Available online at [www.sciencedirect.com](http://www.sciencedirect.com/)

[Electronic Notes in Theoretical Computer Science 341 (2018) 23–44](https://doi.org/10.1016/j.entcs.2018.11.003)

[www.elsevier.com/locate/entcs](http://www.elsevier.com/locate/entcs)

Initial Algebras and Final Coalgebras Consisting of Nondeterministic Finite Trace Strategies

Nathan Bowler[1](#_bookmark0)

*Department of Mathematics, Universit¨at Hamburg, Germany*

Paul Blain Levy[2](#_bookmark0)

*School of Computer Science, University of Birmingham, UK*

Gordon Plotkin[3](#_bookmark0)

*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

# 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](mailto:Nathan.Bowler@uni-hamburg.de)

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

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

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

1571-0661/© 2018 The Author(s). Published by Elsevier B.V.

This is an open access article under the CC BY license (<http://creativecommons.org/licenses/by/4.0/>).

*M, N* ::= Age?(*Mn*)*n∈*N

*|* Happy?(*M, N* )

*|* Continue?(*M* ) *|* Bye*n*

*| M* or *N*

The meaning is as follows.

* The command Age?(*Mn*)*n∈*N prints Age? and pauses. If the user then enters *n*, it executes *Mn*.
* The command Happy?(*M, N* ) prints Happy? and pauses. If the user then enters

Yes or No, it executes *M* or *N* respectively.

* The command Continue?(*M* ) prints Continue? and pauses. If the user then enters Yes, it executes *M* .
* The command Bye*n* prints 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 = Happy?(Bye3*,* Bye5 or Continue?(Bye6))

def

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

Happy? Happy?Yes*.*Bye3 Happy?No*.*Bye5 Happy?No*.*Continue?

Happy?No*.*Continue?Yes*.*Bye6

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

*ε* (the empty play)

Happy?Yes*.* Happy?No*.*

Happy?No*.*Continue?Yes*.*

The following command

*M*1 = Happy?(Bye3*,* Bye5) or Happy?(Bye3*,* Continue?(Bye6))

def

has the same traces as *M*0, i.e. these commands are *trace equivalent* (though not bisimilar). The following questions naturally arise:

1. Given a set *σ* of plays, under what conditions is *σ* the trace set of some com- mand?
2. Can we give an axiomatic theory of trace equivalence? (Cf. the axiomatic analysis of process equivalences in [[1,](#_bookmark45)[25](#_bookmark69)].)

This paper’s first contribution is to answer these questions. The answer to ques- tion ([ii](#_bookmark2)) 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:

Age(*Mn*)*n∈*N or Age(*Mj* )*n∈*N = Age(*Mn* or *Mj* )*n∈*N

*n 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.

1. 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 N,

*{*Yes*,* No*}*, *{*Yes*}* and *∅*. Our results apply no matter what I/O signature is used to generate the language.

1. We may include commands of the form W*n∈*N *Mn*. This command nondeter-

ministically chooses *n ∈* N and then executes *Mn*.

In the second part of the paper (Section [6](#_bookmark40)) 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](#_bookmark62)], which can sometimes be presented by a simple theory [[21](#_bookmark63)]. 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,](#_bookmark47)[6,](#_bookmark50)[7,](#_bookmark51)[13](#_bookmark57)].

**Game semantics.** A program in the language above may be seen as playing a game (Figure [1](#_bookmark3)) 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](#_bookmark64)]. 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 nonde- terministic strategies [[8,](#_bookmark52)[9](#_bookmark53)] may be used, and our results characterize these strategies for these more general games.

**Coalgebraic traces.** Several coalgebraic accounts of traces have ap- peared [[10,](#_bookmark54)[15,](#_bookmark59)[16](#_bookmark60)]. Our account (though it does not subsume these) has the novelty of including both output and input actions. We briefly compare in Section [7.](#_bookmark44)

ACTIVE

Outputs (P-moves) zP2ASSIVE

¸

POSITIONS

Inputs (O-moves)

POSITIONS

Age?

Happy?

¸,

*n*

Yes

z˛

Awaiting age

z\_

Program running ¸,

¸¸

Continue? No

Awaiting happiness

Bye*n*

Yes

Awaiting decision to continue

Fi¸nished*n*

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

*Acknowledgements*

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

# Preliminaries

* 1. *Semilattices*

Given a set *X*, we write *PX* for the set of subsets, *P*f*X* for the set of finite subsets, *P*c*X* for the set of countable subsets. We write *X*, we write *P*+*X* for the set of inhabited (nonempty) subsets and likewise *P*+*X* and *P*+*X*.

f c

A *semilattice* is a poset (*A,* ≤) 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*  *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 *∨* that is commutative, associative and idempotent. A posetal semilattice (*A,* ≤) gives an equational one by setting *x ∨ y* to be the join of *x* and *y*. Conversely, an equational semilattice (*A, ∨*) givesa posetal one by setting *x* ≤ *y* when *x ∨ y* = *y*. These constructions are inverse. A function between semilattices is a map of equational semilattices (i.e. preserves *∨*) 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, ∨*) with a neutral element *⊥*
* an *ω*-semilattice is a semilattice (*A, ∨*) with an operation W: *Aω*  *A*¸satisfying

*x ∨* *yn* = (*x ∨ yn*)

*n∈*N *n∈*N

*x* = *x*

*n∈*N

*xn* = *xm ∨* *xn* (*m ∈* N)

*n∈*N *n∈*N

A more abstract view: the category of semilattices is the Eilenberg-Moore category

for the monad *P*+ on **Set**. Likewise the category of *ω*-semilattices for *P*+, the

f

c

category of almost complete semilattices for *P*+, etc.

* 1. *Language*

## Definition 2.1

* + 1. A *signature* consists of a set *K* of *operations*, and for each operation *k ∈ K*, a set Ar(*k*) of *argument indices*.
    2. A *transition system* over a signature (Ar(*k*))*k∈K* consists of a set *X* and function *ζ* : *X*  *P*¸Σ*k∈K* *i∈*Ar(*k*) *X*. We write *x ⇒k* (*yi*)*i∈*Ar(*k*) when (*k,* (*yi*)*i∈*Ar(*k*)) *∈ ζx*.

Informally *x ⇒k* (*yi*)*i∈*Ar(*k*) means that *x* outputs *k* and pauses, and if the user then inputs *i*, it executes *yi*.

For the sequel, we fix a signature *S* = (Ar(*k*))*k∈K*. The set of commands is defined inductively by the grammar

*M, N* ::= Req *k*?(*Mi*)*i∈*Ar(*k*) *| M* or *N*

The set of commands forms a transition system. The transition relation *M ⇒k*

(*Ni*)*i∈*Ar(*k*) is inductively defined as follows.

*M k i i∈*Ar(*k*)

*⇒* (*N* )

*j k*

*i i∈*Ar(*k*)

*M*

*⇒* (*N* )

Req *k*?(*Mi*)

*k*

*i∈*Ar(*k*) *⇒* (*Mi*)

*i∈*Ar(*k*)

*M* or *Mj*

*k*

*⇒* (*Ni*)*i∈*Ar(*k*)

*M* or *Mj*

*k*

*⇒* (*Ni*)*i∈*Ar(*k*)

For countable nondeterminism, we extend the syntax as follows:

*M* ::= *··· |* *Mn*

*n∈*N

and include the operational rule

*M k*

*n ⇒* (*Ni*)*i∈*Ar(*k*)

*n ∈* N

*Mn*

*n∈*N

*k*

*⇒* (*Ni*)*i∈*Ar(*k*)

**Definition 2.2** A transition system (*X, ζ*) is

* *total* when, for all *x ∈ X*, the set *ζx* is inhabited
* *deterministic* when, for all *x ∈ X*, the set *ζx* has at most one element
* *ﬁnitely nondeterministic* when, for all *x ∈ X*, the set *ζx* is finite
* *countably nondeterministic* when, for all *x ∈ X*, the set *ζx* is countable
* *well-founded* [4](#_bookmark6) when there is no infinite sequence of transitions

*x*0 *k*0

0 0

*i∈*Ar(*k* )

*k*1 1

1 *i∈*Ar(*k* )

## Proposition 2.3

*⇒* (*yi* )

0 *yi*0 = *x*

*⇒* (*yi* )

1 *···*

1. *The set of ﬁnitely nondeterministic commands, as a transition system, is total,* *ﬁnitely nondeterministic and well-founded.*
2. *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*

* 1. *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 *R* on

*i*

*X* such that *x Rxj* implies that, if *x* =*k⇒*

(*yi*)

*i∈*Ar(*k*)

then there exists (*yj*)*i∈*Ar(*k*)

such that *xj* =*k⇒* (*yj*) and *∀i ∈* Ar(*k*)*. yi Ryj*, and vice versa. The greatest

*i i∈*Ar(*k*)

*i*

bisimulation is called *bisimilarity*.

For example, the commands *M*0 and *M*1 in Section [1](#_bookmark1) are not bisimilar. We axiomatize bisimilarity as follows.

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

*M* or *N ≡ N* or *M* (1)

(*M* or *N* ) or *P ≡ M* or (*N* or *P* ) (2)

*M* or *M ≡ M* (3)

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

*M* or *Mn ≡* (*M* or *Mn*) (4)

*n∈*N *n∈*N

*M ≡ M* (5)

*n∈*N

*Mn ≡ Mm* or *Mn* (*m ∈* N) (6)

*n∈*N *n∈*N

4 Cf. the notion of well-founded coalgebra [[2,](#_bookmark46)[24](#_bookmark68)].

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

**Proof.** Induction on *≡* shows that *≡* implies bisimilarity and that it is a bisimula- tion. *2*

* 1. *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*,...* with *kr ∈ K* and *ir ∈* Ar(*kr*). 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](#_bookmark1), 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

* + 1. A *nondeterministic ﬁnite trace strategy* is a set *σ* of passive-ending plays such that *sik ∈ σ* implies *s ∈ σ*.
    2. An active-ending play is *enabled* by *σ* when it is either *ε* (the empty play), or

*si* for *s ∈ σ*.

Henceforth “strategy” means “nondeterministic finite trace strategy”.

**Definition 2.9** Let (*X, ζ*) be a transition system and *x ∈ X*. A play *k*0*, i*0*,.* is a

*trace* of *x* when there is a sequence

*x* = *x k*0 0

0 *k*1 1

0 *⇒* (*yi* )*i∈*Ar(*k*0) *yi*0 = *x*1 *⇒* (*yi* )*i∈*Ar(*k*1) *···*

The strategy consisting of the passive-ending traces of *x* is written Traces *x*.

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

def

=

*N*

0

def

=

*N*

1

Continue?*n* (Bye3) or Continue*ω*

*n∈*N

Continue?*n* (Bye3)

*n∈*N

are trace equivalent, but the infinite play (Continue?Yes*.*)*ω* 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 K¨onig’s lemma.

**Proposition 2.10** *For a ﬁnitely nondeterministic system ζ* : *X*  *P*¸Σ*k∈K* *i∈*Ar(*k*) *X and x ∈ X, an inﬁnite play k*0*, i*0*,... is a trace of x iff all its passive-ending preﬁxes are in* Traces *x.*

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

**Definition 2.11** For *k ∈ K* and a family of strategies (*σi*)*i∈*Ar(*k*), we define the strategy

Req *k*?(*σi*)*i∈I* = *{*(*k*)*}∪ {k.i.s | i ∈* Ar(*k*)*,s ∈ σi}*

def

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

Traces Req *k*?(*Mi*)*i∈*Ar(*k*) = Req *k*?(Traces *Mi*)*i∈*Ar(*k*)

Traces (*M* or *N* ) = Traces *M ∪* Traces *N*

Traces *Mn* = Traces *Mn*

*n∈*N *n∈*N

Secondly, we decompose strategies as follows.

**Definition 2.13** Let *σ* be a strategy. We write Init *σ* for the set of *k ∈ K* such that (*k*) *∈ σ*. For each such *k* and each *i ∈* Ar(*k*), we define the strategy *σ/ki* =

def

*{s | k.i.s ∈ σ}*.

# Well-founded behaviour

* 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 *≡*c, is the least congruence on commands that satisfies the semilattice laws ([1](#_bookmark5))–([3](#_bookmark5)) and commutativity between Req *k*? and or:

Req *k*?(*Mia* or *Ni*)*i∈*Ar(*k*) = Req *k*?(*Mi*)*i∈*Ar(*k*) or Req *k*?(*Ni*)*i∈*Ar(*k*) (*k ∈ K*) (7)

Also, for the countably nondeterministic language, the *ω*-semilattice laws ([4](#_bookmark5))–([6](#_bookmark5)) and commutativity between Req *k*? and W:

Req *k*?( *Mi,n*)*i∈*Ar(*k*) = Req *k*?(*Mi,n*)*i∈*Ar(*k*) (*k ∈ K*) (8)

*n∈*N *n∈*N

For example, the commands *M*0 and *M*1 in Section [1](#_bookmark1) are commuting equivalent.

**Proposition 3.2** *(Soundness) If M ≡*c *N then* Traces *M* = Traces *N.*

**Proof.** This is proved by induction; the soundness of the laws follows from Propo- sition [2.12](#_bookmark9). For example:

Req *k*?( *σi,n*)*i∈*Ar(*k*) = *{k}∪ {k.i.s | i ∈* Ar(*k*)*,s ∈* *σi,n}*

*n∈*N *n∈*N

Req *k*?(*σi,n*)*i∈*Ar(*k*) = (*{k}∪ {k.i.s | i ∈* Ar(*k*)*,s ∈ σi,n}*)

*n∈*N *n∈*N

and these are equal since N is inhabited. Hence ([8](#_bookmark10)) is sound. *2*

We shall now prove completeness, i.e. the converse of Proposition [3.2](#_bookmark11). We use the following.

**Lemma 3.3** *For every command M we have*

*M ≡*c Req *k*?(*Nk,i*)*i∈*Ar(*k*)

*k∈L*

*for some L ∈ P*+*K and family of commands* (*Nk,i*)*k∈L,i∈*Ar(*k*)*. For the countably nondeterministic language, L ∈ P*+*K.*

f

c

**Proof.** Induction on *M* . The case that *M* = Req *k*?(*Nk,i*)*i∈*Ar(*k*) is obvious. In the case *M* = *n∈*N *Mn* we have *Mn ≡*c *k∈Ln* Req *k*?(*Nn,k,i*)*i∈*Ar(*k*). Then

W W

*Mn ≡*c

Req *k*?(*Nn,k,i*)*i∈*Ar(*k*)

*n∈*N

*n∈*N*,k∈Ln*

*≡*c

Req *k*?(*Nn,k,i*)*i∈*Ar(*k*)

*k∈*S

*n∈*N

*Ln n∈*N : *k∈Ln*

*≡*c

*k∈*S

*n∈*N *Ln*

Req *k*?(

*n∈*N : *k∈Ln*

*Nn,k,i*)*i∈*Ar(*k*)

Likewise for the case *M* = *M*0 or *M*1. *2*

Our completeness proof proceeds by characterizing *trace inclusion*, the preorder that relates *M* to *N* when Traces *M ⊆* Traces *N* .

**Proposition 3.4** *Write M* ≤c *N when M* or *N ≡*c *N.*

* + 1. ≤c *is a preorder.*
    2. *M* or *N is a least upper bound of M and N*
    3. W*n∈*N *is a least upper bound of* (*Mn*)*n∈*N*.*
    4. or *and* W*n∈*N *and* Req *k*? *for k ∈ K are all monotone, i.e.* ≤c *is a precongru-*

*ence.*

**Proof.** Parts ([i](#_bookmark13))–([iii](#_bookmark14)) follow the construction of a posetal semilattice from an equa- tional semilattice in Section [2.1](#_bookmark4). Monotonicity of or and W*n∈*N follow from the least upper bound property. Monotonicity of Req *k*? follows from ([7](#_bookmark10)), because the latter

may be viewed as saying that Req *k*? is a map of equational semilattices and hence

a map of posetal semilattices. *2*

**Proposition 3.5** *M* ≤c *N iff* Traces *M ⊆* Traces *N.*

**Proof.** (*⇒*) follows from Proposition [3.2](#_bookmark11). We prove (*⇐*) by induction on *M* . The cases *M* = *M*0 or *M*1 and *M* = W*n∈*N *Mn* follow from the least upper bound property. Suppose *M* = Req *k*?(*Mi*)*i∈*Ar(*k*). Lemma [3.3](#_bookmark12) gives *N ≡*c

W

*k∈L* Req *k*?(*Nk,i*)*i∈*Ar(*k*) and so

Traces Req *k*?(*Mi*)*i∈*Ar(*k*) *⊆* Traces Req *k*?(*Nk,i*)*i∈*Ar(*k*)

*k∈L*

i.e. Req *k*?(Traces *Mi*)*i∈*Ar(*k*) *⊆* Req *k*?(Traces *Nk,i*)*i∈*Ar(*k*)

*k∈L*

Thus *k ∈ L*, and for each *i ∈* Ar(*k*) we have Traces *Mi ⊆* Traces *Nk,i* implying

*Mi* ≤c *Nk,i* by the inductive hypothesis. Hence

*M* = Req *k*?(*Mi*)*i∈*Ar(*k*)

≤c Req *k*?(*Nk,i*)*i∈*Ar(*k*)

≤c W*k∈L* Req *k*?(*Nk,i*)*i∈*Ar(*k*) *≡*c *N*

*2*

**Corollary 3.6** *M ≡*c *N iff* Traces *M* = Traces *N.*

* 1. *Deﬁnability for ﬁnite nondeterminism*

We shall now characterize which strategies are of the form Traces *M* for a finitely nondeterministic command *M* . We also give a second proof of completeness of *≡*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.

* + 1. For an enabled play *s*,a *response* to *s* is an operation *k ∈ K* such that *sk ∈ σ*.
    2. *σ* is a *tree* when every enabled play has a unique response.
    3. *σ* is *total* when every enabled play has at least one response.
    4. *σ* is *deterministic*, or a *partial tree*, when every enabled play has at most one response.
    5. *σ* is *ﬁnitely nondeterministic* when every enabled play has only finitely many responses.
    6. *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:

*{* Happy?*,* Happy?Yes*.*Bye3*,* Happy?Yes*.*Bye5 *}*

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

* + The following strategy is total:

*{* Happy?*,* Happy?Yes*.*Bye3*,* Happy?Yes*.*Bye5 *} ∪ {*Happy?No*.*Bye*n | n ∈* N*}*

It is not finitely nondeterministic, since the enabled play Happy?No*.* has infinitely many responses.

* + The following strategy is total and finitely nondeterministic:

*{* Happy?*,* Happy?Yes*.*Bye3*,* Happy?Yes*.*Bye5 *}*

*∪ {*Happy?No*.*(Continue?Yes*.*)*n* Continue? *| n ∈* N*}*

It is not finitely founded, since the infinite play Happy?No*.*(Continue?Yes*.*)*ω* has all passive-ending prefixes in it.

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

1. *If ζ is total, then so is* Traces *x.*
2. *If ζ is deterministic, then so is* Traces *x.*
3. *If ζ is ﬁnitely nondeterministic, then so is* Traces *x.*
4. *If ζ is ﬁnitely nondeterministic and well-founded, then* Traces *x is ﬁnitely founded.*

**Proof.** The plays enabled by Traces *x* are the active-ending traces of *x*. For ([i](#_bookmark17)) we prove that each such trace *s* has a response, by induction on *s*. The case *s* = *ε* is

easy, and if *s* = *k.i.sj* then there is *z* such that *s* =*k⇒* (*yi*)

*i∈*Ar(*k*)

and *z* = *yi* and *sj*

has a response in Traces *z*, so *k.i.sj* has a response in Traces *x*. The proof of ([ii](#_bookmark19))–([iii](#_bookmark20))

is similar. Part ([iv](#_bookmark21)) follows from Proposition [2.10](#_bookmark8). *2*

Let us mention the deterministic fragment.

**Proposition 3.9** *Deterministic commands, which are given inductively by the grammar M* ::= Req *k*?(*Mi*)*i∈*Ar(*k*)*, correspond via M '→* Traces *M to well-founded trees.*

**Proposition 3.10** *For every ﬁnitely founded, total strategy σ we have σ* =

Traces *M, for some command M, unique up to ≡*c*.*

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

For each *k ∈* Init *σ* and *i ∈* Ar(*k*), suppose *P* (*σ/ki*), so we choose a command

*Nk,i* such that Traces *Nk,i* = *σ/ki*. Then we have

Traces

*k∈*Init *σ*

Req *k*?(*Nk,i*)*i∈*Ar(*k*) =

*k∈*Init *σ*

Req *k*?*σ/ki* = *σ*

For uniqueness, suppose *σ* = Traces *M* . Lemma [3.3](#_bookmark12) gives

*M ≡*c Req *k*?(*Nj*

*k,i*

)*i∈*Ar(*k*)

*k∈L*

Hence *σ* = Traces *M*

= Traces Req *k*?(*Nj*

*k,i*

)*i∈*Ar(*k*)

*k∈L*

= Req *k*?(Traces *Nj*

*k,i*

)*i∈*Ar(*k*)

*k∈L*

Hence *L* = Init *σ* and for each *k ∈* Init *σ* and *i ∈* Ar(*k*) we have Traces *Nj* = *σ/ki*,

giving *Nj ≡*c *Nk,i* by hypothesis. So

*k,i*

*M ≡*c Req *k*?(*Nj*

*k,i*

)*i∈*Ar(*k*)

*k∈L*

*≡*c Req *k*?(*Nk,i*)*i∈*Ar(*k*)

*k∈L*

*2*

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

Finally we obtain our second proof of completeness: if Traces *M* = Traces *N* = *σ*, then *M ≡*c *N* .

* 1. *Deﬁnability for Countable Nondeterminism*

For the countably nondeterministic language, we again want to characterize those strategies *σ* that are of the form Traces *M* . As has often been observed, the situation differs from Definition [3.7](#_bookmark15)([vi](#_bookmark16)): we cannot rule out an infinite play having all passive-

ending prefixes in *σ*. For example, the command W*n∈*N Continue*n*(Bye3) has trace

set

*{*(Continue? Yes*.*)*n* Continue? *| n ∈* N*}∪ {*(Continue? Yes*.*)*n* Bye3 *| n ∈* N*}* (9)

which contains every passive-ending prefix of the infinite play (Continue? Yes)*ω*.

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

**Definition 3.12** Let *σ* be a strategy.

* + 1. *σ* is *countably nondeterministic* when every enabled play has countably many responses.
    2. For an enabled play *s*, a *response tree* to *s* is a tree *τ* such that for all *t ∈ τ* , the concatenation of *s* and *t* is in *σ*.
    3. *σ* is *well-foundedly total* when every enabled play has a well-founded response tree. (Cf. [[18](#_bookmark65), Definition 19] and [[22](#_bookmark66), Appendix].)

We illustrate these conditions with examples.

* + - * In our example signature, *K* is countable, so every strategy is countably nonde- terministic.
      * The strategy ([9](#_bookmark24)) is well-foundedly total.
      * The following strategy is total:

*{*Bye3*}∪ {*(Continue?Yes*.*)*n*Continue? *| n ∈* 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](#_bookmark18) is as follows.

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

1. *If ζ is countably nondeterministic, then so is* Traces *x.*
2. *If ζ is countably nondeterministic, total and well-founded, then* Traces *x is well- foundedly total.*

**Proof.** Part ([i](#_bookmark26)) is analogous to Proposition [3.8](#_bookmark18)([iii](#_bookmark20)). To prove part ([ii](#_bookmark27)), we first choose, for each *z ∈ X*, some *R*(*z*) *∈ ζz*. Any play *k*0*, i*0*,..., kn−*1*, in−*1 enabled by Traces *x* is a trace of *x*, i.e. there is a sequence

*x* = *x k*0 0

0 *k*1 1

*n−*1

0 *⇒* (*yi* )*i∈*Ar(*k*0) *yi*0 = *x*1 *⇒*

(*yi* )*i∈*Ar(*k*1) *··· yin−*1 = *xn*

The trace set of *xn* in the well-founded, total deterministic system (*X, z '→ {R*(*z*)*}*) is a well-founded response tree for *k*0*, i*0*,..., kn−*1*, in−*1. *2*

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

**Proof.** (*⇒*) follows from Proposition [3.13](#_bookmark25). For (*⇐*), we proceed as follows. For *n ∈* N, we write Play*n* for the set of plays *k*0*, i*0*,..., km* with *m < n*. An *n- approximant* of *σ* is a command *M* such that

*σ ∩* Play*n ⊆* Traces *M ⊆ σ*

We show that, for all *n ∈* 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](#_bookmark22)) is a 0-approximant of *σ*.
      * Suppose it is true for *n*. For each *k ∈* Init *σ* and *i ∈* Ar(*k*), let *Mk,i* be an *n*- approximant of *σ/ki*. The set Init *σ* is countable, being the set of responses to *ε*.

So W*k∈*Init *σ* Req *k*?(*Mk,i*)*i∈*Ar(*k*) is an (*n* + 1)-approximant to *σ*.

For each *n ∈* N, let *Mn* be an *n*-approximant to *σ*. Then Traces W*n∈*N *Mn* = *σ*. *2*

# Initial Algebras

* 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 (Ar(*k*))*k∈K*. The following is well-known.

## Definition 4.1

* + 1. An *S-algebra* consists of a set *X* and, for each *k ∈ K*, a function

*θk* : *i∈*Ar(*k*) *X*  *X*¸.

* + 1. An *S*-algebra *homomorphism* (*X,* (*θk*)*k∈K*) (*Y*¸*,* (*φk*)*k∈K*) is a function

*f* : *X*  *Y* ¸satisfying *f* (*θk*(*xi*)*i∈*Ar(*k*))= *φk*(*f xi*)*i∈*Ar(*k*) for all *k ∈ K*.

This leads to a standard result:

**Proposition 4.2** *The set of well-founded trees with* (Req *k*?)*k∈K is an initial S- algebra.*

Our aim is to combine nondeterminism and I/O in a similar way. We generalize Definition [4.1](#_bookmark28) as follows.

**Definition 4.3** Let *C* be a category with products.

1. An *S-algebra* in *C* consists of *X ∈ C* and, for each *k ∈ K*, a morphism

*θk* : *i∈*Ar(*k*) *X*  *X*¸.

1. An *S*-algebra *homomorphism* (*X,* (*θk*)*k∈K*) (*Y*¸*,* (*φk*)*k∈K*) is a morphism

*f* : *X*  *Y*¸such that *i∈*Ar(*k*)

*θk*

*i∈*Ar(*k*) ¸

*f*

*X*

*Y*

*i∈*Ar(*k*)

*φk*

commutes for all

*X*  *Y* ¸

*f*

*k ∈ K*.

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

## Theorem 4.4

1. *For a strategy σ, the following are equivalent:*
   * *σ* = Traces *x, for some element x of a well-founded, ﬁnitely nondeterministic, total system.*
   * *σ is ﬁnitely founded and total.*
2. *An initial S-algebra on* **SL** *(the category of semilattices) is given by the set of ﬁnitely founded total strategies, ordered by inclusion, with* (Req *k*?)*k∈K.*

## Proof.

1. (*⇒*) is Proposition [3.8](#_bookmark18). (*⇐*) follows from Proposition [3.10](#_bookmark23).
2. An *S*-algebra in semilattices consists of a semilattice (*X, ∨*) and a family (*θk*)*k∈K* of functions *θk* : *i∈*Ar(*k*) *X*  *X*¸that are homomorphisms of (equa- tional) semilattices:

*θk*(*xi ∨ yi*)*i∈*Ar(*k*) = *θk*(*xi*)*i∈*Ar(*k*) *∨ θk*(*yi*)*i∈*Ar(*k*)

A homomorphism is a function preserving *∨* and *θk* for all *k ∈ K*. Thus an initial *S*-algebra in semilattices is given by the *≡*c-classes of finitely nondeter- ministic commands. In view of Proposition [3.10](#_bookmark23), these correspond to finitely founded, total strategies.

*2*

Likewise we have the following.

## Theorem 4.5

1. *For a strategy σ, the following are equivalent:*
   * *σ* = Traces *x, for some element x of a well-founded, countably nondetermin- istic, total system.*
   * *σ is countably nondeterministic and well-foundedly total.*
2. *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* (Req *k*?)*k∈K.*

The notion of almost complete semilattices (Section [2.1](#_bookmark4)) gives another variation:

## Theorem 4.6

1. *For a strategy σ, the following are equivalent:*
   * *σ* = Traces *x, for some element x of a well-founded, total system.*
   * *σ is well-foundedly total.*
2. *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* (Req *k*?)*k∈K.*

**Proof.** Let *C* be the set of well-founded total strategies. Let *λ* be the maximum of *ℵ*0 and the cardinalities of *C* and *K*. Thus for any strategy *σ*, every enabled play has ≤ *λ* responses. By extending the countably nondeterministic language with *λ*-

ary nondeterministic choice W*i<λ Mi*, we obtain analogous results to Theorem [4.5](#_bookmark32).

Thus every strategy is Traces *M* for some command *M* , giving part ([i](#_bookmark33)).

Say that a *λ-semilattice* is a semilattice where every inhabited subset of size ≤ *λ* 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* : *C*  *A*¸the unique homomorphism of *S*-algebras in *λ*-semilattices. Any inhabited *R ⊆ C* has cardinality ≤ *λ*, so its supremum is preserved by *f* . Hence *f* is a homomorphism of almost complete semilattices. *2*

* 1. *Initial algebra for an endofunctor*

Recall that Definition [4.3](#_bookmark29) applies to any category *C* with products. If *C*

also has coproducts, then *S*-algebras in *C* are algebras for the endofunctor Σ*k∈K* *i∈*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* (*Aj*)*j∈J be a family of semilattices.*

* + 1. *The coproduct* f *Aj in* **SL** *is the set of pairs* (*U,* (*aj*)*j∈U* )*, with U ∈ P*+*J*

*j∈J*

f

*and each aj ∈ Aj. The order gives* (*U,* (*aj*)*j∈U* ) ≤ (*V,* (*bj*)*j∈V* ) *when U ⊆ V*

*and aj* ≤ *bj for all j ∈ U. For j ∈ J, the jth embedding ej* : *Aj*  ¸ *Aj*

*j∈J*

*sends is a '→* (*{j},* (*a*)*j*)*.*

* + 1. *For a ﬁnite inhabited set L, the L-indexed suprema in* f

*j∈J*

*Aj are given by*

(*Ul,* (*al,j*)*j∈Ul* )= (*V,* (*bj*)*j∈V* )

*l∈L*

*where V* = *l∈L Ul and bj* = W*l∈L* : *j∈Ul al,j.*

* + 1. *For a family of semilattice homomorphisms* (*fj* : *Aj*  *B*)¸*j∈J , the cotuple sends*

(*U,* (*aj*)*j∈U* ) *to* W*j∈U fj*(*aj*)*.*

*We likewise describe a coproduct* c

*j∈J*

*Aj in ω***SL***, and a coproduct*

*j∈J*

*in* **ACSL***.*

Let us reformulate Theorems [4.4](#_bookmark30)–[4.6](#_bookmark34) in these terms. We shall make use of the following constructions.

**Definition 4.8** Let Strat be the set of all strategies.

1. The function Φ: Σ*L∈PK* *k∈L* *i∈*Ar(*k*) Strat ¸Strat sends (*L,* ((*σk,i*)*i∈*Ar(*k*))*k∈L*) to *k∈L* Req *k*?(*σk,i*)*i∈*Ar(*k*).
2. The function Ψ: Strat ¸Σ*L∈PK* *k∈L* *i∈*Ar(*k*) Strat sends *σ* to

(Init *σ,* ((*σ/ki*)*i∈*Ar(*k*))*k∈*Init *σ*).

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

## Theorem 4.9

1. *An initial* f

*k∈K*

*i∈*Ar(*k*)

*-algebra on* **SL** *is given by the set of ﬁnitely founded,*

*total strategies, ordered by inclusion, with structure u '→* Φ*u, whose inverse is*

*σ '→* Ψ*σ.*

1. *Likewise for ω-semilattices, using countably nondeterministic, well-foundedly total strategies.*
2. *for almost complete semilattices, using well-foundedly total strategies.*

**Proof.** Part ([i](#_bookmark37)) is a restatement of Theorem [4.4](#_bookmark30)([ii](#_bookmark31)). By Proposition [4.7](#_bookmark35)([iii](#_bookmark36)), the cotuple of (Req *k*?)*k∈K* is *u '→* Φ*u*. Likewise for parts [ii](#_bookmark38)–[iii](#_bookmark39). *2*

# 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 *≡*c the equation

*M* or die *≡*c die

That is the only change required. The commutativity law between Req *k*? and

die, viz. Req *k*?(die)*i∈*Ar(*k*) = 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 con-

struction. For a family of semilattices (*Aj*)*j∈J* , the bounded semilattice f*⊥ Aj*

*j∈J*

is as in Proposition [4.7](#_bookmark35) but with the empty set included; thus it consists of pairs (*U,* (*aj*)*j∈L*) with *U ∈ P*f*J* . It satisfies the following universal property: for any bounded semilattice *B* and family of semilattice maps (*fj* : *Aj*  *B*¸)*j∈J* , there is a

unique bounded semilattice map *g* : *⊥*

*j∈J*

*Aj*  *B*¸such that *Aj*

*ej*  ¸*⊥*

*j∈J*

*fj*

*g*

com-

mutes for all *j ∈ J* . Explicitly, *g* sends (*U,* (*aj*)*j∈U* ) to

*j∈J*

z

W*j∈U fj*(*aj*). We thus have

*B*

functors

*j∈J*

: **BSL***J*  **SL**¸and f*⊥*

: **SL***J*  **B**¸**SL**

**Theorem 5.1** *An initial* f*⊥*

*k∈K*

*i∈*Ar(*k*)

*-algebra on* **BSL** *is given by the set of*

*ﬁnitely founded strategies, ordered by inclusion, with structure u '→* Φ*u, whose in-*

*verse is σ '→* Ψ*σ.*

**Proof.** A f*⊥*

-algebra may be described as a bounded semilattice *A*

*k∈K*

together with a family of (mere) semilattice maps (*fk* :

*i∈*Ar(*k*) *A*

*i∈*Ar(*k*)

¸

category of such algebras is the category of models for our theory. *2*

*A*)*k∈K*. The

Likewise, writing **B***ω***SL** for the category of bounded *ω*-semilattices, we define

functors

*j∈J*

: **B***ω***SL***J*  *ω*¸**SL** and c*⊥*

: *ω***SL***J*  **B**¸*ω***SL**. Then the set of

countably nondeterministic strategies forms an initial c*⊥*

*j∈J*

*k∈K*

*i∈*Ar(*k*)

-algebra on

**B***ω***SL**.

Likewise, writing **CSL** for the category of complete semilattices, we define func-

tors *j∈J* : **CSL***J*  **A**¸**CSL** and : **ACSL CSL**. Then the set of all

*⊥ J*  ¸

*j∈J*

strategies forms an initial *⊥*

*k∈K*

*i∈*Ar(*k*)

-algebra on **CSL**.

# Non-well-founded behaviour

* 1. *Final coalgebras*

So far we have characterized those strategies that are of the form 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 Σ*k∈K* *i∈*Ar(*k*)-

algebra, so the set of all trees forms a final Σ*k∈K* *i∈*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* Traces *x for some element x of a transition system* (*X, ζ*)*.*

**Proof.** Consider the system *ζ* : Strat *P*¸Σ*k∈K* *i∈*Ar(*k*) Strat where

*ζ* : *σ '→ {*(*k,* (*σ/ki*)*i∈*Ar(*k*)) *| k ∈* Init *σ}*

Then Traces *σ* = *σ* by induction over plays, separating the cases (*k*) and *k.i.sj*. *2*

Our first goal is to show that Strat forms a final *⊥*

*k∈K*

*i∈*Ar(*k*)

-coalgebra on

the category **CSL** of complete semilattices.

**Definition 6.2** Let (*A, ζ*) be a *⊥*

*k∈K*

*trace* of *r* is a play *k*0*, i*0*,...* such that

*i∈*Ar(*k*)

-coalgebra on **CSL**. For *a ∈ A*, a

*a* = *a*0 *ζ*(*a*0) = (*L*0*,* ((*b*0

*k,i*

*b*0 = *a*1 *ζ*(*a*1) = (*L*1*,* ((*b*1

)*i∈*Ar(*k*))*k∈L*0 ) *k*0 *∈ L*0

)*i∈*Ar(*k*))*k∈L* ) *k*1 *∈ L*1

*k*0*,i*0

*···*

*k,i* 1

The strategy consisting of the passive-ending traces of *a* is written Traces *a*.

**Theorem 6.3** *A ﬁnal* *⊥*

*k∈K*

*i∈*Ar(*k*)

*-coalgebra on* **CSL** *is given by* Strat*, ordered*

*by inclusion, with structure σ '→* Ψ*σ, whose inverse is u '→* Φ*u. The unique coalge-*

*bra morphism from* (*A, ζ*) *to the ﬁnal coalgebra is a '→* Traces *a.*

## Proof.

Induction over plays, separating the cases (*k*) and *k.i.sj*. *2*

What is missing from Theorem [6.3](#_bookmark41) is a characterization of the map *x '→* Traces *x*

on a transition system. We give this next, using the following notions.

**Definition 6.4** Given a family of functions (*Xj*  *Aj*¸)*j∈J* where *Xj* is a set and *Aj*

*Xj*

an almost complete semilattice, we write Σ

*j∈J*

*fj* : *P* Σ

*j∈J*

¸*⊥*

*j∈J*

*Aj* for the

unique complete semilattice homomorphism *h* such that *Xj*

*{*in*j —}*

*fj*  *A*¸

*ej*

*j*

*P* Σ

¸

*j∈J Xj h*

commutes for all *j ∈ J* . Explicitly it sends *R* to (*L,* (*yj*)*j∈L*) where

*L* = *{j ∈ J | ∃x ∈ Xj.* in*j x ∈ R}*

*j∈J Aj*

*yj* = *fj*(*x*) for *j ∈ L*.

*x∈Xj* : in*j x∈R*

**Definition 6.5** Let (*X, ζ*)*X* be a transition system, and (*A, ξ*) a *⊥*

*k∈K*

*i∈*Ar(*k*)-

coalgebra on **CSL**. A *map h* : (*X, ζ*) (*A*¸*, ξ*) is a function *X*  *A*¸such that

*X*  *h*  *A*¸

commutes.

*ζ*

Σ

*ξ*

¸*⊥*

Σ

*P k∈K*

*i∈*Ar(*k*) *X k∈K*

*i∈*Ar(*k*) *h*

*k∈K*

*i∈*Ar(*k*) *A*

Note that such a map can be precomposed with a coalgebra morphism (*Xj,ζj*) (*X*¸*, ζ*), or postcomposed with a coalgebra morphism (*A, ξ*) (*A*¸*j, ξj*), by function composition.

**Theorem 6.6** *Let* (*X, ζ*) *be a transition system. Then x '→* Traces *x is the unique map from* (*X, ζ*) *to the ﬁnal coalgebra.*

**Proof.** Induction over plays, separating the case (*k*) and *k.i.sj*. *2*

* 1. *Determinization and Bisimulation*

Because the functor *⊥*

*k∈K*

*i∈*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 *⊥*

*k∈K*

*i∈*Ar(*k*)

-coalgebra on **CSL**. A *bisimulation*

on (*A, ζ*) is a relation *R* on *A* (not necessarily closed under suprema) such that,

for all *a Raj*, if *ζa* = (*L,* ((*bk,i*)*i∈*Ar(*k*))*k∈L* and *ζaj* = (*Lj,* ((*bj*

*k,i*

)*i∈*Ar(*k*))*k∈L′* we have

*L* = *Lj* and *∀k ∈ L. ∀i ∈* Ar(*k*)*. bk,i Rbj* .

*k,i*

**Proposition 6.8** *On a* *⊥*

*k∈K*

*i∈*Ar(*k*)

*-coalgebra, trace equivalence is the largest*

*bisimulation, and closed under suprema.*

We can exploit Proposition [6.8](#_bookmark43) to reason about our transition systems.

**Definition 6.9** Let (*X, ζ*) be a transition system. The *determinization* of (*X, ζ*) is

the *⊥*

*k∈K*

*i∈*Ar(*k*)

-coalgebra (*PX, ζ*ˆ) where *ζ*ˆ is defined by *{−}* : (*X, ζ*) (*P*¸*X, ζ*ˆ)

is a map. That is:

*X*  *{—}*  *P*¸*X*

*ζ*

Σ

*ζ*ˆ

¸*⊥*

Σ

*P k∈K*

*i∈*Ar(*k*) *X*

*k∈K*

*i∈*Ar(*k*)

*{—}*

*k∈K*

*i∈*Ar(*k*) *PX*

via the fact that *PX* is the free complete semilattice on *X*. Explicitly,

*R ∈ PX* to (*L,* ((*Sk,i*)*i∈*Ar(*k*))*k∈L*) where

*L* = *{k ∈ K | ∃x ∈ R. x* =*k⇒* (*y* ) *}*

*i i∈*Ar(*k*)

*S* = *{z ∈ X | ∃x ∈ R., x* =*k⇒* (*y* ) *∧ y* = *z}*

*k,l i i∈*Ar(*k*) *i*

*ζ*ˆ sends

It follows from Theorems [6.3](#_bookmark41)–[6.6](#_bookmark42) that (*X, ζ*) *{—}*  (*P*¸*X, ζ*ˆ)

Traces

Traces

z

(Strat*,* Ψ)

commutes. So, for

a transition system (*X, ζ*), and *x, xj ∈ X*, we conclude that *x, xj* are trace equivalent iff *{x}, {xj}* are trace equivalent, i.e. iff there is a bisimulation relating *{x}* and *{xj}*.

The notion of a bisimulation relating sets of states appears to be new, but numerous authors use bisimulations that relate distributions, e.g. [[4,](#_bookmark48)[5,](#_bookmark49)[11,](#_bookmark55)[12,](#_bookmark56)[23](#_bookmark67)].

# Conclusion and variations

Modulo trace equivalence, we have seen that well-founded programs form an ini- tial algebra, and non-well-founded programs a final coalgebra, for an appropriate endofunctor. We comment on the connections outlined in Section [1.](#_bookmark1)

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 nondetermin- istic, 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 *P*+

c

with the free monad on *S*, meaning that it is gen-

erated by the commutativity laws ([7](#_bookmark10))–([8](#_bookmark10)). (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](#_bookmark57)], the coproduct of these two monads is the monad

*μY .P*+(*−* + Σ

c

*k∈K*

*Y* Ar(*k*)), which by Proposition [2.6](#_bookmark7) models bisimilarity.

Secondly, we have studied transition systems that are *P* Σ*k∈K* *i∈*Ar(*k*)- coalgebras, consisting of active states. Alternatively we could consider

*k∈K P* Σ*i∈*Ar(*k*)-coalgebras, consisting of passive states. The story would be es-

sentially the same. Yet another version would consider systems with both active

and passive states, in a variety of active and passive positions [[17,](#_bookmark61)[19](#_bookmark64)].

The account of traces in [[10](#_bookmark54)] studies coalgebras for *FB,A* = *P*(*B* + *A×−*), where

def

*B* and *A* are sets, especially the case *B* = 1. This is an instance of our functor

*P* Σ*k∈K* *i∈*Ar(*k*). But only *complete* traces, i.e. elements of *A∗ ×B*, are considered,

so the results are different from ours.

The account in [[15](#_bookmark59)] studies coalgebras for *GC,A*

complete semilattice and *A* a set. This resembles our functor

d=ef *C ×*

*a* *∈A*

*P−*, where *C* is a

Σ

*k∈K P*

*i∈*Ar(*k*) but

is not an instance. In that work, the main case of interest is *C* = *PB* (in particular

*B* = 1) giving *GC,A ∼*= *FB,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](#_bookmark59)]. See [[14](#_bookmark58)].

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.

1. Jiˇr´ı Ad´amek, Stefan Milius, Lawrence S Moss, and Lurdes Sousa. Well-Pointed Coalgebras. *Logical Methods in Computer Science*, Volume 9, Issue 3, August 2013.
2. Nathan Bowler, Sergey Goncharov, Paul Blain Levy, and Lutz Schr¨oder. Exploring the boundaries of monad tensorability on set. *Logical Methods in Computer Science*, 9(3), 2013.
3. 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.
4. C. Eisentraut, H. Hermanns, and L. Zhang. Concurrency and composition in a stochastic world. In Paul Gastin and Fran¸cois Laroussinie, editors, *21th International Conference on Concurrency (CONCUR 2010)*, volume 6269 of *LNCS*, pages 21–39. Springer, 2010.
5. P. Freyd. Algebra valued functors in general and tensor products in particular. *Colloquium Mathematicae*, 14(1), 1966.
6. S. Goncharov and L. Schr¨oder. Powermonads and tensors of unranked effects. In *LICS*, 2011.
7. R. Harmer. *Games and full abstraction for nondeterministic languages*. PhD thesis, Univ. of London, 1999.
8. R Harmer and G McCusker. A fully abstract game semantics for finite nondeterminism. In *14th* *Symposium on Logic in Comp. Sci.* IEEE, 1999.
9. I. Hasuo, B. Jacobs, and A. Sokolova. Generic trace semantics via coinduction. *Logical Methods in* *Computer Science*, 3(4), 2007.
10. Matthew Hennessy. Exploring probabilistic bisimulations, part i. *Formal Asp. Comput*, 24(4-6):749– 768, 2012.
11. Holger Hermanns, Jan Krc´al, and Jan Kret´ınsky´. Probabilistic bisimulation: Naturally on distributions.

*CoRR*, abs/1404.5084, 2014.

1. J. M. E. Hyland, G. D. Plotkin, and A. J. Power. Combining effects: sum and tensor. *Theoretical Computer Science*, 357, 2006.
2. 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.
3. Bart Jacobs, Alexandra Silva, and Ana Sokolova. Trace semantics via determinization. *J. Comput. Syst. Sci*, 81(5):859–879, 2015.
4. 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.
5. 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.
6. P. B. Levy. Infinite trace equivalence. *Annals of Pure & Applied Logic*, 151(2–3), 2008.
7. 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.
8. E Moggi. Notions of computation and monads. *Information and Computation*, 93, 1991.
9. 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.
10. A. W. Roscoe. Unbounded non-determinism in CSP. *Journal of Logic and Computation*, 3(2), 1993.
11. D. Sangiorgi and V. Vignudelli. Environmental bisimulations for probabilistic higher-order languages. In Rastislav Bod´ık and Rupak Majumdar, editors, *Proceedings, 43rd Symposium on Principles of Programming Languages (POPL)*, pages 595–607. ACM, 2016.
12. P. Taylor. *Practical Foundations of Mathematics*. Cambridge University Press, 1999.
13. 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.