An Adequacy Theorem Between Mixed Powerdomains and Probabilistic Concurrency

Renato Neves
University of Minho & INESC-TEC, Portugal
nevrenato@di.uminho.pt

We present an adequacy theorem for a concurrent extension of probabilistic GCL. The underlying denotational semantics is based on the so-called mixed powerdomains which combine non-determinism with stochasticity. The theorem itself is formulated via M. Smyth's idea of treating observable properties as open sets of a topological space. The proof hinges on a 'topological generalisation' of König's lemma in the setting of probabilistic programming (a result that is proved in the paper as well).

One application of the theorem is that it entails semi-decidability w.r.t. whether a concurrent program satisfies an observable property (written in a certain form). This is related to M. Escardó's conjecture about semi-decidability w.r.t. may and must probabilistic testing.

1 Introduction

Motivation. The combination of probabilistic operations with concurrent ones – often referred to as probabilistic concurrency – has been extensively studied in the last decades and has various applications [20, 3, 29, 42, 26, 44]. A central concept in this paradigm is that of a (probabilistic) scheduler [5, 42]: a memory-based entity that analyses and (probabilistically) dictates how programs should be interleaved.

Albeit intensively studied and with key results probabilistic concurrency still carries fundamental challenges. In this paper we tackle the following instance: take a concurrent imperative language equipped with probabilistic operations. Then ask whether a given program P in state s satisfies a property ϕ, in symbols $\langle P, s\rangle \models \phi$. For example ϕ may refer to the fact that $\langle P, s\rangle$ terminates with probability greater than $\frac{1}{2}$ under *all* possible schedulers (which is intimately connected to the topic of statistical termination [18, 23]). Or dually ϕ could refer to the existence of *at least one* scheduler under which $\langle P, s\rangle$ terminates with probability greater than $\frac{1}{2}$. Such a question can be particularly challenging to answer for it involves quantifications over the universe of scheduling systems, which as we shall see are *uncountably* many. An even more demanding type of question arises from establishing an observational preorder $\lesssim$ between programs in state s: more specifically we write,

$$\langle P, s\rangle \lesssim \langle Q, s\rangle \text{ iff } \Big(\text{for all properties } \phi.\ \langle P, s\rangle \models \phi \text{ implies } \langle Q, s\rangle \models \phi\Big)$$

and ask whether $\langle P, s\rangle \lesssim \langle Q, s\rangle$ for two programs P and Q in state s. Adding to the quantifications over scheduling systems, we now have a universal quantification over formulae.

Contributions. In this paper we connect the previous questions to domain-theoretical tools [12, 14]. Concretely we start by recalling necessary background (Section 2). Then in order to provide a formal underpinning to these questions, we introduce a concurrent extension of the probabilistic guarded command language (pGCL) [25] and equip it with an operational semantics as well as a logic for reasoning

Submitted to:
Express/SOS 2025

about program properties (Section 3). The logic itself is based on M. Smyth's idea of treating observable properties as open sets of a topological space [37, 43]. We then introduce a domain-theoretical denotational semantics $[\![-]\!]$ (Section 4) and establish its computational adequacy w.r.t. the operational counterpart (Section 5). More formally we establish an equivalence of the form,

$$\langle P, s\rangle \models \phi \text{ iff } [\![P]\!](s) \models \phi \qquad\qquad \text{(for all properties } \phi \text{ in our logic)}$$

whose full details will be given later on. From adequacy we then straightforwardly derive a *full abstraction* result w.r.t. the observational preorder $\lesssim$, *i.e.* we obtain the equivalence,

$$\langle P, s\rangle \lesssim \langle Q, s\rangle \text{ iff } [\![P]\!](s) \leq [\![Q]\!](s)$$

for all programs P and Q and state s.

An interesting and somewhat surprising consequence of our adequacy theorem is that the question of whether $\langle P, s\rangle \models \phi$ becomes semi-decidable for a main fragment of our logic. This includes for example the two questions above concerning the termination of $\langle P, s\rangle$ with probability greater than $\frac{1}{2}$. What is more semi-decidability is realised by an exhaustive search procedure – which looks surprising because as already mentioned the space of schedulers is uncountable and the question of whether $\langle P, s\rangle \models \phi$ involve quantifications over this space. In Section 6 we further detail this aspect and also the topic of statistical termination [18, 23, 24, 11]. Also in Section 6, to illustrate the broad range of applicability of our results we show how to instantiate them to the quantum setting [7, 48, 49]. Briefly the resulting concurrent language treats measurements and qubit resets as probabilistic operations which thus frames the language in the context of probabilistic concurrency. To the best of our knowledge the quantum concurrent language thus obtained is the first one equipped with an adequacy theorem. We then conclude with a discussion of future work.

Outline. We proceed by outlining our denotational semantics. First the fact that we are working with concurrency, more specifically with the interleaving paradigm, suggests the adoption of the 'resumptions model' for defining the semantics. This approach was proposed by R. Milner in the seventies [28] and subsequently adapted to different contexts (see *e.g.* [38, 32, 13]). When adapted to our specific case program denotations $(\!|P|\!)$ are intensional and defined as elements of the *greatest* solution of the equation,

$$X \cong T(S + X \times S)^S \tag{1}$$

where S is a pre-determined set of states and T a functor that encodes a combination of non-determinism (which handles interleaving in concurrency) and stochasticity. In other words such program denotations are elements of (the carrier of) a final coalgebra.

Second we will see that in order to obtain adequacy w.r.t. the operational semantics one needs to extensionally collapse $(\!|-|\!)$ – *i.e.* given a program P one needs to find a suitable way of keeping only the terminating states of $(\!|P|\!)$ whilst accounting for the branching structure encoded in T. Denoting X as the greatest solution of Equation (1), such amounts to finding a suitable morphism $\mathrm{ext} : X \to T(S)^S$. Our approach to this is inspired by [19]: such a morphism arises naturally if T is additionally a monad and X is additionally the *least* solution of Equation (1). Concretely we obtain the following commutative diagram,

$$\begin{tikzcd}
\mathrm{Pr} \arrow[r, dotted, "(\!|-|\!)"] \arrow[d, "\mathrm{op}"'] & X \arrow[r, dotted, "\mathrm{ext}"] \arrow[d, bend left, "\mathrm{unfold}"] \arrow[d, phantom, "\cong"] & T(S)^S \\
T(S + \mathrm{Pr} \times S)^S \arrow[r, "T(\mathrm{id} + (\!|-|\!) \times \mathrm{id})^S"'] & T(S + X \times S)^S \arrow[r, "T(\mathrm{id} + \mathrm{ext} \times \mathrm{id})^S"'] \arrow[u, bend left, "\mathrm{fold}"] & T(S + T(S)^S \times S)^S \arrow[u, "{[\eta^T,\ \mathrm{app}]^{\star S}}"']
\end{tikzcd} \tag{2}$$

where Pr denotes the set of programs, the operations η^T, $(-)^\star$ refer to the monadic structure postulated for T, app denotes the application map, and op denotes our operational semantics in the form of a coalgebra. The fact that the left rectangle commutes (proved in Theorem 4.2) means that the intensional semantics $(\!|-|\!)$ agrees with the 'one-step' transitions that arise from the operational semantics. The fact that the right rectangle commutes is obtained by construction (initiality) and yields a recursive definition of the extensional collapse, with elements of X seen as 'resumption trees'. Our final denotational semantics $[\![-]\!]$ is the composition $\mathrm{ext} \cdot (\!|-|\!)$.

In what regards the choice of T there are three natural candidates which are collectively known as mixed powerdomains [27, 30, 39, 21]. One is associated with angelic (or may) non-determinism, another with demonic (or must) non-determinism, and the other one with both cases. Our computational adequacy result applies to these three powerdomains, a prominent feature of our result being that the aforementioned logic (for reasoning about properties) varies according to whatever mixed powerdomain is chosen. Notably keeping in touch with its name, the proof for demonic non-determinism is much harder to achieve than the angelic case: in order to prove the former we establish what can be intuitively seen as a topological generalisation of König's lemma [10] in the setting of probabilistic programming. This is detailed further in the paper and proved in appendix (Section B) after a series of auxiliary results.

Related work. The sequential fragment of our language is the well-known pGCL [25], a programming language that harbours both probabilistic and non-deterministic choice. The idea of establishing adequacy between pGCL and the three mixed powerdomains was propounded in [22] as an open endeavour – and in this regard the adequacy of the angelic mixed powerdomain was already established in [42]. More recently J. Goubault-Larrecq [15, 16] established computational adequacy at the unit type between an extension of PCF with probabilistic and non-deterministic choice and the three mixed powerdomains. Note however that in this case the associated operational semantics is not based on probabilistic schedulers like in [42] and our case, but rather on what could be called 'angelic and demonic strategies'. Moreover to scale up the adequacy result in the *op. cit.* to a global store (as required in pGCL) demands that states can be directly compared in the language, an assumption that we do not make and that does not hold for example in the quantum setting. We therefore believe that our adequacy result restricted to the sequential fragment is interesting *per se*.

Adequacy at the level of probabilistic concurrency has been studied at various fronts, however as far as we are aware usually in the setting of process algebra [3, 30, 44], a direction orthogonal to ours.

Semi-decidability concerning may and must statistical termination, in a spirit similar to ours, was conjectured in [6]. Note however that the host programming language in the latter case was a *sequential, higher-order* language with non-deterministic and probabilistic choice. The underlying scheduling system in the *op. cit.* was also different from ours: it was encoded via pairs of elements in the Cantor space $2^{\mathbb{N}}$.

Prerequisites and notation. We assume from the reader basic knowledge of category theory, domain theory, and topology (details about these topics are available for example in [1, 12, 14]). We denote the left and right injections into a coproduct by inl and inr respectively, and the application map $X^Y \times Y \to X$ by app. Given a set X we use X^* to denote the respective set of lists, and given an element $x \in X$ we denote the 'constant on x' map $1 \to X$ by $\underline{x}$. We omit subscripts in natural transformations whenever no ambiguity arises from this. We use η and $(-)^\star$ to denote respectively the unit and lifting of a monad $(T, \eta, (-)^\star)$, and whenever relevant we attach T as a superscript to these two operations to clarify which monad we are referring to. We denote the set of positive natural numbers $\{1, 2, \dots\}$ by $\mathbb{N}_+$. Omitted proofs are found in the paper's appendix.

2 Background

Domains and topology. We start by recalling pre-requisite notions for establishing the aforementioned denotational semantics and corresponding adequacy theorem. As usual we call DCPOs those partially ordered sets that are directed-complete. We call domains those DCPOs that are continuous [12, Definition I-1.6]. We call a DCPO pointed it it has a bottom element and we call a map between pointed DCPOs strict if it preserves bottom elements. Similarly we call a functor on pointed DCPOs strict if it preserves strict maps. Given a poset X its Scott topology $\sigma(X)$ consists of all those subsets $U \subseteq X$ that are upper-closed and inaccessible by directed joins: *i.e.* for every directed join $\bigvee_{i \in I} x_i \in U$ there must already exist some element x_i in the family $(x_i)_{i \in I}$ such that $x_i \in U$. Another important topology on X is the lower topology $\omega(X)$. It is generated by the subbasis of closed sets $\{\uparrow x \mid x \in X\}$. Yet another important topology is the Lawson topology $\lambda(X)$ which is generated by the subbasis $\sigma(X) \cup \omega(X)$ (see details in [12, Section III-1]). When treating a poset X as a topological space we will be tacitly referring to its Scott topology unless stated otherwise. A domain X is called coherent if the intersection of two compact saturated subsets in X is again compact. Finally note that every set can be regarded as a coherent domain by taking the discrete order. We will often tacitly rely on this fact.

Let DCPO be the category of DCPOs and continuous maps. Let Dom and CohDom be the full subcategories of DCPO whose objects are respectively domains and coherent domains. We thus obtain the chain of inclusions $\mathsf{CohDom} \hookrightarrow \mathsf{Dom} \hookrightarrow \mathsf{DCPO}$. Next, let C be any full subcategory of DCPO. Since it is a full subcategory the inclusion $\mathsf{C} \hookrightarrow \mathsf{DCPO}$ reflects limits and colimits [1, Definition 13.22]. This property is very useful in domain theory because domain theoreticians often work in full subcategories of DCPO. It is well-known for example that DCPO-products of (coherent) domains where only finitely many are non-pointed are (coherent) domains as well [14, Proposition 5.1.54 and Exercise 8.3.33]. It follows that such limits are also limits in Dom, and if the involved domains are coherent then they are limits in CohDom as well. The same reasoning applies to coproducts: DCPO-coproducts of (coherent) domains are (coherent) domains as well [14, Proposition 5.1.59 and Proposition 5.2.34]. Finally observe that DCPO is distributive and that this applies to any full subcategory of DCPO that is closed under binary (co)products. This includes for example Dom and CohDom.

The probabilistic powerdomain. Let us now detail the probabilistic powerdomain and associated constructions. These take a key rôle not only in the definition of the three mixed powerdomains, but also in our denotational semantics and corresponding adequacy theorem. We start with the pre-requisite notion of a continuous valuation.

Definition 2.1 ([39, 17]). *Consider a topological space X and let $\mathcal{O}(X)$ be its topology. A function $\mu : \mathcal{O}(X) \to [0,\infty]$ is called a continuous valuation on X if for all opens $U,V \in \mathcal{O}(X)$ it satisfies the following conditions:*

- $\mu(\emptyset) = 0$;
- $U \subseteq V \Rightarrow \mu(U) \leq \mu(V)$;
- $\mu(U) + \mu(V) = \mu(U \cup V) + \mu(U \cap V)$;
- $\mu\left(\bigcup_{i \in I} U_i\right) = \bigvee_{i \in I} \mu(U_i)$ *for every directed family of opens* $(U_i)_{i \in I}$.

We use $\mathrm{V}(X)$ to denote the set of continuous valuations on X. We also use $\mathrm{V}_{=1}(X)$ and $\mathrm{V}_{\leq 1}(X)$ to denote respectively the subsets of continuous valuations μ such that $\mu(X) = 1$ and $\mu(X) \leq 1$. An important type of continuous valuation is the *point-mass* (or Dirac) valuation δ_x $(x \in X)$ defined by,

$$\delta_x(U) = \begin{cases} 1 & \text{if } x \in U \\ 0 & \text{otherwise} \end{cases}$$

Recall that a cone is a set C with operations for addition $+ : C \times C \to C$ and scaling $\cdot : \mathbb{R}_{\geq 0} \times C \to C$ that satisfy the laws of vector spaces except for the one that concerns additive inverses [39, Section 2.1]. It is well-known that $\mathrm{V}(X)$ forms a cone with scaling and addition defined pointwise. If X is a domain then $\mathrm{V}(X)$ is also a domain. The order on $\mathrm{V}(X)$ is defined via pointwise extension and a basis (in the domain-theoretic sense) is given by the finite linear combinations $\sum_{i \in I} r_i \cdot \delta_{x_i}$ of point-masses with $r_i \in \mathbb{R}_{\geq 0}$ and $x_i \in X$. We will often abbreviate such combinations to $\sum_i r_i \cdot x_i$. Moreover we will use $\mathrm{V}_{=1,\omega}(X)$ and $\mathrm{V}_{\leq 1,\omega}(X)$ to denote respectively the subsets of continuous valuations in $\mathrm{V}_{=1}(X)$ and $\mathrm{V}_{\leq 1}(X)$ that are finite linear combinations of point-masses. Whenever X is a domain the Scott-topology of the domain $\mathrm{V}(X)$ coincides with the so-called *weak topology* [17] which is generated by the subbasic sets,

$$\bigcirc_p U = \{\mu \in \mathrm{V}(X) \mid \mu(U) > p\} \qquad\qquad (U \text{ an open of } X \text{ and } p \in \mathbb{R}_{\geq 0})$$

The probabilistic powerdomain also gives rise to a category of cones. We briefly detail it next, and direct the reader to the more thorough account in [39, Chapter 2].

Definition 2.2. *A d-cone $(C, \leq, \cdot, +)$ is a cone $(C, \cdot, +)$ such that the pair $(C, \leq)$ is a DCPO and the operations $\cdot : \mathbb{R}_{\geq 0} \times C \to C$ and $+ : C \times C \to C$ are Scott-continuous. If the ordered set $(C, \leq)$ is additionally a domain then we speak of a continuous d-cone. The category* Cone *has as objects continuous d-cones and as morphisms continuous linear maps.*

The forgetful functor $\mathrm{U} : \mathsf{Cone} \to \mathsf{Dom}$ is right adjoint: the respective universal property is witnessed by the construct $\mathrm{V}(-)$. Specifically for every domain X, continuous d-cone C, and Dom-morphism $f : X \to \mathrm{U}(C)$ we have the diagrammatic situation,

$$\begin{tikzcd} X \arrow[r, "x \mapsto \delta_x"] \arrow[dr, "f"'] & \mathrm{U}\mathrm{V}(X) \arrow[d, "\mathrm{U}(f^\star)"] & & \mathrm{V}(X) \arrow[d, dotted, "f^\star"] \\ & \mathrm{U}(C) & & C \end{tikzcd}$$

with $f^\star$ defined by $f^\star(\sum_i r_i \cdot x_i) = \sum_i r_i \cdot f(x_i)$ on basic elements $\sum_i r_i \cdot x_i \in \mathrm{V}(X)$. We thus obtain standardly a functor $\mathrm{V} : \mathsf{Dom} \to \mathsf{Cone}$. Now, the functor $\mathrm{U} : \mathsf{Cone} \to \mathsf{Dom}$ is additionally monadic, with the respective left adjoint given by $\mathrm{V} : \mathsf{Dom} \to \mathsf{Cone}$. Among other things such implies that Cone is as complete as the category Dom. For example Cone has all products of continuous d-cones (recall our previous remarks about Dom and note that d-cones are always pointed [39, page 21]). Note as well that binary products are actually biproducts by virtue of addition being Scott-continuous. Finally let LCone be the full subcategory of Cone whose objects are Lawson-compact (*i.e.* compact in the Lawson topology). $\mathrm{V}(X)$ is Lawson-compact whenever X is a coherent domain [39, Theorem 2.10] and Lawson-compactness of a domain entails coherence [12, Theorem III-5.8]. We thus obtain a functor $\mathrm{V} : \mathsf{CohDom} \to \mathsf{LCone}$ that gives rise to the commutative diagram,

$$\begin{tikzcd} \mathsf{LCone} \arrow[r, tail] \arrow[d, bend left, "\mathrm{U}"] \arrow[d, phantom, "\dashv"] & \mathsf{Cone} \arrow[d, bend left, "\mathrm{U}"] \arrow[d, phantom, "\dashv"] \\ \mathsf{CohDom} \arrow[r, tail] \arrow[u, bend left, "\mathrm{V}"] & \mathsf{Dom} \arrow[u, bend left, "\mathrm{V}"] \end{tikzcd}$$

The following result, which we will use multiple times, is somewhat folklore. Unfortunately we could not find a proof in the literature, so we provide one here.

Theorem 2.1. *Consider a zero-dimensional topological space X (*i.e.* a topological space with a basis of clopens). For all valuations $\mu, \nu \in \mathrm{V}_{=1}(X)$ if $\mu \leq \nu$ then $\mu = \nu$. In other words the order on $\mathrm{V}_{=1}(X)$ is discrete.*

Proof. Observe that if $\mathcal{B}$ is a basis of clopens of X then the set $\mathcal{B}^\vee = \{U_1 \cup \cdots \cup U_n \mid U_1, \ldots, U_n \in \mathcal{B}\}$ is directed and it is constituted only by clopens as well. Next we reason by contradiction and by appealing to the conditions imposed on valuations (Definition 2.1). Suppose that there exists an open $U \subseteq X$ such that $\mu(U) < \nu(U)$. This open can be rewritten as a directed union $\bigcup \mathcal{D}$ where $\mathcal{D} = \{V \subseteq U \mid V \in \mathcal{B}^\vee\}$ and we obtain the strict inequation,

$$\mu(U) < \bigvee_{V \in \mathcal{D}} \nu(V)$$

The latter entails the existence of a clopen $V \in \mathcal{D}$ such that $\mu(V) < \nu(V)$. It then must be the case that $1 = \mu(X) = \mu(V) + \mu(X \backslash V) < \nu(V) + \nu(X \backslash V) = \nu(X)$. This proves that $1 < \nu(X)$, a contradiction. □

The three mixed powerdomains. We now present the so-called geometrically convex powercones $\mathrm{P}_x(-)$ ($x \in \{l, u, b\}$), *viz.* convex lower (P_l), convex upper (P_u), and biconvex (P_b) [39, Chapter 4]. As we will see later on, their composition with the probabilistic powerdomain (which was previously recalled) yields the three mixed powerdomains that were mentioned in the introduction.

We start with some preliminary notions. Given a DCPO X and a subset $A \subseteq X$ we denote the Scott-closure of A by $\overline{A}$. The subset A is called *order convex* whenever for all $x \in X$ if there exist elements $a_1, a_2 \in A$ such that $a_1 \leq x \leq a_2$ then $x \in A$. Given a cone C we call a subset $A \subseteq C$ geometrically convex (or just convex) if $a_1, a_2 \in A$ entails $p \cdot a_1 + (1-p) \cdot a_2 \in A$ for all $p \in [0,1]$. We denote the convex closure of A by $\mathrm{conv} A$. The latter is explicitly defined by,

$$\mathrm{conv} A = \{\textstyle\sum_{i \in I} p_i \cdot a_i \mid \sum_{i \in I} p_i = 1 \text{ and } \forall i \in I.\, a_i \in A\}$$

For two finite subsets $F, G \subseteq C$ the expression $F + G$ denotes Minkowski's sum and $p \cdot F$ denotes the set $\{p \cdot c \mid c \in F\}$. If a subset $A \subseteq C$ is non-empty, Lawson-compact, and order convex we call it a lens. Given a continuous d-cone C we define the following partially ordered sets,

$$\begin{aligned}
\mathrm{P}_l(C) &= \{A \subseteq C \mid A \text{ non-empty, closed, and convex}\} \\
\mathrm{P}_u(C) &= \{A \subseteq C \mid A \text{ non-empty, compact, saturated, and convex}\} \\
\mathrm{P}_b(C) &= \{A \subseteq C \mid A \text{ a convex lens}\}
\end{aligned}$$

with the respective orders defined by $A \leq_l B$ iff ${\downarrow}A \subseteq {\downarrow}B$, $A \leq_u B$ iff ${\uparrow}B \subseteq {\uparrow}A$, and $A \leq_b B$ iff ${\downarrow}A \subseteq {\downarrow}B$ and ${\uparrow}B \subseteq {\uparrow}A$. Whenever working with $\mathrm{P}_b(C)$ we will assume that C is additionally Lawson-compact. All three posets form domains and $\mathrm{P}_b(C)$ is additionally Lawson-compact. In particular the sets of the form $\overline{\mathrm{conv} F}$, ${\uparrow}\mathrm{conv} F$, and $\overline{\mathrm{conv} F} \cap {\uparrow}\mathrm{conv} F$ (for F a finite set) form respectively a basis (in the domain-theoretic sense) for the convex lower, convex upper, and biconvex powercones. For simplicity we will often denote $\overline{\mathrm{conv} F}$ by $l(F)$, ${\uparrow}\mathrm{conv} F$ by $u(F)$, and $\overline{\mathrm{conv} F} \cap {\uparrow}\mathrm{conv} F$ by $b(F)$ – and in the general case *i.e.* when we wish to speak of all three cases at the same time we write $x(F)$.

The Scott topologies of the three powercones have well-known explicit characterisations that are closely connected to the 'possibility' ($\Diamond$) and 'necessity' ($\Box$) operators of modal logic [46, 30]. Specifically the Scott topology of $\mathrm{P}_b(C)$ is generated by the subbasic sets,

$$\Diamond U = \{A \mid A \cap U \neq \emptyset\} \text{ and } \Box U = \{A \mid A \subseteq U\} \qquad (U \text{ an open of } C)$$

the Scott topology of $\mathrm{P}_l(C)$ is generated only by those subsets of the form $\Diamond U$ and the Scott topology of $\mathrm{P}_u(C)$ is generated only by those subsets of the form $\Box U$. When seeing open subsets as observable properties [37, 43], the powercone $\mathrm{P}_l(C)$ is thus associated with the *possibility* of a given property U being true (*i.e.* whether a property U holds in at least one scenario) $\Diamond U$, the powercone $\mathrm{P}_u(C)$ is associated with the *necessity* of a given property U being true (*i.e.* whether a property U holds in all scenarios) $\Box U$ and the powercone $\mathrm{P}_b(C)$ with both cases. These observations together with the previous characterisation of the Scott topology of the probabilistic powerdomain yield a natural logic for reasoning about program properties in concurrent pGCL, as detailed in the following section.

All three powercones are equipped with the structure of a cone and with a continuous semi-lattice operation $\uplus$: the whole structure is defined on basic elements by,

$$x(F) + x(G) = x(F+G) \qquad r \cdot x(F) = x(r \cdot F) \qquad x(F) \uplus x(G) = x(F \cup G)$$

and is thoroughly detailed in [39, Section 4]. Moreover the convex lower and convex upper variants form monads in Cone whereas the biconvex variant forms a monad in LCone. In all three cases the unit $C \to \mathrm{P}_x(C)$ is defined by $c \mapsto x(\{c\})$ and the Kleisli lifting of $f : C \to \mathrm{P}_x(D)$ is defined on basic elements by $f^\star(x(F)) = \uplus\{f(a) \mid a \in F\}$. Finally we often abbreviate the composition $\mathrm{P}_x\mathrm{V}$ simply to PV and by a slight abuse of notation treat $\mathrm{P}_x\mathrm{V}$ as a functor $\mathrm{P}_x\mathrm{V} : \mathsf{Dom} \to \mathsf{Dom}$ if $x \in \{l,u\}$ or as a functor $\mathrm{P}_x\mathrm{V} : \mathsf{CohDom} \to \mathsf{CohDom}$ if $x \in \{b\}$. These three functors constitute the three mixed powerdomains. It follows from composition of adjunctions that every mixed powerdomain is a monad on Dom (resp. CohDom).

Theorem 2.2. *For every $x \in \{l,u\}$ the functor is $\mathrm{P}_x\mathrm{V} : \mathsf{Dom} \to \mathsf{Dom}$ is strong. Given two domains X and Y the tensorial strength $\mathrm{str} : X \times \mathrm{P}_x\mathrm{V}(Y) \to \mathrm{P}_x\mathrm{V}(X \times Y)$ is defined on basic elements as the mapping $(a, x(F)) \mapsto x(\{\delta_a \otimes \mu \mid \mu \in F\})$ where,*

$$\delta_a \otimes \textstyle\sum_i p_i \cdot y_i = \sum_i p_i \cdot (a, y_i)$$

for every linear combination $\sum_i p_i \cdot y_i$. The same applies to the functor $\mathrm{P}_b\mathrm{V}$.

Similarly the following result is also obtained straightforwardly.

Proposition 2.1. *The monads $\mathrm{P}_x : \mathsf{Cone} \to \mathsf{Cone}$ ($x \in \{l,u\}$) are strong w.r.t. coproducts. The corresponding natural transformation $\mathrm{str} : \mathrm{Id} + \mathrm{P}_x \to \mathrm{P}_x(\mathrm{Id} + \mathrm{Id})$ is defined as $\mathrm{str} = [\eta \cdot \mathrm{inl}, \mathrm{P}_x\mathrm{inr}]$. An analogous result applies to the biconvex powercone $\mathrm{P}_b : \mathsf{LCone} \to \mathsf{LCone}$.*

By unravelling the definition of str and by taking advantage of the fact that binary coproducts of cones are also products (recall our previous observations about biproducts in Cone) we obtain the following equivalent formulation of str for all components C, D,

$$\mathrm{str}_{C,D} : C \times \mathrm{P}_x(D) \to \mathrm{P}_x(C \times D) \qquad (c, x(F)) \mapsto x(\{c\} \times F)$$

3 Concurrent pGCL, its operational semantics, and its logic

As mentioned in Section 1 our language is a concurrent extension of pGCL [27, 25]. It is described by the BNF grammar,

$$P ::= \texttt{skip} \mid \texttt{a} \mid P ; P \mid P \parallel P \mid P +_{\texttt{p}} P \mid P + P \mid \texttt{if}\,\texttt{b}\,\texttt{then}\; P \;\texttt{else}\; P \mid \texttt{while}\,\texttt{b}\; P$$

where $\texttt{a}$ is a program from a pre-determined set of programs and $\texttt{b}$ is a condition from a pre-determined set of conditions. A program $P +_{\texttt{p}} Q$ ($\texttt{p} \in [0,1] \cap \mathbb{Q}$) represents a probabilistic choice between either the evaluation of P or Q. $P + Q$ is the non-deterministic analogue. The other program constructs are quite standard so we omit their explanation.

$$\frac{}{\langle \mathtt{a}, s\rangle \longrightarrow [\![\mathtt{a}]\!](s)} \qquad \frac{}{\langle \mathtt{skip}, s\rangle \longrightarrow 1\cdot s} \qquad \frac{\langle P,s\rangle \longrightarrow \sum_i p_i \cdot \langle P_i, s_i\rangle + \sum_j p_j \cdot s_j}{\langle P;Q,s\rangle \longrightarrow \sum_i p_i \cdot \langle P_i;Q, s_i\rangle + \sum_j p_j \cdot \langle Q, s_j\rangle}$$

$$\frac{\langle P,s\rangle \longrightarrow \sum_i p_i \cdot \langle P_i, s_i\rangle + \sum_j p_j \cdot s_j}{\langle P \parallel Q,s\rangle \longrightarrow \sum_i p_i \cdot \langle P_i \parallel Q, s_i\rangle + \sum_j p_j \cdot \langle Q, s_j\rangle}$$

$$\frac{\langle Q,s\rangle \longrightarrow \sum_i p_i \cdot \langle Q_i, s_i\rangle + \sum_j p_j \cdot s_j}{\langle P \parallel Q,s\rangle \longrightarrow \sum_i p_i \cdot \langle P \parallel Q_i, s_i\rangle + \sum_j p_j \cdot \langle P, s_j\rangle}$$

$$\frac{\langle P,s\rangle \longrightarrow \mu \qquad \langle Q,s\rangle \longrightarrow \nu}{\langle P +_{\mathtt{p}} Q,s\rangle \longrightarrow \mathtt{p}\cdot\mu + (1-\mathtt{p})\cdot\nu} \qquad \frac{\langle P,s\rangle \longrightarrow \mu}{\langle P + Q,s\rangle \longrightarrow \mu} \qquad \frac{\langle Q,s\rangle \longrightarrow \mu}{\langle P + Q,s\rangle \longrightarrow \mu}$$

$$\frac{[\![\mathtt{b}]\!](s) = \mathtt{tt}}{\langle \mathtt{if\ b\ then}\ P\ \mathtt{else}\ Q,s\rangle \longrightarrow 1\cdot\langle P,s\rangle} \qquad \frac{[\![\mathtt{b}]\!](s) = \mathtt{ff}}{\langle \mathtt{if\ b\ then}\ P\ \mathtt{else}\ Q,s\rangle \longrightarrow 1\cdot\langle Q,s\rangle}$$

$$\frac{[\![\mathtt{b}]\!](s) = \mathtt{tt}}{\langle \mathtt{while\ b}\ P,s\rangle \longrightarrow 1\cdot\langle P;\mathtt{while\ b}\ P,s\rangle} \qquad \frac{[\![\mathtt{b}]\!](s) = \mathtt{ff}}{\langle \mathtt{while\ b}\ P,s\rangle \longrightarrow 1\cdot s}$$

Figure 1: Small-step operational semantics

We now equip the language with a small-step operational semantics. The latter will be used later on for introducing a big-step counterpart via probabilistic scheduling. First we take an arbitrary set S of states, for each atomic program $\mathtt{a}$ we postulate the existence of a function $[\![\mathtt{a}]\!] : S \to \mathrm{V}_{=1,\omega}(S)$, and for each condition $\mathtt{b}$ we postulate the existence of a function $[\![\mathtt{b}]\!] : S \to \{\mathtt{tt},\mathtt{ff}\}$. Let Pr be the set of programs. The language's small-step operational semantics is the relation $\longrightarrow \subseteq (\mathrm{Pr}\times S)\times \mathrm{V}_{=1,\omega}(S + (\mathrm{Pr}\times S))$ that is generated inductively by the rules in Figure 1. The latter are a straightforward probabilistic extension of the rules presented in the classical reference [36, Chapters 6 and 8], so we omit their explanation.

Let us denote by $\langle P,s\rangle \longrightarrow$ the set $\{\mu \mid \langle P,s\rangle \longrightarrow \mu\}$. The small-step operational semantics enjoys the following key property for adequacy.

Proposition 3.1. *For every program P and state s the set $\langle P,s\rangle \longrightarrow$ is finite.*

Proof. Follows by induction over the syntactic structure of programs and by taking into account that finite sets are closed under binary unions and functional images. □

We now introduce a big-step operational semantics. As mentioned before it is based on the previous small-step semantics and the notion of a probabilistic scheduler [42]. Intuitively a scheduler resolves all non-deterministic choices that are encountered along the evaluation of a program P based on a history h of previous decisions and the current state s.

Definition 3.1. *A probabilistic scheduler $\mathcal{S}$ is a partial function,*

$$\mathcal{S} : \big((\mathrm{Pr}\times S)\times \mathrm{V}_{=1,\omega}(S + (\mathrm{Pr}\times S))\big)^* \times (\mathrm{Pr}\times S) \rightharpoonup \mathrm{V}_{=1,\omega}\mathrm{V}_{=1,\omega}(S + (\mathrm{Pr}\times S))$$

$$\frac{}{h\langle P,s\rangle \Downarrow^{\mathcal{S},0} \bot} \qquad \frac{\mathcal{S}(h\langle P,s\rangle) = \sum_k p_k \cdot v_k \qquad \forall k,i.\, h\langle P,s\rangle v_k \langle P_{k,i}, s_{k,i}\rangle \Downarrow^{\mathcal{S},n} \mu_{k,i}}{h\langle P,s\rangle \Downarrow^{\mathcal{S},n+1} \sum_k p_k \cdot \left(\sum_i p_{k,i} \cdot \mu_{k,i} + \sum_j p_{k,j} \cdot s_{k,j}\right)}$$

Figure 2: Big-step operational semantics

such that whenever $\mathcal{S}(h, \langle P,s\rangle)$ *is well-defined it is a distribution of valuations in* $\langle P,s\rangle \longrightarrow$ (i.e. *it is a distribution of the possible valuations of one-step transitions that originate from* $\langle P,s\rangle$*).*

Note that if $\mathcal{S}(h,\langle P,s\rangle)$ is well-defined it must be a linear combination $\sum_k p_k \cdot v_k$ of valuations v_k of the form $\sum_i p_{k,i} \cdot \langle P_{k,i}, s_{k,i}\rangle + \sum_j p_{k,j} \cdot s_{k,j}$. Every representation v_k is essentially unique by virtue of $S + (\mathrm{Pr} \times S)$ being discrete. Not only this, by Theorem 2.1 the space $\mathrm{V}_{=1,\omega}(S + (\mathrm{Pr} \times S))$ will also be discrete and thus the representation $\sum_k p_k \cdot v_k$ itself will be essentially unique as well. This is important for the definition of the big-step operational semantics and Proposition 3.2 below.

We will often denote a pair of the form $(h, \langle P,s\rangle)$ simply by $h\langle P,s\rangle$. The big-step operational semantics is defined as a relation $h\langle P,s\rangle \Downarrow^{\mathcal{S},n} \mu$ where $n \in \mathbb{N}$ is a natural number and $\mu \in \mathrm{V}_{\leq 1,\omega}(S)$ is a valuation. This relation represents an *n-step partial* evaluation of $\langle P,s\rangle$ w.r.t. $\mathcal{S}$ and h, and it is defined inductively by the rules in Figure 2.

Proposition 3.2. *Take a natural number* $n \in \mathbb{N}$*, configuration* $\langle P,s\rangle$*, history h, and scheduler* $\mathcal{S}$*. The big-step operational semantics has the following properties:*

- (Determinism) *if* $h\langle P,s\rangle \Downarrow^{\mathcal{S},n} \mu$ *and* $h\langle P,s\rangle \Downarrow^{\mathcal{S},n} \nu$ *then* $\mu = \nu$*;*
- (Monotonicity) *if* $h\langle P,s\rangle \Downarrow^{\mathcal{S},n} \mu$ *and* $h\langle P,s\rangle \Downarrow^{\mathcal{S},n+1} \nu$ *then* $\mu \leq \nu$*.*

Proof. Follows straightforwardly by induction over the natural numbers and by appealing to the fact that scaling and addition (of valuations) are monotone. □

The logic. We now introduce the aforementioned logic for reasoning about program properties. It is an instance of geometric propositional logic [43] that allows to express both must ($\Box$) and may ($\Diamond$) statistical termination ($\bigcirc_p$). It is two-layered specifically its formulae ϕ are given by,

$$\phi ::= \Box\,\varphi \mid \Diamond\,\varphi \mid \phi \wedge \phi \mid \bigvee \phi \mid \bot \mid \top \qquad\qquad \varphi ::= \bigcirc_p U \mid \varphi \wedge \varphi \mid \bigvee \varphi \mid \bot \mid \top$$

where U is a subset of the discrete state space S (*i.e.* U is an 'observable property' per our previous remarks) and $p \in \mathbb{R}_{\geq 0}$. Whilst the top layer handles the non-deterministic dimension (*e.g.* all possible interleavings) the bottom layer handles the probabilistic counterpart. Note that the disjunction clauses are not limited to a finite arity but are set-indexed – a core feature of geometric logic. The expression $\bigcirc_p U$ refers to a program 'terminating in U' with probability strictly greater than p. The formulae $\Box\,\varphi$ and $\Diamond\,\varphi$ correspond to universal and existential quantification respectively. For example $\Box \bigcirc_p U$ reads as "it is *necessarily* the case that the program at hand terminates in U with probability strictly greater than p" whilst $\Diamond \bigcirc_p U$ reads as "it is *possible* that the program at hand terminates in U with probability strictly greater than p". It may appear that the choice of such a logic for our language is somewhat *ad-hoc*, but one can easily see that it emerges as a natural candidate after inspecting the Scott topologies of the mixed powerdomains and recalling the motto 'observable properties as open sets' (recall the previous section). In this context conjunctions and disjunctions are interpreted respectively as intersections and unions of open sets.

The next step is to present a satisfaction relation between pairs $\langle P,s\rangle$ and formulae ϕ. To this effect we recur to the notion of a non-blocking scheduler which is presented next.

Definition 3.2. *Consider a pair $(h, \langle P,s\rangle)$ and a scheduler $\mathcal{S}$. We qualify $\mathcal{S}$ as* non-blocking w.r.t. $h\langle P,s\rangle$ *if for every natural number $n \in \mathbb{N}$ we have some valuation μ such that $h\langle P,s\rangle \Downarrow^{\mathcal{S},n} \mu$.*

We simply say that $\mathcal{S}$ is non-blocking – *i.e.* we omit the reference to $h\langle P,s\rangle$ – if the pair we are referring to is clear from the context.

Observe that according to our previous remarks every formula φ corresponds to an open set of $\mathrm{V}(S)$ – in fact we will treat φ as so – and define inductively the relation $\models$ between $\langle P,s\rangle$ and formulae ϕ as follows:

$$\langle P,s\rangle \models \Box\,\varphi \text{ iff } \textit{for all} \text{ non-blocking schedulers } \mathcal{S} \text{ w.r.t. } \langle P,s\rangle.\, \exists n \in \mathbb{N}, \mu \in \varphi.\, \langle P,s\rangle \Downarrow^{\mathcal{S},n} \mu$$

$$\langle P,s\rangle \models \Diamond\,\varphi \text{ iff } \textit{for some} \text{ scheduler } \mathcal{S}.\, \exists n \in \mathbb{N}, \mu \in \varphi.\, \langle P,s\rangle \Downarrow^{\mathcal{S},n} \mu$$

with all other cases defined standardly. The non-blocking condition is needed to ensure that degenerate schedulers do not trivially falsify formulae of the type $\Box\,\varphi$. For example the totally undefined scheduler always falsifies such formulae.

4 Denotational semantics

The intensional step. As alluded to in Section 1 the first step in defining our denotational semantics is to solve a domain equation of resumptions, with the branching structure given by a mixed powerdomain (Section 2). More specifically we solve,

$$X \cong \mathrm{P}_x\mathrm{V}(S + X \times S)^S \qquad\qquad (S \text{ a discrete domain}) \qquad\qquad (3)$$

for a chosen $x \in \{l,u,b\}$ and where $(-)^S$ is the S-indexed product construct. In order to not overburden notation we abbreviate P_x simply to P whenever the choice of x is unconstrained. We also denote the functor $\mathrm{P}_x\mathrm{V}(S + (-) \times S)^S$ by R_x and abbreviate the latter simply to R whenever the choice of x is unconstrained. Now, in the case that $x \in \{l,u\}$ the solution of Equation (3) is obtained standardly, more specifically by employing the standard final coalgebra construction, [12, Theorem IV-5.5], and the following straightforward proposition.

Proposition 4.1. *The functor R_x is locally continuous for every $x \in \{l,u,b\}$.*

The case $x \in \{b\}$ (which moves us from Dom to the category CohDom) is just slightly more complex: one additionally appeals to [12, Exercise IV-4.15]. Notably in all cases the solution thus obtained is always the greatest one *i.e.* the carrier νR of the final R-coalgebra. It is also the smallest one in a certain technical sense: it is the initial R-strict algebra [12, Chapter IV], a fact that follows from R being strict (*i.e.* it preserves strict morphisms) and [12, Theorem IV-4.5]. Both finality and initiality are crucial to our work: we will use the final coalgebra property to define the intensional semantics $(\!|-|\!)$ and then extensionally collapse the latter by recurring to the initial algebra property.

Let us thus proceed by detailing the intensional semantics. It is based on the notion of primitive corecursion [40] which we briefly recall next.

Theorem 4.1. *Let C be a category with binary coproducts and $F : \mathsf{C} \to \mathsf{C}$ be a functor with a final coalgebra* unfold $: \nu F \cong F(\nu F)$. *For every C-morphism $f : X \to F(\nu F + X)$ there exists a unique C-morphism $h : X \to \nu F$ that makes the following diagram commute.*

$$\begin{tikzcd} X \arrow[r, dotted, "h"] \arrow[d, "f"'] & \nu F \arrow[d, bend left, "\text{unfold}"] \arrow[d, phantom, "\cong"] \\ F(\nu F + X) \arrow[r, "F[\mathrm{id},h]"'] & F(\nu F) \arrow[u, bend left, "\text{fold}"] \end{tikzcd}$$

The morphism h is defined as the composition $\mathrm{corec}([F\mathrm{inl} \cdot \mathrm{unfold}, f]) \cdot \mathrm{inr}$*, where* $\mathrm{corec}([F\mathrm{inl} \cdot \mathrm{unfold}, f])$ *is the universal morphism induced by the F-coalgebra* $[F\mathrm{inl} \cdot \mathrm{unfold}, f] : \nu F + X \to F(\nu F + X)$.

With the notion of primitive corecursion at hand one easily defines operators to interpret both sequential and parallel composition, as detailed next.

Definition 4.1 (Sequential composition)**.** *Define the map $\rhd : \nu R \times \nu R \to \nu R$ as the morphism that is induced by primitive corecursion and the composition of the following continuous maps,*

$$\begin{aligned} \nu R \times \nu R &\xrightarrow{\mathrm{unfold} \times \mathrm{id}} \mathrm{PV}(S + \nu R \times S)^S \times \nu R \\ &\xrightarrow{\langle \pi_s \times \mathrm{id} \rangle_{s \in S}} (\mathrm{PV}(S + \nu R \times S) \times \nu R)^S \\ &\xrightarrow{\mathrm{str}^S} \mathrm{PV}((S + \nu R \times S) \times \nu R)^S \\ &\xrightarrow{\cong} \mathrm{PV}((\nu R + \nu R \times \nu R) \times S)^S \\ &\xrightarrow{\mathrm{PV}(\mathrm{inr})^S} \mathrm{PV}(S + (\nu R + \nu R \times \nu R) \times S)^S \end{aligned}$$

We denote this composite by seql.

Intuitively the operation seql unfolds the resumption on the left and performs a certain action depending on whether this resumption halts or resumes after receiving an input: if it halts then seql yields control to the resumption on the right; if it resumes seql simply attaches the resumption on the right to the respective continuation. Note that there exists an analogous operation seqr which starts by unfolding the resumption on the right. Note as well that $\rhd$ is continuous by construction.

Definition 4.2 (Parallel composition)**.** *Define the map $\bowtie : \nu R \times \nu R \to \nu R$ as the morphism that is induced by primitive corecursion and the continuous map* $\uplus^S \cdot \langle \pi_s \times \pi_s \rangle_{s \in S} \cdot \langle \mathrm{seql}, \mathrm{seqr} \rangle$.

We finally present our intensional semantics $(\!|-|\!)$. It is defined inductively on the syntactic structure of programs $P \in \mathrm{Pr}$ and assigns to each program P a denotation $(\!|P|\!) \in \nu R$. For simplicity we will often treat the map unfold as the identity. Note that every $(\!|P|\!)(s)$ (for $s \in S$) is thus an element of a cone with a semi-lattice operation $\uplus$ and that this algebraic structure extends to νR by pointwise extension. The denotational semantics is presented in Figure 3. The isomorphisms used in the last two equations denote the distribution of products over coproducts, more precisely they denote $S \times 2 \cong S + S$ with the (coherent) domain 2 defined as the coproduct $1 + 1$. These two equations also capitalise on the bijective correspondence between elements of νR and maps $S \to \mathrm{PV}(S + \nu R \times S)$. Note as well that in the case of conditionals we prefix the execution of $(\!|P|\!)$ (and $(\!|Q|\!)$) with $(\!|\mathtt{skip}|\!)$. This is to raise the possibility of the environment altering states between the evaluation of $\mathtt{b}$ and carrying on with the respective branch. An analogous approach is applied to the case of while-loops. Finally the fact that the map from which we take the least fixpoint (lfp) is continuous follows from $\rhd$, copairing, and pre-composition being continuous.

The denotational semantics thus defined may seem complex, but it actually has a quite simple characterisation that involves the small-step operational semantics. This is given in the following theorem (and corresponds to the commutativity of the left rectangle in Diagram 2).

Theorem 4.2. *For every program $P \in \mathrm{Pr}$ the denotation $(\!|P|\!)$ is (up-to isomorphism) equal to the map,*

$$s \mapsto x\left(\left\{\textstyle\sum_i p_i \cdot \langle (\!|P_i|\!), s_i \rangle + \sum_j p_j \cdot s_j \mid \langle P, s \rangle \longrightarrow \sum_i p_i \cdot \langle P_i, s_i \rangle + \sum_j p_j \cdot s_j \right\}\right)$$

Proof. Follows from straightforward induction over the syntactic structure of programs, where for the case of while-loops we resort to the fixpoint equation. □

$$\begin{aligned}
\llparenthesis \texttt{skip} \rrparenthesis &= s \mapsto x(\{1 \cdot s\}) \\
\llparenthesis \texttt{a} \rrparenthesis &= s \mapsto x(\{ [\![\texttt{a}]\!] (s)\}) \\
\llparenthesis P; Q \rrparenthesis &= \llparenthesis P \rrparenthesis \triangleright \llparenthesis Q \rrparenthesis \\
\llparenthesis P \parallel Q \rrparenthesis &= \llparenthesis P \rrparenthesis \bowtie \llparenthesis Q \rrparenthesis \\
\llparenthesis P +_{\texttt{p}} Q \rrparenthesis &= \texttt{p} \cdot \llparenthesis P \rrparenthesis + (1 - \texttt{p}) \cdot \llparenthesis Q \rrparenthesis \\
\llparenthesis P + Q \rrparenthesis &= \llparenthesis P \rrparenthesis \uplus \llparenthesis Q \rrparenthesis \\
\llparenthesis \texttt{if}\, \texttt{b}\, \texttt{then} P\, \texttt{else}\, Q \rrparenthesis &= [\llparenthesis \texttt{skip} \rrparenthesis \triangleright \llparenthesis Q \rrparenthesis, \llparenthesis \texttt{skip} \rrparenthesis \triangleright \llparenthesis P \rrparenthesis] \cdot \cong \cdot \langle \mathrm{id}, [\![\texttt{b}]\!] \rangle \\
\llparenthesis \texttt{while}\, \texttt{b}\, P \rrparenthesis &= \mathrm{lfp} \left(r \mapsto [\llparenthesis \texttt{skip} \rrparenthesis, \llparenthesis \texttt{skip} \rrparenthesis \triangleright (\llparenthesis P \rrparenthesis \triangleright r)] \cdot \cong \cdot \langle \mathrm{id}, [\![\texttt{b}]\!] \rangle \right)
\end{aligned}$$

Figure 3: Intensional semantics

The extensional collapse. We now present the extensional collapse of the intensional semantics. Intuitively given a program P the collapse removes all intermediate computational steps of $\llparenthesis P \rrparenthesis$, which as mentioned previously is a 'resumptions tree'. Technically the collapse makes crucial use of the fact that the domain νR is not only the final coalgebra of R-coalgebras but also the initial algebra of strict R-algebras. More specifically we define the extensional collapse as the initial strict algebra morphism,

$$\begin{tikzcd}
\nu R \arrow[rr, dotted, "\mathrm{ext}"] \arrow[d, bend left, "\mathrm{unfold}"] \arrow[d, phantom, "\cong"] & & \mathrm{PV}(S)^S \\
\mathrm{PV}(S + \nu R \times S)^S \arrow[u, bend left, "\mathrm{fold}"] \arrow[rr, "\mathrm{PV}(\mathrm{id} + \mathrm{ext} \times \mathrm{id})^S"'] & & \mathrm{PV}(S + \mathrm{PV}(S)^S \times S)^S \arrow[u, "{[\eta, \mathrm{app}]^{\star S}}"']
\end{tikzcd}$$

Concretely the extensional collapse is defined by $\mathrm{ext}(\llparenthesis P \rrparenthesis) = \bigvee_{n \in \mathbb{N}} \mathrm{ext}_n(\llparenthesis P \rrparenthesis_n)$ where $\llparenthesis P \rrparenthesis_n = \pi_n(\llparenthesis P \rrparenthesis)$ with $\pi_n : \nu R \to R^n(1)$ (recall that νR is a certain limit with projections $\pi_n : \nu R \to R^n(1)$). The maps ext_n on the other hand are defined inductively over the natural numbers (and by recurring to the monad laws) by,

$$\begin{aligned}
\mathrm{ext}_0 &= \bot \mapsto (s \mapsto x(\{\bot\})) : R^0(1) \to \mathrm{PV}(S)^S \\
\mathrm{ext}_{n+1} &= [\eta, \mathrm{app} \cdot (\mathrm{ext}_n \times \mathrm{id})]^{\star S} : R^{n+1}(1) \to \mathrm{PV}(S)^S
\end{aligned}$$

The map ext enjoys several useful properties: it is both strict and continuous (by construction). It is also both linear and $\uplus$-preserving since it is a supremum of such maps. In the appendix (Section A) we show that if the parallel operator is dropped then the semantics obtained from the composite $\mathrm{ext} \cdot \llparenthesis - \rrparenthesis$ (hencerforth denoted by $[\![-]\!]$) is really just a standard monadic semantics induced by the monad PV. In other words, the concurrent semantics obtained from $\mathrm{ext} \cdot \llparenthesis - \rrparenthesis$ is a conservative extension of the latter.

We continue unravelling key properties of the composition $\mathrm{ext} \cdot \llparenthesis - \rrparenthesis$. More specifically we will now show that for every natural number $n \in \mathbb{N}$, program P, and state s, the set $\mathrm{ext}_n(\llparenthesis P \rrparenthesis_n)(s)$ is always *finitely generated*, *i.e.* of the form $x(F)$ for a certain *finite* set F. We will use this property to prove a proposition which formally connects the operational and the extensional semantics $[\![-]\!]$ w.r.t. n-step evaluations. Thus, the fact that every $\mathrm{ext}_n(\llparenthesis P \rrparenthesis_n)(s)$ is finitely generated follows by induction over the natural numbers and by straightforward calculations that yield the equations,

$$\begin{aligned}
\mathrm{ext}_0(\llparenthesis P \rrparenthesis_0)(s) &= x(\{\bot\}) \\
\mathrm{ext}_{n+1}(\llparenthesis P \rrparenthesis_{n+1})(s) &= x\big(\bigcup \big\{ \textstyle\sum_i p_i \cdot F_i + \sum_j p_j \cdot s_j \mid \langle P, s \rangle \longrightarrow \sum_i p_i \cdot \langle P_i, s_i \rangle + \sum_j p_j \cdot s_j \big\}\big)
\end{aligned} \tag{4}$$

with every F_i a finite set that satisfies $\mathrm{ext}_n((\!|P_i|\!)_n)(s_i) = x(F_i)$. Note that the previous equations provide an explicit construction of a finite set F such that $x(F) = \mathrm{ext}_n((\!|P|\!)_n)(s)$. From this we obtain the formulation and proof of Proposition 4.2, which we present next.

Proposition 4.2. *Take a program P, state s, and natural number n. The equation below holds.*

$$\mathrm{conv}\, F = \left\{ \mu \mid \langle P,s \rangle \Downarrow^{\mathcal{S},n} \mu \textit{ with } \mathcal{S} \textit{ a scheduler} \right\}$$

Proof. First recall that for a natural number $n \in \mathbb{N}$, a program P, and a state s, we use the letter F to denote the finite set that generates $\mathrm{ext}_n((\!|P|\!)_n)(s)$. Note as well that from [39, Lemma 2.8] we obtain,

$$\begin{aligned} \mathrm{conv}\, F &= \mathrm{conv}\big(\bigcup \big\{ \textstyle\sum_i p_i \cdot F_i + \sum_j p_j \cdot s_j \mid \langle P,s \rangle \longrightarrow \sum_i p_i \cdot \langle P_i, s_i \rangle + \sum_j p_j \cdot s_j \big\}\big) \\ &= \mathrm{conv}\big(\bigcup \big\{ \textstyle\sum_i p_i \cdot \mathrm{conv}\, F_i + \sum_j p_j \cdot s_j \mid \langle P,s \rangle \longrightarrow \sum_i p_i \cdot \langle P_i, s_i \rangle + \sum_j p_j \cdot s_j \big\}\big) \end{aligned} \tag{5}$$

We crucially resort to this observation for the proof, as detailed next. As stated in the proposition's formulation we need to prove that the equation,

$$\mathrm{conv}\, F = \left\{ \mu \mid \langle P,s \rangle \Downarrow^{\mathcal{S},n} \mu \text{ with } \mathcal{S} \text{ a scheduler} \right\}$$

holds. We will show that the two respective inclusions hold, starting with the case $\supseteq$. The proof follows by induction over the natural numbers and by strengthening the induction invariant to encompass all histories h and not just the empty one. The base case is direct. For the inductive step $n+1$ assume that $h\langle P,s \rangle \Downarrow^{\mathcal{S},n+1} \mu$ for some valuation μ and scheduler $\mathcal{S}$. Then according to the deductive rules of the big-step operational semantics (Figure 2) we obtain the following conditions:

1. $\mathcal{S}(h\langle P,s \rangle) = \sum_k p_k \cdot \nu_k$ for some convex combination $\sum_k p_k \cdot (-)$. Moreover $\forall k.\, \langle P,s \rangle \longrightarrow \nu_k$ with each valuation ν_k of the form $\sum_i p_{k,i} \cdot \langle P_{k,i}, s_{k,i} \rangle + \sum_j p_{k,j} \cdot s_{k,j}$;
2. $\forall k,i.\, h\langle P,s \rangle \nu_k \langle P_{k,i}, s_{k,i} \rangle \Downarrow^{\mathcal{S},n} \mu_{k,i}$ for some valuation $\mu_{k,i}$;
3. and finally $\mu = \sum_k p_k \cdot \big(\sum_i p_{k,i} \cdot \mu_{k,i} + \sum_j p_{k,j} \cdot s_{k,j} \big)$.

It follows from the induction hypothesis and the second condition that for all k,i the valuation $\mu_{k,i}$ is in $\mathrm{conv}\, F_{k,i}$, where $F_{k,i}$ is the finite set that is inductively built from $P_{k,i}$ and $s_{k,i}$ as previously described. Thus for all k each valuation $\sum_i p_{k,i} \cdot \mu_{k,i} + \sum_j p_{k,j} \cdot s_{k,j}$ is an element of the set $\sum_i p_{k,i} \cdot \mathrm{conv}\, F_{k,i} + \sum_j p_{k,j} \cdot s_{k,j}$. Also the latter is contained in $\mathrm{conv}\, F$ according to Equations (5) and the first condition. The proof then follows from the third condition and the fact that $\mathrm{conv}\, F$ is convex.

Let us now focus on the inclusion $\subseteq$. The base case is again direct. For the inductive step $n+1$ recall Equation (5) and take a convex combination $\sum_k p_k \cdot \mu_k$ in the set,

$$\mathrm{conv}\, F = \mathrm{conv}\big(\bigcup \big\{ \textstyle\sum_i p_i \cdot \mathrm{conv} F_i + \sum_j p_j \cdot s_j \mid \langle P,s \rangle \longrightarrow \sum_i p_i \cdot \langle P_i, s_i \rangle + \sum_j p_j \cdot s_j \big\}\big)$$

We can safely assume that every μ_k belongs to some set $\sum_i p_{k,i} \cdot \mathrm{conv}\, F_{k,i} + \sum_j p_{k,j} \cdot s_{k,j}$ generated by a corresponding valuation $\sum_i p_{k,i} \cdot \langle P_{k,i}, s_{k,i} \rangle + \sum_j p_{k,j} \cdot s_{k,j} = \nu_k \in \langle P,s \rangle \longrightarrow$. Each set $F_{k,i}$ is inductively built from $P_{k,i}$ and $s_{k,i}$ in the way that was previously described. Crucially we can also safely assume that all valuations ν_k involved are pairwise distinct, by virtue of all sets $\mathrm{conv}\, F_{k,i}$ being convex and by recurring to the normalisation of subconvex valuations.

Next, by construction every μ_k is of the form $\sum_i p_{k,i} \cdot \mu_{k,i} + \sum_j p_{k,j} \cdot s_{k,j}$ with $\mu_{k,i} \in \mathrm{conv}\, F_{k,i}$, and by the induction hypothesis we deduce that $\forall k,i.\, \langle P_{k,i}, s_{k,i} \rangle \Downarrow^{\mathcal{S}_{k,i},n} \mu_{k,i}$ for some scheduler $\mathcal{S}_{k,i}$. Our next step is to construct a single scheduler $\mathcal{S}$ from all schedulers $\mathcal{S}_{k,i}$. We set $\mathcal{S}(\langle P,s \rangle) = \sum_k p_k \cdot \nu_k$ and

$\forall k,i.\, \mathcal{S}(\langle P,s\rangle \nu_k l) = \mathcal{S}_{k,i}(l)$ for every input l with $\langle P_{k,i}, s_{k,i}\rangle$ as prefix. We set $\mathcal{S}$ to be undefined w.r.t. all other inputs. One then easily proves by induction over the natural numbers and by inspecting the definition of the big-step operational semantics (Figure 2) that the equivalence,

$$\langle P,s\rangle \nu_k \langle P_{k,i}, s_{k,i}\rangle \Downarrow^{\mathcal{S},m} \mu \text{ iff } \langle P_{k,i}, s_{k,i}\rangle \Downarrow^{\mathcal{S}_{k,i},m} \mu$$

holds for all k,i and natural numbers $m \in \mathbb{N}$.

Finally one just needs to apply the definition of the big-step operational semantics (Figure 2) to obtain $\langle P,s\rangle \Downarrow^{\mathcal{S},n+1} \sum_k p_k \cdot \mu_k$ as previously claimed. □

5 Computational adequacy

In this section we prove the paper's main result: the semantics $[\![-]\!]$ in Section 4 is adequate w.r.t. the operational semantics in Section 3. As mentioned in Section 1 the formulation of adequacy relies on the logic that was presented in Section 3. Actually it relies on different fragments of it depending on which mixed powerdomain one adopts: in the lower case (*i.e.* angelic or may non-determinism) formulae containing $\Box\,\varphi$ are forbidden whilst in the upper case (*i.e.* demonic or must non-determinism) formulae containing $\Diamond\,\varphi$ are forbidden. The biconvex case does not impose any restriction, *i.e.* we have the logic *verbatim*. For simplicity, we will use $\mathcal{L}_l$, $\mathcal{L}_u$, and $\mathcal{L}_b$ to denote respectively the lower, upper, and biconvex fragments of the logic.

Recall from Section 3 that we used pairs $\langle P,s\rangle$ and the operational semantics of concurrent pGCL to interpret the logic's formulae. Recall as well that we treat a formula φ as an open subset of the space $\mathrm{V}(S)$ with S discrete. As the next step towards adequacy, note that the elements of $\mathrm{PV}(S)$ also form an interpretation structure for the logic: specifically we define a satisfaction relation $\models$ between the elements $A \in \mathrm{PV}(S)$ and formulae ϕ by,

$$A \models \Diamond\,\varphi \text{ iff } \textit{for some } \mu \in A \text{ we have } \mu \in \varphi \qquad A \models \Box\,\varphi \text{ iff } \textit{for all } \mu \in A \text{ we have } \mu \in \varphi$$

with the remaining cases defined standardly. The formulation of computational adequacy then naturally arises: for every program P and state s, the equivalence below holds for all formulae ϕ.

$$\langle P,s\rangle \models \phi \text{ iff } [\![P]\!](s) \models \phi \tag{6}$$

Of course depending on which mixed powerdomain one adopts ϕ's universe of quantification will vary according to the corresponding fragment $\mathcal{L}_x$ ($x \in \{l,u,b\}$). The remainder of the current section is devoted to proving Equivalence (6). Actually the focus is only on formulae of the type $\Diamond\,\varphi$ and $\Box\,\varphi$, for one can subsequently apply straightforward induction (over the formulae's syntactic structure) to obtain the claimed equivalence for all ϕ of the corresponding fragment. We start with the angelic case.

Theorem 5.1. *Let* $\mathrm{P}_x\mathrm{V}$ *be either the lower convex powerdomain or the biconvex variant (*i.e. $x \in \{l,b\}$*). Then for every program P, state s, and formula* $\Diamond\,\varphi$ *the equivalence below holds.*

$$\langle P,s\rangle \models \Diamond\,\varphi \textit{ iff } [\![P]\!](s) \models \Diamond\,\varphi$$

Proof. The left-to-right direction follows from Proposition 4.2 and the upper-closedness of the open $\Diamond\,\varphi$. The right-to-left direction uses Proposition 4.2, the inaccessibility of the open $\Diamond\,\varphi$, the characterisation of Scott-closure in domains [14, Exercise 5.1.14], and the inaccessibility of the open φ. □

As already mentioned in the introduction, Equivalence (6) w.r.t. formulae of the type $\Box\,\varphi$ is much thornier to prove. In order to achieve it we will need the following somewhat surprising result.

Theorem 5.2. *Consider a program P, a state s, and a formula $\Box\,\varphi$. If $\langle P,s\rangle \models \Box\,\varphi$ then there exists a positive natural number $\mathbf{z} \in \mathbb{N}_+$ such that* for all *non-blocking schedulers $\mathcal{S}$ we have,*

$$\langle P,s\rangle \Downarrow^{\mathcal{S},\mathbf{z}} \mu \textit{ for some valuation } \mu \textit{ and } \mu \in \varphi$$

In other words whenever $\langle P,s\rangle \models \Box\,\varphi$ there exists an upper bound – *i.e.* a natural number $\mathbf{z} \in \mathbb{N}_+$ such that *all* non-blocking schedulers (which are *uncountably* many) reach condition φ in *at most* $\mathbf{z}$-steps. Such a property becomes perhaps less surprising when one recalls König's lemma [10]. The latter in a particular form states that every finitely-branching tree with each path of finite length must have *finite* depth (which means that there is an upper-bound on the length of all paths). From this perspective each non-blocking scheduler intuitively corresponds to a path and the fact that each path has finite length corresponds to the assumption that each non-blocking scheduler reaches condition φ in a finite number of steps (*i.e.* $\langle P,s\rangle \models \Box\,\varphi$).

Our proof of Theorem 5.2 is based on a topological generalisation of the previous analogy to König's lemma, in which among other things the condition 'finitely-branching' is generalised to 'compactly-branching' and the notion of a path is converted to that of a 'probabilistic trace' generated by a scheduler. The technical details of this proof and a series of auxiliary results can be consulted in the appendix (Section B). Finally by appealing to Theorem 5.2 we obtain the desired equivalence and establish computational adequacy.

Theorem 5.3. *Let* $\mathrm{P}_x\mathrm{V}$ *be the upper convex powerdomain or the biconvex variant (*i.e. $x \in \{u,b\}$*). Then for every program P, state s, and formula $\Box\,\varphi$ the equivalence below holds.*

$$\langle P,s\rangle \models \Box\,\varphi \textit{ iff } [\![P]\!](s) \models \Box\,\varphi$$

Proof. The left-to-right direction follows from Theorem 5.2, Proposition 4.2 and the upper-closedness of the opens φ and $\Box\,\varphi$. The right-to-left direction follows from the inaccessibility of the open $\Box\,\varphi$ and Proposition 4.2. □

We can now also derive the full abstraction result mentioned in Section 1. Specifically given programs P, Q, and state s the observational preorder $\lesssim_x$ is defined by,

$$\langle P,s\rangle \lesssim_x \langle Q,s\rangle \text{ iff } \Big(\forall \phi \in \mathcal{L}_x.\, \langle P,s\rangle \models \phi \text{ implies } \langle Q,s\rangle \models \phi\Big)$$

Then by taking advantage of computational adequacy we achieve full abstraction.

Corollary 5.1. *Choose one of the three mixed powerdomains* $\mathrm{P}_x\mathrm{V}$ $(x \in \{l,u,b\})$*, a program P, a program Q, and a state s. The following equivalence holds.*

$$\langle P,s\rangle \lesssim_x \langle Q,s\rangle \textit{ iff } [\![P]\!](s) \leq_x [\![Q]\!](s)$$

Proof. Follows directly from Theorem 5.1, Theorem 5.3, [14, Proposition 4.2.4 and Proposition 4.2.18]. □

6 Concluding notes: semi-decidability, quantum, and future work

Semi-decidability. The tradition of finding denotational counterparts to operational aspects of programming languages is well-known and typically very fruitful (see for example [47, 36]). Our work does not deviate from this tradition and it introduces the two following families of equivalences,

$$\langle P,s\rangle \models \Diamond\varphi \text{ iff } [\![P]\!](s) \models \Diamond\varphi \qquad\qquad \langle P,s\rangle \models \Box\varphi \text{ iff } [\![P]\!](s) \models \Box\varphi$$

for appropriate choices of mixed powerdomains $\mathrm{P}_x\mathrm{V}$ ($x \in \{l,u,b\}$). We also saw that from these it follows a full abstraction theorem w.r.t. the observational preorder $\lesssim$ described in Section 1. Such results thus equip our concurrent language with a rich collection of domain-theoretic tools that one can appeal to in the analysis of several of its aspects. We briefly illustrate this point next with one example.

Statistical termination is a topic that has been extensively studied over the years in probability theory [18, 23, 24]. It started gaining traction recently in the quantum setting as well [11]. Here we prove semi-decidability w.r.t. a main class of may and must statistical termination, an apparently surprising result for it involves quantifications over the uncountable set of probabilistic schedulers.

We first need to introduce some basic assumptions due to concurrent pGCL being parametrised. Take a program P and a state s. We assume that for every natural number $n \in \mathbb{N}$ one can always compute via Equations (4) the finite set F_n that generates $\mathrm{ext}_n((\!|P|\!)_n)(s)$ (recall the end of Section 4). Specifically we assume that every interpretation $[\![\mathtt{a}]\!]$ of an atomic program is computable and only returns linear combinations whose scalars are rational numbers. Similarly all interpretations $[\![\mathtt{b}]\!]$ of conditions must be decidable. Finally given a subset $U \subseteq S$ we assume that the membership function $\in_U : S \to \{\mathtt{tt},\mathtt{ff}\}$ w.r.t. U is decidable.

We start with may statistical termination as given by the formula $\Diamond \bigcirc_p U$ with p any number in the set $[0,1) \cap \mathbb{Q}$. For this case we fix the mixed powerdomain in our denotational semantics to be the lower convex one. Next, recall that the statement $\langle P,s\rangle \models \Diamond\bigcirc_p U$ refers to the existence of a scheduler under which $\langle P,s\rangle$ terminates in U with probability strictly greater than p. Our algorithmic procedure for checking whether such a statement holds is to exhaustively search in the finite sets $F_1, F_2, \ldots$ for a valuation μ that satisfies $\bigcirc_p U$. Such a procedure is computable due to the assumptions above, and in order to prove semi-decidability we show next that it eventually terminates whenever $\langle P,s\rangle \models \Diamond\bigcirc_p U$ holds.

Let us thus assume that $\langle P,s\rangle \models \Diamond\bigcirc_p U$ holds. It follows from Theorem 5.1 and the fact that $\Diamond\bigcirc_p U$ corresponds to an open set of the lower convex powerdomain that there exists a natural number $n \in \mathbb{N}$ such that $\mathrm{ext}_n((\!|P|\!)_n)(s) \models \Diamond\bigcirc_p U$. In other words the set $\overline{\mathrm{conv}\, F_n}$ contains a valuation that satisfies $\bigcirc_p U$ and by the characterisation of Scott-closure in domains [14, Exercise 5.1.14] there also exists a valuation $\mu \in \mathrm{conv}\, F_n$ that satisfies $\bigcirc_p U$. We will show by contradiction that such a valuation exists in F_n as well which proves our claim. By the definition of convex closure we know that $\mu = \sum_{i\in I} p_i \cdot \mu_i$ for some finite convex combination of valuations in F_n. Suppose that none of the valuations μ_i $(i \in I)$ satisfy $\mu_i(U) > p$. From this supposition we take the valuation μ_i $(i \in I)$ with the *largest* value $\mu_i(U)$, denote it by ρ, and clearly $\rho(U) \leq p$. This entails $\sum_i p_i \cdot \rho(U) \leq p$ but $\sum_i p_i \cdot \rho(U) \geq \sum_i p_i \cdot \mu_i(U) > p$, a contradiction.

We now focus on must statistical termination as given by $\Box\bigcirc_p U$ with $p \in [0,1) \cap \mathbb{Q}$. For this case we fix the mixed powerdomain in our denotational semantics to be the upper convex one. Recall that the statement $\langle P,s\rangle \models \Box\bigcirc_p U$ asserts that $\langle P,s\rangle$ terminates in U with probability strictly greater than p under all non-blocking schedulers. Our algorithmic procedure for checking whether such a statement holds is in some sense dual to the previous one: we exhaustively search for a finite set $F_1, F_2, \ldots$ in which all valuations satisfy $\bigcirc_p U$. As before such a procedure is computable, and to achieve semi-decidability we will prove that it eventually terminates whenever $\langle P,s\rangle \models \Box\bigcirc_p U$ holds. Thus assume that the latter holds. It follows from Theorem 5.3 and the fact that $\Box \bigcirc_p U$ corresponds to an open set of the upper convex powerdomain that there exists a natural number $n \in \mathbb{N}$ such that $\mathrm{ext}_n((\!|P|\!)_n)(s) \models \Box \bigcirc_p U$. All valuations in $\uparrow\!\mathrm{conv}\, F_n$ thus satisfy $\bigcirc_p U$ and therefore all valuations in F_n satisfy $\bigcirc_p U$ as well.

Our reasoning for semi-decidability w.r.t. may statistical termination scales up to formulae of the type $\Diamond(\bigcirc_{p_1} U_1 \vee \cdots \vee \bigcirc_{p_n} U_n)$ with all numbers $p_i \in [0,1) \cap \mathbb{Q}$. Dually our reasoning w.r.t the must variant extends to formulae of the type $\Box(\bigcirc_{p_1} U_1 \wedge \cdots \wedge \bigcirc_{p_n} U_n)$ with all numbers $p_i \in [0,1) \cap \mathbb{Q}$.

An application to concurrent quantum computation. We now apply our results to quantum concurrency, specifically we instantiate the language in Section 3 to the quantum setting [31, 45] and obtain computational adequacy for free. Previous works have already introduced denotational semantics to sequential, non-deterministic quantum programs [8, 9]. Unlike us however they do not involve powerdomain structures, although they do mention such would be more elegant. Also as far as we aware they do not establish any connection to an operational semantics, and thus our results are novel already for the sequential fragment.

We first present our concurrent quantum language, and subsequently the corresponding state space and interpretation of atomic programs. We assume that we have at our disposal n bits and m qubits. We use $\mathtt{x_1}, \mathtt{x_2}, \ldots, \mathtt{x_n}$ to identify each bit available and analogously for $\mathtt{q_1}, \mathtt{q_2}, \ldots, \mathtt{q_m}$ and qubits. As for the atomic programs we postulate a collection of gate operations $\mathtt{U}(\tilde{\mathtt{q}})$ where $\tilde{\mathtt{q}}$ is a list of qubits; we also have qubit resets $\mathtt{q_i} \leftarrow |0\rangle$ $(1 \leq \mathtt{i} \leq m)$ and measurements $\mathtt{M}[\mathtt{x_i} \leftarrow \mathtt{q_j}]$ of the $\mathtt{j}$-th qubit $(1 \leq \mathtt{j} \leq m)$ with the outcome stored in the $\mathtt{i}$-th bit $(1 \leq \mathtt{i} \leq n)$. The conditions $\mathtt{b}$ are set as the elements of the free Boolean algebra generated by the equations $\mathtt{x_i} = \mathtt{0}$ and $\mathtt{x_i} = \mathtt{1}$ for $0 \leq \mathtt{i} \leq n$.

Recall that a (pure) 2-dimensional quantum state is a unit vector $|\psi\rangle$ in $\mathbb{C}^2$, usually represented as a density operator $|\psi\rangle\langle\psi| \in \mathbb{C}^{2\times 2}$. Here we are particularly interested on the so-called classical-quantum states [45]. They take the form of a convex combination,

$$\textstyle\sum_i p_i \cdot |x_i\rangle\langle x_i| \otimes |\psi_i\rangle\langle\psi_i| \tag{7}$$

where each x_i is an element of 2^n (*i.e.* a classical state) and each $|\psi_i\rangle$ is a pure quantum state in $\mathbb{C}^{2^{\otimes m}}$. Such elements are thus distributions of n-bit states $|x_i\rangle\langle x_i|$ paired with m-qubit states $|\psi_i\rangle\langle\psi_i|$. The state space S that we adopt is the subset of $\mathbb{C}^{2\times 2^{\otimes n}} \otimes \mathbb{C}^{2\times 2^{\otimes m}}$ that contains precisely the elements of the form $1 \cdot |x\rangle\langle x| \otimes |\psi\rangle\langle\psi|$ as previously described – *i.e.* we know with certainty that $|x\rangle\langle x| \otimes |\psi\rangle\langle\psi|$ is the current state of our classical-quantum system. We call such states *pure* classical-quantum states.

We then interpret atomic programs as maps $S \to \mathrm{V}_{=1,\omega}(S)$ which as indicated by their signature send pure classical-quantum states into arbitrary ones (7). More technically we interpret such programs as restrictions of completely positive trace-preserving operators, a standard approach in the field of quantum information [45]. Completely positive trace-preserving operators are often called quantum channels and we will use this terminology throughout the section.

We will need a few preliminaries for interpreting the atomic programs. First a very useful quantum channel is the trace operation $\mathrm{Tr} : \mathbb{C}^{2\times 2^{\otimes n}} \to \mathbb{C}^{2\times 2^{\otimes 0}} = \mathbb{C}$ which returns the trace of a given matrix [45, Corollary 2.19]. Among other things it induces the partial trace on m-qubits,

$$\mathrm{Tr}_i := (\otimes_{j=1}^{i-1} \mathrm{id}) \otimes \mathrm{Tr} \otimes (\otimes_{j=i+1}^{m} \mathrm{id})$$

which operationally speaking discards the i-th qubit. It is also easy to see that for every density operator $\rho \in \mathbb{C}^{2\times 2^{\otimes n}}$ the map $\rho : \mathbb{C}^{2\times 2^{\otimes 0}} \to \mathbb{C}^{2\times 2^{\otimes n}}$ defined by $1 \mapsto \rho$ is a quantum channel [45, Proposition 2.17]. Finally note that one can always switch the positions i and j of two qubits [45, Corollary 2.27] in a pure quantum state.

We now present the interpretation of atomic programs. First for each gate operation $\mathtt{U}(\tilde{\mathtt{q}})$ we postulate the existence of a unitary operator $U : \mathbb{C}^{2^{\otimes m}} \to \mathbb{C}^{2^{\otimes m}}$. Then we interpret gate operations and resets by,

$$\llbracket \mathtt{U}(\tilde{\mathtt{q}}) \rrbracket \big(|x\rangle\langle x| \otimes |\psi\rangle\langle\psi|\big) = |x\rangle\langle x| \otimes U|\psi\rangle\langle\psi|U^\dagger$$
$$\llbracket \mathtt{q_i} \leftarrow |0\rangle \rrbracket \big(|x\rangle\langle x| \otimes |\psi\rangle\langle\psi|\big) = |x\rangle\langle x| \otimes \mathrm{mov_i}\big(|0\rangle\langle 0| \otimes \mathrm{Tr_i}|\psi\rangle\langle\psi|\big)$$

where $U^\dagger$ is the adjoint of U and $\mathrm{mov_i}$ is the operator that moves the leftmost state (in this case $|0\rangle\langle 0|$) to the $\mathtt{i}$-th position in the list of qubits. In order to interpret measurements we will need a few extra auxiliary operators. First we take the 'quantum-to-classical' channel $\Phi : \mathbb{C}^{2\times 2} \to \mathbb{C}^{2\times 2^{\otimes 2}}$ defined by,

$$\begin{pmatrix} a & b \\ c & d \end{pmatrix} \mapsto a \cdot |0\rangle\langle 0|^{\otimes 2} + d \cdot |1\rangle\langle 1|^{\otimes 2}$$

The fact that it is indeed a channel follows from [45, Theorem 2.37]. In words $\Phi(\rho)$ is a measurement of ρ w.r.t. the computational basis: the outcome $|0\rangle$ is obtained with probability a and analogously for $|1\rangle$. In case we measure $|0\rangle$ we return $|0\rangle\langle 0|^{\otimes 2}$ and analogously for $|1\rangle$. Note the use of $|0\rangle\langle 0|^{\otimes 2}$ (and not $|0\rangle\langle 0|$) so that we can later store a copy of $|0\rangle\langle 0|$ in the classical register. To keep the notation easy to read we use Φ_i to denote the operator that applies Φ in the i-th position and the identity everywhere else. Finally we have,

$$\llbracket \mathtt{M}[\mathtt{x_i} \leftarrow \mathtt{q_j}] \rrbracket \,(|x\rangle\langle x| \otimes |\psi\rangle\langle\psi|) = \mathrm{mov}^{\mathrm{cq}}_{\mathtt{j,i}} \left(\mathrm{Tr}_{\mathtt{i}}|x\rangle\langle x| \otimes \Phi_{\mathtt{j}}|\psi\rangle\langle\psi|\right)$$

where $\mathrm{mov}^{\mathrm{cq}}_{\mathtt{j,i}}$ sends the $\mathtt{j}$-th qubit to the $\mathtt{i}$-th position in the list of bits.

Future work. We plan to explore a number of research lines that stem directly from our work. For example we would like to expand our (brief) study about semi-decidability w.r.t. $\langle P, s\rangle \models \phi$. More specifically we would like to determine the largest class of formulae ϕ in our logic (recall Section 3) under which one can prove semi-decidability. Second it is well-known that under mild conditions the three mixed powerdomains are isomorphic to the so-called prevision models [15]. In other words via Section 4 and Section 5 we obtain for free a connection between our (concurrent) language and yet another set of domain-theoretic tools. Will this unexplored connection provide new insights about the language? It would also be interesting to investigate in what ways a semantics based on event structures [41] complements the one presented here.

We also plan to extend concurrent pGCL in different directions. The notion of fair scheduling for example could be taken into account which leads to countable non-determinism. We conjecture that the notion of a mixed powerdomain will need to be adjusted to this new setting, similarly to what already happens when probabilities are not involved [2]. Another appealing case is the extension of our concurrent language and associated results to the higher-order setting, obtaining a language similar in spirit to concurrent idealised Algol [4]. A promising basis for this is the general theory of PCF combined with algebraic theories [33, 35, 34]. In our case the algebraic theory adopted would need to be a combination of that of states and that of mixed non-determinism.

References

[1] Jirı Adámek, Horst Herrlich & George E Strecker (2009): *Abstract and Concrete Categories; The Joy of Cats 2004*. Reprints in Theory and Applications of Categories.

[2] Krzysztof R Apt & Gordon D Plotkin (1986): *Countable nondeterminism and random assignment*. Journal of the ACM (JACM) 33(4), pp. 724–767.

[3] Christel Baier & Marta Kwiatkowska (2000): *Domain equations for probabilistic processes*. Mathematical Structures in Computer Science 10(6), pp. 665–717.

[4] Stephen Brookes (1996): *The essence of parallel Algol*. In: *Proceedings 11th Annual IEEE Symposium on Logic in Computer Science*, IEEE, pp. 164–173.

[5] Martín Escardó (2004): *Synthetic topology: of data types and classical spaces*. Electronic Notes in Theoretical Computer Science 87, pp. 21–156.

[6] Martín Escardó (2009): *Semi-decidability of may, must and probabilistic testing in a higher-type setting*. Electronic Notes in Theoretical Computer Science 249, pp. 219–242.

[7] Yuan Feng, Runyao Duan & Mingsheng Ying (2011): *Bisimulation for quantum processes*. Acm Sigplan Notices 46(1), pp. 523–534.

[8] Yuan Feng & Yingte Xu (2023): *Verification of Nondeterministic Quantum Programs*. In: *Proceedings of the 28th ACM International Conference on Architectural Support for Programming Languages and Operating Systems, Volume 3*, pp. 789–805.

[9] Yuan Feng, Li Zhou & Yingte Xu (2023): *Refinement calculus of quantum programs with projective assertions*. arXiv preprint arXiv:2311.14215.

[10] Miriam Franchella (1997): *On the origins of Dénes König's infinity lemma*. Archive for history of exact sciences, pp. 3–27.

[11] Jianling Fu, Hui Jiang, Ming Xu, Yuxin Deng & Zhi-Bin Li (2024): *Algorithmic Analysis of Termination Problems for Nondeterministic Quantum Programs*. arXiv preprint arXiv:2402.15827.

[12] Gerhard Gierz, Karl Heinrich Hofmann, Klaus Keimel, Jimmie D Lawson, Michael Mislove & Dana S Scott (2003): *Continuous lattices and domains*. 93, Cambridge university press.

[13] Sergey Goncharov, Stefan Milius & Christoph Rauch (2016): *Complete Elgot monads and coalgebraic resumptions*. Electronic Notes in Theoretical Computer Science 325, pp. 147–168.

[14] Jean Goubault-Larrecq (2013): *Non-Hausdorff topology and domain theory: Selected topics in point-set topology*. 22, Cambridge University Press.

[15] Jean Goubault-Larrecq (2015): *Full abstraction for non-deterministic and probabilistic extensions of PCF I: The angelic cases*. Journal of Logical and Algebraic Methods in Programming 84(1), pp. 155–184.

[16] Jean Goubault-Larrecq (2019): *A Probabilistic and Non-Deterministic Call-by-Push-Value Language*. In: *34th Annual ACM/IEEE Symposium on Logic in Computer Science, LICS 2019, Vancouver, BC, Canada, June 24-27, 2019*, IEEE, pp. 1–13, doi:10.1109/LICS.2019.8785809. Available at `https://doi.org/10.1109/LICS.2019.8785809`.

[17] Jean Goubault-Larrecq (2020): *Probabilistic Powerdomains and Quasi-Continuous Domains*. arXiv preprint arXiv:2007.04189.

[18] Sergiu Hart, Micha Sharir & Amir Pnueli (1983): *Termination of probabilistic concurrent program*. ACM Transactions on Programming Languages and Systems (TOPLAS) 5(3), pp. 356–380.

[19] Matthew CB Hennessy & Gordon D Plotkin (1979): *Full abstraction for a simple parallel programming language*. In: *Mathematical Foundations of Computer Science 1979: Proceedings, 8th Symposium, Olomouc, Czechoslovakia, September 3–7, 1979 8*, Springer, pp. 108–120.

[20] Claire Jones & Gordon D Plotkin (1989): *A probabilistic powerdomain of evaluations*. In: *Proceedings. Fourth Annual Symposium on Logic in Computer Science*, IEEE Computer Society, pp. 186–187.

[21] Klaus Keimel & Gordon Plotkin (2009): *Predicate transformers for convex powerdomains*. Mathematical Structures in Computer Science 19(3), pp. 501–539.

[22] Klaus Keimel, A Rosenbusch & Thomas Streicher (2011): *Relating direct and predicate transformer partial correctness semantics for an imperative probabilistic-nondeterministic language*. Theoretical computer science 412(25), pp. 2701–2713.

[23] Ondřej Lengál, Anthony W Lin, Rupak Majumdar & Philipp Rümmer (2017): *Fair termination for parameterized probabilistic concurrent systems*. In: *Tools and Algorithms for the Construction and Analysis of Systems: 23rd International Conference, TACAS 2017, Held as Part of the European Joint Conferences on Theory and Practice of Software, ETAPS 2017, Uppsala, Sweden, April 22-29, 2017, Proceedings, Part I 23*, Springer, pp. 499–517.

[24] Rupak Majumdar & VR Sathiyanarayana (2024): *Sound and Complete Proof Rules for Probabilistic Termination*. *arXiv preprint arXiv:2404.19724*.

[25] Annabelle McIver & Carroll Morgan (2005): *Abstraction, refinement and proof for probabilistic systems*. Springer Science & Business Media.

[26] Annabelle McIver, Tahiry Rabehaja & Georg Struth (2013): *Probabilistic concurrent Kleene algebra*. *arXiv preprint arXiv:1306.2697*.

[27] Annabelle K McIver & Carroll Morgan (2001): *Partial correctness for probabilistic demonic programs*. *Theoretical Computer Science* 266(1-2), pp. 513–541.

[28] Robin Milner (1975): *Processes: a mathematical model of computing agents*. In: *Studies in Logic and the Foundations of Mathematics*, 80, Elsevier, pp. 157–173.

[29] Michael Mislove (2000): *Nondeterminism and probabilistic choice: Obeying the laws*. In: *International Conference on Concurrency Theory*, Springer, pp. 350–365.

[30] Michael W. Mislove, Joël Ouaknine & James Worrell (2003): *Axioms for Probability and Nondeterminism*. In Flavio Corradini & Uwe Nestmann, editors: *Proceedings of the 10th International Workshop on Expressiveness in Concurrency, EXPRESS 2003, Marseille, France, September 2, 2003*, *Electronic Notes in Theoretical Computer Science* 96, Elsevier, pp. 7–28, doi:10.1016/j.entcs.2004.04.019. Available at `https://doi.org/10.1016/j.entcs.2004.04.019`.

[31] Michael A. Nielsen & Isaac L. Chuang (2016): *Quantum Computation and Quantum Information (10th Anniversary edition)*. Cambridge University Press.

[32] Maciej Piróg & Jeremy Gibbons (2014): *The coinductive resumption monad*. *Electronic notes in theoretical computer science* 308, pp. 273–288.

[33] Gordon Plotkin & John Power (2001): *Adequacy for algebraic effects*. In: *International Conference on Foundations of Software Science and Computation Structures*, Springer, pp. 1–24.

[34] Gordon Plotkin & Matija Pretnar (2008): *A logic for algebraic effects*. In: *2008 23rd Annual IEEE symposium on logic in computer science*, IEEE, pp. 118–129.

[35] Gordon D. Plotkin & John Power (2003): *Algebraic Operations and Generic Effects*. *Appl. Categorical Struct.* 11(1), pp. 69–94, doi:10.1023/A:1023064908962. Available at `https://doi.org/10.1023/A:1023064908962`.

[36] John C Reynolds (1998): *Theories of programming languages*. Cambridge University Press.

[37] Michael B Smyth (1983): *Power domains and predicate transformers: A topological view*. In: *International Colloquium on Automata, Languages, and Programming*, Springer, pp. 662–675.

[38] Ian Stark (1996): *A fully abstract domain model for the/spl pi/-calculus*. In: *Proceedings 11th Annual IEEE Symposium on Logic in Computer Science*, IEEE, pp. 36–42.

[39] Regina Tix, Klaus Keimel & Gordon Plotkin (2009): *Semantic domains for combining probability and non-determinism*. *Electronic Notes in Theoretical Computer Science* 222, pp. 3–99.

[40] Tarmo Uustalu & Varmo Vene (1999): *Primitive (co) recursion and course-of-value (co) iteration, categorically*. *Informatica* 10(1), pp. 5–26.

[41] Daniele Varacca, Hagen Völzer & Glynn Winskel (2006): *Probabilistic event structures and domains*. *Theoretical Computer Science* 358(2-3), pp. 173–199.

[42] Daniele Varacca & Glynn Winskel (2006): *Distributing probability over non-determinism*. *Mathematical structures in computer science* 16(1), pp. 87–113.

[43] Steven Vickers (1989): *Topology via logic*. Cambridge University Press.

[44] Marc de Visme (2019): *Event structures for mixed choice*. In: *The 30th International Conference on Concurrency Theory, CONCUR 2019*.

[45] John Watrous (2018): *The theory of quantum information*. Cambridge university press.

[46] Glynn Winskel (1985): *On powerdomains and modality*. *Theoretical Computer Science* 36, pp. 127–137.

[47] Glynn Winskel (1993): *The formal semantics of programming languages: an introduction*. MIT press.

[48] Mingsheng Ying, Li Zhou & Yangjia Li (2018): *Reasoning about parallel quantum programs*. *arXiv preprint arXiv:1810.11334*.

[49] Mingsheng Ying, Li Zhou, Yangjia Li & Yuan Feng (2022): *A proof system for disjoint parallel quantum programs*. *Theoretical Computer Science* 897, pp. 164–184.

$$\begin{aligned}
[\![\texttt{skip}]\!] &= s \mapsto x(\{1 \cdot s\}) \\
[\![\texttt{a}]\!] &= s \mapsto x(\{[\![\texttt{a}]\!]\,(s)\}) \\
[\![P;Q]\!] &= [\![Q]\!]^{\star} \cdot [\![P]\!] \\
[\![P +_{\texttt{p}} Q]\!] &= \texttt{p} \cdot [\![P]\!] + (1-\texttt{p}) \cdot [\![Q]\!] \\
[\![P + Q]\!] &= [\![P]\!] \uplus [\![Q]\!] \\
[\![\texttt{if}\,\texttt{b}\,\texttt{then}\,P\,\texttt{else}\,Q]\!] &= \big[\,[\![Q]\!]\,, [\![P]\!]\,\big] \cdot \cong \cdot \langle \mathrm{id}, [\![\texttt{b}]\!] \rangle \\
[\![\texttt{while}\,\texttt{b}\,P]\!] &= \mathrm{lfp}\Big(f \mapsto \big[\eta, f^{\star} \cdot [\![P]\!]\,\big] \cdot \cong \cdot \langle \mathrm{id}, [\![\texttt{b}]\!] \rangle\Big)
\end{aligned}$$

Figure 4: Sequential fragment of the extensional semantics

A The sequential fragment

Recall that we denote the extensional semantics $\mathrm{ext} \cdot (\!|-|\!)$ introduced in the main text by $[\![-]\!]$. In this section we prove that $[\![-]\!]$ satisfies the equations listed in Figure 4 whenever the parallel composition operator is not involved. We start with the following lemma.

Lemma A.1. *The equation below holds for all programs $P \in \mathrm{Pr}$ and elements r of νR.*

$$\textstyle\bigvee_{n\in\mathbb{N}} (\mathrm{ext}(r))^{\star} \cdot \mathrm{ext}_n((\!|P|\!)_n) = \mathrm{ext}\,((\!|P|\!) \triangleright r)$$

Proof. We will show that the two respective inequations (*i.e.* $\leq_x$ and $\geq_x$) hold. We start with the case $\leq_x$ which we will prove by showing that for every natural number $n \in \mathbb{N}$ the inequation below holds.

$$(\mathrm{ext}(r))^{\star} \cdot \mathrm{ext}_n((\!|P|\!)_n) \leq_x \mathrm{ext}((\!|P|\!) \triangleright r)$$

We proceed by induction over the natural numbers. The base case is direct and for the inductive step we reason as follows,

$$\begin{aligned}
&(\mathrm{ext}(r))^{\star}(\mathrm{ext}_{n+1}((\!|P|\!)_{n+1})(s)) \\
&\big\{ \text{ Defn. of } \mathrm{ext}_{n+1} \text{ and Theorem 4.2 } \big\} \\
&= (\mathrm{ext}(r))^{\star}\left(\uplus\left\{\textstyle\sum_i p_i \cdot \mathrm{ext}_n((\!|P_i|\!)_n)(s_i) + \sum_j p_j \cdot \eta(s_j) \mid \ldots\right\}\right) \\
&\big\{ \text{ } (\mathrm{ext}(r))^{\star} \text{ is linear and } \uplus\text{-preserving } \big\} \\
&= \uplus\left\{\textstyle\sum_i p_i \cdot (\mathrm{ext}(r))^{\star}(\mathrm{ext}_n((\!|P_i|\!)_n)(s_i)) + \sum_j p_j \cdot \mathrm{ext}(r)(s_j) \mid \ldots\right\} \\
&\big\{ \text{ Induction hypothesis and monotonicity } \big\} \\
&\leq_x \uplus\left\{\textstyle\sum_i p_i \cdot \mathrm{ext}((\!|P_i|\!) \triangleright r)(s_i) + \sum_j p_j \cdot \mathrm{ext}(r)(s_j) \mid \ldots\right\} \\
&\big\{ \text{ Defn. of } (-)^{\star}, \triangleright, \text{ and Theorem 4.2 } \big\} \\
&= [\eta, \mathrm{app} \cdot (\mathrm{ext} \times \mathrm{id})]^{\star}\,((\!|P|\!) \triangleright r(s)) \\
&\big\{ \text{ Defn. of ext } \big\} \\
&= \mathrm{ext}((\!|P|\!) \triangleright r)(s)
\end{aligned}$$

where the ellipsis $(\ldots)$ hides the expression $\langle P,s\rangle \longrightarrow \sum_i p_i \cdot \langle P_i, s_i\rangle + \sum_j p_j \cdot s_j$. The case $\geq_x$ follows from showing that for every natural number $n \in \mathbb{N}$ the inequation,

$$(\mathrm{ext}(r))^\star \cdot \mathrm{ext}_n((\!|P|\!)_n) \geq_x \mathrm{ext}_n(((\!|P|\!) \triangleright r)_n)$$

holds. This is proved by induction over the natural numbers, and since it is completely analogous to the previous case we omit the details. □

Theorem A.1. *The equations in Figure 4 are sound.*

Proof. The first two cases are direct. The third case is obtained from Lemma A.1 and the fact that the post-composition of Scott-continuous maps is Scott-continuous. The fourth and fifth cases follow directly from ext being both linear and $\uplus$-preserving. The case involving conditionals is similar to the case involving while-loops which we detail next.

For while-loops we resort to the fact that both intensional and extensional semantics use Kleene's fixpoint theorem (recall Figure 3 and Figure 4). Specifically they involve mappings $\sigma^n : \nu R \to \nu R$ (recall that νR is treated as a set of functions) and $\tau^n : [S, \mathrm{PV}(S)] \to [S, \mathrm{PV}(S)]$ (for all $n \in \mathbb{N}$). These are defined as follows:

$$\begin{cases} \sigma^0(r) & = \bot \\ \sigma^{n+1}(r) & = [(\!|\mathtt{skip}|\!), (\!|\mathtt{skip}|\!) \triangleright ((\!|P|\!) \triangleright \sigma^n(r))] \cdot \cong \cdot \langle \mathrm{id}, [\![\mathtt{b}]\!]\rangle \end{cases}$$

$$\begin{cases} \tau^0(f) & = \bot \\ \tau^{n+1}(f) & = [\eta, (\tau^n(f))^\star \cdot \mathrm{ext}((\!|P|\!))] \cdot \cong \cdot \langle \mathrm{id}, [\![\mathtt{b}]\!]\rangle \end{cases}$$

It follows from straightforward induction over the natural numbers that the equation $\mathrm{ext}(\sigma^n(\bot)) = \tau^n(\bot)$ holds for all $n \in \mathbb{N}$. Then we reason:

$$\begin{aligned}
& \mathrm{ext}((\!|\mathtt{while}\,\mathtt{b}\,P|\!)) \\
= & \ \{ \text{ Defn. of } (\!|-|\!) \ \} \\
& \mathrm{ext}(\textstyle\bigvee_n \sigma^n(\bot)) \\
= & \ \{ \text{ Scott-continuity of ext } \} \\
& \textstyle\bigvee_n \mathrm{ext}(\sigma^n(\bot)) \\
= & \ \{ \ \mathrm{ext}(\sigma^n(\bot)) = \tau^n(\bot) \ \} \\
& \textstyle\bigvee_n \tau^n(\bot) \\
= & \ \{ \text{Kleene's lfp construction} \} \\
& \mathrm{lfp}\Big(f \mapsto [\eta, f^\star \cdot \mathrm{ext}((\!|P|\!))] \cdot \cong \cdot \langle \mathrm{id}, [\![\mathtt{b}]\!]\rangle\Big)
\end{aligned}$$

□

B Proof of Theorem 5.2

In this section we prove Theorem 5.2, which was presented in the main text. It requires a series of preliminaries which are detailed and proved next.

Theorem B.1. *Consider a family of zero-dimensional coherent domains $(X_i)_{i\in I}$ and also a finite set $F \subseteq \prod_{i\in I} \mathrm{V}_{=1}(X_i) \subseteq \prod_{i\in I} \mathrm{V}(X_i)$. Then $b(F) = \operatorname{conv} F$ and consequently the set $\operatorname{conv} F$ is Lawson-compact.*

Proof. Take $(x_i)_{i\in I} \in \overline{\operatorname{conv} F} \cap {\uparrow}\operatorname{conv} F$. We can assume the existence of an element $(a_i)_{i\in I} \in \operatorname{conv} F$ such that $(x_i)_{i\in I} \geq (a_i)_{i\in I}$. Note that every a_i $(i \in I)$ has total mass 1 by assumption. Since $(x_i)_{i\in I} \in \overline{\operatorname{conv} F}$ it must be the case that every x_i $(i \in I)$ has total mass ≤ 1 [14, Exercise 5.1.14], and therefore every x_i must have total mass 1 due to the inequation $(x_i)_{i\in I} \geq (a_i)_{i\in I}$. This means that both $(x_i)_{i\in I}$ and $(a_i)_{i\in I}$ live in $\prod_{i\in I} V_{=1}(X_i)$ and thus $(x_i)_{i\in I} = (a_i)_{i\in I}$ (by Theorem 2.1). It follows that $(x_i)_{i\in I} \in \operatorname{conv} F$ which proves the claim. □

The next step is to extend the previous theorem to finite sets of tuples of subvaluations, *i.e.* we extend the previous result to finite sets $F \subseteq \prod_{i\in I} \mathrm{V}_{\leq 1}(X_i)$ with every X_i a zero-dimensional coherent domain. We will need the following auxiliary result.

Lemma B.1. *For every domain X there exists an ep-pair between $\mathrm{V}(X+1)$ and $\mathrm{V}(X)$. Specifically the projection $p : \mathrm{V}(X+1) \twoheadrightarrow \mathrm{V}(X)$ is defined as $[\eta^{\mathrm{V}}, \perp]^\star$ whereas the embedding $e : \mathrm{V}(X) \hookrightarrow \mathrm{V}(X+1)$ is defined as $\mathrm{V}(\mathrm{inl})$.*

Proof. The notion of an ep-pair can be consulted for example in [12, Section IV-1] and also [14, Definition 9.6.1]. The result itself follows by straightforward calculations. □

Theorem B.2. *Consider a family of zero-dimensional coherent domains $(X_i)_{i\in I}$ and also a finite set $F \subseteq \prod_{i\in I} \mathrm{V}_{\leq 1}(X_i) \subseteq \prod_{i\in I} \mathrm{V}(X_i)$. Then the set $\operatorname{conv} F$ is Lawson-compact.*

Proof. It follows from Lemma B.1 and the fact that ep-pairs are preserved by products that there exists an ep-pair between $\prod_{i\in I} \mathrm{V}(X_i+1)$ and $\prod_{i\in I} \mathrm{V}(X_i)$. In particular by [12, Exercise O-3.29] we have a Lawson-continuous map $p : \prod_{i\in I} \mathrm{V}(X_i+1) \twoheadrightarrow \prod_{i\in I} \mathrm{V}(X_i)$ defined by $\prod_{i\in I} [\eta^{\mathrm{V}}, \perp]^\star$. Note as well that every $X_i + 1$ is zero-dimensional [14, Proposition 5.1.59] and a coherent domain.

Consider now a tuple $(\mu_i)_{i\in I} \in F \subseteq \prod_{i\in I} \mathrm{V}_{\leq 1}(X_i)$. For every such tuple we define a new one $(\mu_i')_{i\in I} \in \prod_{i\in I} V_{=1}(X_i+1)$ by setting,

$$\mu_i' = \mathrm{V}(\mathrm{inl})(\mu_i) + (1 - \mu_i(X_i)) \cdot \mathrm{inr}(*)$$

We obtain a new set $F' \subseteq \prod_{i\in I} V_{=1}(X_i+1)$ in this way and by Theorem B.1 its convex closure $\operatorname{conv} F'$ is Lawson-compact. Finally by taking advantage of the fact that p is linear we reason $p[\operatorname{conv} F'] = \operatorname{conv} p[F'] = \operatorname{conv} F$, and thus since p is Lawson-continuous the set $\operatorname{conv} F$ must be Lawson-compact. □

Together with [12, Proposition III-1.6] the previous theorem entails that under certain conditions the closure $\overline{\operatorname{conv} F}$ of the previous set $\operatorname{conv} F$ simplifies to ${\downarrow}\operatorname{conv} F$. This is formulated in the following corollary, and it will be useful in the proof of Theorem 5.2.

Corollary B.1. *Consider a family of zero-dimensional coherent domains $(X_i)_{i\in I}$ and also a finite set $F \subseteq \prod_{i\in I} \mathrm{V}_{\leq 1}(X_i) \subseteq \prod_{i\in I} \mathrm{V}(X_i)$. Then we have $\overline{\operatorname{conv} F} = {\downarrow}\operatorname{conv} F$ and in particular we obtain $b(F) = {\downarrow}\operatorname{conv} F \cap {\uparrow}\operatorname{conv} F$.*

The notion of an *n*-step trace. Next we introduce the notion of an n-step trace. Going back to the analogy to König's lemma (recall Section 5), an n-step trace corresponds to a path of length n in a tree. Such traces involve a space of choices (made by a scheduler along a computation) which we detail next. In order to not overburden notation we denote the set of inputs of a scheduler, *i.e.*

$$((\mathrm{Pr} \times S) \times \mathrm{V}_{=1}(S + \mathrm{Pr} \times S))^* \times (\mathrm{Pr} \times S)$$

by I. Then we define the space of choices Ch as the DCPO-product $\prod_{i\in I} \mathrm{V}\big(\mathrm{V}_{=1}(S + \mathrm{Pr} \times S)\big)$. The space $S + \mathrm{Pr} \times S$ is discrete and thus $\mathrm{V}_{=1}(S + \mathrm{Pr} \times S)$ is also discrete (Theorem 2.1), in particular it is a coherent domain. This implies that Ch is an LCone-product per our previous remarks about products in Section 2. All schedulers can be uniquely encoded as elements of Ch. They thus inherit an order which is precisely that of partial functions by Theorem 2.1. The inherited order on schedulers is furthermore a DCPO. Next, given a valuation $\nu \in \mathrm{V}_{=1}(S + \mathrm{Pr} \times S)$ we use the expression $h\langle P, s\rangle \mapsto 1 \cdot \nu$ to denote the element in Ch that associates the input $h\langle P, s\rangle$ to $1 \cdot \nu$ and all other inputs to the zero-mass valuation $\perp$. Given a linear combination $\nu \in \mathrm{V}(S + \mathrm{Pr} \times S)$ we use $\nu^S \in \mathrm{V}(S)$ to denote the respective restriction on S. We now formally introduce the notion of an n-step trace.

Definition B.1. *Take a pair $(h, \langle P, s\rangle)$ and a scheduler $\mathcal{S}$ such that $\mathcal{S}(h\langle P, s\rangle) = \sum_k p_k \cdot \nu_k$ for some convex combination $\sum_k p_k \cdot (-)$ with each ν_k of the form,*

$$\textstyle\sum_i p_{k,i} \cdot \langle P_{k,i}, s_{k,i}\rangle + \sum_j p_{k,j} \cdot s_{k,j}$$

Then we define an $\mathbb{N}_+$-indexed family of partial functions $t^{\mathcal{S},n} : I \to (\mathrm{V}(S) \times \mathrm{Ch})^n$, which send an input to the respective n-step trace, as follows:

$$\begin{aligned} t^{\mathcal{S},1}(h\langle P, s\rangle) &= \textstyle\sum_k p_k \cdot \big(\nu_k^S, h\langle P, s\rangle \mapsto 1 \cdot \nu_k\big) \\ t^{\mathcal{S},n+1}(h\langle P, s\rangle) &= \textstyle\sum_k p_k \cdot \Big((\nu_k^S, h\langle P, s\rangle \mapsto 1 \cdot \nu_k) :: \\ &\qquad \textstyle\sum_i p_{k,i} \cdot t^{\mathcal{S},n}(h\langle P, s\rangle \nu_k \langle P_{k,i}, s_{k,i}\rangle) + \Delta^n(\nu_k^S, \perp)\Big) \end{aligned}$$

The operation $(::)$ is the append function on lists and $\Delta^n : X \to X^n$ is the n-diagonal map which sends an input x to the n-tuple $(x, \ldots, x)$.

Despite looking complicated the previous definition can be easily related to the big-step operational semantics. We detail this in the following theorem.

Theorem B.3. *Take a pair $(h, \langle P, s\rangle)$, a scheduler $\mathcal{S}$, and a positive natural number $n \in \mathbb{N}_+$. If $t^{\mathcal{S},n}(h\langle P, s\rangle) = (\mu_1, c_1) \ldots (\mu_n, c_n)$ for some sequence $(\mu_1, c_1) \ldots (\mu_n, c_n)$ then for all $1 \leq i \leq n$ we have $h\langle P, s\rangle \Downarrow^{\mathcal{S},i} \mu_i$. Conversely if for all $1 \leq i \leq n$ we have $h\langle P, s\rangle \Downarrow^{\mathcal{S},i} \mu_i$ for some sequence of valuations $\mu_1 \ldots \mu_n$ then we must also have $t^{\mathcal{S},n}(h\langle P, s\rangle) = (\mu_1, c_1) \ldots (\mu_n, c_n)$ for some sequence of choices $c_1 \ldots c_n$.*

Proof. The first claim follows from straightforward induction over the positive natural numbers, an unravelling of the trace functions' definition (Definition B.1), and applications of the operational semantics' rules (Figure 2). The second claim follows from induction over the positive natural numbers, the first claim and an analysis of both the operational semantics' rules and the conditions under which a trace function is well-defined. □

The following result involving traces will also be useful.

Lemma B.2. *The two conditions below hold for every pair $(h, \langle P, s\rangle)$.*

- *For every pair of schedulers $\mathcal{S}_1$ and $\mathcal{S}_2$ such that $\mathcal{S}_1 \leq \mathcal{S}_2$ the implication below holds for every natural number $n \in \mathbb{N}_+$.*

$$t^{\mathcal{S}_1,n}(h\langle P, s\rangle) \text{ is well-defined} \Longrightarrow t^{\mathcal{S}_1,n}(h\langle P, s\rangle) = t^{\mathcal{S}_2,n}(h\langle P, s\rangle)$$

- *For every scheduler $\mathcal{S}$ and positive natural number $n \in \mathbb{N}_+$ the trace $t^{\mathcal{S},n}(h\langle P, s\rangle)$ is a prefix of $t^{\mathcal{S},n+1}(h\langle P, s\rangle)$ whenever the latter is well-defined.*

Proof. Both cases follow from simple induction over the positive natural numbers. □

Next, the following lines are devoted to showing that the set of *all* n-step traces of $h\langle P,s\rangle$ is Lawson-compact. Intuitively this is connected to the generalisation of the condition 'finitely-branching' in König's lemma to 'compactly-branching'. In order to prove that the set of all n-step traces $(n \in \mathbb{N}_+)$ of $h\langle P,s\rangle$ is Lawson-compact we recur to the biconvex powercone and associated machinery. In particular we start by considering the continuous map $\mathrm{step} : I \to \mathrm{P}_b(\mathrm{V}(S) \times \mathrm{Ch} \times \mathrm{V}(S+I))$ defined by,

$$h\langle P,s\rangle \mapsto b\left\{\left(\nu^S,\, h\langle P,s\rangle \mapsto 1\cdot\nu,\, \mathrm{V}\big(\mathrm{id} + (x \mapsto h\langle P,s\rangle\nu x)\big)(\nu)\right) \mid \nu \in \langle P,s\rangle \longrightarrow\right\}$$

Next we define a family of maps $(f_n)_{n\in\mathbb{N}_+} : I \to \mathrm{P}_b((\mathrm{V}(S)\times\mathrm{Ch})^n)$ in CohDom (*i.e.* the category of coherent domains and continuous maps) by induction on the size of $n \in \mathbb{N}_+$. Concretely for the inductive step we set,

$$\begin{aligned} f_{n+1} = I &\xrightarrow{\mathrm{step}} \mathrm{P}_b\left(\mathrm{V}(S)\times\mathrm{Ch}\times\mathrm{V}(S+I)\right) \\ &\xrightarrow{\mathrm{P}_b\left(\mathrm{id}\times[\mathrm{stop},f_n]^{\star^{\mathrm{V}}}\right)} \mathrm{P}_b\left(\mathrm{V}(S)\times\mathrm{Ch}\times\mathrm{P}_b((\mathrm{V}(S)\times\mathrm{Ch})^n)\right) \\ &\xrightarrow{\mathrm{str}^{\star^{\mathrm{P}_b}}} \mathrm{P}_b\left((\mathrm{V}(S)\times\mathrm{Ch})^{n+1}\right) \end{aligned}$$

and set $f_1 = \mathrm{P}_b(\pi_1)\cdot\mathrm{step}$ for the base step, with $\mathrm{stop} : S \to \mathrm{P}_b((\mathrm{V}(S)\times\mathrm{Ch})^n)$ as the composite $\eta^{\mathrm{P}_b}\cdot\Delta^n\cdot\langle\eta^{\mathrm{V}},\underline{\perp}\rangle$. Note that every space $f_n(h\langle P,s\rangle) \subseteq (\mathrm{V}(S)\times\mathrm{Ch})^n$ is Lawson-compact by construction. Our next step is to prove that $f_n(h\langle P,s\rangle)$ is actually the set of all n-step traces of $h\langle P,s\rangle$. First, the following equations are obtained from straightforward calculations,

$$f_1(h\langle P,s\rangle) = b\left\{(\nu^S, h\langle P,s\rangle \mapsto 1\cdot\nu) \mid \nu \in \langle P,s\rangle \longrightarrow\right\}$$

$$f_{n+1}(h\langle P,s\rangle) = b\left(\bigcup_{\nu\in\langle P,s\rangle\longrightarrow} \left\{\left(\nu^S, h\langle P,s\rangle \mapsto 1\cdot\nu\right)\right\} \times \left(\textstyle\sum_i p_i\cdot F_i + \Delta^n(\nu^S,\perp)\right)\right)$$

under the assumption that every F_i is a finite set that satisfies the equation $b(F_i) = f_n(h\langle P,s\rangle\nu\langle P_i,s_i\rangle)$ and that all valuations $\nu \in \langle P,s\rangle \longrightarrow$ are of the form $\sum_i p_i\cdot\langle P_i,s_i\rangle + \sum_j p_j\cdot s_j$. It then follows by induction over the positive natural numbers that every $f_n(h\langle P,s\rangle)$ is of the form $b(F)$ for a finite set F that is built inductively by resorting to the two previous equations. We will use the expression F_n to denote such a set w.r.t. $f_n(h\langle P,s\rangle)$ unless stated otherwise. Sometimes we will abbreviate F_n simply to F if n is clear from the context. Second by an appeal to Corollary B.1 we obtain $f_n(h\langle P,s\rangle) = b(F) = {\downarrow}\mathrm{conv}\,F \cap {\uparrow}\mathrm{conv}\,F$ which further simplifies our set description of $f_n(h\langle P,s\rangle)$. Then we concretely connect the spaces $f_n(h\langle P,s\rangle)$ to the trace functions (Definition B.1) via the two following lemmata.

Lemma B.3. *Consider a program P, a state s, an history h, and a positive natural number $n \in \mathbb{N}_+$. For every scheduler $\mathcal{S}$ we have $t^{\mathcal{S},n}(h\langle P,s\rangle) \in \mathrm{conv}\,F \subseteq f_n(h\langle P,s\rangle)$ whenever $t^{\mathcal{S},n}(h\langle P,s\rangle)$ is well-defined. Conversely for every $x \in \mathrm{conv}\,F$ there exists a scheduler $\mathcal{S}$ such that $t^{\mathcal{S},n}(h\langle P,s\rangle) = x$.*

Proof. By an appeal to [39, Lemma 2.8] one deduces that,

$$\begin{aligned} \mathrm{conv}\,F &= \mathrm{conv}\left(\bigcup_{\nu\in\langle P,s\rangle\longrightarrow} \left\{\left(\nu^S, h\langle P,s\rangle \mapsto 1\cdot\nu\right)\right\} \times \left(\textstyle\sum_i p_i\cdot F_i + \Delta^n(\nu^S,\perp)\right)\right) \\ &= \mathrm{conv}\left(\bigcup_{\nu\in\langle P,s\rangle\longrightarrow} \left\{\left(\nu^S, h\langle P,s\rangle \mapsto 1\cdot\nu\right)\right\} \times \left(\textstyle\sum_i p_i\cdot \mathrm{conv}\,F_i + \Delta^n(\nu^S,\perp)\right)\right) \end{aligned}$$

for all $n+1$ with $n \in \mathbb{N}_+$. Both implications in the lemma's formulation rely on this observation. In particular the first implication then follows straightforwardly by induction over the positive natural numbers and by the trace functions' definition (Definition B.1). As for the second implication, we build a scheduler $\mathcal{S}$ by induction over the positive natural numbers in the following manner. For the base case suppose that we have $(\mu, c) \in \operatorname{conv} F \subseteq f_1(h\langle P, s\rangle)$. We can safely assume that it is of the form,

$$(\mu, c) = \sum_k p_k \cdot (\nu_k^S, h\langle P, s\rangle \mapsto 1 \cdot \nu_k)$$

for some convex combination $\sum_k p_k \cdot (-)$. We thus set $\mathcal{S}$ to be $\sum_k p_k \cdot (h\langle P, s\rangle \mapsto 1 \cdot \nu_k)$ and it is clear that $t^{\mathcal{S},1}(h\langle P, s\rangle) = (\mu, c)$ by the trace functions' definition. Regarding the inductive step we take a sequence $x \in \operatorname{conv} F \subseteq f_{n+1}(h\langle P, s\rangle)$. It is then the case that,

$$x = \sum_k p_k \cdot \big((\nu_k^S, h\langle P, s\rangle \mapsto 1 \cdot \nu_k) :: \sum_i p_{k,i} \cdot x_{k,i} + \Delta^n(\nu_k^S, \bot)\big) \tag{8}$$

for some convex combination $\sum_k p_k \cdot (-)$ where $x_{k,i} \in \operatorname{conv} F_{k,i} \subseteq f_n(h\langle P, s\rangle \nu_k \langle P_{k,i}, s_{k,i}\rangle)$ for all indices k, i. As in the proof of Proposition 4.2, we can assume that all valuations ν_k involved are pairwise distinct, by virtue of all sets $\operatorname{conv} F_{k,i}$ being convex and by recurring to the normalisation of subconvex combinations. Next, it follows from the induction hypothesis that for all traces $x_{k,i}$ there exists a scheduler $\mathcal{S}_{k,i}$ such that,

$$t^{\mathcal{S}_{k,i},n}(h\langle P, s\rangle \nu_k \langle P_{k,i}, s_{k,i}\rangle) = x_{k,i}$$

So we set $\mathcal{S} := \sum_k p_k \cdot (h\langle P, s\rangle \mapsto 1 \cdot \nu_k) + \sum_{k,i} \mathcal{S}_{k,i}$ and it is straightforward to show that the equation $t^{\mathcal{S},n+1}(h\langle P, s\rangle) = x$ holds by an appeal to Lemma B.2 and the fact that addition is monotone. □

The proof of the previous lemma provides a recipe for building a scheduler from any element of $\operatorname{conv} F \subseteq f_n(h\langle P, s\rangle)$. The following lemma discloses useful properties about this construction.

Lemma B.4. *The scheduler construction described in the previous proof is functional,* i.e. *exactly one scheduler arises from it. Furthermore the construction is monotone w.r.t. the prefix order of lists.*

Proof. Observe that the previous scheduler construction only relies on the choices $c_1 \ldots c_n$ in a given sequence $(\mu_1, c_1) \ldots (\mu_n, c_n)$. Note as well that in order to not overburden notation, we use $c_1 \ldots c_n$ to denote a sequence obtained from a sequence in $\operatorname{conv} F \subseteq f_n(h\langle P, s\rangle)$ by an application of the map $\pi_2{}^n$.

Let us start with the first claim. We just need to prove that the form (8) that we assume from a sequence of choices is unique. We proceed by case distinction. Suppose that the sequence involved is c_1. We will show that there is only one expression of the form $\sum_k p_k \cdot (h\langle P, s\rangle \mapsto 1 \cdot \nu_k)$ that can be equal to c_1. Thus consider another expression $\sum_j p'_j \cdot (h\langle P, s\rangle \mapsto 1 \cdot \nu_j)$, *i.e.* we change the convex combination involved (which is the only allowed change). Both convex combinations can be *uniquely* determined by inquiring c_1 so indeed the expression is unique. Consider now the case $c_1 \ldots c_{n+1}$. As before suppose that we have,

$$c_1 \ldots c_{n+1} = \sum_k p_k \cdot \Big((h\langle P, s\rangle \mapsto 1 \cdot \nu_k) :: \sum_i p_{k,i} \cdot (d_{k,i}^1 \ldots d_{k,i}^n)\Big)$$
$$c_1 \ldots c_{n+1} = \sum_j p'_j \cdot \Big((h\langle P, s\rangle \mapsto 1 \cdot \nu_j) :: \sum_i p'_{j,i} \cdot (d'^1_{j,i} \ldots d'^n_{j,i})\Big)$$

Both convex combinations $\sum_k p_k \cdot (-)$ and $\sum_j p_j \cdot (-)$ are equal for the same reason as before and thus the only elements in the expressions that could be different are $d_{k,i}^l$ and $d'^l_{k,i}$ for some k, i and some $1 \leq l \leq n$. We will show that even these elements are always equal which proves our first claim. First by the definition of $f_{n+1}(h\langle P,s\rangle)$ it is clear that for all k,i the support of $d^l_{k,i}$ and $d'^l_{k,i}$ $(1 \leq l \leq n)$ can only contain inputs with $h\langle P,s\rangle \nu_k\langle P_{k,i},s_{k,i}\rangle$ as prefix. Then for every such input a we have,

$$\textstyle\sum_k p_k \cdot \sum_i p_{k,i} \cdot d^l_{k,i}(a) = c_{l+1}(a) = \sum_i p_k \cdot \sum_i p_{k,i} \cdot d'^l_{k,i}(a)$$

which entails $p_k \cdot p_{k,i} \cdot d^l_{k,i}(a) = p_k \cdot p_{k,i} \cdot d'^l_{k,i}(a)$ for every k,i and therefore $d^l_{k,i}(a) = d'^l_{k,i}(a)$.

We now focus on the second claim, specifically we will prove that for all positive natural numbers $n \in \mathbb{N}_+$ and sequences,

$$x^1 \dots x^{n+1} \in \operatorname{conv} F_{n+1} \subseteq f_{n+1}(h\langle P,s\rangle)$$

if $\mathcal{S}$ and $\mathcal{S}'$ are schedulers obtained respectively from $x^1 \dots x^n \in \operatorname{conv} F_n \subseteq f_n(h\langle P,s\rangle)$ and $x^1 \dots x^{n+1}$, via the previous scheduler construction, then the inequation $\mathcal{S} \leq \mathcal{S}'$ holds. The proof follows by induction over the positive natural numbers and by exploiting the fact that addition is monotone. The base case follows straightforwardly by unravelling the definition of $\operatorname{conv} F_2$. For the inductive step it is clear that,

$$c_1 \dots c_{n+2} = \textstyle\sum_k p_k \cdot \Big((h\langle P,s\rangle \mapsto 1 \cdot \nu_k) :: \sum_i p_{k,i} \cdot (d^1_{k,i}, \dots, d^{n+1}_{k,i}) \Big)$$

for some convex combination $\sum_k p_k \cdot (-)$ and for sequences of choices $(d^1_{k,i}, \dots, d^{n+1}_{k,i})$ obtained from sequences in $\operatorname{conv}(F_{n+1})_{k,i} \subseteq f_{n+1}(h\langle P,s\rangle \nu_k\langle P_{k,i},s_{k,i}\rangle)$ by an application of the map ${\pi_2}^{n+1}$. This entails the equation,

$$c_1 \dots c_{n+1} = \textstyle\sum_k p_k \cdot \Big((h\langle P,s\rangle \mapsto 1 \cdot \nu_k) :: \sum_i p_{k,i} \cdot (d^1_{k,i}, \dots, d^n_{k,i}) \Big)$$

Now, in order to apply the induction hypothesis we need to show that for all indices k,i the sequence $(d^1_{k,i}, \dots, d^n_{k,i})$ can be obtained from a trace in $\operatorname{conv}(F_n)_{k,i} \subseteq f_n(h\langle P,s\rangle \nu_k\langle P_{k,i},s_{k,i}\rangle)$ by an application of ${\pi_2}^n$. This follows directly from Lemma B.3 and Lemma B.2. We thus derive $\mathcal{S}'_{k,i} \leq \mathcal{S}_{k,i}$ for all indices k,i by the induction hypothesis, and finally we obtain the sequence of inequations,

$$\begin{aligned} \mathcal{S} &= \textstyle\sum_k p_k \cdot (h\langle P,s\rangle \mapsto 1 \cdot \nu_k) + \sum_{k,i} \mathcal{S}_{k,i} \\ &\leq \textstyle\sum_k p_k \cdot (h\langle P,s\rangle \mapsto 1 \cdot \nu_k) + \sum_{k,i} \mathcal{S}'_{k,i} \\ &= \mathcal{S}' \end{aligned}$$

□

Proposition B.1. *Consider a program P, a state s, and an history h. Consider also the set* $\operatorname{conv} F \subseteq f_n(h\langle P,s\rangle)$ *for some number* $n \in \mathbb{N}_+$. *Then the order on* $\operatorname{conv} F$ *is discrete and therefore* $\downarrow\operatorname{conv} F \cap \uparrow\operatorname{conv} F = \operatorname{conv} F$ *by the anti-symmetry property of partial orders.*

Proof. Take two elements $(\mu_1,c_1)\dots(\mu_n,c_n)$ and $(\mu'_1,c'_1)\dots(\mu'_n,c'_n)$ in $\operatorname{conv} F$. We will start by showing that the following implication holds.

$$c_1 \dots c_n \geq c'_1 \dots c'_n \Longrightarrow c_1 \dots c_n = c'_1 \dots c'_n$$

The latter follows by induction over the positive natural numbers $n \in \mathbb{N}_+$. For the base case both c_1 and c'_1 have mass 1 at $h\langle P,s\rangle$ and mass 0 everywhere else. Thus the assumption $c_1 \geq c'_1$ entails $c_1 = c'_1$ by Theorem 2.1. For the inductive step we use the definition of $f_{n+1}(h\langle P,s\rangle)$ to safely assume that,

$$\begin{aligned} c_1 \dots c_{n+1} &= \textstyle\sum_k p_k \cdot \Big((h\langle P,s\rangle \mapsto 1 \cdot \nu_k) :: \sum_i p_{k,i} \cdot (d^1_{k,i} \dots d^n_{k,i}) \Big) \\ c'_1 \dots c'_{n+1} &= \textstyle\sum_k p_k \cdot \Big((h\langle P,s\rangle \mapsto 1 \cdot \nu_k) :: \sum_i p_{k,i} \cdot (d'^1_{k,i} \dots d'^n_{k,i}) \Big) \end{aligned}$$

such that for all indices k,i the respective sequences of choices $d^1_{k,i} \dots d^n_{k,i}$ and $d'^1_{k,i} \dots d'^n_{k,i}$ are obtained from $\operatorname{conv} F_{k,i} \subseteq f_n(h\langle P,s\rangle \nu_k \langle P_{k,i}, s_{k,i}\rangle)$ by an application of the projection map $\pi_2{}^n$. Note that we can safely assume the same convex combination $\sum_k p_k \cdot (-)$ for both cases, because such combinations are uniquely encoded in both c_1 and c'_1 and we already know that $c_1 = c'_1$. Recall as well that we can safely assume that all the valuations ν_k involved in the combination are pairwise distinct, by virtue of the sets $\operatorname{conv} F_{k,i}$ being convex and by normalisation of subconvex valuations.

The next step is to prove that $d^l_{k,i} \geq d'^l_{k,i}$ (for all $1 \leq l \leq n$) to which we then apply the induction hypothesis to obtain $c_1 \dots c_{n+1} = c'_1 \dots c'_{n+1}$. First by the definition of $f_n(h\langle P,s\rangle \nu_k \langle P_{k,i}, s_{k,i}\rangle)$ it is clear that for all k,i the support of $d^l_{k,i}$ and $d'^l_{k,i}$ $(1 \leq l \leq n)$ can only contain inputs with $h\langle P,s\rangle \nu_k \langle P_{k,i}, s_{k,i}\rangle$ as prefix. Second for every such input a we have,

$$\textstyle\sum_k p_k \cdot \sum_i p_{k,i} \cdot d^l_{k,i}(a) = c_{l+1}(a) \geq c'_{l+1}(a) = \sum_k p_k \cdot \sum_i p_{k,i} \cdot d'^l_{k,i}(a)$$

which entails $p_k \cdot p_{k,i} \cdot d^l_{k,i}(a) \geq p_k \cdot p_{k,i} \cdot d'^l_{k,i}(a)$ for every k,i and thus $d^l_{k,i}(a) \geq d'^l_{k,i}(a)$.

Consider again two elements $(\mu_1, c_1) \dots (\mu_n, c_n) \geq (\mu'_1, c'_1) \dots (\mu'_n, c'_n)$ of $\operatorname{conv} F$. From the previous reasoning we know that for all $1 \leq l \leq n$ we have $c_l = c'_l$, which entails that both lists are actually the same since each μ_l (resp. μ'_l) only depends on the choices present in the respective list. This last claim follows from Lemma B.3 and Lemma B.4. □

We finally reach the goal of this section.

Proof of Theorem 5.2. Our proof involves the use of codirected limits in the category CompHaus of compact Hausdorff spaces and continuous maps. For every $n \in \mathbb{N}_+$ we set $X_n := f_n(\langle P,s\rangle)$, a compact Hausdorff space w.r.t. the Lawson topology of the n-fold product $(\mathrm{V}(S) \times \mathrm{Ch})^n$ in CohDom where $\mathrm{V}(S) \times \mathrm{Ch}$ is also a CohDom-product. Given a DCPO X let $L(X)$ denote the carrier of X equipped with respective Lawson topology. It then follows from [14, Proposition 5.1.56, Exercise 9.1.36, and Proposition 9.3.1] that the previous domain $(\mathrm{V}(S) \times \mathrm{Ch})^n$ equipped with the respective Lawson topology is equal to $(L(\mathrm{V}(S)) \times L(\mathrm{Ch}))^n$ where in the latter case the constructs $(\times)$ and $(-)^n$ are products in CompHaus. Let us now take the codirected limit ($\lim \mathscr{D}$) of,

$$\begin{tikzcd} X_1 & X_2 \arrow[l, "\mathrm{lst}_1"'] & X_3 \arrow[l, "\mathrm{lst}_2"'] & \dots \end{tikzcd}$$

in the category CompHaus. Concretely each function $\mathrm{lst}_n : X_{n+1} \to X_n$ forgets the last pair of a given sequence (of pairs) in X_{n+1}. It is both well-defined and continuous because it is (up-to isomorphism) a restriction of a projection on the topological product $(L(\mathrm{V}(S)) \times L(\mathrm{Ch}))^n$ and both Lemma B.2 and Lemma B.3 hold. The space $\lim \mathscr{D}$ is thus compact Hausdorff by construction. Let us equip it with a concrete and more amenable characterisation. First by general topological results $\lim \mathscr{D}$ is concretely defined as the subspace,

$$\lim \mathscr{D} \cong \left\{ (t_i)_{i \in \mathbb{N}_+} \in \textstyle\prod_{i \in \mathbb{N}_+} X_i \mid \mathrm{lst}_i(t_{i+1}) = t_i \right\}$$

Second there exists a map in CompHaus that for a given family $(t_i)_{i \in \mathbb{N}_+} \in \prod_{i \in \mathbb{N}_+} X_i$ it only keeps the last element of each t_i $(i \in \mathbb{N}_+)$. Formally it is defined by,

$$\langle f_i \rangle_{i \in \mathbb{N}_+} : \textstyle\prod_{i \in \mathbb{N}_+} X_i \to (L(\mathrm{V}(S)) \times L(\mathrm{Ch}))^\omega \qquad\qquad f_i = \begin{cases} \pi_1 & \text{if } i = 1 \\ \pi_i \cdot \pi_i & \text{otherwise} \end{cases}$$

It is straightforward to prove by induction over the positive natural numbers that its restriction to $\lim \mathscr{D}$ is injective and therefore a monomorphim in CompHaus. This restriction then yields an epimorphism

$\lim \mathscr{D} \to \mathrm{im}\langle f_i \rangle_{i\in\mathbb{N}_+}$ [1, Examples 14.23] in CompHaus which must also be a monomorphism by general categorical results. Finally since CompHaus is balanced [1, Examples 7.50] we obtain an isomorphism $\lim \mathscr{D} \cong \mathrm{im}\langle f_i \rangle_{i\in\mathbb{N}_+}$. One can thus equivalently treat $\lim \mathscr{D}$ as a compact Hausdorff subspace of $(L(\mathrm{V}(S)) \times L(\mathrm{Ch}))^\omega$ with each element of $\lim \mathscr{D}$ a sequence in $(L(\mathrm{V}(S)) \times L(\mathrm{Ch}))^\omega$ where every prefix of size n is an element of X_n. We now wish to show that,

$$\lim \mathscr{D} \subseteq \bigcup_{i\in\mathbb{N}_+} (\mathrm{V}(S)\times \mathrm{Ch})^{i-1} \times (\varphi \times \mathrm{Ch}) \times (\mathrm{V}(S)\times\mathrm{Ch})^\omega \tag{9}$$

where φ is treated as a Scott-open (note that the union is then an open cover). We proceed by taking a sequence $(\mu_1, c_1)(\mu_2, c_2)\cdots \in \lim \mathscr{D}$. By Lemma B.3, Lemma B.4 and Proposition B.1 for every prefix $(\mu_1, c_1)\ldots(\mu_n, c_n) \in X_n$ we obtain a scheduler $\mathcal{S}_n$ that yields the same trace. More generally we obtain a chain of schedulers $\mathcal{S}_1 \leq \mathcal{S}_2 \leq \ldots$ in this way and denote the corresponding supremum by $\mathcal{S}$. Note that $\mathcal{S}$ is non-blocking w.r.t. $\langle P, s\rangle$ by virtue of Theorem B.3 and Lemma B.2. Then by the assumption on non-blocking schedulers and again Theorem B.3 and Lemma B.2 there must exist a natural number $k \in \mathbb{N}_+$ such that $\mu_k \in \varphi$. This shows that the inclusion (9) indeed holds. Now, since $\lim \mathscr{D}$ is compact we obtain,

$$\lim \mathscr{D} \subseteq \bigcup_{i\in F} (\mathrm{V}(S)\times \mathrm{Ch})^{i-1} \times (\varphi \times \mathrm{Ch}) \times (\mathrm{V}(S)\times\mathrm{Ch})^\omega$$

for $F \subseteq \mathbb{N}_+$ a finite set and we define $\mathbf{z} := \max F$. Finally it follows from Theorem B.3, Lemma B.2, and Lemma B.3 that for every non-blocking scheduler $\mathcal{S}$ the corresponding infinite sequence of traces must be in $\lim \mathscr{D}$ and therefore we obtain $\langle P, s\rangle \Downarrow^{\mathcal{S},i} \mu_i$ with $\mu_i \in \varphi$ for some $i \in F$. It is then clear that $\langle P, s\rangle \Downarrow^{\mathcal{S},\mathbf{z}} \mu_{\mathbf{z}}$ and $\mu_{\mathbf{z}} \in \varphi$ by Proposition 3.2 and the fact that φ is a Scott-open. $\square$