(A) $\{\bot, a, b, c, d\}$, ordered as follows:

$$\bot \text{ minimum} \quad a \leq c, d \quad b \leq c, d$$

is a meet cpo which is not bounded complete.

(B) $\{\bot, a, b, c, d, e\}$, ordered as follows:

$$\bot \text{ minimum} \quad a \leq c, d \quad b \leq c, d \quad c, d \leq e$$

is not a meet cpo (condition (1) of definition 12.1.1 is violated).

(C) $\omega \cup \{a, b\}$, described in example 1.1.6 to give an example of a non-algebraic cpo, also fails to be a meet cpo (condition (2) of definition 12.1.1 is violated).

Figure 12.1: Meet cpo structure: example, and counter-examples

The following function from $\mathbf{T}^3$ to $\mathbf{O}$ due to Berry (and independently to Kleene, see section 14.1) is a stable function:

$$gustave(x, y, z) = \begin{cases} \top & \text{if } x = tt \text{ and } y = ff \\ \top & \text{if } x = ff \text{ and } z = tt \\ \top & \text{if } y = tt \text{ and } z = ff \\ \bot & \text{otherwise .} \end{cases}$$

The simplest way to verify that this function is stable is by checking that its minimal points, i.e., the mimimal elements (x, y, z) of $\mathbf{T}^3$ such that $gustave(x, y, z) = \top$ (see definition 12.2.1), are pairwise incompatible. Indeed, if $(x_1, y_1, z_1) \uparrow (x_2, y_2, z_2)$, $gustave(x_1, y_1, z_1) = \top$ and $gustave(x_2, y_2, z_2) = \top$, then (x_1, y_1, z_1) and (x_2, y_2, z_2) dominate the same minimal point (x, y, z) by the incompatibility of distinct minimal points, hence $(x, y, z) \leq (x_1, y_1, z_1) \wedge (x_2, y_2, z_2)$, and

$$gustave((x_1, y_1, z_1) \wedge (x_2, y_2, z_2)) = \top = gustave(x_1, y_1, z_1) \wedge gustave(x_2, y_2, z_2).$$

The main difficulty in getting a cartesian closed category of cm functions resides in making the evaluation morphism ev stable. The pointwise ordering $\leq_{ext}$ on functions does not work. Consider the identity function id, and the constant function $\top = \lambda x.\top$, both in $\mathbf{O} \to_{cm} \mathbf{O}$. Then $id \leq_{ext} \top$, so that if ev were to be stable, we should have that $ev(id, \bot)$ and $ev(id, \top) \wedge ev(\top, \bot)$ are equal. But in fact

$$ev(id, \top) \wedge ev(\top, \bot) = \top \wedge \top \neq \bot = ev(id, \bot).$$

To overcome the difficulty, Berry defined the stable ordering $\leq_{st}$. Actually, the goal of making ev cm forces the definition of $\leq_{st}$ given next. Indeed, suppose $f \leq_{st} g$ and $y \leq x$. Then we must have

$$f(y) = ev(f \wedge g, x \wedge y) = ev(f, x) \wedge ev(g, y) = f(x) \wedge g(y).$$

Definition 12.1.4 (stable ordering (cm)) *Let D and D' be two meet cpo's, and $f, f' : D \to_{cm} D'$. We define $f \leq_{st} f'$ by*

$$\forall\, x, x' \;\; (x \leq x' \Rightarrow f(x) = f(x') \wedge f'(x)).$$

In particular, if $f \leq_{st} g$, then $f \leq_{ext} g$ (take $x = x'$). (From now on, to avoid ambiguities, we use $_{ext}$ for the pointwise ordering, cf. definition 1.4.4.)

Lemma 12.1.5 *The relation $\leq_{st}$ of definition 12.1.4 is a partial order.*

PROOF. Reflexivity is obvious. For the transitivity, assume $f \leq_{st} f'$, $f' \leq_{st} f''$, and $x \leq x'$:

$$f(x) = f(x') \wedge f'(x) = f(x') \wedge f'(x') \wedge f''(x) = f(x') \wedge f''(x).$$

Antisymmetry follows from the antisymmetry of $\leq_{ext}$. □

Exercise 12.1.6 *Suppose that D, D' are meet cpo's and let $f, f' : D \to_{cm} D'$. (1) Show that $f \leq_{st} f'$ iff*

$$f \leq_{ext} f' \text{ and } \forall\, x, x' \;\; (x \uparrow x' \Rightarrow f(x) \wedge f'(x') = f(x') \wedge f'(x)).$$

(2) Show that if $f \uparrow_{st} f'$, then $\forall\, x, x' \;\; (x \uparrow x' \Rightarrow f(x) \wedge f'(x') = f(x') \wedge f'(x))$.

(3) Conversely, assuming that D' is a distributive meet cpo (see definition 12.1.12), show that if

$$f \uparrow_{ext} f' \text{ (for } \leq_{ext}) \text{ and } \forall\, x, x' \;\; (x \uparrow x' \Rightarrow f(x) \wedge f'(x') = f(x') \wedge f'(x))$$

then $f \uparrow_{st} f'$. Hint for (3): one uses exactly the information in the assumption to show that the pointwise lub of f, f' is their lub in the stable ordering, see the proof of theorem 12.1.13.

Exercise 12.1.7 *Show that if $f \leq_{ext} g \leq_{st} h$ and $f \leq_{st} h$, then $f \leq_{st} g$.*

Exercise 12.1.8 *Let $f, g : D \to E$, with f continuous, g cm, and $f \leq_{st} g$. Show that f is cm.*

Theorem 12.1.9 (cm - CCC) *The category of meet cpo's and conditionally multiplicative functions is a cpo-enriched CCC.*

PROOF. The verification that the composition of two cm functions is cm is immediate. As for the cpo-enriched CCC structure, we content ourselves with the verifications that $D \to_{cm} D'$, ordered by the stable ordering, is a meet cpo, and that the evaluation morphism ev is cm [1].

Directed lub's and binary compatible glb's are defined pointwise (and therefore the continuity property of $\wedge$ comes for free). Let $H \subseteq_{dir} D \to_{cm} D'$, and define h by $h(x) = \bigvee\{f(x) \mid f \in H\}$.

- h is cm:

$$h(x) \wedge h(y) = (\bigvee_{f \in H} f(x)) \wedge (\bigvee_{f \in H} f(y)) = \bigvee\{f(x) \wedge g(y) \mid f, g \in H\}.$$

We conclude by observing that $f(x) \wedge g(y) \leq k(x) \wedge k(y) = k(x \wedge y)$ if k is an upper bound of f, g in H.

- h is an upper bound of H. Let $f_0 \in H$ and $x \leq y$:

$$f_0(y) \wedge h(x) = \bigvee\{f_0(y) \wedge f(x) \mid f \in H\} = \bigvee\{f_0(x) \wedge f(y) \mid f \in H\} = f_0(x) \wedge h(y).$$

A similar argument shows that it is the least upper bound in the stable ordering.

If $f \uparrow_{st} g$, we define $f \wedge g$ by $(f \wedge g)(x) = f(x) \wedge g(x)$. We check $f \wedge g \leq_{st} f$. Let $x \leq y$:

$$\begin{array}{rcll} (f \wedge g)(y) \wedge f(x) & = & g(y) \wedge f(x) & = & g(x) \wedge f(y) \\ & = & g(y) \wedge f(x) \wedge g(x) \wedge f(y) & = & f(x) \wedge g(x) \end{array}$$

(cf. exercise 12.1.6). Suppose $k \leq_{st} f, g$. We show $k \leq_{st} f \wedge g$. Let $x \leq y$:

$$(f \wedge g)(x) \wedge k(y) = f(x) \wedge k(x) = k(x).$$

The stability of ev follows from the definition of the stable ordering:

$$\begin{array}{rcll} ev(f,x) \wedge ev(g,y) & = & f(x) \wedge g(y) & = & f(y) \wedge g(x) \\ & = & f(x) \wedge f(y) \wedge g(x) \wedge g(y) & = & ev(f \wedge g, x \wedge y)\,. \end{array}$$

□

Exercise 12.1.10 *Show that the following combination of order-theoretic compatible binary glb's (where (h,z) is an upper bound of $(f,x),(g,y)$), borrowed from [Tay90a], offers an attractive picture of the proof that ev is cm:*

$$\begin{array}{ccccc} (f \wedge g)(x \wedge y) & \to & (f \wedge g)(x) & \to & f(x) \\ \downarrow & (f \wedge g \text{ stable}) & \downarrow & (f \wedge g \leq_{st} f) & \downarrow \\ (f \wedge g)(y) & \to & (f \wedge g)(z) & \to & f(z) \\ \downarrow & (f \wedge g \leq_{st} g) & \downarrow & (\textit{definition}) & \downarrow \\ g(y) & \to & g(z) & \to & h(z) \end{array}$$

[1] In all the CCC's presented in this chapter, the pairing and currying are the set-theoretical ones (cf. exercises 4.2.11 and 4.2.12). We have written one of the proofs (theorem 12.2.8) in full detail.

(A) $\{\bot, a, b, c, d\}$, ordered as follows:

$$\bot \text{ minimum} \quad a, b, c \leq d$$

(B) $\{\bot, a, b, c, d\}$, ordered as follows:

$$\bot \text{ minimum} \quad a \leq d \quad b \leq c \leq d$$

Figure 12.2: Examples of non-distributive finite lattices

Exercise 12.1.11 *Show that the composition operation on meet cpo's is cm.*

If we assume bounded completeness of the domains, we are led to introduce distributivity to maintain cartesian closure.

Definition 12.1.12 (distributive cpo) *A cpo is called distributive if it is bounded complete (cf. definition 1.4.9) and satisfies*

$$\forall\, x, y, z \ \{x, y, z\} \textit{ compatible} \Rightarrow x \wedge (y \vee z) = (x \wedge y) \vee (x \wedge z).$$

Some counterexamples are given in figure 12.2.

Theorem 12.1.13 *The category of distributive meet cpo's and cm functions is a cpo-enriched CCC.*

PROOF. Let $f \uparrow_{st} g$. We show that $f \vee g$ defined pointwise is also the stable lub of f, g. Let $h(x) = f(x) \vee g(x)$.

- h is stable: On one end we have

$$h(x \wedge y) = (f(x) \wedge f(y)) \vee (g(x) \wedge g(y))$$

and on the other end we have

$$h(x) \wedge (f \vee g)(y) = (f(x) \vee g(x)) \wedge (f(y) \vee g(y)).$$

By distributivity we have

$$\begin{aligned}&(f(x) \vee g(x)) \wedge (f(y) \vee g(y)) = \\ &(f(x) \wedge f(y)) \vee (g(x) \wedge g(y)) \vee (f(x) \wedge g(y)) \vee (g(x) \wedge f(y))\ .\end{aligned}$$

The conclusion follows from the observation that $f \uparrow_{st} g$ implies$f(x) \wedge g(y) = g(x) \wedge f(y)$, hence $f(x) \wedge g(y), g(x) \wedge f(y) \leq f(x) \wedge f(y)$.

- h is a stable upper bound: Let $x \leq y$. We have

$$h(x) \wedge f(y) = (f(x) \wedge f(y)) \vee (g(x) \wedge f(y)) = f(x) \vee (f(x) \wedge g(y)) = f(x).$$

- h is the stable lub: Let $f, g \leq_{st} k$ and $x \leq y$. We have:

$$\begin{array}{rclcl} k(x) \wedge h(y) & = & (k(x) \wedge f(y)) \vee (k(x) \wedge g(y)) & = & (k(y) \wedge f(x)) \vee (k(y) \wedge g(x)) \\ & = & k(y) \wedge h(x) & = & h(x)\,. \end{array}$$

□

12.2 Stable Functions

Stable functions can be defined on arbitrary cpo's. Their definition brings us closer to an operational intuition.

Definition 12.2.1 (stable) *Let D and D' be cpo's. A function $f : D \to D'$ is called stable if it is continuous and if, for any $x \in D$, $x' \in D'$ such that $x' \leq f(x)$:*

$$\exists\, x_0 \leq x \;\; (x' \leq f(x_0) \text{ and } (\forall\, y \leq x \;\; (x' \leq f(y) \Rightarrow x_0 \leq y))).$$

This uniquely determined x_0 is written $m(f, x, x')$, and is called a minimal point of f (relative to x'). We write $D \to_{st} D'$ for the set of stable functions from D to D'. The following set is called the trace *of f:*

$$trace(f) = \{(x, x') \in D \times D' \mid x' \leq f(x) \text{ and } x = m(f, x, x')\}.$$

The function $m(f, \_, \_)$ is called the multi-adjoint *of f (the situation $x' \leq f(y)$ versus $m(f, x, x') \leq y$ is reminiscent of an adjunction).*

In computational terms, $m(f, x, x')$ represents the amount of x which is "read" by f in order to "write" (at least) x'. Stable functions can also be described by glb preservation properties, as we now explain.

Proposition 12.2.2 *1. Let D and D' be cpo's, and let $f : D \to_{st} D'$. Then for any bounded $X \subseteq D$ such that $\bigwedge X$ exists, $f(\bigwedge X) = \bigwedge f(X)$.*

2. Conversely, if D and D' have all non-empty bounded glb's, then a continuous function preserving all such glb's is stable.

PROOF. (1) $f(\bigwedge X)$ is a lower bound of $f(X)$ by monotonicity. Suppose that $z' \leq f(X)$. Let $y \geq X$. Then $z' \leq f(y)$. Let $z_0 = m(f, y, z')$. Pick $x \in X$. Then $z_0 \leq x$ by the minimality of z_0, since $z' \leq f(x)$ and $x \leq y$. Hence $z_0 \leq \bigwedge X$, and $z' \leq f(z_0) \Rightarrow z' \leq f(\bigwedge X)$. Hence $f(\bigwedge X)$ is the glb of $f(X)$.

(2) Let $y \leq f(x)$. Consider $z_0 = \{z \mid z \leq x \text{ and } y \leq f(z)\}$. We claim that z_0 is $m(f, x, y)$. This amounts to $y \leq f(z_0)$, which holds since

$$f(z_0) = \bigwedge\{f(z) \mid z \leq x \text{ and } y \leq f(z)\}.$$

□

In section 12.5, we shall see that stable functions preserve even more glb's (the connected ones, provided they exist). Meanwhile, going from "more" to "less", by proposition 12.2.2, stable functions on meet cpo's are conditonally multiplicative. Berry has provided the following example of a cm, non stable function. Let

$$D = \omega \cup \{\bot\} \text{ with } \bot \leq \cdots \leq n \leq \cdots \leq 1 \leq 0.$$

Let $f : D \to \mathbf{O}$ be defined by: $f(\bot) = \bot$, $f(n) = \top$. Then f is cm, but $m(f, 0, \top)$ does not exist. If we prevent the existence of infinite descending chains, then cm and stable are equivalent notions.

Proposition 12.2.3 *If D and D' are algebraic meet cpo's, and if $\mathcal{K}(D)$ is well-founded, then $f : D \to D'$ is stable iff it is cm.*

PROOF. Let f be cm. Consider $x' \leq f(x)$. By continuity, $x' \leq f(d)$ for some compact $d \leq x$. If d is not minimum with that property, then for some compact $d_1 \leq x$ we have $x' \leq f(d_1)$ and $d \not\leq d_1$. Hence $x' \leq f(d) \wedge f(d_1) = f(d \wedge d_1)$. In this way we construct a strictly decreasing chain $d > d \wedge d_1 > \cdots$ that must eventually end with e satisfying the definition of $m(f, x, x')$. □

The stable ordering between stable functions can be defined in terms of minimal points.

Definition 12.2.4 (stable ordering (stable)) *Let D, D' be cpo's, and $f, f' : D \to_{st} D'$. We write $f \leq_{st} f'$ iff*

$$f \leq_{ext} f' \text{ and } \forall x, x' \ (x' \leq f(x) \Rightarrow m(f, x, x') = m(f', x, x')).$$

Equivalently, $\leq_{st}$ can be defined by the inclusion of traces:

$$f \leq_{st} f' \quad \text{iff} \quad \mathit{trace}(f) \subseteq \mathit{trace}(f').$$

It is immediate that $\leq_{st}$ is a partial order, called the stable ordering. We write $f \uparrow_{st} g$ to mean that f, g are compatible with respect to $\leq_{st}$.

Exercise 12.2.5 *Show that the stable ordering can be equivalently defined as follows:* $f \leq_{st} f'$ *iff* $(f \leq_{ext} f'$ *and* $\forall x, x'\ (x' \leq f(x) \Rightarrow \forall y \leq x\ (x' \leq f'(y) \Rightarrow x' \leq f(y))))$.

The next lemma shows that the stable ordering just defined coincides with the stable ordering on the underlying cm functions.

Lemma 12.2.6 *Let* D *and* D' *be two cpo's, and* $f, f' : D \to_{st} D'$. *The following equivalence holds:*

$$f \leq_{st} f' \Leftrightarrow \forall x, x'\ (x \leq x' \Rightarrow f(x) = f(x') \wedge f'(x))$$

(in particular, the glb's $f(x') \wedge f'(x)$ *exist).*

PROOF. $(\Rightarrow)$ $f(x)$ is a lower bound of $\{f(x'), f'(x)\}$. If $z' \leq f(x')$ and $z' \leq f'(x)$, then $m(f, x', z') \leq x$, since $z' \leq f'(x)$ and by definition of $m(f', x', z')$. Hence $z' \leq f(x)$ since $y = m(f, x', z')$.

$(\Leftarrow)$ In particular, taking $x = x'$, we get $f \leq_{ext} f'$, hence $m(f', z, z') \leq m(f, z, z')$, for $z' \leq f(z)$. From the implication, we get that for any $z_1 \leq z$, $z' \leq f'(z_1)$ implies $z' \leq f(z_1)$, which shows $m(f, z, z') \leq m(f', z, z')$. □

Lemma 12.2.7 *Let* D, D' *be cpo's and let* $f : D \to_{st} D'$. *The following properties hold:*

1. If $\Delta' \subseteq_{dir} D'$, *and if* $\bigvee \Delta' \leq f(x)$, *then*

$$m(f, x, \bigvee \Delta') = \bigvee \{m(f, x, \delta') \mid \delta' \in \Delta'\}.$$

2. If D *and* D' *are bounded complete, and if* $x'_1 \leq f(x)$ *and* $x'_2 \leq f(x)$, *then*

$$m(f, x, x'_1 \vee x'_2) = m(f, x, x'_1) \vee m(f, x, x'_2)$$

(provided these lub's exist).

3. If D *and* D' *are algebraic, then* $f : D \to D'$ *is stable iff for any compact* $x \in \mathcal{K}(D)$, $x' \in \mathcal{K}(D')$, *such that* $x' \leq f(x)$, $m(f, x, x')$ *exists.*

PROOF. We only prove (3). Let $x \in D$, $x' \in D'$, not necessarily compact. We have:

$$\begin{array}{lcll} m(f, x, x') & = & \bigvee \{m(f, x, d') \mid d' \text{ compact and } d' \leq x'\} & \text{by (1)} \\ m(f, x, d') & = & m(f, d, d') \text{ for some } d & \text{by continuity.} \end{array}$$

□

Theorem 12.2.8 (stable - CCC) *The category of distributive meet cpo's*[2] *and stable functions is a cpo-enriched CCC (cf. definition 6.1.1).*

[2]Bounded completeness comes in to get products. Distributivity comes in to show that binary compatible lub's are stable.

PROOF. First of all, we need to check that the composition of two stable functions is stable. Let $f : D \to_{st} D'$ and $f' : D' \to_{st} D''$, and let $x'' \leq f'(f(x))$. Then we claim:

$$m(f' \circ f, x, x'') = m(f, x, m(f', f(x), x'')).$$

Indeed, for $y \leq x$, we have: $m(f', f(x), x'') \leq f(y)$ iff $x'' \leq f'(f(y))$. We leave the reader check that the set-theoretic product, with the componentwise ordering and the usual projections and pairing, is a categorical product. Notice that for $f : D \to_{st} D'$, $g : D \to_{st} D''$:

$$m(\langle f, g\rangle, x, (x', x'')) = m(f, x, x') \vee m(g, x, x'').$$

We check in detail that $D \to_{st} D'$ is a categorical exponent. We first show that $D \to_{st} D'$ is a cpo. Let $H \subseteq_{dir} D \to_{st} D'$. Then a fortiori H is directed for $\leq_{ext}$. Consider h defined by $h(x) = \bigvee\{f(x) \mid f \in H\}$. We check that h is stable by showing $m(h, x, x') = \bigvee_{f \in H} m(f, x, x' \wedge f(x))$, for all x, x' such that $x' \leq h(x)$. Let $y \leq x$. We have, using the continuity of glb's:

$$m(h, x, x') \leq y \Leftrightarrow x' \leq h(y) \Leftrightarrow \bigvee_{f \in H} (x' \wedge f(y)) = x'.$$

On the other hand we have

$$\bigvee_{f \in H} m(f, x, x' \wedge f(x)) \leq y \Leftrightarrow \forall f \in H \;\; x' \wedge f(x) \leq f(y)$$

which can be rephrased as: $\forall f \in H \;\; x' \wedge f(x) = x' \wedge f(y)$. Thus we are left to show:

$$\forall f \in H \;\; (x' \wedge f(x) = x' \wedge f(y) \Leftrightarrow \bigvee_{f \in H} (x' \wedge f(y)) = x').$$

($\Rightarrow$) $\bigvee_{f \in H}(x' \wedge f(y)) = \bigvee_{f \in H}(x' \wedge f(x)) = x' \wedge h(x) = x'$.

($\Leftarrow$) Let $f_0 \in H$. We have (cf. exercise 12.1.6):

$$\begin{array}{rclcl} x' \wedge f_0(x) & = & \bigvee_{f \in H}(x' \wedge f(y) \wedge f_0(x)) & = & \bigvee_{f \in H}(x' \wedge f(x) \wedge f_0(y)) \\ & = & x' \wedge f_0(y) \wedge h(x) & = & x' \wedge f_0(y)\ . \end{array}$$

Hence h is stable. Next we show: $\forall f \in H \;\; f \leq_{st} h$. Let $f \in H$ and $x' \leq f(x)$. Since $f \leq_{ext} h$, we have $m(h, x, x') \leq m(f, x, x')$. On the other hand, since $m(h, x, x') = \bigvee_{f \in H} m(f, x, x' \wedge f(x))$, we have

$$m(f, x, x') = m(f, x, x' \wedge f(x)) \leq m(h, x, x').$$

Finally let k be an upper bound of H in the stable order. We show $h \leq_{st} k$:

$$m(h, x, x') = \bigvee_{f \in H} m(f, x, x' \wedge f(x)) = \bigvee_{f \in H} m(k, x, x' \wedge f(x)) = m(k, x, x').$$

This completes the proof that $D \to_{st} D'$ is a cpo.

Binary compatible glb's exist, and are defined pointwise: for $f, g \leq_{st} h$, define $k(x) = f(x) \wedge g(x)$. This is a continuous function by continuity of the glb operation. Let $x' \leq k(x)$, and $y \leq x$. Then we have

$$(x' \leq k(y) \Leftrightarrow (m(f,x,x') \leq y) \quad \text{and} \quad (m(g,x,x') \leq y) \Leftrightarrow m(h,x,x') \leq y)$$

since $m(f,x,x')$ and $m(g,x,x')$ are both equal to $m(h,x,x')$ by assumption. Hence $m(k,x,x') = m(h,x,x')$. Thus k is stable, $k \leq_{st} f$, and $k \leq_{st} g$. Finally, suppose $k' \leq_{st} f, g$. Then $m(k',x,x') = m(f,x,x') = m(g,x,x')$, hence $m(k',x,x') = m(k,x,x')$. This completes the proof that $k = f \wedge g$ (with respect to the stable ordering). The continuity property $f \wedge (\bigvee H) = \bigvee_{h \in H}(f \wedge h)$ follows from the fact that the operations are defined pointwise.

Binary compatible lub's exist too. Suppose $f, g \leq_{st} h$, and define $k(x) = f(x) \vee g(x)$. The proof that k is stable and is the lub of f, g in the stable ordering is completely similar to the proof of directed completeness $D \to_{st} D'$. One replaces everywhere uses of the continuity of the glb operation by uses of its distributivity. The distributivity equation follows from the fact that the operations are defined pointwise. Thus we have proved that $D \to_{st} D'$ is a distributive meet cpo.

We now prove that ev is stable. Consider (f,x) and x' such that $x' \leq f(x) = ev(f,x)$. We show that $m(ev,(f,x),x') = (g,z)$, where $z = m(f,x,x')$ and $g = \lambda y.x' \wedge f(y \wedge z)$. (By bounded completeness, all binary glb's exist, thus g is well-defined and continuous, cf. lemma 12.1.2.) First, $z \leq x$ by definition. We check $g \leq_{st} f$. We have (for $y \leq y_1$)

$$g(y_1) \wedge f(y) = x' \wedge f(y_1 \wedge z) \wedge f(y) = x' \wedge f(y_1 \wedge z \wedge y) = g(y).$$

Second, we check $x' \leq g(z)$. We actually even have $x' = g(z)$:

$$g(z) = x' \wedge f(z \wedge z) = x' \wedge f(z) = x'.$$

Finally, let $(f_1, x_1) \leq (f,x)$ be such that $x' \leq f_1(x_1)$. Then a fortiori $x' \leq f(x_1)$, hence $z \leq x_1$. Next we show $g \leq_{ext} f_1$:

$$\begin{array}{rcl@{\qquad}rcl} g(y) \wedge f_1(y) & = & x' \wedge (f(y \wedge z) \wedge f_1(y)) & = & x' \wedge f_1(y \wedge z) \\ & = & x' \wedge f_1(y \wedge z) \wedge f(x_1) & = & x' \wedge (f(y \wedge z) \wedge f_1(x_1)) \\ & = & (x' \wedge f_1(x_1)) \wedge f(y \wedge z) & = & g(y)\,. \end{array}$$

Finally, we prove $g \leq_{st} f_1$. Let $y \leq y_1$:

$$g(y_1) \wedge f_1(y) = x' \wedge f(y_1 \wedge z) \wedge f_1(y) = x' \wedge f_1(y_1 \wedge z) \wedge f(y) \leq x' \wedge f(y_1 \wedge z) \wedge f(y) = g(y).$$

This completes the proof that $m(ev,(f,x),x') = (g,z)$, and hence that ev is stable.

Next we show that $\Lambda(f)(x)$ is stable. Let $m(f,(x,x'),x'') = (y,y')$. We show that $m(\Lambda(f)(x),x',x'') = y'$. Since $(y,y') \leq (x,y')$, we have $x'' \leq f(x,y')$, that is, $x'' \leq \Lambda(f)(x)(y')$. If $z' \leq x'$ and $x'' \leq \Lambda(f)(x)(z')$, then $(y,y') \leq (x,z')$, and in particular $y' \leq z'$. This proves the stability of $\Lambda(f)(x)$, and also that $\Lambda(f)$ is monotonic: if $x \leq x_1$, then

$$m(\Lambda(f)(x),x',x'') = m(\Lambda(f)(x_1),x',x'') = y'.$$

Finally, we check that $\Lambda(f)$ is stable. We show, for $g \leq_{st} \Lambda(f)(x)$:

$$m(\Lambda(f),x,g) = \bigwedge T \quad \text{where } T = \{y \mid y \leq x \text{ and } g \leq_{st} \Lambda(f)(y)\}.$$

We have to check that $g \leq_{st} \Lambda(f)(\bigwedge T)$. For any x', since $g(x') \leq f(y,x')$ for any $y \in T$, we have by stability (cf. proposition 12.2.2)

$$g(x') \leq \bigwedge_{y \in T} f(y,x') = f(\bigwedge T, x')$$

i.e., $g \leq_{ext} \Lambda(f)(\bigwedge T)$. By exercise 12.1.7, $g \leq_{st} \Lambda(f)(x)$ and $\Lambda(f)(\bigwedge T) \leq_{st} \Lambda(f)(x)$ imply $g \leq_{st} \Lambda(f)(\bigwedge T)$. Thus we have established the CCC structure.

Finally, we check that Λ is monotonic. Suppose $f \leq_{st} g$. We first show $\Lambda(f) \leq_{ext} \Lambda(g)$, i.e., $\forall\, x\ \Lambda(f)(x) \leq_{st} \Lambda(g)(x)$. Recall that $m(\Lambda(f)(x),x',x'') = y'$, where y' is the second component of $m(f,(x,x'),x'')$. Since $f \leq_{st} g$, we have $m(g,(x,x'),x'') = m(f,(x,x'),x'')$. Hence

$$m(\Lambda(f)(x),x',x'') = m(\Lambda(g)(x),x',x'').$$

Next, suppose $y \leq x$. We have to check $\Lambda(f)(y) = \Lambda(f)(x) \wedge \Lambda(g)(y)$. By the pointwise definition of binary compatible glb's, this amounts to $f(y,x') = f(x,x') \wedge g(y,x')$, which holds since $f \leq_{st} g$. □

Exercise 12.2.9 (trace factorisation [Tay90b]) *Show that trace(f), ordered by the induced componentwise order, is a cpo. Consider the following functions:*

$$\begin{array}{lll} \pi : \mathit{trace}(f) \to D & \pi(x,x') & = x \\ \pi' : \mathit{trace}(f) \to D' & \pi'(x,x') & = x' \\ h : D \to \mathit{trace}(f) & h(x) & = \{(m(f,x,f(x)),f(x)) \mid x \in D\}\,. \end{array}$$

(1) Show that $\pi \dashv h$, i.e., $(x_1,y_1) \leq h(x) \Leftrightarrow x_1 \leq x$. (2) A monotonic function $f : X \to Y$ between two partial orders is called a fibration if for any pair (x,y) such that $y \leq f(x)$, there exists an element $\omega(f,x,y)$ of D such that:

$$\begin{array}{ll} (\Phi_0) & \omega(f,x,y) \leq x \\ (\Phi_1) & f(\omega(f,x,y)) = y \\ (\Phi_2) & \forall\, z \leq x\ \ (f(z) \leq y \Rightarrow z \leq \omega(f,x,y))\,. \end{array}$$

Show that $\pi' : \mathit{trace}(f) \to D'$ is a stable fibration, by which we mean that it is stable, and that it is a fibration with $\omega(f,\_,\_)$ as multi-adjoint. (Equivalently, a stable fibration

can be defined as a fibration (with ω), which is stable (with m) and is such that all fibers are groupoids, i.e., all subsets $f^{-1}(x)$ consist of non comparable points.) (3) Show that the sets

$\mathcal{M}$ *of functions with a left adjoint and*
$\mathcal{E}$ *of stable fibrations*

form a factorisation system for stable functions, by which we mean: (a) Any stable function f factorises as $f = \pi' \circ h$, with $h \in \mathcal{M}$ and $\pi' \in \mathcal{E}$. (b) $\mathcal{M}$ and $\mathcal{E}$ contain the order-isomorphisms and are closed under composition with order-isomorphisms. (c) For every commuting square $g \circ h = l \circ f$ where $h \in \mathcal{M}$ and $l \in \mathcal{E}$, there exists a unique stable ϕ (called diagonal fill-in) such that $l \circ \phi = g$ and $\phi \circ h \leq f$. (The unique diagonal fill-in property allows us to show the uniqueness of the $\mathcal{E}$-$\mathcal{M}$ factorisation.)

Exercise 12.2.10 *Show that the category of cpo's and stable functions is not cartesian. Hints: consider example (B) in figure 12.1 (ahead). Call this domain D and define a pair of functions $f : D \to \mathbf{O}$ and $g : D \to \mathbf{O}$ such that the pairing fails to be stable.*

Exercise 12.2.11 ** Develop a theory of stable, partial functions by analogy with the continuous case. Discuss lifting and sum in this framework.*

12.3 dI-domains and Event Structures

We now address algebraicity. In continuous domain theory, the compact functions are finite lub's of step functions $d \to e$ (cf. proposition 1.4.8). Step functions are stable, but they do not serve as approximations of functions as in the continuous case. In the continuous case, one simply has $(d \to e) \leq_{ext} f$ iff $e \leq f(d)$. However it is not true in general that $e \leq f(d)$ (or even $m(f, d, e) = d$) implies $(d \to e) \leq_{st} f$. The point is that for $e_1 < e$, one may have $m(f, d, e_1) < d$, which precludes $(d \to e) \leq_{st} f$, since $m(d \to e, d, e_1) = d$. This suggests to "saturate" our candidate $d \to e$ by forming a lub $(d \to e) \vee \cdots \vee (d_i \to e_i) \vee \cdots$, with $e_i < e$ and $d_i = m(f, d, e_i)$. To ensure the finiteness of this saturation process, one is lead to assume the following property I, which may be read as "finite is really finite".

Definition 12.3.1 (dI-domain) *Let D be an algebraic cpo. Property I is defined as follows:*

(I) Each compact element dominates finitely many elements.

An algebraic, bounded complete and distributive cpo satisfying property I is called a dI-domain.

Exercise 12.3.2 *Show that an algebraic domain satisfies I iff each compact element dominates finitely many compact elements. Hint: for any $y \leq x$ the approximants of y are also approximants of x, hence are finitely many.*

Clearly, property I implies well-foundedness, hence under the assumption that property I is satisfied stable functions are the same as cm functions.

The dI-domains are due to Berry, who showed that they form a cartesian closed category. In fact, distributivity is not needed. We take a concrete approach, based on event structures. An event structure can be perceived as the specification of how to build data out of distinct discrete pieces, or events, respecting consistency and causality requirements. These intuitions come from the theory of concurrency, and, indeed, event structures have been investigated in connection with Petri nets. Winskel [Win80, Win86] noticed that they could be used for domain theory, and this is what concerns us here. Any event structure generates a cpo, and dI-domains are recast from a subclass of event structures satisfying an axiom corresponding to distributivity.

Definition 12.3.3 (event structure) *An event structure $(E, Con, \vdash)$ (E for short) is given by:*

- *a set E whose elements are called events,*
- *a non-empty predicate $Con \subseteq \mathcal{P}_{fin}(E)$, called* consistency*, satisfying:*

$$(\subseteq Con) \quad (X \in Con \text{ and } Y \subseteq X) \Rightarrow Y \in Con.$$

- *a relation $\vdash \subseteq Con \times E$, called the* enabling *relation; if $X \vdash e$, we say that X is an enabling of e.*

Enablings serve to define proof trees for events. A proof of an event e is a tree labelled by events, formed recursively as follows. If $\vdash e$, then e is a proof of e. If $t_1, \ldots, t_n$ are proofs of $e_1, \ldots, e_n$, and if $\{e_1, \ldots, e_n\} \vdash e$, then the tree formed by placing a root labelled with e above $t_1, \ldots, t_n$ is a proof of e.

A state *(or configuration) of an event structure E is a subset x of E which is:*

- consistent*, that is, $\forall X \subseteq_{fin} E \;\; X \in Con$*
- safe*, that is, for any $e \in x$ there exists a proof tree for e whose nodes are all in x.*

We write $D(E, Con, \vdash)$ for the collection of states, ordered by inclusion.

An event structure is called stable *if for any state x, for any X,Y, and e such that $e \in x$, $X \subseteq_{fin} x$, $Y \subseteq_{fin} x$, then $X \vdash e$ and $Y \vdash e \Rightarrow X = Y$.*

A partial order is called (stable) event domain *if it is generated by some (stable) event structure, i.e., if it is isomorphic to $D(E, Con, \vdash)$ for some (stable) event structure $(E, Con, \vdash)$.*

The stability condition on event structures allows us to define the glb of two compatible states as their set-theoretic intersection.

Proposition 12.3.4 *Event domains are Scott domains satisfying property I. The minimum element is the empty state which is indifferently written $\perp$ or $\emptyset$. Stable event domains are dI-domains.*

PROOF. Let E be an event structure. We first show that $D(E, Con, \vdash)$ (D for short) is a bounded complete cpo. Let Δ be a directed set of states, and consider its set-theoretic union x. We prove that x is a state. Let $X \subseteq_{fin} x$. Then, by directedness, $X \subseteq_{fin} \delta$ for some $\delta \in \Delta$. Hence $X \in Con$. Safety is obvious for a union. Let now x, y, z be states such that $x, y \leq z$. The set-theoretic union of x and y is again a state, by the same argument. The algebraicity of D follows from the observation that finite states are compact, and that every state x is the union of the finite states underlying the proof trees of the events $e \in x$. Moreover the compact states are exactly the finite ones, from which property I follows.

Let us now assume that E is stable. Distributivity follows from the set-theoretic distributivity of intersection over union, since the binary compatible glb of two states is its intersection, thanks to the condition of stability. □

We shall see that in fact dI-domains and stable event domains are the same (proposition 12.3.10).

Example 12.3.5 *Consider $E = \{e_1, e_2, e_3, e_4\}$, and $\vdash$ given by*

$$\vdash e_1 \quad \vdash e_2 \quad \{e_1, e_2\} \vdash e_3 \quad \{e_1, e_2\} \vdash e_4.$$

And consider the two following consistency predicates Con_1 and Con_2, described by their maximal elements:

$$\{e_1, e_2, e_3\} \in Con_1 \quad \{e_1, e_2, e_4\} \in Con_1$$
$$\{e_1, e_2, e_3, e_4\} \in Con_2 \ .$$

Then $(E, Con_1, \vdash)$ is a stable event structure, and $(E, Con_2, \vdash)$ is an event structure which is not stable (consider $\{e_1, e_2, e_3, e_4\}$).

Where does the consistency predicate and the enabling come from? We let them arise from the consideration of a of a stable function f, viewed as a state (anticipating theorem 12.3.6).

- If $(d_1, e_1), (d_2, e_2) \in trace(f)$ and if $d_1, d_2 \leq d$, then $e_1, e_2 \leq f(d)$. Therefore $\{(d_1, e_1), (d_2, e_2)\}$ should not be consistent if $d_1 \uparrow d_2$ and $e_1 \not\asymp e_2$.
- If $(d_1, e), (d_2, e) \in trace(f)$ (with $d_1 \neq d_2$), then $d_1 \not\asymp d_2$ by definition of a stable function. Therefore $\{(d_1, e_1), (d_2, e_2)\}$ should not be consistent if $d_1 \uparrow d_2$, $d_1 \neq d_2$, and $e_1 = e_2$.

- Let $(d, e) \in \mathit{trace}(f)$ and $e_1 < e$. Then $e_1 \leq f(d)$ implies $(m(f, d, e_1), e_1) \in \mathit{trace}(f)$. Thus the should be closed under some form of enabling, such that $(m(f, d, e_1), e_1)$ occurs in the proof of (d, e) in $\mathit{trace}(f)$ (cf. the discussion on step functions at the beginning of the section.)

Theorem 12.3.6 (event domains - CCC) *The category of event domains and stable (or cm) functions is a cpo-enriched CCC.*

PROOF. We only give the construction of the product and exponent objects. Specifically, given E and E', we construct $E \times E'$ and $E \to E'$ in such a way that $D(E \times E')$ is the product of $D(E)$ and $D(E')$ in **Poset**, and that $D(E \to E') = D(E) \to_{st} D(E')$. We define $E \times E'$ as follows:

- The collection of events is the disjoint union of E and E'.
- Consistency is defined componentwise: X is consistent iff both $X \cap E$ and $X \cap E'$ are consistent.
- The enabling relation is the disjoint union of the component enabling relations.

We define $E \to E'$ as follows:

- Events are pairs (x, e') where x is a finite state of E, and $e' \in E'$.
- A finite set $\{(x_i, e'_i) \mid i \in I\}$ is consistent iff

$$\forall J \subseteq I \ \{x_j \mid j \in J\} \text{ bounded} \Rightarrow \{e'_j \mid j \in J\} \in \mathit{Con}, \text{ and}$$
$$\forall i, j \ e'_i = e'_j \Rightarrow (x_i = x_j \text{ or } x_i \not\uparrow x_j).$$

- $\{(x_i, e'_i) \mid i \in I\} \vdash (x, e')$ iff $\forall i \ x_i \subseteq x$ and $\{e'_i \mid i \in I\} \vdash e'$.

Axiom ($\subseteq \mathit{Con}$) is trivially satisfied. We show that there is an order-isomorphism between $D(E) \to_{st} D(E')$ ordered by the stable ordering and $D(E \to E')$ ordered by inclusion. With $f : D(E) \to_{st} D(E')$ we associate[3]

$$\mathit{trace}(f) = \{(x, e') \mid e' \in f(x) \text{ and } (y \leq x \Rightarrow e' \notin f(y))\}.$$

We show that $\mathit{trace}(f)$ is a state. Consider $\{(x_i, e'_i) \mid i \in I\} \subseteq \mathit{trace}(f)$ and $J \subseteq I$ such that $\{x_j \mid j \in J\}$ has a bound x. Then $\{e'_j \mid j \in J\} \subseteq f(x)$, hence is consistent. The second condition follows from the definition of stable function. As for safety, consider $(x, e') \in \mathit{trace}(f)$ and a proof of e' in $f(x)$. We can attach to any node e'_1 in this proof the minimal point under x where f reaches e'_1. In this way we obtain a proof of (x, e') in $\mathit{trace}(f)$.

[3]This definition of trace is a variation of the one given in definition 12.2.1, taylored to event structures.

That $f \leq_{st} g$ is equivalent to $trace(f) \subseteq trace(g)$ is just the definition of the stable ordering (cf. definition 12.2.4; see also lemma 12.2.7(3)). The converse transformation is defined as follows. Given a state z of $E \to E'$, we define

$$fun(z)(x) = \{e' \mid \exists y \ (y, e') \in z \text{ and } y \subseteq x\}.$$

We first have to check that $fun(z)(x)$ is a state. Its consistency follows from the first condition in the definition of higher-order consistency. Its safety follows from the safety of z, noticing that all the nodes in a proof of $(y, e') \in z$ have the form (y_1, e'_1), with $y_1 \leq y$. The definition of $fun(z)$ ensures that it is continuous (notice that the y in the right hand side is finite). As for the stability, suppose that y_1 and y_2 are minimal under x relative to e'. Then by the definition of $fun(z)$, it must be the case that $(y_1, e'), (y_2, e') \in z$, hence $y_1 = y_2$ by the definition of higher-order consistency.

Finally, it is easy to check that *trace* and *fun* are inverse bijections. □

The distributivity plays no role in the proof of theorem 12.3.6. But it can be added without harm.

Theorem 12.3.7 (dI-domains - CCC) *The category* **dI-Dom** [4] *of stable event domains and stable functions is a cpo-enriched CCC.*

PROOF HINT. Check that $E \to E'$ is stable if E and E' are stable. □

What we lack at this point are representation theorems, in the style of theorem 10.2.12, giving an abstract order-theoretic characterisation of event domains and stable event domains. Droste [Dro89] has provided a representation theorem for event domains (adapted from [Win80]). We present this material in the form of a (difficult) exercise, which relies on the following definition.

Definition 12.3.8 *In a partial order, a prime interval is defined as a pair of elements x, y such that $x \prec y$, i.e., $x < y$ and $\nexists z \ x < z < y$.*

Prime intervals capture the intuition of an event as a discrete increment.

Exercise 12.3.9 ** Show that the event domains are the algebraic cpo's which satisfy I as well as the following two axioms on compact elements:*

$$(C) \quad (x \prec y, x \prec z, y \neq z, y \uparrow z) \Rightarrow (y \vee z \text{ exists}, y \prec y \vee z, z \prec y \vee z)$$
$$(S) \quad [x, x'] \asymp [y, y'], x \leq y \Rightarrow x' \leq y' \ .$$

In axiom (S), $[x, x']$ stands for a prime interval, and $\asymp$ stands for the reflexive, symmetric and transitive closure of the relation $[x, y] \prec [z, y \vee z]$ (where x, y, z satisfy the

[4] This name will be justified by proposition 12.3.10.

assumptions of (C)). The idea is to take as events the equivalence classes of prime intervals. Hints: If x, y are compact and $x \leq y$, there exists $x = z_0 \prec \cdots \prec z_n \prec y$. Such a sequence is called a chain from x to y. If $z_0, \ldots, z_m$ and $z'_0, \ldots, z'_{n'}$ are two chains from x to y, then for any equivalence class e of prime intervals $\sharp\{i \mid [z_i, z_{i+1}] \in e\} = \sharp\{j \mid [z'_j, z'_{j+1}] \in e\}$. Show the following implication: $x \prec x' \leq y \prec y' \Rightarrow \neg([x, x'] \asymp [y, y'])$.

If distributivity is assumed, then the characterisation is much friendlier: the stable event domains are exactly dI-domains.

Proposition 12.3.10 *The following classes of cpo's coincide:*

1. *stable event domains,*
2. *dI-domains,*
3. *coprime algebraic Scott domains (cf. definition 10.2.1) satisfying I.*

PROOF. (1) $\Rightarrow$ (2) This is the second statement of proposition 12.3.4.

(2) $\Rightarrow$ (3) Let D be a dI-domain.We use the characterisation given in proposition 10.4.3, and show that the compact elements of D are finite lub's of compact coprime elements. We follow a proof of Zhang [Zha91]. We first claim:

Claim. The compact coprimes are those compact elements that cover exactly one element.

To prove the claim, we notice that by property I, for any compact d, $\{e \mid e \prec d\}$ is finite, and if $d_1 \prec d$, $d_2 \prec d$ and $d_1 \neq d_2$, then we must have $d_1 \vee d_2 = d$, and hence d is not coprime. Conversely, if d covers exactly one element d_1, let $d \leq \bigvee X$ for a finite bounded X. By distributivity we get $d = \bigvee\{d \wedge x \mid x \in X\}$. Pick $x \in X$. If $d \wedge x \neq d$, by property I we can find an element covered by d above $d \wedge x$, which by assumption means $d \wedge x \leq d_1$. Hence at least one $d \wedge x$ must be such that $d \wedge x = d$ (and hence d is coprime) as otherwise we would have $d = \bigvee\{d \wedge x \mid x \in X\} \leq d_1$.

Now we show that any compact element $d \neq \bot$ is a lub of finitely many compact coprimes. Consider the tree rooted at d formed by taking as sons of the root all the distinct elements $d_1, \ldots, d_n$ covered by d if there are at least two such elements, and continuing so recursively. Notice that $d_i \neq \bot$ for all i, otherwise we would have

$$d_i = \bot \prec d$$
$$d_i = \bot < d_j < d \quad (\text{picking } j \neq i).$$

Then d is the lub of all the leaves of the tree, which are coprime since they cover exactly one element.

(3) $\Rightarrow$ (1) Let D be as in (3). We define $(E, \leq, \vdash)$ as follows.

E consists of the compact coprime elements of D,
Con consists of the finite bounded subsets of E,
$\{e \mid e \prec d\} \vdash d$for any $d \in E$, i.e., the unique enabling of e is $\{e \mid e \prec d\}$.

This is clearly an event structure, and the uniqueness of enablings makes it a fortiori stable. We show that D is order-isomorphic to $D(E, Con, \vdash)$. To $x \in D$ we associate $g(x) = \{e \mid e \text{ compact coprime and } e \leq x\}$: $g(x)$ is consistent since it is bounded, and it is safe since by property I any event has a unique finite proof tree. Conversely, to any state $y \in D(E, Con, \vdash)$ we associate $\bigvee y$ which exists by bounded completeness. The composition $\bigvee \circ g$ is the identity of D by definition of coprime-algebraic. If $e' \leq \bigvee y$, then $e' \leq e$ for some $e \in y$ since e' is compact coprime. Then, by the definition of enabling, e' occurs in the proof tree of e and is therefore in y by safety. Hence $g \circ \bigvee$ is the identity on $D(E, Con, \vdash)$. □

Special classes of stable event structures are those of Girard's qualitative domains, and of Girard's coherence spaces. Coherence spaces will be discussed at length in section 13.1.

Definition 12.3.11 (qualitative domain) *Qualitative domains are event domains all of whose events are initial: $\vdash = \{\vdash e \mid e \in E\}$. Then, clearly, $Con = \mathcal{K}(D(E))$. If moreover Con is specified by means of a reflexive and symmetric relation $\frown$, i.e., $(X \in Con \Leftrightarrow \forall x, y \in X \;\; x \frown y)$, then we say that we have a coherence space (see definition 13.1.1).*

Exercise 12.3.12 *Show that the qualitative domains are the dI-domains in which the compact coprime elements p are atomic, i.e., $\bot \prec p$.*

Exercise 12.3.13 *Show that the category of qualitative domains and stable functions is a cpo-enriched CCC.*

Exercise 12.3.14 *Show that the dI-domains are the distributive cpo's such that the finite stable projections (see definition 12.4.2 ahead) form a directed set (with respect to the stable ordering) which has as lub the identity. Use this characterisation to give another proof that the category of dI-domains and stable functions is cartesian closed. Hints: consider for each $X \subseteq_{fin} \mathcal{K}(D)$ the projection defined by $p(x) = \bigvee\{d \wedge x \mid d \in X\}$; proceed as in the proof of proposition 5.2.4.*

Exercise 12.3.15 (stable neighborhoods [Zha91]) *Let D be an ω-algebraic meet cpo satisfying property I. (1) Characterise $\{f^{-1}(\top) \mid f : D \to_{st} \mathbf{O}\}$. (2) Such sets are called stable neighborhoods. Prove that they are closed by intersection but not by union. (3) Show that there is no topology for which the stable functions are continuous. Hint: consider the stable functions from $\mathbf{O} \times \mathbf{O}$ to $\mathbf{O}$. There are four possible choices of a topology for $\mathbf{O}$; show that for each choice the sets of stable and continuous functions do not coincide. (4) Characterise stable functions as those functions that preserve stable neighborhoods by inverse image.*

Exercise 12.3.16 *Show that property I may not be preserved by the function space construction with the pointwise ordering. Hints: take $(\omega_\bot \to \mathbf{O}) \to \mathbf{O}$; define $f_n(x) = \top$ iff $x \leq n$, and consider the step functions $f_n \to \top$.*

Exercise 12.3.17 *(1) If D is a complete ω-algebraic lattice with property I, show: $(D \to_{st} \mathbf{O}) \cong \mathcal{K}(D)$, with the flat ordering on $\mathcal{K}(D)$. (2) On the other hand, given a flat cpo $E_\perp$, show that $(E_\perp \to_{st} \mathbf{O}) \cong (\mathcal{P}(E), \subseteq) + \mathbf{O}$, where $+$ is the coalesced sum (cf. definition 1.4.22).*

12.4 Stable Bifinite Domains *

We investigate the stable version of bifiniteness. Stable projections are better behaved than the continuous ones. Stable bifinite domains enjoy a characterisation similar to that for bifinite domains (cf. theorem 5.2.7). They lend themselves to a simple theory of retractions. In particular, there is a retraction of all retractions. Also, there exists a universal bifinite domain.

Proposition 12.4.1 *Let D be a meet cpo, and let $p, q : D \to_{st} D'$ be such that $p, q \leq_{st} id$. Then:*

1. *$p \circ q = p \wedge q$.*
2. *p is a projection.*
3. *$im(p)$ is downward closed.*

PROOF. (1) Remark first that since p and q are bounded their glb exists. Next observe

$$qd \leq d \Rightarrow p(qd) = pd \wedge qd = (p \wedge q)(d).$$

(2) For $p = q$ we obtain from (1): $p(pd) = pd \wedge pd = pd$.

(3) $d \leq pd' \Rightarrow pd = p(pd') \wedge d = pd' \wedge d = d$ □

Proposition 12.4.1 justifies the following definition.

Definition 12.4.2 (stable projection) *Let D be a meet cpo. A stable function $p : D \to_{st} D$ such that $p \leq_{st} id_D$ is called a stable projection. If moreover $im(p)$ is finite, p is called finite. A stable injection-projection pair is an injection-projection pair whose projection is a stable projection.*

Now we define stable bifinite domains (cf. definition 5.2.2).

Definition 12.4.3 (stable bifinite) *A meet cpo D is called a stable bifinite domain if the finite stable projections $p : D \to_{st} D$ form a directed set which has as lub the identity (in the stable ordering). We call $\mathbf{Bif}_\wedge$ the category of stable bifinite domains and stable functions.*

Proposition 12.4.4 (stable bifinites-CCC) *1. Stable bifinite domains are algebraic and satisfy property I. The compact elements are those of the form $p(x)$, where p is a finite stable projection.*

2. The category $\mathbf{Bif}_\wedge$ of stable bifinites and stable functions is cartesian closed.

PROOF. Cf. the proof of proposition 5.2.4.

(1) The satisfaction of I follows from proposition 12.4.1 (3).

(2) All we have to do is to check that if p, q are finite stable projections, then both $p \times q$ and r defined by $r(f) = \lambda x.q(f(p(x)))$ are stable projection.

- $(d, e) \leq (d', e') \Rightarrow (p(d), q(e)) = (p(d') \wedge d, q(e') \wedge e) = (p(d'), q(e')) \wedge (d, e)$.
- We have to show that $f \leq_{st} g$ implies $r(f) = r(g) \wedge id$, that is, for all d:

$$qn(f(p(d))) = q(g(p(d))) \wedge f(d).$$

We set $\alpha(d) = q(f(p(d)))$ and $\beta(d) = q(g(p(d))) \wedge f(d)$. Observe:

$$\begin{array}{rcl} f(p(d)) \leq f(d), q \leq_{st} id & \Rightarrow & q(f(p(d))) = q(f(d)) \wedge f(p(d)) \\ f(d) \leq g(d), q \leq_{st} id & \Rightarrow & q(f(d)) = q(g(d)) \wedge f(d) \\ p(d) \leq d, f \leq_{st} g & \Rightarrow & f(p(d)) = f(d) \wedge g(p(d)) \,. \end{array}$$

Therefore: $\alpha(d) = q(g(d)) \wedge f(d) \wedge g(p(d))$. On the other hand:

$$g(p(d)) \leq g(d), q \leq_{st} id \quad \Rightarrow \quad \beta(d) = q(g(d)) \wedge g(p(d)) \wedge f(d).$$

So $\alpha(d) = \beta(d)$. □

Exercise 12.4.5 *A stable bifinite domain D is called a stable ω-bifinite domain if it is ω-algebraic. (1) Show that D is a stable ω-bifinite domain iff id is the lub of a countable increasing chain of finite stable projections. (2) Show that the stable ω-bifinite domains form a full sub cartesian closed category of* $\mathbf{Bif}_\wedge$.

Characterisation of stable bifinites. The main result here is a characterisation of the objects in $\mathbf{Bif}_\wedge$. Roughly they are algebraic meet cpo's satisfying a combination of properties M (definition 5.2.6) and I that we call $(MI)^\infty$. In first approximation, the combined property $(MI)^\infty$ consists in asking that the iteration of the operator that computes the mub's and the operator that computes the principal ideals on a finite collection of compacts returns a finite collection. It is convenient to decompose property I in simpler properties.

Definition 12.4.6 *Let D be a cpo. We define the three properties I_1, I_2, and I_3 as follows:*

(I_1) *Every decreasing sequence of compacts is finite:*

$$(\{x_n\}_{n\in\omega} \subseteq \mathcal{K}(D) \text{ and } \forall\, n \in \omega \;\; x_n \geq x_{n+1}) \Rightarrow \{x_n\}_{n\in\omega} \text{ is finite.}$$

(I_2) *Every increasing sequence of compacts under a compact is finite:*

$$(\{x\} \cup \{x_n\}_{n\in\omega} \subseteq \mathcal{K}(D) \text{ and } \forall\, n \in \omega \;\; x_n \leq x_{n+1} \leq x) \Rightarrow \{x_n\}_{n\in\omega} \text{ finite.}$$

(I_3) *The immediate predecessors of a compact are in finite number:*

$$x \in \mathcal{K}(D) \Rightarrow pred(x) \text{ is finite}$$

where $pred(x) = \{y \in D \mid y \prec x\}$

Proposition 12.4.7 *Let D be an algebraic cpo. Then it has property I iff it has properties I_1, I_2, and I_3.*

Proof. ($\Rightarrow$) Observe that $\{x_n\}_{n\in\omega}$ and $pred(x)$ are contained in $\downarrow d$ (where d is an appropriately chosen compact).

($\Leftarrow$) Let $d \in \mathcal{K}(D)$. First observe that $\downarrow d \subseteq \mathcal{K}(D)$. If there is a non compact element x under d, then $\downarrow x \cap \mathcal{K}(D)$ is directed since D is an algebraic cpo, and $\bigvee(\downarrow x \cap \mathcal{K}(D)) = x$. So we can build an infinite strictly ascending chain under d, contradicting I_2. Property I_2 also implies that $pred(d)$ is complete in the sense that

$$e < d \Rightarrow \exists\, e' \in pred(d)\ \ e \leq e' < d.$$

Otherwise we can again build a strictly growing chain under d. Now define

$$X_0 = \{d\} \qquad X_{n+1} = \bigcup\{pred(x) \mid x \in X_n\} \cup X_n.$$

Then:

$$\begin{array}{ll} \bigcup_{n\in\omega} X_n = \downarrow d \text{ and } \exists\, n \in \omega\ \ X_{n+1} = X_n & \text{(by property } I_1\text{)} \\ \forall\, x \in \mathcal{K}(D)\ \ pred(x) \text{ is finite} & \text{(by property } I_3\text{)}. \end{array}$$

Hence all X_n's are finite, which implies that $\downarrow d$ is finite[5]. □

Figure 12.3 presents typical situations where property I fails.

Lemma 12.4.8 *1. If D is an algebraic cpo satisfying property I_1, then $\mathcal{K}(D) \models m$ (cf. definition 5.2.6).*

2. If D is an algebraic meet cpo such that $\mathcal{K}(D) \models m$, then D is an L-domain (cf. definition 5.2.11.

3. Stable bifinite domains are L-domains.

Proof. (1) Given any upper bound y of X, there exists a compact $y' \leq y$ that is also an upper bound for X. By the property I_1 there exists $y'' \leq y'$ such that $y'' \in MUB(X)$. Otherwise we could build an infinite decreasing chain under y'.

(2) Let $A \subseteq \mathcal{K}(D)$ and $x \in UB(A)$, and suppose $y_1, y_2 \leq x$ and $y_1, y_2 \in MUB(A)$. Then $y_1 \wedge y_2 \in MUB(A)$, which forces $y_1 = y_2$. We then conclude by exercise 5.2.13.

(3) This follows immediately from (1) and (2), since stable bifinite domains are algebraic and satisfy property I by proposition 12.4.4. □

[5]This is the contrapositive of König's lemma adapted to directed acyclic graphs.

(A) I_1 fails for $\{\bot\} \cup \underline{\omega}$, with $\underline{\omega} = \{\underline{n} \mid n \in \omega\}$, ordered as follows:

$$\bot \text{ minimum} \quad (\underline{m} \leq \underline{n} \text{ iff } n \leq m)$$

(B) I_2 fails for $\omega \cup \{\infty, a\}$, ordered as follows:

$$x \leq y \text{ iff } y = a \text{ or } (y = \infty \text{ and } x \in \omega \cup \{\infty\}) \text{ or } (x, y \in \omega \text{ and } x \leq y)$$

(C) I_3 fails for $\omega \cup \{\bot, a\}$, ordered as follows:

$$\forall\, n \in \omega \;\; \bot \leq n \leq a$$

Figure 12.3: Failure of property I

As a consequence, the operator U (cf. theorem 5.2.7) is idempotent for stable bifinite domains (cf. proposition 5.2.15). This indicates that a more liberal operator than U has to be introduced in order to characterise stable bifiniteness in a way similar to the characterisation of bifiniteness. We have already exploited the fact that images of projections are downward closed. This should motivate the following definition.

Definition 12.4.9 (property $(MI)^\infty$) *Let $(P, \leq)$ be a poset, and let $X \subseteq_{fin} P$. We set $U{\downarrow}\,(X) = U({\downarrow}\,(X))$. Let $(U{\downarrow})^\infty(X)$ be the least set containing X and closed with respect to the $U{\downarrow}$ operator. We say that X has property $(MI)^\infty$ if $(U{\downarrow})^\infty(X)$ is finite. If D is an algebraic meet cpo, then we say that D has property $(MI)^\infty$ if*

$$\forall\, X \subseteq_{fin} \mathcal{K}(D) \;\; X \textit{ has property } (MI)^\infty.$$

Let D be an algebraic meet cpo. If D has property $(MI)^\infty$ then it also has property I and property M, as if $x, y \in \mathcal{K}(D)$ then

$$\downarrow x \subseteq (U{\downarrow})^\infty(\{x\}) \quad \text{and} \quad U(\{x, y\}) \subseteq (U{\downarrow})^\infty(\{x, y\}).$$

The converse does not hold: see example 12.4.13.

Theorem 12.4.10 *A cpo D is a stable bifinite iff D is an algebraic meet cpo with property $(MI)^\infty$.*

PROOF. Cf. the proof of theorem 5.2.7. ($\Rightarrow$) If $X \subseteq_{fin} \mathcal{K}(D)$, then $X \subseteq p(D)$ for some finite stable projection. The argument in the proof of theorem 5.2.7 yields not only $U(X) \subseteq p(D)$, but also $(U{\downarrow})(X) \subseteq p(D)$.

($\Leftarrow$) Let $A \subseteq_{fin} \mathcal{K}(D)$, and consider p_A defined by $p_A(y) = \bigvee((U{\downarrow})^\infty(A) \cap {\downarrow}\, y)$. Notice the use of the $(U{\downarrow})$ operator, instead of U. The fact that $B = (U{\downarrow})^\infty(A)$ is

downward closed serves in establishing $p_A \leq_{st} id$, $(A \subseteq B \Rightarrow p_A \leq_{st} p_B)$, and that p_A is stable. We only check $p_A \leq_{st} id$. Let $x \leq y$. We check $p_A(y) \wedge x \leq p_A(x)$. Since $p_A(D) = (U\downarrow)^\infty(A)$ is downward closed, we have $p_A(y) \wedge x \in (U\downarrow)^\infty(A)$, hence $p_A(y) \wedge x \leq p_A(X)$ by definition of p_A. □

This characterisation of stable bifinites allows us to prove that the category of event domains is a full subcategory of the category of stable bifinites.

Exercise 12.4.11 *Show that any event domain is stable bifinite. Hints: In a Scott domain, all glb's exist, and* $U(X) = \{\bigvee Y \mid Y \subseteq_{fin} X \text{ and } Y \text{ bounded}\}$, *for all finite* X. *Show that* $(U\downarrow)^\infty(X) \subseteq \bigcup X$, *for any* n.

We now list without proof some results from [Ama91a] towards the goal of a Smyth like theorem: is $\mathbf{Bif}_\wedge$ the maximum cartesian closed full subcategory of ω-algebraic meet cpo's and stable functions? It turns out that properties M, I_1 , and I_2 are necessary to enforce the ω-algebraicity of function spaces. One can also show that property I_3 is necessary under a rather mild hypothesis. The necessity of property $(MI)^\infty$ is still open. In the first place, a stable version of theorem 5.2.17 holds: in any full subcategory of algebraic meet cpo's if the terminal object, the product, and the exponent exist then they coincide up to isomorphism with the ones defined in $\mathbf{Cpo}_\wedge$. The proof is basically the same as in the continuous case. Here is a summary of the results in [Ama91a]

- If D and $D \to_{st} D$ are ω-algebraic meet cpo's, then D has properties M, I_1 and I_2. (Property M is not necessary if we do not ask for the countability of compact elements.)
- If D and $D \to_{st} D$ are ω-algebraic meet cpo's and, for each $d \in \mathcal{K}(D)$, $\downarrow d \to_{st} \downarrow d$ is an ω-algebraic meet cpo, then D has property I_3.

The following properties serve as stepping stones in the proof. If D and $D \to_{st} D$ are ω-algebraic meet cpo's, then:

If $d \in D$, then $\downarrow d$ is an ω-algebraic lattice.
If $d \in \mathcal{K}(D)$ and $\downarrow d$ is distributive, then $\downarrow d$ is finite.
If $\downarrow d$ is distributive for each $d \in \mathcal{K}(D)$, then D has property I.

Of the two following examples 12.4.12 and 12.4.13, the first, due to Berry, illustrates a situation where I_2 does not hold, while the second shows that $M+I$ does not imply $(MI)^\infty$.

Example 12.4.12 *Let* D *be example (B) from figure 12.3. This domain is a well-founded chain, hence* $D \to_{st} D = D \to_{cm} D = D \to_{cont} D$. *We claim that any continuous function* h *such that* $h(a) = a$ *is compact. If* $h \leq_{st} \bigvee K$, *then*

$$a = h(a) \leq \bigvee\{k(a) \mid k \in K\}$$

hence $a = k(a)$ *for some* $k \in K$, *by compactness of* a. *We prove the following subclaim:*

$$(h(a) = a\ ,\ k(a) = a\ ,\ h \uparrow_{st} k) \Rightarrow h = k.$$

Indeed, $k(x) = k(x) \wedge h(a) = k(a) \wedge h(x) = h(x)$. By the subclaim we have $h \in K$, which proves a fortiori that h is compact. But the set $\{h \mid h(a) = a\}$ is not countable (any monotonic function from natural numbers to natural numbers yields an h in the set, and a diagonalisation argument can be applied). Therefore $D \to D$, ordered by the stable ordering, is not ω-algebraic.

Example 12.4.13 *Let $D = \{\bot\} \cup \omega_B \cup \omega_T$ where $\omega_B = \{n_B \mid n \in \omega\}$ and $\omega_T = \{n_T \mid n \in \omega\}$, ordered as follows:*

$$\begin{array}{l} \bot \text{ is the minimum,} \\ n_B \leq n_T, (n+1)_T \ (n \geq 0) \\ n_B \leq (n-1)_T \ (n \geq 1) \ . \end{array}$$

Observe that $(U{\downarrow})^\infty(\{i_T\}) = D$ and that i_T is compact. If D were bifinite there would exist a finite stable projection p_n such that $p_n(i_T) = i_T$, which implies $(U{\downarrow})^\infty(\{i_T\}) \subseteq im(p_n)$. Contradiction. As a matter of fact this example also shows that the iteration of the $U{\downarrow}$ operator does not need to collapse at any finite level.

Example 12.4.13 also serves to illustrate the case of a compact function with an infinite trace. Let D be as in this example. Clearly $trace(id_D)$ is infinite. We show that id_D (id for short) is a compact element of the functional space. We write $f =_k id$ if $\forall\, i \leq k \ \ f(i_B) = i_B$ and $f(i_T) = i_T$. We first claim:

$$f =_k id, f \leq_{st} id \Rightarrow f =_{k+1} id.$$

This follows from

$$f \leq_{st} g, (k+1)_B \leq k_T \Rightarrow f((k+1)_B) = k_T \wedge (k+1)_B = (k+1)_B$$

($f((k+1)_T) = (k+1)_T$ is proved similarly).

$k_B \leq f((k+1)_T)$: $f \leq_{st} g, k_B \leq (k+1)_T \Rightarrow k_B = f(k_B) = f((k+1)_T) \wedge k_B$.

$(k+1)_B \leq f((k+1)_T)$: $(k+1)_B = f((k+1)_B) \leq f((k+1)_T)$.

$f((k+1)_T) \leq (k+1)_T$: This follows obviously from $f \leq_{st} id$.

Applying the claim repetitively, we get that if $f =_0 id$ (and $f(\bot) = \bot$) and $f \leq_{st} id$, then $f = id$. Suppose now that $id = \bigvee \Delta$ (cf. remark 12.4.20). Then we may choose $f \in \Delta$ such that $f =_0 id$, hence $f = id$, and id is compact.

A retraction of all retractions. Scott [Sco80] has shown that the collection of finitary retractions (cf. definition 7.4.1) over a bounded complete algebraic cpo D is the image of a finitary retraction over the space $D \to_{cont} D$ of continuous functions. In the stable case Berardi [Ber91] was apparently the first to observe that when working over dI-domains the image of a stable retraction is still a dI-domain. It was then possible to adapt Scott's technique to show that the set of retractions over a dI-domain is itself the (image of a) retraction. We shall give the corresponding of Berardi's result for stable bifinites. The proof exploits the fact that stable bifinites can be described as directed colimits of stable projections. A retraction of all retractions serves to provide a model for a type theory with a type of all types (see exercise 11.3.5).

Proposition 12.4.14 *Let D be a stable bifinite and rD be a stable retraction over D. Then $r(D)$ is a stable bifinite.*

PROOF. Let $p : D \to_{st} D$ be a finite projection. Define $q = r \circ p \circ r$. We have $q \leq_{st} r \circ id \circ r = r$; moreover $im(q)$ is finite. Moreover, since the lub of the p's is id, the lub of the q's is r. □

We give a simple proof of the fact that the collection $Ret(D)$ of the stable retractions over a stable bifinite D is a retract of its functional space $D \to_{st} D$. The keyvault of the construction is to observe that given $f : D \to_{st} D$, with $im(f)$ finite, there is a natural way to associate to f a retraction, namely iterate f a finite number of times. First we recall a simple combinatorial fact.

Lemma 12.4.15 *Let X be a set and let $f : X \to X$, with $im(f)$ finite. Then $\sharp\{f^k \mid k \geq 1\} \cap Ret(X) = 1$.*

PROOF. First observe $\forall\, k \geq 1 \;\; im(f^{k+1}) \subseteq im(f^k)$. Since $im(f)$ is finite the following h is well defined: $h = \min\{k \geq 1 \mid im(f^{k+1}) = im(f^k)\}$. Hence the restriction of f to $im(f^h)$ is a permutation (being a surjection from $im(f^k)$ onto itself). Let $n = \sharp im(f^h)$: then $(f^h)^{n!}$ is the identity on $im(f^h)$, and therefore is a retraction over X. As for the uniqueness observe that if $f^i \circ f^i = f^i$ and $f^j \circ f^j = f^j$ for $i, j \geq 1$ then $f^i = f^{ij} = f^j$. □

Lemma 12.4.16 *Let D be a stable bifinite domain. Then for any $f : D \to_{st} D$ and any $p \leq_{st} id$:*

$$\sharp\{(f \circ p \circ f)^k \mid k \geq 1\} \cap Ret(D) = 1.$$

PROOF. The finiteness of $im(p)$ implies the finiteness of $im(f \circ p \circ f) \subseteq f(im(p \circ f))$, and the conclusion then follows from lemma 12.4.15. □

Lemma 12.4.17 *If D is a meet cpo, then $Ret(D)$ is a meet cpo (with the order induced by $D \to_{st} D$).*

PROOF. Analogous to the continuous case. □

Theorem 12.4.18 *Given a stable bifinite D the collection $Ret(D)$ of stable retractions is a retract of the functional space $D \to_{st} D$.*

PROOF. In the hypotheses of lemma 12.4.16 we write

$$f_p = f \circ p \circ f \qquad k_p = \sharp im(p)! \,.$$

Note that k_p is a multiple of the least k such that $f_p^k \in Ret(D)$, and is independent from f. The crucial remark is that

$$r \in Ret(D) \Rightarrow r_p \in Ret(D)$$

because by the definition of stable order, for any x:

$$r_p \leq_{st} r \circ r = r, r_p(x) \leq r(x) \Rightarrow r_p(r_p(x)) = r_p(r(x)) \wedge r(r_p(x)) = r_p(x).$$

Notice that the form of f_p has been precisely chosen to have $r_p(rd) = r_p d$ and $r(r_p d) = r_p d$. We define $\rho : (D \to_{st} D) \to_{st} Ret(D)$ as follows:

$$\rho(f) = \bigvee_{p \leq_{st} id} (f_p)^{k_p}.$$

We check that this definition is correct. First, we observe, for $p \leq_{st} q$:

$$(f_p)^{k_p} = (f_p)^{k_p k_q} \leq_{st} (f_q)^{k_p k_q} = (f_q)^{k_q}.$$

It follows that $\{(f_p)^{k_p} \mid p \leq_{st} id\}$ is directed. Hence $\rho(f)$ is defined and is a retraction, since the join of a directed set of retractions is a retraction. Also, ρ is a retraction, because

$$r \in Ret(D) \Rightarrow \rho(r) = \bigvee_{p \leq_{st} id} (r_p)^{k_p} = \bigvee_{p \leq_{st} id} r_p = r \circ r = r.$$

Next we show that ρ preserves binary compatible glb's. Suppose $f, g \leq_{st} h$. Since the composition operation is cm (cf. exercise 12.1.11), we have

$$\begin{array}{rclcl} \rho(f \wedge g) & = & \bigvee_{p \leq_{st} id}((f \wedge g)_p)^{k_p} & = & \bigvee_{p \leq_{st} id}(f_p \wedge g_p)^{k_p} \\ & = & \bigvee_{p \leq_{st} id}(f_p)^{k_p} \wedge (g_p)^{k_p} & = & (\bigvee_{p \leq_{st} id}(f_p)^{k_p}) \wedge (\bigvee_{p \leq_{st} id}(g_p)^{k_p}) \\ & & & = & \rho(f) \wedge \rho(g)\ . \end{array}$$

It remains to show that ρ preserves directed sets. Let H be a directed set in $D \to_{st} D$. We have

$$\begin{array}{l} (\bigvee H)_p = (\bigvee H) \circ p_p \circ (\bigvee H) = \bigvee_{h \in H}(h \circ p_p \circ h) = \bigvee_{h \in H} h_p \\ (\bigvee_{h \in H} h_p)^{k_p} = \bigvee_{h \in H}(h_p)^{k_p}\ . \end{array}$$

Hence

$$\rho(\bigvee H) = \bigvee_{p \leq_{st} id} ((\bigvee H)_p)^{k_p} = \bigvee_{p \leq_{st} id} \bigvee_{h \in H} (h_p)^{k_p} = \bigvee_{h \in H} \bigvee_{p \leq_{st} id} (h_p)^{k_p} = \bigvee_{h \in H} \rho(h).$$

□

Exercise 12.4.19 *1. Let D be a meet cpo and suppose that p is a stable projection. Show that if D is an (ω-)algebraic meet cpo (stable bifinite) then $im(p)$ is an (ω-)algebraic meet cpo (stable bifinite).*

2. Show that if D is a stable bifinite then $Prj(D) =\downarrow (id_D)$ is a stable bifinite and a lattice.

Exercise 12.4.20 *Show that the identity is always a maximal element in the stable ordering. (In particular, the only stable closure is the identity.)*

Exercise 12.4.21 *Let D be the cpo of example 12.1(A). Show that it is not the case that $Prj(D)$ is (the image of) a projection of $D \to_{st} D$. Hints. Consider p_1, p_2, f defined by*

$$p_1(x) = \begin{cases} a & \text{if } x \geq a \\ \bot & \text{otherwise} \end{cases} \qquad p_2(x) = \begin{cases} b & \text{if } x \geq b \\ \bot & \text{otherwise} \end{cases} \qquad f(x) = \begin{cases} d & \text{if } x = c \\ x & \text{otherwise}\ . \end{cases}$$

Show that p_1, p_2 are stable projections, that f is stable, and that $p_1, p_2 \leq_{st} f$. Suppose that $\pi : (D \to_{st} D) \to_{st} Prj(D)$ is a projection. Then $p_1, p_2 \leq_{st} \pi(f) \leq_{st} f$. Derive a contradiction by showing that there is no stable projection p such that $p_1, p_2 \leq_{st} p$ other than id, using that $MUB(\{a, b\}) = \{c, d\}$.

We end the section with a brief account of universality in the stable bifinite framework [DR93].

Proposition 12.4.22 *Let $\mathbf{Bif}_\wedge{}^{ip_s}$ be the category whose objects are the ω-algebraic cpo's which are stable bifinite domains and whose arrows are the stable injection-projection pairs (notation: $(i, j) : D \to_{ip_s} D'$). The following properties hold:*

1. $\mathbf{Bif}_\wedge{}^{ip_s}$ is an ω-algebroidal category and the collection of compact objects has the amalgamation property.

2. $\mathbf{Bif}_\wedge{}^{ip_s}$ has a universal homogeneous object.

PROOF HINT. (1) The proof follows the pattern of the one for the continuous case. Let us just show that $\mathbf{Bif}_\wedge{}^{ip_s}$ has the amalgamation property. Consider three finite posets $(E, \leq), (D_1, \leq_1), (D_2, \leq_2)$ with functions $(h_i^+, h_i^-) : E \to_{ip_s} D_i$, $(i \in \{1, 2\})$, in $\mathbf{Bif}_\wedge{}^{ip_s}$. Without loas of generality we may assume $E = D_1 \cap D_2$. Then

$$\forall\, e, e' \in E \;\; e \leq e' \Leftrightarrow (e \leq_1 e' \text{ and } e \leq_2 e').$$

Now we define the amalgam as $F = E \cup (D_1 \backslash E) \cup (D_2 \backslash E)$. It is helpful to recall that E is downward closed in D_i, so we define:

$$f \leq_F f' \Leftrightarrow \exists\, i \in \{1, 2\} \;\; f, f' \in D_i \text{ and } f \leq_i f'.$$

We are left with the definition of the morphisms $(k_i^+, k_i^-) : D_i \to_{ip_s} F$ $(i \in \{1, 2\})$. Take the inclusions for k_i^+. Define:

$$k_1^-(f) = \begin{cases} f & \text{if } f \in D_1 \\ h_2^-(f) & \text{otherwise .} \end{cases}$$

k_2^- is defined symmetrically.

(2) By theorem 7.3.11. □

Exercise 12.4.23 *Prove that $\mathbf{Cpo}_\wedge$ has limits of ω^{op}-diagrams. By analogy with what was done in chapter 7, study the representation problem of the functors over $\mathbf{Bif}_\wedge{}^{ip_s}$ as stable function over $Prj(U)$, where U is some universal (homogeneous) domain. Show that product and exponent are representable.*

12.5 Connected glb's *

Following Taylor [Tay90a], we focus on a characterisation of stable functions by the property of preservation of all connected glb's. This leads to another cartesian closed category of stable functions, exploiting a different kind of distributivity (of connected

glb's over directed lub's), whereas in the previous section we had focused on distributivity of binary glb's over binary compatible lub's. In section 12.6, we investigate the objects of the latter category in more depth.

First we introduce the notions of connected set and of connnected meet cpo. These notions are related with those of L-domain and of continuous dcpo investigated in chapter 5.

Definition 12.5.1 (connected) *Let X be a partial order. We say that $Y \subseteq X$ is connected if for any two points x, y of Y there exists a zigzag between them in Y, that is, $x = z_0 \star z_1 \star \cdots \star z_n = y$, where $\star$ stands for $\leq$ or $\geq$, and where $z_i \in Y$ for all i.*

The notion of zigzag induces a natural equivalence relation over any subset $Y \subseteq X$: for x, y in Y, write $x \approx y$ if there exists a zigzag from x to y in Y. The equivalence classes for this relation can be seen as the disjoint connected components of Y.

Proposition 12.5.2 *If X is a connected partial order, then its Alexandrov topology is locally connected, i.e., every open is a disjoint union of connected opens. If D is a connected dcpo, then its Scott topology is locally connected.*

PROOF. First note that if Y is upper closed, then the connected components are also upper closed, and that if X is a cpo and if Y is Scott open, then the connected components are also Scott open.

Let U, V be opens such that $U \cap V = \emptyset$ and $Y \subseteq U \cup V$. Suppose that $x = z_0 \star z_1 \star \cdots \star z_n = y$ is a zigzag in Y from $x \in U$ to $y \in V$. Let i be such that $z_i \in U$ and $z_{i+1} \in V$. Then, since U, V are upper closed, either $z_i \in V$ or $z_{i+1} \in U$, contradicting $U \cap V = \emptyset$. Conversely, if Y cannot be divided in components, then it has only one equivalence class for the zigzag relation, i.e., it is connected in the graph-theoretic sense. □

Lemma 12.5.3 *A partial order X has compatible binary glb's iff any zigzag, viewed as a collection of points, has a glb.*

PROOF. By induction on the length of the zigzag, in the notation of definition 12.5.1. If the last $\star$ is $\leq$, then clearly $z_0 \wedge \cdots \wedge z_n = z_0 \wedge \cdots \wedge z_{n-1}$; if it is $\geq$, then $z_0 \wedge \cdots \wedge z_{n-1}$ and z_n both have z_{n-1} as an upper bound, hence $z_0 \wedge \cdots \wedge z_n = (z_0 \wedge \cdots \wedge z_{n-1}) \wedge z_n$ exists. □

Definition 12.5.4 *In a partial order X, a multilub of a subset $Y \subseteq X$ is a set J of upper bounds of Y that is multiversal, i.e., such that any upper bound x of Y dominates a unique element of J.*

Proposition 12.5.5 *For a partial order, the following properties are equivalent:*

1. *All compatible binary glb's and codirected glb's exist.*
2. *All connected glb's exist.*
3. *All $\downarrow x$'s have all glb's.*
4. *All $\downarrow x$'s have all lub's.*
5. *All subsets have multilub's.*

We call such partial orders L partial orders.

PROOF. (1) $\Rightarrow$ (2) Let $Y \subseteq X$ be connected. Let Z be the set of the glb's of all finite zigzags in Y (Z is well defined by lemma 12.5.3). Clearly, if Z has a glb, then its glb is also the glb of Y. Thus it is enough to show that Z is codirected. Let $z_0 \wedge \cdots \wedge z_n \in Z$ and $z'_0 \wedge \cdots \wedge z'_{n'} \in Z$. Then by connectedness one may build a zigzag between z_n and z'_0. Then the glb of the zigzag obtained by joining these three zigzags is in Z and is a lower bound of $z_0 \wedge \cdots \wedge z_n$ and $z'_0 \wedge \cdots \wedge z'_{n'}$.

(2) $\Rightarrow$ (3) Let $Y \subseteq\downarrow x$. Then $Y \cup \{x\}$ is connected, hence has a glb in X, which is the same as the glb of Y (this includes the limit case $Y = \emptyset$).

(3) $\Rightarrow$ (1) Let x_1, x_2 be a bounded pair. Its glb exists in $\downarrow x$, for any upper bound x of x_1, x_2, and is their glb in X. For codirected glb's, notice that if Y is codirected, and $x \in Y$, then Y and $Y \cap \downarrow x$ have the same glb if any.

(3) $\Leftrightarrow$ (4) For a partial order, having all glb's is equivalent to having all lub's.

(4) $\Rightarrow$ (5) Let $Y \subseteq X$. Consider the collection Z of all upper bounds of Y. We form the set $J = \{\bigvee^z Y \mid z \in Z\}$, where $\bigvee^z$ denotes a lub taken in $\downarrow z$. Clearly, this is a set of upper bounds of Y, and by construction every upper bound $z \in Z$ of Z dominates $\bigvee^z Y \in J$. We are left to show the uniqueness: if $z \geq \bigvee^{z_1} Y$, then $\bigvee^{z_1} Y \geq \bigvee^z Y$ since $\bigvee^{z_1} Y$ is an upper bound of Y in $\downarrow z$. Next, $z_1 \geq \bigvee^z Y$ follows, since $z_1 \geq \bigvee^{z_1} Y$. Finally we have $\bigvee^z Y \geq \bigvee^{z_1} Y$ (whence the uniqueness), since since $\bigvee^z Y$ is an upper bound of Y in $\downarrow z_1$.

(5) $\Rightarrow$ (4) Obvious. □

The terminology of L partial order is in accordance with that of L-domain (cf. definition 5.2.11), as shown in exercise 12.5.6.

Exercise 12.5.6 *(1) Show that a cpo is an L partial order iff all its finite subsets have multilub's. (Hint: use characterisation (5) of proposition 12.5.5.) (2) Show that, relaxing the finiteness assumption, proposition 5.2.15 provides a sixth characterisation of L partial orders.*

Proposition 12.5.7 *1. Let D and D' be cpo's, and lef $f : D \to D'$ be stable. Then for any connected $X \subseteq D$ such that $\bigwedge X$ exists, $f(\bigwedge X) = \bigwedge f(X)$.*

2. If all connected glb's exist, the stable functions are exactly the continuous functions preserving connected glb's.

PROOF. (1) $f(\bigwedge X)$ is a lower bound of $f(X)$ by monotonicity. Suppose that $z' \leq f(X)$. We show that all $m(f, x, z')$'s are equal, for x ranging over X. This follows obviously from the fact that for two comparable x_1, x_2, we have $m(f, x_1, z') = m(f, x_2, z')$. Let z stand for this common value. Then we have $z \leq \bigwedge X$ and $z' \leq f(z)$. Therefore $z' \leq f(\bigwedge X)$.

(2) This follows from proposition 12.2.2, observing that the preservation of the glb of a bounded set M can be rephrased as the preservation of the glb of the connected set $M \cup \{x\}$, where x is an upper bound of M. □

To get cartesian closedness, similarly to the cm case, a property of distributivity (or continuity of glb's) is required, namely that connected glb's distribute over directed lub's. Equivalently, the domains are required to be continuous L-domains (see section 12.6).

Definition 12.5.8 (connected meet cpo) *A connected meet cpo is a cpo which is an L partial order such that connected glb's distribute over directed lub's, that is, if $\{\Delta_j\}_{j\in J}$ is an indexed collection of directed sets, and if $\{\bigvee \Delta_j \mid j \in J\}$ is connected, then*

$$\bigwedge_{j\in J}(\bigvee \Delta_j) = \bigvee\{\bigwedge_{j\in J} x_j \mid \{x_j\}_{j\in J} \in \Pi_{j\in J}\Delta_j\}.$$

Theorem 12.5.9 (continuous L-domains - CCC) *The category* **CLDom**[6] *of connected meet cpo's and stable functions is a cpo-enriched CCC.*

Proof. The composition of two stable functions is stable, because a monotonic function maps connected sets to connected sets. As for cm functions and stable functions, directed lub's and binary compatible glb's of stable functions are defined pointwise.

Let $H \subseteq_{dir} D \to_{st} D'$. The lub of H is h defined by $h(x) = \bigvee\{f(x) \mid f \in H\}$. We check that h is stable. Let $X = \{x_i \mid i \in I\}$ be connected:

$$\begin{array}{lcl} h(\bigwedge X) & = & \bigvee_{f\in H}(f(\bigwedge X)) \\ \bigwedge_{i\in I} h(x_i) & = & \bigwedge_{i\in I}(\bigvee_{f\in H} f(x_i)) \quad = \quad \bigvee\{\bigwedge_{i\in I} f_i(x_i) \mid \{f_i\}_{i\in I} \in \Pi_{i\in I} H\}\ . \end{array}$$

The distributivity gives us too many glb's: we are only interested in the families $\{f_i\}$ which are constant. We cannot use the same argument as in the cm case, because we do not have an upper bound available for a family like $\{f_i\}$. We claim:

$$\bigwedge_{i\in I} f_i(x_i) = \bigwedge\{f_i(x_j) \mid i, j \in I\} \quad (= \bigwedge_{i\in I} f_i(\bigwedge X)).$$

The claim can be reformulated as $\forall\, i, j \ \bigwedge_{i\in I} f_i(x_i) \leq f_i(x_j)$. For fixed i, we prove the inequality $\bigwedge_{i\in I} f_i(x_i) \leq f_i(x_j)$ by induction on the length of the zigzag from x_i to x_j. Let x_k be the point preceding x_j in the zigzag. Thus by induction $\bigwedge_{i\in I} f_i(x_i) \leq f_i(x_k)$. There are two cases:

- $x_k \leq x_j$: $\bigwedge_{i\in I} f_i(x_i) \leq f_i(x_j)$ follows obviously by monotonicity.
- $x_j \leq x_k$: Let $f \in H$ be such that $f_i, f_j \leq f$. We have

  $$f_i(x_j) = f_i(x_k) \wedge f(x_j) \geq f_i(x_k) \wedge f_j(x_j).$$

  Using induction, we get

  $$\bigwedge_{i\in I} f_i(x_i) \leq f_i(x_k) \wedge f_j(x_j) \leq f_i(x_j).$$

[6] This name will be justified by theorem 12.6.6.

Turning back to the stability of h, we are left to show:

$$\bigvee_{f\in H}(f(\bigwedge X)) = \bigvee\{\bigwedge_{i\in I} f_i(\bigwedge X) \mid \{f_i\}_{i\in I} \in \Pi_{i\in I}H\}.$$

($\leq$) Take the constant family f.

($\geq$) $\bigwedge_{i\in I} f_i(\bigwedge X) \leq \bigvee_{i\in I} f_i(\bigwedge X) \leq \bigvee_{f\in H}(f(\bigwedge X))$.

Let K be a connected subset of $D \to_{st} D$. Its glb k is defined by $k(x) = \bigwedge_{f\in K} f(x)$.

• k is stable: the preservation of glb's is obvious, but the continuity requires a proof, which is somewhat dual to the proof of stability of $\bigvee H$. We write $K = \{f_i \mid i \in I\}$.

$$\begin{array}{lcl} k(\bigvee \Delta) & = & \bigwedge_{i\in I}(\bigvee f_i(\Delta)) \quad = \quad \bigvee\{\bigwedge_{i\in I} f_i(\delta_i) \mid \{\delta_i\}_{i\in I} \in \Pi_{i\in I}\Delta\} \\ \bigvee k(\Delta) & = & \bigvee_{\delta\in\Delta} k(\delta)\ . \end{array}$$

We claim:

$$\bigwedge_{i\in I} f_i(\delta_i) = \bigwedge\{f_j(\delta_i) \mid i,j \in I\} \quad (= \bigwedge_{i\in I} k(\delta_i)).$$

The claim can be reformulated as $\forall\, i,j \ \bigwedge_{i\in I} f_i(\delta_i) \leq f_j(\delta_i)$. For fixed i, we prove the inequality $\bigwedge_{i\in I} f_i(\delta_i) \leq f_j(\delta_i)$ by induction on the length of the zigzag from f_i to f_j. Let f_k be the point preceding f_j in the zigzag. Thus by induction $\bigwedge_{i\in I} f_i(\delta_i) \leq f_k(\delta_i)$. There are two cases:

- $f_k \leq f_j$: $\bigwedge_{i\in I} f_i(\delta_i) \leq f_j(\delta_i)$ follows obviously by monotonicity.
- $f_j \leq f_k$: Let $\delta \in \Delta$ such that $\delta_i, \delta_j \leq \delta$. We have

  $$f_j(\delta_i) = f_k(\delta_i) \wedge f_j(\delta) \geq f_k(\delta_i) \wedge f_j(\delta_j).$$

  Using induction, we get

  $$\bigwedge_{i\in I} f_i(\delta_i) \leq f_k(\delta_i) \wedge f_j(\delta_j) \leq f_j(\delta_i).$$

Turning back to the continuity of k, we are left to show:

$$\bigvee_{\delta\in\Delta} k(\delta) = \bigvee\{\bigwedge_{i\in I} k(\delta_i) \mid \{\delta_i\}_{i\in I} \in \Pi_{i\in I}\Delta\}.$$

($\leq$) Take the constant family δ.

($\geq$) $\bigwedge_{i\in I} k(\delta_i) \leq \bigvee_{i\in I} k(\delta_i) \leq \bigvee_{\delta\in\Delta} k(\delta)$.

• k is a lower bound of K: Let $x \leq y$, and let $f_0 \in K$. We have to prove that $z \leq f_0(x), k(y)$ implies $z \leq k(x)$, i.e., $z \leq f_1(x)$ for all $f_1 \in K$. It is enough to check this for $f_0 \leq_{st} f_1$ or $f_1 \leq_{st} f_0$:

- $f_0 \leq_{st} f_1$: then a fortiori $f_0 \leq_{ext} f_1$, hence $z \leq f_0(x) \leq f_1(x)$.
- $f_1 \leq_{st} f_0$: then a fortiori $z \leq f_0(x), f_1(y)$, hence $z \leq f_0(x) \wedge f_1(y) = f_1(x)$.

• k is the greatest lower bound of K: Suppose $k_1 \leq_{st} K$. We show $k_1 \leq_{st} k$. Let $x \leq y$, and let $f_0 \in K$: $k(x) \wedge k_1(y) = k(x) \wedge f_0(x) \wedge k_1(y) = k(x) \wedge k_1(x) = k_1(x)$. □

Summarizing, we have obtained cartesian closure for two categories of stable functions exploiting two different sorts of distributivity: the (compatible) distributivity of binary meets over binary joins, and the distributivity of connected glb's over directed lub's, respectively. The proof techniques are quite different too, since Berry's proof uses the definition of stability through minimal points, while in Taylor's category the techniques used for meet cpo's and cm functions are extended to the connected glb's.

Exercise 12.5.10 *Show that a dI-domain satisfies the distributivity of connected glb's over directed lub's. Hint: go through stable event structures.*

12.6 Continuous L-domains *

In this section we show that connected meet cpo's can be alternatively presented as continuous L-domains. We call continuous lattice a partial order which is both a complete lattice and a continuous cpo (cf. definition 5.1.1). We first investigate some properties of continuous lattices. Recall example B.5.3: if X, Y are partial orders which have all glb's (i.e., are complete lattices), a monotonic function $f : X \to Y$ has a left adjoint iff $f : X \to Y$ preserves all glb's.

Remark 12.6.1 *The complete lattice assumption in example B.5.3 can be weakened to the requirement that the glb's of the form $\bigwedge\{z \mid y \leq f(z)\}$ exist. (They are the ones involved in the proof.)*

Remark 12.6.2 *Stable functions do not preserve enough glb's to have a left adjoint: the set $\{z \mid y \leq f(z)\}$ is not bounded in general, nor connected. But stable functions preserve enough glb's to be characterised as having a multi-adjoint (cf. definition 12.2.1). Indeed, the proof of proposition 12.2.2 is a variation of the proof of example B.5.3.*

We shall apply example B.5.3 to (subclasses of) continuous dcpo's. First we characterise continuous dcpo's by an adjunction property.

Proposition 12.6.3 *A dcpo D is continuous iff $\bigvee : Ide(D) \to D$ has a left adjoint.*

Proof. ($\Leftarrow$) Call g the left adjoint of $\bigvee$. For any ideal I and any x we have: $x \leq \bigvee I$ iff $g(x) \subseteq I$. We show that $g(x) = \Downarrow(x)$.

- $\Downarrow(x) \subseteq g(x)$: We have $x \leq \bigvee g(x)$ by adjointness. Hence if $y \ll x$, we have $y \in g(x)$ by definition of $\ll$ and of ideal. Thus $\Downarrow(x) = g(x)$ is directed, and $x = \bigvee(\Downarrow(x))$ since x dominates $\Downarrow(x)$.
- $g(x) \subseteq \Downarrow(x)$: If $y \in g(x)$, then for any ideal I such that $x \leq \bigvee I$ we have $y \in I$. Hence for any directed Δ such that $x \leq \bigvee \Delta$, we have $y \leq \delta$ for some $\delta \in \Delta$, which means exactly $y \ll x$.

($\Rightarrow$) Obvious. □

Proposition 12.6.4 *A complete lattice D is continuous iff arbitrary glb's distribute over directed lub's, that is, if $\{\Delta_j\}_{j\in J}$ is an indexed collection of directed sets, then*

$$\bigwedge_{j\in J}(\bigvee \Delta_j) = \bigvee\{\bigwedge_{j\in J} x_j \mid \{x_J\}_{j\in J} \in \Pi_{j\in J}\Delta_j\}.$$

PROOF. We first show that ideals are closed under intersection. Let $\{I_j\}_{j\in J}$ be a collection of ideals. Take $x_1, x_2 \in \bigcap_{j\in J} I_j$. In each I_j we can pick $y_j \geq x_1, x_2$. Then $\bigwedge_{j\in J} y_j$ is an upper bound for x_1, x_2 in $\bigcap_{j\in J} I_j$.

By proposition 12.6.3 and example B.5.3, D is continuous iff $\bigvee$ preserves the intersection of ideals. Hence D is continuous iff

$$\bigvee \bigcap_{j\in J} I_j = \bigwedge_{j\in J}(\bigvee I_j) \quad \text{for any } \{I_j\}_{j\in J}$$

which is equivalent to the equality of the statement since $\downarrow \{\bigwedge_{j\in J} x_j \mid \{x_J\}_{j\in J} \in \Pi_{j\in J}\Delta_j\} = \bigcap_{j\in J} \downarrow(\Delta_j)$. □

We can require less glb's. Indeed, connectedness suffices to make the above proof work. We now adapt proposition 12.6.4 to L-domains, i.e., L partial orders which are complete (cf. definition 5.2.11).

Lemma 12.6.5 *Let D be an L-domain. If $\{I_j\}_{j\in J}$ is an indexed collection of ideals of D, and if $\{\bigvee I_j \mid j \in J\}$ is connected, then $\bigcap_{j\in J} I_j$ is an ideal.*

PROOF. Take $x_1, x_2 \in \bigcap_{j\in J} I_j$, and pick $y_j \geq x_1, x_2$ in each I_j. We show that the collection $\{y_j\}_{j\in J}$ is connected. Indeed, for any $j_1, j_2 \in J$, we have

$$y_{j_1} \leq \bigvee I_{j_1} \star \cdots \star \bigvee I_{j_2} \geq y_{j_2}.$$

Hence $\bigwedge_{j\in J} y_j$ exists, and is a bound for x_1, x_2 in $\bigcap_{j\in J} I_j$. □

Theorem 12.6.6 *An L-domain D is continuous iff it is a connected meet cpo.*

PROOF. We adapt the proof of proposition 12.6.4. We know that D is continuous iff $\bigvee$ preserves the intersection of ideals, provided "enough" of these intersections exist: by remark 12.6.1, we have to check that $\{I \mid y \leq \bigvee I\}$ satisfies the conditions of lemma 12.6.5: $y = \bigvee(\downarrow y)$ implies $\downarrow y \in \{I \mid y \leq \bigvee I\}$, from which the connectedness of $\{\bigvee I \mid y \leq \bigvee I\}$ follows. Therefore D is continuous iff $\bigvee(\bigcap_{j\in J} I_j) = \bigwedge_{j\in J}(\bigvee I_j)$ for any collection $\{I_j\}_{j\in J}$ of ideals such that $\{\bigvee I_j \mid j \in J\}$ is connected. This is equivalent to the following property for any collection of directed sets Δ_j such that $\{\bigvee \Delta_j \mid j \in J\}$ is connected:

$$\bigvee\{\bigwedge_{j\in J} x_j \mid \{x_j\}_{j\in J} \in \Pi_{j\in J}\Delta_j\} = \bigwedge_{j\in J}(\bigvee \Delta_j)$$

provided the glb's $\bigwedge_{j\in J} x_j$ in the equality exist. Since $\{\bigvee \Delta_j \mid j \in J\}$ is connected, we also have that $\{\bigvee \Delta_j \mid j \in J\} \cup \{x_j \mid j \in J\}$ is connected, and its glb is the glb of $\{x_j \mid j \in J\}$. □

(1) meet cpo's and cm functions

(1') distributive meet cpo's and cm functions

(2) distributive meet cpo's and stable functions

(3) connected meet cpo's and stable functions

Domains satisfying I (stable = cm)

(4) stable bifinite domains axiomatised via:

- finite stable projections
- property $(MI)^\infty$

(5) event domains axiomatised via:

- event structures (concrete)
- I, (C), (S) (abstract)

(6) dI-domains axiomatised via:

- d, I (abstract)
- coprime algebraic + I (abstract)
- bounded complete + finite projections (abstract)
- coprime event structures (concrete)
- stable event structures (concrete)

(7) qualitative domains

(8) coherence spaces

$$(8) \subseteq (7) \subseteq (6) \subseteq \begin{cases} (5) \subseteq (4) \text{ (exercise 12.4.11)} \subseteq (1) \\ (1') \subseteq (1) \\ (2) \\ (3) \text{ (exercise 12.5.10)} \end{cases}$$

Figure 12.4: CCC's of stable and cm functions

Chapter 13

Towards Linear Logic

Girard's linear logic [Gir87] is an extension of propositional logic with new connectives providing a logical treatment of resource control. As a first hint, consider the linear λ-terms, which are the λ-terms defined with the following restriction: when an abstraction $\lambda x.M$ is formed, then x occurs exactly once in M. Linear λ-terms are normalised in linear time, that is, the number of reduction steps to their normal form is proportional to their size. Indeed, when a β-reduction $(\lambda x.M)N$ occurs , it involves no duplication of the argument N. Thus all the complexity of normalisation comes from non-linearity.

Linear logic pushes the limits of constructivity much beyond intuitionistic logic, and allows in particular for a constructive treatment of (linear) negation. A proper proof-theoretic introduction to linear logic is beyond the scope of this book. In this chapter, we content ourselves with a semantic introduction. By doing so, we actually follow the historical thread: the connectives of linear logic arose from the consideration of (a particularly simple version of) the stable model.

In section 13.1, we examine stable functions between coherence spaces, and discover two decompositions. First the function space $E \to E'$ is isomorphic to a space $(!E) \multimap E'$, where $\multimap$ constructs the space of linear functions, and where $!$ is a constructor which allows reuse of data. Intuitively, linear functions, like linear terms, can use their input only once. On the other hand, the explicit declaration of reusability $!$ allows to recover all functions and terms. The second decomposition is the linear version of the classical definition of implication: $E \multimap E'$ is the same as $E^{\perp} \parr E'$, where $\perp$ is the negation of linear logic and where $\parr$ is a disjunction connective (due to resource sensitiveness, there are two different conjunctions and two different disjunctions in linear logic).

In section 13.2, we introduce the categorical material needed to express the properties of the new connectives. We introduce a sequent calculus for linear logic, and we sketch its categorical interpretation.

In the rest of the chapter, we investigate other models in which linear logic can be interpreted. In section 13.3, we present Bucciarelli-Ehrhard's notion of strong stability, and Ehrhard's model of hypercoherences [BE94, Ehr93]. Strongly stable

functions provide an extensional (although not an order-extensional) higher-order lifting of first-order sequentiality (cf. section 6.5). A non-extensional treatment of sequentiality, where morphisms at all orders are explicitly sequential, and in which a part of linear logic can also be interpreted, is offered in chapter 14. In section 13.4 we present the model of bistructures, which combines the stable order of chapter 12 and the pointwise order of chapter 1 in an intriguing way [CPW96]. Finally, in section 13.5, we show that also Scott continuous functions lend themselves to a linear decomposition based on the idea of presentations of (Scott) topologies [Lam94].

Summarizing, linear logic cuts accross most of the flavours of domain theory met in this book: continuity, stability, and sequentiality.

13.1 Coherence Spaces

Coherence spaces offer an extremely simple framework for stability. They were briefly mentioned in section 12.3.

Definition 13.1.1 (coherence space) *A coherence space $(E, \frown)$ (E for short) is given by a set E of events, or tokens, and by a binary reflexive and symmetric relation $\frown$ over E. E is called the* web *of $(E, \frown)$, and we sometimes write $E = |(E, \frown)|$. A state (or clique) of E is a set x of events satisfying the following consistency condition:*

$$\forall e_1, e_2 \in x \;\; e_1 \frown e_2.$$

We denote with $D(E)$ the set of states of E, ordered by inclusion. If $(E, \frown)$ is a coherence space, its incoherence[1] *is the relation $\asymp$ defined by*

$$e_1 \asymp e_2 \Leftrightarrow \neg(e_1 \frown e_2) \text{ or } e_1 = e_2.$$

Clearly, coherence can be recovered from incoherence:

$$e_1 \frown e_2 \Leftrightarrow \neg(e_1 \asymp e_2) \text{ or } e_1 = e_2.$$

Since a coherence space E is a special case of event structure (cf. definition 12.3.11), we already know from proposition 12.3.4 that $D(E)$ is a dI-domain whose compact elements are the finite states, whose minimum is $\perp = \emptyset$, and whose bounded lub's are set unions.

Definition 13.1.2 *We call* **Coh** *the category whose objects are coherence spaces and whose homsets are the stable functions:*

$$\mathbf{Coh}[E, E'] = D(E) \to_{st} D(E').$$

[1]Notice that the incoherence is not the complement of the coherence, since the coherence and the incoherence are both reflexive.

Proposition 13.1.3 *The category* **Coh** *is cartesian closed.*

PROOF. The category **Coh** can be viewed as a full subcategory of the category of dI-domains, and the following constructions show that the terminal dI-domain is a coherence space, and that the products and the exponents of coherence spaces are coherence spaces.

- $1 = (\emptyset, \emptyset)$
- The events of $E \times E'$ are either either $e.1$, with $e \in E$, or $e'.2$, with $e' \in E'$ (with an explicit notation for disjoint unions), and the coherence is

$$(e_1.i) \frown (e_2.j) \Leftrightarrow i \neq j \text{ or } (i = j \text{ and } e_1 \frown e_2).$$

- The events of $E \to E'$ are pairs (x, e'), where x is a finite state of E, and where $e' \in E'$, and the coherence is

$$(x_1, e'_1) \frown (x_2, e'_2) \Leftrightarrow (x_1 \uparrow x_2 \Rightarrow (e'_1 \frown e'_2 \text{ and } (x_1 \neq x_2 \Rightarrow e'_1 \neq e'_2))).$$

The proposition thus follows as a corollary of theorem 12.3.6. □

The key observation that served as a starting point to linear logic is that the dissymetry in the pairs (state,event) indicates that $\to$ should not be taken as primitive. Instead, Girard proposed a unary constructor ! and a binary constructor (such that $E \to E' = (!E)\ (\ E'$.

Definition 13.1.4 (linear exponent – coherence spaces) *The linear exponent $E\ (\ E'$ of two coherence spaces E and E' is the coherence space whose events are the pairs (e, e') where $e \in E$ and $e' \in E'$, and whose coherence is given by*

$$(e_1, e'_1) \frown (e_2, e'_2) \Leftrightarrow (e_1 \frown e_2 \Rightarrow (e'_1 \frown e'_2 \text{ and } (e_1 \neq e_2 \Rightarrow e'_1 \neq e'_2))).$$

Lemma 13.1.5 *In $E\ (\ E'$, the following equivalences hold (and thus may alternatively serve as definition of coherence):*

$$\begin{array}{lll} (1) & (e_1, e'_1) \frown (e_2, e'_2) & \Leftrightarrow \quad (e_1 \frown e_2 \Rightarrow e'_1 \frown e'_2) \text{ and } (e'_1 \smile e'_2 \Rightarrow e_1 \smile e_2) \\ (2) & (e_1, e'_1) \smile (e_2, e'_2) & \Leftrightarrow \quad e_1 \frown e_2 \text{ and } e'_1 \smile e'_2\ . \end{array}$$

PROOF. The equivalence (1) is clearly a rephrasing of the equivalence given in definition 13.1.4 (turning $(e'_1 \smile e'_2 \Rightarrow e_1 \smile e_2)$ into $\neg(e_1 \smile e_2) \Rightarrow \neg(e'_1 \smile e'_2)$). We next show that (2) is equivalent to (1). We have, by successive simple Boolean manipulations:

$$\begin{array}{ll} & \neg((e_1 \frown e_2 \Rightarrow e'_1 \frown e'_2) \text{ and } (e'_1 \smile e'_2 \Rightarrow e_1 \smile e_2)) \\ \Leftrightarrow & (e_1 \frown e_2 \text{ and } \neg(e'_1 \frown e'_2)) \text{ or } (e'_1 \smile e'_2 \text{ and } \neg(e_1 \smile e_2)) \\ \Leftrightarrow & e_1 \frown e_2 \text{ and } (\neg(e'_1 \frown e'_2) \text{ or } (e'_1 \smile e'_2 \text{ and } \neg(e_1 \smile e_2))) \\ \Leftrightarrow & e_1 \frown e_2 \text{ and } e'_1 \smile e'_2 \text{ and } (\neg(e'_1 \frown e'_2) \text{ or } \neg(e_1 \smile e_2))\ . \end{array}$$

We next observe:

$$e_1 \frown e_2 \text{ and } e_1' \asymp e_2' \Rightarrow ((e_1, e_1') = (e_2, e_2') \text{ or } \neg(e_1' \frown e_2') \text{ or } \neg(e_1 \asymp e_2)).$$

which we use as follows:

$$\begin{array}{rcl} (e_1, e_1') \asymp (e_2, e_2') & \Leftrightarrow & (e_1, e_1') = (e_2, e_2') \text{ or } \neg((e_1, e_1') \frown (e_2, e_2')) \\ & \Leftrightarrow & \left\{ \begin{array}{l} (e_1, e_1') = (e_2, e_2') \text{ or} \\ (e_1 \frown e_2 \text{ and } e_1' \asymp e_2' \text{ and } (\neg(e_1' \frown e_2') \text{ or } \neg(e_1 \asymp e_2))) \end{array} \right. \\ & \Leftrightarrow & e_1 \frown e_2 \text{ and } e_1' \asymp e_2' \,. \end{array}$$

□

The states of $E \parr E'$ are in one-to-one correspondence with the linear functions from $D(E)$ to $D(E')$, which we define next.

Definition 13.1.6 (linear function) *Let $(E, \frown)$ and $(E', \frown)$ be two coherence spaces. A stable function $f : D(E) \to D(E')$ is called linear if*

$$\begin{array}{l} f(\bot) = \bot \text{ and} \\ \forall\, x, y \;\; (x \uparrow y \Rightarrow f(x \vee y) = f(x) \vee f(y)) \,. \end{array}$$

We write $D(E) \to_{lin} D(E')$ for the set of linear functions from $D(E)$ to $D(E')$.

Proposition 13.1.7 *Let E and E' be coherence spaces. A stable function $f : D(E) \to D(E')$ is linear if and only its trace (cf. theorem 12.3.6) consists of pairs of the form $(\{e\}, e')$. Hence we freely write, for a linear function:*

$$\mathit{trace}(f) = \{(e, e') \mid e' \in f(\{e\})\}$$

Moreover, trace is an order isomorphism from $D(E) \to_{lin} D(E')$ (ordered by the stable order) to $D(E \parr E')$ (ordered by inclusion).

PROOF. ($\Rightarrow$) Let f be linear, and let $(x, e') \in \mathit{trace}(f)$, and suppose that x is not a singleton. If $x = \bot$, then $e' \in f(\bot)$, and this violates $f(\bot) = \bot$. Otherwise, since in a coherence space any subset of a state is a state, x can be written as $x_1 \cup x_2$, with $x_1, x_2 < x$. Then $e' \notin f(x_1)$ and $e' \notin f(x_2)$ by definition of a trace, therefore $e' \notin f(x_1) \cup f(x_2) = f(x_1) \vee f(x_2)$, violating $f(x) = f(x_1 \vee x_2) = f(x_1) \vee f(x_2)$.

($\Leftarrow$) Suppose that $f(\bot) \neq \bot$, and let $e' \in f(\bot)$. Then $(\bot, e') \in \mathit{trace}(f)$ by definition of a trace, violating the assumption on $\mathit{trace}(f)$. Suppose that $x_1 \uparrow x_2$, and let $e' \in f(x_1 \cup x_2)$. Then there exists $(\{e\}, e') \in \mathit{trace}(f)$ such that $\{e\} \subseteq (x_1 \cup x_2)$, which obviously implies $\{e\} \subseteq x_1$ or $\{e\} \subseteq x_2$, and therefore $e' \in f(x_1)$ or $e' \in f(x_2)$.

Finally, the isomorphism $D(E) \to_{lin} D(E') \cong D(E \parr E')$ follows from the observation that a set ϕ of pairs (e, e') is a state of $E \parr E'$ iff $\{(\{e\}, e') \mid (e, e') \in \phi\}$ is a state of $E \parr E'$. □

Remark 13.1.8 *A computational interpretation of the characterisation of a linear function f given in proposition 13.1.7 can be given as follows. In order to produce an atomic piece of output e', f needs to build, or explore, or consume an atomic piece of input e. In contrast, if (x, e') is in the trace of a stable function f and if x is not a singleton, then f needs to look at x "more than once", specifically $\sharp x$ times, before it can produce e'. In this framework, events can be viewed as units of resource consumption (see remark 14.3.22 for a different flavour of resource counting).*

Proposition 13.1.9 *The composition of two linear functions f and g is linear, and its trace is the relation composition of $trace(f)$ and $trace(g)$.*

PROOF. Let, say, $f : D(E) \to D(E')$ and $g : D(E') \to D(E'')$. The first part of the statement is obvious using the characterisation of linearity by lub and meet preservation properties. We show $trace(g \circ f) \subseteq trace(g) \circ trace(f)$. Let $(e, e'') \in trace(g \circ f)$. By linearity, there exists e' such that $(e', e'') \in trace(g)$, and $e' \in f(\{e\})$ (that is, $(e, e') \in trace(f)$). We now show $trace(g) \circ trace(f) \subseteq trace(g \circ f)$. Let $(e, e') \in trace(f)$ and $(e', e'') \in trace(g)$. Since $e' \leq f(\{e\})$ and $e'' \leq g(\{e'\})$, we have $e'' \leq g(f(\{e\}))$, that is, $(e, e'') \in trace(g \circ f)$. □

This characterisation of the composition of linear functions by trace composition holds in general for dI-domains.

Exercise 13.1.10 *Show the dI-domain version of proposition 13.1.9. Hint: traces then consist of pairs of prime elements.*

Definition 13.1.11 *The category* $\mathbf{Coh}_l$ *is the category whose objects are coherence spaces, and whose morphisms are the linear functions:*

$$\mathbf{Coh}_l[E, E'] = D(E) \to_{lin} D(E').$$

Proposition 13.1.12 *The category* $\mathbf{Coh}_l$ *is cartesian. The terminal object and the products are those of* **Coh**.

PROOF HINT. The projection functions are linear, and the pairing of two linear functions is linear. □

Definition 13.1.13 (exponential –coherence spaces) *Let $(E, \frown)$ be a coherence space. The exponential $!E$ (pronounce "of course E", or "bang E") is the coherence space whose events are the finite states of E, and whose coherence is given by $(x_1 \frown x_2 \Leftrightarrow x_1 \uparrow x_2)$.*

Proposition 13.1.14 *The operation $!$ extends to a functor $! : \mathbf{Coh} \to \mathbf{Coh}_l$ which is left adjoint to the inclusion functor $\subseteq : \mathbf{Coh}_l \to \mathbf{Coh}$*

PROOF HINT. The natural bijections from $\mathbf{Coh}[E, E']$ to $\mathbf{Coh}_l[!E, E']$ are obtained by replacing the pairs (x, e') by pairs $(\{x\}, e')$ in the traces. □

Remark 13.1.15 *Here are some more abstract comments on the adjunction* $! \dashv \subseteq$. *The finite states of* $D(!E)$ *can be seen as presentations of the states of* E, *via the lub operation associating* $\bigvee X$ *with* $X = \{x_1, \ldots, x_n\}$. *There are two presentations of* $\bot$: $\emptyset$, *and* $\{\bot\}$. *It follows that* $D(!E)$ *contains a lifting of* $D(E)$[2] *(cf. definition 1.4.16). Keeping this in mind, it becomes clear how an arbitrary function from* $D(E)$ *to* $D(E')$ *becomes a strict function from* $D(!E)$ *to* $D(E')$ *(cf. proposition 1.4.18).*

The second equivalence of the statement of lemma 13.1.5 naturally suggests a further decomposition of $E \multimap E'$ as $E^\bot \parr E'$, where the new constructors $^\bot$ and $\parr$ are defined as follows.

Definition 13.1.16 (linear negation – coherence spaces) *Let* $(E, \frown)$ *be a coherence space. The linear negation* $E^\bot$ *(pronounce "E perp") of a coherence space* $(E, \frown)$ *is defined as* $E^\bot = (E, \asymp)$.

Definition 13.1.17 (par – coherence spaces) *Let* E, E' *be coherence spaces. Their multiplicative disjunction* $E \parr E'$ *(pronounce "E par E'") is the coherence space whose events are pairs* (e, e') *where* $e \in E$ *and* $e' \in E'$, *and whose incoherence is given by*

$$(e_1, e'_1) \asymp (e_2, e'_2) \Leftrightarrow (e_1 \asymp e_2 \text{ and } e'_1 \asymp e'_2).$$

Other connectives can be defined by De Morgan duality. The dual of $\times$ is another disjunction $\oplus$, called additive. The dual of 1 is written 0. The dual of ! is written ? and called "why not". The dual of $\parr$ is the tensor product, whose direct definition, dictated by $(E \otimes E')^\bot = E^\bot \parr E'^\bot$, is as follows.

Definition 13.1.18 (tensor – coherence spaces) *The tensor product (or multiplicative conjunction)* $E \otimes E'$ *of two coherence spaces* E *and* E' *is the coherence space whose events are pairs* (e, e') *where* $e \in E$ *and* $e' \in E'$, *and whose coherence is given by*

$$(e_1, e'_1) \frown (e_2, e'_2) \Leftrightarrow (e_1 \frown e_2 \text{ and } e'_1 \frown e'_2).$$

Finally, there is a connective called tensor unit:

$$I = (\{\star\}, id).$$

The dual of I is written $\bot$. These connectives obey some categorical constructions, which ensure that they allow to interpret linear logic. Some of them were already discussed in this section (propositions 13.1.3 and 13.1.14). The rest will be completed in the next section.

[2]Actually, this containment is strict. For example, $!\mathbf{O}$ is isomorphic to $\mathbf{O} \times \mathbf{O}$, which has four elements, while $(\mathbf{O})_\bot$ has three elements and is not a coherence space.

13.2 Categorical Interpretation of Linear Logic

The connectives introduced in section 13.1 fall into three groups, which Girard has named as follows:

multiplicatives:	$I, \bot, \otimes, \wp$, and linear negation.
additives:	$1, 0, \times, \oplus$.
exponentials:	$!, ?$.

In figure 13.2, we present a sequent calculus for linear logic. A much better presentation of the proofs of linear logic is by means of certain graphs called proof nets, which forget some irrelevant details of syntax, and are the gate to a geometric understanding of logic and computation. This goes beyond the scope of this book. We simply refer to [Gir87] and [Dan90], and mention that sequential algorithms introduced in chapter 14 are in the same spirit. The sequents of figure 13.2 are of the form $\vdash \Gamma$, where Γ is a list of formulas, possibly with repetitions. In the rule (*Exchange*), $\sigma(\Gamma)$ means any permutation of the list Γ.

Here are brief comments on this proof system:

- There are no weakening and contraction rules. Weakening allows to add assumptions to a sequent, contraction allows to identify repeated assumptions with a single assumption. They express the two aspects of non-linearity (cf. definition 13.1.6): weakening allows non-strictness, while contraction allows repeated use of resources.

- The rule ($\otimes$) expresses a splitting of resources: Γ for A, and Δ for B. Multiplicative connectives correspond to a form of parallelism without communication. The corresponding categorical notion is that of monoidal category, introduced below.

- The rule ($\times$) expressed sharing of resources: Γ is used both by A and B. The corresponding categorical construction is the product

- The exponential rules regulate the explicit reusability of resources. Rule (*Promotion*) says that a formula proved under reusable assumptions is itself reusable. Rule (*Dereliction*) says that a resource that can be used once is a resource which can be used n times, for some n. Since n can in particular be 0, some information is lost when this rule is applied. Rules (*Contraction*) and Rule (*Weakening*) say that reusable data can be duplicated.

We now sketch the categorical interpretation of the formlas and proofs of linear logic. We first introduce a few categorical notions, building upon the structure of monoidal category [ML71, Bar91b].

Definition 13.2.1 (monoidal) *A monoidal category is a category* **C** *equipped with:*

LOGICAL RULES

$$(Axiom)\quad \frac{}{\vdash A, A^{\perp}} \qquad (Cut)\quad \frac{\vdash A, \Gamma \qquad \vdash A^{\perp}, \Delta}{\vdash \Gamma, \Delta}$$

$$(Exchange)\quad \frac{\vdash \Gamma}{\vdash \sigma(\Gamma)}$$

MULTIPLICATIVES

$$(I)\quad \vdash I \qquad (\perp)\quad \frac{\vdash \Gamma}{\vdash \perp, \Gamma}$$

$$(\otimes)\quad \frac{\vdash A, \Gamma \qquad \vdash B, \Delta}{\vdash A \otimes B, \Gamma, \Delta} \qquad (\wp)\quad \frac{\vdash A, B, \Gamma}{\vdash A \wp B, \Gamma}$$

ADDITIVES

$$(1)\quad \vdash 1, \Gamma \qquad (\oplus)\quad \frac{\vdash A, \Gamma}{\vdash A \oplus B, \Gamma} \quad \frac{\vdash B, \Gamma}{\vdash A \oplus B, \Gamma}$$

$$(\times)\quad \frac{\vdash A, \Gamma \qquad \vdash B, \Gamma}{\vdash A \times B, \Gamma}$$

EXPONENTIALS

$$(Promotion)\quad \frac{\vdash A, ?B_1, \ldots, ?B_n}{\vdash !A, ?B_1, \ldots, ?B_n} \qquad (Dereliction)\quad \frac{\vdash A, \Gamma}{\vdash ?A, \Gamma}$$

$$(Contraction)\quad \frac{\vdash ?A, ?A, \Gamma}{\vdash ?A, \Gamma} \qquad (Weakening)\quad \frac{\vdash \Gamma}{\vdash ?A, \Gamma}$$

Figure 13.1: Sequent calculus for linear logic

- *a functor* $\otimes : \mathbf{C} \times \mathbf{C} \to \mathbf{C}$*, called* tensor product*,*
- *a distinguished object* I*, called* tensor unit*, and*
- *natural isomorphisms, also called canonical isomorphisms:*

$$\begin{array}{l} \alpha : A \otimes (B \otimes C) \to (A \otimes B) \otimes C \\ \iota_l : I \otimes A \to A \\ \iota_r : A \otimes I \to A \end{array}$$

satisfying the following two so-called coherence equations:

$$\begin{array}{lll} (\alpha - \alpha) & \alpha \circ \alpha & = & (\alpha \otimes id) \circ \alpha \circ (id \otimes \alpha) \\ (\alpha - \iota) & (\iota_r \otimes id) \circ \alpha & = & id \otimes \iota_l \, . \end{array}$$

Where do the two coherence equations of definition 13.2.1 come from? As observed by Huet (unpublished), a good answer comes from rewriting theory (which came much after monoidal categories were defined in 1963 by Mac Lane). Consider the domains and codomains of the canonical isomorphisms and of the equated arrows as the left and right hand sides of rewriting rules and rewriting sequences, respectively:

$$\begin{array}{lllll} (\alpha) & A \otimes (B \otimes C) & \to & (A \otimes B) \otimes C \\ (\iota_l) & I \otimes A & \to & A \\ (\iota_r) & A \otimes I & \to & A \\ \\ (\alpha - \alpha) & A \otimes (B \otimes (C \otimes D)) & \to^* & ((A \otimes B) \otimes C) \otimes D \\ (\alpha - \iota) & A \otimes (B \otimes I) & \to^* & A \otimes B \, . \end{array}$$

Then the two coherence equations correspond to equating different reduction sequences: $\alpha \circ \alpha$ encodes

$$A \otimes (B \otimes (C \otimes D)) \to (A \otimes B) \otimes (C \otimes D) \to ((A \otimes B) \otimes C) \otimes D$$

while $(\alpha \otimes id) \circ \alpha \circ (id \otimes \alpha)$ encodes

$$A \otimes (B \otimes (C \otimes D)) \to A \otimes ((B \otimes C) \otimes D) \to^* ((A \otimes B) \otimes C) \otimes D.$$

Similarly, the two sides of the second equation encode

$$\begin{array}{l} A \otimes (I \otimes B) \to (A \otimes I) \otimes B \to A \otimes B \\ A \otimes (I \otimes B) \to A \otimes B \, . \end{array}$$

More precisey, these reductions correspond to the so-called critical pairs of the rewriting system on objects induced by $\alpha, \iota_l,$ and ι_r. We pursue this in exercise 13.2.2.

Exercise 13.2.2 (coherence – monoidal) *1. Find all the critical pairs of the rewriting system underlying α, ι_l, and ι_r, and show that the corresponding equations between canonical isomorphisms are derivable from the two equations given in definition 13.2.1.*

2. Prove the so-called coherence theorem for monoidal categories: every two canonical arrows (that is, written by means of the canonical isomorphisms and their inverses only) with the same domain and codomain are equal. Hints: (1) There are three other critical pairs; exploit the fact that α, ι_l, and ι_r are isos. (2) Remove first $\alpha^{-1}, \iota_l^{-1}$, and ι_r^{-1}, and proceed as in the proof of Knuth-Bendix theorem (confluence of critical pairs implies local confluence) [HO80].

Definition 13.2.3 (symmetric monoidal) *A symmetric monoidal category is a monoidal category together with an additional natural isomorphism $\gamma : A \otimes B \to B \otimes A$ satisfying:*

$$\begin{array}{lll} (\gamma - \gamma) & \gamma \circ \gamma & = & id \\ (\alpha - \gamma) & \alpha \circ \gamma \circ \alpha & = & (\gamma \otimes id) \circ \alpha \circ (id \otimes \gamma)\,. \end{array}$$

The coherence theorem case still holds in the symmetric monoidal case, but needs more care in its statement: clearly we do not want to identify $\gamma : A \otimes A \to A \otimes A$ and $id : A \otimes A \to A \otimes A$. Category theorists exclude this by speaking, not of terms, but of natural transformations:

$$\gamma : (\lambda(A,B).A \otimes B) \to (\lambda(A,B).B \otimes A) \quad id : (\lambda(A,B).A \otimes B) \to (\lambda(A,B).A \otimes B).$$

do not have the same codomain. A more elementary point of view is to restrict attention to linear terms for objects.

Exercise 13.2.4 (coherence – symmetric monoidal) ** Show that, in a symmetric monoidal category, any two canonical natural transformations between the same functors are equal. Hints: Use monoidal coherence, and the following presentation of the symmetric group by means of the transpositions σ_i which permute two successive elements i and $i+1$:*

$$\sigma_i \circ \sigma_i = id \quad \sigma_i \circ \sigma_j = \sigma_j \circ \sigma_i \;\; (j - i > 1) \quad \sigma_i \circ \sigma_{i+1} \circ \sigma_i = \sigma_{i+1} \circ \sigma_i \circ \sigma_{i+1}.$$

Definition 13.2.5 (monoidal closed) *A monoidal closed category is a monoidal category $\mathbf{C}$ such that for any object A the functor $\lambda C.(C \otimes A)$ has a right adjoint, written $\lambda B.(A \multimap B)$. In other words, for every objects A, B, there exists an object $A \multimap B$, called* linear exponent*, and natural bijections (for all C):*

$$\Lambda_l : \mathbf{C}[C \otimes A, B] \to \mathbf{C}[C, A \multimap B].$$

We write $ev_l = \Lambda_l^{-1}(id)$.

Notice that there are no accompanying additional coherence equations for monoidal categories. This comes from the difference in nature between the constructions $\otimes$ and (: the latter is given together with a universal construction (an adjunction), while the first is just a functor with some associated isomorphisms. This difference is often referred to as the difference between "additional structure" ($\otimes$) and "property" ((). The notion of dualising object, introduced next, is "additional structure".

Definition 13.2.6 ($\star$-autonomous) *A symmetric monoidal closed category* $\mathbf{C}$ *is called $\star$-autonomous if it has a distinguished object* $\bot$*, called* dualising object*, such that for any* A *the morphisms (called canonical)*

$$\Lambda_l(ev_l \circ \gamma) : \mathbf{C}_l[A, (A \parr \bot) \parr \bot]$$

have an inverse. If no ambiguity can arise, we write $A^\bot$ *for* $A \parr \bot$*, and* $A^{\bot\bot}$ *for* $(A^\bot)^\bot$*.*

Proposition 13.2.7 *Let* $\mathbf{C}$ *be a $\star$-autonomous category.*

1. *There exists a natural bijection between* $\mathbf{C}[A, B]$ *and* $\mathbf{C}[B^\bot, A^\bot]$*.*
2. *There exists a natural isomorphism* $(A \parr B)^\bot \cong A \otimes B^\bot$*.*
3. *There exists a natural isomorphism* $I \cong \bot^\bot$*.*

PROOF HINT. (1) By functoriality of (, with every $f : A \to B$ we can associate $(f \parr \bot) : B^\bot \to A^\bot$. In the other direction, starting from $g : B^\bot \to A^\bot$, we arrive at $(g \parr \bot) : A^{\bot\bot} \to B^{\bot\bot}$, which up to natural isomorphism is in $\mathbf{C}[A, B]$.

(2) In one direction, by associativity, and using ev_l twice, we get a morphism from $(A \otimes B^\bot) \otimes (A \parr B)$ to $\bot$. In the other direction, we exploit the following chain of transformations:

$$\begin{array}{lcll} \mathbf{C}[(A \parr B)^\bot, A \otimes B^\bot] & \cong & \mathbf{C}[(A \otimes B^\bot)^\bot, (A \parr B)^{\bot\bot}] & \text{(by (1))} \\ & \cong & \mathbf{C}[(A \otimes B^\bot)^\bot, A \parr B] & (\bot \text{ is dualising}) \\ & \cong & \mathbf{C}[A \parr B^{\bot\bot}, A \parr B] & (\mathbf{C} \text{ is closed}) \\ & \cong & \mathbf{C}[A \parr B, A \parr B] & (\bot \text{ is dualising}) \,. \end{array}$$

(3) We proceed similarly, as suggested by

$$\begin{array}{lcll} id : & I \otimes \bot & \cong & \bot \to \bot \\ ev_l : & I^\bot & \cong & I^\bot \otimes I \to \bot \,. \end{array}$$

□

Part (3) of proposition 13.2.7 shows in restrospective that the name $\bot$ for the dualising object is deserved: we can indeed understand it as the multiplicative false. Often, the linear negation comes first in the semantics. The following approach is thus helpful.

Proposition 13.2.8 *Suppose that* $\mathbf{C}$ *is a symmetric monoidal category, and that* $(\_)^\perp : \mathbf{C}^{op} \to \mathbf{C}$ *is a functor (which we shall call the dualising functor) which is given together with:*

1. *A natural isomorphism* $A \cong A^{\perp\perp}$.
2. *A natural bijection* $\mathbf{C}[I, (A \otimes B^\perp)^\perp] \cong \mathbf{C}[A, B]$.

Then $\mathbf{C}$ *is monoidal closed, with* $($ *defined by* $A \,(\, B = (A \otimes B^\perp)^\perp$.

PROOF. We have:

$$\begin{array}{lcll} \mathbf{C}[A, B \,(\, C] & = & \mathbf{C}[A, (B \otimes C^\perp)^\perp] & \text{(by definition)} \\ & \cong & \mathbf{C}[I, (A \otimes (B \otimes C^\perp)^{\perp\perp})^\perp] & \text{(by (2))} \\ & \cong & \mathbf{C}[I, (A \otimes (B \otimes C^\perp))^\perp] & \text{(by (1))} \\ & \cong & \mathbf{C}[I, ((A \otimes B) \otimes C^\perp)^\perp] & \text{(by associativity)} \\ & \cong & \mathbf{C}[A \otimes B, C] & \text{(by (2))}\,. \end{array}$$

□

Remark 13.2.9 *The above data are close to ensuring that* $\mathbf{C}$ *is* $\star$*-autonomous. Indeed, from* $A \cong A^{\perp\perp}$ *and*

$$A \,(\, I^\perp = (A \otimes I^{\perp\perp})^\perp \cong (A \otimes I)^\perp \cong A^\perp$$

we obtain $A \cong (A \,(\, I^\perp) \,(\, I^\perp$. *However, one would need to impose tedious coherence axioms relating the natural isomorphisms and bijections of proposition 13.2.8, in order to ensure that this isomorphism is indeed the one obtained by twisting and currying the evaluation morphism. One such condition is that the composition of isomorphisms*

$$\mathbf{C}[A, B] \cong \mathbf{C}[I, (A \otimes B^\perp)^\perp] \cong \mathbf{C}[I, (B^\perp \otimes A)^\perp] \cong \mathbf{C}[I, (B^\perp \otimes A^{\perp\perp})^\perp] \cong \mathbf{C}[B^\perp, A^\perp]$$

has to be the action of the functor $^\perp$ *on* $\mathbf{C}[A, B]$.

The last ingredient we need is the notion of comonad, which is dual to that of monad (cf. definition B.8.1).

Definition 13.2.10 (comonad) *A comonad over a category* $\mathbf{C}$ *is a triple* (T, ϵ, δ) *where* $T : \mathbf{C} \to \mathbf{C}$ *is a functor,* $\epsilon : T \to id_{\mathbf{C}}$, $\delta : T \to T^2$ *are natural transformations, and the following equations hold:*

$$\epsilon_{TA} \circ \delta_A = id_{TA} \quad T\epsilon_A \circ \delta_A = id_{TA} \quad \delta_{TA} \circ \delta_A = T\delta_A \circ \delta_A.$$

The following derived operation is useful. For all $f : TA \to B$, *one constructs* $\kappa(f) : TA \to TB$ *as follows:*

$$\kappa(f) = !f \circ \delta.$$

We define the co-Kleisli category $\mathbf{cK}_T$ *(often simply called Kleisli category) as follows. The objects of* $\mathbf{cK}_T$ *are the objects of* $\mathbf{C}$, *and for any* A, B:

$$\mathbf{cK}_T[A, B] = \mathbf{C}[TA, B].$$

The identity morphisms are given by ϵ, and composition $\circ_{cK}$ is defined by

$$g \circ_{cK} f = g \circ \kappa(f).$$

As with monads, every adjunction induces a comonad.

Proposition 13.2.11 *Every adjunction (L, R, η, ϵ), where $F : \mathbf{C} \to \mathbf{C}'$ and $G : \mathbf{C}' \to \mathbf{C}$, induces a comonad $(F \circ G, \epsilon, \delta)$ on $\mathbf{C}'$, where ϵ is the counit of the adjunction, and where $\delta = F\eta G$, i.e. $\delta_B = F(\eta_{GB})$ (for all B). The Kleisli category associated with the comonad is equivalent to the full subcategory of $\mathbf{C}$ whose objects are in the image of R.*

We follow [Bar91b, See89] in our definition of a category allowing to interpret linear logic. Recently, it was pointed out by Bierman that Seely's definition does not validate all the proof reductions rules one would wish for. He proposed a satisfactory definition of a model of intuitionistic linear logic. We refer to [Bie95] for details (see also remarks 13.2.18 and 13.2.22). Here we stick to Seely's style of definition, which is more synthetic and is good enough for introductory purposes.

Definition 13.2.12 (!$\star$-autonomous category) *A !$\star$-autonomous category is a structure consisting of the following data:*

1. *A $\star$-autonomous category $\mathbf{C}_l$ which is at the same time cartesian.*
2. *A comonad $(!, \epsilon, \delta)$ over $\mathbf{C}_l$, called the* exponential, *together with the preceding structure by two natural isomorphisms:*

$$!(A \times B) \cong (!A) \otimes (!B) \quad !1 \cong I.$$

Remark 13.2.13 *If $\mathbf{C}_l$ is only symmetric monoidal closed (that is, if it is not equipped with a dualising object), but has the rest of the structure of a !$\star$-autonomous category, then we can interpret in it intuitionistic linear logic only.*

Remark 13.2.14 *It is often the case that the comonad $! = F \circ G$ is defined via an adjunction $F \dashv G$ between two functors $F : \mathbf{C} \to \mathbf{C}_l$ and $G : \mathbf{C}_l \to \mathbf{C}$. By proposition 13.2.11, if each object of $\mathbf{C}$ is isomorphic to some object in the image of G, then the Kleisli category associated with the comonad is equivalent to $\mathbf{C}$. This is a fortiori the case when G is the (surjective) identity on objects, as in the stable model.*

We next complete the work of section 13.1 and show that coherence spaces form a !$\star$-autonomous category.

Theorem 13.2.15 *The category $\mathbf{Coh}_l$ together with the comonad on $\mathbf{Coh}_l$ induced by the adjunction $! \dashv \subseteq$ is a !$\star$-autonomous category whose Kleisli category is equivalent to $\mathbf{Coh}$.*

PROOF. For the symmetric monoidal structure, we just notice that at the level of events the canonical isomorphisms are given by

$$((e,e'),e'') \leftrightarrow (e,(e',e'')) \quad (e,\star) \leftrightarrow e \quad (\star,e) \leftrightarrow e.$$

There is a natural bijection $\mathbf{Coh}_l[I,E] \cong E$, since $(\star,e_1) \frown (\star,e_2)$ boils down to $e_1 \frown e_2$. Hence we have

$$\begin{array}{rcll} \mathbf{Coh}_l[I,(E \otimes E'^{\perp})^{\perp}] & \cong & \left|(E \otimes E'^{\perp})^{\perp}\right| & \\ & = & |E \parr E'| & \\ & \cong & \mathbf{Coh}_l[E,E'] & \text{by proposition 13.1.7 .} \end{array}$$

Then the closed structure follows from proposition 13.2.8.

To see that $\mathbf{Coh}_l$ is $\star$-autonomous, we set $\perp = I^{\perp}\ (= I)$, and we observe the trace of $\Lambda_l(ev_l \circ \gamma) : A \to (A \parr \perp) \parr \perp$, which is $\{(e,((e,\star),\star)) \mid e \in E\}$. It clearly has as inverse the function whose trace is $\{(((e,\star),\star),)e \mid e \in E\}$.

That **Coh** is equivalent to the Kleisli category follows from remark 13.2.14. We are left to verify the two natural isomorphisms. The first one holds by proposition 13.2.17. For the second one, notice that $D(1)$ is a singleton. □

We examine some consequences of our definition of !$\star$-autonomous category.

Proposition 13.2.16 *If $\mathbf{C}_l$ is a !$\star$-autonomous category, then the associated co-Kleisli (Kleisli for short) category $\mathcal{K}(\mathbf{C}_l)$ is cartesian closed.*

PROOF. As product on objects and as pairing of arrows we take the product on objects and the pairing of arrows of $\mathbf{C}_l$. As projections we take $\pi \circ \epsilon$ and $\pi' \circ \epsilon$. We check one commutation diagram:

$$\begin{array}{rcll} (\pi \circ \epsilon) \circ_{cK} \langle f,f' \rangle & = & \pi \circ (\epsilon \circ \kappa(\langle f,f' \rangle)) & \\ & = & \pi \circ \langle f,f' \rangle & = \quad f\ . \end{array}$$

Next we define $A \to B = (!A) \parr B$. The natural bijections are obtained via the following chain, where we abbreviate $\mathcal{K}(\mathbf{C}_l)$ as $\mathbf{C}$:

$$\begin{array}{rcl@{\qquad}rcl} \mathbf{C}[A \times B, C] & = & \mathbf{C}_l[!(A \times B), C] & \cong & \mathbf{C}_l[(!A) \otimes (!B), C] \\ & \cong & \mathbf{C}_l[!A, (!B) \parr C] & = & \mathbf{C}_l[!A, B \to C] \\ & & & = & \mathbf{C}[A, B \to C]\ . \end{array}$$

□

Conversely, the first of the natural isomorphisms of definition 13.2.12 is implied by the CCC structure of the Kleisli category.

Proposition 13.2.17 *Let $\mathbf{C}_l$ be a $\star$-autonomous category which is at the same time cartesian, and which is equipped with a comonad $(!,\epsilon,\delta)$ such that the associated Kleisli category is cartesian closed. Then there exists a natural isomorphism from $(!A) \otimes (!B)$ to $!(A \times B)$.*

PROOF. Note first that the assumptions of the statement are slightly redundant since we have already seen that the cartesian structure on the Kleisli category is implied. We still denote with $\mathbf{C}$ the Kleisli category. We derive the desired isomorphisms by exploiting the following chains of transformations:

$$\begin{array}{rcl} \mathbf{C}_l[!(A\times B), !(A\times B)] & = & \mathbf{C}[A\times B, !(A\times B)] \\ & \cong & \mathbf{C}[A, B \to !(A\times B)] \\ & = & \mathbf{C}_l[!A, (!B)\,(!(A\times B)] \\ & \cong & \mathbf{C}_l[(!A)\otimes(!B), !(A\times B)]\,. \end{array}$$

We obtain the desired arrow from $(!A)\otimes(!B)$ to $!(A\times B)$ by starting from $id : !(A\times B) \to !(A\times B)$. The arrow in the other direction is constructed similarly. □

Remark 13.2.18 *It would be more agreeable to have the other natural isomorphism $!1 \cong I$ implied as well, but the categorical structure considered here is not rich enough to provide us with an arrow from I to $!1$. This anomaly is repaired in the setting of [Bie95], where the existence of an isomorphism $I \cong !I$ is postulated.*

Another implied structure is that each object of the form $!A$ is endowed with the structure of a comonoid: there are two arrows

$$e : !A \to I \quad d : !A \to (!A)\otimes(!A)$$

satisfying the three (categorical versions of the) comonoid laws (see exercise 13.2.19). These arrows are constucted as follows:

$$\begin{array}{rcll} e & = & !(!) & \text{(the second } ! \text{ is } ! : A \to 1) \\ d & \cong & !(\langle id, id\rangle) & \end{array}$$

where $\cong$ is to be read as "composition with the isomorphisms $!1 \cong I$ and $!(A\times A) \cong (!A)\otimes(!A)$".

Exercise 13.2.19 *Let d and e be as just defined. Show that the following equations are satisfied:*

$$\begin{array}{rcl} \iota_l \circ (e\otimes id)\circ d & = & id \\ \iota_r \circ (id\otimes e)\circ d & = & id \\ \alpha\circ(id\otimes d)\circ d & = & (d\otimes id)\circ d \\ \gamma\circ d & = & d\,. \end{array}$$

This implicit comonoid structure may serve as a guide in constructing a $!\star$-autonomous category. Some authors have even insisted that ! should be a free comonoid construction. We content ourselves with the following exercise and the following remark.

Exercise 13.2.20 *Show that, for any symmetric monoidal category $\mathbf{C}$, the category $\mathbf{Com}(\mathbf{C}_l)$ whose objects are comonoids (A, d, e) over $\mathbf{C}$ and whose morphisms are comonoid morphisms (i.e. morphisms of $\mathbf{C}_l$ which commute with d, e in the obvious sense) is cartesian, and that the forgetful functor $U : \mathbf{Com}(\mathbf{C})_\mathbf{l} \to \mathbf{C}_l$ maps products to tensor products.*

Remark 13.2.21 *Exercise 13.2.20 suggests a "recipe" for constructing a $!\star$-autonomous category out of a cartesian and $\star$-autonomous category $\mathbf{C}_l$.*

- *Focus on an appropriate (full) subcategory $\mathbf{C}$ of $\mathbf{Com}(\mathbf{C}_l)$ which has the same products as $\mathbf{Com}(\mathbf{C}_l)$.*
- *Construct an adjunction of the form $U \vdash G$ between $\mathbf{C}$ and $\mathbf{C}_l$), where U is the restriction of the forgetful functor to $\mathbf{C}$.*

Then, setting $! = U \circ G$, the natural isomorphism $!(A \times B) \cong (!A) \otimes (!B)$ comes for free:

$$\begin{array}{ll} G(A \times B) \cong G(A) \times G(B) & \textit{(right adjoints preserve limits)} \\ U(G(A) \times G(B)) = (U(G(A)) \otimes (U(G(B)) & \textit{cf. exercise 13.2.20}. \end{array}$$

Interpretation of linear logic proofs. Finally, we sketch the interpretation of the sequents of linear logic in a $!\star$-autonomous category $\mathbf{C}_l$. A proof of a sequent $\vdash A_1, \ldots, A_n$ is interpreted by a morphism $f : I \to (A_1 \parr \ldots \parr A_n)$ (confusing the formulas with their interpretations as objects of $\mathbf{C}_l$). The rules are interpreted as follows:

(I) $\vdash I$ is interpreted by $id : I \to I$.

$(\bot)$ Obvious, since $\bot$ is the dual of I (cf. proposition 13.2.7).

$(\otimes)$ If $f : I \to (A \parr \Gamma)$ and $g : I \to (B \parr \Delta)$, then by the isomorphisms $A \parr \Gamma \cong \Gamma \parr A \cong \Gamma^\bot \multimap A$ (and similarly $B \parr \Delta \cong \Delta^\bot \multimap B$) and by uncurrying, we can consider $f : \Gamma^\bot \to A$ and $g : \Delta^\bot \to B$. Then we form $f \otimes g : \Gamma^\bot \otimes \Delta^\bot \to A \otimes B$ which by similar manipulations yields:

$$f \otimes g : I \to (A \otimes B) \parr (\Gamma^\bot \otimes \Delta^\bot)^\bot = (A \otimes B) \parr (\Gamma \parr \Delta).$$

$(\parr)$ Obvious by associativity of $\parr$.

$(Axiom)$ Since $A \parr A^\bot \cong A \multimap A$. we interpret $\vdash A, A^\bot$ by the currying of the identity considered as from $I \otimes A$ to A.

(Cut) Similar to $(\otimes)$: from $f : I \to (A \parr \Gamma)$ and $g : I \to (A^\bot \parr \Delta)$ we get $ev_l \circ \gamma \circ (f \otimes g) : \Gamma^\bot \otimes \Delta^\bot \to \bot$, which we can read as a morphism from I to $(\Gamma \parr \Delta) \parr \bot \cong \Gamma \parr \Delta$.

$(Exchange)$ By associativity and commutativity.

(1) Interpreting $\vdash 1 \parr \Gamma$ amounts to give an arrow from $\Gamma^\bot$ to 1. Since 1 is terminal, we take the unique such arrow.

$(\times)$ The pairing of $f : I \to (A \parr \Gamma)$ and $g : I \to (B \parr \Gamma)$ yields $\langle f, g \rangle : \Gamma^\bot \to (A \times B)$.

($\oplus$) Given $f : I \to (A \parr \Gamma)$, we build $f \circ \pi : A^\perp \times B^\perp \to \Gamma$, which we can consider as a morphism from I to $(A \oplus B) \parr \Gamma$.

(*Dereliction*) Given $f : I \to (A \parr \Gamma)$, we build $f \circ \epsilon : !(A^\perp) \to \Gamma$, where ϵ is the first natural transformation of the comonad.

(*Weakening*) Let $f : I \to \Gamma$. Then we can consider that $f \circ d = !(A^\perp) \to \Gamma$ (where d comes from the comonoid structure induced on $!A$) is a proof of $\vdash ?A, \Gamma$.

(*Contraction*) This case is similar to the case (*Weakening*), replacing d by e.

(*Promotion*) Let $f : I \to (A \parr ?B_1 \parr \ldots \parr ?B_n)$, which we can consider as an arrow from $!(B_1^\perp \times \ldots \times B_n^\perp)$ to A. Here we have made an essential use of the natural isomorphisms required in the definition of $!\star$-autonomous category. Then we can consider $\kappa(f) : !(B_1^\perp \times \ldots \times B_n^\perp) \to !A$ as a proof of $\vdash !A, ?B_1, \ldots, ?B_n$.

Remark 13.2.22 *The interpretation given above for the rule (Promotion) has the drawback of appealing explicitly to products. This is a bit odd since linear logic without additive connectives, but with exponentials, is interesting enough. In the setting of [Bie95], the existence of isomorphisms* $!(A \otimes B) \cong (!A) \otimes (!B)$ *is postulated. Then (Promotion) is interpreted as follows. Let* $f : I \to (A \parr ?B_1 \parr \ldots \parr ?B_n)$*, which we can consider as an arrow from* $!(B_1^\perp) \otimes \ldots \otimes !(B_n^\perp)$ *to* A*. Consider*

$$\delta \otimes \ldots \otimes \delta : !(B_1^\perp) \otimes \ldots \otimes !(B_n^\perp) \to !!(B_1^\perp) \otimes \ldots \otimes !!(B_n^\perp) \cong !(!(B_1^\perp) \otimes \ldots \otimes !(B_n^\perp)).$$

Then $!f \circ (\delta \otimes \ldots \otimes \delta)$ *can be considered as an arrow from* $!(B_1^\perp) \otimes \ldots \otimes !(B_n^\perp)$ *to* $!A$*, hence is a valid interpretation for* $\vdash !A, ?B_1, \ldots, ?B_n$*.*

Remark 13.2.23 *In some models, like the sequential model presented in chapter 14, the terminal object is the unit. Then we can define projections* $\pi_l : A \otimes B \to A$ *and* $\pi'_l : A \otimes B \to B$ *as follows. One goes, say, from* $A \otimes B$ *to* $A \otimes 1$ *(using that* 1 *is terminal) and then to* A *by a coherent isomorphism. A consequence is that the usual weakening rule:*

$$\frac{\vdash \Gamma}{\vdash A, \Gamma}$$

is valid. Indeed, given $f : I \to \Gamma$*, we build* $f \circ \pi_l : I \otimes A^\perp \to \Gamma$*. Intuitionistic linear logic plus weakening is called affine logic and can be interpreted in any symmetrical monoidal closed category which has a* $!$ *and where the terminal object is the unit.*

13.3 Hypercoherences and Strong Stability

In this section, we investigate Ehrhard's hypercoherences. They have arisen from an algebraic characterisation of sequential functions (see theorem 13.3.16).

Definition 13.3.1 (hypercoherence) *A hypercoherence is a pair (E,Γ), where E is a set, called the web of E, and where Γ is a subset of $\mathcal{P}^\star_{fin}(E)$, called* atomic coherence *(or simply coherence), such that for any $a \in E$, $\{a\} \in \Gamma$. We write $\Gamma^\star = \{u \in \Gamma \mid \sharp u > 1\}$, and call $\Gamma^\star$ the strict atomic coherence. If needed from the context, we write $E = |(E,\Gamma)|$ (or simply $E = |E|$) and $\Gamma = \Gamma(E)$. A hypercoherence is called* hereditary *if*

$$\forall\, u \in \Gamma, v \;\; (v \subseteq u \Rightarrow v \in \Gamma).$$

A state of a hypercoherence (E,Γ) is a set $x \subseteq E$ such that

$$\forall\, u \subseteq^\star_{fin} x \;\; u \in \Gamma$$

where $u \subseteq^\star_{fin} x$ means that u is a finite and non-empty subset of x. We call $D(E)$ the set of states of E, ordered by inclusion.

Proposition 13.3.2 *Let (E,Γ) be a hypercoherence. The poset $D(E)$, ordered by inclusion, is a qualitative domain (cf. definition 12.3.11), whose compact elements are the finite states.*

PROOF. We observe that singletons are states, and that the definition of state enforces that ($u \in \mathcal{K}(D(E))$ and $v \subseteq u$) imply $v \in \mathcal{K}(D(E))$. □

Remark 13.3.3 *Since any qualitative domain can be obviously viewed as a hereditary hypercoherence, we can say in view of proposition 13.3.2 that qualitative domains and hereditary hypercoherences are the same (see proposition 13.3.23). But of course there are more hypercoherences than those arising naturally from qualitative domains (see definition 13.3.14).*

The atomic coherence gives also rise to a collection of distinguished subsets of states, enjoying some interesting closure properties. They will allow us to get a more abstract view of the morphisms of the hypercoherence model, in the same way as linear or stable functions are more abstract than their representing traces. We need a technical definition.

Definition 13.3.4 (multisection) *Let E be a set, and let $u \subseteq E$ and $A \subseteq \mathcal{P}(E)$. We write $u \triangleleft A$, and say that u is a multisection of A, iff*

$$(\forall\, e \in u \;\; \exists\, x \in A \;\; e \in x) \quad \textit{and} \quad (\forall\, x \in A \;\; \exists\, e \in u \;\; e \in x).$$

Remark 13.3.5 *If both u and A are finite, $u \triangleleft A$ holds exactly when we can find a list of pairs $(e_1,x_1),\ldots,(e_n,x_n)$ such that $e_i \in x_i$ for all i and*

$$u = \{e_i \mid 1 \leq i \leq n\} \quad A = \{x_i \mid 1 \leq i \leq n\}.$$

Definition 13.3.6 (state coherence) *Let (E,Γ) be a hypercoherence. We define a set $\mathcal{C}(E) \subseteq \mathcal{P}^{\star}_{fin}(D(E))$, called the state coherence of (E,Γ), as follows:*

$$\mathcal{C}(E) = \{A \subseteq^{\star}_{fin} D(E) \mid \forall\, u \subseteq^{\star}_{fin} E \;\; u \lhd A \Rightarrow u \in \Gamma\}.$$

We recall the convex ordering from theorem 9.1.6. Given a partial order D and two subsets B, A of D, we write $B \leq_c A$ for

$$(\forall\, y \in B \;\; \exists\, x \in A \;\; y \leq x) \quad \text{and} \quad (\forall\, x \in A \;\; \exists\, y \in B \;\; y \leq x).$$

(Notice that this obeys the same pattern as the definition of $\lhd$.)

Lemma 13.3.7 *Let (E,Γ) be a hypercoherence. The state coherence $\mathcal{C}(E)$ satisfies the following properties:*

1. *If $x \in D(E)$, then $\{x\} \in \mathcal{C}(E)$.*
2. *If $A \in \mathcal{C}(E)$ and $B \leq_c A$, then $B \in \mathcal{C}(E)$.*
3. *If $A \subseteq^{\star}_{fin} D(E)$ has an upper bound in $D(E)$, then $A \in \mathcal{C}(E)$.*
4. *If $A \subseteq^{\star}_{fin} D(E)$ and $\emptyset \in A$, then $A \in \mathcal{C}(E)$.*

PROOF. (1) If $u \lhd \{x\}$, then a half of the definition of $\lhd$ says $u \subseteq x$, and $u \in \Gamma$ then follows by definition of a state.

(2) Let $A \in \mathcal{C}(E)$, $B \leq_c A$, and $u \lhd B$. It follows from the definitions of $\leq_c$ and $\lhd$ that $u \lhd A$, and from $A \in \mathcal{C}(E)$ that $u \in \Gamma$.

(3) This follows from (1) and (2), since we can express that A has an upper bound z as $A \leq_c \{z\}$.

(4) If $\emptyset \in A$, then we cannot find a u such that $u \lhd A$, and the condition characterising $A \in \mathcal{C}(E)$ holds vacuously. □

Definition 13.3.8 (strongly stable) *Let (E,Γ) and (E',Γ') be hypercoherences. A continuous function $f : D(E) \to D(E')$ is called strongly stable from (E,Γ) to (E',Γ') if*

$$\forall\, A \in \mathcal{C}(E) \;\; (f(A) \in \mathcal{C}(E') \text{ and } f(\textstyle\bigwedge A) = \bigwedge f(A)).$$

We call **HCoh** *the category of hypercoherences and strongly stable functions.*

Strongly stable functions are to form (up to equivalence) the Kleisli category of our model. We turn to the definition of the linear category.

Definition 13.3.9 (linear exponent – hypercoherences) *Let (E,Γ), (E',Γ') be two hypercoherences. The linear exponent $E \multimap E'$ is the hypercoherence whose events are the pairs (e,e') where $e \in E$ and $e' \in E'$, and whose atomic coherence consists of the finite non-empty w's such that*

$$\pi(w) \in \Gamma \Rightarrow (\pi'(w) \in \Gamma' \text{ and } (\sharp\pi'(w) = 1 \Rightarrow \sharp\pi(w) = 1)).$$

Definition 13.3.10 *The category* $\mathbf{HCoh}_l$ *is the category whose objects are hypercoherences, and whose morphisms are given by*

$$\mathbf{HCoh}_l[E, E'] = D(E \multimap E')$$

for all E, E'*, with identity relations and relation composition as identities and composition.*

Proposition 13.3.11 *The data of definition 13.3.10 indeed define a category.*

PROOF. The only non-obvious property to check is that the relation composition of two states is a state. Suppose thus that (E, Γ), (E', Γ'), and (E'', Γ'') are given, as well as $\phi \in D(E \multimap E')$ and $\phi' \in D(E' \multimap E'')$. We first claim that if $(e, e'') \in \phi' \circ \phi$, then there exists a unique $e' \in E'$ such that $(e, e') \in \phi$ and $(e', e'') \in \phi'$. Let w' be a finite non-empty subset of $W' = \{e' \mid (e, e') \in \phi \text{ and } (e', e'') \in \phi'\}$ (e, e'' fixed). Considering $\{(e, e') \mid e' \in w'\}$, we obtain that $w' \in \Gamma'$ since $\{e\} \in \Gamma$. Then, considering $\{(e', e'') \mid e' \in w'\}$, we get $\sharp w' = 1$ since $\sharp\{e''\} = 1$. Therefore W' is also a singleton. Let now w be a finite non-empty subset of $\phi' \circ \phi$ such that $\pi(w) \in \Gamma$. Consider

$$\begin{array}{rcl} u & = & \{(e, e') \in \phi \mid \exists e''\ (e', e'') \in \phi' \text{ and } (e, e'') \in w\} \\ v & = & \{(e', e'') \in \phi' \mid \exists e\ (e, e') \in \phi \text{ and } (e, e'') \in w\}\ . \end{array}$$

The claim implies that u and v are finite. We have $\pi(u) = \pi(w)$ by definition of u and of $\phi' \circ \phi$. It follows that $\pi'(u) \in \Gamma'$ since we have assumed $\pi(w) \in \Gamma$. But $\pi'(u) = \pi(v)$, hence $\pi'(w) = \pi'(v) \in \Gamma''$. If furthermore $\sharp\pi'(w) = 1$, then $\sharp\pi(v) = 1 = \sharp\pi'(u)$, and $\sharp\pi(u) = 1 = \sharp\pi(w)$. □

Proposition 13.3.12 *The category* $\mathbf{HCoh}_l$ *is cartesian. If* (E, Γ) *and* (E', Γ') *are hypercoherences, their product* $E \times E'$ *is defined as follows:*

- *Events are either* $e.1$ *where* $e \in E$ *or* $e'.2$ *where* $e' \in E'$*.*
- *The atomic coherence consists of the (finite non-empty) subsets* w *such that*

$$(w\lceil_E = \emptyset \Rightarrow w\lceil_{E'} \in \Gamma') \text{ and } (w\lceil_{E'} = \emptyset \Rightarrow w\lceil_E \in \Gamma)$$

  where, say, $w\lceil_E = \{e \mid e.1 \in w\}$*.*

We have

$$\begin{array}{rcl} D(E \times E') & \cong & D(E) \times D(E') \\ A \in \mathcal{C}(E \times E') & \Leftrightarrow & (A\lceil_E \in \mathcal{C}(E) \text{ and } A\lceil_{E'} \in \mathcal{C}(E')) \end{array}$$

where, say, $A\lceil_E = \{u\lceil_E \mid u \in A\}$*. The terminal object is the empty hypercoherence.*

PROOF. The projection morphisms and pairing are given by

$$\begin{array}{lcl} \pi & = & \{(e.1, e) \mid e \in E\} \\ \pi' & = & \{(e'.2, e') \mid e' \in E'\} \\ \langle \phi, \phi' \rangle & = & \{(e'', e.1) \mid (e'', e) \in \phi\} \cup \{(e'', e'.2) \mid (e'', e') \in \phi'\} \ . \end{array}$$

The only point which requires care is the verification that $\langle \phi, \phi' \rangle$ is a morphism. Let $w \subseteq^{\star}_{fin} \langle \phi, \phi' \rangle$ be such that $\pi(w) \in \Gamma$. Notice that, in order to tell something about $\pi'(w)\lceil_E$, we are led to consider $w_E = \{(e'', e) \mid (e'', e.1) \in w\}$. If $\pi(w_E)$ is a strict subset of $\pi(w)$, we don't know whether $\pi(w_E)$ is in the atomic coherence of E, unless (E, Γ) is hereditary (see remark 13.3.13). These considerations should help to motivate the definition of $\Gamma(E \times E')$. Indeed, all we have to check is that if $\pi'(w)\lceil_{E'} = \emptyset$ then $\pi'(w)\lceil_E \in \Gamma$ (the other implication being proved similarly). The assumption $\pi'(w)\lceil_{E'} = \emptyset$ implies $w = w_E$, $\pi(w_E) \in \Gamma''$, and $\pi'(w)\lceil_E = \pi'(w_E) \in \Gamma$. If moreover $\sharp\pi'(w) = 1$, then, say, $\pi'(w) = \{e.1\}$, hence $\pi'(w_E) = \{e\}$, which entails $\sharp\pi(w_E) = 1$, and $\sharp\pi(w) = 1$ since $\pi'(w) = \{e.1\}$ a fortiori implies $\pi'(w)\lceil_{E'} = \emptyset$.

We show that $\lambda x.(x\lceil_E, x\lceil_{E'})$ defines a bijection (actually an order isomorphism) from $D(E \times E')$ to $D(E) \times D(E')$. All what we have to do is to show that this mapping and its inverse have indeed $D(E) \times D(E')$ and $D(E \times E')$ as codomains, respectively. This follows easily from the following observation (and from the similar observation relative to E'):

$$\forall\, u \subseteq^{\star}_{fin} x\lceil_E \quad (u \in \Gamma \Leftrightarrow \{e.1 \mid e \in u\} \in \Gamma(E \times E')).$$

The same observation serves to prove the characterisation of $\mathcal{C}(E \times E')$, since in order to check that $A \in \mathcal{C}(E \times E')$, we need only to consider u's such that either $u\lceil_E = \emptyset$ or $u\lceil_{E'} = \emptyset$. □

Remark 13.3.13 *The cartesian product of two hereditary hypercoherences is not hereditary in general. Indeed, given any finite $u \subseteq E$ and $u' \subseteq E'$, where, say, $u \notin \Gamma(E)$, we have $\{e.1 \mid e \in u\} \cup \{e'.2 \mid e' \in u'\} \in \Gamma(E \times E')$ as soon as both u and u' are non-empty, but $\{e.1 \mid e \in u\} \notin \Gamma(E \times E')$. (See also proposition 13.3.15.)*

The full subcategory of hereditary hypercoherences, being equivalent to that of qualitative domains, has a product, though, which is given by the same set of events, but a different atomic coherence, consisting of the w's such that

$$w\lceil_E \in \Gamma \text{ and } w\lceil_{E'} \in \Gamma'.$$

The product structure gives rise to an interesting class of hypercoherences, which are the key to the link between sequentiality and strong stability, and which are therefore at the root of the theory of hypercoherences and strong stability. Coherence spaces $(E, \frown)$ are special qualitative domains, and can therefore be seen as hereditary hypercoherences. But they can also be endowed with another hypercoherence structure, which we define next.

Definition 13.3.14 *1. Let $(E, \frown)$ be a coherence space. It induces a hypercoherence (E, Γ_L), called* linear hypercoherence*, where*

$$\Gamma_L^\star = \{u \subseteq^\star_{fin} E \mid \exists\, e_1, e_2 \in E \ \ (e_1 \neq e_2 \text{ and } e_1 \frown e_2)\}.$$

2. Let X be a set. The hypercoherence $X_\perp = (X, \{\{x\} \mid x \in X\})$ is called the flat hypercoherence associated to X. Clearly, $D(X_\perp) = X_\perp$, whence the name and the notation.

Proposition 13.3.15 *Let $E_1, \ldots, E_n$ be flat hypercoherences. Let E be the product of $E_1, \ldots, E_n$ in the category of coherence spaces. Then (E, Γ_L) is the product of $E_1, \ldots, E_n$ in* $\mathbf{HCoh}_l$.

Proof. Without loss of generality, we consider a product of three flat hypercoherences E_1, E_2, and E_3. By definition, the product hypercoherence consists of those w's such that

$$\begin{array}{l}(\pi_1(w) \neq \emptyset \text{ or } \pi_2(w) \neq \emptyset \text{ or } \sharp\pi_3(w) = 1) \text{ and}\\ (\pi_1(w) \neq \emptyset \text{ or } \pi_3(w) \neq \emptyset \text{ or } \sharp\pi_2(w) = 1) \text{ and}\\ (\pi_2(w) \neq \emptyset \text{ or } \pi_3(w) \neq \emptyset \text{ or } \sharp\pi_1(w) = 1)\,.\end{array}$$

Under the assumption $\sharp w > 1$,we have, say:

$$(\pi_1(w) \neq \emptyset \text{ or } \pi_2(w) \neq \emptyset \text{ or } \sharp\pi_3(w) = 1) \ \ \Rightarrow \ \ (\pi_1(w) \neq \emptyset \text{ or } \pi_2(w) \neq \emptyset)$$

hence the strict coherence of the product consists of the w's such that

$$\begin{array}{l}(\pi_1(w) \neq \emptyset \text{ or } \pi_2(w) \neq \emptyset) \text{ and}\\ (\pi_1(w) \neq \emptyset \text{ or } \pi_3(w) \neq \emptyset) \text{ and}\\ (\pi_2(w) \neq \emptyset \text{ or } \pi_3(w) \neq \emptyset)\end{array}$$

or (generalising from 3 to n)

$$\exists\, i, j \leq n \ \ i \neq j \text{ and } (\pi_i(w) \neq \emptyset \text{ and } \pi_j(w) \neq \emptyset)$$

which by proposition 13.1.3 and by definition is the linear coherence on E. □

We have now all the ingredients to show where strong stability comes from.

Theorem 13.3.16 *Suppose that $E_1, \ldots, E_n, E$ are flat hypercoherences. A function $f : D(E_1) \times \ldots \times D(E_n) \to D(E)$ is sequential iff it is strongly stable from $(E_1 \times \ldots \times E_n, \Gamma_L)$ to E.*

Proof. By proposition 13.3.15, $(E_1 \times \ldots \times E_n, \Gamma_L)$ is the product $E_1 \times \ldots \times E_n$ in $\mathbf{HCoh}_l$. Hence, by proposition 13.3.12:

$$A \in \mathcal{C}_L(E) \Leftrightarrow \forall\, i \in \{1, \ldots, n\} \ \ \pi_i(A) \in \mathcal{C}(E_i)$$

where $\mathcal{C}_L(E)$ is the state coherence associated with Γ_L. If $A = \{x_1, \ldots, x_k\}$ and $x_j = (x_{1j}, \ldots, x_{nj})$ for all $j \leq k$, then we can rephrase this as

$$\{x_1, \ldots, x_k\} \in \mathcal{C}_L(E) \Leftrightarrow \forall i \in \{1, \ldots, n\}\ (\exists j \leq k\ x_{ij} = \perp) \text{ or } (x_{i1} = \ldots = x_{ik} \neq \perp).$$

On the other hand, by theorem 6.5.4, f is sequential iff it is invariant under the relations S_{k+1} defined by

$$(x_1, \ldots, x_{k+1}) \in S_{k+1} \quad \Leftrightarrow \quad (\exists j \leq k\ \ x_j = \perp) \text{ or } (x_1 = \ldots = x_{k+1} \neq \perp).$$

The conclusion then follows from the following easy observations:

- $(x_{i1}, \ldots, x_{i(k+1)}) \in S_{k+1}$ may be rephrased as $(x_{i1}, \ldots, x_{ik}) \in \mathcal{C}(D)$ and $\bigwedge_{1 \leq j \leq k} x_{ij} \leq x_{i(k+1)}$, hence $\bigwedge_{1 \leq j \leq k} x_j \leq x_{k+1}$ and

$$(x_1, \ldots, x_{k+1}) \in S^n_{k+1} \Leftrightarrow (x_1, \ldots, x_k) \in \mathcal{C}((D_1 \times \ldots \times D_n)_L).$$

- $f(\bigwedge A) = \bigwedge f(A)$ can be rephrased as $\forall x\ \ x \geq \bigwedge A \Rightarrow f(x) \geq \bigwedge f(A)$. □

Lemma 13.3.17 *In E ($E' = E''$, the following equivalences hold (and thus may alternatively serve as definition of atomic coherence):*

(1) $w \in \Gamma'' \quad \Leftrightarrow \quad (\pi(w) \in \Gamma \Rightarrow \pi'(w) \in \Gamma')$ *and* $(\pi(w) \in \Gamma^\star \Rightarrow \pi'(w) \in \Gamma'^\star)$
(2) $w \in \Gamma''^\star \quad \Leftrightarrow \quad \pi(w) \notin \Gamma$ *or* $\pi'(w) \in \Gamma'^\star$.

PROOF. The equivalence (1) is just a rephrasing of the equivalence given in definition 13.3.9. By Boolean manipulations, we get successively:

right hand side of (1)

$$\Leftrightarrow \quad \begin{cases} (\pi(w) \notin \Gamma \text{ and } (\pi(w) \notin \Gamma^\star \text{ or } \pi'(w) \in \Gamma'^\star)) \text{ or} \\ (\pi'(w) \in \Gamma' \text{ and } (\pi(w) \notin \Gamma^\star) \text{ or } \pi'(w) \in \Gamma'^\star)) \end{cases}$$

$$\Leftrightarrow \quad \pi(w) \notin \Gamma \text{ or } \pi'(w) \in \Gamma'^\star \text{ or } (\pi'(w) \in \Gamma' \text{ and } \pi(w) \notin \Gamma^\star)\ .$$

Now we suppose that $\sharp w > 1$. Then either $\sharp\pi(w) > 1$ or $\sharp\pi'(w) > 1$. If $\sharp\pi'(w) > 1$, then $\pi'(w) \in \Gamma'$ is the same as $\pi'(w) \in \Gamma'^\star$. Similarly, if $\sharp\pi'(w) > 1$, then $\pi(w) \notin \Gamma^\star$ is the same as $\pi(w) \notin \Gamma$, Hence, under the assumption $\sharp w > 1$:

$$(\pi'(w) \in \Gamma' \text{ and } \pi(w) \notin \Gamma^\star) \Rightarrow (\pi(w) \notin \Gamma \text{ or } \pi'(w) \in \Gamma'^\star)$$

and the right hand side of (1) boils down to the right hand side of (2), which completes the proof. □

As in the stable case, the equivalence (2) of lemma 13.3.17 directly suggests a definition of tensor product and of linear negation.

Definition 13.3.18 (tensor – hypercoherences) *Let* (E,Γ) *and* (E',Γ') *be hypercoherences. Their tensor product* $E \otimes E'$ *is the hypercoherence whose web is* $E \times E'$*, and whose atomic coherence consists of the non-empty finite* w*'s such that* $\pi(w) \in \Gamma$ *and* $\pi'(w) \in \Gamma$*.*

Definition 13.3.19 (linear negation – hypercoherences) *Let* (E,Γ) *be a hypercoherence. The linear negation* $E^\perp$ *of* E *is the hypercoherence whose web is* E*, and whose atomic coherence is* $\mathcal{P}^\star_{fin}(E)\backslash\Gamma^\star$*. Or, alternatively,* $u \in \Gamma^\star(E^\perp)$ *iff* $u \notin \Gamma(E)$*.*

Proposition 13.3.20 *The category* $\mathbf{HCoh}_l$ *is* $\star$*-autonomous. Its unit is the unique hypercoherence whose web is the singleton* $\{\star\}$*.*

PROOF. The proof is a straightforward adaptation of the proof of theorem 13.2.15. We even have $\left|(E \otimes E'^\perp)^\perp\right| = \mathbf{HCoh}_l[E,E']$. □

Definition 13.3.21 (exponential – hypercoherences) *Let* (E,Γ) *be a hypercoherence. The exponential* $!E$ *is the hypercoherence whose events are the finite states of* E *and whose atomic coherence consists of the the* A*'s such that*

$$\forall\, u \subseteq^\star_{fin} E \;\; (u \lhd A \Rightarrow u \in \Gamma).$$

Proposition 13.3.22 *The operation* $!$ *extends to a functor* $! : \mathbf{HCoh} \to \mathbf{HCoh}_l$ *which is left adjoint to the inclusion functor* $\subseteq$*:* $\mathbf{HCoh}_l \to \mathbf{HCoh}$ *defined as follows:*

$$\subseteq(E,\Gamma) = (E,\Gamma) \quad \subseteq(\phi) = \mathit{fun}(\phi)$$

where

$$\mathit{fun}(\phi)(x) = \{e' \mid \exists\, e \;\; (e,e') \in \phi \text{ and } e \in x\}.$$

PROOF. We exhibit inverse bijections between $D(!E \multimap E')$ and $\mathbf{HCoh}[E,E']$. Given a strongly stable function f, we define

$$\mathit{trace}(f) = \{(x,e') \mid e' \in f(x) \text{ and } (\forall\, y < x \;\; e' \notin f(x))\}.$$

Conversely, given $\phi \in D(!E \multimap E')$, we define

$$\mathit{fun}(\phi)(x) = \{e' \mid \exists\, y \;\; (y,e') \in \phi \text{ and } y \subseteq x\}.$$

This definition of *fun* extends that given in the statement, up to the identification of events e with singletons $\{e\}$. Therefore, the subsequent proof also establishes that $\subseteq$ is well-defined. That *trace* and *fun* are inverses is proved exactly as in theorem 12.3.6. We prove that *trace* and *fun* are well-defined:

• *trace*(f) is a state: Let $w_0 \subseteq^\star_{fin} \mathit{trace}(f)$, and suppose that $\pi(w_0) \in \Gamma(!E) = \mathcal{C}(E)$. By definition of *trace*, $\pi'(w_0) \lhd f(\pi(w_0))$. Hence $\pi'(w_0) \in \Gamma'$. Suppose

moreover that $\pi'(w_0) = \{e'\}$. Then $e' \in \bigwedge f(\pi(w_0)) = f(\bigwedge \pi(w_0))$. Let $z \leq \bigwedge \pi(w_0)$ be such that $(z, e') \in \mathit{trace}(f)$. Then, by minimality, z is equal to each of the elements of $\pi(w_0)$.

• $\mathit{fun}(\phi)$ is strongly stable: First of all, we have to check that $f(x) \in D(E')$, for any (finite) $x \in D(E)$. Let $v_1 \subseteq^{\star}_{\mathit{fin}} f(x)$. Let $w_1 = \{(z, e') \in \phi \mid z \subseteq x \text{ and } e' \in v_1\}$. By definition, w_1 is finite and non-empty. Hence $w_1 \in \Gamma(E \multimap E')$. We have $\pi(w_1) \in \Gamma(!E)$ since $\pi(w_1)$ is bounded. Hence $v_1 = \pi'(w_1) \in \Gamma'$. Next we prove that $\mathit{fun}(\phi)$ is strongly stable. Let $A \in \mathcal{C}(E)$, and let $v_2 \triangleleft \mathit{fun}(A)$. Let

$$w_2 = \{(z, e') \in \phi \mid (z \subseteq x \text{ for some } x \in A \text{ and } e' \in v_2)\}$$

and let $x \in A$. Since $v_2 \triangleleft \mathit{fun}(A)$, there exists $e' \in v_2$ such that $e' \in \mathit{fun}(x)$. Hence there exists $z \in \pi(w_2)$ such that $z \subseteq x$, by definition of fun and w_2. It follows that $\pi(w_2) \leq_c A$. Hence $\pi(w_2) \in \mathcal{C}(E)$ by lemma 13.3.7. This entails $v_2 = \pi'(w_2) \in \Gamma'$. Hence $\mathit{fun}(A) \in \mathcal{C}(E')$. We show moreover that $\bigwedge \mathit{fun}(\phi)(A) \leq \mathit{fun}(\phi)(\bigwedge A)$. Let $e' \in \bigwedge \mathit{fun}(\phi)(A)$, i.e. $\{e'\} \triangleleft \mathit{fun}(\phi)(A)$. Then, instantiating above v_2 as $v_2 = \{e'\}$ (and the corresponding w_2), we have $\sharp\pi(w_2) = 1$, since $\sharp\pi'(w_2) = \sharp\{e'\} = 1$. Let $\pi(w_2) = \{x\}$. Then $x \leq A$ by definition of w_2, hence $e' \in \mathit{fun}(x) \subseteq \mathit{fun}(\phi)(\bigwedge A)$. □

Proposition 13.3.23 *1. The category* $\mathbf{HCoh}_l$ *is equivalent to the category of hypercoherences and linear and strongly stable functions. Notice that this is slightly redundant, since the preservation of bounded glb's, which is part of our definition of linear function (cf. definition 13.1.6) is a consequence of strong stability.*

2. The full subcategory of $\mathbf{HCoh}_l$ *whose objects are the hereditary hypercoherences is equivalent to the category of qualitative domains and linear functions.*

Proof. (1) This is proved as in the stable case (cf. proposition 13.1.7).

(2) We have already observed (remark 13.3.3) that at the level of objects qualitative domains are the same as hereditary hypercoherences. We claim that if (E, Γ) is hereditary, then

$$A \in \mathcal{C}(E) \quad \Leftrightarrow \quad (\emptyset \in A) \text{ or } (A \text{ is bounded}).$$

The direction $\Leftarrow$ holds in any hypercoherence, by lemma 13.3.7. Suppose conversely that $A \in \mathcal{C}(E)$, $\emptyset \notin A$, and A is not bounded , i.e. $\bigcup A$ is not a state. Then there exists $u \subseteq \bigcup A$ which is not in the atomic coherence Γ. Let v be such that $u \subseteq v$ and $v \triangleleft A$ (such a v exists, since $\emptyset \notin A$). We reach a contradiction as follows:

$v \in \Gamma$ by definition of the state coherence
$u \in \Gamma$ by hereditarity .

Suppose that (E, Γ) and (E', Γ') are hereditary hypercoherences. We show that $f : D(E) \to D(E')$ is linear and strongly stable iff it is linear. We only have to check that "linear implies strongly stable". If f is linear and $A \in \mathcal{C}(E)$, there are two cases, by the claim:

- $\emptyset \in A$: Then, by linearity, $\emptyset \in f(A)$ (hence $f(A) \in \mathcal{C}(E')$), and

  $f(\bigwedge A) = f(\emptyset) = \emptyset = \bigwedge f(A)$.

- A is bounded: Then $f(A)$ is bounded, and $f(\bigwedge A) = \bigwedge f(A)$ by stability. □

Proposition 13.3.24 *The category* $\mathbf{HCoh}_l$*, together with the comonad induced by the adjunction of proposition 13.3.22, is a* $!\star$*-autonomous category.*

PROOF. We are left to check the natural isomorphisms. The isomorphism $!1 \cong I$ is immediate as with coherence spaces. We check $!(E \times E') \cong (!E) \otimes (!E')$ (notice that we did not prove that **HCoh** is cartesian closed: this will rather follow as a corollary, cf. proposition 13.2.16). We have

$$\begin{array}{rcl} |!(E \times E')| & = & \mathcal{K}(D(E \times E')) \\ & \cong & \mathcal{K}(D(E) \times D(E')) \\ & = & \mathcal{K}(D(E)) \times \mathcal{K}(D(E')) \\ & = & |!E| \times |!E'| \\ & = & |(!E) \otimes (!E')| \end{array}$$

and

$$\begin{array}{rcll} A \in \Gamma(!(E \times E')) & \Leftrightarrow & A \in \mathcal{C}(E \times E') & \\ & \Leftrightarrow & (A\lceil_E \in \mathcal{C}(E) \text{ and } A\lceil_{E'} \in \mathcal{C}(E') & \text{(by proposition 13.3.12)} \\ & \Leftrightarrow & (A\lceil_E, A\lceil_{E'}) \in \Gamma((!E) \otimes (!E')) \, . & \end{array}$$

□

13.4 Bistructures *

Berry [Ber79] combined the stable approach and the continuous approach by defining bidomains, which maintain extensionality and stability together, and thus offer an order-extensional account of stability (cf. sections 6.3 and 6.4). His work was then revisited by Winskel, resulting in a theory of stable event structures [Win80]. The account offered here, based on [CPW96] is informed by linear logic (which was not available at the time of [Ber79, Win80]. We introduce a $!\star$-autonomous category of event bistructures, where the definition of composition is proved correct by means of an interaction between two order relations over events.

We build up on coherence spaces (cf. section 13.1). Let E and E' be two coherence spaces. Recall that the structure $E \to E'$ has as events the pairs (x, e') of a finite state of E and an event of E', that these events, when put together to form a higher-order state ϕ, describe the minimal points of the function represented by ϕ, and that the inclusion of states naturally corresponds to the stable ordering. In $E \to E'$, there arises a natural order between events, which is inspired by contravariance:

$$(x_1, e'_1) \leq^L (x, e') \Leftrightarrow (x \subseteq x_1 \text{ and } e'_1 = e').$$

The superscript ^{L} will be explained later. The order $\leq^L$ allows us to describe the pointwise order between stable functions, at the level of traces.

Definition 13.4.1 *Let E and E' be two coherence spaces. We define a partial order $\sqsubseteq$ on $D(E \to E')$ by $\phi \sqsubseteq \psi \Leftrightarrow \forall (x, e') \in \phi \ \exists x' \subseteq x \ (x', e') \in \psi$, or, equivalently:*

$$\phi \sqsubseteq \psi \Leftrightarrow \forall e'' \in \phi \ \exists e''_1 \ (e'' \leq^L e''_1 \text{ and } e''_1 \in \psi).$$

Proposition 13.4.2 *Let E and E' be coherence spaces, and $f, g : D(E) \to_{st} D(E')$. Then the following equivalence holds: $f \leq_{ext} g \Leftrightarrow trace(f) \sqsubseteq trace(g)$.*

We shall see (lemma 13.4.12) that $\phi \sqsubseteq \psi$ can be factored as $\phi \sqsubseteq^L \chi \subseteq \psi$, where χ is the part of ψ used in the check of $\phi \sqsubseteq \psi$ (notice that, given (x, e') the x' is unique).

Next we want to force stable functionals to be order extensional, that is, we want to retain only the functionals H such that $\forall \phi, \psi \ (\phi \sqsubseteq \psi \Rightarrow H(\phi) \leq H(\psi))$ (where we freely confuse functions with their traces), which, by the definition of $\sqsubseteq^L$, can be rephrased as

$$\forall \phi, \psi \ (\phi \sqsubseteq^L \psi \Rightarrow H(\phi) \leq H(\psi)).$$

Therefore we ask for

$$\forall e''' \in H \ \forall e'''_1 \ (e'''_1 \leq^R e''' \Rightarrow \exists e'''_2 \in H \ e'''_1 \leq^L e'''_2)$$

where the order $\leq^R$ is defined by

$$(\phi_1, e'_1) \leq^R (\phi, e') \Leftrightarrow (\phi \sqsubseteq^L \phi_1 \text{ and } e'_1 = e').$$

Definition 13.4.3 (bistructure) *A bistructure $(E, \asymp, \leq^L, \leq^R)$ (or E for short) is given by a set E of events, by a reflexive and symmetric binary relation $\asymp$ on E, called coherence relation, and by partial orders $\leq^L$ and $\leq^R$, satisfying the following axioms.*

(B1) $\forall e_1, e_2 \in E \ (e_1 \downarrow^L e_2 \Rightarrow e_1 \asymp e_2)$

where $e_1 \downarrow^L e_2$ means $\exists e \in E \ (e \leq^L e_1$ and $e \leq^L e_2)$, and where $e_1 \asymp e_2$ means $\neg(e_1 \frown e_2)$ or $e_1 = e_2$ (cf. section 13.1).

(B2) $\forall e_1, e_2 \in E \ (e_1 \uparrow^R e_2 \Rightarrow e_1 \frown e_2)$

where $\uparrow$ is upward compatibility with respect to $\leq^R$.

(B3) $\forall e_1, e_2 \in E \ (e_1 \leq e_2 \Rightarrow (\exists e \in E \ e_1 \leq^L e \leq^R e_2))$

where $\leq = (\leq^L \cup \leq^R)^\star$.

(B4) The relation $\preceq = (\geq^L \cup \leq^R)^\star$ is a partial order.

(B5) $\forall e \in E \ \{e' \in E \mid e' \preceq e\}$ *is finite.*

Remark 13.4.4 *In the presence of axiom (B5), which is the bistructure version of axiom I, axiom (B4) is equivalent to requiring the non-existence of infinite sequences $\{e_n\}_{n<\omega}$ such that for all n $e_{n+1} \prec e_n$, where $e \prec e'$ means $e \preceq e'$ and $e \neq e'$.*

The axioms of bistructures are strong enough to imply the uniqueness of the decomposition of $\leq = (\leq^L \cup \leq^R)^\star$, and that $\leq$ is a partial order.

Lemma 13.4.5 *Let E be a bistructure. For all $e_1, e_2 \in E$, the following properties hold:*

$$\begin{array}{lcl} (e_1 \downarrow^L e_2 \text{ and } e_1 \uparrow^R e_2) & \Rightarrow & e_1 = e_2 \\ e_1 \leq e_2 & \Rightarrow & \exists !e \ (e_1 \leq^L e \leq^R e_2)\ . \end{array}$$

PROOF. (1) If $e_1 \downarrow^L e_2$ and $e_1 \uparrow^R e_2$, then $e_1 \asymp e_2$ and $e_1 \frown e_2$, which implies $e_1 = e_2$ by definition of $\asymp$.

(2) Suppose that $e_1 \leq^L e \leq^R e_2$ and $e_1 \leq^L e' \leq^R e_2$. Then $e_1 \downarrow^L e_2$ and $e_1 \uparrow^R e_2$, therefore $e_1 = e_2$ by (1). □

Lemma 13.4.6 *The relation $\leq = (\leq^L \cup \leq^R)^\star$ of definition 13.4.3 is a partial order.*

PROOF. We only have to prove that $\leq$ is antisymmetric. Suppose $e \leq e' \leq e$, and let ϵ, ϵ' be such that $e \leq^L \epsilon' \leq^R e'$ and $e' \leq^L \epsilon \leq^R e$. We factor $e \leq \epsilon$: for some ϵ'', $e \leq^L \epsilon'' \leq^R \epsilon$. Since $e \leq^L \epsilon'' \leq^R e$, we get

$$\begin{array}{ll} \epsilon'' = e & \text{by lemma 13.4.5} \\ \epsilon = e & \text{by the antisymmetry of } \leq^R\ . \end{array}$$

We then have $e' \leq^L \epsilon = e \leq^L \epsilon' \leq^R e'$, and

$$\begin{array}{ll} \epsilon' = e' & \text{by lemma 13.4.5} \\ e = e' & \text{by the antisymmetry of } \leq^L\ . \end{array}$$

□

We next define states of bistructures.

Definition 13.4.7 *Let $E = (E, \asymp, \leq^L, \leq^R)$ be a bistructure. A state of E is a subset x of E satisfying:*

$$\begin{array}{ll} \forall\, e_1, e_2 \in x \ \ (e_1 \frown e_2) & \textit{(consistency)} \\ \forall\, e \in x \ \ \forall\, e_1 \leq^R e \ \ \exists e_2 \ \ (e_1 \leq^L e_2 \text{ and } e_2 \in x) & \textit{(extensionality)}\ . \end{array}$$

We write $(D(E), \sqsubseteq, \sqsubseteq^R)$ for the collection of states of E, equipped with two orders $\sqsubseteq^R$ and $\sqsubseteq$, which are called the stable order and the extensional order, respectively, and which are defined as follows:

$$\begin{array}{l} \sqsubseteq^R \textit{ is the set-theoretic inclusion} \\ x \sqsubseteq y \Leftrightarrow \forall\, e \in x \ \ \exists e_1 \in y \ \ (e \leq^L e_1)\ . \end{array}$$

Observe that axiom (B1) enforces the uniqueness of e_2 in the extensionality condition of states, and of e_1 in the definition of $\sqsubseteq$. We also define a third relation $\sqsubseteq^L$ between states by

$$x \sqsubseteq^L y \Leftrightarrow x \sqsubseteq y \text{ and } (\forall\, y_1 \in D(E) \ \ (x \sqsubseteq y_1 \text{ and } y_1 \sqsubseteq^R y) \Rightarrow y_1 = y).$$

We shall next examine some properties of the three relations $\sqsubseteq^R$, $\sqsubseteq$, and $\sqsubseteq^L$.

Lemma 13.4.8 *Let E be a bistructure, and let $x \in D(E)$. If e is in the $\leq$ downward closure of x, then it is in the $\leq^L$ downward closure of x.*

PROOF. Let $e_1 \in x$ be such that $e \leq e_1$:

$$\begin{array}{ll} \exists\, e_2 \;\; e \leq^L e_2 \leq^R e_1 & \text{(by factorisation)} \\ \exists\, e_3 \in x \;\; e_2 \leq^L e_3 & \text{(by extensionality)}\ . \end{array}$$

Then $e \leq^L e_3$ fits. □

Lemma 13.4.9 *Let $\sqsubseteq^R$ and $\sqsubseteq$ be as in definition 13.4.7. Then, for all states x, y:*

$$(x \sqsubseteq y \text{ and } y \sqsubseteq^R x) \Rightarrow x = y.$$

PROOF. Let $e \in x$, and let $e_1 \in y$ be such that $e \leq^L e_1$. Then, since a fortiori $e_1 \in x$ and $e \downarrow^L e_1$, we conclude that $e = e_1 \in y$. □

Definition 13.4.10 *Let E be a bistructure, and let $x \in D(E)$. We write $\preceq_x$ for the reflexive and transitive closure of the following relation $\preceq^1_x$ between events of x:*

$$e_1 \preceq^1_x e_2 \;\; \Leftrightarrow \;\; (e_1, e_2 \in x \text{ and } \exists\, e \;\; e_1 \geq^L e \leq^R e_2)\ .$$

The following is a key lemma.

Lemma 13.4.11 *Let E be a bistructure, let $x, y \in D(E)$ and $e_2 \in E$ such that $x \uparrow^R y$ and $e_2 \in x \cap y$. Then the following implication holds, for any $e \in x$ (in particular, $e_1 \in y$):*

$$e_1 \preceq_x e_2 \Rightarrow e_1 \preceq_y e_2.$$

PROOF. It is clearly enough to show this for the one step relation $\preceq^1_x$. Let thus e be such that $e_1 \geq^L e \leq^R e_2$ By extensionality of y, and since $e_2 \in y$, there exists $e_1' \in y$ such that $e \leq^L e_1'$. But (B1) and the consistency of y force $e_1' = e_1$, hence $e_1 = e_1' \preceq_y e_2$. □

Lemma 13.4.12 *Let $\sqsubseteq^R, \sqsubseteq$, and $\sqsubseteq^L$ be as in definition 13.4.7. The following properties hold.*

1. *$\sqsubseteq$ is $(\sqsubseteq^L \cup \sqsubseteq^R)^\star$, and satisfies (B3).*
2. *For all states x, y: $x \sqsubseteq^L y \Rightarrow (\forall\, e \in y \;\; \exists\, e_0 \in x, e_1 \in y \;\; e \preceq_y e_1 \geq^L e_0)$.*
3. *$\sqsubseteq^L$ is a partial order.*

PROOF. (1) Let $x \sqsubseteq y$. The subset $\{e_1 \in y \mid \exists\, e_0 \in x \;\; e_0 \leq^L e_1\}$ represents the part of y actually used to check $x \sqsubseteq y$. But we have to close this subset to make it extensional. Define thus

$$y_1 = \{e \in y \mid \exists\, e_0 \in x, e_1 \in y \;\; e \preceq_y e_1 \geq^L e_0\}.$$

which is clearly consistent, as a subset of y. If $e \in y_1$ and $e_1 \leq^R e$, since y is extensional, there exists $e_2 \in y$ such that $e_1 \leq^L e_2$, and $e_2 \in y_1$ by construction. Thus y_1 is a state.

We show $x \sqsubseteq^L y_1$. Suppose that $x \sqsubseteq y_1' \sqsubseteq^R y_1$, and let $e \in y_1$. Let by construction $e_0 \in x$ and $e_1 \in y$ be such that $e \preceq_y e_1 \geq^L e_0$. Since $x \sqsubseteq y_1'$, $e_0 \leq^L e_1'$ for some $e_1' \in y_1'$. On one hand we have $e_1 \downarrow^L e_1'$ by construction, on the other hand $e_1 \frown e_1'$ follows from $e_1 \in y_1$, $e_1' \in y_1'$, and $y_1' \sqsubseteq^R y_1$. Hence $e_1 = e_1'$, which implies $e \in y_1'$ by lemma 13.4.11. Hence $y_1 \sqsubseteq^R y_1'$, which completes the proof of $x \sqsubseteq^L y_1$. The decomposition $x \sqsubseteq^L y_1 \sqsubseteq^R y$ shows that $\sqsubseteq$ is contained in $(\sqsubseteq^L \cup \sqsubseteq^R)^\star$. The converse inclusion is obvious.

(2) follows obviously from the proof of (1).

(3) The reflexivity and the antisymmetry follow from $(\sqsubseteq^L) \subseteq (\sqsubseteq)$. Let $x \sqsubseteq^L y' \sqsubseteq^L y$, and let $e \in y$. By (2), there exist $e_0 = e' \in y'$ and $e_0' \in x$ such that $e \preceq_y e'$ and $e' \preceq_x e_0'$, or in full:

$$\begin{array}{ll} e' = e_0 \leq^L e_1 \geq^R e_2 \cdots \leq^L e_{2i+1} = e & e_{2j+1} \in y \text{ for all } 0 \leq j \leq i \\ e_0' \leq^L e_1' \geq^R e_2' \cdots \leq^L e_{2i'+1}' = e' & e_{2j+1}' \in y' \text{ for all } 0 \leq j \leq i' \,. \end{array}$$

Since $y' \sqsubseteq y$ and $e_1' \in y'$, there exists e_1'' such that $e_1' \leq^L e_1''$ and $e_1'' \in y$. Since $e_2' \leq^R e_1' \leq^L e_1''$, there exists e_2'' such that $e_1' \leq^L e_2'' \leq^R e_1''$. Since y is extensional, there exists $e_3'' \in y$ such that $e_2'' \leq^L e_3''$. In order to continue this lifting of the e_i''s relative to y' to a sequence of e_i'''s relative to y, we have to make sure that $e_3' \leq^L e_3''$:

$$\begin{array}{ll} e_3' \leq^L e_3''' \in y \text{ for some } e_3''' \in y & \text{since } y' \sqsubseteq y \\ e_3''' = e_3'' & \text{since } e_2' \leq^L e_3'', e_2' \leq^L e_3''', \text{ and } e_3'', e_3''' \in y \,. \end{array}$$

Continuing in this way, we get

$$e'' = e_0' \leq^L e_1'' \geq^R e_2'' \cdots \leq^L e_{2i'+1}'' = e' = e_0 \leq^L e_1 \geq^R e_2 \cdots \leq^L e_{2i+1} = e$$

with $e_{2j+1} \in y$ for all $0 \leq j \leq i$ and $e_{2j+1}'' \in y$ for all $0 \leq j \leq i'$, which completes the proof of $x \sqsubseteq^L y$. □

We explore some of the finiteness and completeness properties of these two orders.

Lemma 13.4.13 *Let E be a bistructure, let $e \in x \in D(E)$. Then there exists a finite state $[e]_x$ such that $e \in [e]_x \sqsubseteq^R x$ and $(\forall\, y \in D(E)\ \ (e \in y \sqsubseteq^R x) \Rightarrow ([e]_x \sqsubseteq^R y))$.*

PROOF. Similar to that of lemma 13.4.12. We just exhibit the definition of $[e]_x$:

$$[e]_x = \{e' \in x \mid e' \preceq_x e\}.$$

The finiteness of $[e]_x$ follows from axiom (B5). □

Proposition 13.4.14 *Let E be a bistructure. The following properties hold.*

1. *All $\sqsubseteq$ and $\sqsubseteq^R$ directed lub's exist in $(D(E), \sqsubseteq, \sqsubseteq^R)$.*
2. *The $\sqsubseteq$ and $\sqsubseteq^R$ lub's of a $\sqsubseteq^R$ directed set coincide.*
3. *A state is $\sqsubseteq$ compact iff it is $\sqsubseteq^R$ compact iff it is finite.*

PROOF. (1) Let Δ be $\sqsubseteq$ directed. We show:

$$z = \{e \in \bigcup\Delta \mid e \text{ is } \leq^L \text{ maximal in } \bigcup\Delta\} \text{ is the } \sqsubseteq \text{ lub of } \Delta.$$

We first check that z is a state. If $e_1, e_2 \in z$, then $e_1 \in \delta_1$, $e_2 \in \delta_2$ for some $\delta_1, \delta_2 \in \Delta$. Let $\delta \in \Delta$ be such that $\delta_1, \delta_2 \sqsubseteq \delta$. Then by definition of z and $\sqsubseteq$, it follows that $e_1, e_2 \in \delta$. Therefore $e_1 \frown e_2$. If $e \in z$ and $e_1 \leq^R e$, let $\delta \in \Delta$ be such that $e \in \delta$. By extensionality of δ, there exists $e_2 \in \delta$ such that $e_1 \leq^L e_2$. By definition of z and by (B4) and (B5) (cf. remark 13.4.4), we can find $e_3 \in z$ such that $e_2 \leq^L e_3$. Hence z is a state. It is obvious from the definition of z that $\delta \sqsubseteq z$ holds for any $\delta \in \Delta$, and that if z_1 is an $\sqsubseteq$ upper bound of Δ then $z \sqsubseteq z_1$. The $\sqsubseteq^R$ bounded lub's exist: if $X \subseteq D(E)$ and if x is an $\sqsubseteq^R$ upper bound of X, then $\bigcup X$ is consistent as a subset of x and extensional as a union of extensional sets of events.

(2) Let Δ be $\sqsubseteq^R$ directed. We prove that $\sqcup^R\Delta = \sqcup\Delta = \bigcup\Delta$ (where $\sqcup^R$ and $\sqcup$ are relative to $\sqsubseteq^R$ and $\sqsubseteq$, respectively). We have to show that any $e \in \bigcup\Delta$ is $\leq^L$ maximal. Suppose there exists $e_1 \in \bigcup\Delta$ such that $e \leq^L e_1$. Then a fortiori $e \downarrow^L e_1$, and since by $\sqsubseteq^R$ directedness $e_1, e_2 \in \delta$ for some $\delta \in \Delta$, we get $e = e_1$.

(3) We decompose the proof into three implications:

- x is finite $\Rightarrow x$ is $\sqsubseteq$ compact: Let $\{e_1, \ldots, e_n\} \sqsubseteq \sqcup\Delta$. There exist $e'_1, \ldots, e'_n \in \sqcup\Delta$ such that $e_i \leq^L e'_i$ for all i. Let $\delta_1, \ldots \delta_n \in \Delta$ such that $e'_i \in \delta_i$ for all i, and let $\delta \in \Delta$ be such that $\delta_i \sqsubseteq \delta$ for all i. Then by the $\leq^L$ maximality of $e'_1, \ldots, e'_n$ we get $e'_i \in \delta$ for all i. Hence $\{e_1, \ldots, e_n\} \sqsubseteq \delta$.

- x is $\sqsubseteq$ compact $\Rightarrow x$ is $\sqsubseteq^R$ compact: If $x \sqsubseteq^R \sqcup^R\Delta$, then a fortiori $x \sqsubseteq \sqcup\Delta$, therefore $x \sqsubseteq \delta$ for some $\delta \in \Delta$. We show that actually $x \sqsubseteq^R \delta$ holds. Let $e \in x$, and let $e_1 \in \delta$ such that $e \leq^L e_1$. Then we get $e = e_1$ from $e, e_1 \in \bigcup\Delta$.

- x is $\sqsubseteq^R$ compact $\Rightarrow x$ is finite: We claim that, for any z, $\{y \mid y \text{ finite and } y \sqsubseteq^R z\}$ is $\sqsubseteq^R$ directed and has z as lub. The directedness is obvious. We have to check that $z \sqsubseteq^R \{y \mid y \text{ finite and } y \sqsubseteq^R z\}$, that is, for all $e \in z$, there exists a finite y such that $y \sqsubseteq^R z$ and $e \in y$. The state $[e]_x$ (cf. lemma 13.4.13) does the job. □

We define a monoidal closed category of bistructures.

Definition 13.4.15 (linear exponent – bistructures) *Let E and E' be two bistructures. The linear exponent bistructure $E \multimap E'$ is defined as follows:*

events are pairs (e, e') where $e \in E$ and $e' \in E'$,
$(e_1, e'_1) \asymp (e_2, e'_2) \Leftrightarrow e_1 \frown e_2$ *and* $e'_1 \asymp e'_2$,
$(e_1, e'_1) \leq^L (e, e') \Leftrightarrow e \leq^R e_1$ *and* $e'_1 \leq^L e'$,
$(e_1, e'_1) \leq^R (e, e') \Leftrightarrow e \leq^L e_1$ *and* $e'_1 \leq^R e'$.

As for coherence spaces , the coherence in the linear exponent can be defined by either of the following equivalences (cf. lemma 13.1.5):

$$\begin{array}{lcl} (e_1, e'_1) \frown (e_2, e'_2) & \Leftrightarrow & (e_1 \frown e_2 \Rightarrow (e'_1 \frown e'_2 \text{ and } (e_1 \neq e_2 \Rightarrow e'_1 \neq e'_2))) \\ (e_1, e'_1) \frown (e_2, e'_2) & \Leftrightarrow & (e_1 \frown e_2 \Rightarrow e'_1 \frown e'_2) \text{ and } (e'_1 \asymp e'_2 \Rightarrow e_1 \asymp e_2)\ . \end{array}$$

The definition of linear exponent suggests what the linear negation should be, and what the connective $\wp$ should be. We shall define these connectives rightaway, and prove that they are correctly defined. The correctness of the definition of $E \parr E'$ will then follow.

Definition 13.4.16 (linear negation – bistructures) *Let $(E, \frown, \leq^L, \leq^R)$ be a bistructure. The linear negation $E^\perp$ is defined by*

$$E^\perp = (E, \smile, \geq^R, \geq^L).$$

Proposition 13.4.17 *$E^\perp$ is a well-defined bistructure.*

Proof. We put subscripts that make clear to which bistructures we refer. We have

$$e_1 \downarrow^L_{E^\perp} e_2 \Leftrightarrow e_1 \uparrow^R_E e_2 \Rightarrow e_1 \frown_E e_2 \Leftrightarrow e_1 \smile_{E^\perp} e_2$$

and similarly for (B2). The satisfaction of (B3) to (B5) follows from

$$(\leq^L_{E^\perp} \cup \leq^R_{E^\perp} = \geq^R \cup \geq^L) \quad \text{and} \quad (\geq^L_{E^\perp} \cup \leq^R_{E^\perp} = \leq^R \cup \geq^L).$$

□

Definition 13.4.18 (par – bistructures) *Let E and E' be two bistructures. The bistructure $E \wp E'$ is defined as follows:*

events are pairs (e, e') where $e \in E$ and $e' \in E'$,
$(e_1, e'_1) \smile (e_2, e'_2) \Leftrightarrow e_1 \smile e_2$ *and* $e'_1 \smile e'_2$,
$(e_1, e'_1) \leq^L (e, e') \Leftrightarrow e_1 \leq^L e$ *and* $e'_1 \leq^L e'$,
$(e_1, e'_1) \leq^R (e, e') \Leftrightarrow e_1 \leq^R e$ *and* $e'_1 \leq^R e'$.

Proposition 13.4.19 *$E \wp E'$ is a well-defined bistructure.*

Proof. $(B1)$ Let $(e_1, e'_1) \downarrow^L (e, e')$. We have $e_1 \smile e$ from $e_1 \downarrow^L e$ and $e'_1 \smile e'$ from $e'_1 \downarrow^L e'$.

$(B2)$ Let $(e_1, e'_1) \uparrow^R (e, e')$, and suppose $(e_1, e'_1) \smile (e, e')$. As in the previous case, we have $e_1 \frown e$ and $e'_1 \frown e'$, which, combined with the definition of $(e_1, e'_1) \smile (e, e')$, gives $e_1 = e$ and $e'_1 = e'$.

The other axioms follow from the componentwise definition of the orders. □

Proposition 13.4.20 *Let E, E', and E'' be bistructures, let $\phi \in D(E \parr E')$ and $\psi \in D(E' \parr E'')$. The graph composition $\psi \circ \phi$ of ϕ and ψ is a state of $E \parr E''$.*

Proof. Let $(e_1, e''_1), (e_2, e''_2) \in \psi \circ \phi$, and let $e'_1, e'_2 \in E'$ be such that

$$(e_1, e'_1) \in \phi \quad (e'_1, e''_1) \in \psi \quad (e_2, e'_2) \in \phi \quad (e'_2, e''_2) \in \psi.$$

Suppose $e_1 \frown e_2$. Since $(e_1, e'_1), (e_2, e'_2) \in \phi$ we have $e'_1 \frown e'_2$. Since $(e'_1, e''_1), (e'_2, e''_2) \in \psi$, we have $e''_1 \frown e''_2$. Similarly, $e''_1 \smile e''_2$ implies $e_1 \smile e_2$. Thus $\psi \circ \phi$ is consistent.

We now check that $\psi \circ \phi$ is extensional. Let $(e, e'') \in \psi \circ \phi$, and $(e_1, e_1'') \leq^R (e, e'')$. Thus $e \leq^L e_1$ and $e_1'' \leq^R e''$, and there exists e' such that $(e, e') \in \phi$ and $(e', e'') \in \psi$. By extensionality of ϕ, and since $(e_1, e') \leq^R (e, e')$, there exists $(e_2, e_2') \in \phi$ such that $(e_1, e') \leq^L (e_2, e_2')$, that is, $e_2 \leq^R e_1$ and $e' \leq^L e_2'$. By extensionality of ψ, and since $(e_2', e_1'') \leq^R (e', e'')$, there exists $(e_3', e_3'') \in \psi$ such that $e_3' \leq^R e_2'$ and $e_1'' \leq^L e_3''$. By extensionality of ϕ, and since $(e_2, e_3') \leq^R (e_2, e_2')$, there exists $(e_4, e_4') \in \phi$ such that $e_4 \leq^R e_2$ and $e_3' \leq^L e_4'$. In this way we build sequences such that

$$\begin{array}{ll} e \leq^L e_1 \geq^R e_2 \geq^R e_4 \geq^R \cdots & (e, e'), (e_2, e_2'), (e_4, e_4'), \ldots \in \phi \\ e' \leq^L e_2' \geq^R e_3' \leq^L e_4' \geq^R \ldots & \\ e'' \geq^R e_1'' \leq^L e_3'' \leq^L \cdots & (e', e''), (e_3', e_3''), \ldots \in \psi \, . \end{array}$$

By axiom (B4), the sequence $\{e_n'\}_{n<\omega}$ becomes stationary. Let i be such that $e_{2i}' = e_{2i+1}'$. Then we have:

$$\begin{array}{ll} (e_{2i}, e_{2i+1}'') \in \psi \circ \phi & \text{since by construction } (e_{2i}, e_{2i}') \in \phi \text{ and } (e_{2i+1}', e_{2i+1}'') \in \psi \\ (e_1, e_1') \leq^L (e_{2i}, e_{2i+1}'') & \text{since by construction } e_1 \geq^R e_{2i} \text{ and } e_1'' \leq^L e_{2i+1}'' \, . \end{array}$$

. □

Thus we have all the ingredients to define a linear category of bistructures.

Definition 13.4.21 *We define the category* $\mathbf{BS}_l$ *as follows: objects are bistructures, and for any* E, E'*, we set*

$$\mathbf{BS}_l[E, E'] = D(E \multimap E').$$

Composition is relation composition, and the identities are the identity relations: $id_E = \{(e, e) \mid e \in E\}$.

Remark 13.4.22 *The morphisms of* $\mathbf{BS}_l$*, unlike in the case of coherence spaces, do not enjoy a simple abstract characterisation as "linear and extensional functions". Recall that linearity amounts to requiring all minimal points to be prime. In coherence spaces, events are in one to one correspondence with the prime states, which are singletons* $\{e\}$*. In the present framework,* $\{e\}$ *has no reason to be extensional in general, and there may be several ways to extend* $\{e\}$ *into a minimal state. These considerations should explain why we have chosen to concentrate on a concrete description of the morphisms of* $\mathbf{BS}_l$*.*

Next we define a tensor product. Its definition is dictated by (and its correctness follows from) the equation $(E \otimes E')^\perp = E^\perp \parr E'^\perp$.

Definition 13.4.23 (tensor – bistructures) *Let* E *and* E' *be two bistructures. The bistructure* $E \parr E'$ *is defined as follows:*

events are pairs (e, e') *where* $e \in E$ *and* $e' \in E'$,
$(e_1, e_1') \frown (e_2, e_2') \Leftrightarrow e_1 \frown e_2$ *and* $e_1' \frown e_2'$,
$(e_1, e_1') \leq^L (e, e') \Leftrightarrow e_1 \leq^L e$ *and* $e_1' \leq^L e'$,
$(e_1, e_1') \leq^R (e, e') \Leftrightarrow e_1 \leq^R e$ *and* $e_1' \leq^R e'$.

The operation $\otimes$ is extended to a functor as follows. Let $\phi \in D(E_1 \parr E_1')$ and $\psi \in D(E_2 \parr E_2')$, and set

$$\phi \otimes \psi = \{((e_1, e_2), (e_1', e_2')) \mid (e_1, e_1') \in \phi \text{ and } (e_2, e_2') \in \psi\}.$$

Theorem 13.4.24 *The category $\mathbf{BS}_l$ is $\star$-autonomous. The unit is defined by $I = (\{\star\}, id, id, id)$.*

Proof. The proof is a straightforward extension of the proof that $\mathbf{Coh}_l$ is $\star$-autonomous (theorem 13.2.15). We have, say:

$$\begin{array}{rcl} ((e_1, e_1'), e_1'') \leq^L_{E \otimes E' \parr E''} ((e_2, e_2'), e_2'') & \Leftrightarrow & e_2 \leq^R_E e_1 \text{ and } e_1' \leq^R_{E'} e_2' \text{ and } e_1'' \leq^R_{E''} e_2'' \\ & \Leftrightarrow & (e_1, (e_1', e_1'')) \leq^R_{E \parr (E' \parr E'')} (e_2, (e_2', e_2'')). \end{array}$$

□

Remark 13.4.25 *Like in the coherence model, we have $I^\perp = I$.*

We now define a related cartesian closed category of order-extensional stable maps.

Definition 13.4.26 *We define the category* **BS** *as follows: objects are bistructures; and for any $E.E'$, $\mathbf{BS}[E, E']$ consists of the functions from $D(E)$ to $D(E')$ which are $\sqsubseteq^R$ stable and $\sqsubseteq$ monotonic.*

We did not require $\sqsubseteq$ continuity in definition 13.4.26, because it is an implied property.

Lemma 13.4.27 *Let E and E' be bistructures. If $f : D(E) \to D(E')$ is $\sqsubseteq^R$ continuous and $\sqsubseteq$ monotonic, then it is also $\sqsubseteq$ continuous.*

Proof. Let $\{e_1, \ldots, e_n\} \sqsubseteq f(x)$. There exist $e_1', \ldots, e_n' \in f(x)$ such that $e_i \leq^L e_i'$ for all i. By $\sqsubseteq^R$ continuity, there exists a finite $x_1 \sqsubseteq^R x$ such that $e_1', \ldots, e_n' \in f(x_1)$, hence $\{e_1, \ldots, e_n\} \sqsubseteq f(x_1)$. □

Definition 13.4.28 (product – bistructures) *Let E and E' be two bistructures. The bistructure $E \times E'$ is defined as follows:*

events are either $e.1$ where $e \in E$ or $e'.2$ where $e' \in E'$,
$(e_1.i) \frown (e_2.j) \Leftrightarrow i = j$ and $e_1 \frown e_2$,
$(e_1.i) \leq^L (e_2.j) \Leftrightarrow i = j$ and $e_1 \leq^L e_2$,
$(e_1.i) \leq^R (e_2.j) \Leftrightarrow i = j$ and $e_1 \leq^R e_2$.

Proposition 13.4.29 *The category $\mathbf{BS}_l$ is cartesian. The terminal object is $1 = (\emptyset, \emptyset, \emptyset, \emptyset)$.*

We now relate the categories $\mathbf{BS}_l$ and **BS** through an adjunction that corresponds to the fundamental decomposition $E \to E' = (!E) \parr E'$. We define an "inclusion" functor $\subseteq$: $\mathbf{BS}_l \to \mathbf{BS}$ as follows.

Definition 13.4.30 *We set*

$$\subseteq(E) = E \quad \subseteq(\phi)(x) = \{e' \mid \exists\, e \in x \;\; (e, e') \in \phi\}.$$

Proposition 13.4.31 *The data of definition 13.4.30 define a functor from* $\mathbf{BS}_l$ *to* $\mathbf{BS}$.

PROOF HINT. To check that $\subseteq(\phi)$ is $\sqsubseteq$ monotonic, we use a technique similar to the one used in the proof of proposition 13.4.20 (application is a special case of composition). □

The following connective **!** allows us to go the other way around, from **BS** to $\mathbf{BS}_l$.

Definition 13.4.32 (exponential – bistructures) *Let E be a bistructure. The bistructure $!E$ is defined as follows:*

the events are the finite states of E,
$x_1 \frown x_2 \Leftrightarrow x_1 \uparrow^R x_2$,
$\leq^L$ *is* $\sqsubseteq^L$,
$\leq^R$ *is* $\sqsubseteq^R$.

Proposition 13.4.33 $!E$ *is a well-defined bistructure.*

PROOF. Obviously, $\sqsubseteq^R$ is a partial order. By lemma 13.4.12, $\sqsubseteq^L$ is a partial order and (B3) holds. And (B2) holds a fortiori, by definition of $\frown$.

$(B1)$ By the definition of $\frown_{!E}$ we can rephrase (B1) as

$$(x_1 \downarrow^L x_2 \text{ and } x_1 \uparrow^R x_2) \Rightarrow x_1 = x_2.$$

Let $x_3 \sqsubseteq^L x_1, x_2$, and let $e \in x_1$. Let $e_0 \in x_3$ and $e_1 \in x_1$ be such that $e \preceq_{x_1} e_1 \geq^L e_0$. Exploiting $x_3 \sqsubseteq x_2$ and $x_1 \uparrow^R x_2$, we get $e_1 \in x_2$, and $e \in x_2$ then follows by lemma 13.4.11. This completes the proof of $x_1 \sqsubseteq^R x_2$. The converse inclusion is proved symmetrically, exploiting $x_3 \sqsubseteq^L x_2$ and $x_3 \sqsubseteq x_1$.

$(B4)$ We show that $(\geq^L \cup \leq^R)^\star$ is antisymmetric. Since $\sqsubseteq^R$ and $\sqsupseteq^L$ are both partial orders, we can consider a sequence $x_0 \sqsubseteq^R x'_0 \sqsupseteq^L x_1 \cdots x'_{n-1} \sqsupseteq^L x_n = x_0$ and prove that $x_0 = x'_0 = x_1 = \cdots = x'_{n-1} = x_n$. The proof goes through two claims.

Claim 1. $X = \bigcap_{i=1\ldots n} x_i$ is a state.

X is clearly consistent as a subset of, say, x_0. Let $e \in X$ and $e_1 \leq^R e$. Since $e \in x_i$, there exists $e_1^i \in x_i$ such that $e_1 \leq^L e_1^i$, for all $i \geq 1$. Since $x_i \sqsubseteq x'_{i-1}$, there exists $e_1'^i \in x'_{i-1}$ such that $e_1^i \leq^L e_1'^i$, for all $i \geq 1$. But from $e_1 \leq^L e_1'^i$, $e_1 \leq^L e_1^{i-1}$ and $x_{i-1} \sqsubseteq^R x'_{i-1}$ we conclude $e_1'^i = e_1^{i-1}$, therefore $e_1^{i-1} \geq^L e_1^i$. On the other hand, from $e_1 \leq^L e_1^0$, $e_1 \leq^L e_1^n$ and $x_0 = x_n$, we obtain $e_1^0 = e_1^n$. Since $\leq^L$ is a partial order, we get $e_1^0 = \cdots = e_1^i = \cdots = e_1^n$, hence $e_1^0 \in X$.

Claim 2. $x'_0 \sqsubseteq^R X$.

Let $e_0 \in x'_0$. By lemma 13.4.12, we can find $e'_0 \in x'_0$ and $e_1 \in x_1$ such that $e_0 \preceq_{x'_0} e'_0 \geq^L e_1$. We continue in this way and find

$$\begin{array}{l} e'_1 \in x'_1, e_2 \in x_2 \text{ such that } e_1 \preceq_{x'_1} e'_1 \geq^L e_2, \ldots, \\ e'_{n-1} \in x'_{n-1}, e_n \in x_n = x_0 \text{ such that } e_{n-1} \preceq_{x'_{n-1}} e'_{n-1} \geq^L e_n \ . \end{array}$$

We continue round the clock, generating $e_{n+i} \in x_i$, $e_{2n+i} \in x_i, \ldots, e_{kn+i} \in x_i$. Since x_i is finite, there exist k_1, k_2 for which $e_{k_1 n} = e_{k_2 n} \in x_0$. By the antisymmetry of $\preceq$ (in E), we obtain

$$e_{k_1 n} = e_{k_1 n+1} = \cdots = e_{(k_1+1)n} = \cdots = e_{k_2 n}.$$

Therefore $e_{k_1 n} \in X$, since $e_{k_1 n+i} \in x_i$ for all i. Our next goal is to carry this back to e_0, and show $e_0 \in X$. Suppose that we have proved $e_m \in X$. We have

$$\begin{array}{ll} e_m = e'_{m-1} & \text{since } e_m, e'_{m-1} \in x'_{m-1} \text{ and } e_m \downarrow^L e'_{m-1} \\ e_{m-1} \preceq_X e'_{m-1} & \text{since by assumption } e_m \in X \ . \end{array}$$

Hence $e_{m-1} \in X$. Finally, we arrive at $e_0 \in X$.

We can now prove (B4). From $x_0 \sqsubseteq^R x'_0 \sqsubseteq^R X$ we get $x_0 = x'_0 = X$. From $x'_0 \sqsupseteq^L x_1$ and $x'_0 = X \sqsubseteq^R x_1$ we get $x'_0 = x_1$ by remark 13.4.9, and, progressively, $x_0 = x'_0 = x_1 = \cdots = x'_{n-1} = x_n$, as desired.

($B5$) Let x be a finite state. Let $\overline{x} = \{e \mid \exists e_0 \in x \ \ e \preceq e_0\}$. This set is finite. It follows from lemma 13.4.12 that any y such that $y \ (\sqsupseteq^L \cup \sqsubseteq^R)^\star \ x$ is a subset of $\overline{x}$, from which (B5) follows. □

Lemma 13.4.34 *Let E and E' be bistructures. A function $f : D(E) \to D(E')$ is $\sqsubseteq^R$ continuous iff, for any e', x, $e' \in f(x)$ implies $e' \in f(y)$ for some finite $y \sqsubseteq^R x$.*

Proof. If f is continuous and $e' \in f(x)$, then $[e']_{f(x)} \sqsubseteq^R f(x)$ (cf. lemma 13.4.13), and by continuity there exists a finite x_1 such that $[e']_{f(x)} \sqsubseteq^R f(x_1)$, hence a fortiori $e' \in f(x_1)$. Conversely, let $\{e_1, \ldots, e_n\} \sqsubseteq^R f(x)$. Then let $x_1, \ldots, x_n$ be finite and such that $e_i \in f(x_i)$ for all i. Then $\bigcup_{i=1\cdots n} x_i$ is finite, and $\{e_1, \ldots, e_n\} \sqsubseteq^R f(\bigcup_{i=1\cdots n} x_i)$. □

Theorem 13.4.35 *The operation ! extends to a functor ! : **BS** $\to$ **BS**$_l$ which is left adjoint to $\subseteq$.*

Proof. The correspondences between states of $!E \multimap E'$ and the stable and order-extensional functions are as follows:

$$\begin{array}{rcl} f & \mapsto & \mathit{trace}(f) = \{(x, e') \mid e' \in f(x) \text{ and } (y \sqsubseteq^R x \text{ and } e' \in f(y) \Rightarrow y = x)\} \\ \phi & \mapsto & \mathit{fun}(\phi)(z) = \{e' \mid \exists y \sqsubseteq^R z \ \ (y, e') \in \phi\} \ . \end{array}$$

We show that $\mathit{trace}(f)$ is a state. It is a set of events of $!E \multimap E'$ by lemma 13.4.34. Let $(x_1, e'_1), (x_2, e'_2) \in \mathit{trace}(f)$ with $x_1 \frown x_2$, i.e. $x_1 \uparrow^R x_2$. Let x be such that $x_1, x_2 \leq^R x$. Then $e'_1 \frown e'_2$, since $e'_1, e'_2 \in f(x)$. Suppose moreover $e'_1 = e'_2$. Then $x_1 = x_2$, by stability. This completes the proof of consistency. Let $(x, e') \in \mathit{trace}(f)$ and

$(x_1, e_1') \leq^R (x, e')$. We look for $(x_2, e_2') \in \mathit{trace}(f)$ such that $(x_1, e_1') \leq^L (x_2, e_2')$. From $e' \in f(x)$ we get that e' is in the $\leq$ downward closure of $f(x_1)$ by $\sqsubseteq$ monotonicity. Since $e_1' \leq^R e'$, e_1' is in the $\leq^L$ downward closure of $f(x_1)$ by lemma 13.4.8, that is, $e_2' \in f(x_1)$ for some $e_2' \geq^L e_1'$. Therefore, by the definition of the trace, $(x_2, e_2') \in \mathit{trace}(f)$ for some $x_2 \sqsubseteq^R x_1$. This pair fits, and this completes the proof of extensionality.

In the other direction, the $\sqsubseteq^R$ stability of $\mathit{fun}(\phi)$ is obvious from the definition of *fun* and the consistency of ϕ. We check that $\mathit{fun}(\phi)$ is $\sqsubseteq$ monotonic. Let $y \sqsubseteq z$, and let $e' \in \mathit{fun}(\phi)(y)$. Let $x_1 \sqsubseteq^R y$ be such that $(x_1, e') \in \phi$. Let x_2 be such that $x_1 \sqsubseteq^L x_2 \sqsubseteq^R z$. Since $(x_2, e') \sqsubseteq^L (x_1, e')$, by extensionality there exists $(x_3, e_3') \in \phi$ such that $(x_2, e') \sqsubseteq^L (x_3, e_3')$. We have $e' \leq^L e_3'$, and $e_3' \in f(z)$ since $e_3' \in f(x_3)$ and $x_3 \sqsubseteq^R z$. □

The required isomorphisms relating additives and multiplicatives are as follows:

- $!1 \cong I$: The only state of 1 is $\emptyset$, which corresponds to the unique event $\star$ of I.
- $!(A \times B) \cong (!A) \otimes (!B)$: By the definition of product, x is a state of $E \times E'$ iff $x_1 = \{e \in E \mid (e.1) \in x\}$ and $x_2 = \{e \in E \mid (e.2) \in x\}$ are states of E, E' and $x = x_1 \cup x_2$. This establishes a bijective correspondence $x \leftrightarrow (x_1, x_2)$ between the events of $!(A \times B)$ and those of $(!A) \otimes (!B)$.

Altogether, we have constructed a $!\star$-autonomous category of bistructures.

13.5 Chu Spaces and Continuity

The Chu construction (see [abPC79, Bar91b]) allows to construct a $\star$-autonomous category out of a monoidal category with finite limits. Here we present the construction over the category of sets, which is enough for our purposes. By imposing some order-theoretic axioms, we arrive at the notion of casuistry. Roughly speaking, a casuistry is a dcpo together with a choice of an appropriate collection of Scott opens. A linear morphism between two casuistries is a function whose inverse image maps chosen opens back to chosen opens. For any continuous function f, the inverse image $f^{-1}(U)$ of a chosen open U is open, but is not necessarily a chosen open. An exponential construction allows to fill the gap between the morphisms of casuistries and the continuous functions. The material of this section is based on [Lam94].

Definition 13.5.1 (Chu space) *Let K be a set. A Chu space over K (Chu space for short) is a triple $\mathbf{A} = (A_\star, A^\star, \langle \_, \_ \rangle)$ where $A_\star$ and $A^\star$ are sets and $\langle \_, \_ \rangle$ is a function from $A_\star \times A^\star$ to K, called* agreement function. *A morphism $\mathbf{f} : \mathbf{A} \to \mathbf{B}$ of Chu spaces is a pair $(f_\star, f^\star)$ of functions, where $f_\star : A_\star \to B_\star$ and $f^\star : B^\star \to A^\star$, satisfying, for all $x \in A_\star, \beta \in B^\star$:*

$$(\star_\star) \quad \langle f_\star(x), \beta \rangle = \langle x, f^\star(\beta) \rangle.$$

The mapping $\langle \_, \_ \rangle$ can be equivalently presented as a function $l : A_\star \to (A^\star \to K)$ or a function $r : A^\star \to (A_\star \to K)$. If l is injective, we say that $\mathbf{A}$ is left-separated, and symmetrically, if r is injective, we say that $\mathbf{A}$ is right-separated. A separated *Chu space*

is a Chu space which is both left and right-separated. We write **Chu** *for the category of Chu spaces, and* $\mathbf{Chu}_s$ *for the full subcategory of separated Chu spaces.*

There are two obvious forgetful functors $_\star$ *and* * *(covariant and contravariant, respectively) from* **Chu** *to the category of sets:*

$$\begin{array}{ll} \mathbf{A}_\star = A_\star & (f_\star, f^\star)_\star = f_\star \\ \mathbf{A}^\star = A^\star & (f_\star, f^\star)^\star = f^\star \,. \end{array}$$

Lemma 13.5.2 *The following are equivalent formulations of* $(^\star_\star)$*:*

$$\begin{array}{ll} (^\star_{\star l}) & l(f_\star(x)) = l(x) \circ f^\star \\ (^\star_{\star r}) & r(f^\star(\beta)) = r(\beta) \circ f_\star \,. \end{array}$$

Lemma 13.5.3 *Every right-separated Chu space is isomorphic to a Chu space* $\mathbf{A}$ *where* $A^\star$ *is a set of functions from* $A_\star$ *to* K*, and where* $\langle x, f\rangle = f(x)$*. If moreover* $K = \{\bot, \top\}$*, every right-separated Chu space is isomorphic to a Chu space* $\mathbf{A}$*, where* $A^\star$ *is a set of subsets of* $A_\star$*, and where agreement is membership. Such a Chu space is called right-strict. Left-strict separated Chu spaces are defined similarly.*

Example 13.5.4 *Every topological space* (X, Ω) *is a right-strict Chu space. It is left-separated exactly when it is* T_0 *(cf. section 1.2). Moreover, the morphisms between topological spaces viewed as Chu spaces are exactly the continuous functions (see lemma 13.5.5), so that topological spaces and continuous functions may be considered a full subcategory of* $\mathbf{Chu}_s$*.*

Lemma 13.5.5 *A morphism of right-separated Chu spaces* $\mathbf{A}$ *and* $\mathbf{B}$ *is equivalently defined as a function* $f_\star : A_\star \to B_\star$ *such that*

$$\forall\, \beta \in B^\star \;\; \exists\, \alpha \in A^\star \;\; r(\beta) \circ f_\star = r(\alpha).$$

If moreover $\mathbf{A}$ *and* $\mathbf{B}$ *are right-strict, this condition boils down to*

$$\forall\, \beta \in B^\star \;\; f^{-1}(\beta) \in A^\star.$$

Proof. Let $(f_\star, f^\star)$ be a morphism. Then $f_\star$ satisfies the condition of the statement by $(^\star_{\star r})$. Conversely, the injectivity of r guarantees the uniqueness of α, hence the formula of the statement defines a function $f^\star : B^\star \to A^\star$. □

Lemma 13.5.6 *Let* $\mathbf{A}$ *and* $\mathbf{B}$ *be Chu spaces, and suppose that* $\mathbf{A}$ *is right-separated and that* $\mathbf{B}$ *is left-separated. Then a morphism from* $\mathbf{A}$ *to* $\mathbf{B}$ *is equivalently defined as a function* $h : A_\star \times B^\star \to K$ *whose curryings factor through* l_B *and* r_A*.*

Proof. Let $(f_\star, f^\star)$ be a morphism. Then the required h is defined by:

$$h(x, \beta) = \langle f_\star(x), \beta\rangle = \langle x, f^\star(\beta)\rangle$$

and the factorisations are given by $(^\star_{\star l})$ and $(^\star_{\star r})$, respectively. Conversely, the two factorisations determine two functions $f_\star : A_\star \to B_\star$ and $f^\star : B^\star \to A^\star$, and $(^\star_\star)$ holds by construction. □

Definition 13.5.7 (tensor – Chu spaces) *Let* $\mathbf{A}$ *and* $\mathbf{A}'$ *be two Chu spaces. Their tensor product* $\mathbf{A} \otimes \mathbf{A}'$ *is defined as follows:*

- $(\mathbf{A} \otimes \mathbf{A}')_\star = A_\star \times A'_\star$.
- $(\mathbf{A} \otimes \mathbf{A}')^\star$ *consists of the pairs of functions* (f, g)*, with* $f : A_\star \to A'^\star$ *and* $g : A'_\star \to A^\star$*, which satisfy* $\langle x, g(x') \rangle = \langle x', f(x) \rangle$*, for all* $x \in A_\star, x' \in A'_\star$.
- $\langle (x, x'), (f, g) \rangle = \langle x, g(x') \rangle = \langle x', f(x) \rangle$.

Definition 13.5.8 (linear negation – Chu spaces) *The linear negation of a Chu space is defined as follows (where* γ *is as in definition 13.2.3):*

$$(A_\star, A^\star, \langle \_, \_ \rangle_A)^\perp = (A^\star, A_\star, \langle \_, \_ \rangle \circ \gamma).$$

Proposition 13.5.9 *1. The category* **Chu** *is* $\star$*-autonomous. The tensor product is as given in definition 13.5.7, the unit is* $I = (\{\star\}, K, \pi')$ *and the dualising object is* $\perp = (K, \{\star\}, \pi)$.

2. There exists a natural bijection $\mathbf{Chu}[I, \mathbf{A}] \cong A_\star$.

Proof. For the symmetric monoidal structure, we just check that $\mathbf{A} \otimes I \cong \mathbf{A}$. The $_\star$ component of $\mathbf{A} \otimes I$ is $A_\star \times \{\star\}$, which is $A_\star$ up to natural bijection. An element of the * component of $\mathbf{A} \otimes I$ can be considered as a pair of a function $f : A_\star \to K$ and an element $\alpha \in A^\star$. Looking at the condition linking f and α, we see that it boils down to the definition of f as $\lambda x.\langle x, \alpha \rangle$.

By a similar reasoning, we get (2). To establish the closed structure, we rely on proposition 13.2.8. The dualising functor is given by definition 13.5.8. The required isomorphisms $\mathbf{A}^{\perp\perp} \cong \mathbf{A}$ are actually identities. We verify the bijections:

$$\mathbf{Chu}[I, (\mathbf{A} \otimes \mathbf{B}^\perp)^\perp] \cong \mathbf{Chu}[\mathbf{A}, \mathbf{B}].$$

Let $\mathbf{f} : \mathbf{A} \to \mathbf{B}$:

$$\begin{array}{ll} \mathbf{f} \in (\mathbf{A} \otimes \mathbf{B}^\perp)^\star & \text{by the definition of } \otimes \\ \mathbf{f} \in (\mathbf{A} \otimes \mathbf{B}^\perp)^\perp_\star & \text{by the definition of } ^\perp . \end{array}$$

We conclude by using (2). To see that **Chu** is $\star$-autonomous, we proceed essentially as in proposition 13.2.15: the canonical morphism from $\mathbf{A}$ to $(\mathbf{A}\ (\perp)\ (\perp$ is the identity modulo identifications similar to those used above for proving $\mathbf{A} \otimes I \cong \mathbf{A}$. □

Lemma 13.5.10 (slice condition) *The tensor product of two right-separated Chu spaces* $\mathbf{A}$ *and* $\mathbf{A}'$ *is right-separated, and can be reformulated as follows:*

- $(\mathbf{A} \otimes \mathbf{A}')_\star = A_\star \times A'_\star$.
- $(\mathbf{A} \otimes \mathbf{A}')^\star$ *consists of the functions* $h : A_\star \times A'_\star \to K$ *whose curryings factor through* $A^\star$ *and* $A'^\star$*, that is such that, for some* f *and* g*:*

$$\Lambda_l(h) = r \circ f \qquad \Lambda_l(h \circ \gamma) = r \circ g.$$

- $\langle (x,y), h\rangle = h(x,y)$.

If moreover $\mathbf{A}$ *and* $\mathbf{A}'$ *are right-strict, then the reformulation says that* $(\mathbf{A}\otimes\mathbf{A}')^\star$ *consists of the subsets* U *of* $A_\star \times B_\star$ *satisfying the following condition, called slice condition:*

$$(\forall\, x \in A_\star\ \{x' \mid (x,x') \in U\} \in A'^\star) \quad \textit{and} \quad (\forall\, y \in A'_\star\ \{x \mid (x,x') \in U\} \in A^\star).$$

Proof. By definition of $^\perp$ and by proposition 13.5.9, an element of $(\mathbf{A}\otimes\mathbf{B})^\star$ can be described as a morphism of $\mathbf{Chu}[\mathbf{A},\mathbf{B}^\perp]$. The conclusion then follows from lemma 13.5.6. □

Unlike right-separation, left separation has to be forced upon the tensor product structure.

Lemma 13.5.11 *With every Chu space* $\mathbf{A}$ *we associate a left-separated Chu space* $\mathbf{A}_l$ *as follows:*

- $(A_l)^\star = A^\star$.
- $(A_l)_\star = A_\star/\approx$, *where* $\approx$ *is the equivalence relation defined by*

$$x_1 \approx x_2 \Leftrightarrow \forall\, \alpha \in A^\star\ \langle x_1,\alpha\rangle = \langle x_2,\alpha\rangle.$$

- $\langle [x],\alpha\rangle = \langle x,\alpha\rangle$.

If moreover $\mathbf{A}$ *is right-separated, then* $\mathbf{A}_l$ *is separated. There is a symmetric construction* $\mathbf{A}_r$ *which forces right separation.*

Proposition 13.5.12 *The statement of proposition 13.5.9 holds true replacing* $\mathbf{Chu}$ *by* $\mathbf{Chu}_s$ *and redefining the tensor product as* $\mathbf{A}\otimes_s\mathbf{B} = (\mathbf{A}\otimes\mathbf{B})_l$. *(We shall omit the subscript in* $\otimes_s$ *if no ambiguity can arise.)*

Proof. The proof is by a straightforward adaptation of the proof of proposition 13.5.9. Notice that $\mathbf{f}:\mathbf{A}\to\mathbf{B}$ reads as $\mathbf{f}\in(\mathbf{A}\otimes_s\mathbf{B}^\perp)^\star$ since $(\mathbf{A}\otimes_s\mathbf{B}^\perp)^\star = (\mathbf{A}\otimes\mathbf{B}^\perp)^\star$. □

Our last step consists in adding directed lub's, more precisely directed unions. Fron now on, we assume that $K=\{\perp,\top\}$, and confuse freely a function h into K with the set it is the characteristic function of.

Definition 13.5.13 (casuistry) *A casuistry is a separated Chu space* $\mathbf{A}$ *such that both* $A_\star$ *and* $A^\star$ *are dcpo's under the induced orders defined by*

$$x \le x' \Leftrightarrow l(x) \subseteq l(x')$$

(and symmetrically for $A^\star$*), and moreover* $l(\bigvee\Delta) = \bigcup l(\Delta)$ *for any directed* Δ, *and similarly for* r. *We call* $\mathbf{Cas}$ *the full subcategory of* $\mathbf{Chu}_s$ *whose objects are casuistries.*

Exercise 13.5.14 *Given a right-strict Chu space* $\mathbf{A}$*, we say that* $x \in A_\star$ *is empty if* $\forall U \in A^\star \;\; x \notin U$*. Consider now two casuistries* $\mathbf{A}$ *and* $\mathbf{B}$*. Show that, for any* $[x,y] \in (\mathbf{A} \otimes_s \mathbf{B})_\star$*:*

$$[x,y] = \begin{cases} \{(x,y)\} & \text{if neither } x \text{ nor } y \text{ are empty} \\ \text{the empty element of } (\mathbf{A} \otimes_s \mathbf{B})_\star & \text{otherwise.} \end{cases}$$

Lemma 13.5.15 *A topological space* (X,Ω) *viewed as a Chu space is a casuistry iff its topology is* T_0*, its specialisation order is a dcpo, and every open is Scott open.*

Proof. A topology is a fortiori closed under directed unions. Thus the requirement concerns X. Notice that $l(\bigvee \Delta) = \bigcup l(\Delta)$ reads as

$$\forall U \in \Omega \;\; \bigvee \Delta \in U \Leftrightarrow (\exists \delta \in \Delta \;\; \delta \in U).$$

□

Lemma 13.5.16 *All morphisms between casuistries preserve directed lub's.*

Proof. It is enough to check $l(f(\bigvee \Delta)) \subseteq \bigcup l(f(\Delta))$. We have

$$\begin{array}{rcl} l(f(\bigvee \Delta)) & = & \{U \mid f(\bigvee \Delta) \in U\} \\ & = & \{U \mid \bigvee \Delta \in f^{-1}(U)\} \end{array}$$

and we conclude by exploiting that $f^{-1}(U)$ is Scott open. □

Proposition 13.5.17 *The category* **Cas** *is* $\star$*-autonomous, and all the constructions are the restrictions of the constructions on* $\mathbf{Chu}_s$*. Lub's in* $(\mathbf{A} \otimes_s \mathbf{B})_\star$ *are coordinate-wise.*

Proof. Let $\mathbf{A}$ and $\mathbf{B}$ be casuistries. We sketch the proof that $\mathbf{A} \otimes_s \mathbf{B}$ is a casuistry. Consider a directed subset Δ of $(\mathbf{A} \otimes_s \mathbf{B})^\star = (\mathbf{A} \otimes \mathbf{B})^\star$. The reason why $\bigcup \Delta$ satisfies the slice conditions (cf. lemma 13.5.10) is that a slice of a directed union is a directed union of slices. Consider now a directed subset Δ of $(\mathbf{A} \otimes_s \mathbf{B})_\star$. Without loss of generality, we can assume that the empty element is not in Δ, hence (cf. exercise 13.5.14) that $\Delta \subseteq A_\star \times B_\star$. We claim:

$$l(\bigvee \pi(\Delta), \bigvee \pi'(\Delta)) = \bigcup l(\Delta).$$

In the direction $\supseteq$, one first establishes pointwise monotonicity:

$$(x_1 \leq x_2 \text{ and } y_1 \leq y_2) \Rightarrow l(x_1,y_1) \subseteq l(x_2,y_2).$$

If $(x_1,y_1) \in U$, then $x_1 \in \{x \mid (x,y_1) \in U\}$. Hence

$$x_2 \in \{x \mid (x,y_1) \in U\} \text{ since } \{x \mid (x,y_1) \in U\} \text{ is open and } x_1 \leq x_2.$$

We have obtained $(x_2,y_1) \in U$, from which we obtain $(x_2,y_2) \in U$ by a similar reasoning, now using the slice $\{y \mid (x_2,y) \in U\}$. The direction $\subseteq$ is proved similarly, making use of the fact that the slices are Scott open by definition of casuistries. □

Proposition 13.5.18 *The categories* **Chu**, $\mathbf{Chu}_s$, *and* **Cas** *are cartesian. The terminal object is* $(\{\star\}, \emptyset)$ *(with vacuous* $\langle \_, \_ \rangle$*). The product in* **Chu** *is given by*

$$\begin{array}{l} (\mathbf{A} \times \mathbf{A}')_\star = A_\star \times A'_\star \\ (\mathbf{A} \times \mathbf{A}')^\star = A^\star + A^\star \\ \langle (x, x'), \alpha \rangle = \begin{cases} \langle x, \alpha \rangle & \text{if } \alpha \in A^\star \\ \langle x', \alpha \rangle & \text{if } \alpha \in A'^\star \end{cases} . \end{array}$$

In $\mathbf{Chu}_s$, *and* **Cas**, *right separation has to be forced, and the product* $\times_s$ *(or* $\times$ *if no ambiguity can arise) is given by*

$$\mathbf{A} \times_s \mathbf{A}' = (\mathbf{A} \times \mathbf{A}')_r.$$

If A *and* A' *are right-strict, then we can reformulate their product (in* $\mathbf{Chu}_s$, *and* **Cas***) as follows:*

$$\begin{array}{l} (\mathbf{A} \times \mathbf{A}')_\star = A_\star \times A'_\star \\ (\mathbf{A} \times \mathbf{A}')^\star = \{U \times A' \mid U \in A^\star\} \cup \{A \times U' \mid U' \in A^\star\} . \end{array}$$

The order induced on $A_\star \times A'_\star$ *is the pointwise ordering.*

PROOF. We only show that 1 is terminal, and that the induced order on products is pointwise. By the vacuity of $1^\star$, being a morphism into 1 amounts to being a function to $!_\star = \{\star\}$. Suppose that $(x, x') \leq (y, y')$, and that $x \in U$. Then $(x, x') \in U \times A' \in (\mathbf{A} \times \mathbf{A}')^\star$. Hence $(y, y') \in U \times A'$, i.e. $y \in A$. Similarly we get $x' \leq y'$. The converse direction is proved similarly. □

Finally, we define an adjunction between the category of casuistries and the category of dcpo's.

Proposition 13.5.19 *Consider the two following functors* $\subseteq$: **Dcpo** $\to$ **Cas** *and* $!$: **Cas** $\to$ **Dcpo**, *defined as follows:*

$$\begin{array}{ll} \subseteq (D, \leq) = (D, \tau_S(D), \in) & \subseteq (f) = f \\ !(X_\star, X^\star, \langle \_, \_ \rangle) = (X_\star, \leq) & !(f) = f \end{array}$$

where $\tau_S(D)$ *denotes Scott topology (cf. definition 1.2.1), and where* $\leq$ *in the second line is the induced ordering (cf. definition 13.5.13). Then* $\subseteq \dashv !$. *Moreover, the induced comonad* $\subseteq \circ !$, *written simply* $!$, *satisfies the isomorphisms of definition 13.2.12, i.e. casuistries together with* $!$ *form a* $!\star$*-autonomous category, whose Kleisli category is equivalent to the category* **Dcpo**.

PROOF. To establish the adjunction, we have to prove that, given $(D, \leq)$ and $(X_\star, X^\star, \in)$, a function $f : D \to X_\star$ is a Chu morphism from $(D, \tau_S(D), \in)$ to $(X_\star, X^\star, \in)$ iff it is a directed lub preserving function from $(D, \leq)$ to $(X_\star, \leq)$. If f is a Chu morphism, then it preserves directed lub's with respect to the induced orders by lemma 13.5.16. But (cf. lemma 1.2.3) the induced, or specialised, order of a Scott topology is the original order, i.e. $! \circ \subseteq = id$. Hence f preserves the lub's with respect to the order $\leq$ of D. If f

preserves directed lub's, then it is continuous with respect to the Scott topologies, and a fortiori it is a Chu morphism from $(D, \tau_S(D), \in)$ to $(X_\star, X^\star, \in)$, since $X^\star \subseteq \tau_S(X_\star)$ by proposition 13.5.15.

We already observed that $! \circ \subseteq = id$, hence a fortiori $!$ is surjective on objects. As a consequence (cf. remark 13.2.14), the Kleisli category is equivalent to **Dcpo**, and thus is cartesian closed, which in turn entails the isomorphisms $!(\mathbf{A} \times \mathbf{B}) \cong (!\mathbf{A}) \otimes (!\mathbf{B})$, by proposition 13.2.17. We are left to show $!1 \cong I$. Recall that I, formulated as a right-strict Chu space, is $(\{\star\}, \{\emptyset, \{\star\}\})$. We have

$$I_\star = (!1)_\star \quad \text{and} \quad I^\star \text{ is the Scott topology over } \{\star\}.$$

□

Remark 13.5.20 *We have expressed the comonad for the stable model and for the hypercoherence model via an adjunction of the form $! \dashv \subseteq$, while we just presented a continuous model via an adjunction of the form $\subseteq \dashv !$. One should not take that too seriously. In each situation, we have called $\subseteq$ the obvious inclusion at hand. But both the stable $\subseteq$ and the continuous $!$ are faithful functors: in particular, morphisms in* **Cas** *can be considered as special Scott-continuous functions (those mapping (chosen) opens to chosen opens).*

A more liberal $!$ (leading to a larger Kleisli category), also taken from [Lam94], is described in exercise 13.5.21.

Exercise 13.5.21 *Call a topological space (X, Ω) anti-separated if $\{x \mid (x,x) \in U\} \in \Omega$, for any subset U of $X \times X$ satisfying the slice condition (cf. lemma 13.5.10). Show that* **Dcpo** *is a full subcategory of the category* $\mathbf{\Delta Cas}$ *of anti-separated topological spaces which moreover viewed as Chu spaces are casuistries. Show that* **Cas** *together with the following definition of $!$ yields a $!\star$-autonomous category whose Kleisli category is equivalent to* $\mathbf{\Delta Cas}$*:*

$$!\mathbf{A} \quad = \quad \textit{the smallest anti-separated topology on } A_\star \textit{ containing } A^\star.$$

Hints: (1) The anti-separation condition says that the diagonal function $\lambda x.(x,x)$ is continuous from X (viewed as a separated Chu space) to $X \otimes X$. (2) Follow the guidelines of remark 13.2.21. (2) In order to prove $U \dashv !$, give an inductive definition of $!(\mathbf{A})^\star$.

Chapter 14

Sequentiality

This chapter is devoted to the semantics of sequentiality. At first-order, the notion of sequential function is well-understood, as summarized in theorem 6.5.4. At higher orders, the situation is not as simple. Building on theorem 13.3.16, Ehrhard and Bucciarelli have developped a model of strongly stable functions, which we have described in section 13.3. But in the strongly stable model an explicit reference to a concept of sequentiality is lost at higher orders. Here there is an intrinsic difficulty: there does not exist a cartesian closed category of sequential functions (see theorem 14.1.16). Berry suggested that replacing functions by morphisms of a more concrete nature, retaining informations on the order in which the input is explored in order to produce a given part of the output, could be a way to develop a theory of higher-order sequentiality. This intuition gave birth to the model of sequential algorithms of Berry and Curien, which is described in this chapter.

In section 14.1 we introduce Kahn and Plotkin's (filiform and stable) concrete data structures and sequential functions between concrete data structures. This definition generalizes Vuillemin's definition 6.5.1. A concrete data structure consists of cells that can be filled with a value, much like a PASCAL record field can be given a value. A concrete data structure generates a cpo of states, which are of sets of pairs (cell,value), also called events (cf. section 12.3). Cells generalize the notion of argument position that plays a central role in Vuillemin's definition of sequential function. Kahn-Plotkin's definition of sequential function is based on cells, and reads roughly as follows: for a given input x and output cell c', if c' is filled in $f(y)$ for some $y \geq x$, then there exists a cell c, depending on x and c' only, such c is filled in any such y. In other words, it is necessary to compute the value of c in order to fill c'. Such a cell c is called a sequentiality index at (x, c'). The category of sequential functions on concrete data structures is cartesian, but not cartesian closed.

In section 14.2, we define sequential algorithms on concrete data structures. They can be presented in different ways. We first define an exponent concrete data structure, whose states are called sequential algorithms. The notion of ab-

stract algorithm provides a more intuitive presentation. An abstract algorithm is a partial function that maps a pair of a (finite) input state x and an output cell c' to either an output value v' (if $(c', v') \in f(x)$ or to an input cell c, where c is a sequentiality index at (x, c'). Hence sequential algorithms involve an explicit choice of sequentiality indexes. Many functions admit more than one sequentiality index for a given pair (x, c'). For example, adding two numbers requires computing these two numbers. In the model of sequential algorithms, there exist *two* addition algorithms, one which computes the first argument then the second before adding them, while the other scans its input in the converse order. We show that sequential algorithms form a category. Due to the concrete nature of the morphisms, it takes some time until we can recognize the structure of a category. A third presentation of sequential algorithms as functions between domains containing error values is given in section 14.4.

In section 14.3, we present a linear decomposition of the category of sequential algorithms. We define symmetric algorithms, which are pairs of sequential functions, mapping input values to output values, and output exploration trees to input exploration trees, respectively. It is convenient to work with sequential data structures, which are a more symmetric reformulation of (filiform and stable) concrete data structures. Sequential data structures and symmetric algorithms are the objects and morphisms of a symmetric monoidal closed category called **AFFALGO**, which is related to the category of seauential algorithms through an adjunction. The category **AFFALGO** is also cartesian. Moreover the unit is terminal. Due to this last property, our decomposition is actually an affine decomposition (cf. remark 13.2.23). The category of symmetric algorithms is a full subcategory of a category of games considered by Lamarche [Lam92b]. Related categories are studied in [Bla72, Bla92, AJ92].

In section 14.4 we investigate an extension of PCF with a control operator *catch* (cf. section 8.5), and show that the model of sequential algorithms is fully abstract for this extension.

14.1 Sequential Functions

First we define the concrete data structures (cds's). We give some examples of cds's, and define the product of two cds's. We then define Kahn-Plotkin sequential functions [KP93], which generalise the first-order sequential functions of definition 6.5.1. The category of cds's and sequential functions is cartesian but not cartesian closed.

Definition 14.1.1 *A* concrete data structure *(or* cds*)* $\mathbf{M} = (C, V, E, \vdash)$ *is given by three sets* C*,* V*, and* E *of* cells*, of* values*, and of* events*, such that*

$$E \subseteq C \times V \quad \textit{and} \quad \forall c \in C \ \exists v \in V \ (c, v) \in E$$

and a relation $\vdash$ *between finite parts of* E *and elements of* C*, called* enabling *relation. We write simply* $e_1, \ldots, e_n \vdash c$ *for* $\{e_1, \ldots, e_n\} \vdash c$*. A cell* c *such that* $\vdash c$ *is called* initial*. Proofs of cells* c *are sets of events defined recursively as follows: If* c *is initial, then it has an empty proof. If* $(c_1, v_1), \ldots, (c_n, v_n) \vdash c$*, and if* $p_1, \ldots, p_n$ *are proofs of* $c_1, \ldots, c_n$*, then* $p_1 \cup \{(c_1, v_1)\} \cdots \cup p_n \cup \{(c_n, v_n)\}$ *is a proof of* c*. A* state *is a subset* x *of* E *such that:*

1. $(c, v_1), (c, v_2) \in x \Rightarrow v_1 = v_2$,
2. *if* $(c, v) \in x$*, then* x *contains a proof of* c .

The conditions (1) and (2) are called consistency and safety, respectively. The set of states of a cds $\mathbf{M}$*, ordered by set inclusion, is a partial order denoted by* $(D(\mathbf{M}), \leq)$ *(or* $(D(\mathbf{M}), \subseteq)$*). If* D *is isomorphic to* $D(\mathbf{M})$*, we say that* $\mathbf{M}$ *generates* D*. We assume moreover that our cds's are* well-founded, stable, *and* filiform*, by which we mean:*

- *Well-founded: The reflexive closure of the relation* $\ll$ *defined on* C *by*

  $$c_1 \ll c \text{ iff some enabling of } c \text{ contains an event } (c_1, v)$$

  is well founded, that is, there is no infinite sequence $\{c_n\}_{n \geq 0}$ *such that* $\cdots c_{n+1} \ll c_n \ll \cdots c_0$*.*

- *Stable: For any state* x *and any cell* c *enabled in* x*, if* $X \vdash c$*,* $X' \vdash c$*, and* $X, X' \subseteq x$*, then* $X = X'$*.*

- *Filiform: All the enablings contain at most one event.*

Remark 14.1.2 *Well-foundedness allows us to reformulate the safety condition as a local condition:*

2'. *If* $(c, v) \in x$*, then* x *contains an enabling* $\{e_1, \ldots, e_n\}$ *of* c*.*

Remark 14.1.3 *Almost all the constructions of sections 14.1 and 14.2 go through for well-founded and stable cds's that are not necessarily well-founded. But it simplifies notation to work with filiform cds's. In particular, in a filiform cds, a proof of a cell boils down to a sequence* $(c_1, v_1), \ldots, (c_n, v_n)$ *such that* $(c_i, v_i) \vdash c_{i+1}$ *for all* i*. Filiform cds's are in any case enough for our purposes. We shall make it explicit when "stable", or "filiform" are essential.*

Definition 14.1.4 *Let* x *be a set of events of a cds. A cell* c *is called:*

filled (with v) *in* x	*iff* $(c, v) \in x$
enabled in x *iff* x	*contains an enabling of* c
accessible from x	*iff it is enabled, but not filled in* x .

We denote by $F(x)$, $E(x)$, and $A(x)$ the sets of cells which are filled, enabled, and accessible in or from x ("F", "E" and "A" as "Filled", "Enabled" and "Accessible"), respectively. We write:

$$\begin{array}{ll} x <_c y & \text{if } c \in A(x), c \in F(y) \text{ and } x < y \\ x \prec_c y & \text{if } x <_c y \text{ and } x \prec y \text{ (cf. definition 12.3.8)} . \end{array}$$

Proposition 14.1.5 *1. Let $\mathbf{M}$ be a cds. The partial order $(D(\mathbf{M}), \leq)$ is a Scott domain whose compact elements are the finite states. Upper-bounded lub's are set unions.*

2. If $\mathbf{M}$ is stable, then $(D(\mathbf{M}), \leq)$ is a dI-domain. For any upper-bounded set X of states of $\mathbf{M}$, the set intersection $\bigcap X$ is a state of $\mathbf{M}$, and hence is the glb of X in $D(\mathbf{M})$.

PROOF. We only check the last part of the statement. Let z be an upper bound of X, and $c \in F(\bigcap X)$. By stability, c has the same proof in all the elements of X, namely the proof of c in z. □

Example 14.1.6 *1. Flat cpo's. The flat cpo $X_\perp$ is generated by the following cds, which we denote by $X_\perp$ to avoid useless inflation of notation:*

$$X_\perp = (\{?\}, X, \{?\} \times X, \{\vdash ?\}).$$

2. The following cds **LAMBDA** $= (C, V, E, \vdash)$ *generates the (possibly infinite) terms of the untyped λ-calculus with constants, including Ω (cf. definition 2.3.1) (the typed case is similar):*

$$\begin{array}{l} C = \{0,1,2\}^\star \quad V = \{\cdot\} \cup \{x, \lambda x \mid x \in Var\} \cup Cons \quad E = C \times V \\ \vdash \epsilon \quad (u, \lambda x) \vdash u0 \quad (u, \cdot) \vdash u1, u2 \end{array}$$

where Var is the set of variables and Cons is the set of constants. For example, the term $t = (\lambda x.y)x$ is represented by $\{(\epsilon, \cdot), (1, \lambda x)), (10, y), (2, x)\}$. Here cells are occurrences, cf. definition 2.1.4.

Products of cds's are obtained by putting the component structures side by side, and by renaming the cells in each cds to avoid confusion.

Definition 14.1.7 *Let $\mathbf{M}$ and $\mathbf{M}'$ be two cds's. We define the product $\mathbf{M} \times \mathbf{M}' = (C, V, E, \vdash)$ of $\mathbf{M}$ and $\mathbf{M}'$ by:*

$$\begin{array}{l} C = \{c.1 \mid c \in C_{\mathbf{M}}\} \cup \{c'.2 \mid c' \in C_{\mathbf{M}'}\} \\ V = V_{\mathbf{M}} \cup V_{\mathbf{M}'} \\ E = \{(c.1, v) \mid (c, v) \in E_{\mathbf{M}}\} \cup \{(c'.2, v') \mid (c', v') \in E_{\mathbf{M}'}\} \\ (c_1.1, v_1) \vdash c.1 \Leftrightarrow (c_1, v_1) \vdash c \quad \textit{(and similarly for } \mathbf{M}') . \end{array}$$

Clearly, $\mathbf{M} \times \mathbf{M}'$ generates $D(\mathbf{M}) \times D(\mathbf{M}')$ (the ordered set product).

Definition 14.1.8 (sequential function (Kahn-Plotkin)) *Let* $\mathbf{M}$ *and* $\mathbf{M}'$ *be two cds's. A continuous function* $f : D(\mathbf{M}) \to D(\mathbf{M}')$ *is called sequential at* x *if for any* $c' \in A(f(x))$ *one of the following properties hold:*

(1) $\forall y \geq x \;\; c' \notin F(f(y))$
(2) $\exists c \in A(x) \;\; \forall y > x \;\; (f(x) <_{c'} f(y) \;\Rightarrow\; x <_c y)$

A cell c *satisfying condition (2) is called a* sequentiality index *of* f *at* (x, c'). *The index is called strict if (1) does not hold. If (1) holds, then any cell* c *in* $A(x)$ *is a (vacuous) sequentiality index. The function* f *is called sequential from* $\mathbf{M}$ *to* $\mathbf{M}'$ *if it is sequential at all points. We denote by* $\mathbf{M} \to_{seq} \mathbf{M}'$ *the set of these functions. A sequential function is called* strongly sequential *if, for any cell* c' *and any state* x *where it has a strict index, this index is unique.*

Examples of sequential functions are given in lemma 14.1.9 and in exercises 14.1.12, 14.1.14, and 2.4.4. The concrete data structures and the sequential functions form a cartesian category.

Lemma 14.1.9 *1. The identity functions, the first and second projection functions, and the constant functions are strongly sequential; the composition and the pairing of two sequential functions is sequential.*

2. If $\mathbf{M}$ *and* $\mathbf{M}'$ *are cds's and* $f : D(\mathbf{M}) \to D(\mathbf{M}')$ *is an order-isomorphism, then* f *is sequential.*

The sequential functions are stable, but not conversely. The counter-example given in the proof of the next proposition is due to Kleene and Berry, independently.

Proposition 14.1.10 *Let* $\mathbf{M}$ *and* $\mathbf{M}'$ *be two cds's. The following properties hold.*

1. Sequential functions from $\mathbf{M}$ *to* $\mathbf{M}'$ *are stable.*

2. If g *is sequential, if* f *is continuous, and if* $f \leq_{st} g$, *then* f *is sequential, and for any* x *and* $c' \in A(f(x))$, *if* f *has a strict index at* x, *then* $c' \in A(g(x))$, *and any index of* g *at* x *for* c' *is also an index of* f *at* x *for* c'.

3. There exist stable functions that are not sequential.

PROOF. (1) If $x \uparrow y$ and $g(x \wedge y) < g(x) \wedge g(y)$, then $g(x \wedge y) <_{c'} g(x) \wedge g(y)$, for some c'. Let c be a sequentiality index at $(x \wedge y, c')$. Then $x \wedge y <_c x$ and $x \wedge y <_c y$. Let v and w be such that $(c, v) \in x$ and $(c, w) \in y$. By proposition 14.1.5, $x \uparrow y$ implies $v = w$, which in turn implies $c \in F(x \wedge y)$ by stability. This contradicts $c \in A(x \wedge y)$. Hence g is stable.

(2) Let f be such that $f \leq_{st} g$, and let $c' \in A(f(x))$. If $c' \in F(g(x))$, then $c' \in A(f(y))$ for any $y \geq x$, since $F(f(x)) = F(f(y)) \cap F(g(x))$. If $c' \in A(g(x))$, $x \leq y$ and $f(x) <_{c'} f(y)$, then a fortiori $g(x) <_{c'} g(y)$, and $x <_c y$, where c is a

(strict) index of g at (x, c'). Hence f is sequential, and c is a strict index of f at (x, c').

(3) Let BK be the following stable function from $(\mathbf{B}_\perp)^3$ to $\mathbf{O}$:

$$BK(x,y) = \begin{cases} \top & \text{if } x = tt \text{ and } y = ff \\ \top & \text{if } x = ff \text{ and } z = tt \\ \top & \text{if } y = tt \text{ and } z = ff \\ \perp & \text{otherwise .} \end{cases}$$

One checks easily that BK has no index at $\perp$ for ?. □

Exercise 14.1.11 *Show that the restriction of any stable function to a principal ideal $\downarrow x$ is sequential.*

Exercise 14.1.12 *Let* $\mathbf{M}$ *and* $\mathbf{M}'$ *be two cds's and let (ϕ, ψ) be a stable injection-projection pair from $D(\mathbf{M})$ to $D(\mathbf{M}')$ (cf. definition 12.4.2). Show that ϕ and ψ are strongly sequential.*

Exercise 14.1.13 *1. We say that a cds* $\mathbf{M} = (C, V, E, \vdash)$ *is* included *in a cds* $\mathbf{M}' = (C', V', E', \vdash')$ *(and we write* $\mathbf{M} \subseteq \mathbf{M}'$*) if*

$$C \subseteq C' \quad V \subseteq V' \quad E \subseteq E' \quad \vdash \subseteq \vdash' .$$

Show that $(id, \lambda x'.x' \cap E)$ is a stable injection-projection pair from $D(\mathbf{M})$ to $D(\mathbf{M}')$.

2. Conversely, given D, D', each generated by some cds, and a stable injection-projection pair (ϕ, ψ) from D to D', show that there exist two cds's $\mathbf{M}$ *and* $\mathbf{M}'$ *such that*

$$D \cong D(\mathbf{M}) \quad D' \cong D(\mathbf{M}') \quad \mathbf{M} \subseteq \mathbf{M}' .$$

Exercise 14.1.14 *Define a cds* **BÖHM** *of Böhm trees, and show that theorem 2.4.3 reads as: BT is sequential from* **LAMBDA** *to* **BÖHM**.

The following exercise justifies the terminology of stable cds.

Exercise 14.1.15 *Let* $\mathbf{M}$ *be a cds. (1) Show that the functions $\underline{c} : D(\mathbf{M}) \to D(\mathbf{O})$ defined by $\underline{c}(x) = \top$ iff $c \in F(x)$ are linear (cf. definition 13.1.6). (2) Show that* $\mathbf{M}$ *is stable iff the functions $\underline{c}$ are stable. (3) Show that if* $\mathbf{M}$ *is stable and filiform, then it is sequential, by which we mean that the functions $\underline{c}$ are sequential.*

We now show that the category of cds's and sequential functions is not cartesian closed.

Theorem 14.1.16 *The category* **SEQ** *of cds's and sequential functions is not cartesian closed.*

PROOF. The following simple proof is due to Ehrhard [Ehr96]. (The original proof [Cur86] was similar to the proof of proposition 5.2.17). First we observe that if a category $\mathbf{C}$ has enough points, (cf. definition enough-points-CCC), then the products, projections and pairings are the set-theoretical ones. Also, we can take $\mathbf{C}[A, B]$ as the underlying set of the exponent $A \to B$, and the application and currying are the set-theoretical ones (due to the bijection between $\mathbf{C}[1, A \to B]$ and $\mathbf{C}[A, B]$). We assume that **SEQ** is cartesian closed. The proof by contradiction goes through successive claims.

1. For any $\mathbf{M}$ and $\mathbf{M}'$, the function $\lambda y.\bot : \mathbf{M} \to \mathbf{M}'$ is the minimum of $D(\mathbf{M} \to \mathbf{M}')$. To establish $\lambda y.\bot \leq f$ (f fixed and arbitrary), consider $g : \mathbf{O} \times D(\mathbf{M}) \to D(\mathbf{M}')$, defined by:

$$g(x, y) = \begin{cases} \bot & \text{if } x = \bot \\ f(y) & \text{if } x \neq \bot \, . \end{cases}$$

The function g is sequential, and therefore is a morphism of **Seq**. Hence we can consider $\Lambda(g)$, which is a fortiori monotonic:

$$\lambda y.\bot = \Lambda(g)(\bot) \leq \Lambda(g)(\top) = f.$$

2. For any $\mathbf{M}$, there exists an (initial) cell c in $\mathbf{M} \to \mathbf{O}$ such that

$$\forall f \in D(\mathbf{M} \to \mathbf{O}) \;\; (f \neq \lambda y.\bot \Rightarrow c \in F(f)).$$

Indeed, the set-theoretical application, being the evaluation morphism, is sequential. It is non-strict in its second argument ($ev(f, \bot) = f(\bot) \neq \bot$ for, say, any constant function different from $\lambda y.\bot$). Hence ev has a sequentiality index of the form $c.1$ at $((\bot, \bot), ?)$. Let then $f \in D(\mathbf{M} \to \mathbf{O})$ be such that $f \neq \lambda y.\bot$, i.e., $? \in F(ev(f, z)) = F(f(z))$ for some z. By sequentiality we get $c \in F(f)$.

3. Finally, consider the following form h of the conditional function from $\mathbf{O}^2 \times \mathbf{B}_\bot$ to $\mathbf{O}$:

$$h((x, y), z) = \begin{cases} \bot & \text{if } z = \bot \\ x & \text{if } z = tt \\ y & \text{if } z = ff \, . \end{cases}$$

Then we have

$$\Lambda(h)(\bot, \bot) = \lambda z.\bot \quad \Lambda(h)(\top, \bot) \neq \lambda z.\bot \quad \Lambda(h)(\bot, \top) \neq \lambda z.\bot$$

from which we derive:

$$\begin{array}{ll} c \notin F(\Lambda(h)(\bot, \bot)) & \text{by claim 1} \\ c \in F(\Lambda(h)(\top, \bot)) \text{ and } c \in F(\Lambda(h)(\bot, \top)) & \text{by claim 2} \, . \end{array}$$

But this contradicts the sequentiality of $\Lambda(h)$. □

14.2 Sequential Algorithms

A sequential function having at a given point more than one sequentiality index may be computed in different ways according to the order in which these indices are explored. For example the addition function on $\omega_\perp \times \omega_\perp$ has ?.1 and ?.2 as sequentiality indices at $\perp$. The left addition computes ?.1, then ?.2, whereas the right addition does the same computations in the inverse order. The sequential algorithms formalise these ideas. For all cds's $\mathbf{M}$ and $\mathbf{M}'$, we define an exponent cds $\mathbf{M} \to \mathbf{M}'$, whose states are called the sequential algorithms from $\mathbf{M}$ to $\mathbf{M}'$. We give an abstract characterisation of a sequential algorithm by a function describing both its input-output behaviour and its computation strategy. The characterisation serves to define the composition of sequential algorithms.

Definition 14.2.1 (exponent cds) *If* $\mathbf{M}$ *and* $\mathbf{M}'$ *are two cds's, the cds* $\mathbf{M} \to \mathbf{M}'$ *is defined as follows:*

- *If* x *is a finite state of* $\mathbf{M}$*, and if* c' *is a cell of* $\mathbf{M}'$*, then* xc' *is a cell of* $\mathbf{M} \to \mathbf{M}'$*.*
- *The values and the events are of two types, called "valof" and "output", respectively:*
  - *If* c *is a cell of* $\mathbf{M}$*, then* $valof\, c$ *is a value of* $\mathbf{M} \to \mathbf{M}'$*, and* $(xc', valof\, c)$ *is an event of* $\mathbf{M} \to \mathbf{M}'$ *iff* c *is accessible from* x*;*
  - *if* v' *is a value of* $\mathbf{M}'$*, then* $output\, v'$ *is a value of* $\mathbf{M} \to \mathbf{M}'$*, and* $(xc', output\, v')$ *is an event of* $\mathbf{M} \to \mathbf{M}'$ *iff* (c', v') *is an event of* $\mathbf{M}'$*.*
- *The enablings are also of two types:*

$$\begin{array}{lll} (yc', valof\ c) \vdash xc' & \text{iff} & y \prec_c x \qquad (\text{"valof"}) \\ (x_1c'_1, output\ v'_1) \vdash xc' & \text{iff} & x = x_1 \text{ and } (c'_1, v'_1) \vdash c' \qquad (\text{"output"}) \,. \end{array}$$

A state of $\mathbf{M} \to \mathbf{M}'$ *is called a* sequential algorithm*, or simply an algorithm. If* a *and* x *are states of* $\mathbf{M} \to \mathbf{M}'$ *and* $\mathbf{M}$*, respectively, we write*

$$a \bullet x = \{(c', v') \mid \exists\, y \leq x \;\; (yc', output\, v') \in a\}.$$

The function $\lambda x.(a \bullet x)$ *is called the input-output function computed by* a*.*

Example 14.2.2 *1. The left addition algorithm* $ADD_l : \omega_\perp \times \omega_\perp \to \omega_\perp$ *consists of the following events:*

$$\begin{array}{ll} (\emptyset?, valof\ ?.1) & \\ (\{(?.1, i)\}?, valof\ ?.2) & (i \in \omega) \\ (\{(?.1, i), (?.2, j)\}?, output\ i + j) & (i, j \in \omega) \,. \end{array}$$

$$
\begin{array}{lcll}
OR_{sl} & = & (\emptyset?, valof\ ?.1) & \\
 & & (\{(?.1, i)\}?, valof\ ?.2) & (i \in \mathbf{B}) \\
 & & (\{(?.1, i), (?.2, j)\}?, output\ OR(i,j)) & (i,j \in \mathbf{B}) \\
 & & & \\
OR_{sr} & = & (\emptyset?, valof\ ?.2) & \\
 & & (\{(?.2, i)\}?, valof\ ?.1) & (i \in \mathbf{B}) \\
 & & (\{(?.2, i), (?.1, j)\}?, output\ OR(j,i)) & (i,j \in \mathbf{B}) \\
 & & & \\
OR_{l} & = & (\emptyset?, valof\ ?.1) & \\
 & & (\{(?.1, tt)\}?, output\ tt) & \\
 & & (\{(?.1, ff)\}?, valof\ ?.2) & \\
 & & (\{(?.1, ff), (?.2, j)\}?, output\ j) & (j \in \mathbf{B}) \\
 & & & \\
OR_{r} & = & (\emptyset?, valof\ ?.2) & \\
 & & (\{(?.2, tt)\}?, output\ tt) & \\
 & & (\{(?.2, ff)\}?, valof\ ?.1) & \\
 & & (\{(?.2, ff), (?.1, j)\}?, output\ j) & (j \in \mathbf{B})
\end{array}
$$

Figure 14.1: The four disjunction algorithms

The right addition algorithm ADD_r is defined similarly.

2. There are four different disjunction algorithms from $\mathbf{B}_\perp \times \mathbf{B}_\perp$ to $\mathbf{B}_\perp$. The two algorithms (OR_{sl} and OR_{sr}) compute the disjunction function that is strict in both its arguments; They are similar to ADD_l and ADD_r. The two algorithms OR_l and OR_r compute the left and right addition functions that are strict in one of their arguments only, respectively. The four algorithms are described in figure 14.1. In this figure, OR is the usual interpretation of disjunction over $\mathbf{B} = \{tt, ff\}$.

Lemma 14.2.3 *Let $\mathbf{M}$ and $\mathbf{M}'$ be two cds's. If $\mathbf{M}'$ is well founded (filiform), then $\mathbf{M} \to \mathbf{M}'$ is well founded (filiform).*

PROOF. We observe that if $xc' \ll yd'$, then $x < y$ (with y finite) or $c' \ll d'$ □

The stability condition is essential to ensure that $\lambda x.(a \bullet x)$ is a function from $D(\mathbf{M})$ to $D(\mathbf{M}')$. The following example shows that this is not true in general.

Let

$$\begin{array}{rcl} \mathbf{M} & = & (\{c\},\{v\},\{(c,v)\},\vdash) \\ & & \text{with } \vdash c \\ \mathbf{M}' & = & (\{c'_1,c'_2,c'_3\},\{1,2\},\{(c'_i,j) \mid 1 \leq i \leq 3 \text{ and } 1 \leq j \leq 2\},\vdash) \\ & & \text{with } \vdash c'_1 \quad \vdash c'_2 \quad (c'_1,1) \vdash c'_3 \quad (c'_2,1) \vdash c'_3 \end{array}$$

where $\mathbf{M}'$ is not stable, since c'_3 has two enablings in $\{(c'_1,1),(c'_2,1)\}$. We choose a and x as follows:

$$\begin{array}{rcl} a & = & \{(\perp c'_1, \mathit{output}\ 1),(\perp c'_2, \mathit{valof}\ c),(\{(c,v)\}c'_2, \mathit{output}\ 1), \\ & & (\perp c'_3, \mathit{output}\ 1),(\{(c,v)\}c'_3, \mathit{output}\ 2)\} \\ x & = & \{c,v\}\ . \end{array}$$

Then a and x are states of $\mathbf{M} \to \mathbf{M}'$ and $\mathbf{M}$, respectively, but $a{\bullet}x$ is not a state of $\mathbf{M}'$, since it contains both $(c'_3,1)$ and $(c'_3,2)$.

The following is a key technical proposition.

Proposition 14.2.4 *Let* $\mathbf{M}$ *and* $\mathbf{M}'$ *be cds's, and let* a *be a state of* $\mathbf{M} \to \mathbf{M}'$. *The following properties hold:*

1. If $(xc',u),(zc',w) \in a$ *and* $x \uparrow z$, *then* $x \leq z$ *or* $z \leq x$; *if* $x < z$, *there exists a chain*

$$x = y_0 \prec_{c_0} y_1 \cdots y_{n-1} \prec_{c_{n-1}} y_n = z$$

such that $\forall\, i < n\ \ (y_i c', \mathit{valof}\ c_i) \in a$. *If* u *and* w *are of type "output", then* $x = z$.

2. The set $a{\bullet}x$ *is a state of* $\mathbf{M}'$, *for all* $x \in D(\mathbf{M})$.

3. For all $xc' \in F(a)$, xc' *has only one enabling in* a; *hence* $\mathbf{M} \to \mathbf{M}'$ *is stable.*

4. The function $\lambda x.(a{\bullet}x)$ *is stable.*

PROOF. We prove (1), (2) and (3) together, by induction on c'. At each induction step we prove (1), (2') and (3), where (2') is the following property:

2'. The set $(a{\bullet}x)_{c'} = \{(d',v') \mid \exists\, y \leq x\ \ (yd', \mathit{output}\ v') \in a \text{ and } d' \ll^+ c'\}$ is a state, for all $x \in D(\mathbf{M})$.

Property (2) is indeed a consequence of (1) and (2'): by (2'), the set $a{\bullet}x$ is safe, and, by (1), it is consistent.

(2') Let x be a state of $\mathbf{M}$, and let $(d',v') \in (a{\bullet}x)_{c'}$. Then, by definition, $\exists\, y \leq x\ \ yd' \in F(a)$. By analyzing a proof of yd' in a, we check easily that d' is enabled in $(a{\bullet}x)_{c'}$. Suppose $(d',v'_1),(d',v'_2) \in (a{\bullet}x)_{c'}$. Then

$$\exists\, y_1, y_2 \leq x\ \ (y_1 d', \mathit{output}\ v'_1),(y_2 d', \mathit{output}\ v'_2) \in a$$

whence we derive $y_1 = y_2$ by induction hypothesis (1), and $v'_1 = v'_2$ by consistency of a. Hence $(a{\bullet}x)_{c'}$ is a state.

(1) We first remark that the last assertion of (1) follows from the others, since if $x \neq z$, then the existence of a chain between x and y as described in the statement entails that u or w is of type "valof". Let s and t be two proofs of xc' and zc' in a, respectively; here is the detail of s until the first enabling of type "output" is met:

$$(xc', u), (x_{k-1}c', \mathit{valof}\ c_{k-1}), \ldots, (x_1c', \mathit{valof}\ c_1), (x_0c', u_0), (x_0c'^1, \mathit{output}\ v'^1).$$

The following properties hold, by definition of the enablings of $\mathbf{M} \to \mathbf{M}'$:

- If $k = 0$, then $u_0 = u$.

  If $k > 0$, then $\exists\, c_0\ \ u_0 = \mathit{valof}\ c_0$, and $\forall\, i < k\ \ x_i \prec_{c_i} x_{i+1}$ (we write $x = x_k$).

- $(c'^1, v'^1) \vdash c'$.

Similar properties hold in t, replacing $x, x_0, \ldots, x_k, x^1, u_0, c_0, \ldots, c_{k-1}, c'^1, v'^1$ by

$$z, z_0, \ldots, z_m, w_0, d_0, \ldots, d_{m-1}, c'^2, v'^2.$$

First we prove that $x_0 = z_0$. Let y be such that $x, z \leq y$. The set $(a \bullet y)_{c'}$, which is a state by induction hypothesis (2′), contains (c'^1, v'^1) and (c'^2, v'^2), which are two enablings of c'. We have:

$c'^1 = c'^2$ (since $\mathbf{M}'$ is stable),
$x_0 = z_0$ (by the induction hypothesis (1)).

Property (1) clearly holds if $m = 0$ or $k = 0$. Hence we may suppose $k, m \neq 0$ and $k \leq m$ (by symmetry). We show by induction on i that $x_i = z_i$ if $i \leq k$. Using the induction hypothesis we may rewrite $x_{i-1} \prec_{c_{i-1}} x_i$ as $z_{i-1} \prec_{d_{i-1}} x_i$ (note that $c_{i-1} = d_{i-1}$ by consistency of a). As we also have $z_{i-1} \prec_{d_{i-1}} z_i$ and $x_i \uparrow z_i$, we derive $x_i = z_i$. If $k < m$, the chain $x = z_k \prec_{d_k} z_{k+1} \prec \cdots \prec_{d_{m-1}} z$ has the property stated in (1). If $k = m$, then $x = z$.

(3) We exploit the proof of (1): if we start with the assumption that $x = z$, then the part of the proof s which we have displayed coincides with the corresponding part of the proof t; in particular, xc' has the same enabling in both proofs. Hence xc' has only one enabling in a.

(4) Finally, we prove that $\lambda x.(a \bullet x)$ is stable. We first check continuity. Let X be directed, let $(c', v') \in a \bullet (\bigvee X)$, and let $x \leq \bigvee X$ be such that $(xc', \mathit{output}\ v') \in a$. Since x is finite, $\exists\, y \in X\ \ x \leq y$. Hence $(c', v') \in a \bullet y$. As for the stability, let $x \uparrow y$, and let $c' \in F(a \bullet x) \cap F(a \bullet y)$. Then, by (1) there exists $z \leq x, y$ and v' such that $(zc', \mathit{output}\ v') \in a$; hence $(c', v') \in a \bullet (x \wedge y)$. □

Now we present an abstract characterisation of sequential algorithms. Intuitively, if a sequential algorithm a contains (xc', u), an information on the computation of a at x is given: if $u = \mathit{output}\ v'$, then $a \bullet x$ contains (c', v'); if $u = \mathit{valof}\ c$,

then the contents of c must be computed in order to fill c' (we show in proposition 14.2.9 that the function computed by a is indeed sequential). These informations remain true at $y > x$, supposing in the second case that c is still not filled in y. As yc' is not filled in a, the "indications" given by a are not limited to cells filled in a. For example ADD_l "indicates" *valof* ?.1 at $\{(?.2, 0)\}$ as well as at $\perp$. The following definition and proposition formalise these ideas.

Definition 14.2.5 (abstract algorithm) *Let* $\mathbf{M}$ *and* $\mathbf{M}'$ *be cds's. An abstract algorithm from* $\mathbf{M}$ *to* $\mathbf{M}'$ *is a partial function* $f : C_{\mathbf{M}\to\mathbf{M}'} \rightharpoonup V_{\mathbf{M}\to\mathbf{M}'}$ *satisfying the following axioms, for any states* x *and cells* c'*:*

(A1) If $f(xc') = u$*, then* $(xc', u) \in E_{\mathbf{M}\to\mathbf{M}'}$.

(A2) If $f(xc') = u$*,* $x \leq y$ *and* $(yc', u) \in E_{\mathbf{M}\to\mathbf{M}'}$*, then* $f(yc') = u$.

(A3) Let $f{\bullet}y = \{(c', v') \mid f(yc') = \mathit{output}\ v'\}$*. Then*

$$f(yc')\downarrow \Rightarrow (c' \in E(f{\bullet}y) \text{ and } (z \leq y \text{ and } c' \in E(f{\bullet}z) \Rightarrow f(zc')\downarrow)).$$

We write $f(xc') = \omega$ *if* f *is not defined at* xc'*. When writing* $f(xc') = u$*, we suppose* $u \neq \omega$*. An easy consequence of (A3) is that* $f{\bullet}z$ *is a state. We denote by* $(\mathcal{A}(\mathbf{M}, \mathbf{M}'), \leq)$ *the set of abstract algorithms from* $\mathbf{M}$ *to* $\mathbf{M}'$ *ordered as follows:*

$$f \leq f' \quad \textit{iff} \quad (f(xc') = u \Rightarrow f'(xc') = u).$$

It will be convenient (when defining the composition of algorithms) to extend an abstract algorithm f to a partial function from $D(\mathbf{M}) \times \mathbf{C}'$ to $V_{\mathbf{M}\to\mathbf{M}'}$. We keep the same name f for the extended function:

$$f(xc') = u \text{ iff } \begin{cases} f(yc') = u \text{ for some finite } y \leq x \text{ and} \\ \text{either } (u = \mathit{valof}\ c \text{ and } c \in A(x)), \text{ or } u = \mathit{output}\ v' \text{ .} \end{cases}$$

Exercise 14.2.6 *Show that an abstract algorithm between two cds's* $\mathbf{M}$ *and* $\mathbf{M}'$ *may be axiomatised as a partial function from* $D(\mathbf{M}) \times \mathbf{C}'$ *to* $V_{\mathbf{M}\to\mathbf{M}'}$ *which satisfies the following axiom in addition to the axioms (A1), (A2), and (A3):*

(A0) If $f(xc') = u$*, then* $f(yc') = u$*, for some finite* $y \leq x$.

The abstract algorithms may be viewed as pairs $(\lambda x.(f{\bullet}x), i)$ where i, which may be called a computation strategy, is the function defined by "restricting" f to its control aspects, that is, $i(xc') = c$ iff $f(xc') = \mathit{valof}\ c$.

Exercise 14.2.7 *Show that an abstract algorithm from a cds* $\mathbf{M}$ *to a cds* $\mathbf{M}'$ *may equivalently be defined as a pair of a sequential function* f *from* $\mathbf{M}$ *to* $\mathbf{M}'$*, and of a computation strategy* i *for it, which is a partial function* $i : D(\mathbf{M}) \times \mathbf{C}' \rightharpoonup \mathbf{C}$ *that satisfies the following axioms:*

1. *If $i(xc') = c$, then $c \in A(x)$ and $c' \in A(f(x))$.*
2. *If $i(xc') = c$, then $i(yc') = c$ for some finite $y \leq x$.*
3. *If $c' \in A(f(x))$ and $c' \in F(f(y))$ for some $y \geq x$, then $i(xc')$ is defined and is a sequentiality index for f at (x, c').*
4. *If $i(xc') = c$, if $x \leq y$ and $c \in A(y)$, then $i(yc') = c$.*
5. *If $i(xc') = c$ and $y \leq x$ is such that $c' \in A(f(y))$, then $i(yc')$ is defined.*

The next theorem relates the abstract algorithms with the states of the exponent cds's.

Proposition 14.2.8 *Let $\mathbf{M}$ and $\mathbf{M}'$ be cds's. Let a be a state of $\mathbf{M} \to \mathbf{M}'$. Let $a^+ : C_{\mathbf{M}\to\mathbf{M}'} \rightharpoonup V_{\mathbf{M}\to\mathbf{M}'}$ be given by*

$$a^+(xc') = u \quad \textit{iff} \quad \exists\, y \leq x \;\; (yc', u) \in a \textit{ and } (xc', u) \in E_{\mathbf{M}\to\mathbf{M}'}.$$

Let $f \in \mathcal{A}(\mathbf{M}, \mathbf{M}')$. We set

$$f^- = \{(xc', u) \mid f(xc') = u \textit{ and } (y < x \Rightarrow f(yc') \neq u)\}.$$

The following properties hold:

1. *For all $a \in D(\mathbf{M} \to \mathbf{M}')$, a^+ is an abstract algorithm from $\mathbf{M}$ to $\mathbf{M}'$.*
2. *For all $f \in \mathcal{A}(\mathbf{M}, \mathbf{M}')$, f^- is a state of $\mathbf{M} \to \mathbf{M}'$.*
3. *$(\_)^+$ is an isomorphism from $(D(\mathbf{M} \to \mathbf{M}'), \leq)$ onto $(\mathcal{A}(\mathbf{M}, \mathbf{M}'), \leq)$, and has $(\_)^-$ as inverse; if $f, f' \in \mathcal{A}(\mathbf{M}, \mathbf{M}')$ and $f \leq f'$, then $(\lambda x.(f \bullet x)) \leq_{st} (\lambda x.(f' \bullet x))$.*

PROOF. (1) Let a be a state of $\mathbf{M} \to \mathbf{M}'$. Clearly, a^+ satisfies (A1) and (A2) by definition. Suppose $a^+(yc') = u$; then $\exists\, x \leq y \;\; (xc', u) \in a$. Let s be a proof of xc' in a, with the notation of the proof of proposition 14.2.4. Since $(c'^1, v'^1) \in a \bullet x$, we have $c' \in E(a \bullet x)$. Clearly, $a \bullet x = a^+ \bullet x \leq a^+ \bullet y$ (if $(xc', \textit{output } v')$ is an event of $\mathbf{M} \to \mathbf{M}'$, so is $(yc', \textit{output } v')$); hence $c' \in E(a^+ \bullet y)$. Suppose $z \leq y$ and $c' \in E(a^+ \bullet z)$. As $a^+ \bullet z = a \bullet z \uparrow a \bullet x$, c' has the same enabling in $a \bullet x$ and $a \bullet z$. So, by definition of $a \bullet z$:

$$\exists\, z_0 \leq z \;\; (z_0 c'^1, \textit{output } v'^1) \in a.$$

By proposition 14.2.4, we get $z_0 = x_0$, whence we derive $x_0 \leq z$. Let i be maximum such that $x_i \leq z$. We prove $a^+(zc') = a^+(x_i c')$, hence a fortiori $a^+(zc') \neq \omega$ (we have $a^+(x_i c') \neq \omega$, since $x_i c' \in F(a)$). The case where $i = k$ and u is of type "output" is trivial, since then, as above, (zc', u) is an event of $\mathbf{M} \to \mathbf{M}'$. We prove that $(zc', \textit{valof } c_i)$ is an event, that is, $c_i \in A(z)$ (if $i = k$ and u is of type "valof" we write $u = \textit{valof } c_k$). First, $c_i \in E(z)$, since $c_i \in A(x_i) \subseteq E(x_i) \subseteq E(z)$. Suppose $c_i \in F(z)$. We distinguish two cases:

- $i < k$: Then $x_{i+1} \leq z$, since $x_i \prec_{c_i} x_{i+1}$ and $x_{i+1} \uparrow z$. This contradicts the maximality of i.

- $i = k$: Then $a^+(yc') = \mathit{valof}\ c_k$ implies $(yc', \mathit{valof}\ c_k) \in E_{\mathbf{M}\to\mathbf{M}'}$, which implies $c_k \in A(y)$. This contradicts $c_k \in F(z) \subseteq F(y)$.

(2) Let f be an abstract algorithm from $\mathbf{M}$ to $\mathbf{M}'$. We prove that f^- is a state of $\mathbf{M} \to \mathbf{M}'$. It is consistent by definition, since $(xc', u) \in f^- \Rightarrow f(xc') = u$. We prove by induction on c' that

$$f^-\backslash c' = \{(yd', w) \in f^- \mid d' \ll^\star c'\}$$

is safe, which will imply the safety of f^-. If $(xc', u) \in f^-$, then by (A3) $f\bullet x$ contains an enabling (c'^1, v'^1) of c'. Let $x_0 \leq x$ be minimal such that $f(x_0c'^1) = \mathit{output}\ v'^1$, that is, $(x_0c'^1, \mathit{output}\ v'^1) \in f^-$. We construct a chain

$$x_0 \prec_{c_0} x_1 \prec \cdots x_{k-1} \prec_{c_{k-1}} x_k = x$$

as follows. Suppose that we have built the chain up to i, with $x_i < x$. Then we define c_i by $f(x_ic') = \mathit{valof}\ c_i$ ($f(x_ic') \neq \omega$ by (A3), and is not of type "output" by minimality of x). Then $x_i <_{c_i} x$, since $f(xc') = f(x_ic')$ would again contradict the minimality of x; we choose x_{i+1} characterised by $x_i \prec_{c_i} x_{i+1} \leq x$. We show by induction:

$$\forall\, i < k\ \ (x_ic', \mathit{valof}\ c_i) \in f^-$$

which together with the induction hypothesis will establish the safety of $f^-\backslash c'$. Suppose that there exists $z < x_i$ such that $f(zc') = \mathit{valof}\ c_i$. By (A3) again, f^- would contain $(z_0c'^2, \mathit{output}\ v'^2)$ such that $(c'^2, v'^2) \vdash c'$. By induction, we may suppose that $a = f^-\backslash c'^1 \cup f^-\backslash c'^2$ is safe; it is actually a state (consistency follows from $a \subseteq f^-$). Hence $a\bullet x$ is a state by proposition 14.2.4, whence we derive $c'^1 = c'^2$ by stability, and, $x_0 = z_0$ by proposition 14.2.4, which implies $x_0 \leq z$. Let j be maximum $\leq i$ such that $x_j \leq z$. Then $j < i$, hence $f(zc') = \mathit{valof}\ c_j$ by maximality of j, contradicting $f(zc') = f(x_ic')$, since $c_j \in F(x_i)$. This shows $(x_ic', \mathit{valof}\ c_i) \in f^-$, and ends the proof of (2).

(3) Let $a, a' \in \mathbf{M} \to \mathbf{M}'$. Clearly $(a^+)^- \subseteq a$. Reciprocally, if $(xc', u), (zc', w) \in a$ and $x < z$, we have $u \neq w$, since u is $\mathit{valof}\ c$ for some $c \in F(z)$. It follows easily that $a \subseteq (a^+)^-$. If $a \leq a'$, then $a^+ \leq a'^+$ is an immediate consequence of the definition of $(\_)^+$. Let $f, f' \in \mathcal{A}(\mathbf{M}, \mathbf{M}')$. One checks easily $f \subseteq (f^-)^+$ by (A1), and $(f^-)^+ \subseteq f$ by (A2). Let $f \leq f'$. We first prove the last assertion of (3). If $x, y \in D(\mathbf{M})$ and $y \leq x$, we have to prove $(f\bullet x) \wedge (f'\bullet y) \subseteq f\bullet y$, that is, for any c':

$$(f(xc') = \mathit{output}\ v' \text{ and } f'(yc') = \mathit{output}\ v') \Rightarrow f(yc') = \mathit{output}\ v'.$$

We proceed by induction on c'. Since $f \leq f'$, we only have to prove $f(yc') \neq \omega$, and hence to show $c' \in E(f\bullet y)$. As $f\bullet x \leq f'\bullet x$ and $f'\bullet y \leq f'\bullet x$, c' has the same enabling (c'^1, v'^1) in $f\bullet x$ and $f'\bullet y$. Hence $f(xc'^1) = f'(yc'^1) = \mathit{output}\ v'^1$, whence we derive by induction $f(yc'^1) = \mathit{output}\ v'^1$, proving $c' \in E(f\bullet y)$.

Finally, we prove $f^- \leq f'^-$. Suppose $(xc', u) \in f^-$. Then $f'(xc') = f(xc') = u$. Suppose $f'(yc') = u$ for some $y < x$. Then $c' \in E(f'\bullet y)$. As we also have $c' \in E(f\bullet x)$, we obtain $c' \in E(f\bullet y)$ by what has just been proved. Hence $f(yc') \neq \omega$, implying $f(yc') = f'(yc') = u$ and contradicting the minimality of x. □

We now relate abstract algorithms to sequential functions.

Proposition 14.2.9 *Let* $\mathbf{M}$ *and* $\mathbf{M}'$ *be cds's. If* $a, a' \in D(\mathbf{M} \to \mathbf{M}')$, *we write*

$$a =_{ext} a' \quad \textit{iff} \quad (\forall\, x \in D(\mathbf{M})\ \ a\bullet x = a'\bullet x).$$

The partial orders $(D(\mathbf{M} \to \mathbf{M}')/=_{ext}, \leq\ /=_{ext})$ *and* $(\mathbf{M} \to_{seq} \mathbf{M}', \leq_{st})$ *are isomorphic; in particular, for any* $a \in D(\mathbf{M} \to \mathbf{M}')$, $\lambda x.(a\bullet x)$ *is sequential.*

Proof. First we prove that if $a \in D(\mathbf{M} \to \mathbf{M}')$, then $\lambda x.(a\bullet x)$ is sequential. If $c' \in A(a\bullet x)$ and if there exist $y > x$ and v' such that $(c', v') \in a\bullet y$, then $a^+(yc') = \textit{output}\ v'$, and by (A3) $a^+(xc') = u \neq \omega$. Specifically, u has the form $\textit{valof}\ c$, since $c' \in A(a\bullet x)$, and $c \in F(y)$ by (A2), and $c \in A(x)$ by (A1). Hence $\lambda x.(a\bullet x)$ is sequential and has c as index at x for c'. By proposition 14.2.8:

$$\forall\, a \leq a' \in D(\mathbf{M} \to \mathbf{M}')\ \ (\lambda x.(a\bullet x) = a^+\bullet x) \leq_{st} (\lambda x.(a'\bullet x) = a'^+\bullet x).$$

So we only have to prove, for all $g, g' \in \mathbf{M} \to_{seq} \mathbf{M}'$ such that $g \leq_{st} g'$:

$$\exists\, a, a' \in D(\mathbf{M} \to \mathbf{M}')\ \ (g = (\lambda x.(a\bullet x)), g' = (\lambda x.(a'\bullet x)) \text{ and } a \leq a').$$

We build a and a' progressively. For any cell c', we define the sets $(X^n_{g',c'})_{n\geq 0}$ and a function $V_{g',c'}$ as follows, by induction on n:

- $X^0_{g',c'} = \{x \in m_e(g', c') \mid \exists\, z \geq x\ \ c' \in F(g'(z))\}$

  where $m_e(g', c')$ is the set of the minimal x's such that $c' \in E(g'(x))$ (the elements of $X^0_{g',c'}$ are finitely many and incompatible).

- For all $x \in X^n_{g',c'}$:

  - $V_{g',c'}(x) = \textit{valof}\ c$ if $c' \in A(g'(x))$ and if c is an arbitrarily chosen sequentiality index of g' at (x, c');
  - $V_{g',c'}(x) = \textit{output}\ v'$ if $(c', v') \in g'(x)$.

- $X^{n+1}_{g',c'}$ is the smallest set such that, for all $x \in X^n_{g',c'}, y \in D(\mathbf{M})$:

$$(V_{g',c'}(x) = \textit{valof}\ c, x \prec_c y, (\exists\, z \geq y\ \ c' \in F(g'(z)))) \Rightarrow y \in X^{n+1}_{g',c'}.$$

The definition of $V_{g',c'}$ is unambiguous, since $X^n_{g',c'} \cap X^m_{g',c'} = \emptyset$ $(n \neq m)$ follows easily from

$$\forall\, x, x' \in X^0_{g',c'}\ \ x \neq x' \Rightarrow x \not\uparrow x'.$$

So $V_{g',c'}$ is well defined. Let $X_{g',c'} = \bigcup\{X^n_{g',c'} \mid n \geq 0\}$. We define likewise $X_{g,c'}$, $V_{g,c'}$ such that $X_{g',c'}$ contains $X_{g,c'}$ and $V_{g,c'}$ is the restriction of $V_{g',c'}$ to $X_{g,c'}$ (this may be done by proposition 14.1.10). Let

$$a' = \bigcup\{(xc', V_{g',c'}(x)) \mid x \in X_{g',c'}, c' \in C'\}.$$

We define likewise a. By construction, a and a' are consistent, and $a \leq a'$. We check that a' is safe. This is clear by construction for an event $(xc', V_{g',c'}(x))$ where $x \in X^n_{g',c'}$ and $n > 0$. If $n = 0$, then by construction $x \in m_e(g', c')$, hence x is minimal such that $d' \in F(g'(x))$, for some $d' \ll c'$. Then safety follows from the fact that by construction $X_{g',c'}$ contains all minimal points z such that $d' \in g'(z)$, for all $d' \in C'$. Finally, it is evident by construction that $(c', v') \in g'(x)$ iff $(c', v') \in a'\bullet x$. The same arguments can be applied to a. □

Exercise 14.2.10 *Let* $\mathbf{M}$ *and* $\mathbf{M}'$ *be cds's. Show that a function* $f : D(\mathbf{M}) \to D(\mathbf{M}')$ *is sequential if and only if it is continuous and sequential at any compact point. Hint: Use proposition 14.2.9.*

Exercise 14.2.11 *Let* $\mathbf{M}$ *and* $\mathbf{M}'$ *be cds's, and let* f *be a strongly sequential function. Show that there exists a minimum algorithm* a *such that* $f = \lambda x.(a\bullet x)$.

Exercise 14.2.12 *Let* $\mathbf{M}$ *and* $\mathbf{M}'$ *be two (well-founded and stable) sequential cds's. Show that* $\mathbf{M} \times \mathbf{M}'$ *and* $\mathbf{M} \to \mathbf{M}'$ *are sequential (cf. exercise 14.1.15).*

We next define the composition of sequential algorithms, using their abstract characterisation. We first discuss the composition of sequential algorithms informally. If a and a' are algorithms from $\mathbf{M}$ to $\mathbf{M}'$ and from $\mathbf{M}'$ to $\mathbf{M}''$, respectively, then the input-output function of $a' \circ a$ should be the composition of the input-output functions of a and a', that is, for any state x of $\mathbf{M}$:

$$(a' \circ a)\bullet x = a'\bullet(a\bullet x).$$

How can this equation help in the characterisation of the events of $a' \circ a$? By definition of the operator $\bullet$ we obtain

$$\exists\, z \leq x\ \ (zc'', \mathit{output}\ v'') \in a' \circ a \Leftrightarrow \exists\, z' \leq a\bullet x\ \ (z'c'', \mathit{output}\ v'') \in a'.$$

This equivalence allows us to describe events that are "almost" in $a' \circ a$. Using the notation of proposition 14.2.8, we get

$$(a' \circ a)^+(xc'') = \mathit{output}\ v'' \ \text{ iff } \ a'^+((a\bullet x)c'') = \mathit{output}\ v''.$$

The equation does not characterise events belonging to $a' \circ a$, but events where $(a' \circ a)^+$ is defined. Hence it seems natural to define the composition of sequential algorithms using their abstract characterisation.

What about the computation strategy of $a' \circ a$? The definition of sequential functions suggests an output-directed computation: "in order to compute c', the index c has to be computed". Hence it is natural to compose these strategies: if a' indicates *valof* c' at $a \bullet x$ for c'', and if a indicates *valof* c at x for c', then $a' \circ a$ indicates *valof* c at x for c'', which is summarised by the following equivalence:

$$(a' \circ a)^+(xc'') = \textit{valof } c \text{ iff } \begin{cases} a'^+((a \bullet x)c'') = \textit{valof } c' \text{ and} \\ a^+(xc') = \textit{valof } c\,. \end{cases}$$

The next proposition shows that these equivalences indeed define an abstract algorithm.

Proposition 14.2.13 *Let* $\mathbf{M}$, $\mathbf{M}'$ *and* $\mathbf{M}''$ *be cds's, and let* a *and* a' *be two states of* $\mathbf{M} \to \mathbf{M}'$ *and* $\mathbf{M}' \to \mathbf{M}''$, *respectively. The function* $f : C_{\mathbf{M} \to \mathbf{M}''} \rightharpoonup V_{\mathbf{M} \to \mathbf{M}''}$, *defined as follows, is an abstract algorithm from* $\mathbf{M}$ *to* $\mathbf{M}''$*:*

$$f(xc'') = \begin{cases} \textit{output } v'' \textit{ if } a'^+((a \bullet x)c'') = \textit{output } v'' \\ \\ \textit{valof } c \textit{ if } \begin{cases} a'^+((a \bullet x)c'') = \textit{valof } c' \textit{ and} \\ a^+(xc') = \textit{valof } c\,. \end{cases} \end{cases}$$

PROOF. $(A1), (A2)$ If $f(xc'') = \textit{output } v''$ and $x \leq y$, then $(xc'', \textit{output } v'')$ is an event, since it follows from the definition of a'^+ that $(c'', v'') \in E_{\mathbf{M}''}$, and

$$a \bullet x \leq a \bullet y \Rightarrow a'^+((a \bullet y)c'') = \textit{output } v'' = f(yc'').$$

If $f(xc'') = \textit{valof } c$, $x \leq y$ and $c \in A(y)$, then $a'^+((a \bullet x)c'') = \textit{valof } c'$ and $a^+(xc') = \textit{valof } c$; hence $c \in A(x)$, since $(xc', \textit{valof } c)$ is an event. Also, $c \in A(y)$ implies $a^+(yc') = \textit{valof } c$. In particular, $c' \notin F(a \bullet y)$, hence $c' \in A(a \bullet y)$ by (A3) applied to a^+. Then we obtain $a'^+((a \bullet y)c'') = \textit{valof } c'$ by (A2) applied to a'^+, which yields $f(yc'') = \textit{valof } c$.

$(A3)$ If $f(yc'') \neq \omega$, then $a'^+((a \bullet y)c'') \neq \omega$. Hence $c'' \in E(f \bullet y)$, since it is easily checked that $f \bullet y = a' \bullet (a \bullet y)$. Moreover if $z \leq y$ and $c'' \in E(f \bullet z)$, then $a \bullet z \leq a \bullet y$ and $c'' \in E(a' \bullet (a \bullet z))$, whence we derive $a'^+((a \bullet z)c'') \neq \omega$ by (A3) applied to a'^+. If $a'^+((a \bullet z)c'') = \textit{output } v''$, then $f(zc'') = \textit{output } v''$ by definition. If $a'^+((a \bullet z)c'') = \textit{valof } c'$, then $c' \in A(a \bullet z) \subseteq E(a \bullet z)$. We show $a^+(yc') \neq \omega$. There are two cases:

1. $c' \in F(a \bullet y)$: Then $a^+(yc') \neq \omega$ by definition of $a \bullet y$.
2. $c' \in A(a \bullet y)$: Then $a'^+((a \bullet y)c'') = \textit{valof } c'$ by (A2) applied to a'^+, which forces $a^+(yc') = \textit{valof } c$, for some c, since $f(yc') \neq \omega$.

In both cases, $a^+(yc') \neq \omega$, hence $a^+(zc') \neq \omega$ by (A3) applied to a^+; moreover $a^+(zc')$ is of type "valof", since the contrary would imply $c' \in F(a \bullet z)$. Hence $f(zc'') \neq \omega$. □

We remark that the definition of f in proposition 14.2.13 makes sense, since we have seen that an abstract algorithm can be extended to (a subset of) $D(\mathbf{M}) \times \mathbf{C}'$.

Theorem 14.2.14 *Cds's and sequential algorithms form a category called* **ALGO**. *Let a, a', and f be as in proposition 14.2.13. We define the composition $a' \circ a$ of a' and a by the following equation:*

$$a' \circ a = f^-.$$

For any cds $\mathbf{M}$ *there exists a unique algorithm id such that $\lambda x.(id \bullet x)$ is the identity function. It is characterised by:*

$$\begin{array}{ll} id^+(xc) = output\ v & iff\ \ (c, v) \in x \\ id^+(xc) = valof\ c & iff\ \ c \in A(x)\ . \end{array}$$

In particular, the input-output function of $a' \circ a$ is the composition of the input-output functions of a and a'.

14.3 Algorithms as Strategies

We first define sequential data structures, which enhance the implicit symmetry between events and enablings in a filiform cds. Then we define the affine exponent $\mathbf{S}$ ($\mathbf{S}'$ of two sequential data structures $\mathbf{S}$ and $\mathbf{S}'$. The states of $\mathbf{S}$ ($\mathbf{S}'$ are called affine algorithms. Like sequential algorithms, affine algorithms can be equivalently presented abstractly. The abstract affine algorithms are called symmetric algorithms. A symmetric algorithm is a pair (f, g) of a function f from input strategies to output strategies, and of a partial function g from output counter-strategies to input counter-strategies. The composition of two affine algorithms can be defined either abstractly (proposition 14.3.34) or concretely (proposition 14.3.38). The concrete description serves to establish the monoidal closed structure of the category of affine algorithms, while the abstract characterisation serves to define a functor **cds** from the category of sequential data structures and affine algorithms to the category of concrete data structures and sequential algorithms. Finally, we show that **cds** has a left adjoint, which together with the affine exponent yields a decomposition of the exponent of **ALGO**.

Definition 14.3.1 *A* sequential data structure *(sds for short)* $\mathbf{S} = (C, V, P)$ *is given by two sets C and V of cells and values, which are assumed disjoint, and by a collection P of non-empty words p of the form*

$$c_1 v_1 \cdots c_n v_n \ \text{ or } \ c_1 v_1 \cdots c_{n-1} v_{n-1} c_n$$

where $n \geq 0$ and where $c_i \in C$ and $v_i \in V$ for all i. Thus any $p \in P$ is alternating and starts with a cell. Moreover, it is assumed that P is closed under non-empty prefixes. We call the elements of P positions *of* $\mathbf{S}$. *We call* move *any element of $M = C \cup V$. We use m to denote a move. A position ending with a value is called a* response, *and a position ending with a cell is called a* query. *We use p (or s, or t), q, and r, to range over positions, queries, and responses, respectively. We denote by Q and R the sets of queries and responses, respectively.*

A strategy *of* $\mathbf{S}$ *is a subset x of R that is closed under response prefixes and binary non-empty glb's:*

$$r_1, r_2 \in x, r_1 \wedge r_2 \neq \epsilon \;\; \Rightarrow \;\; r_1 \wedge r_2 \in x$$

where ϵ denotes the empty word. A counter-strategy *is a non-empty subset of Q that is closed under query prefixes and under binary glb's. We use $x, y, \ldots$ and $\alpha, \beta, \ldots$ to range over strategies and counter-strategies, respectively.*

Both sets of strategies and of counter-strategies are ordered by inclusion. They are denoted by $D(\mathbf{S})$ and $D^{\perp}(\mathbf{S})$, respectively. Notice that $D(\mathbf{S})$ has always a minimum element (the empty strategy, written $\emptyset$ or $\perp$), while $D^{\perp}(\mathbf{S})$ has no minimum element in general. If a partial order is isomorphic to some $D(\mathbf{S})$, it is called an sds domain *generated by* $\mathbf{S}$.

Among the strategies are the sets of response prefixes of a response r. By abuse of notation we still call r the resulting strategy. It is easy to see that those r's are exactly the prime elements of $D(\mathbf{S})$ (cf.definition 10.1.5).

Definition 14.3.2 *Let x be a strategy:*

- *If $qv \in x$ for some v, we write $q \in F(x)$ (q is filled in x).*
- *If $r \in x$ and $q = rc$ for some c, we say that q is enabled in x.*
- *If q is enabled but $q \notin F(x)$, we write $q \in A(x)$ (q is accessible from x).*

Likewise we define $r \in F(\alpha), r \in A(\alpha)$ for a response r and a counter-strategy α.

Sds's and (filiform) cds's are essentially the same notion, as shown in proposition 14.3.3, lemma 14.3.5, and exercise 14.3.7.

Proposition 14.3.3 *Let $\mathbf{S} = (C, V, P)$ be an sds, and let Q and R be the associated sets of queries and responses. Let $\mathbf{cds}(\mathbf{S}) = (Q, R, E, \vdash)$, with*

$$E = \{(q, qv) \mid qv \in P\} \quad \vdash c \quad \text{if } c \in C \cap P \quad (q, qv) \vdash qvc \quad \text{if } qvc \in P.$$

Then $\mathbf{cds}(\mathbf{S})$ is a filiform cds and $D(cds(\mathbf{S}))$ is isomorphic to $D(\mathbf{S})$.

PROOF. With a strategy x of $\mathbf{S}$, we associate $\mathbf{cds}(x) = \{(q, qv) \mid qv \in x\}$, which is consistent and safe by the definition of a strategy. More precisely, consistency follows from the closure under glb's, and safety follows from the closure under prefixes. This transformation is clearly bijective. □

Proposition 14.3.4 *If* $\mathbf{S}$ *is an sds, then* $D(\mathbf{S})$ *is a dI-domain, whose compact elements are the finite strategies.* $D^{\perp}(\mathbf{S})$ *enjoys the same properties (except for the existence of a minimum element). Upper bounded lub's and glb's are set-theoretic unions and intersections.*

PROOF. The proof is similar to the proof of proposition 14.1.5. (The first part of the statement is a consequence of propositions 14.1.5 and 14.3.3.) □

Lemma 14.3.5 *Let* $\mathbf{M}$ *be a well-founded, stable, and filiform cds. For any cell* c*, any two distinct proofs* t_1 *and* t_2 *of* c*, there exists a common prefix* s*, a cell* d*, and two distinct values* v_1 *and* v_2 *such that* $s,(d,v_1)$ *is a prefix of* t_1 *and* $s,(d,v_2)$ *is a prefix of* t_2*.*

PROOF. We proceed by induction on c. We observe:

$t_1 \cup t_2$ is safe	by construction
$t_1 \cup t_2$ is not a state	by stability
no cell is repeated along t_1, nor along t_2	by well-foundedness .

These observations entail that $(d,w_1) \in t_1$ and $(d,w_2) \in t_2$ for some cell $d \ll^{+} c$ and for some distinct w_1 and w_2. If the proofs of d in t_1 and t_2 are distinct, the conclusion follows by applying induction to d. If the proofs are the same, then the conclusion follows rightaway. □

Remark 14.3.6 *Conversely, the property stated in lemma 14.3.5 implies that* $\mathbf{M}$ *is stable, hence we could have used it to define the notion of stable (filiform) cds.*

Exercise 14.3.7 *Let* $\mathbf{M} = (C,V,E,\vdash)$ *be a well-founded, stable, and filiform cds. Show that* $\mathbf{sds}(\mathbf{M}) = (C,V,P)$*, where*

$$\begin{aligned} P \;=\; & \{c_1 v_1 \cdots c_n v_n c_{n+1} \mid (c_1,v_1),\ldots,(c_n,v_n) \text{ is a proof of } c\} \cup \\ & \{rcv \mid rc \in P \text{ and } (c,v) \in E\}\,. \end{aligned}$$

is an sds such that $D(\mathbf{sds}(\mathbf{M}))$ *and* $D(\mathbf{M})$ *are isomorphic. Hint: use lemma 14.3.5.*

Example 14.3.8 *1. Flat cpo's. In the setting of sds's:*

$$X_{\perp} = (\{?\}, X, \{?\} \cup \{?v \mid v \in X\}).$$

2. The following generates $\mathbf{B}^2_{\perp}$ *(see definition 14.3.43 for the general case):*

$$(\{?.1,?.2\},\{tt,ff\},\{?.1,?.2\} \cup \{(?.1)tt,(?.1)ff,(?.2)tt,(?.2)ff\}).$$

3. An sds generating the partial terms over a signature, say, $\Sigma = \{a^0, f^1, g^2\}$*, where the superscripts indicate the arities, is given as follows:* $C = \{\epsilon,1,2\}$*,* $V = \Sigma$*, and* P *consists of the positions respecting the arities: the positions ending*

with a are maximal, the positions ending with f can only be followed by 1, and the positions ending with g can be followed by 1 or 2. All the positions start with ϵ (which serves only to that purpose). For example, the strategy representing $g(a, f(a))$ is

$$\{\epsilon g, \epsilon g1a, \epsilon g2f, \epsilon g2f1a\}.$$

Here is a counter-strategy:

$$\{\epsilon, \epsilon f1, \epsilon f1f1, \epsilon f1g2, \epsilon g1\}.$$

In example 14.3.8 (3), a counter-strategy can be read as an exploration tree, or a pattern. The root is investigated first; if the function symbol found at the root is g, then its left son is investigated next; otherwise, if the function symbol found at the root is f, then its son is investigated next, and the investigation goes further if the symbol found at node 1 is either f or g.

A more geometric reading of the definitions of sds, strategy and counter-strategy is the following:

- An sds is a labelled forest, where the ancestor relation alternates cells and values, and where the roots are labelled by cells.
- A strategy is a sub-forest which is allowed to branch only at values.
- A counter-strategy α is a non-empty sub-tree (if it contained c_1 and c_2 as positions of length 1, they should contain their glb, which is ϵ, contradicting $\alpha \subseteq P$) which is allowed to branch only at cells.

The pairs cell – value, query – response, and strategy – counter-strategy give to sds's a flavour of symmetry. These pairs are related to other important dualities in programming: input – output, constructor – destructor (cf. example 14.3.8 (3)). It is thus tempting to consider the counter-strategies of an sds $\mathbf{S}$ as the strategies of a dual structure $\mathbf{S}^\perp$ whose cells are the values of $\mathbf{S}$ and whose values are the cells of $\mathbf{S}$. However, the structure obtained in this way is not an sds anymore, since positions now start with a value. We refer to [Lam92a] for an elaboration of a theory of sds's with polarities, where both $\mathbf{S}$ and $\mathbf{S}^\perp$ can live (see also exercises 14.3.23 and 14.3.40).

We now offer a reading of sds's as games. An sds can be considered as a game between two persons, the *opponent* and the *player*. The values are the player's moves, and the cells are the opponent's moves. A player's strategy consists in having ready answers for (some of) the opponent's moves. Counter-strategies are opponent's strategies. The following proposition makes the analogy more precise.

Definition 14.3.9 (play) *Let $\mathbf{S}$ be an sds, x be a strategy and α be a counter-strategy of $\mathbf{S}$, one of which is finite. We define $x \mid \alpha$, called a* play*, as the set of positions p which are such that all the response prefixes of p are in x and all the query prefixes of p are in α.*

Proposition 14.3.10 *Given x and α as in definition 14.3.9, the play $x \mid \alpha$ is non-empty and totally ordered, and can be confused with its maximum element, which is uniquely characterised as follows:*

$$\begin{array}{ll} x \mid \alpha \text{ is the unique element of } x \cap A(\alpha) & \text{if } x \mid \alpha \text{ is a response} \\ x \mid \alpha \text{ is the unique element of } \alpha \cap A(x) & \text{if } x \mid \alpha \text{ is a query}. \end{array}$$

PROOF. A counter-strategy is non-empty by definition, and contains a (unique) query c of length 1, which is also in $x \mid \alpha$ by definition of a play. Suppose that $p_1, p_2 \in x \mid \alpha$. We show that p_1 and p_2 are comparable, by contradiction. Thus suppose $p_1 \wedge p_2 < p_1$ and $p_1 \wedge p_2 < p_2$. Let q_1 be the largest query prefix of p_1, let r_1 be the largest prefix of p_1 which is a response or ϵ, and let q_2 and r_2 be defined similarly. We show:

$$p_1 \wedge p_2 = q_1 \wedge q_2 = r_1 \wedge r_2.$$

The inequality $q_1 \wedge q_2 \leq p_1 \wedge p_2$ follows by the monotonicity of $\wedge$. For the other direction, we remark that by the maximality of q_1, $p_1 \wedge p_2 < p_1$ implies $p_1 \wedge p_2 \leq q_1$; and, similarly, we deduce $p_1 \wedge p_2 \leq q_2$, which completes the proof of $p_1 \wedge p_2 \leq q_1 \wedge q_2$. The equality $p_1 \wedge p_2 = r_1 \wedge r_2$ is proved similarly. But by definition of a strategy and of a counter-strategy, $q_1 \wedge q_2$ is a query, and $r_1 \wedge r_2$ is either a response or ϵ. The equalities just proven imply that $p_1 \wedge p_2$ is of both odd and even length: contradiction. Thus $x \mid \alpha$ is totally ordered. It has a maximum element, since the finiteness of x or α implies the finiteness of $x \mid \alpha$.

To prove the rest of the statement, we first observe that $x \cap A(\alpha) \subseteq x \mid \alpha$ and $\alpha \cap A(x) \subseteq x \mid \alpha$, by definition of $x \mid \alpha$. We next show that $x \cap A(\alpha)$ and $\alpha \cap A(x)$ have at most one element. If $p_1, p_2 \in x \cap A(\alpha)$, then by the first part of the statement p_1 and p_2 are comparable, say $p_1 \leq p_2$. But if $p_2 \in A(\alpha)$ and $p_1 < p_2$, then $p_1 \in F(\alpha)$, contradicting the assumption $p_1 \in A(\alpha)$. Hence $p_1 = p_2$. The proof is similar for $\alpha \cap A(x)$. Finally, if $x \mid \alpha$ viewed as a position is a response, then $x \mid \alpha \in x$, $x \mid \alpha$ is enabled in α, and the maximality of $x \mid \alpha$ implies that $x \mid \alpha$ is not filled in α. Hence $x \mid \alpha \in x \cap A(\alpha)$, i.e., $x \cap A(\alpha) = \{x \mid \alpha\}$. □

Definition 14.3.11 (winning) *Let x and α be as in definition 14.3.9. If $x \mid \alpha$ is a response, we say that x wins against α, and we denote this predicate by $x \lhd \alpha$. If $x \mid \alpha$ is a query, we say that α wins against x, and we write $x \rhd \alpha$, thus $\rhd$ is the negation of $\lhd$. To stress who is the winner, we write:*

$$x \mid \alpha = \begin{cases} x \lhd\mid \alpha & \text{when } x \text{ wins} \\ x \mid\rhd \alpha & \text{when } \alpha \text{ wins}. \end{cases}$$

The position $x \mid \alpha$ formalises the interplay between the player with strategy x and the opponent with strategy α. If $x \mid \alpha$ is a response, then the player wins since he made the last move, and if $x \mid \alpha$ is a query, then the opponent wins. Here is a game-theoretic reading of $x \mid \alpha$. At the beginning the opponent makes

a move c: his strategy determines that move uniquely. Then either the player is unable to move (x contains no position of the form cv), or his strategy determines a unique move. The play goes on until one of x or α does not have the provision to answer its opponent's move. As an example, if x and α are the strategy and counter-strategy of example 14.3.8 (3), then $x \mid \alpha = \epsilon g 1 a$, and the player wins. We show a few technical lemmas.

Lemma 14.3.12 *Let* $\mathbf{S}$ *be an sds,* x *be a strategy and* α *be a counter-strategy of* $\mathbf{S}$*. The following properties hold:*

1. *If* $x \lhd \alpha$*, then* $(x \lhd\mid \alpha) \lhd \alpha$.
2. *If* $x \lhd \alpha$ *and* $x \leq y$*, then* $y \lhd \alpha$ *and* $x \lhd\mid \alpha = y \lhd\mid \alpha$.
3. *If* $x \rhd \alpha$ *and* $y \leq x$*, then* $y \rhd \alpha$.

Similar implications hold with the assumptions $x \rhd \alpha$*,* ($x \rhd \alpha$ *and* $\alpha \leq \beta$*), and* ($x \rhd \alpha$ *and* $\beta \leq \alpha$*), respectively.*

PROOF. The properties (1) and (2) follow obviously from the characterisation of $x \lhd\mid \alpha$ as the unique element of $x \cap A(\alpha)$. Property (3) is a consequence of (2) by contraposition. □

Lemma 14.3.13 *Let* $\mathbf{S}$ *be an sds,* x *be a strategy and* q *be a query of* $\mathbf{S}$*. The following implications hold:*

1. $q \in F(x) \Rightarrow x \lhd q$,
2. $q \in A(x) \Rightarrow x \rhd q$,
3. $(q \in F(x), y \leq x, y \lhd q) \Rightarrow q \in F(y)$.

Similar implications hold with a counter-strategy and a response of $\mathbf{S}$*.*

PROOF. If $q \in F(x)$, then $qv \in x$ for some v, hence $qv \in x \cap A(q)$, which means $x \lhd q$. If $q \in A(x)$, then $q \in q \cap A(x)$, which means $x \rhd q$. If $q \in F(x), y \leq x$, and $y \lhd q$, let $q_1 v_1$ be the unique element of $y \cap A(q)$. In particular, $q_1 \leq q$. Suppose $q_1 < q$: then $q_1 v_1 \wedge qv = q_1$, since $q_1 v_1 \not\leq q$. On the other hand, the glb of $q_1 v_1$ and qv, cannot be a query, by definition of a strategy: contradiction. □

The converse of lemma 14.3.13 (1) is not true: we may have $x \lhd\mid q = q_1 v_1$ and $q = q_1 v_2$, with $v_1 \neq v_2$.

Lemma 14.3.14 *Let* $\mathbf{S} = (C, V, P)$ *be an sds,* x *be a strategy and let* $q \in A(x)$*. The following properties hold:*

1. For any $r \in x$*,* $q \wedge r$ *is* ϵ *or is a response, and thus, for any* $qv \in P$*,* $x \cup \{qv\}$ *is a strategy.*

2. If $q_1 \neq q$ *and* $q_1 \in A(x)$*, then* $q_1 \wedge q$ *is a strict prefix of* q_1 *and* q *and is* ϵ *or a response*

Similar properties hold with a counter-strategy α *and a response* r *such that* $r \in A(\alpha)$

PROOF. We prove only (1). Let $q = r_1 c$. We claim that $q \wedge r \leq r_1$. Suppose $q \wedge r \not\leq r_1$. Then $q \wedge r = q$ since $q \wedge r \leq q = r_1 c$. Hence $q < r$, contradicting $q \in A(x)$. The claim in turn implies $q \wedge r = r_1 \wedge r$. The conclusion follows, since by definition of a strategy $r_1 \wedge r$ is ϵ or is a response. □

Lemma 14.3.15 *Let* $\mathbf{S}$ *be an sds, and let* $x \in D(\mathbf{S})$ *and* $q \in F(x)$. *Then* $x - q = \{r \in x \mid q \not< r\}$ *is a strategy.*

PROOF. Since $\{r \in R \mid q \not< r\}$ is closed under response prefixes, so is $x - q$. □

Lemma 14.3.16 *If* $r_1, r_2 \in R$ *and* $r_1 \wedge r_2$ *is* ϵ *or is a response, then* $\{r \in R \mid r \leq r_1$ *or* $r \leq r_2\}$ *is a strategy, and is* $r_1 \vee r_2$.

PROOF. $\{r \in R \mid r \leq r_1 \text{ or } r \leq r_2\}$ is obviously closed under response prefixes. Pick r_3, r_4 in this set. If they are both prefixes of, say, r_1, then they are comparable, hence, say, $r_3 \wedge r_4 = r_3$ is a response. Thus we may suppose, say, $r_3 \leq r_1$, $r_3 \not\leq r_2$, $r_4 \leq r_2$, and $r_4 \not\leq r_1$. This entails $r_3 > r_1 \wedge r_2$ and $r_4 > r_1 \wedge r_2$, and therefore $r_3 \wedge r_4 = r_1 \wedge r_2$. □

Exercise 14.3.17 *Let* $\mathbf{S}$ *and* $\mathbf{S}'$ *be sds's. Show that a continuous function* $f : D(\mathbf{S}) \to D(\mathbf{S}')$ *is sequential iff, for any pair* $(x, \alpha') \in \mathcal{K}(D(\mathbf{S})) \times \mathcal{K}(D^{\perp}(\mathbf{S}'))$ *such that* $f(x) \triangleright \alpha'$ *and* $f(z) \triangleleft \alpha'$ *for some* $z \geq x$, *there exists* $\alpha \in \mathcal{K}(D^{\perp}(\mathbf{S}))$, *called generalised sequentiality index (index for short) of* f *at* (x, α'), *such that* $x \triangleright \alpha$ *and for any* $y \geq x$, $f(y) \triangleleft \alpha'$ *implies* $y \triangleleft \alpha$.

We next define the affine exponent of two sds's, which will serve to define the morphisms of a category of affine algorithms.

Definition 14.3.18 *Given sets* $A, B \subseteq A$, *for any word* $w \in A^{\star}$, *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 14.3.19 (affine exponent – sds) *Given two sds's* $\mathbf{S} = (C, V, P)$ *and* $\mathbf{S}' = (C', V', P')$, *we define* $\mathbf{S}$ ($\mathbf{S}' = (C'', V'', P'')$ *as follows. The sets* C'' *and* V'' *are disjoint unions:*

$$\begin{aligned} C'' &= \{\textit{request } c' \mid c' \in C'\} \cup \{\textit{is } v \mid v \in V\} \\ V'' &= \{\textit{output } v' \mid v' \in V'\} \cup \{\textit{valof } c \mid c \in C\}\ . \end{aligned}$$

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

$$s\lceil_{\mathbf{S}'} \in P', (s\lceil_{\mathbf{S}} = \epsilon \textit{ or } s\lceil_{\mathbf{S}} \in P), \textit{ and}$$
$$s \textit{ has no prefix of the form } s(\textit{valof } c)(\textit{request } c').$$

We often omit the tags $\textit{request}, \textit{valof}, \textit{is}, \textit{output}$, *as we have just done in the notation* $s\lceil_{\mathbf{S}} = s\lceil_{C \cup V}$ *(and similarly for* $s\lceil_{\mathbf{S}'}$*).*

We call affine sequential algorithms *(or affine algorithms) from* $\mathbf{S}$ *to* $\mathbf{S}'$ *the strategies of* $\mathbf{S} \multimap \mathbf{S}'$. *The identity sequential algorithm* $id \in D(\mathbf{S} \multimap \mathbf{S}')$ *is defined as follows:*

$$id = \{copycat(r) \mid r \text{ is a response of } \mathbf{S}\}$$

where copycat is defined as follows:

$$\begin{array}{lcl} copycat(\epsilon) & = & \epsilon \\ copycat(rc) & = & copycat(r)(request\ c)(valof\ c) \\ copycat(qv) & = & copycat(q)(is\ v)(output\ v)\ . \end{array}$$

The word *copycat* used in the description of the identity algorithm has been proposed by Abramsky, and corresponds to a game-theoretic understanding: the player always repeats the last move of the opponent.

Remark 14.3.20 *The definition also implies that* P' *contains no position of the form* $sv'v$. *Suppose it does: then since* $(sv'v)\lceil_{\mathbf{S}} \in P$, s *contains a prefix* s_1c *such that* $(sv'v)\lceil_{\mathbf{S}} = ((s_1c)\lceil_{\mathbf{S}})v$. *Let* m *be the move following* s_1c *in* sv'. *Then*

$m \notin V$ *since* $(sv'v)\lceil_{\mathbf{S}} = ((s_1c)\lceil_{\mathbf{S}})v$,
$m \notin C'$ *by the definition of* $\mathbf{S} \multimap \mathbf{S}'$.

The constraint "no *scc*'" can be formulated more informally as follows. Thinking of *valof c* as a call to a subroutine, the principal routine cannot proceed further until it receives a result v from the subroutine.

Example 14.3.21 *1. It should be clear that the following is an affine algorithm which computes the boolean negation function:*

$\{(request\ ?)(valof\ ?),$
$(request\ ?)(valof\ ?)(is\ tt)(output\ ff),$
$(request\ ?)(valof\ ?)(is\ ff)(output\ tt)\}$.

2. On the other hand, the left disjunction function cannot be computed by an affine algorithm. Indeed, attempting to write an sds version of the algorithm OR_l *of example 14.2.2 would result in*

$\{(request\ ?)(valof\ ?.1),$
$(request\ ?)(valof\ ?.1)(is\ tt)(output\ tt),$
$(request\ ?)(valof\ ?.1)(is\ ff)(valof\ ?.2),$
$(request\ ?)(valof\ ?.1)(is\ ff)(valof\ ?.2)(is\ tt)(output\ tt),$
$(request\ ?)(valof\ ?.1)(is\ ff)(valof\ ?.2)(is\ ff)(output\ ff)\}$.

which is not a subset of the set of positions of $(\mathbf{B}_\perp)^2 \multimap \mathbf{B}_\perp$, *because the projections on* $(\mathbf{B}_\perp)^2$ *of the last two sequences of moves are not positions of* $(\mathbf{B}_\perp)^2$.

3. Every constant function gives rise to an affine algorithm, whose responses have the form $(request\ c'_1)(output\ v'_1)\ldots(request\ c'_n)(output\ v'_n)$.

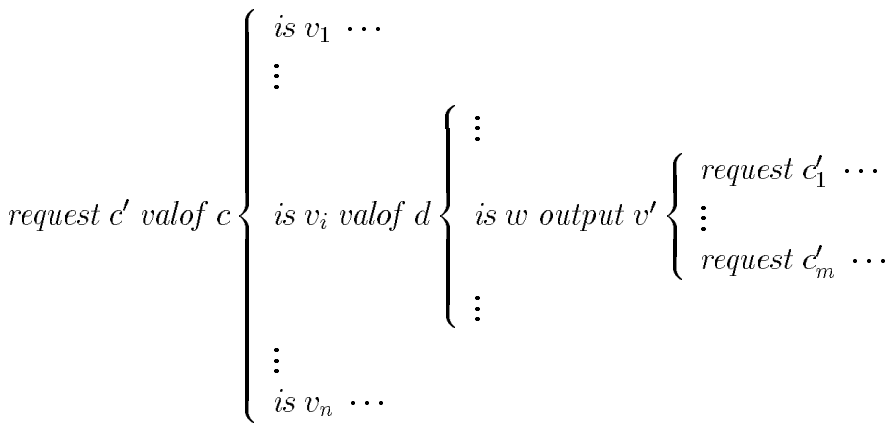
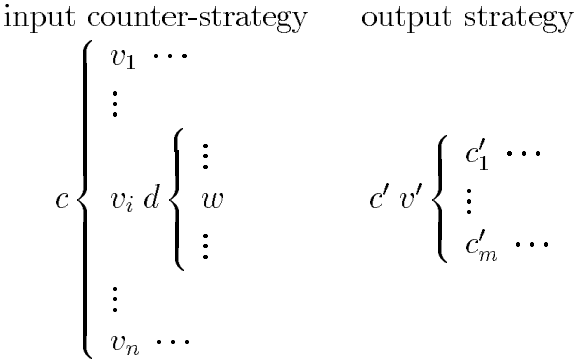


Figure 14.2: A generic affine algorithm

Remark 14.3.22 *Example 14.3.21 suggests the difference between affine and general sequential algorithms. Both kinds of algorithms ask successive queries to their input, and proceed only when they get responses to these queries. An affine algorithm is moreover required to ask these queries monotonically: each new query must be an extension of the previous one. The "unit" of resource consumption (cf. remark 13.1.8) is thus a sequence of queries/responses that can be arbitrarily large, as long as it builds a position of the input sds. The disjunction algorithms are not affine, because they may have to ask successively the queries* ?.1 *and* ?.2*, which are not related by the prefix ordering.*

A generic affine algorithm, as represented in figure 14.2, can be viewed as a "combination" of the following (generic) output strategy and input counter-strategy (or exploration tree):



An alternative presentation of the affine exponent, due to Lamarche [Lam92b] is given in exercise 14.3.23.

Exercise 14.3.23 *This exercise is based on a small variant of the presentation of an sds, whose advantage is to give a tree structure rather than a forest structure to the sds and to strategies. In this variant, an sds is a structure $(C, V \cup \{\bullet\}, P)$ where $\bullet$ is a distinguished element that does not belong to V (nor C), and where all positions of P start with $\bullet$ (this being the only place where $\bullet$ can occur). In this setting, strategies have to be non-empty. We say that a move $m \in C \cup (V \cup \{\bullet\})$ has:*

$$\begin{array}{ll} \textit{polarity} \bullet & \textit{if } m \in V \cup \{\bullet\} \\ \textit{polarity} \circ & \textit{if } m \in C\ . \end{array}$$

(1) Establish a precise correspondence between sds's and the present variants of sds's. (2) Based on these variants, construct the affine exponent of two sds's $(C, V \cup \{\bullet\}, P)$ and $(C', V' \cup \{\bullet\}, P')$ along the following lines.

(a) The moves of the affine exponent are pairs (m, m') of moves $m \in C \cup V$ and $m' \in C' \cup V'$ whose polarities are not in the combination $(\circ, \bullet)$.

(b) The moves (m, m') of polarity $\circ$ and those of polarity $\bullet$ are as indicated by the following table:

m	m'	(m, m')
$\bullet$	$\bullet$	$\bullet$
$\bullet$	$\circ$	$\circ$
$\circ$	$\bullet$	*undefined*
$\circ$	$\circ$	$\bullet$

(c) One moves only on one side at a time: if (m, m') is a move, it is followed by a move of the form (n, m') or (m, n').

We next state a key technical property.

Lemma 14.3.24 *Let $\phi : \mathbf{S} \to \mathbf{S}'$ be an affine algorithm between two sds's $\mathbf{S}$ and $\mathbf{S}'$. The following properties hold.*

1. The function $\lambda s.(s\lceil_{\mathbf{S}}, s\lceil_{\mathbf{S}'})$ is an order-isomorphism from ϕ to its image, ordered componentwise by the prefix ordering.

2. If two elements s_1 and s_2 of ϕ are such that $(s_1\lceil_{\mathbf{S}}) \wedge (s_2\lceil_{\mathbf{S}})$ is either ϵ or is a response, and if $s_1\lceil_{\mathbf{S}'}$ and $s_2\lceil_{\mathbf{S}'}$ are comparable, then s_1 and s_2 are comparable.

3. If two elements s_1 and s_2 of ϕ are such that $(s_1\lceil_{\mathbf{S}'}) \wedge (s_2\lceil_{\mathbf{S}'})$ is a query, and if $s_1\lceil_{\mathbf{S}}$ and $s_2\lceil_{\mathbf{S}}$ are comparable, then s_1 and s_2 are comparable.

PROOF. (2)(or (3)) $\Rightarrow$ (1) It is obvious that $\lambda s.(s\lceil_{\mathbf{S}}, s\lceil_{\mathbf{S}'})$ is monotonic. Suppose that $s_1\lceil_{\mathbf{S}} \leq s\lceil_{\mathbf{S}}$, $s_1\lceil_{\mathbf{S}'} \leq s\lceil_{\mathbf{S}'}$, and $s_1 \not\leq s$. Then $s \leq s_1$ by the second part of the statement, and by monotonicity $s\lceil_{\mathbf{S}} \leq s_1\lceil_{\mathbf{S}}$ and $s\lceil_{\mathbf{S}'} \leq s_1\lceil_{\mathbf{S}'}$. Hence $s_1\lceil_{\mathbf{S}} = s\lceil_{\mathbf{S}}$, $s_1\lceil_{\mathbf{S}'} = s\lceil_{\mathbf{S}'}$, and $s = s_1$ follows, since $s < s_1$ would imply either $s\lceil_{\mathbf{S}} < s_1\lceil_{\mathbf{S}}$ or $s\lceil_{\mathbf{S}'} < s_1\lceil_{\mathbf{S}'}$.

(2) Let $t = s_1 \wedge s_2$, which is ϵ or is a response, since ϕ is a strategy. Suppose that $t < s_1$ and $t < s_2$. If t has the form $t_1 c$, then $t < s_1$ and $t < s_2$ imply

that $tv_1 \leq s_1$ and $tv_2 \leq s_2$ for some v_1 and v_2, which must be different since $t = s_1 \wedge s_2$: but then $(s_1\lceil_{\mathbf{S}}) \wedge (s_2\lceil_{\mathbf{S}})$ is a query, contradicting the assumption. If t is ϵ or has the form $t_1 v'$, then $t < s_1$ and $t < s_2$ imply that $tc'_1 < s_1$ and $tc'_2 < s_2$ for some c'_1 and c'_2, which must be different since $t = s_1 \wedge s_2$: this contradicts the assumption that $s_1\lceil_{\mathbf{S}'}$ and $s_2\lceil_{\mathbf{S}'}$ are comparable. Hence $t = s_1$ or $t = s_2$, i.e., $s_1 \leq s_2$ or $s_2 \leq s_1$. The proof of (3) is similar. □

Remark 14.3.25 *Any pair $(s\lceil_{\mathbf{S}}, s\lceil_{\mathbf{S}'})$ in the image of ϕ under the mapping $\lambda s.(s\lceil_{\mathbf{S}}, s\lceil_{\mathbf{S}'})$ is either a pair of responses or a pair of queries. It is a pair of responses iff s ends with a value v'; it is a pair of queries iff s ends with a cell c.*

There exists a more abstract description of affine algorithms, which we shall come to after some preliminaries.

Definition 14.3.26 (affine function) *Let $\mathbf{S}$ and $\mathbf{S}'$ be two sds's. We call a function $f : D(\mathbf{S}) \to D(\mathbf{S}')$ affine when it is stable and satisfies the following condition:*

$$r' \in f(x) \Rightarrow (\exists\, r \in x \;\; r' \in f(r)).$$

Equivalently, an affine function can be defined as a stable function preserving lub's of pairs of compatible elements. The definition applies also to (partial) functions $g : D^{\perp}(\mathbf{S}') \rightharpoonup D^{\perp}(\mathbf{S})$.

If a fonction $f : D(\mathbf{S}) \to D(\mathbf{S}')$ is affine, then it is natural to adopt the following definition of trace:

$$\mathit{trace}(f) = \{(r, r') \mid r' \leq f(r) \text{ and } (\forall\, r_0 < r \;\; r' \not\leq f(r_0))\} \subseteq (R \cup \{\epsilon\}) \times R'$$

(and likewise for $g : D^{\perp}(\mathbf{S}') \rightharpoonup D^{\perp}(\mathbf{S})$).

Lemma 14.3.27 *The composition of two affine functions is affine, and its trace is the relation composition of the traces of f and g.*

PROOF. This is a straightforward variant of (the dI-domain version of) proposition 13.1.9 (cf. exercise 13.1.10). □

Proposition 14.3.28 *Any affine function f between two sds's is strongly sequential.*

PROOF. Let $q' \in A(f(x))$ be such that $r' = q'v' \in f(z)$ for some v' and $z \geq x$. Let r be the unique response such that $(r, r') \in \mathit{trace}(f)$ and $r \in z$. Let q be the unique query such that $q < r$ and $q \in A(x)$. Now consider $z_1 \geq x$ such that $q' \in F(f(z_1))$, and define $r'_1 = q'v'_1, r_1, q_1$ similarly. By lemma 14.3.14 (2), if $q_1 \neq q$, then $q_1 \wedge q$ is a strict prefix of q_1 and q, and is ϵ or a response. But then $q < r$ and $q_1 < r_1$ imply $q_1 \wedge q = r_1 \wedge r$. Therefore $r_1 \uparrow r$ by lemma 14.3.16, which implies $r'_1 \uparrow r'$ by definition of a trace. Therefore $v'_1 = v'$, hence $r'_1 = r'$, which implies $r_1 = r$ by stability, and $q_1 = q$ by construction. Thus q is a sequentiality

index of f at (x, q'). Suppose now that q_1 is another sequentiality index of f at (x, q'). Let z be as above, and consider $z - q$ and $z - q_1$ (cf. lemma 14.3.15). By affinity, $f(z) = f(z - q) \vee f(z - q_1)$, therefore, say, $q' \in F(f(z - q))$, which contradicts the fact that q is a sequentiality index. □

The converse is not true: there are strongly sequential functions that are not affine: the left and the right disjunction functions are examples.

Now we are ready to give an abstract description of affine algorithms.

Definition 14.3.29 (symmetric algorithm) *Let* $\mathbf{S}$ *and* $\mathbf{S}'$ *be two sds's. A symmetric algorithm from* $\mathbf{S}$ *to* $\mathbf{S}'$ *is a pair*

$$(f : D(\mathbf{S}) \to D(\mathbf{S}'), g : D^{\perp}(\mathbf{S}') \rightharpoonup D^{\perp}(\mathbf{S}))$$

of a function and a partial function that are both continuous and satisfy the following axioms:

$$(L) \quad (x \in D(\mathbf{S}), \alpha' \in \mathcal{K}(D^{\perp}(\mathbf{S}')), f(x) \lhd \alpha') \Rightarrow \begin{cases} x \lhd g(\alpha') \text{ and} \\ m(f, x, \alpha') = x \lhd\!\mid g(\alpha') \end{cases}$$

$$(R) \quad (\alpha' \in D^{\perp}(\mathbf{S}'), x \in \mathcal{K}(D(\mathbf{S})), x \rhd g(\alpha') \Rightarrow \begin{cases} f(x) \rhd \alpha' \text{ and} \\ m(g, \alpha', x) = f(x) \mid\!\rhd \alpha' \,. \end{cases}$$

where $m(f, x, \alpha')$ is the minimum $y \leq x$ such that $f(y) \lhd \alpha'$ ($m(g, \alpha', x)$ is defined similarly). We set as a convention, for any x and any α' such that $g(\alpha')$ is undefined:

$$x \lhd g(\alpha') \text{ and } x \lhd\!\mid g(\alpha') = \emptyset.$$

Thus the conclusion of (L) is simply $m(f, x, \alpha') = \emptyset$ when $g(\alpha')$ is undefined. In contrast, when we write $x \rhd g(\alpha')$ in (R), we assume that $g(\alpha')$ is defined. (This convention is consistent with the setting of exercise 14.3.23.) The collection of symmetric algorithms is ordered componentwise by the pointwise ordering:

$$(f_1, g_1) \leq (f_2, g_2) \quad \textit{iff} \quad ((\forall x \;\; f_1(x) \leq f_2(x)) \text{ and } (\forall \alpha \;\; g_1(\alpha) \downarrow \Rightarrow g_1(\alpha) \leq g_2(\alpha))).$$

These axioms enable us, knowing f and g, to reconstruct the traces of f and g. They also imply that f and g are affine (and sequential). Moreover, g allows to compute the sequentiality indices of f, and conversely.

Proposition 14.3.30 *Let f and g be as in the previous definition. Then f and g are affine and satisfy the following two axioms:*

(LS) If $x \in D(\mathbf{S})$, $\alpha' \in \mathcal{K}(D^{\perp}(\mathbf{S}'))$, $f(x) \rhd \alpha'$ and $f(y) \lhd \alpha'$ for some $y > x$, then $x \rhd g(\alpha')$, and $x \mid\!\rhd g(\alpha')$ is a sequentiality index of f at (x, α').

(RS) If $\alpha' \in D^{\perp}(\mathbf{S}')$, $x \in \mathcal{K}(D(\mathbf{S}))$, $x \lhd g(\alpha')$ and $x \rhd g(\beta')$ for some $\beta' > \alpha'$, then $f(x) \lhd \alpha'$, and $f(x) \lhd\!\mid \alpha'$ is a sequentiality index of g at (α', x).

PROOF. We first show that f is affine. Suppose $q'v' \in f(x)$. Then $f(x) \lhd q'$. By (L), $x \lhd g(q')$ and $f(r) \lhd q'$, where $r = x \lhd\!| \, g(q')$. Let $q'_1 v'_1 = f(r) \lhd\!| \, q'$, and suppose $q'_1 < q'$. On one hand $q'_1 v'_1 \in A(q')$ implies $q'_1 v'_1 \not\leq q'$. On the other hand:

$$\begin{array}{ll} q'_1 v'_1 \in f(x) & \text{since } q'_1 v'_1 \in f(r) \text{ and } r \leq x \\ q'_1 v'_1 \leq q' & \text{since } q'v', q'_1 v'_1 \in f(x)\ . \end{array}$$

Hence $q'_1 = q'$, and moreover $v'_1 = v'$ since $q'v', q'_1 v'_1 \in f(x)$. We have proved $f(r) \lhd\!| \, q' = q'v'$, and a fortiori $q'v' \in f(r)$.

We now prove that Axiom (L) implies property (LS). Suppose $x \in D(\mathbf{S})$ and $\alpha' \in \mathcal{K}(D^{\perp}(\mathbf{S}'))$, $f(x) \rhd \alpha'$ and $f(y) \lhd \alpha'$ for some $y > x$. By (L), we have $f(r_1) \lhd \alpha'$, where $r_1 = y \lhd\!| \, g(\alpha')$, which implies $r_1 \not\in x$ since $f(x) \rhd \alpha'$. Let r be the largest response prefix of r_1 contained in x, and let rc be such that $rc < r_1$. We claim that $x \mid g(\alpha') = rc$. From $r_1 \in A(g(\alpha'))$ and $rc < r_1$, we get $rc \in g(\alpha')$. We have $r \in x$ by construction, thus rc is enabled in x. If rc is filled in x, it must be filled with the same value v in x and r_1, contradicting the maximality of r. Hence $rc \in g(\alpha') \cap A(x)$, which proves the claim. The proof of (LS) is completed by observing that $rc < r_1, r_1 \leq y$ imply $rc \in F(y)$. □

A familiar feature of stability is not apparent in definition 14.3.29: the order is not defined as Berry's stable ordering. But the stable ordering is a derived property.

Exercise 14.3.31 *Show that if $(f_1, g_1) \leq (f_2, g_2)$ (cf. definition 14.3.29), then $f_1 \leq_{st} f_2$ and $g_1 \leq_{st} g_2$. Hint: apply (LS) to (f_1, g_1), (R) to (f_2, g_2), and (LS) to (f_2, g_2).*

We show the equivalence between the two presentations of affine algorithms, as strategies, and as pairs (f, g).

Theorem 14.3.32 *Let $\mathbf{S}$ and $\mathbf{S}'$ be two sds's. Given $\phi \in D(\mathbf{S} \multimap \mathbf{S}')$, we define a pair (f, g) of a function and a partial function as follows:*

$$\begin{array}{lcl} f(x) & = & \{r' \mid r' = s\lceil_{\mathbf{S}'} \text{ and } s\lceil_{\mathbf{S}} \in x \text{ for some } s \in \phi\} \\ g(\alpha') & = & \{q \mid q = s\lceil_{\mathbf{S}} \text{ and } s\lceil_{\mathbf{S}'} \in \alpha' \text{ for some } s \in \phi\}. \end{array}$$

By convention, if for some α' the right-hand side of the definition of g is empty, we interpret this definitional equality as saying that $g(\alpha')$ is undefined.

Conversely, given a symmetric algorithm (f, g) from $\mathbf{S}$ to $\mathbf{S}'$, we construct an affine algorithm $\phi \in D(\mathbf{S} \multimap \mathbf{S}')$ as follows. We build the positions s of ϕ by induction on the length of s:

- *If $s \in \phi$, if $s\lceil_{\mathbf{S}}$ and $s\lceil_{\mathbf{S}'}$ are responses, and if $q' = (s\lceil_{\mathbf{S}'})c'$ for some c', then:*

$$\begin{array}{ll} sc'c \in \phi & \text{if } (s\lceil_{\mathbf{S}})c \in g(q') \\ sc'v' \in \phi & \text{if } q'v' \in f(s\lceil_{\mathbf{S}})\ . \end{array}$$

- *If $s \in \phi$, if $s\lceil_{\mathbf{S}}$ and $s\lceil_{\mathbf{S}'}$ are queries, and if $r = (s\lceil_{\mathbf{S}})v$ for some v, then:*

$$\begin{array}{ll} svc \in \phi & \text{if } rc \in g(s\lceil_{\mathbf{S}'}) \\ svv' \in \phi & \text{if } (s\lceil_{\mathbf{S}'})v' \in f(r)\ . \end{array}$$

(The cases in the definition of ϕ are mutually exclusive, by (L).)

These two transformations define order-isomorphisms between $D(\mathbf{S} \parallel \mathbf{S}')$, ordered by inclusion, and the set of symmetric algorithms from $\mathbf{S}$ to $\mathbf{S}'$, ordered pointwise componentwise.

PROOF. We check only that (f, g) satisfies (L). If $x \in D(\mathbf{S})$, $\alpha' \in \mathcal{K}(D^{\perp}(\mathbf{S}'))$ and $f(x) \triangleleft \alpha'$, let $q'v' = f(x) \triangleleft\!| \alpha'$, and let $s \in \phi$ be such that $q'v' = s\lceil_{\mathbf{S}'}$ and $s\lceil_{\mathbf{S}} \in x$. Then s ends with v' (cf. remark 14.3.25). We claim:

$$\begin{array}{ll} i. & s\lceil_{\mathbf{S}} = x \triangleleft\!| g(\alpha') \\ ii. & s\lceil_{\mathbf{S}} = m(f, x, \alpha')\ . \end{array}$$

(i) Since $s\lceil_{\mathbf{S}} \in x$, we are left to show $s\lceil_{\mathbf{S}} \in A(g(\alpha'))$. Since $q'v' = f(x) \triangleleft\!| \alpha'$, we have $q'v' \in A(\alpha')$, hence $q' \in \alpha'$. We first show that $s\lceil_{\mathbf{S}}$ is enabled in $g(\alpha')$. Let $s\lceil_{\mathbf{S}} = qv$, and let s_1 be the least prefix of s such that $s_1\lceil_{\mathbf{S}} = q$. We claim that $s_1\lceil_{\mathbf{S}'} \in \alpha'$. By the definition of s_1, and since s ends with v', s_1 is a strict prefix of s and $s_1\lceil_{\mathbf{S}'} < s\lceil_{\mathbf{S}'}$. Hence $s_1\lceil_{\mathbf{S}'} \leq q'$, which implies the claim. Since $s_1\lceil_{\mathbf{S}} = q$, the claim implies $q \in g(\alpha')$ by definition of g, and that $s\lceil_{\mathbf{S}} = qv$ is enabled in $g(\alpha')$. Suppose now that $s\lceil_{\mathbf{S}}$ is filled in $g(\alpha')$. Then there exist c and $s_2 \in \phi$ such that $(s\lceil_{\mathbf{S}})c = s_2\lceil_{\mathbf{S}}$ and $s_2\lceil_{\mathbf{S}'} \in \alpha'$. By lemma 14.3.14 (1) and by lemma 14.3.24 (3), s and s_2 are comparable. But, since$(s\lceil_{\mathbf{S}})c = s_2\lceil_{\mathbf{S}}$, we cannot have $s_2 \leq s$, and since $s_2\lceil_{\mathbf{S}'} \in \alpha'$ and $s\lceil_{\mathbf{S}'} \in A(\alpha')$, we cannot have $s \leq s_2$: contradiction.

(ii) By definition of f, we have $s\lceil_{\mathbf{S}'} \in f(s\lceil_{\mathbf{S}})$, hence $f(s\lceil_{\mathbf{S}}) \triangleleft \alpha'$. Suppose now that $y \leq x$ and $f(y) \triangleleft \alpha'$. By lemma 14.3.12 (2), $f(y) \triangleleft\!| \alpha' = f(x) \triangleleft\!| \alpha'$, thus $q'v' \in f(y)$. Let $s_3 \in \phi$ be such that $q'v' = s_3\lceil_{\mathbf{S}'}$ and $s_3\lceil_{\mathbf{S}} \in y$. By lemma 14.3.24 (2), s and s_3 are comparable. Since s ends with v' and since $s_3\lceil_{\mathbf{S}'} = s\lceil_{\mathbf{S}'}$, s_3 cannot be a proper prefix of s. Thus $s \leq s_3$, and this entails $s\lceil_{\mathbf{S}} \in y$ since $s\lceil_{\mathbf{S}} \leq s_3\lceil_{\mathbf{S}}$ and $s_3\lceil_{\mathbf{S}} \in y$. □

The definition of f (the function computed by ϕ) in theorem 14.3.32 is so compact that it may hide the underlying operational semantics. The application of ϕ to a strategy x of $\mathbf{S}$ involves an interplay between ϕ and x that is very similar to the situation described in definition 14.3.9. We have already suggested that an affine algorithm "contains" input counter-strategies. Let ϕ be the generic algorithm of figure 14.2, and let x be the following input strategy, represented

suggestively as a forest:

$$\left\{\begin{array}{l} c\, v_i \left\{\begin{array}{l} d\, w \\ \vdots \\ d_1 \cdots \end{array}\right. \\ \vdots \\ c_1 \cdots \end{array}\right.$$

The matching of ϕ against x results in the "play" cv_1dw.

We turn to the composition of affine algorithms.

Definition 14.3.33 *Let* $\mathbf{S}$, $\mathbf{S}'$ *and* $\mathbf{S}''$ *be sds's, and let* (f,g) *and* (f',g') *be symmetric algorithms from* $\mathbf{S}$ *to* $\mathbf{S}'$ *and from* $\mathbf{S}'$ *to* $\mathbf{S}''$. *We define their composition* (f'',g'') *from* $\mathbf{S}$ *to* $\mathbf{S}''$ *as follows:*

$$f'' = f' \circ f \quad \textit{and} \quad g'' = g \circ g'.$$

Proposition 14.3.34 *The pair* (f'',g'') *in definition 14.3.33 indeed defines a symmetric algorithm.*

PROOF. We only check axiom (L). Suppose $f'(f(x)) \lhd \alpha''$. By (L) applied to (f',g'), we have $f(x) \lhd g'(\alpha'')$ and $m(f', f(x), \alpha'') = f(x) \lhd\!\mid g'(\alpha'')$. By (L) applied to (f,g), from $f(x) \lhd g'(\alpha'')$ we get $x \lhd g(g'(\alpha''))$ and $m(f, x, g'(\alpha'')) = x \lhd\!\mid g(g'(\alpha''))$. We have to prove $m(f' \circ f, x, \alpha'') = x \lhd\!\mid g(g'(\alpha''))$. We set $r = x \lhd\!\mid g(g'(\alpha''))$. Since $m(f, x, g'(\alpha'')) = r$, we have $f(r) \lhd g'(\alpha'')$. We claim that $f'(f(r)) \lhd \alpha''$. Suppose the contrary, that is, $f'(f(r)) \rhd \alpha''$. Then, by (LS) applied to (f',g') at $(f(r), \alpha'')$, we have $f(r) \rhd g'(\alpha'')$, which contradicts our previous deduction that $f(r) \lhd g'(\alpha'')$. Hence the claim holds. We are left to prove that, for any $y \leq x$ such that $f'(f(y)) \lhd \alpha''$, then $r \leq y$. Since $m(f, x, g'(\alpha'')) = r$, this second claim can be rephrased as $f(y) \lhd g'(\alpha'')$. We set $r' = f(x) \lhd\!\mid g'(\alpha'')$. Since $f(y) \leq f(x)$ and since $m(f', f(x), \alpha'') = r'$, we have $r' \leq f(y)$ by the first claim. But $r' \lhd g'(\alpha'')$ by definition of r' and by lemma 14.3.12 (1), and the conclusion follows by lemma 14.3.12 (2). □

Definition 14.3.35 *The category* **AFFALGO** *is defined as follows. Its objects are the sequential data stuctures and its morphisms are the affine algorithms. If* $\phi \in D(\mathbf{S}\ (\ \mathbf{S}')$ *and* $\phi' \in D(\mathbf{S}'\ (\ \mathbf{S}'')$, *if* (f,g) *and* (f',g') *are the symmetric algorithms associated with* ϕ *and* ϕ', *respectively, then* $\phi' \circ \phi$ *is the affine algorithm* ϕ'' *associated with* $(f' \circ f, g \circ g')$.

We interchangeably look at morphisms as affine algorithms or as symmetric algorithms. In particular, there are two descriptions of the identity morphism.

Exercise 14.3.36 *Show that* (id, id) *is the symmetric algorithm corresponding to the strategy id described in definition 14.3.19.*

Alternatively, composition can be defined operationally. This idea goes back to [BC85]. The form presented here is, mutatis mutandis, due to Abramsky [AJ92].

Lemma 14.3.37 *Let ϕ and (f,g) be as in the statement of theorem 14.3.32. Then we have the following equalities, where r, r' range over responses and q, q' range over queries:*

(1) $\mathit{trace}(f) = \{(r,r') \mid r = s\lceil_{\mathbf{S}} \text{ and } r' = s\lceil_{\mathbf{S}'} \text{ for some } s \in \phi\}$
(2) $\mathit{trace}(g) = \{(q',q) \mid q' = s\lceil_{\mathbf{S}'} \text{ and } q = s\lceil_{\mathbf{S}} \text{ for some } s \in \phi\}$.

PROOF. (1) If $r = s\lceil_{\mathbf{S}}$ and $q'v' = r' = s\lceil_{\mathbf{S}'}$, for some $s \in \phi$, then a fortiori $s\lceil_{\mathbf{S}} \leq r$, thus $r' \in f(r)$. Suppose that $r' \in f(r_1)$ for some $r_1 < r$. Let $s_1 \in \phi$ be such that $r' = s_1\lceil_{\mathbf{S}'}$ and $s_1\lceil_{\mathbf{S}} \leq r_1$. Then $(s_1\lceil_{\mathbf{S}}, s_1\lceil_{\mathbf{S}'}) < (s\lceil_{\mathbf{S}}, s\lceil_{\mathbf{S}'})$, which by lemma 14.3.24 implies $s_1 < s$. But by the definition of $\mathbf{S} \parallel \mathbf{S}'$, $r' = s\lceil_{\mathbf{S}'}$ implies that s ends with v', and hence $s_1\lceil_{\mathbf{S}'} < s\lceil_{\mathbf{S}'}$, contradicting $r' = s_1\lceil_{\mathbf{S}'}$. Thus $(r,r') \in \mathit{trace}(f)$. Reciprocally, if $(r,r') \in \mathit{trace}(f)$, then let $s \in \phi$ be such that $r' = s\lceil_{\mathbf{S}'}$ and $s\lceil_{\mathbf{S}} \leq r$. Then, by minimality of r, we must have $s\lceil_{\mathbf{S}} = r$. The proof of (2) is similar. □

Proposition 14.3.38 *Let $\mathbf{S} = (C,V,P)$, $\mathbf{S}' = (C',V',P')$ and $\mathbf{S}'' = (C'',V'',P'')$ be three sds's. Let $\phi \in D(\mathbf{S} \parallel \mathbf{S}')$, $\phi' \in D(\mathbf{S}' \parallel \mathbf{S}'')$. Then*

$$\phi' \circ \phi = \{s\lceil_{\mathbf{S}\cup\mathbf{S}''} \mid s \in \mathcal{L}(\mathbf{S},\mathbf{S}',\mathbf{S}''), s\lceil_{\mathbf{S}\cup\mathbf{S}'} \in \phi, \text{ and } s\lceil_{\mathbf{S}'\cup\mathbf{S}''} \in \phi'\}$$

where $\mathcal{L}(\mathbf{S},\mathbf{S}',\mathbf{S}'')$ denotes the set of words in $(C \cup V \cup C' \cup V' \cup C'' \cup V'')^\star$ such that two consecutive symbols are not such that one is in $C \cup V$ and the other is in $C'' \cup V''$.

PROOF HINT. One verifies easily that this defines a strategy of $\mathbf{S} \parallel \mathbf{S}''$. Then, by lemma 14.3.37, and by the injectivity of $\lambda s.(s\lceil_{\mathbf{S}}, s\lceil_{\mathbf{S}'})$ (lemma 14.3.24), it is enough to check

$$\{(s\lceil_{\mathbf{S}}, s\lceil_{\mathbf{S}''}) \mid s \in \mathcal{L}(\mathbf{S},\mathbf{S}',\mathbf{S}''), s\lceil_{\mathbf{S}\cup\mathbf{S}'} \in \phi, \text{ and } s\lceil_{\mathbf{S}'\cup\mathbf{S}''} \in \phi'\} =$$
$$\{(p,p'') \mid p = s_1\lceil_{\mathbf{S}}, s_1\lceil_{\mathbf{S}'} = s_2\lceil_{\mathbf{S}'}, \text{ and } p'' = s_2\lceil_{\mathbf{S}''}, \text{ for some } s_1 \in \phi, s_2 \in \phi'\}.$$

Obviously, the left-hand side is included in the right-hand side, taking $s_1 = s\lceil_{\mathbf{S}\cup\mathbf{S}'}$ and $s_2 = s\lceil_{\mathbf{S}'\cup\mathbf{S}''}$. For the other direction we construct s from s_1 and s_2 by replacing every $c'v'$ in s_2 by the corresponding portion $c'c_1v_1 \cdots c_nv_nv'$ of s_1. By construction $s \in \mathcal{L}(\mathbf{S},\mathbf{S}',\mathbf{S}'')$. □

This alternative definition of composition is convenient to establish the symmetric monoidal structure of the category **AFFALGO**.

Definition 14.3.39 (tensor – sds) *Let $\mathbf{S} = (C,V,P)$ and $\mathbf{S}' = (C',V',P')$ be two sds's. We define the sds $\mathbf{S} \otimes \mathbf{S}' = (C'',V'',P'')$ as follows. The sets C'' and V'' are disjoint unions:*

$$\begin{aligned} C'' &= \{c.1 \mid c \in C\} \cup \{c'.2 \mid c' \in C'\} \\ V'' &= \{v.1 \mid v \in V\} \cup \{v'.2 \mid v' \in V'\} \ . \end{aligned}$$

P'' consists of the alternating non-empty positions s which are such that:

$s\lceil_{\mathbf{S}} \in P \cup \{\epsilon\}$ *and* $s\lceil_{\mathbf{S}'} \in P' \cup \{\epsilon\}$, *and*
s has no prefix of the form scv' .

Let $\mathbf{S}_1, \mathbf{S}_2, \mathbf{S}'_1, \mathbf{S}'_2$ be sds's, and let $\phi_1 \in D(\mathbf{S}_1 \parr \mathbf{S}'_1)$ and $\phi_2 \in D(\mathbf{S}_2 \parr \mathbf{S}'_2)$. We define $\phi_1 \otimes \phi_2 \in D((\mathbf{S}_1 \otimes \mathbf{S}_2) \parr (\mathbf{S}'_1 \otimes \mathbf{S}'_2))$ as follows. It consists of the positions of $(\mathbf{S}_1 \otimes \mathbf{S}_2) \parr (\mathbf{S}'_1 \otimes \mathbf{S}'_2)$ whose projections on $\mathbf{S}_1 \cup \mathbf{S}'_1$ and on $\mathbf{S}_2 \cup \mathbf{S}'_2$ are in ϕ_1 and in ϕ_2, respectively.

As for definition 14.3.19, the second constraint in definition 14.3.39 implies that P'' contains no position of the form $sc'v$.

Exercise 14.3.40 *Construct the tensor product along the same lines as in exercise 14.3.23, using the following table of polarities (which is obtained through the encoding of $\mathbf{S} \otimes \mathbf{S}'$ as $(\mathbf{S} \parr \mathbf{S}'^{\perp})^{\perp}$):*

m	m'	(m, m')
•	•	•
•	○	○
○	•	○
○	○	*undefined*

Proposition 14.3.41 *The data of definition 14.3.39 indeed define a functor which, together with the empty sds $(\emptyset, \emptyset, \emptyset)$ as unit, turns* **AFFALGO** *into a symmetric monoidal category.*

PROOF. The coherent isomorphisms are based on the bijective correspondences wich associate, say, a move $m.1$ in $\mathbf{S} \otimes (\mathbf{S}' \otimes \mathbf{S}'')$ to the move $m.1.1$ in $(\mathbf{S} \otimes \mathbf{S}') \otimes \mathbf{S}''$. □

Proposition 14.3.42 *The category* **AFFALGO** *is symmetric monoidal closed.*

PROOF. Loosely, $((\mathbf{S} \otimes \mathbf{S}') \parr \mathbf{S}'')$ and $\mathbf{S} \parr (\mathbf{S}' \parr \mathbf{S}'')$ coincide (up to tags). Given $\phi \in D(\mathbf{S}_1 \parr \mathbf{S})$ and $\psi \in D(\mathbf{S} \parr (\mathbf{S}' \parr \mathbf{S}''))$, in order to turn a position s whose projection on $\mathbf{S}_1 \cup (\mathbf{S}' \parr \mathbf{S}'')$ is in $\psi \circ \phi$ into a position whose projection on $(\mathbf{S}_1 \otimes \mathbf{S}') \cup \mathbf{S}''$ is in the corresponding composed morphism from $\mathbf{S}_1 \otimes \mathbf{S}'$ to $\mathbf{S}''$, we replace every portion $c'v'$ of s by $c'c'v'v'$ (cf. the description of id). □

The category **AFFALGO** is also cartesian.

Definition 14.3.43 (product – sds) *Let $\mathbf{S} = (C, V, P)$ and $\mathbf{S}' = (C', V', P')$ be two sds's. We define $\mathbf{S} \times \mathbf{S}' = (C'', V'', P'')$ as follows:*

- *C'' and V'' are as in definition 14.3.39.*

- $P'' = \{p.1 \mid p \in P\} \cup \{p'.2 \mid p' \in P'\}$ *where* $p.1$ *is a shorthand for the position formed by tagging all the moves of* p *with 1, and similarly for* p'.

Proposition 14.3.44 *The category* **AFFALGO** *is cartesian. The binary products are as specified in definition 14.3.43, and the terminal object is the empty sds* $(\emptyset, \emptyset, \emptyset)$.

PROOF. It is easily seen that $D(\mathbf{S} \times \mathbf{S}')$ is the set-theoretical product of $D(\mathbf{S})$ and $D(\mathbf{S}')$, and that $D^{\perp}(\mathbf{S} \times \mathbf{S}')$ is the disjoint union of $D^{\perp}(\mathbf{S})$ and $D^{\perp}(\mathbf{S}')$. The first projection is the symmetric algorithm (π, inl) where π and inl are the set-theoretical projection and injection, respectively. Similarly, the second projection is (π', inr). If $(f, g) : \mathbf{S} \to \mathbf{S}'$ and $(f', g') : \mathbf{S}' \to \mathbf{S}''$, then $\langle (f, g), (f', g') \rangle$ is defined as $(\langle f, f' \rangle, [g, g'])$, where $\langle \_, \_ \rangle$ and $[\_, \_]$ denote the set-theoretical pairing and copairing. □

Thus, in **AFFALGO**, the empty sds is both the unit of the tensor and a terminal object. It is this property which makes **AFFALGO** a model of affine logic (cf. remark 13.2.23).

Finally, we relate the two categories **ALGO** and **AFFALGO** by an adjunction.

Proposition 14.3.45 *The mapping* **cds** *from sds's to cds's defined in proposition 14.3.3 extends to a functor* $\mathbf{cds} : \mathbf{AFFALGO} \to \mathbf{ALGO}$ *as follows. Let* $\mathbf{S}$ *and* $\mathbf{S}'$ *be two sds's, and let* (f, g) *be a symmetric algorithm from* $\mathbf{S}$ *to* $\mathbf{S}'$. *We define an abstract algorithm* $\mathbf{cds}(f, g) : \mathbf{cds}(\mathbf{S}) \to \mathbf{cds}(\mathbf{S}')$ *as follows:*

$$\mathbf{cds}(f, g)(xq') = \begin{cases} valof\ (x \mid\!\triangleright\, g(q')) & if\ x \triangleright g(q') \\ output\ q'v' & if\ q'v' \in f(x) \end{cases}$$

where we freely confuse $x \in D(\mathbf{S})$ *with the associated state* $\mathbf{cds}(x) \in D(\mathbf{cds}(\mathbf{S}))$. *(As in theorem 14.3.32, the cases in the definition of* $\mathbf{cds}(f, g)$ *are mutually exclusive.)*

PROOF. We only prove $\mathbf{cds}(f' \circ f, g \circ g') = \mathbf{cds}(f', g') \circ \mathbf{cds}(f, g)$. Given x and q'', there are three cases:

- $q''v'' \in f'(f(x))$: Then, obviously:

$$\mathbf{cds}(f' \circ f, g \circ g')(xq'') = output\ q''v'' = (\mathbf{cds}(f', g') \circ \mathbf{cds}(f, g))(xq'').$$

- $x \triangleright g(g'(q''))$: Then $f(x) \triangleright g'(q'')$ by (R). It follows that $\mathbf{cds}(f', g')(f(x)q'') = valof\ q'$, where $q' = f(x) \mid\!\triangleright\, g'(q'')$. We claim that $x \triangleright g(q')$. Suppose not: since $q' \leq g'(q'')$ and $x \triangleright g(g'(q''))$, this would entail $f(x) \triangleleft q'$ by (RS), which

is a contradiction since the contrary holds by definition of q'. Then, by the claim, $cds(f,g)(xq') = valof\ q$, where $q = x \mid\!\triangleright g(q')$. It follows that

$$\mathbf{cds}(f' \circ f, g \circ g')(xq'') = valof\ q = (\mathbf{cds}(f',g') \circ \mathbf{cds}(f,g))(xq'')$$

since $x \mid\!\triangleright g(q') = x \mid\!\triangleright g(g'(q''))$ by lemma 14.3.12.

- $\mathbf{cds}(f' \circ f, g \circ g')(xq'') = \omega$. In particular $q''v'' \notin f'(f(x))$, hence $(\mathbf{cds}(f',g') \circ \mathbf{cds}(f,g))(xq'')$ could only be defined if we had

$$f(x) \triangleright g'(q'') \text{ and } x \triangleright g(f(x) \mid\!\triangleright g'(q'')).$$

But then we would have $x \triangleright g(g'(q''))$, contradicting the assumption. □

We now show that the functor **cds** has a left adjoint.

Definition 14.3.46 (exponential – sds) *Let $\mathbf{M} = (C, V, E, \vdash)$ be a (filiform) cds. The following recursive clauses define a set $P_!$ of alternating words over $C \cup V$:*

$$\begin{array}{ll} rc \in P_! & \text{if } c \in A(state(r)) \\ rcv \in P_! & \text{if } rc \in P_! \text{ and } state(rcv) \in D(M) \end{array}$$

where state is the following function mapping responses (or ϵ) of $P_!$ to states of **M***:*

$$state(\epsilon) = \emptyset \quad state(rcv) = state(r) \cup \{(c,v)\}.$$

The sds $(C, V, P_!)$ is called $!\mathbf{M}$*. We define an abstract algorithm $\eta : \mathbf{M} \to cds(!\mathbf{M})$ by*

$$\eta(x(rc)) = \begin{cases} valof\ c & \text{if } state(r) \subseteq x \text{ and } c \in A(x) \\ output\ (rcv) & \text{if } state(r) \cup \{(c,v)\} \subseteq x\ . \end{cases}$$

(Hence $\eta \bullet x = \{r \mid state(r) \subseteq x\}$.)

Remark 14.3.47 *The reader should compare the definitions of* $\mathbf{sds}(\mathbf{M})$ *(exercise 14.3.7) and of* $!\mathbf{M}$*. In* $\mathbf{sds}(\mathbf{M})$*, positions are made from the proofs of the cells of* **M***, in* $!\mathbf{M}$*, they encode (safety respecting) enumerations of the events contained in the finite states of* **M***. Back to example 14.3.21, it should be now clear that, say, OR_l can be considered as an affine algorithm from $!((\mathbf{B}_\perp)^2)$ to $\mathbf{B}_\perp$.*

Theorem 14.3.48 *The transformation ! described in definition 14.3.46 extends to a functor* $! : \mathbf{ALGO} \to \mathbf{AFFALGO}$ *which is left adjoint to* **cds***, with η as unity. Moreover, the co-Kleisli category associated to the comonad* $! \circ \mathbf{cds} : \mathbf{AFFALGO} \to \mathbf{AFFALGO}$ *is equivalent to* **ALGO***. We shall freely abbreviate* $! \circ \mathbf{cds}$ *as* $!$*.*

PROOF HINT. Let $a \in D(\mathbf{M} \to \mathbf{cds}(\mathbf{S}'))$. We associate a response of $!\mathbf{M}$ ($\mathbf{S}'$ with each event (xq', u) of a as follows:

- If xq' is enabled in a by $(x_1q', \mathit{valof}\, c_1)$ (with $x = x_1 \cup \{(c_1, v_1)\}$ for some v_1), and if sc_1 is the response associated with $(x_1q', \mathit{valof}\ c_1)$, then the response associated with (xq', u) is

$$\begin{array}{ll} sc_1v_1c & \text{if } u = \mathit{valof}\ c \\ sc_1v_1v' & \text{if } u = \mathit{output}\ (q'v')\ . \end{array}$$

- If xq' is enabled in a by $(xq'_1, \mathit{output}\ (q'_1v'_1))$ (with $q' = q'_1v'_1c'$ for some c'), and if sv'_1 is the response associated with $(xq'_1, \mathit{output}\ (q'_1v'_1))$, then the response associated with (xq', u) is

$$\begin{array}{ll} sv'_1c'c & \text{if } u = \mathit{valof}\ c \\ sv'_1c'v' & \text{if } u = \mathit{output}\ (q'v')\ . \end{array}$$

We denote with $\zeta(a)$ the set of responses associated to the events of a in this way. We omit the tedious verification of the two equations:

$$\mathbf{cds}(\zeta(a)) \circ \eta = a \qquad \zeta(\mathit{cds}(\phi) \circ \eta) = \phi.$$

Roughly, the mapping ζ makes the proofs of cells explicit, while **cds** "undoes" the job of ζ by "filtering" input states against the positions of ϕ. The second part of the statement follows from the fact that any object $\mathbf{M}$ of **ALGO** is isomorphic to an object of the form $\mathbf{cds}(\mathbf{S})$ (specifically, to $\mathbf{cds}(\mathit{sds}(\mathbf{M}))$, cf. exercise 14.3.7). □

Theorem 14.3.49 *The category* **ALGO** *is cartesian closed. There are natural isomorphisms in* **AFFALGO** *between* $(!\mathbf{S}) \otimes (!\mathbf{S}')$ *and* $!(\mathbf{S} \times \mathbf{S}')$.

PROOF. The first part of the statement is a consequence of the second part, by proposition 13.2.16. For the second part, notice:

- A cell of $!(\mathbf{S} \times \mathbf{S}')$ is of the form, say, $(c_1.1)(v_1.1)\cdots(c_n.1)$ where $c_1v_1\cdots c_n$ is a query of $\mathbf{S}$, while the corresponding cell of $(!\mathbf{S}) \otimes (!\mathbf{S}')$ is $(c_1v_1\cdots c_n).1$.

- A position $(!\mathbf{S}) \otimes (!\mathbf{S}')$ encodes a shuffling of safety respecting enumerations of a strategy x of $\mathbf{S}$ and of a strategy x' of $\mathbf{S}'$, which is the same as the encoding of a safety respecting enumeration of (x, x'). □

Exercise 14.3.50 *Let* $\mathbf{S} = (C, V, P)$ *be an sds, and consider the sds:*

$$\mathbf{S}^{\uparrow} = (V \cup \{\circ\,|, |\circ\}C, \{\circ p \mid p \in P\}).$$

Show that this operation extends to a functor from **AFFALGO** *to* $\mathbf{AFFALGO}^{op}$ *which is adjoint to itself.*

We end the section with some remarks and comparisons, and by stating two open problems.

• A more abstract setting for sequential algorithms, into which our theory can be embedded, has been developped by Bucciarelli and Ehrhard [BE93]. The basic

idea is to abstract from a cell c by axiomatizing it as a predicate "c is filled" (cf. exercise 14.1.15). Bucciarelli and Ehrhard define sequential structures of the form $(X_\star, X^\star)$ where $X_\star$ plays the role of $D(\mathbf{M})$, and where $X^\star$ is a set of linear functions from $X_\star$ to $\mathbf{O}$. Their morphisms are defined in the style of exercise 14.2.7.

• Sequential algorithms bear a striking similarity with the oracles that Kleene has developed in his late works on the semantics of higher-order recursion theory. In a series of papers [Kle78, Kle80, Kle82, Kle85], he developed up to rank 3 a theory of unimonotonous functions, which are closely related to sequential algorithms (see [Buc93] for a precise correspondence). He lacked synthetic tools to develop a theory at all ranks.

• Berry-Curien's sequential algorithms, as well as Ehrhard's hypercoherences (cf. section 13.3), yield standard models of PCF. Indeed, it is easily checked that **ALGO** and **HCoh** are cpo-enriched CCC's, and we know from theorem 6.5.4 that the standard interpretations of all first-order constants of PCF are sequential and strongly stable. Ehrhard [Ehr96] has proved that the hypercoherence model of PCF is actually the extensional collapse of the model of sequential algorithms (cf. exercise 4.5.6). This quite difficult result relies on the following steps:

1. For any hypercoherence (E, Γ), any $A \in \mathcal{C}(E)$ and any $n \geq \sharp A$, there exists $G \in \mathcal{C}(\omega_\perp^n)$ and a strongly stable function $g : \omega_\perp^n \to E$ such that $g(G) = A$.

2. Every compact element y of the model at any type τ is 2- PCF-definable, which means that there exists a term $x_1 : \sigma_1, \ldots, x_n : \sigma_n \vdash M : \tau$, with $\sigma_1, \ldots, \sigma_n$ of rank at most 2 (cf. definition 4.5.10), such that $y = [\![M]\!](x_1, \ldots, x_n)$ for some $x_1, \ldots, x_n$.

3. The 2-PCF-definability allows to prove the surjectivity (hence the functionality, cf. section 4.5) of the logical relation between the model of sequential algorithms and the hypercoherence model generated by the identity at ground types.

Exercise 14.3.51 (the semantics as an interpreter) ** Show that the interpretation function of* PCF *in* **ALGO** *is actually computed by a sequential algorithm (representing* PCF *terms through a cds, like* **LAMBDA**, *cf. example 14.1.6).*

We mention the following two pen problems.

1. Denoting simply by $[\![]\!]$ the "natural" algorithm that computes the semantic function $[\![]\!]$ (cf. exercise 14.3.51) and denoting likewise by BT the minimum algorithm computing BT, does the equality (of algorithms) $[\![]\!] = [\![]\!] \circ BT$ hold? In other words, does the semantic evaluation respect the indications of sequentiality provided by the syntax itself?

2. If a and a' are two definable algorithms such that $a < a'$, can we find N and N' such that $N < N'$ (in the sense of definition 2.3.1), $a = [\![N]\!]$, and $a' = [\![N']\!]$? In other words, does the order on algorithms reflect the syntactic ordering?

14.4 Full Abstraction for PCF + *catch* *

In this section we extend the language PCF with an operation *catch*, which is inspired from the constructions "catch" and "throw" found in several dialects of LISP. The model of sequential algorithms is fully abstract for this extension of PCF, called SPCF. This stands in sharp contrast with the situation of this model with respect to PCF (cf. section 6.5). The material of this section is adapted from [CF92, CCF94].

Observing sequential algorithms. Before we come to the proper subject of this section, we present a third characterisation of sequential algorithms, in addition to the descriptions as states and as abstract algorithms. Although sequential algorithms are not functions in the ordinary sense, it would be useful to be able to compare two algorithms by applying them to (extended) inputs. The explicit consideration of an error element allows this.

Definition 14.4.1 (observable state) *We assume once and for all that there exists a reserved, non-empty set Err of* error values, *which is disjoint from any set V of values of any cds $\mathbf{M} = (C, V, E, \vdash)$. We stress this by calling an element of V a* proper value. *Unless stated otherwise explicitly, we assume that Err is a singleton, and we write $Err = \{\mathsf{e}\}$.*

Given a cds $\mathbf{M} = (C, V, E, \vdash)$, we call observable state *of $\mathbf{M}$ a set x of pairs (c, w), where either $(c, w) \in E$ or $w \in Err$, satisfying the conditions that define a state of a cds. The set of observable states of $\mathbf{M}$ is denoted $D_{\mathcal{O}}(\mathbf{M})$. Note that states are a fortiori observable states: this may be stressed by calling the states of $\mathbf{M}$* error free. *With each observable state x, we associate an error-free state x_{-e} defined by*

$$x_{-e} = x \cap E.$$

This definition implies that enablings are not allowed to contain error values, because the enabling relation is part of the structure of a cds, which we did not change. In the tree representation of an observable state, error values can occur only at the leaves. As an example, the cds $\omega_\perp$ has (up to isomorphism) $\omega_\perp \cup Err$ as its set of extended states. Next we explain how sequential algorithms act on observable states.

Definition 14.4.2 *Let $\mathbf{M}$ and $\mathbf{M}'$ be two cds's. Every sequential algorithm $a : \mathbf{M} \to \mathbf{M}'$ determines an* observable input-output *function from $D_{\mathcal{O}}(\mathbf{M})$ to $D_{\mathcal{O}}(\mathbf{M}')$, defined by*

$$\begin{aligned} a \bullet x \;=\; & \{(c', \mathit{output}\ v') \mid \exists y \leq x \;\; (yc', \mathit{output}\ v') \in a\} \cup \\ & \{(c', \mathsf{e}) \mid \exists y \leq x \;\; (yc', \mathit{valof}\ c) \in a \text{ and } (c, \mathsf{e}) \in x\}\ . \end{aligned}$$

On error free states, this definition agrees with the definition of $a \bullet x$ given in definition 14.2.1. The second component of the union is only "active" when the input contains error values, and "implements" a propagation of these values to the output.

Lemma 14.4.3 *The function* $\lambda x.(a \bullet x) : D_{\mathcal{O}}(\mathbf{M}) \to D_{\mathcal{O}}(\mathbf{M}')$ *of definition 14.4.2 is well defined and continuous.*

PROOF. Continuity obviously follows from the definition. We have to show that $a \bullet x$ is (i) consistent, and (ii) safe. We claim that once c' is fixed, then there is at most one y which ensures that c' is filled in $a \bullet x$. Property (i) immediately follows from the claim. We prove the claim by contradiction. There are three cases:

1. $(y_1 c', \mathit{output}\ v'_1), (y_2 c', \mathit{output}\ v'_2) \in a$. Then $y_1 = y_2$ by proposition 14.2.4, contradicting the assumption.

2. $(y_1 c', \mathit{output}\ v'_1) \in a$, $(y_2 c', \mathit{valof}\ c_2) \in a$ and $(c_2, \mathsf{e}) \in x$. By proposition 14.2.4, $y_2 < y_1$, and moreover c_2 must be filled in y_1, and hence in x, with a proper value, since y_2 is error free. This contradicts the consistency of x, since we also assumed $(c_1, \mathsf{e}) \in x$.

3. $(y_1 c', \mathit{valof}\ c_1) \in a$, $(c_1, \mathsf{e}) \in x$ and $(y_2 c', \mathit{valof}\ c_2) \in a$, $(c_2, \mathsf{e}) \in x$. Then, say, $y_1 < y_2$, and the reasoning is the same as in case 2.

Next we show safety. From the above analysis, it follows that $a \bullet x$ is the disjoint union of $\{(c', \mathit{output}\ v') \mid \exists y \leq x \ \ (yc', \mathit{output}\ v') \in a\}$ and $\{(c', \mathsf{e}) \mid \exists y \leq x \ \ (yc', \mathit{valof}\ c) \in a \text{ and } (c, \mathsf{e}) \in x\}$. The first of these sets is $a \bullet (x_{-e})$, which is a state by proposition 14.2.4. If c' is filled in the second set, then by definition $(yc', \mathit{valof}\ c) \in a$ for some y, which entails $c' \in A(a \bullet y)$ and $c' \in E(a \bullet x)$. □

The following proposition shows what the consideration of errors is useful for.

Proposition 14.4.4 *Let* $\mathbf{M}$ *and* $\mathbf{M}'$ *be two cds's. If* $a \bullet x \leq a' \bullet x$ *for all* $x \in D_{\mathcal{O}}(\mathbf{M})$, *then* $a \leq a'$.

PROOF. The proof is by contradiction. Let yc' be a minimal cell of $\mathbf{M} \to \mathbf{M}'$ such that yc' is filled in a, and is either not filled, or filled with a different value in a'. We shall call witness an observable z such that $a.z \not\leq a'.z$. There are two cases:

1. If $(yc', \mathit{output}\ v') \in a$, then $(c', v') \in a \bullet y$. If $(c', \mathit{output}\ v') \notin a' \bullet y$, then y is a witness. If $(c', \mathit{output}\ v') \in a' \bullet y$, then, since y is error free, there exists $z \leq y$ such that $(zc', \mathit{output}\ v') \in a'$.

2. If $(yc', \mathit{valof}\ c) \in a$, then $(c', \mathsf{e}) \in a \bullet (y \cup \{(c, \mathsf{e})\})$. If $(c', \mathsf{e}) \notin a' \bullet (y \cup \{(c, \mathsf{e})\})$, then $y \cup \{(c, \mathsf{e})\}$ is a witness. If $(c', \mathsf{e}) \in a' \bullet (y \cup \{(c, \mathsf{e})\})$, then, by definition of the observable input-output function:

$$\exists z \leq y \cup \{(c, \mathsf{e})\} \ \ (zc', \mathit{valof}\ c_1) \in a') \text{ and } (c_1, \mathsf{e}) \in y \cup \{(c, \mathsf{e})\}.$$

Then $z \leq y$, since z is error free and $z \leq y \cup \{(c, \mathsf{e})\}$. Also, $(c_1, \mathsf{e}) \in y \cup \{(c, \mathsf{e})\}$ implies $c_1 = c$, since y is error free.

Both cases 1 and 2 reduce to the situation where $(yc', u) \in a$ and $(zc', u) \in a'$, for some $z \leq y$ and for some u. This forces $z < y$, since by assumption $(yc', u) \notin a'$. Also, $(zc', u) \in a$, by the minimality assumption. But the conjunction of $(yc', u) \in a$, $(zc', u) \in a$, and $y < z$ is excluded by proposition 14.2.4, which forces $u = \mathit{valof}\ c$ and $y <_c z$ for some c, and precludes $(zc', \mathit{valof}\ c)$ to be an event. □

We gather more material in exercises 14.4.6, 14.4.7, and 14.4.8.

Exercise 14.4.5 *Let* $\mathbf{M}$ *be a cds Let* $a \in D_{\mathcal{O}}(\mathbf{M} \to \omega_\perp)$. *Show the following properties, for any cell* $x? \in E(a)$*:*

$$\begin{array}{lll} (1) & a{\bullet}x = \{(?, n)\} & \Leftrightarrow \quad (x?, \mathit{output}\ n) \in a \\ (2) & (a{\bullet}x = \perp \textit{ and } a{\bullet}(x \cup \{(c, \mathsf{e})\}) = \{(?, \mathsf{e})\}) & \Leftrightarrow \quad (x?, \mathit{valof}\ c) \in a \\ (3) & a{\bullet}x = \{(?, \mathsf{e})\} & \Leftrightarrow \quad (x?, \mathsf{e}) \in a\ . \end{array}$$

Generalise these properties for $a \in D_{\mathcal{O}}(\mathbf{M}_1 \to \cdots \to \mathbf{M}_n \to \omega_\perp)$. *Hint: these properties are variations of proposition 14.2.4.*

Exercise 14.4.6 *Let* $\mathbf{M}$ *and* $\mathbf{M}'$ *be two cds's. The observable input-output function of an observable algorithm, that is, of an observable state of* $\mathbf{M} \to \mathbf{M}'$*, is defined as follows:*

$$\begin{array}{rcl} a{\bullet}x & = & \{(c', \mathit{output}\ v') \mid \exists\, y \leq x\ \ (yc', \mathit{output}\ v') \in a\} \cup \\ & & \{(c', \mathsf{e}) \mid \exists\, y \leq x\ \ (yc', \mathsf{e}) \in a\} \cup \\ & & \{(c', \mathsf{e}) \mid \exists\, y \leq x\ \ (yc', \mathit{valof}\ c) \in a \textit{ and } (c, \mathsf{e}) \in x\}\ . \end{array}$$

(Here we do not assume that Err is a signleton, and e *is used to denote a generic element of Err.) Show that the statement of proposition 14.4.4 fails for observable algoithms if* $Err = \{\mathsf{e}\}$*, and holds if Err contains at least two elements. Hint: Consider* $a = \{(\emptyset?, \mathit{valof}\ ?)\}$ *and* $a' = \{(\emptyset?, \mathsf{e})\}$*, between, say, two flat domains.*

Exercise 14.4.7 *Let* $\mathbf{M}$ *and* $\mathbf{M}'$ *be two cds's. (1) Show that there exists an order-isomorphism between* $D(\mathbf{M} \to \mathbf{M}')$ *and the pointwise ordered set of functions* h *from* $D_{\mathcal{O}}(\mathbf{M})$ *to* $D_{\mathcal{O}}(\mathbf{M}')$ *which are:*

- *error-sensitive: For any* x *and* c' *such that* $c' \in A(h(x))$ *and* $c' \in F(h(z))$ *for some* $z > x$*, there exists* $c \in A(x)$*, called sequentiality index, such that*

$$\begin{array}{l} \forall\, y > x\ \ (h(x) <_{c'} h(y)\ \Rightarrow\ x <_c y) \\ \forall\, \mathsf{e} \in \mathit{Err}\ \ h(x \cup \{(c, \mathsf{e})\}) = h(x) \cup \{(c', \mathsf{e})\}\ ; \end{array}$$

- *error-reflecting: For any* c'*,* e *and* y*, if* $(c', \mathsf{e}) \in h(y)$*, then* h *has a sequentiality index* c *at* (x, c') *for some* $x < y$*, and* $(c, \mathsf{e}) \in y$.

(2) Show that sequentiality indexes of error-sensitive functions are unique (for fixed x *and* c'*). (3) Show that this isomorphism extends to an isomorphism from* $D_{\mathcal{O}}(\mathbf{M} \to \mathbf{M}')$ *to the set of pointwise ordered error-sensitive functions from* $D_{\mathcal{O}}(\mathbf{M})$ *to* $D_{\mathcal{O}}(\mathbf{M}')$*. Hints: use proposition 14.4.4; for the surjectivity of the mapping from* a *to* $\lambda x.(a{\bullet}x)$*, proceed as in the proof of proposition 14.2.9.*

$$\frac{\Gamma \vdash M : \sigma}{\Gamma \vdash \mathit{catch}(M) : \iota} \qquad \frac{}{\Gamma \vdash \mathsf{e} : \iota}$$

Figure 14.3: The additional constants of SPCF and of SPCF(Err)

Exercise 14.4.8 *Show that the category whose objects are cds's and whose arrows are observable algorithms (cf. exercise 14.4.6) is cartesian closed. Hint: use the characterisation given in exercise 14.4.7.*

We come now to the proper subject of this section. We extend PCF with a family of unary operators *catch* at each PCF type σ. The resulting extended language is called SPCF. Just as in the semantics, it may be convenient to introduce explicit errors in the syntax. We thus occasionally work with SPCF(Err), which is SPCF plus constants $\mathsf{e} \in Err$ of basic type, which are interpreted using error values with the same name. The typing rules for the constants of SPCF and of SPCF(Err) are summarized in figure 14.3. As for PCF, a program is a closed term of basic type, and Ω is an additional constant such that $[\![\vdash \Omega]\!] = \bot$, at each basic type.

As for PCF, we use the same name for the operator *catch* and for its interpretation in the category **ALGO**, i.e., we write

$$[\![\Gamma \vdash \mathit{catch}(M) : \iota]\!] = \mathit{catch}^\sigma \circ [\![\Gamma \vdash M : \sigma]\!]$$

where the right-hand side *catch* is given in figure 14.4. In this figure, and in the rest of this section, we shall adopt the following conventions:

- $\vec{\bot}?$ denotes the initial cell $\bot \ldots \bot?$ of (the interpretation of) any type.
- We freely switch between curryied and uncurryied algorithms (for example, in the third line of figure 14.4, $(\vec{\bot}?).i$ is a cell of $\sigma_1 \times \cdots \times \sigma_m$).

The algorithm *catch* asks its unique argument about the value of its initial cell ("what do you do if you know nothing about your argument"?). If this cell is filled with *output* n, i.e., if the argument is the constant n, then *catch* outputs $m + n$. If instead the argument asks about the initial cell of its ith argument, then *catch* outputs $i - 1$.

Operational semantics. We next describe the operational semantics of SPCF. It is convenient to use *evaluation* contexts (cf. section 8.5). They have a unique hole, where "the next reduction takes place". They are declared as follows:

$$E ::= [\,] \mid fE \mid EM \mid \mathit{catch}(\lambda \vec{x}.E)$$

where $f \in \{\mathit{succ}, \mathit{pred}, \mathit{zero}?, \mathit{cond}\}$ and where $\vec{x}$ abbreviates $x_1 \ldots x_n$ (the intended subscripts may vary). In particular, n may be 0, i.e., $\mathit{catch}(E)$ is an evaluation context. We denote by $E[M]$ the result of filling the hole of E with M.

$$
\begin{array}{lcll}
catch^{(\sigma_1\to\cdots\to\sigma_m\to\iota)\to\iota} & = & \{(\bot?, valof\ \vec{\bot}?), & \\
 & & (\{(\vec{\bot}?, output\ n)\}?, output\ m+n), & (n\in\omega)\\
 & & (\{(\vec{\bot}?, valof\ (\vec{\bot}?).i)\}?, output\ i-1\} & (1\le i\le m)
\end{array}
$$

Figure 14.4: Interpretation of *catch* in **ALGO**

$$\frac{M\to M'}{E[M]\mapsto E[M']}$$

$$
\begin{array}{lcll}
catch(\lambda x_1\ldots x_m.E[x_i]) & \to & i-1 & (i\le m, x_i \text{ free in } E[x_i])\\
catch(\lambda x_1\ldots x_m.n) & \to & m+n & \\
catch(f) & \to & 0 & (f\in\{succ, pred, zero?, cond\})
\end{array}
$$

Figure 14.5: Operational semantics for SPCF

The rules are given at two levels: there are axioms of the form $M\to M'$, and evaluation steps of the form $E[M]\mapsto E[M']$. The axioms are those for PCF plus three axioms for *catch*. The evaluation rule and the additional axioms are given in figure 14.5. The *catch* rules deserve some explanation. The constant *catch* is a so-called control operator. If the argument of *catch* is strict in its *ith* argument, then the value $i-1$ is returned. It the argument f of *catch* is a constant function, then *catch* returns that constant (plus the arity of f, since the outputs $0,\ldots,m-1$ have a special meaning in this context). The reader may check that $catch(add_l)\mapsto^* 0$ and $catch(add_r)\mapsto^* 1$, where add_l and add_r are the PCF terms denoting the left and right addition algorithms (cf. exercise 6.3.3).

We extend the operational semantics to SPCF(Err) (cf. figure 14.3) by adding a second evaluation rule:

$$E[\mathsf{e}]\mapsto \mathsf{e}.$$

Exercise 14.4.9 *Show that the following properties hold:*

- *If E, E' are evaluation contexts, then $E[E']$ is an evaluation context.*
- *If $M\mapsto M'\neq \mathsf{e}$, then $E[M]\mapsto E[M']$; if $M\mapsto \mathsf{e}$, then $E[M]\mapsto \mathsf{e}$.*

Exercise 14.4.10 (Soundness) *Show that if $M\mapsto M'$, then $[\![M]\!]=[\![M']\!]$.*

Exercise 14.4.11 * *Show the following properties.*

(1) If $M\Omega\cdots\Omega \mapsto^\star \mathsf{e}$ *, then* $catch(M) \mapsto^\star \mathsf{e}$.

(2) If $catch(M) \mapsto^\star \mathsf{e}$ *, then* $M\Omega\cdots\Omega \mapsto^\star \mathsf{e}$.

(3) If $M\Omega\cdots\Omega \mapsto^\star n$*, where* M *is of type* $\sigma_1 \to \cdots \sigma_m \to \iota$*, then* $catch(M) \mapsto^\star n+m$.

(4) If $MQ_1\cdots Q_m \mapsto^\star \mathsf{e}$ $(m \geq 1)$ *, where all* Q_j *'s are* Ω *, except* $Q_i = \lambda\vec{y}.\mathsf{e}$ *, and if* $M\Omega\cdots\Omega \mapsto^\star \mathsf{e}$ *does not hold, then* $catch(M) \mapsto^\star i-1$.

Exercise 14.4.12 (adequacy) ** (1) Let* M *be a* SPCF(Err) *program. Show that the following equivalences hold:*

$$\begin{array}{lcl} [\![M]\!] = n & \Leftrightarrow & M \mapsto^\star n \\ [\![M]\!] = \mathsf{e} & \Leftrightarrow & M \mapsto^\star \mathsf{e}\,. \end{array}$$

(2) Let M *be a* SPCF *program. Show that the following equivalence holds:*

$$[\![M]\!] = n \quad \Leftrightarrow \quad M \mapsto^\star n.$$

Hints: for (1), adapt the proof of theorem 6.3.6, and use exercise 14.4.11; for (2), use (1) and the observation that if M *is an* SPCF *term and* $M \mapsto^\star n$ *in* SPCF(Err)*, then* $M \mapsto^\star n$ *in* SPCF.

Full abstraction. We first prove, by a semantic argument, that SPCF is a sequential language (cf. section 6.4).

Proposition 14.4.13 *If* C *is a* SPCF *program context with several holes, if*

$$[\![\vdash C[\Omega,\ldots,\Omega]]\!] = \bot \quad \textit{and} \quad \exists M_1,\ldots,M_n \ [\![\vdash C[M_1,\ldots,M_n]]\!] \neq \bot$$

then there exists an i*, called sequentiality index, such that:*

$$\forall\, N_1,\ldots,N_{i-1},N_{i+1},\ldots,N_n \quad \left\{ \begin{array}{l} [\![\vdash C[N_1,\ldots,N_{i-1},\Omega,N_{i+1},\ldots,N_n]]\!] = \bot \\ [\![\vdash C[N_1,\ldots,N_{i-1},\mathsf{e},N_{i+1},\ldots,N_n]]\!] = \{(?,\mathsf{e})\}\,. \end{array} \right.$$

(Here, $M_1,\ldots,M_n,N_1,\ldots,N_n$ *are ranging over closed* SPCF *terms.)*

PROOF. Let $a = [\![\vdash \lambda x_1\cdots x_n.C[x_1,\ldots,x_n]]\!]$. We have, by the validity of β, for all closed $M_1,\ldots,M_n$:

$$[\![\vdash C[M_1,\ldots,M_n]]\!] = a\bullet[\![\vdash M_1]\!]\bullet\ldots\bullet[\![\vdash M_n]\!].$$

We have:

$$\begin{array}{lcl} \exists M_1,\ldots,M_n \ [\![\vdash C[M_1,\ldots,M_n]]\!] \neq \bot & \Rightarrow & a \neq \emptyset \\ [\![\vdash C[\Omega,\ldots,\Omega]]\!] = \bot & \Rightarrow & \nexists n \ ((\vec{\bot}?, \mathit{output}\ n) \in a)\,. \end{array}$$

Hence $(\vec{\bot}?, \mathit{valof}\ (\vec{\bot}?).i) \in a$ for some i, and the conclusion follows by the definition of the composition of sequential algorithms. □

We recall the (semantic formulation of the) full abstraction property, which we want to prove for **ALGO** with respect to SPCF:

$$\forall M, N \quad (\forall C \ \ [\![\vdash C[M]]\!] \leq [\![\vdash C[N]]\!]) \Leftrightarrow [\![M]\!] \leq [\![N]\!]$$

where C ranges over program contexts. We have been used to link full abstraction and definability, cf. section 6.4. However, proposition 6.4.6 applies to an order-extensional model. By proposition 14.4.4, this is fine for SPCF(Err) (see exercise 14.4.17), but not for SPCF. Fortunately, we can use contexts other than the applicative ones to show full abstraction for SPCF from definability.

Lemma 14.4.14 *Let* $\mathbf{M}$ *be a cds, and let* $x, y \in D(\mathbf{M})$. *If* $x \not\leq y$, *then there exists a finite sequential algorithm* $a : \mathbf{M} \to \omega_\perp$ *such that* $a\bullet x \not\leq a\bullet y$.

PROOF. Let c be a minimal cell such that $(c, v) \in F(x)$ and either $c \notin F(x)$ or c is filled in y with a different value. Let $(c_0, v_0), \ldots, (c_n, v_n)$ be the proof of c in x. Define

$$x_0 = \emptyset, \ldots, x_n = x_{n-1} \cup \{(c_{n-1}, v_{n-1})\}, x_{n+1} = x_n \cup \{(c, v)\}.$$

Then we set

$$a = \{(x_0?, \mathit{valof}\ c_0), \ldots, (x_n?, \mathit{valof}\ c_n), (x_{n+1}?, \mathit{output}\ 1)\}.$$

We have $(?, 1) \in a\bullet x$ and $(?, 1) \notin a\bullet y$, hence $a\bullet x \not\leq a\bullet y$. □

Theorem 14.4.15 (definability for SPCF**)** *Let* τ *be a* PCF *type. Any finite state* d *of the cds* $\mathbf{M}^\tau$ *interpreting* τ *in* **ALGO** *is definable.*

PROOF. Let $B \in \mathcal{K}(D(\mathbf{M}^{\tau_1 \to \ldots \to \tau_k \to \kappa}))$. We take $\kappa = \iota$ without loss of generality. Let

$$\emptyset = B_0 \prec \cdots \prec B^{\alpha-1} \prec_{\vec{b}^\alpha ?} B^\alpha \prec \cdots \prec B^\beta = B$$

be a chain from $\emptyset$ to B, where $\vec{b}^\alpha?$ is an abbreviation for $b_1^\alpha \ldots b_k^\alpha?$. We shall associate with B^α a term $x_1 : \tau_1, \ldots, x_k : \tau_k \vdash P^\alpha : \iota$, as well as an injection i^α from $A(B^\alpha) \cap F(B)$ into the set of occurrences of Ω in P^α. The construction is by a lexicographic induction on $(\mathit{rank}(\tau), \sharp B)$ (cf. definition 4.5.10).

(Base case) $P^\emptyset = \Omega$.
The only initial cell of $\mathbf{M}^{\tau_1 \to \ldots \to \tau_k \to \kappa}$ is $\vec{\perp}?$, and we associate with it the unique occurrence of Ω in $P^\emptyset$.

(Induction case) Let $P^{\alpha-1} = C[\Omega]$, where C is the context corresponding to $\vec{b}^\alpha?$ (that is, $u = i^{\alpha-1}(\vec{b}^\alpha?)$, where u is the unique occurrence of the unique hole [ ] of C). We distinguish the following cases:

1. $(\vec{b}^\alpha?, \mathit{output}\ n) \in B$. Then we set $P^\alpha = C[n]$.

2. $(\vec{b}^{\alpha}?, \mathit{valof}\ c.i) \in B$, for some $c \in A(b_i^{\alpha})$. Any cell in $(A(B^{\alpha+1}) \cap F(B)) \setminus A(B^{\alpha})$ has the form $b_1 \dots b_{i-1} b b_{i+1} \dots b_k?$, where $b = b_i \cup \{(c,u)\}$ for some u, and is thus determined by this u. Let $\tau_i = \sigma_1 \to \dots \to \sigma_l \to \iota$ and $c = a_1 \cdots a_l?$. The set U^{α} of the u's can be decomposed as follows:

$$U^{\alpha} = \begin{cases} \{\mathit{output}\ n_1, \dots, \mathit{output}\ n_{q_0}\} \cup \\ \{\mathit{valof}\ c_{11}.1, \dots, \mathit{valof}\ c_{1q_1}.1\} \cup \dots \cup \\ \{\mathit{valof}\ c_{j1}.j, \dots, \mathit{valof}\ c_{jq_j}.j\} \cup \dots \cup \\ \{\mathit{valof}\ c_{l1}.l, \dots, \mathit{valof}\ c_{lq_l}.l\}\ . \end{cases}$$

We further analyze the type $\sigma_j = \rho_1 \to \dots \to \rho_p \to \kappa$. Let us consider an auxiliary type $\sigma'_j = \rho_1 \to \dots \to \rho_p \to \kappa \to \dots \to \kappa$, of $p + q_j$ arguments. Cells (and observable states) can be injected from $\mathbf{M}^{\sigma_j}$ to $\mathbf{M}^{\sigma'_j}$ in the following way: a cell $d = z_1 \dots z_p?$ becomes $\hat{d} = z_1 \dots z_p \perp \dots \perp?$, and an observable state a becomes $\hat{a} = \{(\hat{d}, u) \mid (d,u) \in a\}$. We set

$$a'_j = \widehat{a_j} \cup \{(\widehat{c_{j1}}, \mathit{valof}\ ?.(p+1))\} \cup \dots \cup \{(\widehat{c_{jq_j}}, \mathit{valof}\ ?.(p+q_j))\}.$$

In particular, if $q_j = 0$, then $a'_j = a_j$. Since $\mathit{rank}(\sigma'_j) = \mathit{rank}(\sigma_j) < \mathit{rank}(\tau))$, we can apply induction and get terms $M'_1, \dots, M'_l$ defining $a'_1, \dots, a'_l$. We set, for all $j \le l$:

$$M_j = \lambda z_1 \dots z_p . M'_j z_1 \dots z_p y_{j1} \cdots y_{jq_j}$$

where $y_{j1}, \dots, y_{jq_j}$ are fresh and distinct variable names. Finally we define (using the syntax of section 6.4):

$$R = \mathit{case}\ \mathit{catch}(S)\ [F]$$

where

$$S = \lambda \vec{y}_1 \cdots \vec{y}_l . x_i M_1 \dots M_l \quad (\vec{y}_j \text{ stands for } y_{j1} \cdots y_{jq_j})\ .$$

and where F is the partial function that places an Ω at branches matching the elements of U^{α}. More precisely:

$$F(r) = \begin{cases} \Omega & \text{if } r < q_1 + \dots + q_l \\ \Omega & \text{if } r = q_1 + \dots + q_l + n_1 \\ \vdots & \\ \Omega & \text{if } r = q_1 + \dots + q_l + n_{q_0} \\ \text{undefined} & \text{otherwise} \end{cases}$$

To keep notation readable, we shall write

$$\begin{array}{ll} \text{“}\mathit{valof}\ c_{jm}.j\text{”} & \text{instead of } q_1 + \dots + q_{j-1} + m - 1 \\ \text{“}\mathit{output}\ n_m\text{”} & \text{instead of } q_1 + \dots + q_l + n_m\ . \end{array}$$

We set $P^{\alpha+1} = C[R]$.

The proof goes via two successive claims. The definition of i^{α} for $\alpha > 0$ will be given in the proof of the second claim.

Claim 1. Let $c = a_1 \cdots a_l?, n_m$ $(m \leq q_0)$, and c_{jm} $(m \leq q_j)$ be as above, and let $d_1, \ldots, d_k$ be observable states such that $c \in E(d_i)$. The following properties hold.

$$\begin{array}{lll} (1) & (c, \mathit{output}\ n_m) \in d_i & \Leftrightarrow \quad (\vec{\perp}?, \mathit{output}\ n_m) \in [\![S]\!]\bullet(d_1, \ldots, d_k) \\ (2) & (c, \mathit{valof}\ c_{jm}.j) \in d_i & \Leftrightarrow \quad (\vec{\perp}?, \mathit{valof}\ ?.p + m) \in [\![S]\!]\bullet(d_1, \ldots, d_k) \\ (3) & (c, \mathsf{e}) \in d_i & \Leftrightarrow \quad (\vec{\perp}?, \mathsf{e}) \in [\![S]\!]\bullet(d_1, \ldots, d_k)\ . \end{array}$$

For all observable states $d_1, \ldots, d_k, e_{11}, \ldots e_{lq_l}$, we have, by definition of S:

$$[\![S]\!]\bullet(d_1, \ldots, d_k)\bullet\vec{e}_1\bullet\cdots\bullet\vec{e}_l \;=\; d_i\bullet([\![M_1]\!]\bullet\vec{e}_1)\bullet\cdots\bullet([\![M_l]\!]\bullet\vec{e}_l).$$

Thus we are led to examine the $[\![M_j]\!]\bullet\vec{e}_j$'s. For all observable states $z_1, \ldots, z_p$, we have, by definition of M_j, M'_j, and a'_j:

$$\begin{array}{rcll} [\![M_j]\!]\bullet\vec{e}_j\bullet z_1\bullet\cdots\bullet z_p & = & [\![M'_j]\!]\bullet z_1\bullet\cdots\bullet z_p\bullet\vec{e}_j & \\ & = & a'_j\bullet z_1\bullet\cdots\bullet z_p\bullet\vec{e}_j & \\ & = & \begin{cases} \{(?, \mathsf{e})\} & \text{if } \exists\, m \leq q_j \ \ (c_{jm} \leq \vec{z} \text{ and } e_{jm} = \{(?, \mathsf{e})\}) \\ a_j\bullet z_1\bullet\cdots\bullet z_p & \text{otherwise} \end{cases} & \end{array}$$

where $c_{jm} \leq \vec{z}$ is a shorthand for

$$c_{jm} = z_{1m} \cdots z_{pm}? \text{ and } (z_{1m} \leq z_1, \ldots, z_{pm} \leq z_p).$$

We single out two consequences of this computation. First, setting $\vec{e}_j = \vec{\perp}$, we get $[\![M_j]\!]\bullet\vec{\perp} = a_j$, hence

$$(\dagger) \quad [\![S]\!]\bullet(d_1, \ldots, d_k)\bullet\vec{\perp}\bullet\cdots\bullet\vec{\perp} = d_i\bullet a_1\bullet\cdots\bullet a_l\ .$$

Second, if $e_{jm} = \{(?, \mathsf{e})\}$ and $e_{j1} = \cdots = e_{j(m-1)} = e_{j(m+1)} = \cdots e_{jq_j} = \perp$, then

$$(\ddagger) \quad [\![M_j]\!]\bullet\vec{e}_j = a_j \cup \{(c_{jm}, \mathsf{e})\}.$$

We now prove property (1) of the claim.

$$\begin{array}{lll} & (\vec{\perp}?, \mathit{output}\ n_m) \in [\![S]\!]\bullet(d_1, \ldots, d_k) & \\ \Leftrightarrow & [\![S]\!]\bullet(d_1, \ldots, d_k)\bullet\perp\bullet\cdots\bullet\perp = \{(?, n_m)\} & \\ \Leftrightarrow & d_i\bullet a_1 \cdots\bullet a_l = \{(?, n_m)\} & \text{(by } (\dagger)\text{)} \\ \Leftrightarrow & (c, \mathit{output}\ n_m) \in d_i & \text{(by exercise 14.4.5 (1))}\ . \end{array}$$

Properties (2) and (3) are proved much in the same way, making use of $(\dagger)$, $(\ddagger)$ and exercise 14.4.5 (2), and of exercise 14.4.5 (3), respectively.

Claim 2. For any $b_1 \ldots b_k? \in A(B^\alpha) \cap F(B)$ (abbreviated as $\vec{b}?$), for any observable states $d_1, \ldots, d_k$, and for any $x_1 : \tau_1, \ldots, x_k : \tau_k \vdash N : \iota$:

$$[\![C[N]]\!]\bullet(d_1, \ldots, d_k) \;=\; \begin{cases} [\![N]\!]\bullet(d_1, \ldots, d_k) & \text{if } b_i \leq d_i \text{ for all } i \leq k \\ B^\alpha\bullet d_1\bullet\cdots\bullet d_k & \text{otherwise} \end{cases}$$

where C is the context associated with $\vec{b}?$.

We first show that the statement follows from claim 2. More precisely, we prove:

$$[\![\lambda x_1 \ldots x_k.P^\alpha]\!] = B^\alpha.$$

By proposition 14.4.4 it is enough to show, for all observable $d_1, \ldots, d_k$:

$$[\![C[\Omega]]\!]\bullet(d_1, \ldots, d_k) \quad = \quad B^\alpha \bullet d_1 \bullet \cdots \bullet d_k.$$

If $b_i \leq d_i$ for all i, then

$$\begin{array}{lcll} [\![C[\Omega]\bullet(d_1, \ldots, d_k)]\!] & = & \bot\bullet(d_1, \ldots, d_k) & \text{(by the claim)} \\ & = & \bot & \\ & = & B^\alpha\bullet d_1\bullet\cdots\bullet d_k & \text{(by exercise 14.4.5, since } \vec{b}? \in A(B^\alpha)) \text{ .} \end{array}$$

Otherwise, the conclusion is given by claim 2 directly.

We now prove claim 2. We write $P^{\alpha-1} = C[\Omega]$, where C is the context associated with $\vec{b}^\alpha?$, and $P^\alpha = C[R]$. Consider $\vec{b}? \in A(B^\alpha) \cap F(B)$ (hence, in particular, $\vec{b}? \neq \vec{b}^\alpha?$). There are two cases:

(I) $\vec{b}? \in A(B^{\alpha-1})$. Then we set $i^\alpha(\vec{b}?) = i^{\alpha-1}(\vec{b}?)$. We write $P^{\alpha-1} = D[\Omega][\Omega_1]_1$, where D is a context with two holes [ ] and [ ]$_1$ occurring each once and corresponding to $\vec{b}^\alpha?$ and $\vec{b}?$, respectively. Let now $d_1, \ldots, d_k$ be observable states. We distinguish three cases.

(A) $\forall\, i \leq k \;\; b_i \leq d_i$. By induction we have, for all N:

$$(1) \quad [\![C[\Omega][N]_1]\!]\bullet(d_1, \ldots, d_k) = [\![N]\!]\bullet(d_1, \ldots, d_k).$$

In particular, $[\![C[\Omega][m]_1]\!]\bullet(d_1, \ldots, d_k) = \{(?, m)\}$ (m arbitrary). By induction, we can also suppose that $d_i = [\![Q_i]\!]$ for some Q_i, for all i. Let $D = C[\vec{Q}/\vec{x}]$. Then $[\![D[\Omega][m]_1]\!] = \{(?, m)\}$ by what we just noticed. It follows that [ ] is not a sequentiality index. Hence the sequentiality index, which exists by proposition 14.4.13, is [ ]$_1$. In particular:

$$(2) \quad [\![C[R][\Omega]_1]\!]\bullet(d_1, \ldots, d_k) = \bot.$$

We have to prove $[\![C[R][N]_1]\!]\bullet(d_1, \ldots, d_k) = [\![N]\!]\bullet(d_1, \ldots, d_k)$, for all N. We distinguish two cases.

(a) $[\![N]\!]\bullet(d_1, \ldots, d_k) \neq \bot$: Then the conclusion follows from (1) by monotonicity.

(b) $[\![N]\!]\bullet(d_1, \ldots, d_k) = \bot$: Then the conclusion boils down to (2).

(B) $(\exists\, j \;\; b_j \not\leq d_j)$ and $(\forall\, i \leq k \;\; b_i^\alpha \leq d_i)$. By induction, we have, for all L:

$$(3) \quad [\![C[L][\Omega]_1]\!]\bullet(d_1, \ldots, d_k) = [\![L]\!]\bullet(d_1, \ldots, d_k).$$

and our goal is to prove $[\![C[R][N]_1]\!]\bullet(d_1, \ldots, d_k) = B^\alpha\bullet d_1\bullet\cdots\bullet d_k$, for all N. We distinguish three cases.

(a) $(\vec{b}^{\alpha}?, \mathit{output}\ n) \in B^{\alpha}$. Then $B^{\alpha}\bullet d_1\bullet\cdots\bullet d_k = \{(?,n)\}$, by the definition of $\bullet$. On the other hand, $R = \{(?,n)\}$ by construction, and the conclusion then follows from (3) by monotonicity.

(b) $(\vec{b}^{\alpha}?, \mathit{valof}\ c.i) \in B^{\alpha}$ and $(c,\mathsf{e}) \in d_i$. Then $B^{\alpha}\bullet d_1\bullet\cdots\bullet d_k = \{(?,\mathsf{e})\}$. On the other hand, by claim 1, we have $[\![\mathit{catch}(S)]\!]\bullet(d_1,\ldots,d_k) = \{(?,\mathsf{e})\}$, hence $[\![R]\!]\bullet(d_1,\ldots,d_k) = \{(?,\mathsf{e})\}$ The conclusion follows again from (3) by monotonicity.

(c) $(\vec{b}^{\alpha}?, \mathit{valof}\ c.i) \in B^{\alpha}$ and $(c,\mathsf{e}) \notin d_i$. Then $B^{\alpha}\bullet d_1\bullet\cdots\bullet d_k = \bot$. On the other hand, since all the branches of R are Ω's, we have

$$[\![R]\!]\bullet(d_1,\ldots,d_k) \neq \bot \Rightarrow [\![\mathit{catch}(S)]\!]\bullet(d_1,\ldots,d_k) = \{(?,\mathsf{e})\}.$$

Hence, by claim (1) and from the assumption $(c,\mathsf{e}) \notin d_i$, we get

$$(4)\quad [\![R]\!]\bullet(d_1,\ldots,d_k) = \bot.$$

Reasoning as with (1) above, we conclude from (3) that $[\ ]$ is the sequentiality index. Hence, for all N:

$$(5)\quad [\![C[\Omega][N]_1]\!]\bullet(d_1,\ldots,d_k) = \bot.$$

The conclusion then follows from (4) and (5).

(C) $(\exists j\ \ b_j \not\leq d_j)$ and $(\exists j_1\ \ b^{\alpha}_{j_1} \not\leq d_{j_1})$. By induction we have, for all N and L:

$$(6)\quad [\![C[\Omega][N]_1]\!]\bullet(d_1,\ldots,d_k) = B^{\alpha}\bullet d_1\bullet\cdots\bullet d_k$$
$$(6)_1\quad [\![C[L][\Omega]_1]\!]\bullet(d_1,\ldots,d_k) = B^{\alpha}\bullet d_1\bullet\cdots\bullet d_k\ .$$

There are two cases:

(a) $B^{\alpha}\bullet d_1\bullet\cdots\bullet d_k \neq \bot$. Then the conclusion follows from (6) by monotonicity.

(b) $B^{\alpha}\bullet d_1\bullet\cdots\bullet d_k \neq \bot$. Then since (6), $(6)_1$ hold in particular for $N = \{(?,\mathsf{e})\}$, $L = \{(?,\mathsf{e})\}$, respectively, we conclude that neither $[\ ]$ nor $[\ ]_1$ can be sequentiality indexes. Hence the conclusion $[\![C[R][N]]\!]\bullet(d_1,\ldots,d_k) = \bot$ follows, as otherwise there would exist a sequentiality index, by proposition 14.4.13.

(II) $\vec{b}? \notin A(B^{\alpha-1})$. This can only happen if $\vec{b}^{\alpha}?$ is filled with some $\mathit{valof}\ c.i$ in B^{α}, and if $\vec{b}$ has the following form:

$$\begin{array}{ll} b_j = b^{\alpha}_j & \text{if } j \neq i \\ b_i = b^{\alpha}_i \cup \{(c,u)\} & \text{for some } u\ . \end{array}$$

By construction, this u is associated with one of the branches of R, which we represent by means of a context $R = C_u[\Omega]$. Then we define $i^{\alpha}(\vec{b}?)$ as the occurrence of $[\ ]$ in $C[C_u]$. We distinguish the same cases as for (I).

(A) $\forall i \leq k \;\; b_i \leq d_i$. Our goal is to show $[\![C[C_u[N]]]\!]\bullet(d_1,\ldots,d_k) = [\![N]\!]\bullet(d_1,\ldots,d_k)$. Since we have a fortiori $b_i^\alpha \leq d_i$ for all i, we have by induction:

$$(7) \quad [\![C[C_u[N]]]\!]\bullet(d_1,\ldots,d_k) = [\![C_u[N]]\!]\bullet(d_1,\ldots,d_k).$$

On the other hand, we have $[\![\mathit{catch}(S)]\!]\bullet(d_1,\ldots,d_k) =$ "u" by claim 1, since $(c,u) \in d_i$. By the definition of C_u, this implies (for all N):

$$(8) \quad [\![C_u[N]]\!]\bullet(d_1,\ldots,d_k) = [\![N]\!]\bullet(d_1,\ldots,d_k).$$

Then the conclusion follows from (7) and (8).

(B) $(\exists j \;\; b_j \not\leq d_j)$ and $(\forall i \leq k \;\; b_i^\alpha \leq d_i)$. It follows from these assumptions that $b \not\subseteq d_i$, and that $(c,u) \notin d_i$. We still have (7) by induction. Our goal is to show $[\![C[C_u[N]]]\!]\bullet(d_1,\ldots,d_k) = (B^{\alpha-1} \cup \{(\vec{b}^\alpha?, \mathit{valof}\ c.i)\})\bullet d_1 \bullet \cdots \bullet d_k$, for all N. By definition of $\bullet$, we have:

$$(9) \quad (B^{\alpha-1} \cup \{(\vec{b}^\alpha?, \mathit{valof}\ c.i)\})\bullet d_1 \bullet \cdots \bullet d_k = \begin{cases} \{(?,\mathsf{e})\} & \text{if } (c,\mathsf{e}) \in d_i \\ \bot & \text{otherwise .} \end{cases}$$

On the other hand, by the definition of C_u, we can have $[\![C_u[N]]\!]\bullet(d_1,\ldots,d_k) \neq \bot$ only if either of the two following properties hold.

(a) $[\![\mathit{catch}(S)]\!]\bullet(d_1,\ldots,d_k) =$ "u". This case is impossible by claim 1, since $(c,u) \notin d_i$.

(b) $[\![\mathit{catch}(S)]\!]\bullet(d_1,\ldots,d_k) = \{(?,\mathsf{e})\}$. By claim (1), this happens exactly when $(c,\mathsf{e}) \in d_i$, and then $[\![C_u[N]]\!]\bullet(d_1,\ldots,d_k) = \{(?,\mathsf{e})\}$

The conclusion follows from this case analysis and from (9).

(C) $(\exists j \;\; b_j \not\leq d_j)$ and $(\exists j_1 \;\; b_{j_1}^\alpha \not\leq d_{j_1})$. We have, for all L:

$$(10) \quad \begin{array}{rcll} [\![C[L]]\!]\bullet(d_1,\ldots,d_k) & = & B^{\alpha-1}\bullet d_1 \bullet \cdots \bullet d_k & \text{(by induction)} \\ & = & B^{\alpha}\bullet d_1 \bullet \cdots \bullet d_k & \text{(since } b_{j_1}^\alpha \not\leq d_{j_1}) \text{ .} \end{array}$$

Then the conclusion follows by instantiating (10) to $L = C_u[N]$. □

Theorem 14.4.16 (full abstraction for SPCF**)** *The model of sequential algorithms is fully abstract for* SPCF.

PROOF. Let M and N be such that $[\![M]\!] \not\leq [\![N]\!]$. We can assume M, N closed since currying is monotonic. By lemma 14.4.14 and by theorem 14.4.15, there exists an algorithm a defined by a closed term F such that

$$[\![\vdash FM]\!] = (a\bullet[\![\vdash M]\!]) \not\leq (a\bullet[\![\vdash N]\!]) = [\![\vdash FN]\!].$$

The context $C = F[\ ]$ witnesses $M \not\leq_{obs} N$. □

Exercise 14.4.17 *Adapt the proof of theorem 14.4.15 to show that the model of observable algorithms (cf. exercise 14.4.8) is fully abstract for* SPCF(*Err*).

Chapter 15

Domains and Realizability

Kleene [Kle45] first introduced a realizability interpretation of Heyting arithmetic (HA) as a tool for proving its consistency. This interpretation provides a standard link between constructive mathematics (as formalized in HA) and classical recursion theory. Moreover, it has the merit of giving a solid mathematical content to Brouwer-Heyting-Kolmogorov explanation of constructive proofs (see, e.g., [TvD88]).

Let us consider Peano arithmetic formalized in an intuitionistic first order logic with equality and a signature with symbols 0 for zero, and s for successor. Let $\mathcal{N}$ be the intended interpretation of the signature over the structure of natural numbers. We write $\mathcal{N} \models t = s$ if the formula $t = s$ is valid in $\mathcal{N}$. We define a realizability binary relation $\Vdash \subseteq \omega \times Form$ between numbers and formulae, by induction on the formulae, as follows:

$$
\begin{array}{lll}
n \Vdash t = s & \text{if } \mathcal{N} \models t = s & \\
n \Vdash \phi \wedge \psi & \text{if } \pi_1 n \Vdash \phi \text{ and } \pi_2 n \Vdash \psi & (1) \\
n \Vdash \phi \vee \psi & \text{if } (\pi_1 n = 0 \text{ and } \pi_2 n \Vdash \phi) \text{ or } (\pi_1 n = 1 \text{ and } \pi_2 n \Vdash \psi) & \\
n \Vdash \phi \to \psi & \text{if for each } m\, (m \Vdash \phi \text{ implies } \{n\}m \Vdash \psi) & (2) \\
n \Vdash \forall x.\phi & \text{if for each } m\, (\{n\}m \downarrow \text{ and } \{n\}m \Vdash \phi[\underline{m}/x]) & (3) \\
n \Vdash \exists x.\phi & \text{if } \pi_2 n \Vdash \phi[\underline{\pi_1 n}/x] &
\end{array}
$$

where: (1) π_1, π_2, are the first and second projections with respect to an injective coding $\langle \_, \_ \rangle : \omega^2 \to \omega$. (2) $\{n\}m$ is the n-th Turing machine applied to the input m. (3) $\underline{m}$ is a numeral in the system HA corresponding to the natural number m. We note that the formula $\perp$ is never realized. $t \downarrow$ denotes the fact that the expression t is defined. Kleene's equality $\cong$ is defined as $t \cong s$ iff $(t \downarrow \Leftrightarrow s \downarrow)$ and $(t \downarrow \Rightarrow t = s)$. In the standard equality the arguments are supposed defined $t = s$ implies $t \downarrow$ and $s \downarrow$. Whenever $ts \downarrow$ it is the case that $t \downarrow$ and $s \downarrow$.

Let us turn towards potential applications of this interpretation. To any formula ϕ in HA we can associate the set $[\![\phi]\!]$ of its realizers $[\![\phi]\!] = \{n \mid n \Vdash \phi\}$. It is easy to prove a soundness theorem saying that any provable formula in HA

has a non empty collection of realizers (i.e. it is realizable). The consistency of *HA* is an immediate corollary. More interestingly, realizability can be used to check the consistency of various extensions of Heyting arithmetic. For instance let us consider the formalization in *HA* of two popular axiom schemata known as Church Thesis (*CT*) and Markov Principle (*MP*). In the following ϕ is a primitive recursive predicate, i.e. a formula without unbounded quantifications.

- $(CT)\ \ \forall n.\exists! m.\phi(n,m) \to \exists k.\forall n.\exists m.(\phi(n,Um) \wedge Tknm)$

Where U is a function and T is a predicate (called Kleene predicate) such that $k(n) \cong U(\mu m\, Tknm)$ (μ is the minimalization operator, cf. appendix A). Intuitively, Um is the final result of a computation m, and $Tknm$ holds iff the program k with input n produces a terminating computation m. Church thesis states that any single-valued relation over the natural numbers that is definable in *HA* is computable by some recursive function. The reason being that from any (constructive) proof of a Π^0_2 sentence, $\forall n.\exists! m.\phi(n,m)$, we can (effectively) extract an algorithm that given n finds the m such that $\phi(n,m)$.

- $(MP)\ \ (\forall n.(\phi(n) \vee \neg\phi(n)) \wedge \neg\neg\exists n.\phi(n)) \to \exists n.\phi(n)$

The intuition behind (*MP*) is the following: if we have a decidable predicate $(\forall n.(\phi(n) \vee \neg\phi(n))$ and an oracle that tells us that such predicate is non-empty $(\neg\neg\exists n.\phi(n))$ then we can effectively find an element satisfying the predicate simply by enumerating the candidates and checking the predicate on them.

(*CT*) and (*MP*) are not provable in *HA* but they can be consistently added to it. This fact, which is not obvious, can be proved by showing that (*CT*) and (*MP*) are realized in Kleene interpretation.

Having provided some historical and technical perspective on realizability we can outline the main theme of this chapter. Our goal is to generalize Kleene interpretation in two respects: (1) We want to model Type Theories (not just *HA*). (2) We want to interpret Proofs/Programs and not just Propositions/Types. In order to obtain some results in this direction we will concentrate on a special class of "realizability models". Two basic features of these models are:

(1) Types can be regarded as *constructive* sets.

(2) There is a distinction between a *typed value* and its *untyped realizers*.

The first feature relates to a general programme known as *synthetic domain theory* (see [Hyl90]) that advocates the construction of a mathematical framework in which data types can be regarded as sets. A number of examples show that classical set theory is not well-suited to this purpose, think of models for recursive functions definitions, untyped λ-calculus, and polymorphism. On the other hand some promising results have been obtained when working in a universe of *constructive* sets. In particular *realizability* has been the part of constructive mathematics that has been more successful in implementing this plan. Historically this programme was first pushed by Scott and his students McCarty and Rosolini [McC84, Ros86], whose work relates in particular to the effective topos

[Hyl82] (but see also [Mul81] for another approach). Related results can be found in [Ama89, AP90, FMRS92, Pho90]. In a realizability universe the size of function spaces, and dependent and second order products can be surprisingly small. A typical result is the validity of a *Uniformity Principle* which plays an important role in the interpretation of second order quantification as intersection (more on this in section 15.2).

The second feature relates to the way "constructivity" is built into the realizability model. We rely on a *partial combinatory algebra* (pca) which is an untyped applicative structure satisfying weaker requirements than a λ-model. We build over a pca, say D, a set-theoretical universe where every set, say X, is equipped with a realizability relation $\Vdash_X \subseteq D \times X$. If $d \Vdash_X x$ then d can be regarded as a realizer of x. Morphisms between the "sets" $(X, \Vdash_X)$ and $(Y, \Vdash_Y)$ are set-theoretical functions between X and Y that can be actually realized in the underlying pca in a sense that we will make clear later. In the programming practice there is a distinction between the explicitly typed program which is offered to the type-checker, and its untyped run time representation which is actually executed. Intuitively the typed-terms $\lambda x : nat.x$ and $\lambda x : bool.x$ may well have the same run-time representation, say $\lambda x.x$. This aspect is ignored by the domain-theoretical interpretation we have considered so far. In this interpretation $\lambda x : nat.x$ and $\lambda x : bool.x$ live in *different* universes. Realizers can also be regarded as untyped "implementations" of typed programs. Models based on realizability offer a two-levels view of computation: one at a typed and another at an untyped level. This aspect will be exploited to provide an interpretation of *type-assignment* (section 15.3) and *subtyping* systems (section 15.7).

The technical contents of the chapter is organised as follows. In section 15.1 we build a category of D-sets over a pca D. These are sets equipped with a realizability relation (as described above) and provide a nice generalization of Kleene realizability.

In section 15.2 we interpret system F (cf. chapter 11) in the category of *partial equivalence relations* (per) which is a particularly well behaved subcategory of the category of D-sets.

In section 15.3 we exploit the two-levels structure of realizability models to interpret *type-assignment systems* which are formal systems where types are assigned to untyped terms (cf. section 3.5). We prove a completeness theorem by relying on a standard term model construction.

In section 15.4 we study the notion of partiality in the category of partial equivalence relations and obtain in this way a pCCC of per's. We exploit the dominance Σ of the pCCC to define an "intrinsic" preorder on the points of a per. The full subcategory of the per's for which this preorder is a partial order (i.e. antisymmetric) forms a *reflective* subcategory of the category of per's. We refer to these per's as *separated* per's or Σ-per's for short.

In section 15.5 we work with Kleene's pca $(\omega, \bullet)$ and we introduce a subcat-

egory of *complete* separated per's which have lub's of (effectively given) chains. In this framework we prove a generalization of Myhill-Shepherdson theorem (cf. chapter 1) asserting that all realized functions are Scott-continuous.

In section 15.6 we concentrate on a D_∞ λ-model and identify in this framework a full subcategory of *complete, uniform* per's, where we can solve recursive domain equations up to equality.

In section 15.7 we introduce a *theory of subtyping for recursive types* and present a sound interpretation in the category of complete uniform per's.

15.1 A Universe of Realizable Sets

In Kleene interpretation the basic *realizability structure* is given by the collection of natural numbers with an operation of partial application of a number, seen as a program, to another number, seen as an input. A convenient generalization of this notion is that of *partial combinatory algebra* (cf. [Bet88]).

Definition 15.1.1 (partial combinatory algebra) *A partial combinatory algebra (pca) is a structure* $\underline{D} = (D, k, s, \bullet)$ *where* $k, s \in D$, $\bullet : D \times D \rightharpoonup D$, *and* $kxy = x$, $sxy\downarrow$, *and* $sxyz \cong xz(yz)$.

In the following, the application $t \bullet s$ is abbreviated as ts. Whenever $ts\downarrow$ it is the case that $t\downarrow$ and $s\downarrow$. In pca's it is possible to simulate λ-abstraction. Let $\underline{D}$ be a pca and let t be a closed term over the pca enriched with a constant d for every element $d \in D$. Clearly every term either denotes an element in D or is undefined. Given a term t we define inductively on the structure of t, a new term $\lambda^* d.t$ in which the element d is "abstracted":

$$\begin{array}{ll} \lambda^* d.d & = skk \\ \lambda^* d.t & = kt \quad (d \text{ does not occur in } t) \\ \lambda^* d.ts & = s(\lambda^* d.t)(\lambda^* d.s)\ . \end{array}$$

It is easy to verify that for any $d \in D$, $(\lambda^* d.t)d \cong t$.

Example 15.1.2 *(1) An important example of pca is Kleene's* $(\omega, \bullet)$ *where* ω *is the set of natural numbers and* $n \bullet m$ *is the* n*-th Turing machine applied to the input* m. *(2) Another canonical example of pca is that of a non-trivial domain* D *that is a retract of its partial function space,* $(D \rightharpoonup D) \lhd D$, *in the category of directed complete partial orders and partial continuous morphisms.*

Given a pca we have to decide how to interpret formulas and proofs, and more generally types and programs. In Kleene interpretation, formulas are interpreted as subsets of natural numbers, on the other hand no mention is made of morphisms, hence no obvious interpretation of proofs is available.

The first attempt could consist in interpreting types as subsets of the realizability structure. But in order to have a model of type theory we need at least a CCC, so which are the morphisms? It is clear that, to build an interesting structure, morphisms have to be somehow *realized*. It appears that some structure is missing to get a CCC, for this reason we seek a finer description of types. Rather than identifying types with a collection of realizers we consider a type as a *partial equivalence relation* (per) over the collection of realizers. There is now an obvious notion of morphism that makes the category into a CCC.

Supposing $A, B, \ldots$ binary relations over a set D we write $d\,A\,e$ as an abbreviation for $(d,e) \in A$ and we set:

$$[d]_A = \{e \in D \mid d\,A\,e\} \quad [A] = \{[d]_A \mid d\,A\,d\} \quad |A| = \{d \in D \mid d\,A\,d\}\ .$$

Definition 15.1.3 (partial equivalence relations) *Let D be a pca. The category of per's over D ($\mathbf{per}_D$) is defined as follows:*

$$\begin{array}{ll} \mathbf{per}_D & = \{A \mid A \subseteq D \times D \text{ and } A \text{ is symmetric and transitive}\} \\ \mathbf{per}_D[A,B] & = \{f : [A] \to [B] \mid \exists \phi \in D\, \forall d \in D\, (d\,A\,d \Rightarrow \phi d \in f([d]_A))\}\ . \end{array}$$

If ϕ is a realizer for the morphism $f : A \to B$, i.e. $\forall d \in D\,(d\,A\,d \Rightarrow \phi d \in f([d]_A))$, we may denote f with $[\phi]_{A \to B}$ (consistently with the definition of exponent in $\mathbf{per}_D$ given in the following proposition).

Theorem 15.1.4 ($\mathbf{per}_D$ is a CCC) *The category, $\mathbf{per}_D$, of partial equivalence relations over a pca D is cartesian closed.*

PROOF. Mimicking what is done in the λ-calculus, we can define pairing as $\langle d_1, d_2\rangle \equiv \lambda^* p.(pd_1)d_2$, and projections as $\pi_1 d \equiv dk$ and $\pi_2 d \equiv d(k(skk))$.

- Terminal object. We set $1 = D \times D$. For any $d \in D$ the "constant function" $\lambda^* e.d$ realizes the unique morphism from a per A into 1.
- Product. We define for the product per:

$$d\,A \times B\,e \ \text{ iff } \ \pi_1 d\,A\,\pi_1 e \text{ and } \pi_2 d\,B\,\pi_2 e\ .$$

It is immediate to verify that pairing and projections in the pca realize the pairing and projections morphisms of the category.

- Exponent. We define for the exponent per:

$$h\,B^A\,k \ \text{ iff } \ \forall d, e\,(d\,A\,e \Rightarrow hd\,B\,ke)\ .$$

Morphisms can be regarded as equivalence classes. The evaluation is realized by $\lambda^* d.(\pi_1 d)(\pi_2 d)$, the natural isomorphism Λ is realized by $\lambda^*\phi.\lambda^* c.\lambda^* a.\phi\langle a, c\rangle$. □

The following exercises should motivate the shift from subsets of D to per's.

Exercise 15.1.5 *One can identify the subsets of the realizability structure with those per's that have at most one equivalence relation. Show that the full subcategory composed of these per's is cartesian closed, but each object is either initial or terminal.*

Exercise 15.1.6 *Consider a category of per's over the pca D, say $\mathbf{C}$, in which, as above, per's have at most one equivalence relation but where morphisms are defined as follows: $\mathbf{C}[A,B] = \{\phi \in D \mid \forall d \in D\,(d \in |A| \Rightarrow \phi d \in |B|)\}$. Show that $\mathbf{C}$ is not cartesian closed.*

Exercise 15.1.7 *Show that $\mathbf{per}_D$ has all finite limits and colimits.*

Given a pca D we introduce the category of D-sets in which we can pinpoint a full reflective sub-category of *modest sets*, say $\mathbf{M}_D$, that is equivalent to $\mathbf{per}_D$. The category of D-sets intuitively justifies our claim of working in a "constructive" universe of sets. Formally one can show that D-sets form a full subcategory of the *effective topos* [Hyl82]. [1]

Definition 15.1.8 *A D-*set *is a pair $(X, \Vdash_X)$ where X is a set and $\Vdash_X \subseteq D \times X$ is an* onto *realizability relation that is $\forall x \in X\, \exists d \in D\, d \Vdash_X x$. A* morphism *of D-sets, say $f : (X, \Vdash_X) \to (Y, \Vdash_Y)$, is a function $f : X \to Y$, such that:*

$$\exists \phi \in D\, \forall d, x\, (d \Vdash_X x \Rightarrow \phi d \Vdash_Y f(x)) \ .$$

A D-set $(X, \Vdash_X)$ is modest *if the relation $\Vdash_X$ is* single valued*, that is $d \Vdash_X x$ and $d \Vdash_X y$ implies $x = y$. We denote with D-**set** the category of D-sets and with $\mathbf{M}_D$ the full subcategory of modest sets.*

Proposition 15.1.9 *(1) The categories $\mathbf{M}_D$ and $\mathbf{per}_D$ are equivalent.*
*(2) The category $\mathbf{M}_D$ is a reflective subcategory of D-**set**.*

PROOF. (1) To the modest set $(X, \Vdash_X)$ we associate the per $P(X, \Vdash_X)$ defined as:

$$d\, P(X, \Vdash_X)\, e \ \text{ iff } \ \exists x \in X\, (d \Vdash_X x \text{ and } e \Vdash_X x) \ .$$

(2) The basic observation is that if $f : (X, \Vdash_X) \to (Y, \Vdash_Y)$ where $(Y, \Vdash_Y)$ is modest then:

$$d \Vdash_X x \text{ and } d \Vdash_X y \ \Rightarrow \ f(x) = f(y) \ .$$

If f is realized by ϕ then $\phi d \Vdash_Y f(x)$ and $\phi d \Vdash_Y f(y)$, which forces $f(x) = f(y)$. Let us now describe how to associate a modest set $(Y, \Vdash_Y)$ to a D-set $(X, \Vdash_X)$. First we define a relation R over X as:

$$x R y \ \text{ iff } \ \exists d \in D\, (d \Vdash_X x \text{ and } d \Vdash_X y) \ .$$

[1] A *topos* is an intuitionistic generalization of set theory formalized in the language of category theory.

$$\begin{array}{ll} [\![x]\!]\rho & = \rho(x) \\ [\![\lambda x.P]\!]\rho & = \lambda^* d.[\![P]\!]\rho[d/x] \\ [\![PQ]\!] & = ([\![P]\!]\rho)([\![Q]\!]\rho) \end{array}$$

Figure 15.1: Interpretation of λ-terms in a pca

Let R^+ denote the equivalence relation obtained by the transitive closure of R. We consider the quotient set $Y = [X]_{R^+}$ equipped with the relation $\Vdash_Y$ defined as follows:

$$d \Vdash_Y [x]_{R^+} \quad \text{iff} \quad \exists z \in [x]_{R^+}\, d \Vdash_X z \ .$$

The pair $(Y, \Vdash_Y)$ is the modest set we looked for. □

15.2 Interpretation of System F

We define an interpretation of system F (cf. chapter 11) in the category of per's. Since $\mathbf{per}_D$ is a CCC we already know how to interpret the simply typed fragment of system F. On the other hand the interpretation of the clauses $(\forall_I)$ and $(\forall_E)$ is more problematic. In order to show that the interpretation is well defined we introduce an auxiliary interpretation of the underlying untyped λ-terms which allows to express a certain *uniformity* of the main interpretation with respect to type abstraction and type application.

Definition 15.2.1 (type interpretation) *Let Tvar be the set of type variables. Given a type environment, say $\eta : Tvar \to \mathbf{per}_D$, the interpretation of a type is a per defined by induction as follows:*

$$\begin{array}{ll} [\![t]\!]\eta & = \eta(t) \\ [\![\sigma \to \tau]\!]\eta & = [\![\tau]\!]\eta^{[\![\sigma]\!]\eta} \\ [\![\forall t.\sigma]\!]\eta & = \bigcap_{A \in per_D} [\![\sigma]\!]\eta[A/t] \ . \end{array}$$

Let Var denote the set of term variables and $\rho : Var \to D$ be an environment. In figure 15.1 we define the interpretation of an untyped λ-term in the pca D.

Theorem 15.2.2 *Suppose $\Gamma \vdash M : \sigma$, where $\Gamma \equiv x_1 : \sigma_1, \ldots, x_n : \sigma_n$, then for any type assignment η, and for any d_i, e_i such that $d_i\, [\![\sigma]\!]\eta\, e_i$, for $i = 1, \ldots, n$, we have:*

$$([\![er(M) : \sigma]\!][\vec{d}/\vec{x}])\ ([\![\sigma]\!]\eta)\ ([\![er(M) : \sigma]\!][\vec{e}/\vec{x}]) \ .$$

PROOF. This is a simple induction on the length of the typing judgment, and can be regarded as yet another variation on the fundamental lemma of logical relations 4.5.3. $(Asmp)$ is satisfied by hypothesis. The interpretations for $(\to_I)$ and $(\to_E)$ just use the realizers of the natural transformation Λ and of the evaluation morphism. The crucial point is $(\forall_I)$ where we use the side condition $t \notin FV_t(\Gamma)$. $(\forall_E)$ follows by the interpretation of second order quantification as intersection. $\Box$.

The basic idea is to interpret a typed term as the equivalence class in the appropriate per of the interpretation of the erased term. Given $\Gamma \equiv x_1 : \sigma_1, \ldots, x_n : \sigma_n$ and η type environment let

$$[\![\Gamma]\!]\eta = (\cdots(1 \times [\![\sigma_1]\!]\eta) \times \cdots \times [\![\sigma_n]\!]\eta) \ .$$

The interpretation of a term $\Gamma \vdash M : \sigma$ should be a morphism $f : [\![\Gamma]\!]\eta \to [\![\sigma]\!]\eta$. Equivalently, we can determine this morphism by taking the equivalence class of a realizer ϕ in the exponent per $([\![\sigma]\!]\eta)^{[\![\Gamma]\!]\eta}$. The existence of the realizer ϕ follows by theorem 15.2.2. Hence we have the following definition.

Definition 15.2.3 (typed term interpretation) *Given a type environment η the interpretation of a judgment $\Gamma \vdash M : \sigma$ in system F is defined as follows:*

$$[\![\Gamma \vdash M : \sigma]\!]\eta = [\lambda^* d.[\![er(M)]\!][\vec{\pi_{n,i}} d / \vec{x_i}]]_{[\![\sigma]\!]\eta^{[\![\Gamma]\!]\eta}}$$

where $\pi_{n,i}(d) = \pi_2(\pi_1(\cdots \pi_1(d)))$, with π_1 iterated $(n-i)$ times.

Exercise 15.2.4 *(1) Show that two terms with the same type and with identical erasures receive the same interpretation in per models. (2) Verify that $\beta\eta$-convertible terms are equated in per models.*

The interpretation in $\mathbf{per}_D$ can be extended to handle dependent types. In the following we outline some results in this direction.

Definition 15.2.5 *Given a D-set $(X, \Vdash_X)$ and a function $F : X \to D$-**set** we define the D-set $([\Pi_X F], \Vdash_{\Pi F})$ as follows (cf. section 11.1):*

- $[\Pi_X F] = \{f \in \Pi_X F \mid \exists \phi \in D \, \phi \Vdash_{\Pi F} f\}$.
- $\phi \Vdash_{\Pi F} f$ *if* $\forall x \in X \, \forall d \in D \, (d \Vdash_X x \Rightarrow \phi d \Vdash_{F(x)} f(x))$

If we look at modest sets as the collection of types then a function $F : X \to \mathbf{M}_D$ can be regarded as a dependent type. It is easy to prove the following exercise.

Exercise 15.2.6 *In the hypotheses of definition 15.2.5, if $F : X \to \mathbf{M}_D$ then the D-set $([\Pi_X F], \Vdash_{\Pi F})$ is modest. Hint: check that $\Vdash_{\Pi F}$ is single valued using the fact that the realizability relations associated to modest sets are single valued.*

The construction defined above can be shown to be a categorical product in a suitable framework.

Exercise 15.2.7 ** Define an interpretation of the system LF (cf. chapter 11) in the category of D-sets.*

In figure 15.1 we have interpreted second order universal quantification as intersection. Actually, this interpretation is compatible with the idea that a universal quantification is interpreted as a product. We hint to this fact and refer to [LM92] for a more extended discussion. Since $\mathbf{M}_D$ is not a small category we consider transformations $F : \mathbf{per}_D \to \mathbf{M}_D$. Moreover we regard the collection of per's as a D-set, say $\underline{per}_D$, by equipping it with the full realizability relation $\underline{per}_D = (per_D, D \times per_D)$.

Proposition 15.2.8 *Given a function $F : \underline{per}_D \to \mathbf{M}_D$, we have:*

$$([\Pi_{per} F], \|\!-_{\Pi F}) \cong \bigcap_{A\ per} P(F(A)) \ .$$

Where if $(X, \|\!-_X)$ is a modest set then $P(X, \|\!-_X)$ is the corresponding per, as defined in proposition 15.1.9.

PROOF HINT. The isomorphism from the product to the intersection is realized by $\lambda^*\phi.\phi d_0$, for some $d_0 \in D$, and its inverse is realized by $\lambda^* d.\lambda d'.d$. □

15.3 Interpretation of Type Assignment

We consider the problem of building *complete* formal systems for assigning types to untyped λ-terms [CF58, BCD83, Hin83]. In chapter 4 we have referred to this approach as typing à la Curry and we have pointed out its relevance in the definition of algorithms that reconstruct automatically the type information that is not explicitly available in the program.

We develop a type assignment system that is parametric with respect to: (1) a λ-theory $\mathcal{E}$, and (2) a collection of typing hypotheses B on variables and closed λ-terms. To interpret type assignment we introduce *type structures* which are a slight generalization of per models.

Definition 15.3.1 *A* type frame $\mathcal{T}$ *is made up of three components:*
(1) A $\lambda\beta$-model $(D, \bullet)$.
(2) A collection $T \subseteq per_D$ closed under exponentiation:

$$X, Y \in T \text{ implies } Y^X \in T \ .$$

(3) A collection $[T \to T] \subseteq \mathbf{Set}[T, T]$ closed under intersection:

$$F \in [T \to T] \text{ implies } \bigcap_{A \in T} F(A) \in T \ .$$

Condition (1) is natural since we consider systems to assign types to λ-terms and not to combinators. Conditions (2) and (3) are obvious generalizations of properties satisfied by the per model. The type interpretation defined in 15.2.1 generalizes immediately to an arbitrary type frame.

Definition 15.3.2 *The types of system F are interpreted in a type frame $\mathcal{T}$ parametrically with respect to a type environment $\eta : Tvar \to T$ as follows:*

$$\begin{array}{ll} [\![t]\!]\eta & = \eta(t) \\ [\![\sigma \to \tau]\!]\eta & = [\![\tau]\!]\eta^{[\![\sigma]\!]\eta} \\ [\![\forall t.\sigma]\!]\eta & = \bigcap_{A \in T} [\![\sigma]\!]\eta[A/t] \ . \end{array}$$

A type frame is a type structure *whenever the interpretation of intersection is correct, that is for any σ, $\lambda A \in T.[\![\sigma]\!]\eta[A/t] \in [T \to T]$ (cf. definition of λ-model in chapter 3).*

Definition 15.3.3 *A type structure has no* empty types *if $\forall A \in T\,(A \neq \emptyset)$.*

Exercise 15.3.4 *Give an example of type structure without empty types.*

A type free λ-term is interpreted in the $\lambda\beta$-model $(D, \bullet)$ according to the definition 3.2.2. We recall that the interpretation is parametric in an environment ρ.

Definition 15.3.5 (basis) *A basis B is a set $\{P_i : \sigma_i\}_{i \in I}$ where P_i is either a closed untyped λ-term or a variable, and all variables are distinct.*

Let us fix an untyped λ-theory, say $\mathcal{E}$, (cf. chapter 4). We define a system to assign types to untyped λ-terms assuming a basis B and modulo a λ-theory $\mathcal{E}$. For instance, we may be interested in a system to type terms under a basis $B = \{\lambda x.x : t \to s\}$ and modulo the (η) rule. The basis B asserts that every term having type t has also type s and the rule (η) forces extensionality.

Definition 15.3.6 (type assignment system) *Given a λ-theory $\mathcal{E}$ we define in figure 15.2 a type assignment system whose judgments are of the form $B \vdash P : \sigma$, where B is a basis, P is an untyped λ-term, and σ is a type of system F.*

Definition 15.3.7 (interpretation) *Let $\mathcal{T}$ be a type structure over the λ-model D and let $Th(D)$ be the λ-theory induced by D (cf. chapter 4). We write $\mathcal{T} \models \mathcal{E}$ if $\mathcal{E} \subseteq Th(D)$. Given a type structure $\mathcal{T}$ such that $\mathcal{T} \models \mathcal{E}$ we write:*

$$\begin{array}{ll} B \models_{\mathcal{T}} P : \sigma & \text{if } \forall \rho : Var \to D, \eta : Tvar \to T\,(\rho, \eta \models B \ \Rightarrow \ \rho, \eta \models P : \sigma) \\ \rho, \eta \models P : \sigma & \text{if } [\![P]\!]\rho \in |[\![\sigma]\!]\eta| \\ \rho, \eta \models \{P_i : \sigma_i\}_{i \in I} & \text{if } \forall i \in I\,(\rho, \eta \models P_i : \sigma_i) \ . \end{array}$$

When the type structure is fixed we omit writing $\mathcal{T}$.

$$(Asmp)\quad \frac{P:\sigma \in B}{B \vdash P:\sigma}$$

$$(Eq)\quad \frac{B \vdash P':\sigma \quad P =_{\mathcal{E}} P'}{B \vdash P:\sigma}$$

$$(weak)\quad \frac{B \vdash P:\sigma \quad B \cup B' \text{ well-formed}}{B \cup B' \vdash P:\sigma}$$

$$(rmv)\quad \frac{B \cup \{x:\sigma\} \vdash P:\tau \quad x \notin FV(P)}{B \vdash P:\tau}$$

$$(\to_I)\quad \frac{B \cup \{x:\sigma\} \vdash P:\tau}{B \vdash \lambda x.P:\sigma \to \tau}$$

$$(\to_I^{\eta})\quad \frac{B \vdash \lambda x.Px:\sigma \to \tau \quad x \notin FV(P)}{B \vdash P:\sigma \to \tau}$$

$$(\to_E)\quad \frac{B \vdash P:\sigma \to \tau \quad B \vdash Q:\sigma}{B \vdash PQ:\tau}$$

$$(\forall_I)\quad \frac{B \vdash P:\sigma \quad t \notin FV_t(B)}{B \vdash P:\forall t.\sigma}$$

$$(\forall_E)\quad \frac{B \vdash P:\forall t.\sigma}{B \vdash P:\sigma[\tau/t]}$$

Figure 15.2: Type assignment system for second order types

Proposition 15.3.8 (soundness) *Let $\mathcal{T}$ be a type structure without empty types such that $\mathcal{T} \models \mathcal{E}$. If $B \vdash P : \sigma$ (modulo $\mathcal{E}$) then $B \models_{\mathcal{T}} P : \sigma$.*

PROOF. The statement is not obvious because of the $(\to_I)$ rule. For the sake of simplicity we suppose σ, τ closed. Then we have:

$$\begin{array}{ll} x : \sigma \models x : \tau & \text{iff } |[\![\sigma]\!]| \subseteq |[\![\tau]\!]| \\ \models \lambda x.x : \sigma \to \tau & \text{iff } [\![\sigma]\!] \subseteq [\![\tau]\!] \,. \end{array}$$

For this reason we have to generalize our statement. We define:

$$\begin{array}{ll} \lambda\vec{x}.\lambda\vec{y}.P & \equiv \lambda x_1 \ldots \lambda x_n.\lambda y_1 \ldots \lambda y_m.P \\ \vec{\sigma} \to \vec{\tau} \to \sigma & \equiv \sigma_1 \to \cdots \to \sigma_n \to \tau_1 \to \cdots \to \tau_m \to \sigma \end{array}$$

where $n, m \geq 0$, $\{x_1 : \sigma_1, \ldots, x_n : \sigma_n\} \subseteq B$, $y_j \notin dom(B)$, for $j = 1, \ldots, m$. With these conventions we show by induction on the length of the derivation: $B \vdash P : \sigma \;\Rightarrow\; B \models \lambda\vec{x}.\lambda\vec{y}.P : \vec{\sigma} \to \vec{\tau} \to \sigma$. □

Remark 15.3.9 *For a type structure with empty types the rule (rmv) is not sound as from an hypothesis which is never realized we can derive everything. For instance we have $\{x : \forall t.t\} \models_{per} \lambda x.x : \forall t.t$ and $\not\models_{per} \lambda x.x : \forall t.t$. If we eliminate the rule (rmv) then the type assignment system is sound for arbitrary type structures.*

The type assignment system in figure 15.2 is sound and *complete* to derive all judgments which are valid in type structures *without* empty types. This result can be extended to arbitrary type structures [Mit88]. In this case one introduces additional rules to reason about types' emptyness. For instance, one may enrich the basis with assertions $empty(\sigma)$ which hold if σ's interpretation is empty and then add the following typing rules.

$$\frac{}{\{x : \sigma, empty(\sigma)\} \vdash M : \tau} \qquad \frac{B \cup \{x : \sigma\} \vdash M : \tau \quad B \cup \{empty(\sigma)\} \vdash M : \tau}{B \vdash M : \tau} \,.$$

Exercise 15.3.10 *Check the soundness of the typing rules above.*

Theorem 15.3.11 (completeness) *Let $\mathcal{E}$ be a λ-theory and B be a basis. It is possible to build a type structure without empty types $\mathcal{T}_{\mathcal{E},B}$ over the term $\lambda\beta$-model induced by the λ-theory $\mathcal{E}$ so that:*

$$B \vdash P : \sigma \;\; \textit{iff} \;\; B \models_{\mathcal{T}_{\mathcal{E},B}} P : \sigma \,.$$

PROOF. The proof can be decomposed in two parts: (1) The proof that we can *conservatively* adjoin to the basis B a countable collection of type assignments $x_i : \sigma$, where σ is any type, $i \in \omega$, and x_i is a fresh variable. (2) The construction

of a type structure starting from a basis B' containing countably many type assignments $x_i : \sigma$.

Proof of (1). Let θ be an injective substitution from (type) variables to (type) variables such that $Var \backslash cod(\theta)$ and $Tvar \backslash cod(\theta)$ are infinite. We observe that:

$$B \vdash P : \sigma \text{ iff } \theta(B \vdash P : \sigma)$$

where the substitution is distributed componentwise. Given (i) a basis B, (ii) an enumeration of the types $\{\sigma_i\}_{i\in\omega}$ where each type occurs countably many times, (iii) an injective substitution θ as above, and (iv) a sequence $\{x_i\}_{i\in\omega}$ of distinct variables such that $\{x_i\}_{i\in\omega} \cap cod(\theta) = \emptyset$, we define:

$$B' = \{\theta(P : \sigma) \mid P : \sigma \in B\} \cup \{x_i : \sigma_i\}_{i\in\omega} .$$

The following facts can be easily verified:

- Given a type structure without empty types $\mathcal{T}$ such that $\mathcal{T} \models \mathcal{E}$,

$$B \models_{\mathcal{T}} P : \sigma \text{ iff } B' \models_{\mathcal{T}} \theta(P : \sigma) .$$

Hint: Since types are non-empty we can canonically extend any ρ, η such that $\rho, \eta \models B$ to ρ', η' such that $\rho', \eta' \models B'$.

- If $B' \vdash \theta(P : \sigma)$ then $B \vdash P : \sigma$. Hint: Use first compactness (if there is a proof, there is a proof that uses a finite part of the basis) to get a derivation with respect to a finite basis, then use (rmv) to eliminate the remaining adjoined variables.

Proof of (2). Given a basis B' as above, we define a type structure $\mathcal{T}_{\mathcal{E},B}$ without empty types as follows:

(1) Let D be the term $\lambda\beta$-model induced by the λ-theory $\mathcal{E}$ (cf. chapter 4). Let $[P]$ denote a generic element in D, that is the equivalence class of P modulo $\mathcal{E}$.

(2) We consider the collection of per's $T = \{\langle\sigma\rangle \mid \sigma \text{ type}\}$ defined as follows:

$$[P]\langle\sigma\rangle[Q] \text{ iff } B' \vdash P : \sigma \text{ and } B' \vdash Q : \sigma .$$

(3) As for the type functionals we consider the "definable" ones:

$$[T \to T] = \{F : T \to T \mid \exists \sigma, t\, F(\langle\tau\rangle) = \langle\sigma[\tau/t]\rangle\} .$$

Next we verify that this is a type structure without empty types.

- The type structure is without empty types because $x_i : \forall t.t \in B'$.
- Closure under exponentiation amounts to verify:

$$[P]\langle\sigma \to \tau\rangle[Q] \text{ iff } \forall [P'], [Q'] \, ([P']\langle\sigma\rangle[Q'] \Rightarrow [PP']\langle\tau\rangle[QQ']) .$$

Hint: The direction ($\Leftarrow$) follows from the following deduction where x is a variable adjoined to the basis B.

$$\begin{array}{rcl} B' \vdash x : \sigma & \Rightarrow & B' \vdash Px : \tau \\ & \Rightarrow & B' \vdash \lambda x.Px : \sigma \to \tau \ \text{ by } (\to_I) \\ & \Rightarrow & B' \vdash P : \sigma \to \tau \ \text{ by } (\to_I^{\eta}) \ . \end{array}$$

- Closure under intersection follows from:

$$[P]\langle \forall t.\sigma \rangle [Q] \ \text{ iff } \ \text{for all } \tau \, ([P]\langle \sigma[\tau/t] \rangle [Q]) \ .$$

If $B \vdash P : \sigma$ then $B \models_{\mathcal{T}_{\mathcal{E},B}} P : \sigma$, by soundness. Vice versa suppose $B \models_{\mathcal{T}_{\mathcal{E},B}} P : \sigma$. Then $B' \models_{\mathcal{T}_{\mathcal{E},B}} \theta(P : \sigma)$. Pick up the environment ρ_o, η_o defined as:

$$\rho_o(x) = [x] \quad \eta_o(t) = \langle t \rangle \ .$$

One can check $\rho_o, \eta_o \models B'$. From this we know $\rho_o, \eta_o \models \theta(P : \sigma)$ which is the same as $B' \vdash \theta(P : \sigma)$. Hence we can extract a proof of $B \vdash P : \sigma$. □

15.4 Partiality and Separation in per

In the following we concentrate on the problem of giving a per interpretation of type theories including *recursion* on terms and types. As usual we are naturally led towards a notion of *complete* partially ordered set. At the same time we want to stay faithful to our goal of regarding data types as particular sets of our realizability universe. Hence we look for a collection of "sets" on which it is possible to find an *intrinsic order* that is preserved by all set-theoretical functions. The method will be that of restricting the attention to *full* subcategories of $\mathbf{per}_D$. Hence, as in the classical approach described in chapter 1 we restrict our attention to certain sets endowed with structure, however, as opposed to that approach, we consider all "set-theoretic" functions and not just the continuous ones. The fact that functions are continuous is a theorem and not an hypothesis.

In chapter 8 we have introduced some basic notions about partial cartesian closed categories (pCCC) and their properties. We recall that every pCCC has an object Σ, called dominance, that classifies the admissible subobjects. In a pCCC the morphisms from an object a to the dominance Σ play the role of *convergence tests*. These tests induce a preorder $\leq_a$ on the points of an object. The idea of ordering points by tests bears a striking analogy with the one encountered in operational semantics of ordering terms by observations. Following [Ros86] we focus on the full subcategory of separated objects, which are composed of those objects for which $\leq_a$ is antisymmetric.

By convention, we write $x : a$ to indicate that x is a point of a, that is a morphism $x : 1 \to a$. Since we will be dealing with CCC's and pCCC's we

confuse points in the objects $a \to b$ and $a \rightharpoonup b$ with morphisms in $\mathbf{C}[a,b]$ and $\mathbf{pC}[a,b]$, respectively. For instance, $f : a \to b$ can be seen both as a morphism from a to b and as a point in $a \to b$. We introduce a convergence predicate, say $\Downarrow$, as follows: if $x : a$, $p = [m, f] : a \rightharpoonup b$, with $m : d \to a$, $f : d \to b$ then:

$$p \circ x \Downarrow \quad \text{iff} \quad \exists h : 1 \to d\,(m \circ h = x).$$

We write $p : a \rightharpoonup b$ and $p : a \to (b)_\perp$ interchangeably. When x is a point, we shall often abbreviate $p \circ x$ with px.

Definition 15.4.1 (intrinsic preorder) *Let $(\mathbf{C}, M)$ be a pCCC with dominance Σ, and a be an object of $\mathbf{C}$. We define a preorder $\leq_a$, called intrinsic preorder, on the points of a as:*

$$x \leq_a y \quad \textit{iff} \quad \forall p : a \to \Sigma\,(p \circ x \Downarrow \ \textit{implies}\ p \circ y \Downarrow)\ .$$

The intuition is that x is less then y in a, if every convergence test $p : a \to \Sigma$ that succeeds on x also succeeds on y. In the following we also write $p \circ x \leq p \circ y$ for $(p \circ x \Downarrow \Rightarrow p \circ y \Downarrow)$ and $p \circ x \cong p \circ y$ for $(p \circ x \Downarrow \Leftrightarrow p \circ y \Downarrow)$.

Definition 15.4.2 (category of Σ-objects) *Given a pCCC $(\mathbf{C}, M)$ with dominance Σ we denote with $\Sigma\mathbf{C}$ the full subcategory of $\mathbf{C}$ whose objects enjoy the property that the intrinsic preorder is anti-symmetric. An object a such that $\leq_a$ is a partial order is called a Σ-object or, equivalently, a separated object.*

Proposition 15.4.3 *Let $(\mathbf{C}, M)$ be a pCCC with dominance Σ. Then:*
(1) Morphisms preserve the intrinsic preorder.
(2) Σ-objects are closed under subobjects.
(3) Moreover, if $(\mathbf{C}, M)$ has enough points and a is an object then Σ^a is a Σ-object.

PROOF. (1) Let $f : a \to b$ and $x, y : a$. Suppose $x \leq_a y$, then given any $p : b \to \Sigma$ we have by hypothesis $p \circ f \circ x \leq p \circ f \circ y$, since $p \circ f : a \to \Sigma$. Hence $f \circ x \leq_b f \circ y : b$.

(2) Let $m : a \to b$ be a mono and b be a Σ-object. If x and y are two distinct points in a then $m \circ x$ and $m \circ y$ are two distinct points in b. Hence, since b is a Σ-object, they are separable by a morphism $p : b \to \Sigma$. Then the morphism $p \circ m$ separates the points x and y.

(3) If $f, g : \Sigma^a$ and $f \neq g$ then, by the enough point assumption, there is a $x : a$ such that $\neg(f \circ x \cong g \circ x)$. Take $\lambda h : \Sigma^a . h \circ x : \Sigma^a \to \Sigma$ as separator for f and g. □

Partiality is explicitly given in a pca D, and by generalizing basic facts of recursion theory (i.e. r.e. sets are exactly the domains of computable functions) it also provides a notion of semi-computable predicate on D. We elaborate this point in the following.

Definition 15.4.4 *Let D be a pca, for any $d \in D$ let $dom(d) = \{e \in D \mid de\downarrow\}$. Then we define a collection of semi-computable predicates on D as follows:*

$$\Sigma(D) = \{dom(d) \mid d \in D\} \ .$$

The collection of predicates $\Sigma(D)$ induces a refinement preorder on D defined as (this is the untyped intrinsic preorder):

$$d \leq_D e \ \text{ iff } \ \forall W \in \Sigma(D)\,(d \in W \Rightarrow e \in W) \ .$$

We observe that the operation of application preserves this preorder:

$$\forall e \in D\,(d \leq_D d' \Rightarrow ed \leq_D ed') \ .$$

Moreover, if the pca is not total then $\Sigma(D)$ can be seen as a basis for a topology as: (1) $\emptyset, D \in \Sigma(D)$, taking respectively the always divergent and always convergent morphism. (2) If $W, W' \in \Sigma(D)$ then $W \cap W' = dom(\lambda^* d.(\lambda^* x.\lambda^* y.c)(ed)(e'd)) \in \Sigma(D)$, where $c \in D$.

We show that given any per, say A, $\Sigma(D)$ induces a collection, say $\Sigma(A)$, of semi-computable predicates on A. From this structure it is easy to obtain a family $\mathcal{M}_D$ of admissible monos on $\mathbf{per}_D$ that turns the category into a pCCC.

Definition 15.4.5 *Let $A \in \mathbf{per}_D$. Then we define:*

$$\Sigma(A) = \{B \in \mathbf{per}_D \mid [B] \subseteq [A] \text{ and } \exists W \in \Sigma(D)\,(|A| \cap W = |B|)\} \ .$$

In other words B belongs to $\Sigma(A)$ if the equivalence classes in B form a subset of those in A and there is a set $W \subseteq \Sigma(D)$ that separates $[B]$ from the other equivalence classes in $[A]$.

Proposition 15.4.6 *Let D be a non-total pca. Then $\Sigma(A)$ enjoys closure properties analogous to $\Sigma(D)$: (1) $\emptyset, A \in \Sigma(A)$, and (2) If $B, B' \in \Sigma(A)$ then $B'' \in \Sigma(A)$, where B'' is the per corresponding to the partial partition $[B] \cap [B']$.*

PROOF HINT. By applying the related properties of $\Sigma(D)$. □

Remark 15.4.7 *We observe that if $d \leq_D d'$, $A \in \mathbf{per}_D$, and $d, d' \in |A|$ then a fortiori $[d]_A \leq_A [d']_A$. Suppose $B \in \Sigma(A)$ and $[d]_A \in [B]$, then there is a $W \in \Sigma(D)$ such that $|A| \cap W = |B|$. But by hypothesis $d' \in W$ and therefore $[d']_A \in [B]$. This fact corresponds to the intuition that if two elements cannot be separated in the* type free *universe of the realizability structure D then a fortiori they cannot be separated in the typed structure of per's.*

Definition 15.4.8 *Define $\mathcal{M}_D$ as the following family of monos:*

$$m : A' \to A \in \mathcal{M}_D(A) \ \text{ iff } \ A' \in \Sigma(A) \text{ and } m \text{ is the inclusion morphism} \ .$$

Note that the morphism m is realized by the identity. It is easy to check that this collection of monos is indeed admissible. The conditions for identity and composition are clear. Let us consider the case for the pullbacks. Assume $f : A \to B$ and $m : C \to B$ with ϕ realizer of f and $|B| \cap dom(\psi) = |C|$. To construct the pullback consider $W' = dom(\lambda^* d.\psi(\phi d))$ and the related admissible subobject of A. We leave to the reader the proof of the following propositions.

Proposition 15.4.9 *The category* $(\mathbf{per}_D, \mathcal{M}_D)$ *of per's and partial morphisms is equivalent to the category* $\mathbf{pper}_D$ *defined as follows:*

$$\begin{array}{ll} Ob\mathbf{pper}_D & = Ob\mathbf{per}_D \\ \mathbf{pper}_D[A,B] & = \{f : [A] \rightharpoonup [B] \mid \exists \phi \in D\, \forall d \\ & \quad d\, A\, d \Rightarrow ((\phi d)\downarrow \Leftrightarrow f([d]_A)\downarrow) \text{ and } ((\phi d)\downarrow \Rightarrow \phi d \in f([d]_A))\} \ . \end{array}$$

Proposition 15.4.10 *The category* $(\mathbf{per}_D, \mathcal{M}_D)$ *is a pCCC. The partial exponent is defined as:*

$$f\, pexp(A,B)\, g \quad \textit{iff} \quad \forall d, e\, (d\, A\, e \Rightarrow f d \cong_B g e)$$

where $\cong_B$ *is Kleene equality relativized to* B*, namely* $t \cong_B s$ *iff* $(t\downarrow \Leftrightarrow s\downarrow)$ *and* $(t\downarrow \Rightarrow tBs)$*.*

We remark that the category $\mathbf{per}_D$ has enough points. The terminal object is any per with one equivalence class, say $1 = D \times D$. The dominance is $\Sigma = 1 \rightharpoonup 1 = \{\bot, \top\}$, where $\bot = \{d \in D \mid \forall e\, (de\not\downarrow)\}$, and $\top = \{d \in D \mid \forall e\, (de\downarrow)\}$. We can then specialize definition 15.4.2 as follows.

Definition 15.4.11 *The category* $\mathbf{\Sigma per}_D$ *is the full subcategory of* $\mathbf{per}_D$ *whose objects are* Σ*-objects.*

Proposition 15.4.3 can be used to establish some elementary facts about $\mathbf{\Sigma per}_D$.

Theorem 15.4.12 *The category* $\Sigma\mathbf{per}_D$ *is a full reflective subcategory of* $\mathbf{per}_D$*.*

PROOF. The simple idea for obtaining a Σper $L_\Sigma(A)$ from the per A is to collapse equivalence classes that cannot be separated by $\Sigma(A)$. Given a per A and the intrinsic preorder $\leq_A$ we define an equivalence relation, $\approx_A$, on $[A]$ as:

$$[d]_A \approx_A [e]_A \ \text{ iff } \ d, e \in |A| \text{ and } [d]_A \leq_A [e]_A \text{ and } [e]_A \leq_A [d]_A \ .$$

Let the reflector $L_\Sigma : \mathbf{per} \to \Sigma\mathbf{per}$ be as follows:

$$\begin{array}{ll} dL_\Sigma(A)e \ \text{ iff } \ [d]_A \approx_A [e]_A & \text{for } A \in \mathbf{per} \\ L_\Sigma(f)([d]_{L_\Sigma(A)}) = f([d]_A) & \text{for } f : A \to B \ . \end{array}$$

One can easily verify that: (1) $L_\Sigma(A)$ is a Σper. Actually it is the least Σper containing A (as a relation). (2) $d\, L_\Sigma(A)\, e$ implies $f([d]_A) = f([e]_A)$, as B is separated. (3) Every morphism from a per A to a Σper B can be uniquely extended to a morphism from $L_\Sigma(A)$ to B. From these facts it is easy to exhibit the natural isomorphism of the adjunction. □

The following corollary summarizes our progress. We have managed to build a full sub-category of per's that has the same closure properties of $\mathbf{per}_D$, and moreover has an intrinsic notion of partial order that will turn out to be useful in the interpretation of recursion.

Corollary 15.4.13 *The category* $\Sigma\mathbf{per}_D$ *is cartesian closed and it has all limits and colimits of* $\mathbf{per}_D$.

PROOF. The existence of limits and colimits is guaranteed by the reflection. Let us check that $\Sigma\mathbf{per}_D$ is closed under the usual definition of exponent in $\mathbf{per}_D$. Suppose $B \in \Sigma\mathbf{per}_D$ and $f, g : A \to B$. Suppose that f and g are distinct, then there is a point $x : A$ such that fx, gx are distinct and, by hypothesis, separable by means of $k : B \to \Sigma$. Then the morphism $\lambda h : A \to B.k(hx)$ separates f and g. □

15.5 Complete per's

We are interested in finding an analogous of the notion of ω-completeness in a realizability framework. In the first place we need an object N that can play the role of the natural numbers. More precisely a *natural number object* (nno) is a diagram $1 \xrightarrow{0} N \xrightarrow{s} N$ that is initial among all diagrams of the shape: $1 \xrightarrow{x} A \xrightarrow{f} A$. In this section we work over Kleene's pca $(\omega, \bullet)$ and we define as nno:

$$N = \{\{n\} \mid n \in \omega\}\ .$$

In particular we shall make use of the fact that for $K = \{n \mid nn \downarrow\}$ and $O = \{K, K^c\}$, $\{K^c\} \notin \Sigma(O)$.

We will concentrate on (N-)complete Σper's, that is Σper's such that any ascending sequence on them, that is definable as a morphism in the category, has a lub with respect to the intrinsic order.

When restricting the attention to complete Σper it is possible to prove a variant of Myhill-Shepherdson's theorem (see chapter 1) asserting that all morphisms preserve lub's of chains. This will arise as a corollary of the fact that for any per A the elements of $\Sigma(A)$ are Scott opens.

A corollary of this result is that the full subcategory of complete, separated per's can be seen as a sort of pre-O-category in that the morphisms are partially

ordered, there are lub's of *definable* chains, and the operation of composition preserves this structure.

When stating the completeness condition for a Σper A we will only be interested in the existence of the lub's of the chains, $\chi : N \to A$, that are definable as morphisms from the nno N to A.

Definition 15.5.1 *We write $\chi : AS(A)$ (AS for ascending sequence) if*

$$\chi : N \to A \text{ and } \forall n : N\,(\chi n \leq_A \chi(n+1)) \ .$$

Remark 15.5.2 *Observe that whenever we select a subset of the equivalence classes of a (separated) per we can naturally consider it as a (separated) per. For example $AS(A)$ is a subset of $[N \to A]$ and $U \in \Sigma(A)$ is a subset of $[A]$.*

According to a constructive reading the existence of the lub of every ascending sequence implies the existence of a method to find this lub given a realizer for the sequence. Indeed as soon as we consider the problem of the closure of the collection of N-complete objects with respect to the function space constructor it becomes important to have a realizer that uniformly, for every ascending sequence of a given type, computes the lub (we refer to [Pho90] for more information on the closure properties of this category). This motivates the following definition.

Definition 15.5.3 *A separated per A is* complete *if $\forall \chi : AS(A)\, \exists \bigvee_A \chi$, where the existence of lub has to be interpreted constructively, that is:*

$$\exists \sigma_A : AS(A) \to A\, \forall \chi : AS(A)\, (\sigma_A(\chi) = \bigvee_A \chi) \ .$$

Since the morphism σ_A, if it exists, is uniquely determined we will simply indicate with A rather than with (A, σ_A) a complete separated per.

Remark 15.5.4 *The category of complete separated per's is non-trivial as every separated object A in which all elements are incomparable is complete (one can define $\sigma_A = \lambda \chi : AS(A).\chi(0)$, where 0 is the zero of the nno, as every ascending sequence is constant).*

Remark 15.5.5 *The definition of completeness highlights the difference between a classical set-theoretical definition (say in a system like ZF) and a constructive one. When working in a realizability universe it is a good habit to read definitions and theorems constructively. This approach will not be pursued in this introductory chapter, the problem being that a rigorous exposition requires some background on the* internal *logic of the* effective topos, *basically a higher order intuitionistic type theory that includes principles like the countable axiom of choice (AC_ω), the computability of all the morphisms on natural numbers (Church Thesis), the Uniformity Principle, and Markov Principle (see [Hyl82]).*

Definition 15.5.6 *Let A be a per. A subset U of $[A]$ is a Scott open (cf. definition 1.2.1) and we write $U \in \tau(A)$ iff*
(1) $\forall x, y : A\,(x : U$ and $x \leq_A y \Rightarrow y : U)$, and
(2) $\forall \chi : AS(A)\,(\exists \bigvee_A \chi : U \Rightarrow \exists n : N(\chi n : U))$.

Note that this definition makes sense in any preorder. It is immediate to check that $\tau(A)$ defines a topology over $[A]$.

Theorem 15.5.7 *If A is a separated, complete per and $U \in \Sigma(A)$ then $U \in \tau(A)$.*

PROOF. The first condition of upward closure follows by the definition of intrinsic order. Take $x : U$ and suppose $x \leq_A y$. Then $y : U$ as:

$$x \leq_A y \text{ iff } \forall U \in \Sigma(A)\,(x : U \Rightarrow y : U)\ .$$

The proof of the second condition takes advantage of the specific recursion-theoretical character of the pca $(\omega, \bullet)$, indeed the following argument is a keyvault of the theory.

Consider the set $K = \{n \mid nn \downarrow\}$ and the per $O = \{K, K^c\}$. We observe $\{K\} \in \Sigma(O)$ and $\{K^c\} \notin \Sigma(O)$. The predicate $nn \downarrow i$ means that the computation nn of the n-th machine applied to the input n will stop in at most i steps. This is a decidable predicate.

Now let us proceed by contradiction assuming there is $\chi : AS(A)$ such that:

$$\exists \bigvee_A \chi : U \text{ and } \forall n : N\ \neg(\chi n : U)\ .$$

The crucial idea is to build a function $h : O \to A$ mapping K to χn, for some n, and K^c to $\bigvee_A \chi$. By the pullback condition we derive the contradiction:

$$h^{-1}(U) = \{K^c\} \in \Sigma(O)\ .$$

For any n we define an ascending sequence $\lambda i.c(n,i) : N \to A$. In the following $\mu k \leq i.nn \downarrow k$ is the least element $k \leq i$ such that $nn \downarrow k$.

$$c(n,i) = \begin{cases} \chi i & \text{if } \neg(nn \downarrow i) \\ \chi(\mu k \leq i.nn \downarrow k) & \text{otherwise}\ . \end{cases}$$

We observe that for any given n if $n \in K$ then $\lambda i.c(n,i)$ coincides with the ascending sequence χ up to the first k such that $nn \downarrow k$ and then becomes definitely constant; on the other hand if $n \in K^c$ then $\lambda i.c(n,i)$ coincides with χ. We note that $\lambda i.c(n,i) : AS(A)$. Using the existence of a morphism σ_A that uniformly realizes the lub of ascending sequences we define a morphism $h : O \to A$ such that:

$$h([n]_O) = \sigma_A(\lambda i.c(n,i))\ .$$

We have just observed $h([n]_O) = \bigvee_A \chi$ if $n \in K^c$ and $h([n]_O) \in \{\chi n \mid n : N\}$ otherwise, from this we can obtain the desired contradiction. □

Remark 15.5.8 *For the logically inclined reader we mention that this proof by contradiction can be turned into a* constructive proof *via Markov Principle.*

Definition 15.5.9 *Let A, B be separated per's. We say that $f : A \to B$ preserves chains if*

$$\forall \chi : AS(A)\,(\exists \bigvee_A \chi : A \Rightarrow (\exists \bigvee_B f \circ \chi : B \text{ and } f(\bigvee_A \chi) = \bigvee_B f \circ \chi)) .$$

Proposition 15.5.10 *Suppose A, B are complete separated per's. Then any morphism $f : A \to B$ preserves chains and it is Scott continuous.*

PROOF. (1) Consider $\chi : AS(A)$ and assume $\exists \bigvee_A \chi : A$. In order to show $\exists \bigvee_B f \circ \chi = f(\bigvee_A \chi)$ we prove that for any upper bound $y : B$ of $f \circ \chi : AS(B)$ we have $f(\bigvee_A \chi) \leq_B y$. We recall that:

$$f(\bigvee_A \chi) \leq_B y \text{ iff } \forall U \in \Sigma(B)\,(f(\bigvee_A \chi) : U \Rightarrow y : U) .$$

Now $U \in \Sigma(B)$ implies, by the pullback condition of admissible domains, $f^{-1}(U) \in \Sigma(A)$, that is by theorem 15.5.7, $f^{-1}(U) \in \tau(A)$. Since $f(\bigvee_A \chi) : U$, we have $\bigvee_A \chi : f^{-1}(U)$, that implies by the definition of open set $\exists n : N\,(\chi n : f^{-1}(U))$. Therefore $\exists n : N\,(f(\chi n) : U)$, and this implies $y : U$.

(2) We take $U \in \tau(B)$ and we consider $f^{-1}(U)$. This is upward closed by the fact that f is monotonic. Moreover, let $\chi : AS(A)$ and suppose $\exists \bigvee_A \chi : f^{-1}(U)$. Then by hypothesis $f(\bigvee_A \chi) = \bigvee_B f \circ \chi : U$. Therefore $\exists n : N(f(\chi n) : U)$ i.e. $\chi n : f^{-1}(U)$. □

Extensional per's. * So far the theory has been developed in a rather synthetic and abstract way. To use the theory in practice it is often useful (if not necessary) to have a *concrete* presentation of the denotational model. For instance we would like to characterize the order on function spaces, to compute lub's explicitly,... In the following we introduce a category of *extensional* per's for which we can provide answers to these questions (an even more concrete category based on a different pca will be presented in the next section). The initial idea is to look at per's of the shape Σ^A. First we need to develop a few notions.

Definition 15.5.11 (Σ-linked) *A per A is Σ-linked if for all $x, y \in [A]$,*

$$x \leq_A y \Rightarrow \exists f : \Sigma \to A\,(f\bot = x \text{ and } f\top = y) .$$

We note that if $f\bot = x$ and $f\top = y$ then $x \leq_A y$, by monotonicity. We shall prove in proposition 15.5.20 that all complete separated per's are Σ-linked), but there are separated per's which are not Σ-linked. The proof of this fact relies on a rather deep recursion-theoretical result.

Definition 15.5.12 *Let $X, Y \subseteq \omega$ be sets. We say that X is* many-reducible *to Y and write $X \leq_m Y$ if there is a total recursive function f such that:*

$$x \in X \text{ iff } f(x) \in Y \ .$$

The following proposition is due to Post (a proof can be found in [Soa87]).

Proposition 15.5.13 *Any r.e. set X is many reducible to the set $K = \{n \mid nn \downarrow\}$. There is an r.e., non-recursive set to which K cannot be many-reduced.*

Proposition 15.5.14 *(1) If X is a r.e. non-recursive set then $A = \{X, X^c\}$ is a separated per where $X^c <_A X$.*
(2) The dominance Σ is isomorphic to the separated per $\{K, K^c\}$.
(3) There is an r.e. set such that $\{X, X^c\}$ is not Σ-linked.

Proof. (1) If a partial morphism from $\{X, X^c\}$ to the terminal object converges on X^c and diverges on X then it contradicts the hypothesis that X is not recursive.

(2) We recall that $\Sigma = \{\bot, \top\}$ where $\bot = \{n \mid \forall m\, nm \not\downarrow\}$ and $\top = \{n \mid \forall m\, nm \downarrow\}$. From Σ to $\{K^c, K\}$ consider the morphism realized by the identity. In the other direction consider the map realized by $\lambda^* n.\lambda^* m.nn$.

(3) First we observe for $X, Y \subseteq \omega$:

$$X \leq_m Y \text{ iff } \exists h : \{X, X^c\} \to \{Y, Y^c\} \text{ mono such that } h(X) = Y \ .$$

Then pick up a r.e., non-recursive set X to which K cannot be many-reduced. The separated per $\{X, X^c\}$ is not Σ-linked. □

Hence, we can build two separated per's having the same order as Sierpinski space that are not isomorphic!

Exercise 15.5.15 *Show that there is a set $X \subset \omega$ such that $\{X, X^c\}$ is not separated.*

Definition 15.5.16 *Let $A = \Pi_{i \in I} A_i$ be a product in* **per** *with projections $\{\pi_i\}_{i \in I}$. We say that A is ordered* pointwise *if*

$$x \leq_A y \text{ iff } \forall i \in I\, (\pi_i \circ x \leq_{A_i} \pi_i \circ y) \ .$$

Proposition 15.5.17 *(1) The dominance Σ is Σ-linked.*
(2) Let $A = \Pi_{i \in I} A_i$ be a product of Σ-linked per's with projections $\{\pi_i\}_{i \in I}$. Then A is ordered pointwise and Σ-linked.
(3) If A is Σ-linked and $[B] \subseteq [A]$ then B is Σ-linked and the order on B is the restriction of the order on A.

Proof. (1) Take the identity function.

(2) Consider $x, y : \Pi_{\in I} A_i$ such that $\forall i \in I\, (\pi_i \circ x \leq_{A_i} \pi_i \circ y)$ (x is pointwise smaller than y). By hypothesis:

$$\forall i \in I\, \exists f_i : \Sigma \to A_i\, (f_i \circ \bot = \pi_i \circ x \text{ and } f_i \circ \top = \pi_i \circ y) \ .$$