ then $f(x) \leq^c f(y)$. To show this – actually, the natural thing to show is that more generally if $f \leq^c g$ and $x \leq^c y$ then $f(x) \leq^c f(y)$ –, we extend the definition of $\leq^c$ by congruence to all the terms involved in the rewriting system of Definition 3.8. It is straightforward to show, for each of the rules of the rewriting system, that if $t \leq^c t'$ and $t' \to t'_1$ then there exists t_1 such that $t \to t_1$. Informally, the only difference of behaviour is that $\langle\, g \parallel y \,\rangle$ might terminate before $\langle\, f \parallel x \,\rangle$, because of a new $\top$ in g or y. Notice that this argument is the same as the one used in the separation theorem of ludics (and, by the way, Girard's designs also form a bistable biorder) [12, 8].

Finally, we show that if f is observably sequential then it is bistable. One proves that f is $\leq$-monotonic much in the same way as for $\sqsubseteq$-monotonicity. The argument for $\wedge$-bistability is standard. One proves actually the preservation of all $\leq^s$ compatible binary $\leq^s$ greatest lower bounds (glb). Suppose that $x, y \leq^s z$ and $f(x \wedge^s y) <^s f(x) \wedge^s f(y)$. Let q' be accessible from $f(x \wedge^s y)$ and filled in $f(x)$ and $f(y)$. Let q be a sequentiality index at $x \wedge^s y, c'$. Then by sequentiality q is filled in both x and y, and with the same value since x, y have a $\leq^s$ upper bound. But then q is also filled in $x \wedge^s y$, contrary to the assumption. This shows a fortiori $\wedge$-bistability since $x \updownarrow y$ implies a fortiori that x, y have a $\leq^s$-upper bound, and that $x \wedge y = x \wedge^s y$.

As for $\vee$-bistability, suppose that $f(x) \vee f(y) < f(x \vee y)$. Let q' be accessible from $f(x) \vee f(y)$ and filled in $f(x \vee y)$. Then q' must be already accessible from $f(x \wedge y)$, as the order $\leq$ does not add new queries. Let q be the sequentiality index for q' at $x \wedge y$. Then we have that q is filled in $x \vee y$ by sequentiality, i.e. it is filled in either x or y, say,

[1] Here, and also in the next section, we make the simplifying assumption that the strategies are finite. We sketch here what adjustments have to be made to remove the finiteness restriction. This is rather standard domain theory: bistable biorders have to be directed complete, and bistable functions have to be Scott-continuous. We refer to [15] for the relevant definitions. With a little care, all our arguments go through, making use of the continuity assumption. For example, to prove observable sequentiality, we used the finiteness assumption when writing $x = x[q_0 \leftarrow \bot, \ldots, q_n \leftarrow \bot]$, with n finite. If instead we have

$$x = x[q_0 \leftarrow \bot, \ldots, q_n \leftarrow \bot, \ldots,]\ ,$$

then the proof of sequentiality goes through because by continuity the $\wedge$-bistability extends to preservation of greatest lower bounds of arbitrary subsets of a $\updownarrow$-component. For the proof of observable sequentiality, the finiteness of n was essential. But continuity saves us again: by continuity, we can take x, y finite, and we need only care about the q_j's that are filled in y, which are finitely many.

in x. By definition of $\leq$, since q is accessible from $x \wedge y$, q must be filled with $\top$ in x. But then q' is filled with $\top$ in $f(x)$ (and a fortiori in $f(x \vee y)$), by observable sequentiality: contradiction. □

Proposition 5.2 *The category of sds's and observably sequential algorithms is cartesian closed.*

PROOF. This is an easy corollary of Theorem 5.1, Proposition 4.2, and Theorem 3.10. Indeed, all we have to check for a full subcategory of a cartesian closed category to be itself cartesian-closed is that the (product and) function space construction of the larger category, when applied to objects of the smaller, yields an object of the smaller. Laird's function space $D^\top(\mathbf{S}) \to D^\top(\mathbf{S}')$ is the set of bistable and $\sqsubseteq$-monotonic functions, equipped with the pointwise and bistable orderings, which is isomorphic to $D^\top(\mathbf{S} \to \mathbf{S}')$. □

A direct proof of Proposition 5.2 is given in [6], but it is more complicated than the proof of Proposition 4.2.

6 Affine decomposition

In this section, we prove that affine sequential algorithms can also be defined as observably sequential functions that are affine with respect to the stable order, and hence as affine bistable and $\sqsubseteq$-monotonic functions (provided this makes sense, see below).

Definition 6.1 *An affine function is a function that preserves stable upper bounds of stably compatible elements.*

(The difference with linear functions, which may be more familiar to the linear logic oriented reader, is that preservation of $\perp$ is not required.)

Proposition 6.2 *Affine sequential algorithms are in order-isomorphic correspondence with affine observably sequential functions.*

PROOF. Let f be a sequential algorithm which is not affine. It contains at least a query of the form $r\,c'q_1\,r_1\,q_2$, where r_1 is not a prefix of q_2. Let x be the input strategy read off along the path from the root up to q_2. Formally, $x = \mathit{state}(\mathbf{r})$, where $\mathbf{r}$ is the projection of $r\,c'\,q_1\,r_1$ on the input sds. Let $y = (x \setminus \{r_1\}) \cup \{q_2\top\}$. Let q' be the projection of rc' on the output sds. Then q' is filled neither in $f(x)$ nor in $f(y)$, but is filled (with $\top$) in $f(x \vee^s y)$. Conversely, suppose that f is an affine sequential algorithm, and let $q'w' \in f(x \vee^s y)$. Let s be the position of f visited along the normalisation against $x \vee^s y$ towards $q'w'$ (it is obtained by travelling in f from the root starting with the initial move of q', and solving ambiguities using $x \vee^s y$ and $q'w'$ when going up after a player's move, cf. definition 3.8).

Consider the last input response r on this position of f. Then r belongs to either x or y, say x. But all other responses along the path are prefixes of r, by the constraint of affinity, hence all belong to x. Therefore $q'w' \in f(x)$ (or $q'w' \in f(y)$). □

However, bistable biorders are too poor to express the stable ordering. In further work, Laird [16] has defined the full subcategory of locally boolean domains in which not only the pointwise and bistable orders can be defined, but also the stable one [16] (we refer to this paper for the definition, which takes as primitive an operation $^{\neg}$ of involution which in terms of sds's consists in exchanging $\bot$' with $\top$'). In fact, Laird shows a *representation theorem*, by means of elegant abstract decompositions: every locally boolean domain is isomorphic to the set of strategies of some sds (and every such set is a locally boolean domain). Summing up, we end up with an equivalence (actually two equivalences) of categories.

Proposition 6.3 *The categories of sds's and (observably) sequential algorithms and of locally boolean domains and $\sqsubseteq$-monotonic and bistable functions are equivalent. This equivalence cuts down to an equivalence between the respective subcategories of affine morphisms (Definitions 3.5 and 6.1).*

Proof. Propositions 5.1 and 6.2 have established full embeddings. The equivalence of categories follows from the fullness on objects of the functors co-restricted to locally boolean domains. □

Since both in [17, 7] and [16] the comonads leading to models of (affine, intuitionistic) linear logic are defined as adjoint to an inclusion functor from the affine subcategory, the two decompositions are the same up to isomorphism. In other words, Laird's work provides a completely traditional domain-theoretic account of Berry-Cartwright-Curien-Felleisen-Lamarche's sequential algorithms model.

Finally, we mention two properties which add to the ambient symmetry:

Proposition 6.4 1. *Observably sequential functions are* costable, *i.e.:*

$$\forall x, y \quad (\exists z\ z \leq^c z \text{ and } z \leq^c y) \Rightarrow f(x \vee^c y) = f(x) \vee^c f(y)$$

2. *Affine observably sequential functions are* coaffine, *i.e. :*

$$\forall x, y \quad (\exists z\ z \leq^c z \text{ and } z \leq^c y) \Rightarrow f(x \wedge^c y) = f(x) \wedge^c f(y)$$

The proofs are not difficult and are in the same vein as the ones given above.

However, observably sequential functions fail to be cocontinuous, i.e., fail to preserve costable glbs of costable filters (see [18][Example 3.5.8] for a counter-example): this evidences the operational dissymmetry between $\top$ and $\bot$, since the latter is also used to denote non-termination.

We end the section with a brief sketch of how the concrete structure of sds's can be "abstracted" to that of locally boolean domain. More precisely, we exhibit Laird's decomposition first in concrete terms, and then more abstractly, thus suggesting some of the ideas underlying Laird's representation theorem mentioned above.

First, we observe that the compact primes (i.e. the elements x such that, for any Y, $x \leq^s \bigvee Y$ implies $x \leq^s y$ for some $y \in Y$) of $D^\top(\mathbf{S})$ are in one-to-one correspondence with the queries and the responses of $\mathbf{S}$:

$$r\left\{\begin{array}{l} c_1 \perp \\ \vdots \\ c_n \perp \end{array}\right. \qquad\qquad r\left\{\begin{array}{l} c_1 \perp \\ \vdots \\ c \top \\ \vdots \\ c_n \perp \end{array}\right.$$

$$\text{response} \quad (x = x \cap \neg x) \qquad \text{query} \quad (x \neq x \cap \neg x)$$

Next, any sds $\mathbf{S}$ can be written as

$$D^\top(\mathbf{S}) \approx \prod_i D_i$$

where

$$D_i \;=\; \{x \mid x \leq^c \{c_i\top\} \cup \{c\perp \mid c \in Q \text{ and } c \neq c_i\}\}$$

Abstractly:

$$D^\top(\mathbf{S}) \approx \prod_{\{p \mid p\ \leq^c-\text{maximal compact prime}\}} \{x \mid x \leq^c p\}$$

Each D_i is represented by an sds $\mathbf{S}_i$ (i.e., $D_i \approx D^\top(\mathbf{S}_i)$) that has a unique initial cell c_i. This property can be captured abstractly as:

$$\top \text{ is prime (and } \neq \perp)$$

The next step in the decomposition is to write

$$D_i = D_i' \cup \{\perp, \top\} \overset{\text{def}}{=} (D_i')^\uparrow$$

with $D_i' = D^\top(\mathbf{S}'_i)$, where $\mathbf{S}'_i$ is obtained from $\mathbf{S}_i$ by stripping off the unique initial cell c_i.

Finally, we decompose D_i' as follows:

$$D_i' \approx \Sigma_j D_{ij}$$

where

$$D_{ij} = \{x \in D_i \mid x \text{ starts with } v_j\} = \{x \mid x \geq^s \{v_j\bot\}\}$$

or, abstractly:

$$D_i' \approx \sum_{\{m \mid m\ \leq^s - \text{minimal}\}} \{x \mid x \geq^s m\}$$

Summing up, we have:

$$\begin{array}{lcll} D^\top(\mathbf{S}) & \approx & \prod_i D_i & \text{(product of domains indexed} \\ & & & \text{over } \leq^c - \text{maximal prime elements)} \\ & & & \\ D_i & = & (D_i')^\uparrow & \text{(lifting of predomain)} \\ & & & \\ D_i' & \approx & \sum_j D_{ij} & \text{(sum of domains indexed} \\ & & & \text{over } \leq^s - \text{minimal elements)} \end{array}$$

7 Conclusion

We should note that a (weaker) version of the full embedding was proved indirectly in Laird's paper [15]: he proves that the bistable model is fully abstract for the language $\Lambda_\bot^\top$ (the simply typed λ-calculus with a single base type, and two constants $\bot$ and $\top$ of base type). This calculus is essentially the same as Cartwright-Curien-Felleisen's language SPCF with respect to which the model of sequential algorithms is fully abstract [6]. Hence, by uniqueness up to isomorphism of fully abstract models, the two models are isomorphic *on the simply typed hierarchy.* Our result is more general and does not refer to (the interpretation of) types.

As for every correspondence result, there are mutual benefits in the equivalence that we have proved. The algorithmic side provides an operational explanation of the various orders. The abstract domain-theoretic side provides simpler proofs sometimes (for example, of cartesian closedness [2], and opens the way to extend the coverage of standard domain-theoretic tools for reasoning about sequential languages. Current work of Laird goes in that direction.

References

[1] R. Amadio and P.-L. Curien, Domains and Lambda-calculi, Cambridge University Press (1998).

[2]Note however that it is not known whether the subcategory of locally boolean domains is cartesian closed. We have used here the cartesian closedness of the larger category of bistable biorders.

[2] G. Berry, Stable models of typed lambda-calculi, Proc. ICALP 1978, Springer Lect. Notes in Comp. Sci. 62 (1978).

[3] G. Berry and P.-L. Curien, Sequential algorithms on concrete data structures, Theoretical Computer Science 20, 265-321 (1982).

[4] G. Berry and P.-L. Curien, Theory and practice of sequential algorithms: the kernel of the applicative language CDS, in Algebraic methods in semantics, M. Nivat, J. Reynolds eds, Cambridge Univ. Press, 35-87 (1985).

[5] R. Cartwright and M. Felleisen, Observable sequentiality and full abstraction, in Proc. POPL'92 (1992).

[6] R. Cartwright, P.-L. Curien, and M. Felleisen, Fully abstract semantics for observably sequential languages, Information and Computation 111 (2), 297-401 (1994).

[7] P.-L. Curien, On the symmetry of sequentiality, Proceedings of Mathematical Foundations of Programming Semantics 1993, Lecture Notes in Computer Science 802, 29-71 (1994).

[8] P.-L. Curien, Introduction to Linerar logic and ludics, parts I and II, Advances of Mathematics (China) 34 (5), 513-544 (2005), and 35 (1), 1-44 (2006).

[9] P.-L. Curien, Combinateurs catégoriques, algorithmes séquentiels et programmation fonctionnelle,Thèse d'Etat, Université Paris 7 (1983); English version: Categorical combinators, sequential algorithms and functional programming, Research notes in theoretical computer science, Pitman (1986); second, revised edition, Birkhaüser (1993).

[10] P.-L. Curien, Symmetry and interactivity in programming, Bulletin of Symbolic Logic, Vol. 9(2), 169-180 (2003).

[11] J.-Y. Girard, Linear logic, Theoretical Computer Sciences 50, 1-102 (1987).

[12] J.-Y. Girard, Locus Solum, Mathematical Structures in Computer Science (2001).

[13] G. Kahn and D. MacQueen, Coroutines and networks of parallel processes, in: Proc. IFIP'77, B. Gilchrist (ed.), North-Holland, pp.993-998 (1977).

[14] G. Kahn and G. Plotkin, Concrete domains, Theoretical Computer Science 12, 187-277 (1993).

[15] J. Laird, Bistability: an extensional characterization of sequentiality, in Proc. Computer Science and Logic 2003, Lecture Notes in Computer Science 2803, Springer (2003).

[16] J. Laird, Locally boolean domains, Theoretical Computer Science 342, 132 - 248 (2005).

[17] F. Lamarche, Sequentiality, games and linear logic, in Proc. CLICS Workshop, Aarhus University, DAIMI-397-II (1992).

[18] T. Loew, Locally boolean domains and universal models for infinitary sequential languages, PhD Thesis, Technische Universität Darmstadt (2006).

[19] T. Streicher, Laird domains, draft (2002).