Initial algebras and final coalgebras consisting of nondeterministic finite trace strategies

Bowler, Nathan; Levy, Paul Blain; Plotkin, Gordon

DOI:
10.1016/j.entcs.2018.11.003

Document Version
Publisher's PDF, also known as Version of record

Citation for published version (Harvard):
Bowler, N, Levy, PB & Plotkin, G 2018, Initial algebras and final coalgebras consisting of nondeterministic finite trace strategies. in S Staton (ed.), *Proceedings of the 34th Conference on the Mathematical Foundations of Programming Semantics (MFPS XXXIV).* Electronic Notes in Theoretical Computer Science, vol. 341, pp. 23-44, 34th Conference on the Mathematical Foundations of Programming Semantics (MFPS 2018), Halifax, Canada, 6/06/18. https://doi.org/10.1016/j.entcs.2018.11.003

Available online at www.sciencedirect.com

ScienceDirect

Electronic Notes in Theoretical Computer Science 341 (2018) 23–44

Electronic Notes in Theoretical Computer Science

www.elsevier.com/locate/entcs

Initial Algebras and Final Coalgebras Consisting of Nondeterministic Finite Trace Strategies

Nathan Bowler[1]

Department of Mathematics, Universität Hamburg, Germany

Paul Blain Levy[2]

School of Computer Science, University of Birmingham, UK

Gordon Plotkin[3]

LFCS, University of Edinburgh, UK

Abstract

We study programs that perform I/O and finite or countable nondeterministic choice, up to finite trace equivalence. For well-founded programs, we characterize which strategies (sets of traces) are definable, and axiomatize trace equivalence by means of commutativity between I/O and nondeterminism. This gives the set of strategies as an initial algebra for a polynomial endofunctor on semilattices. The strategies corresponding to non-well-founded programs constitute a final coalgebra for this functor.

Keywords: final coalgebra, nondeterministic strategies, trace, algebraic effects, semilattices

1 Introduction

This paper is about nondeterministic programs that perform I/O. To illustrate the ideas, let us consider the following (infinitary) imperative language:

[1] Email: Nathan.Bowler@uni-hamburg.de

[2] Email: P.B.Levy@cs.bham.ac.uk

[3] Email: gdp@inf.ed.ac.uk

https://doi.org/10.1016/j.entcs.2018.11.003

$$\begin{aligned} M, N ::= \ & \texttt{Age?}(M_n)_{n\in\mathbb{N}} \\ & \mid \texttt{Happy?}(M, N) \\ & \mid \texttt{Continue?}(M) \mid \texttt{Bye}_n \\ & \mid M \texttt{ or } N \end{aligned}$$

The meaning is as follows.

- The command $\texttt{Age?}(M_n)_{n\in\mathbb{N}}$ prints `Age?` and pauses. If the user then enters n, it executes M_n.
- The command $\texttt{Happy?}(M, N)$ prints `Happy?` and pauses. If the user then enters `Yes` or `No`, it executes M or N respectively.
- The command $\texttt{Continue?}(M)$ prints `Continue?` and pauses. If the user then enters `Yes`, it executes M.
- The command $\texttt{Bye}_n$ prints $\texttt{Bye}_n$ and pauses. No further input is possible.
- The command M `or` N nondeterministically chooses to execute M or N.

A *play* is an alternating sequence of outputs and inputs, e.g.

`Happy?Yes.Age?93.Age?27.Continue?Yes.Happy?`

A command's *traces* are the plays it may give rise to. For example, let

$$M_0 \stackrel{\text{def}}{=} \texttt{Happy?}(\texttt{Bye}_3, \texttt{Bye}_5 \texttt{ or } \texttt{Continue?}(\texttt{Bye}_6))$$

It has the following *passive-ending* traces (i.e. ones ending with execution paused):

$\texttt{Happy?}$

$\texttt{Happy?Yes.Bye}_3$

$\texttt{Happy?No.Bye}_5$

$\texttt{Happy?No.Continue?}$

$\texttt{Happy?No.Continue?Yes.Bye}_6$

and the following *active-ending* traces (i.e. ones ending with the program executing):

ε (the empty play)

$\texttt{Happy?Yes.}$

$\texttt{Happy?No.}$

$\texttt{Happy?No.Continue?Yes.}$

The following command

$$M_1 \stackrel{\text{def}}{=} \texttt{Happy?}(\texttt{Bye}_3, \texttt{Bye}_5) \texttt{ or } \texttt{Happy?}(\texttt{Bye}_3, \texttt{Continue?}(\texttt{Bye}_6))$$

has the same traces as M_0, i.e. these commands are *trace equivalent* (though not bisimilar). The following questions naturally arise:

(i) Given a set σ of plays, under what conditions is σ the trace set of some command?

(ii) Can we give an axiomatic theory of trace equivalence? (Cf. the axiomatic analysis of process equivalences in [1,25].)

This paper's first contribution is to answer these questions. The answer to question (ii) is surprisingly simple: we take the ordinary theory of `or` (commutativity, associativity and idempotency), together with the fact that each I/O operation *commutes* with `or`. For example:

$$\mathtt{Age}(M_n)_{n\in\mathbb{N}} \ \mathtt{or} \ \mathtt{Age}(M'_n)_{n\in\mathbb{N}} \ = \ \mathtt{Age}(M_n \ \mathtt{or} \ M'_n)_{n\in\mathbb{N}}$$

We give our results not only for the language above but also for some variations, as we shall now explain. The language has two parts—I/O and nondeterminism—and each can be varied.

(i) The I/O part is determined by a *signature*, a collection of operations each with a specified arity—a set of argument indices. The language above has four I/O operations—Age, Happy, Continue and Bye—of respective arity $\mathbb{N}$, $\{\mathsf{Yes}, \mathsf{No}\}$, $\{\mathsf{Yes}\}$ and $\emptyset$. Our results apply no matter what I/O signature is used to generate the language.

(ii) We may include commands of the form $\bigvee_{n\in\mathbb{N}} M_n$. This command nondeterministically chooses $n \in \mathbb{N}$ and then executes M_n.

In the second part of the paper (Section 6) we consider non-well-founded program behaviours, up to (finite) trace equivalence. We see that the familiar duality—initial algebra for well-founded behaviours vs final coalgebra for non-well-founded ones—arises also in the setting of finite traces.

The significance of these results is shown by their connection to several areas of semantics.

Effects and monads. I/O operations and nondeterministic choice are examples of *computational effects*. A collection of effects is often described by a monad on **Set** [20], which can sometimes be presented by a simple theory [21]. Each of our combination of effects give rise to such a monad on **Set** corresponding to programs modulo trace equivalence, which is moreover a *tensor* of the monads for I/O and nondeterminism [3,6,7,13].

Game semantics. A program in the language above may be seen as playing a game (Figure 1) with one active position (indicating that the program is executing), and several passive positions (indicating that execution is paused). The games that arise in game semantics may have several active *and* several passive positions [19]. A different terminology is used: with outputs called "P-moves" and inputs called "O-moves". Nonetheless, where finite traces are studied, the same notions of nondeterministic strategies [8,9] may be used, and our results characterize these strategies for these more general games.

Coalgebraic traces. Several coalgebraic accounts of traces have appeared [10,15,16]. Our account (though it does not subsume these) has the novelty of including both output and input actions. We briefly compare in Section 7.

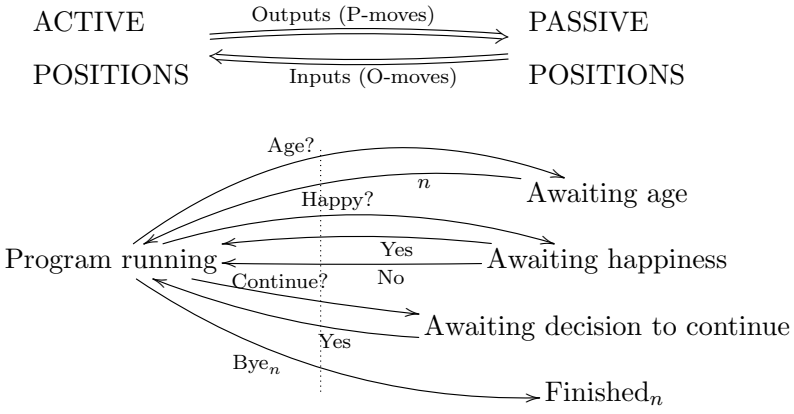


Fig. 1. The game that programs play against the user.

Acknowledgements

We thank V. Vignudelli for explaining the notion of bisimulation between distributions and suggesting a link to trace equivalence.

2 Preliminaries

2.1 Semilattices

Given a set X, we write $\mathcal{P}X$ for the set of subsets, $\mathcal{P}_{\mathsf{f}}X$ for the set of finite subsets, $\mathcal{P}_{\mathsf{c}}X$ for the set of countable subsets. We write X, we write $\mathcal{P}^{+}X$ for the set of inhabited (nonempty) subsets and likewise $\mathcal{P}^{+}_{\mathsf{f}}X$ and $\mathcal{P}^{+}_{\mathsf{c}}X$.

A *semilattice* is a poset $(A, \leqslant)$ with all binary joins. It is *bounded* when it has a least element, an *ω-semilattice* when every countable subset has a supremum, an *almost complete semilattice* when every inhabited subset has a supremum, and a *complete semilattice* when every subset has a supremum.

A *map* $A \longrightarrow B$ of semilattices is a monotone map that preserves binary join. A map of bounded semilattices must preserve the least element, a map of ω-semilattices must preserve countable joins, a map of almost complete semilattices must preserve suprema of inhabited subsets, and a map of complete semilattices must preserve arbitrary suprema.

Rather than describing a semilattice posetally as above, we may also describe it equationally: a set A with a binary operation $\vee$ that is commutative, associative and idempotent. A posetal semilattice $(A, \leqslant)$ gives an equational one by setting $x \vee y$ to be the join of x and y. Conversely, an equational semilattice $(A, \vee)$ gives a posetal one by setting $x \leqslant y$ when $x \vee y = y$. These constructions are inverse. A function between semilattices is a map of equational semilattices (i.e. preserves $\vee$) iff it is a map of posetal semilattices (i.e. is monotone and preserves binary join).

Continuing the equational style:

- a bounded semilattice is a semilattice $(A, \vee)$ with a neutral element $\bot$

- an ω-semilattice is a semilattice $(A, \vee)$ with an operation $\bigvee\colon A^\omega \longrightarrow A$ satisfying

$$x \vee \bigvee_{n\in\mathbb{N}} y_n = \bigvee_{n\in\mathbb{N}} (x \vee y_n)$$
$$\bigvee_{n\in\mathbb{N}} x = x$$
$$\bigvee_{n\in\mathbb{N}} x_n = x_m \vee \bigvee_{n\in\mathbb{N}} x_n \quad (m \in \mathbb{N})$$

A more abstract view: the category of semilattices is the Eilenberg-Moore category for the monad $\mathcal{P}_{\mathsf{f}}^+$ on **Set**. Likewise the category of ω-semilattices for $\mathcal{P}_{\mathsf{c}}^+$, the category of almost complete semilattices for $\mathcal{P}^+$, etc.

2.2 Language

Definition 2.1

(i) A *signature* consists of a set K of *operations*, and for each operation $k \in K$, a set $\mathsf{Ar}(k)$ of *argument indices*.

(ii) A *transition system* over a signature $(\mathsf{Ar}(k))_{k\in K}$ consists of a set X and function $\zeta\colon X \longrightarrow \mathcal{P} \sum_{k\in K} \prod_{i\in\mathsf{Ar}(k)} X$. We write $x \overset{k}{\Rightarrow} (y_i)_{i\in\mathsf{Ar}(k)}$ when $(k, (y_i)_{i\in\mathsf{Ar}(k)}) \in \zeta x$.

Informally $x \overset{k}{\Rightarrow} (y_i)_{i\in\mathsf{Ar}(k)}$ means that x outputs k and pauses, and if the user then inputs i, it executes y_i.

For the sequel, we fix a signature $S = (\mathsf{Ar}(k))_{k\in K}$. The set of commands is defined inductively by the grammar

$$M, N ::= \quad \mathtt{Req}\, k?(M_i)_{i\in\mathsf{Ar}(k)} \;\mid\; M \;\mathtt{or}\; N$$

The set of commands forms a transition system. The transition relation $M \overset{k}{\Rightarrow} (N_i)_{i\in\mathsf{Ar}(k)}$ is inductively defined as follows.

$$\frac{}{\mathtt{Req}\, k?(M_i)_{i\in\mathsf{Ar}(k)} \overset{k}{\Rightarrow} (M_i)_{i\in\mathsf{Ar}(k)}} \qquad \frac{M \overset{k}{\Rightarrow} (N_i)_{i\in\mathsf{Ar}(k)}}{M \;\mathtt{or}\; M' \overset{k}{\Rightarrow} (N_i)_{i\in\mathsf{Ar}(k)}} \qquad \frac{M' \overset{k}{\Rightarrow} (N_i)_{i\in\mathsf{Ar}(k)}}{M \;\mathtt{or}\; M' \overset{k}{\Rightarrow} (N_i)_{i\in\mathsf{Ar}(k)}}$$

For countable nondeterminism, we extend the syntax as follows:

$$M ::= \quad \cdots \;\mid\; \bigvee_{n\in\mathbb{N}} M_n$$

and include the operational rule

$$\frac{M_n \overset{k}{\Rightarrow} (N_i)_{i\in\mathsf{Ar}(k)}}{\bigvee_{n\in\mathbb{N}} M_n \overset{k}{\Rightarrow} (N_i)_{i\in\mathsf{Ar}(k)}}\; n \in \mathbb{N}$$

Definition 2.2 A transition system (X, ζ) is

- *total* when, for all $x \in X$, the set ζx is inhabited
- *deterministic* when, for all $x \in X$, the set ζx has at most one element
- *finitely nondeterministic* when, for all $x \in X$, the set ζx is finite
- *countably nondeterministic* when, for all $x \in X$, the set ζx is countable
- *well-founded* [4] when there is no infinite sequence of transitions

$$x_0 \overset{k_0}{\Rightarrow} (y_i^0)_{i \in \mathsf{Ar}(k_0)} \quad y_{i_0}^0 = x_1 \overset{k_1}{\Rightarrow} (y_i^1)_{i \in \mathsf{Ar}(k_1)} \quad \cdots$$

Proposition 2.3

(i) *The set of finitely nondeterministic commands, as a transition system, is total, finitely nondeterministic and well-founded.*

(ii) *The set of countably nondeterministic commands, as a transition system, is total, countably nondeterministic and well-founded.*

Proof. We prove by induction on M that ζM is finite/countable and inhabited, and there is no infinite sequence of transitions from M. □

2.3 Bisimulation

Although it is not used in the sequel, we briefly look at bisimulation.

Definition 2.4 A *bisimulation* on a transition system (X, ζ) is a a relation $\mathcal{R}$ on X such that $x \mathcal{R} x'$ implies that, if $x \overset{k}{\Longrightarrow} (y_i)_{i \in \mathsf{Ar}(k)}$ then there exists $(y_i')_{i \in \mathsf{Ar}(k)}$ such that $x' \overset{k}{\Longrightarrow} (y_i')_{i \in \mathsf{Ar}(k)}$ and $\forall i \in \mathsf{Ar}(k).\, y_i \mathcal{R} y_i'$, and vice versa. The greatest bisimulation is called *bisimilarity*.

For example, the commands M_0 and M_1 in Section 1 are not bisimilar. We axiomatize bisimilarity as follows.

Definition 2.5 *Basic equivalence*, written $\equiv$, is the least congruence on commands sastifying the semilattice laws:

$$M \texttt{ or } N \equiv N \texttt{ or } M \tag{1}$$

$$(M \texttt{ or } N) \texttt{ or } P \equiv M \texttt{ or } (N \texttt{ or } P) \tag{2}$$

$$M \texttt{ or } M \equiv M \tag{3}$$

Also, for the countably nondeterministic language, the ω-semilattice laws:

$$M \texttt{ or } \bigvee_{n \in \mathbb{N}} M_n \equiv \bigvee_{n \in \mathbb{N}} (M \texttt{ or } M_n) \tag{4}$$

$$\bigvee_{n \in \mathbb{N}} M \equiv M \tag{5}$$

$$\bigvee_{n \in \mathbb{N}} M_n \equiv M_m \texttt{ or } \bigvee_{n \in \mathbb{N}} M_n \quad (m \in \mathbb{N}) \tag{6}$$

[4] Cf. the notion of well-founded coalgebra [2,24].

Proposition 2.6 *For both the finitely and countably nondeterministic languages, basic equivalence is bisimilarity.*

Proof. Induction on $\equiv$ shows that $\equiv$ implies bisimilarity and that it is a bisimulation. □

2.4 Traces and Strategies

Our basic notion of interaction is as follows.

Definition 2.7 A *play* is a sequence $k_0, i_0, k_1, i_1, \ldots$ with $k_r \in K$ and $i_r \in \mathsf{Ar}(k_r)$. It is *active-ending*, *passive-ending* or *infinite* according as its length is even, odd or infinite.

As illustrated by M_0 and M_1 in Section 1, two commands are trace equivalent iff they have the same passive-ending traces; the active-ending traces are redundant. This motivates the following.

Definition 2.8

(i) A *nondeterministic finite trace strategy* is a set σ of passive-ending plays such that $sik \in \sigma$ implies $s \in \sigma$.

(ii) An active-ending play is *enabled* by σ when it is either ε (the empty play), or si for $s \in \sigma$.

Henceforth "strategy" means "nondeterministic finite trace strategy".

Definition 2.9 Let (X, ζ) be a transition system and $x \in X$. A play $k_0, i_0, \ldots$ is a *trace* of x when there is a sequence

$$x = x_0 \overset{k_0}{\Rightarrow} (y_i^0)_{i \in \mathsf{Ar}(k_0)} \quad y_{i_0}^0 = x_1 \overset{k_1}{\Rightarrow} (y_i^1)_{i \in \mathsf{Ar}(k_1)} \quad \cdots$$

The strategy consisting of the passive-ending traces of x is written $\mathsf{Traces}\, x$.

Clearly the active-ending traces of x are the plays enabled by $\mathsf{Traces}\, x$. By contrast, the infinite traces are, in general, not derivable from $\mathsf{Traces}\, x$. For example, in a non-well-founded extension of the language, the following commands

$$N_0 \overset{\text{def}}{=} \bigvee_{n \in \mathbb{N}} \mathtt{Continue}?^n\, (\mathtt{Bye}_3)\ \mathtt{or}\ \mathtt{Continue}^\omega$$

$$N_1 \overset{\text{def}}{=} \bigvee_{n \in \mathbb{N}} \mathtt{Continue}?^n\, (\mathtt{Bye}_3)$$

are trace equivalent, but the infinite play $(\mathtt{Continue?Yes}.)^\omega$ is a trace of N_0 and not of N_1. This cannot happen in a well-founded system, because there are no infinite traces. Nor can it happen in a finitely nondeterministic system, because of the following version of König's lemma.

Proposition 2.10 *For a finitely nondeterministic system $\zeta \colon X \longrightarrow \mathcal{P} \sum_{k \in K} \prod_{i \in \mathsf{Ar}(k)} X$ and $x \in X$, an infinite play $k_0, i_0, \ldots$ is a trace of x iff all its passive-ending prefixes are in $\mathsf{Traces}\, x$.*

Let us look at some useful operations on strategies. Firstly, we put strategies together as follows.

Definition 2.11 For $k \in K$ and a family of strategies $(\sigma_i)_{i \in \mathsf{Ar}(k)}$, we define the strategy

$$\mathsf{Req}\, k?(\sigma_i)_{i \in I} \stackrel{\text{def}}{=} \{(k)\} \cup \{k.i.s \mid i \in \mathsf{Ar}(k), s \in \sigma_i\}$$

Proposition 2.12 *On commands, we can give* Traces *compositionally:*

$$\begin{aligned} \mathsf{Traces}\, \mathsf{Req}\, k?(M_i)_{i \in \mathsf{Ar}(k)} &= \mathsf{Req}\, k?(\mathsf{Traces}\, M_i)_{i \in \mathsf{Ar}(k)} \\ \mathsf{Traces}\, (M \ \mathtt{or}\ N) &= \mathsf{Traces}\, M \cup \mathsf{Traces}\, N \\ \mathsf{Traces} \bigvee_{n \in \mathbb{N}} M_n &= \bigcup_{n \in \mathbb{N}} \mathsf{Traces}\, M_n \end{aligned}$$

Secondly, we decompose strategies as follows.

Definition 2.13 Let σ be a strategy. We write $\mathsf{Init}\, \sigma$ for the set of $k \in K$ such that $(k) \in \sigma$. For each such k and each $i \in \mathsf{Ar}(k)$, we define the strategy $\sigma/ki \stackrel{\text{def}}{=} \{s \mid k.i.s \in \sigma\}$.

3 Well-founded behaviour

3.1 Commuting Equivalence

We begin with the matter of axiomatizating trace equivalence. As we shall see, the appropriate axiomatization is as follows.

Definition 3.1 *Commuting equivalence*, written $\equiv_{\mathsf{c}}$, is the least congruence on commands that satisfies the semilattice laws (1)–(3) and commutativity between $\mathtt{Req}\, k?$ and $\mathtt{or}$:

$$\mathtt{Req}\, k?(M_i a \ \mathtt{or}\ N_i)_{i \in \mathsf{Ar}(k)} = \mathtt{Req}\, k?(M_i)_{i \in \mathsf{Ar}(k)} \ \mathtt{or}\ \mathtt{Req}\, k?(N_i)_{i \in \mathsf{Ar}(k)} \quad (k \in K) \tag{7}$$

Also, for the countably nondeterministic language, the ω-semilattice laws (4)–(6) and commutativity between $\mathtt{Req}\, k?$ and $\bigvee$:

$$\mathtt{Req}\, k?(\bigvee_{n \in \mathbb{N}} M_{i,n})_{i \in \mathsf{Ar}(k)} = \bigvee_{n \in \mathbb{N}} \mathtt{Req}\, k?(M_{i,n})_{i \in \mathsf{Ar}(k)} \quad (k \in K) \tag{8}$$

For example, the commands M_0 and M_1 in Section 1 are commuting equivalent.

Proposition 3.2 *(Soundness) If* $M \equiv_{\mathsf{c}} N$ *then* $\mathsf{Traces}\, M = \mathsf{Traces}\, N$.

Proof. This is proved by induction; the soundness of the laws follows from Proposition 2.12. For example:

$$\mathsf{Req}\, k?(\bigcup_{n \in \mathbb{N}} \sigma_{i,n})_{i \in \mathsf{Ar}(k)} = \{k\} \cup \{k.i.s \mid i \in \mathsf{Ar}(k), s \in \bigcup_{n \in \mathbb{N}} \sigma_{i,n}\}$$

$$\bigcup_{n \in \mathbb{N}} \mathsf{Req}\, k?(\sigma_{i,n})_{i \in \mathsf{Ar}(k)} = \bigcup_{n \in \mathbb{N}} (\{k\} \cup \{k.i.s \mid i \in \mathsf{Ar}(k), s \in \sigma_{i,n}\})$$

and these are equal since $\mathbb{N}$ is inhabited. Hence (8) is sound. □

We shall now prove completeness, i.e. the converse of Proposition 3.2. We use the following.

Lemma 3.3 *For every command M we have*

$$M \equiv_{\mathsf{c}} \bigvee_{k \in L} \mathtt{Req}\, k?(N_{k,i})_{i \in \mathsf{Ar}(k)}$$

for some $L \in \mathcal{P}^+_{\mathsf{f}} K$ and family of commands $(N_{k,i})_{k \in L, i \in \mathsf{Ar}(k)}$. For the countably nondeterministic language, $L \in \mathcal{P}^+_{\mathsf{c}} K$.

Proof. Induction on M. The case that $M = \mathtt{Req}\, k?(N_{k,i})_{i \in \mathsf{Ar}(k)}$ is obvious. In the case $M = \bigvee_{n \in \mathbb{N}} M_n$ we have $M_n \equiv_{\mathsf{c}} \bigvee_{k \in L_n} \mathtt{Req}\, k?(N_{n,k,i})_{i \in \mathsf{Ar}(k)}$. Then

$$\begin{aligned}
\bigvee_{n \in \mathbb{N}} M_n &\equiv_{\mathsf{c}} \bigvee_{n \in \mathbb{N}, k \in L_n} \mathtt{Req}\, k?(N_{n,k,i})_{i \in \mathsf{Ar}(k)} \\
&\equiv_{\mathsf{c}} \bigvee_{k \in \bigcup_{n \in \mathbb{N}} L_n} \bigvee_{n \in \mathbb{N} \,:\, k \in L_n} \mathtt{Req}\, k?(N_{n,k,i})_{i \in \mathsf{Ar}(k)} \\
&\equiv_{\mathsf{c}} \bigvee_{k \in \bigcup_{n \in \mathbb{N}} L_n} \mathtt{Req}\, k?\Big(\bigvee_{n \in \mathbb{N} \,:\, k \in L_n} N_{n,k,i} \Big)_{i \in \mathsf{Ar}(k)}
\end{aligned}$$

Likewise for the case $M = M_0 \;\mathtt{or}\; M_1$. □

Our completeness proof proceeds by characterizing *trace inclusion*, the preorder that relates M to N when $\mathsf{Traces}\, M \subseteq \mathsf{Traces}\, N$.

Proposition 3.4 *Write $M \leqslant_{\mathsf{c}} N$ when $M \;\mathtt{or}\; N \equiv_{\mathsf{c}} N$.*

(i) *$\leqslant_{\mathsf{c}}$ is a preorder.*

(ii) *$M \;\mathtt{or}\; N$ is a least upper bound of M and N*

(iii) *$\bigvee_{n \in \mathbb{N}}$ is a least upper bound of $(M_n)_{n \in \mathbb{N}}$.*

(iv) *$\mathtt{or}$ and $\bigvee_{n \in \mathbb{N}}$ and $\mathtt{Req}\, k?$ for $k \in K$ are all monotone, i.e. $\leqslant_{\mathsf{c}}$ is a precongruence.*

Proof. Parts (i)–(iii) follow the construction of a posetal semilattice from an equational semilattice in Section 2.1. Monotonicity of $\mathtt{or}$ and $\bigvee_{n \in \mathbb{N}}$ follow from the least upper bound property. Monotonicity of $\mathtt{Req}\, k?$ follows from (7), because the latter may be viewed as saying that $\mathtt{Req}\, k?$ is a map of equational semilattices and hence a map of posetal semilattices. □

Proposition 3.5 *$M \leqslant_{\mathsf{c}} N$ iff $\mathsf{Traces}\, M \subseteq \mathsf{Traces}\, N$.*

Proof. ($\Rightarrow$) follows from Proposition 3.2. We prove ($\Leftarrow$) by induction on M. The cases $M = M_0 \;\mathtt{or}\; M_1$ and $M = \bigvee_{n \in \mathbb{N}} M_n$ follow from the least upper bound property. Suppose $M = \mathtt{Req}\, k?(M_i)_{i \in \mathsf{Ar}(k)}$. Lemma 3.3 gives $N \equiv_{\mathsf{c}} \bigvee_{k \in L} \mathtt{Req}\, k?(N_{k,i})_{i \in \mathsf{Ar}(k)}$ and so

$$\mathsf{Traces}\,\mathsf{Req}\,k?(M_i)_{i\in\mathsf{Ar}(k)} \subseteq \mathsf{Traces}\bigvee_{k\in L}\mathsf{Req}\,k?(N_{k,i})_{i\in\mathsf{Ar}(k)}$$

$$\text{i.e. } \mathsf{Req}\,k?(\mathsf{Traces}\,M_i)_{i\in\mathsf{Ar}(k)} \subseteq \bigcup_{k\in L}\mathsf{Req}\,k?(\mathsf{Traces}\,N_{k,i})_{i\in\mathsf{Ar}(k)}$$

Thus $k \in L$, and for each $i \in \mathsf{Ar}(k)$ we have $\mathsf{Traces}\,M_i \subseteq \mathsf{Traces}\,N_{k,i}$ implying $M_i \leqslant_{\mathsf{c}} N_{k,i}$ by the inductive hypothesis. Hence

$$\begin{aligned} M &= \mathsf{Req}\,k?(M_i)_{i\in\mathsf{Ar}(k)} \\ &\leqslant_{\mathsf{c}} \mathsf{Req}\,k?(N_{k,i})_{i\in\mathsf{Ar}(k)} \\ &\leqslant_{\mathsf{c}} \textstyle\bigvee_{k\in L}\mathsf{Req}\,k?(N_{k,i})_{i\in\mathsf{Ar}(k)} \;\equiv_{\mathsf{c}} N \end{aligned}$$

□

Corollary 3.6 *$M \equiv_{\mathsf{c}} N$ iff $\mathsf{Traces}\,M = \mathsf{Traces}\,N$.*

3.2 Definability for finite nondeterminism

We shall now characterize which strategies are of the form $\mathsf{Traces}\,M$ for a finitely nondeterministic command M. We also give a second proof of completeness of $\equiv_{\mathsf{c}}$ that is direct in the sense of not involving trace inclusion (but it appears not to adapt to the setting of countable nondeterminism).

We consider the following conditions on strategies.

Definition 3.7 Let σ be a strategy.

(i) For an enabled play s, a *response* to s is an operation $k \in K$ such that $sk \in \sigma$.

(ii) σ is a *tree* when every enabled play has a unique response.

(iii) σ is *total* when every enabled play has at least one response.

(iv) σ is *deterministic*, or a *partial tree*, when every enabled play has at most one response.

(v) σ is *finitely nondeterministic* when every enabled play has only finitely many responses.

(vi) σ is *finitely founded* when it is finitely nondeterministic and no infinite play has all passive-ending prefixes in σ. A tree or partial tree with the latter property is also called *well-founded*.

Let us illustrate these conditions with examples.

- The following strategy is finitely founded:

$$\{\ \mathtt{Happy?},\ \mathtt{Happy?Yes.Bye}_3,\ \mathtt{Happy?Yes.Bye}_5\ \}$$

It is not total, since the enabled play `Happy?No.` has no response.

- The following strategy is total:

$$\{\ \texttt{Happy?},\ \texttt{Happy?Yes.Bye}_3,\ \texttt{Happy?Yes.Bye}_5\ \}\ \cup\ \{\texttt{Happy?No.Bye}_n \mid n \in \mathbb{N}\}$$

  It is not finitely nondeterministic, since the enabled play `Happy?No.` has infinitely many responses.
- The following strategy is total and finitely nondeterministic:

$$\{\ \texttt{Happy?},\ \texttt{Happy?Yes.Bye}_3,\ \texttt{Happy?Yes.Bye}_5\ \}$$
$$\cup\ \{\texttt{Happy?No.}(\texttt{Continue?Yes.})^n\,\texttt{Continue?} \mid n \in \mathbb{N}\}$$

  It is not finitely founded, since the infinite play $\texttt{Happy?No.}(\texttt{Continue?Yes.})^\omega$ has all passive-ending prefixes in it.

Proposition 3.8 *Let (X, ζ) be a transition system, and let $x \in X$.*

(i) *If ζ is total, then so is $\mathsf{Traces}\, x$.*

(ii) *If ζ is deterministic, then so is $\mathsf{Traces}\, x$.*

(iii) *If ζ is finitely nondeterministic, then so is $\mathsf{Traces}\, x$.*

(iv) *If ζ is finitely nondeterministic and well-founded, then $\mathsf{Traces}\, x$ is finitely founded.*

Proof. The plays enabled by $\mathsf{Traces}\, x$ are the active-ending traces of x. For (i) we prove that each such trace s has a response, by induction on s. The case $s = \varepsilon$ is easy, and if $s = k.i.s'$ then there is z such that $s \stackrel{k}{\Longrightarrow} (y_i)_{i \in \mathsf{Ar}(k)}$ and $z = y_i$ and s' has a response in $\mathsf{Traces}\, z$, so $k.i.s'$ has a response in $\mathsf{Traces}\, x$. The proof of (ii)–(iii) is similar. Part (iv) follows from Proposition 2.10. □

Let us mention the deterministic fragment.

Proposition 3.9 *Deterministic commands, which are given inductively by the grammar $M ::= \ \mathsf{Req}\, k?(M_i)_{i \in \mathsf{Ar}(k)}$, correspond via $M \mapsto \mathsf{Traces}\, M$ to well-founded trees.*

Proposition 3.10 *For every finitely founded, total strategy σ we have $\sigma = \mathsf{Traces}\, M$, for some command M, unique up to $\equiv_{\mathsf{c}}$.*

Proof. For a finitely founded, total strategy σ, let $P(\sigma)$ assert that $\sigma = \mathsf{Traces}\, M$, for some command M, unique up to $\equiv_{\mathsf{c}}$. We shall show that $\forall k \in \mathsf{Init}\, \sigma.\, \forall i \in \mathsf{Ar}(k).\, P(\sigma/ki)$ implies $P(\sigma)$. This implies our result because otherwise Dependent Choice gives an infinite trace whose passive-ending prefixes are all in σ.

For each $k \in \mathsf{Init}\, \sigma$ and $i \in \mathsf{Ar}(k)$, suppose $P(\sigma/ki)$, so we choose a command $N_{k,i}$ such that $\mathsf{Traces}\, N_{k,i} = \sigma/ki$. Then we have

$$\mathsf{Traces} \bigvee_{k \in \mathsf{Init}\, \sigma} \mathsf{Req}\, k?(N_{k,i})_{i \in \mathsf{Ar}(k)} \ = \ \bigcup_{k \in \mathsf{Init}\, \sigma} \mathsf{Req}\, k?\sigma/ki \ = \ \sigma$$

For uniqueness, suppose $\sigma = \mathsf{Traces}\, M$. Lemma 3.3 gives

$$\begin{aligned} M &\equiv_{\mathsf{c}} \bigvee_{k \in L} \mathsf{Req}\, k?(N'_{k,i})_{i \in \mathsf{Ar}(k)} \\ \text{Hence } \sigma &= \mathsf{Traces}\, M \\ &= \mathsf{Traces} \bigvee_{k \in L} \mathsf{Req}\, k?(N'_{k,i})_{i \in \mathsf{Ar}(k)} \\ &= \bigcup_{k \in L} \mathsf{Req}\, k?(\mathsf{Traces}\, N'_{k,i})_{i \in \mathsf{Ar}(k)} \end{aligned}$$

Hence $L = \mathsf{Init}\, \sigma$ and for each $k \in \mathsf{Init}\, \sigma$ and $i \in \mathsf{Ar}(k)$ we have $\mathsf{Traces}\, N' = \sigma/ki$, giving $N'_{k,i} \equiv_{\mathsf{c}} N_{k,i}$ by hypothesis. So

$$\begin{aligned} M &\equiv_{\mathsf{c}} \bigvee_{k \in L} \mathsf{Req}\, k?(N'_{k,i})_{i \in \mathsf{Ar}(k)} \\ &\equiv_{\mathsf{c}} \bigvee_{k \in L} \mathsf{Req}\, k?(N_{k,i})_{i \in \mathsf{Ar}(k)} \end{aligned}$$

□

Corollary 3.11 *A strategy is of the form* $\mathsf{Traces}\, M$*, for a finitely nondeterministic command* M*, iff it is total and finitely founded.*

Finally we obtain our second proof of completeness: if $\mathsf{Traces}\, M = \mathsf{Traces}\, N = \sigma$, then $M \equiv_{\mathsf{c}} N$.

3.3 Definability for Countable Nondeterminism

For the countably nondeterministic language, we again want to characterize those strategies σ that are of the form $\mathsf{Traces}\, M$. As has often been observed, the situation differs from Definition 3.7(vi): we cannot rule out an infinite play having all passive-ending prefixes in σ. For example, the command $\bigvee_{n \in \mathbb{N}} \mathsf{Continue}^n(\mathsf{Bye}_3)$ has trace set

$$\{(\mathsf{Continue?}\ \mathsf{Yes}.)^n\ \mathsf{Continue?} \mid n \in \mathbb{N}\} \cup \{(\mathsf{Continue?}\ \mathsf{Yes}.)^n\ \mathsf{Bye}_3 \mid n \in \mathbb{N}\} \qquad (9)$$

which contains every passive-ending prefix of the infinite play $(\mathsf{Continue?}\ \mathsf{Yes})^\omega$.

The appropriate conditions are the following, as we shall see.

Definition 3.12 Let σ be a strategy.

(i) σ is *countably nondeterministic* when every enabled play has countably many responses.

(ii) For an enabled play s, a *response tree* to s is a tree τ such that for all $t \in \tau$, the concatenation of s and t is in σ.

(iii) σ is *well-foundedly total* when every enabled play has a well-founded response tree. (Cf. [18, Definition 19] and [22, Appendix].)

We illustrate these conditions with examples.

- In our example signature, K is countable, so every strategy is countably nondeterministic.
- The strategy (9) is well-foundedly total.
- The following strategy is total:

$$\{\mathtt{Bye}_3\} \cup \{(\mathtt{Continue?Yes.})^n \mathtt{Continue?} \mid n \in \mathbb{N}\}$$

It is not well-foundedly total, since the enabled play `Continue?Yes.` has no well-founded response tree.

The counterpart of Proposition 3.8 is as follows.

Proposition 3.13 *Let (X, ζ) be a transition system, and let $x \in X$.*

(i) *If ζ is countably nondeterministic, then so is $\mathsf{Traces}\, x$.*

(ii) *If ζ is countably nondeterministic, total and well-founded, then $\mathsf{Traces}\, x$ is well-foundedly total.*

Proof. Part (i) is analogous to Proposition 3.8(iii). To prove part (ii), we first choose, for each $z \in X$, some $R(z) \in \zeta z$. Any play $k_0, i_0, \ldots, k_{n-1}, i_{n-1}$ enabled by $\mathsf{Traces}\, x$ is a trace of x, i.e. there is a sequence

$$x = x_0 \overset{k_0}{\Rightarrow} (y_i^0)_{i \in \mathsf{Ar}(k_0)} \quad y_{i_0}^0 = x_1 \overset{k_1}{\Rightarrow} (y_i^1)_{i \in \mathsf{Ar}(k_1)} \quad \cdots \quad y_{i_{n-1}}^{n-1} = x_n$$

The trace set of x_n in the well-founded, total deterministic system $(X, z \mapsto \{R(z)\})$ is a well-founded response tree for $k_0, i_0, \ldots, k_{n-1}, i_{n-1}$. □

Theorem 3.14 *A strategy σ is of the form $\mathsf{Traces}\, M$, for some command M, iff it is countably nondeterministic and well-foundedly total.*

Proof. ($\Rightarrow$) follows from Proposition 3.13. For ($\Leftarrow$), we proceed as follows. For $n \in \mathbb{N}$, we write Play_n for the set of plays $k_0, i_0, \ldots, k_m$ with $m < n$. An *n-approximant* of σ is a command M such that

$$\sigma \cap \mathsf{Play}_n \;\subseteq\; \mathsf{Traces}\, M \;\subseteq\; \sigma$$

We show that, for all $n \in \mathbb{N}$, every countably nondeterministic, totally well-founded strategy σ has an n-approximant, by induction on n.

- To show it is true for 0, let τ be a tree response to ε, then the corresponding deterministic command (Proposition 3.9) is a 0-approximant of σ.
- Suppose it is true for n. For each $k \in \mathsf{Init}\, \sigma$ and $i \in \mathsf{Ar}(k)$, let $M_{k,i}$ be an n-approximant of σ/ki. The set $\mathsf{Init}\, \sigma$ is countable, being the set of responses to ε. So $\bigvee_{k \in \mathsf{Init}\, \sigma} \mathsf{Req}\, k?(M_{k,i})_{i \in \mathsf{Ar}(k)}$ is an $(n+1)$-approximant to σ.

For each $n \in \mathbb{N}$, let M_n be an n-approximant to σ. Then $\mathsf{Traces} \bigvee_{n \in \mathbb{N}} M_n = \sigma$. □

4 Initial Algebras

4.1 Initial algebra for a signature

The purpose of this section is to recast our results in a way that does not mention the languages. Recall that S is our signature $(\mathsf{Ar}(k))_{k\in K}$. The following is well-known.

Definition 4.1

(i) An *S-algebra* consists of a set X and, for each $k \in K$, a function $\theta_k \colon \prod_{i\in\mathsf{Ar}(k)} X \longrightarrow X$.

(ii) An S-algebra *homomorphism* $(X, (\theta_k)_{k\in K}) \longrightarrow (Y, (\phi_k)_{k\in K})$ is a function $f \colon X \longrightarrow Y$ satisfying $f(\theta_k(x_i)_{i\in\mathsf{Ar}(k)}) = \phi_k(fx_i)_{i\in\mathsf{Ar}(k)}$ for all $k \in K$.

This leads to a standard result:

Proposition 4.2 *The set of well-founded trees with $(\mathsf{Req}\, k?)_{k\in K}$ is an initial S-algebra.*

Our aim is to combine nondeterminism and I/O in a similar way. We generalize Definition 4.1 as follows.

Definition 4.3 Let $\mathcal{C}$ be a category with products.

(i) An *S-algebra* in $\mathcal{C}$ consists of $X \in \mathcal{C}$ and, for each $k \in K$, a morphism $\theta_k \colon \prod_{i\in\mathsf{Ar}(k)} X \longrightarrow X$.

(ii) An S-algebra *homomorphism* $(X, (\theta_k)_{k\in K}) \longrightarrow (Y, (\phi_k)_{k\in K})$ is a morphism $f \colon X \longrightarrow Y$ such that $\prod_{i\in\mathsf{Ar}(k)} X \xrightarrow{\prod_{i\in\mathsf{Ar}(k)} f} \prod_{i\in\mathsf{Ar}(k)} Y$ commutes for all $\theta_k \downarrow$, $\phi_k \downarrow$, $X \xrightarrow[f]{} Y$ $k \in K$.

We now formulate our results for finite nondeterminism, without mentioning the language.

Theorem 4.4

(i) *For a strategy σ, the following are equivalent:*
- *$\sigma = \mathsf{Traces}\, x$, for some element x of a well-founded, finitely nondeterministic, total system.*
- *σ is finitely founded and total.*

(ii) *An initial S-algebra on* **SL** *(the category of semilattices) is given by the set of finitely founded total strategies, ordered by inclusion, with $(\mathsf{Req}\, k?)_{k\in K}$.*

Proof.

(i) ($\Rightarrow$) is Proposition 3.8. ($\Leftarrow$) follows from Proposition 3.10.

(ii) An S-algebra in semilattices consists of a semilattice $(X, \vee)$ and a family $(\theta_k)_{k\in K}$ of functions $\theta_k \colon \prod_{i\in \mathsf{Ar}(k)} X \longrightarrow X$ that are homomorphisms of (equational) semilattices:

$$\theta_k(x_i \vee y_i)_{i\in\mathsf{Ar}(k)} = \theta_k(x_i)_{i\in\mathsf{Ar}(k)} \vee \theta_k(y_i)_{i\in\mathsf{Ar}(k)}$$

A homomorphism is a function preserving $\vee$ and θ_k for all $k \in K$. Thus an initial S-algebra in semilattices is given by the $\equiv_{\mathsf{c}}$-classes of finitely nondeterministic commands. In view of Proposition 3.10, these correspond to finitely founded, total strategies.

□

Likewise we have the following.

Theorem 4.5

(i) *For a strategy σ, the following are equivalent:*
- *$\sigma = \mathsf{Traces}\, x$, for some element x of a well-founded, countably nondeterministic, total system.*
- *σ is countably nondeterministic and well-foundedly total.*

(ii) *An initial S-algebra on ω**SL** (the category of ω-semilattices) is given by the set of countably nondeterministic, well-foundedly total strategies, ordered by inclusion, with $(\mathsf{Req}\, k?)_{k\in K}$.*

The notion of almost complete semilattices (Section 2.1) gives another variation:

Theorem 4.6

(i) *For a strategy σ, the following are equivalent:*
- *$\sigma = \mathsf{Traces}\, x$, for some element x of a well-founded, total system.*
- *σ is well-foundedly total.*

(ii) *An initial S-algebra on category of* **ACSL** *(almost complete semilattices) is given by the set of well-foundedly total strategies, ordered by inclusion, with $(\mathsf{Req}\, k?)_{k\in K}$.*

Proof. Let C be the set of well-founded total strategies. Let λ be the maximum of $\aleph_0$ and the cardinalities of C and K. Thus for any strategy σ, every enabled play has $\leqslant \lambda$ responses. By extending the countably nondeterministic language with λ-ary nondeterministic choice $\bigvee_{i<\lambda} M_i$, we obtain analogous results to Theorem 4.5. Thus every strategy is $\mathsf{Traces}\, M$ for some command M, giving part (i).

Say that a λ-*semilattice* is a semilattice where every inhabited subset of size $\leqslant \lambda$ has a supremum, and a *homomorphism* is a function that preserves these suprema. Then C forms an initial S-algebra in λ-semilattices. Let A be an S-algebra in almost complete semilattices, and $f \colon C \longrightarrow A$ the unique homomorphism of S-algebras in λ-semilattices. Any inhabited $R \subseteq C$ has cardinality $\leqslant \lambda$, so its supremum is preserved by f. Hence f is a homomorphism of almost complete semilattices. □

4.2 Initial algebra for an endofunctor

Recall that Definition 4.3 applies to any category $\mathcal{C}$ with products. If $\mathcal{C}$ also has coproducts, then S-algebras in $\mathcal{C}$ are algebras for the endofunctor $\sum_{k\in K}\prod_{i\in\mathsf{Ar}(k)}$. Each of our categories—semilattices, ω-semilattices and almost complete semilattices—has coproducts that admit a simple explicit description.

Proposition 4.7 *Let $(A_j)_{j\in J}$ be a family of semilattices.*

(i) *The coproduct $\bigoplus^{\mathsf{f}}_{j\in J} A_j$ in* **SL** *is the set of pairs $(U,(a_j)_{j\in U})$, with $U\in\mathcal{P}^+_{\mathsf{f}}J$ and each $a_j\in A_j$. The order gives $(U,(a_j)_{j\in U})\leqslant(V,(b_j)_{j\in V})$ when $U\subseteq V$ and $a_j\leqslant b_j$ for all $j\in U$. For $j\in J$, the jth embedding $e_j\colon A_j\longrightarrow\bigoplus_{j\in J}A_j$ sends is $a\mapsto(\{j\},(a)_j)$.*

(ii) *For a finite inhabited set L, the L-indexed suprema in $\bigoplus^{\mathsf{f}}_{j\in J}A_j$ are given by*

$$\bigvee_{l\in L}(U_l,(a_{l,j})_{j\in U_l})=(V,(b_j)_{j\in V})$$

where $V=\bigcup_{l\in L}U_l$ and $b_j=\bigvee_{l\in L\,:\,j\in U_l}a_{l,j}$.

(iii) *For a family of semilattice homomorphisms $(f_j\colon A_j\longrightarrow B)_{j\in J}$, the cotuple sends $(U,(a_j)_{j\in U})$ to $\bigvee_{j\in U}f_j(a_j)$.*

*We likewise describe a coproduct $\bigoplus^{\mathsf{c}}_{j\in J}A_j$ in ω***SL***, and a coproduct $\bigoplus_{j\in J}$ in* **ACSL***.*

Let us reformulate Theorems 4.4–4.6 in these terms. We shall make use of the following constructions.

Definition 4.8 Let Strat be the set of all strategies.

(i) The function $\Phi\colon\sum_{L\in\mathcal{P}K}\prod_{k\in L}\prod_{i\in\mathsf{Ar}(k)}\mathsf{Strat}\longrightarrow\mathsf{Strat}$ sends $(L,((\sigma_{k,i})_{i\in\mathsf{Ar}(k)})_{k\in L})$ to $\bigcup_{k\in L}\mathsf{Req}\,k?(\sigma_{k,i})_{i\in\mathsf{Ar}(k)}$.

(ii) The function $\Psi\colon\mathsf{Strat}\longrightarrow\sum_{L\in\mathcal{P}K}\prod_{k\in L}\prod_{i\in\mathsf{Ar}(k)}\mathsf{Strat}$ sends σ to $(\mathsf{Init}\,\sigma,((\sigma/ki)_{i\in\mathsf{Ar}(k)})_{k\in\mathsf{Init}\,\sigma})$.

Note that Φ and Ψ are inverse. Recall also Lambek's Lemma: the structure of an initial algebra is an isomorphism.

Theorem 4.9

(i) *An initial $\bigoplus^{\mathsf{f}}_{k\in K}\prod_{i\in\mathsf{Ar}(k)}$-algebra on* **SL** *is given by the set of finitely founded, total strategies, ordered by inclusion, with structure $u\mapsto\Phi u$, whose inverse is $\sigma\mapsto\Psi\sigma$.*

(ii) *Likewise for ω-semilattices, using countably nondeterministic, well-foundedly total strategies.*

(iii) *for almost complete semilattices, using well-foundedly total strategies.*

Proof. Part (i) is a restatement of Theorem 4.4(ii). By Proposition 4.7(iii), the cotuple of $(\mathsf{Req}\,k?)_{k\in K}$ is $u\mapsto\Phi u$. Likewise for parts ii–iii. □

5 Neutral Element

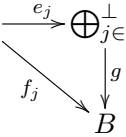
Suppose we add to our language a command `die` that has no transitions.

- To characterize definability, we drop the conditions of totality and well-founded totality. Thus a strategy is definable by a finitely nondeterministic command iff it is finitely founded. And it is definable by a countably nondeterministic command iff it is countably nondeterministic.
- To characterize trace equivalence, we add to the definition of $\equiv_{\mathsf{c}}$ the equation

  $$M \texttt{ or die} \equiv_{\mathsf{c}} \texttt{die}$$

  That is the only change required. The commutativity law between $\texttt{Req}\,k?$ and `die`, viz. $\texttt{Req}\,k?(\texttt{die})_{i\in\mathsf{Ar}(k)} = \texttt{die}$, must be omitted, as it is unsound.

We want to view the set of finitely bounded strategies as an initial algebra on **BSL** (the category of bounded semilattices). We do so using the following construction. For a family of semilattices $(A_j)_{j\in J}$, the bounded semilattice $\bigoplus^{\mathsf{f}\perp}_{j\in J} A_j$ is as in Proposition 4.7 but with the empty set included; thus it consists of pairs $(U, (a_j)_{j\in L})$ with $U \in \mathcal{P}_{\mathsf{f}} J$. It satisfies the following universal property: for any bounded semilattice B and family of semilattice maps $(f_j \colon A_j \longrightarrow B)_{j\in J}$, there is a unique bounded semilattice map $g \colon \bigoplus^{\perp}_{j\in J} A_j \longrightarrow B$ such that $A_j \xrightarrow{e_j} \bigoplus^{\perp}_{j\in J}$ commutes for all $j \in J$. Explicitly, g sends $(U, (a_j)_{j\in U})$ to $\bigvee_{j\in U} f_j(a_j)$. We thus have functors $\prod_{j\in J} \colon \mathbf{BSL}^J \longrightarrow \mathbf{SL}$ and $\bigoplus^{\mathsf{f}\perp}_{j\in J} \colon \mathbf{SL}^J \longrightarrow \mathbf{BSL}$



Theorem 5.1 *An initial $\bigoplus^{\mathsf{f}\perp}_{k\in K} \prod_{i\in\mathsf{Ar}(k)}$-algebra on* **BSL** *is given by the set of finitely founded strategies, ordered by inclusion, with structure $u \mapsto \Phi u$, whose inverse is $\sigma \mapsto \Psi\sigma$.*

Proof. A $\bigoplus^{\mathsf{f}\perp}_{k\in K} \prod_{i\in\mathsf{Ar}(k)}$-algebra may be described as a bounded semilattice A together with a family of (mere) semilattice maps $(f_k \colon \prod_{i\in\mathsf{Ar}(k)} A \longrightarrow A)_{k\in K}$. The category of such algebras is the category of models for our theory. □

Likewise, writing **B**ω**SL** for the category of bounded ω-semilattices, we define functors $\prod_{j\in J} \colon \mathbf{B}\omega\mathbf{SL}^J \longrightarrow \omega\mathbf{SL}$ and $\bigoplus^{\mathsf{c}\perp}_{j\in J} \colon \omega\mathbf{SL}^J \longrightarrow \mathbf{B}\omega\mathbf{SL}$. Then the set of countably nondeterministic strategies forms an initial $\bigoplus^{\mathsf{c}\perp}_{k\in K} \prod_{i\in\mathsf{Ar}(k)}$-algebra on **B**$\omega$**SL**.

Likewise, writing **CSL** for the category of complete semilattices, we define functors $\prod_{j\in J} \colon \mathbf{CSL}^J \longrightarrow \mathbf{ACSL}$ and $\bigoplus^{\perp}_{j\in J} \colon \mathbf{ACSL}^J \longrightarrow \mathbf{CSL}$. Then the set of all strategies forms an initial $\bigoplus^{\perp}_{k\in K} \prod_{i\in\mathsf{Ar}(k)}$-algebra on **CSL**.

6 Non-well-founded behaviour

6.1 Final coalgebras

So far we have characterized those strategies that are of the form $\mathsf{Traces}\, x$ for some element x of a well-founded system. We now turn to non-well-founded systems. Recall that, just as the set of well-founded trees forms an initial $\sum_{k\in K}\prod_{i\in\mathsf{Ar}(k)}$-algebra, so the set of all trees forms a final $\sum_{k\in K}\prod_{i\in\mathsf{Ar}(k)}$-coalgebra. We shall see a similar phenomenon arising for nondeterministic systems.

We treat the unrestricted case only. It is straightforward to enforce finite or countable nondeterminism and/or to enforce totality, if desired.

Proposition 6.1 *Every strategy σ is of the form $\mathsf{Traces}\, x$ for some element x of a transition system (X, ζ).*

Proof. Consider the system $\zeta\colon \mathsf{Strat} \longrightarrow \mathcal{P}\sum_{k\in K}\prod_{i\in\mathsf{Ar}(k)}\mathsf{Strat}$ where

$$\zeta : \sigma \mapsto \{(k, (\sigma/ki)_{i\in\mathsf{Ar}(k)}) \mid k \in \mathsf{Init}\,\sigma\}$$

Then $\mathsf{Traces}\,\sigma = \sigma$ by induction over plays, separating the cases (k) and $k.i.s'$. □

Our first goal is to show that Strat forms a final $\bigoplus^{\perp}_{k\in K}\prod_{i\in\mathsf{Ar}(k)}$-coalgebra on the category **CSL** of complete semilattices.

Definition 6.2 Let (A, ζ) be a $\bigoplus^{\perp}_{k\in K}\prod_{i\in\mathsf{Ar}(k)}$-coalgebra on **CSL**. For $a \in A$, a *trace* of r is a play $k_0, i_0, \ldots$ such that

$$a = a_0 \quad \zeta(a_0) = (L_0, ((b^0_{k,i})_{i\in\mathsf{Ar}(k)})_{k\in L_0}) \quad k_0 \in L_0$$

$$b^0_{k_0,i_0} = a_1 \quad \zeta(a_1) = (L_1, ((b^1_{k,i})_{i\in\mathsf{Ar}(k)})_{k\in L_1}) \quad k_1 \in L_1$$

$$\ldots$$

The strategy consisting of the passive-ending traces of a is written $\mathsf{Traces}\, a$.

Theorem 6.3 *A final $\bigoplus^{\perp}_{k\in K}\prod_{i\in\mathsf{Ar}(k)}$-coalgebra on* **CSL** *is given by* Strat*, ordered by inclusion, with structure $\sigma \mapsto \Psi\sigma$, whose inverse is $u \mapsto \Phi u$. The unique coalgebra morphism from (A, ζ) to the final coalgebra is $a \mapsto \mathsf{Traces}\, a$.*

Proof.

Induction over plays, separating the cases (k) and $k.i.s'$. □

What is missing from Theorem 6.3 is a characterization of the map $x \mapsto \mathsf{Traces}\, x$ on a transition system. We give this next, using the following notions.

Definition 6.4 Given a family of functions $(X_j \longrightarrow A_j)_{j\in J}$ where X_j is a set and A_j an almost complete semilattice, we write $\sum^{\sharp}_{j\in J} f_j\colon \mathcal{P}\sum_{j\in J} X_j \longrightarrow \bigoplus^{\perp}_{j\in J} A_j$ for the

unique complete semilattice homomorphism h such that

$$\begin{array}{ccc} X_j & \xrightarrow{f_j} & A_j \\ {\scriptstyle \{\mathsf{in}_j\ -\}}\downarrow & & \downarrow{\scriptstyle e_j} \\ \mathcal{P}\sum_{j\in J} X_j & \xrightarrow[h]{} & \bigoplus_{j\in J} A_j \end{array}$$

commutes for all $j \in J$. Explicitly it sends R to $(L, (y_j)_{j\in L})$ where

$$L = \{j \in J \mid \exists x \in X_j.\, \mathsf{in}_j\, x \in R\}$$
$$y_j = \bigvee_{x\in X_j\,:\,\mathsf{in}_j\, x\in R} f_j(x) \quad \text{for } j \in L.$$

Definition 6.5 Let $(X, \zeta)X$ be a transition system, and (A, ξ) a $\bigoplus^{\perp}_{k\in K}\prod_{i\in \mathsf{Ar}(k)}$-coalgebra on **CSL**. A *map* $h\colon (X,\zeta) \longrightarrow (A,\xi)$ is a function $X \longrightarrow A$ such that

$$\begin{array}{ccc} X & \xrightarrow{\quad h \quad} & A \\ {\scriptstyle \zeta}\downarrow & & \downarrow{\scriptstyle \xi} \\ \mathcal{P}\sum_{k\in K}\prod_{i\in \mathsf{Ar}(k)} X & \xrightarrow[\sum^{\sharp}_{k\in K}\prod_{i\in \mathsf{Ar}(k)} h]{} & \bigoplus^{\perp}_{k\in K}\prod_{i\in \mathsf{Ar}(k)} A \end{array}$$

commutes.

Note that such a map can be precomposed with a coalgebra morphism $(X', \zeta') \longrightarrow (X, \zeta)$, or postcomposed with a coalgebra morphism $(A, \xi) \longrightarrow (A', \xi')$, by function composition.

Theorem 6.6 *Let (X, ζ) be a transition system. Then $x \mapsto \mathsf{Traces}\, x$ is the unique map from (X, ζ) to the final coalgebra.*

Proof. Induction over plays, separating the case (k) and $k.i.s'$. □

6.2 Determinization and Bisimulation

Because the functor $\bigoplus^{\perp}_{k\in K}\prod_{i\in \mathsf{Ar}(k)}$ is a lift of a polynomial functor on **Set**, we may describe its coalgebras as "deterministic". As we shall see, they enjoy an important property of deterministic systems: trace equivalence coincides with bisimilarity.

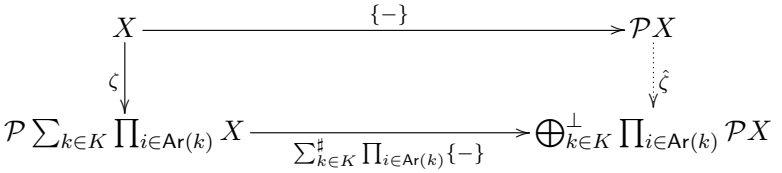
Definition 6.7 Let (A, ζ) be a $\bigoplus^{\perp}_{k\in K}\prod_{i\in \mathsf{Ar}(k)}$-coalgebra on **CSL**. A *bisimulation* on (A, ζ) is a relation $\mathcal{R}$ on A (not necessarily closed under suprema) such that, for all $a\,\mathcal{R}\,a'$, if $\zeta a = (L, ((b_{k,i})_{i\in \mathsf{Ar}(k)})_{k\in L}$ and $\zeta a' = (L', ((b'_{k,i})_{i\in \mathsf{Ar}(k)})_{k\in L'}$ we have $L = L'$ and $\forall k \in L.\, \forall i \in \mathsf{Ar}(k).\, b_{k,i}\,\mathcal{R}\, b'_{k,i}$.

Proposition 6.8 *On a $\bigoplus^{\perp}_{k\in K}\prod_{i\in \mathsf{Ar}(k)}$-coalgebra, trace equivalence is the largest bisimulation, and closed under suprema.*

We can exploit Proposition 6.8 to reason about our transition systems.

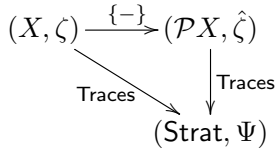
Definition 6.9 Let (X, ζ) be a transition system. The *determinization* of (X, ζ) is the $\bigoplus^{\perp}_{k\in K}\prod_{i\in \mathsf{Ar}(k)}$-coalgebra $(\mathcal{P}X, \hat{\zeta})$ where $\hat{\zeta}$ is defined by $\{-\}\colon (X,\zeta) \longrightarrow (\mathcal{P}X, \hat{\zeta})$

is a map. That is:



via the fact that $\mathcal{P}X$ is the free complete semilattice on X. Explicitly, $\hat{\zeta}$ sends $R \in \mathcal{P}X$ to $(L, ((S_{k,i})_{i\in \mathsf{Ar}(k)})_{k\in L})$ where

$$L = \{k \in K \mid \exists x \in R.\, x \xrightarrow{k} (y_i)_{i\in\mathsf{Ar}(k)}\}$$
$$S_{k,l} = \{z \in X \mid \exists x \in R., x \xrightarrow{k} (y_i)_{i\in\mathsf{Ar}(k)} \wedge y_i = z\}$$

It follows from Theorems 6.3–6.6 that 

commutes. So, for a transition system (X, ζ), and $x, x' \in X$, we conclude that x, x' are trace equivalent iff $\{x\}, \{x'\}$ are trace equivalent, i.e. iff there is a bisimulation relating $\{x\}$ and $\{x'\}$.

The notion of a bisimulation relating sets of states appears to be new, but numerous authors use bisimulations that relate distributions, e.g. [4,5,11,12,23].

7 Conclusion and variations

Modulo trace equivalence, we have seen that well-founded programs form an initial algebra, and non-well-founded programs a final coalgebra, for an appropriate endofunctor. We comment on the connections outlined in Section 1.

Firstly, each signature S and notion of strategy gives rise to a monad on **Set**. For example: the monad sending a set X to the set of countably nondeterministic, well-foundedly total strategies for $S + X$, i.e. the signature that extends S with X-many constants. Our axiomatization of trace equivalence shows this to be the tensor of the monad $\mathcal{P}^+_{\mathsf{c}}$ with the free monad on S, meaning that it is generated by the commutativity laws (7)–(8). (These laws are redundant in the case that k is constant, so adding constants to S does not give rise to additional laws.) By contrast, as shown in [13], the coproduct of these two monads is the monad $\mu_Y.\mathcal{P}^+_{\mathsf{c}}(- + \sum_{k\in K} Y^{\mathsf{Ar}(k)})$, which by Proposition 2.6 models bisimilarity.

Secondly, we have studied transition systems that are $\mathcal{P}\sum_{k\in K}\prod_{i\in\mathsf{Ar}(k)}$-coalgebras, consisting of active states. Alternatively we could consider $\prod_{k\in K}\mathcal{P}\sum_{i\in\mathsf{Ar}(k)}$-coalgebras, consisting of passive states. The story would be essentially the same. Yet another version would consider systems with both active and passive states, in a variety of active and passive positions [17,19].

The account of traces in [10] studies coalgebras for $F_{B,A} \stackrel{\text{def}}{=} \mathcal{P}(B + A \times -)$, where B and A are sets, especially the case $B = 1$. This is an instance of our functor

$\mathcal{P}\sum_{k\in K}\prod_{i\in \mathsf{Ar}(k)}$. But only *complete* traces, i.e. elements of $A^* \times B$, are considered, so the results are different from ours.

The account in [15] studies coalgebras for $G_{C,A} \stackrel{\text{def}}{=} C \times \prod_{a\in A} \mathcal{P}-$, where C is a complete semilattice and A a set. This resembles our functor $\prod_{k\in K} \mathcal{P}\sum_{i\in \mathsf{Ar}(k)}$ but is not an instance. In that work, the main case of interest is $C = \mathcal{P}B$ (in particular $B = 1$) giving $G_{C,A} \cong F_{B,A}$ and again it is complete traces that are considered.

Despite the focus on complete traces, these accounts share some general structure with ours, especially the analysis of determinization in [15]. See [14].

Notable areas of future work are probabilistic programs and infinite traces.

References

[1] Samson Abramsky and Steven Vickers. Quantales, observational logic and process semantics. *Mathematical Structures in Computer Science*, 3(2):161–227, 1993.

[2] Jiří Adámek, Stefan Milius, Lawrence S Moss, and Lurdes Sousa. Well-Pointed Coalgebras. *Logical Methods in Computer Science*, Volume 9, Issue 3, August 2013.

[3] Nathan Bowler, Sergey Goncharov, Paul Blain Levy, and Lutz Schröder. Exploring the boundaries of monad tensorability on set. *Logical Methods in Computer Science*, 9(3), 2013.

[4] Y. Deng, Y. Feng, and U. Dal Lago. On coinduction and quantum lambda calculi. In *26th International Conference on Concurrency Theory (CONCUR)*, volume 42 of *LIPIcs*. Schloss Dagstuhl - Leibniz-Zentrum fuer Informatik, 2015.

[5] C. Eisentraut, H. Hermanns, and L. Zhang. Concurrency and composition in a stochastic world. In Paul Gastin and François Laroussinie, editors, *21th International Conference on Concurrency (CONCUR 2010)*, volume 6269 of *LNCS*, pages 21–39. Springer, 2010.

[6] P. Freyd. Algebra valued functors in general and tensor products in particular. *Colloquium Mathematicae*, 14(1), 1966.

[7] S. Goncharov and L. Schröder. Powermonads and tensors of unranked effects. In *LICS*, 2011.

[8] R. Harmer. *Games and full abstraction for nondeterministic languages*. PhD thesis, Univ. of London, 1999.

[9] R Harmer and G McCusker. A fully abstract game semantics for finite nondeterminism. In *14th Symposium on Logic in Comp. Sci.* IEEE, 1999.

[10] I. Hasuo, B. Jacobs, and A. Sokolova. Generic trace semantics via coinduction. *Logical Methods in Computer Science*, 3(4), 2007.

[11] Matthew Hennessy. Exploring probabilistic bisimulations, part i. *Formal Asp. Comput*, 24(4-6):749–768, 2012.

[12] Holger Hermanns, Jan Krcál, and Jan Kretínský. Probabilistic bisimulation: Naturally on distributions. *CoRR*, abs/1404.5084, 2014.

[13] J. M. E. Hyland, G. D. Plotkin, and A. J. Power. Combining effects: sum and tensor. *Theoretical Computer Science*, 357, 2006.

[14] B. Jacobs, P. B. Levy, and J. Rot. Steps and traces. to appear, 14th IFIP WG 1.3 International Workshop on Coalgebraic Methods in Computer Science (CMCS), 2018.

[15] Bart Jacobs, Alexandra Silva, and Ana Sokolova. Trace semantics via determinization. *J. Comput. Syst. Sci*, 81(5):859–879, 2015.

[16] B. Klin and J. Rot. Coalgebraic trace semantics via forgetful logics. In Andrew M. Pitts, editor, *Foundations of Software Science and Computation Structures (FoSSaCS)*, volume 9034 of *LNCS*, pages 151–166. Springer, 2015.

[17] Dexter Kozen. Realization of coinductive types. In *Proceedings, 27th Conference on the Mathematical Foundations of Programming Semantics*, volume 276 of *ENTCS*, pages 237–246, 2011.

[18] P. B. Levy. Infinite trace equivalence. *Annals of Pure & Applied Logic*, 151(2–3), 2008.

[19] Paul Blain Levy and Sam Staton. Transition systems over games. In Thomas A. Henzinger and Dale Miller, editors, *Joint Meeting of the Twenty-Third EACSL Annual Conference on Computer Science Logic (CSL) and the Twenty-Ninth Annual ACM/IEEE Symposium on Logic in Computer Science (LICS), CSL-LICS '14, Vienna, Austria, July 14 - 18, 2014*, pages 64:1–64:10. ACM, 2014.

[20] E Moggi. Notions of computation and monads. *Information and Computation*, 93, 1991.

[21] G. Plotkin and J. Power. Notions of computation determine monads. In *Proceedings, Foundations of Software Science and Computation Structures, 2002*, volume 2303 of *LNCS*, pages 342–356. Springer, 2002.

[22] A. W. Roscoe. Unbounded non-determinism in CSP. *Journal of Logic and Computation*, 3(2), 1993.

[23] D. Sangiorgi and V. Vignudelli. Environmental bisimulations for probabilistic higher-order languages. In Rastislav Bodík and Rupak Majumdar, editors, *Proceedings, 43rd Symposium on Principles of Programming Languages (POPL)*, pages 595–607. ACM, 2016.

[24] P. Taylor. *Practical Foundations of Mathematics*. Cambridge University Press, 1999.

[25] R. J. van Glabbeek. The linear time — branching time spectrum I. The semantics of concrete, sequential processes. In J. A. Bergstra, A. Ponse, and S. A. Smolka, editors, *Handbook of Process Algebra*, pages 3–99. North-Holland, 2001.