

Concurrent valuation algebras

PhD thesis review seminar

Naso Evangelou-Oost

Advisors: Ian J. Hayes, Larissa Meinicke

14 July 2023



- Concurrent & distributed systems are everywhere, yet really hard to reason about.
- Compositional methods mitigate the combinatorial explosion of possibilities when many interacting systems combine.
- Concurrent valuation algebras (CVAs) are . . .
 - compositional, modular, local,
 - algebraic, lattice-based (refinement algebra),
 - able to detect subtle information inconsistency (e.g. sequential inconsistency),
 - algorithmically tractable.

Background

- Refinement algebras, trace semantics, topological spaces, category theory, simplicial sets. . .

Contextuality in distributed systems

- RAMiCS23 submission (published) (Evangelou-Oost, Bannister and Hayes 2023).
- Specifications of distributed systems via valuation algebras.
- Lattices of distributed specifications.
- Models program behaviours as *relative* execution traces (independent of a global clock).
- Manifests sequential consistency as contextuality.

Trace models of concurrent valuation algebras

- ICFEM23 submission (under review) (Evangelou-Oost, Bannister, Meinicke et al. 2023).
- Reformulates the classical definition of ordered valuation algebras (OVAs); introduces Concurrent Valuation Algebras (CVAs).
- Presents 3 trace models representing different paradigms of concurrent/distributed computing, and their interrelations via morphisms.
- Highlights potential for applying local computation framework to model checking.

Computer formalisation

- Computer formalisation of the CVA paper in the proof assistant Isabelle/HOL.

- 1 Background
 - Refinement algebras
 - Trace models of refinement algebras
 - Project goal: localising refinement algebras
 - Topologies and presheaves

- 2 Contextuality in distributed systems

- 3 Trace models of concurrent valuation algebras
 - (covariant) Grothendieck construction
 - Ordered valuation algebras (OVAs)
 - Concurrent valuation algebras (CVAs), refinement & reasoning
 - Local computation

- 4 Computer formalisation

- 5 Summary & plan for the rest of the project

- A **refinement algebra** is a partially-ordered set (Φ, \preceq) of specifications/commands with a commutative monoidal (i.e. unital, associative, order-preserving) binary operation

$$\parallel : \Phi \times \Phi \rightarrow \Phi$$

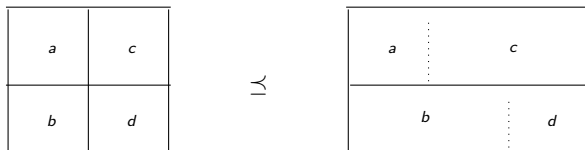
and a monoidal binary operation

$$\circledast : \Phi \times \Phi \rightarrow \Phi,$$

satisfying a **weak exchange law**. for all $a, b, c, d \in \Phi$,

$$(a \parallel b) \circledast (c \parallel d) \preceq (a \circledast c) \parallel (b \circledast d)$$

- $a \preceq b$ means a **refines** b ; we think of b as the **abstract specification** and a as the **concrete implementation**.
- The weak exchange law abstractly characterises parallel and sequential composition w.r.t. refinement:



- Examples of refinement algebras: Concurrent Kleene Algebra (CKA) (Hoare et al. 2009), Concurrent Refinement Algebra (CRA) (Hayes 2016).

Trace models.

- Specifications $a \in \Phi$ are subsets of *traces* = lists of 'observations' $t = [t_1, \dots, t_n]$.
- Can be coupled to a clock or not. What does the i in t_i mean?
- Decoupling from a global clock = stuttering invariance (traces of states);

$$[a, a, b, c, c, c] \sim [a, b, b, c]$$

Two ways to encode stuttering-invariance:

- Stuttering closure (subsets): if the first trace belongs to a specification, so does the second.
- Stuttering reduction (quotients): both traces are identified in a quotient.

Parallel & sequential products in trace models depend on what the elements t_i in a trace $t = [t_1, \dots, t_n]$ mean:

① **States:** t_i is a state of the system.

- Parallel = sync = \cap ,

$$\{[a, b, c]\} \cap \{[a, b, c]\} = \{[a, b, c]\}$$

- Sequential = gluing concatenation = \smile .

$$\{[a, b, c]\} \smile \{[c, d]\} = \{[a, b, c, d]\}$$

② **Actions:** t_i is an action taken by the system.

- Parallel = shuffle/interleave = \sqcup ,

$$\{[a, b, c]\} \sqcup \{[d, e]\} = \{[a, b, d, e, c], [d, a, b, c, e], [a, b, d, c, e], \dots\}$$

- Sequential = concatenation = \frown ,

$$\{[a, b, c]\} \frown \{[d, e]\} = \{[a, b, c, d, e]\}$$

Project goal: localising refinement algebras

Problem area:

- Using a refinement algebra Φ to model a distributed system means commands $a \in \Phi$ must specify the behaviour over the *entire state space of the system*.
- If the system is very large, this may be impractical. Moreover, in a modular design process, the overall state space may be unknown.
- For both verification and model checking, decomposing the system into smaller components is imperative for scalability.

Our project's overall goal:

- To identify a useful, and tractable concept of a *localised refinement algebra* where specifications need only refer to the relevant subset of the state space, that is suitable for modular, compositional concurrent/distributed systems specifications, and is also computationally tractable.

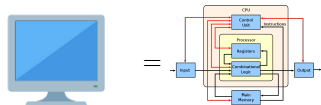
Topologies and presheaves

Topologies encode the connective structure of distributed and concurrent systems:

- Distributed systems: networks.



- Concurrent systems: shared memory and resources.



A **prealgebra** is a poset-valued contravariant functor on a topology \mathcal{T} ,

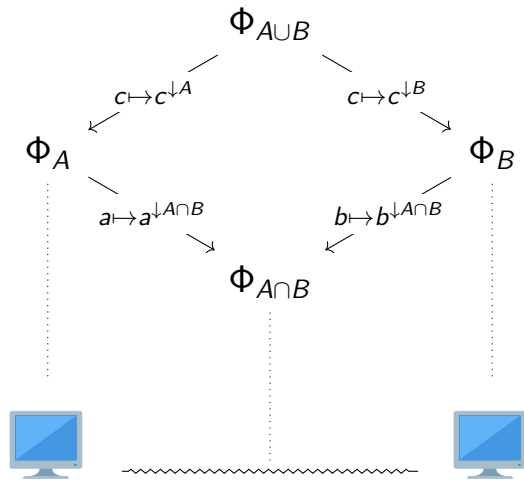
$$\Phi : \mathcal{T}^{op} \rightarrow \mathcal{P}_{\mathcal{O}\mathcal{A}}$$

This consists of the following data:

- **Object map.** A function sending each domain $A \in \mathcal{T}$ to a poset Φ_A (e.g. of traces, or specifications).
- **Arrow map.** A function sending each inclusion $B \subseteq A$ to a monotone **restriction map**

$$a \mapsto a^{\downarrow B} : \Phi_A \rightarrow \Phi_B$$

- **Transitivity.** For $C \subseteq B \subseteq A$ and $a \in \Phi_A$, $(a^{\downarrow B})^{\downarrow C} = a^{\downarrow C}$.
- **Identity law.** For each $A \in \mathcal{T}$ and $a \in \Phi_A$, $a^{\downarrow A} = a$.



- 1 Background
 - Refinement algebras
 - Trace models of refinement algebras
 - Project goal: localising refinement algebras
 - Topologies and presheaves
- 2 Contextuality in distributed systems
- 3 Trace models of concurrent valuation algebras
 - (covariant) Grothendieck construction
 - Ordered valuation algebras (OVAs)
 - Concurrent valuation algebras (CVAs), refinement & reasoning
 - Local computation
- 4 Computer formalisation
- 5 Summary & plan for the rest of the project

Summary of RAMiCS23 paper (Evangelou-Oost, Bannister and Hayes 2023).

- Shows how an established concept: **ordered valuation algebras (OVAs)** can be used to model distributed systems.
- An OVA is a prealgebra $\Phi : \mathcal{T}^{op} \rightarrow \mathcal{Pos}$ with an associative binary operation

$$\otimes : \prod_{A \in \mathcal{T}} \Phi_A \times \prod_{A \in \mathcal{T}} \Phi_A \rightarrow \prod_{A \in \mathcal{T}} \Phi_A$$

obeying certain axioms.

- Interprets \otimes as synchronisation of valuations defined as subsets of **relative traces**.
- Defines **(distributed) specifications** as subsets of valuations defined over a **cover** (i.e. maximal antichain) of \mathcal{T} , and shows these form a complete lattice.
- **Contextuality** is an abstract form of information inconsistency where a collection of information **locally agrees but globally disagrees**, that is realisable in a valuation algebra. We show that in this context, contextuality corresponds to **sequential inconsistency**.
- Gives an example of contextuality in the classic **dining philosophers** scenario.

- 1 Background
 - Refinement algebras
 - Trace models of refinement algebras
 - Project goal: localising refinement algebras
 - Topologies and presheaves
- 2 Contextuality in distributed systems
- 3 Trace models of concurrent valuation algebras
 - (covariant) Grothendieck construction
 - Ordered valuation algebras (OVAs)
 - Concurrent valuation algebras (CVAs), refinement & reasoning
 - Local computation
- 4 Computer formalisation
- 5 Summary & plan for the rest of the project

Summary of ICFEM23 submission (Evangelou-Oost, Bannister, Meinicke et al. 2023).

- Reformulates an OVA in terms of categorical notions, relaxing the commutativity condition, imposing a stability/naturality condition for neutral elements, and imposing a global (Grothendieck) ordering.
- Introduces a *concurrent valuation algebra (CVA)* comprising two OVA structures, a weak exchange law, and relations between neutral elements, as a candidate for a *localised refinement algebra*.
- Presents three trace models of CVAs:
 - 1 Event trace model with shuffle as parallel and concatenation as parallel.
 - 2 State trace model with synchronisation as parallel and gluing concatenation as sequential.
 - 3 Relative state trace model where traces are *stuttering-reduced*.
- Defines morphisms of CVAs, and shows a surjective morphism from the state model to the relative state model, effectively exhibiting the relative trace model as a quotient of the state model.
- Suggests how to import an efficient *local computation* framework for CVAs.

(covariant) Grothendieck construction

Given $\Phi : \mathcal{T}^{op} \rightarrow \mathcal{P}os$, combine all the posets Φ_A into one big poset $\int \Phi$.

$$\int \Phi := \{(A, a) \mid A \in \mathcal{T}, a \in \Phi_A\}$$

Partial order \preceq :

$$(A, a) \preceq (B, b) \iff B \subseteq A \text{ and } a^{\downarrow B} \leq_{\Phi_B} b$$

Shorthand: write a instead of (A, a) leaving domain implicit; use **domain** map d to recover domain

$$d : \int \Phi \rightarrow \mathcal{T}^{op}$$

$$da = \text{domain of } a$$

Example.

If $A = \{x, y\}$, $B = \{x\}$, and

$$a = \{[(x_0, y_0), (x_1, y_1)], [(x_0, y_2), (x_1, y_3)]\}, \quad b = \{[x_0, x_1], [x_2, x_3]\}$$

then,

$$B \subseteq A \text{ and } a^{\downarrow B} = \{[x_0, x_1]\} \subseteq b$$

so, $(A, a) \preceq (B, b)$.

Ordered valuation algebras (OVAs)

Definition. An **ordered valuation algebra (OVA)** consists of...

- a prealgebra

$$\Phi : \mathcal{T}^{op} \rightarrow \mathcal{P}_{\mathcal{O}\mathcal{S}}$$

- a binary operator

$$\otimes : \int \Phi \times \int \Phi \rightarrow \int \Phi$$

- a natural transformation

$$\epsilon : \mathbf{1} \Rightarrow \Phi$$

(i.e. an element $\epsilon_A \in \Phi_A$ for each A such that $\epsilon_A^{\downarrow B} = \epsilon_B$ whenever $B \subseteq A$.)

Obeying the following axioms...

- **Ordered semigroup.**

\otimes is associative and monotone

- **Labelling.**

$$d(a \otimes b) = da \cup db$$

- **Neutrality.**

$$\epsilon_{da} \otimes a = a = a \otimes \epsilon_{da}$$

- **Combination.**

$$(a \otimes b)^{\downarrow da} = a \otimes b^{\downarrow da \cap db} \text{ and } (a \otimes b)^{\downarrow db} = a^{\downarrow da \cap db} \otimes b$$

Theorem. Let $(\Phi, \otimes, \epsilon)$ be an OVA. For each inclusion $B \subseteq A$ in \mathcal{T} , there is an adjunction/Galois connection

$$a^{\downarrow B} \leq_{\Phi_B} b \iff a \leq_{\Phi_A} \epsilon_a \otimes b$$

We call $b \mapsto \epsilon_a \otimes b$ **extension of b to A** and write $b^{\uparrow A} := \epsilon_a \otimes b$.

Extension takes a specification defined on a domain B and lifts it to a domain $A \supseteq B$ by allowing any behaviour outside B .

Some facts:

- Extension is a functor: if $C \subseteq B \subseteq A$ and $c \in \Phi_C$, then

$$(c^{\uparrow B})^{\uparrow A} = c^{\uparrow A} \text{ and } c^{\uparrow C} = c$$

- If Φ has the **strongly neutral property**: $\forall A, B \in \mathcal{T}$ we have $\epsilon_A \otimes \epsilon_B = \epsilon_{A \cup B}$, then for all $B \subseteq A$,

$$\epsilon_B^{\uparrow A} = \epsilon_A$$

- Restriction after extension is the identity map, i.e., for $B \subseteq A$ and $b \in \Phi_B$,

$$(b^{\uparrow A})^{\downarrow B} = b$$

- Extension after restriction is a **closure operator**, i.e., for $B \subseteq A$ and $a \in \Phi_A$,

$$a \leq_{\Phi_A} (a^{\downarrow B})^{\uparrow A} \quad \text{and} \quad (((a^{\downarrow B})^{\uparrow A})^{\downarrow B})^{\uparrow A} = (a^{\downarrow B})^{\uparrow A}$$

- If for each $A \in \mathcal{A}$, Φ_A is a complete lattice, then so is $\int \Phi$.

Definition. A *concurrent valuation algebra (CVA)* consists of . . .

- A commutative OVA

$$(\Phi, \parallel, \delta)$$

- A (generally noncommutative) OVA

$$(\Phi, \circ, \epsilon)$$

Obeying the following axioms. . .

- *Weak exchange.*

$$(a \parallel b) \circ (c \parallel d) \preceq (a \circ c) \parallel (b \circ d)$$

- *Neutral laws.*

$$\epsilon_A \preceq \epsilon_A \parallel \epsilon_A \text{ and } \delta_A \circ \delta_A \preceq \delta_A \quad ^1$$

¹These are actually equalities.

- *Refinement.*

$$a \preceq b \iff db \subseteq da \text{ and } a^{\downarrow db} \leq_{\Phi_{db}} b$$

- *Hoare triple (Hoare logic).*

$$p \{a\} q := p \circ a \preceq q$$

- *Jones quintuple (rely-guarantee logic).*

$$p \ r \ \{a\} \ g \ q := p \ \{a \parallel r\} \ q \text{ and } a \preceq g$$

Example (concurrency law).

Let $p, p', a, a', q, q' \in \mathcal{J} \Phi$. Then

$$p \{a\} q \text{ and } p' \{a'\} q' \implies (p \parallel p') \{a \parallel a'\} (q \parallel q')$$

Proof. Assume $p \circ a \preceq q$ and $p' \circ a' \preceq q'$. By weak exchange and monotonicity,

$$(p \parallel p') \circ (a \parallel a') \preceq (p \circ a) \parallel (p' \circ a') \preceq q \parallel q'$$

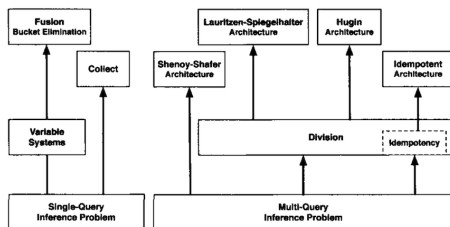
Thus, $(p \parallel p') \{a \parallel a'\} (q \parallel q')$.

Efficient computation is enabled by the *combination* law:

$$(a \otimes b)^{\downarrow da} = a \otimes b^{\downarrow da \cap db}$$

$$(a \otimes b)^{\downarrow db} = a^{\downarrow da \cap db} \otimes b$$

The LHS is usually *expensive* to compute, but the RHS can be *much cheaper* if $da \cap db$ is small.

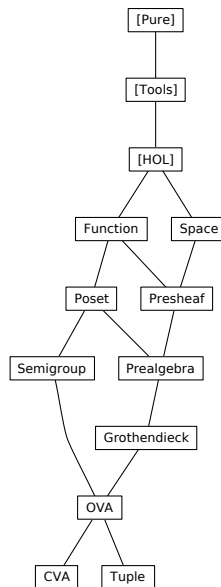


- Local computation framework specialises to standard optimal algorithms in databases, probabilistic graphical models, constraint satisfaction problems, dynamic programming, ...
- We can already use these algorithms in a CVA on just parallel combinations.
- Need to generalise to noncommutative case & with 2 combine operators.
- Theoretically feasible from the generalised combination rule, but not yet implemented.

- 1 Background
 - Refinement algebras
 - Trace models of refinement algebras
 - Project goal: localising refinement algebras
 - Topologies and presheaves
- 2 Contextuality in distributed systems
- 3 Trace models of concurrent valuation algebras
 - (covariant) Grothendieck construction
 - Ordered valuation algebras (OVAs)
 - Concurrent valuation algebras (CVAs), refinement & reasoning
 - Local computation
- 4 Computer formalisation
- 5 Summary & plan for the rest of the project

Summary.

- Section 2 of (Evangelou-Oost, Bannister, Meinicke et al. 2023), that comprises the abstract theory of CVAs, has been fully formalised in Isabelle/HOL.
- Formalisation of the remaining sections is underway.
- Available online at github.com/nasosev/cva. Collaborators welcome :)
- Acknowledgements: Thank You to the local Isabelle experts for your support: Brae Webb, Callum Bannister, Kait Lam, Scott Heiner ♥



Formalisation strategy

Light embedding of mathematical structures in Isabelle/HOL's simple type system (i.e. without dependent types) using records and validity predicates.

```
record ('x, 'y) Function =  
  cod :: "'y set"  
  func :: "('x × 'y) set"
```

```
definition dom :: "('x, 'y) Function  $\Rightarrow$  'x set" where  
  "dom f  $\equiv$  {x.  $\exists$ y. (x, y)  $\in$  func f}"
```

```
definition valid_map :: "('x, 'y) Function  $\Rightarrow$  bool" where  
  "valid_map f  $\equiv$   
    let  
      welldefined =  $\forall$ x y. (x, y)  $\in$  func f  $\longrightarrow$  y  $\in$  cod f;  
      deterministic =  $\forall$ x y y'. (x, y)  $\in$  func f  $\wedge$  (x, y')  $\in$  func f  $\longrightarrow$  y = y'  
    in welldefined  $\wedge$  deterministic"
```

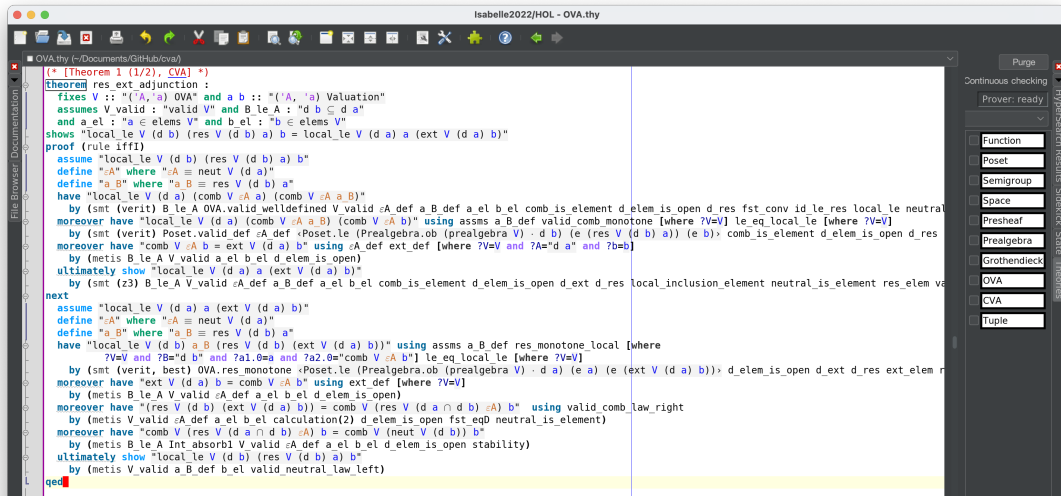
Some other type definitions (corresponding validity predicates not shown) ...

```
record 'a Semigroup =  
  mult :: "('a \<times> 'a, 'a) PosetMap"  
  
record ('A, 'a) Prealgebra =  
  space :: "'A Space"  
  ob :: "('A Open, 'a Poset) Function "  
  ar :: "('A Inclusion, ('a, 'a) PosetMap) Function"  
  
record ('A, 'a, 'b) PrealgebraMap =  
  nat :: "('A Open, ('a, 'b) PosetMap) Function"  
  dom :: "('A, 'a) Prealgebra"  
  cod :: "('A, 'b) Prealgebra"  
  
record ('A, 'a) OVA =  
  prealgebra :: "('A, 'a) Prealgebra"  
  neutral    :: "('A, unit, 'a) PrealgebraMap"  
  semigroup  :: "((('A, 'a) Valuation) Semigroup)"  
  
record ('A, 'a) CVA =  
  par_algebra :: "('A, 'a) OVA"  
  seq_algebra :: "('A, 'a) OVA"
```


Theorem 1 formalisation

Theorem (restriction-extension adjunction). Let $(\Phi, \otimes, \epsilon)$ be an OVA. Then for all $a, b \in f \Phi$ with $db \subseteq da$,

$$a^{\downarrow db} \leq_{\Phi_{db}} b \iff a \leq_{\Phi_{da}} b^{\uparrow da}$$



```
(* [Theorem 1 (1/2), CVA] *)
theorem res_ext_adjunction :
  fixes V :: "('A, 'a) OVA" and a b :: "('A, 'a) Valuation"
  assumes V_valid : "valid V" and B_le_A : "d b ⊆ d a"
  and a_el : "a ∈ elems V" and b_el : "b ∈ elems V"
  shows "local_le V (d b) (res V (d b) a) b = local_le V (d a) a (ext V (d a) b)"
proof (rule iffI)
  assume "local_le V (d b) (res V (d b) a) b"
  define "εA" where "εA ≡ neut V (d a)"
  define "a_B" where "a_B ≡ res V (d b) a"
  have "local_le V (d a) (comb V εA a) (comb V εA a_B)"
  by (smt (verit) B_le_A OVA.valid welldefined V_valid εA def a_B def a_el comb_is_element d_elem_is_open d_res fst conv id le res local_le neutral
  moreover have "local_le V (d a) (comb V εA a_B) (comb V εA b)" using assms a_B def valid comb_monotone [where ?V=V] le_eq_local_le [where ?V=V]
  by (smt (verit) Poset.valid def εA def <Poset.le (Prealgebra.ob (prealgebra V) · d b) (e (res V (d b) a)) (e b)> comb_is_element d_elem_is_open d_res
  moreover have "comb V εA b = ext V (d a) b" using εA def ext_def [where ?V=V and ?A="d a" and ?b=b]
  by (metis B_le_A V_valid a_el b_el d_elem_is_open)
  ultimately show "local_le V (d a) a (ext V (d a) b)"
  by (smt (z3) B_le_A V_valid εA def a_B def a_el b_el comb_is_element d_elem_is_open d_ext d_res local_inclusion_element neutral_is_element res_elem va
next
  assume "local_le V (d a) a (ext V (d a) b)"
  define "εA" where "εA ≡ neut V (d a)"
  define "a_B" where "a_B ≡ res V (d b) a"
  have "local_le V (d b) a_B (res V (d b) (ext V (d a) b))" using assms a_B def res_monotone_local [where
    ?V=V and ?B="d b" and ?a1.0=a and ?a2.0="comb V εA b"] le_eq_local_le [where ?V=V]
  by (smt (verit, best) OVA.res_monotone <Poset.le (Prealgebra.ob (prealgebra V) · d a) (e a) (e (ext V (d a) b))> d_elem_is_open d_ext d_res ext_elem r
  moreover have "ext V (d a) b = comb V εA b" using ext_def [where ?V=V]
  by (metis B_le_A V_valid εA def a_el b_el d_elem_is_open)
  moreover have "(res V (d b) (ext V (d a) b)) = comb V (res V (d a ∩ d b) εA) b" using valid_comb_law_right
  by (metis V_valid εA def a_el b_el calculation(2) d_elem_is_open fst eqd neutral_is_element)
  moreover have "comb V (res V (d a ∩ d b) εA) b = comb V (neut V (d b)) b"
  by (metis B_le_A Int.absorb1 V_valid εA def a_el b_el d_elem_is_open stability)
  ultimately show "local_le V (d b) (res V (d b) a) b"
  by (metis V_valid a_B def b_el valid_neutral_law_left)
qed
```

- 1 Background
 - Refinement algebras
 - Trace models of refinement algebras
 - Project goal: localising refinement algebras
 - Topologies and presheaves
- 2 Contextuality in distributed systems
- 3 Trace models of concurrent valuation algebras
 - (covariant) Grothendieck construction
 - Ordered valuation algebras (OVAs)
 - Concurrent valuation algebras (CVAs), refinement & reasoning
 - Local computation
- 4 Computer formalisation
- 5 Summary & plan for the rest of the project






Summary & plan for the rest of the project

Summary.

- We reformulated the notion of an OVA, and showed how these can model concurrent/distributed systems, and manifest sequential inconsistency as contextuality.
- We introduced CVAs as candidates for localised refinement algebras, modular/compositional structures for reasoning about concurrent/distributed systems.
- CVAs support classical refinement reasoning methodologies such as Hoare and rely-guarantee logics.
- We described 3 example trace models of CVAs and related them by morphisms.
- We indicated a path to efficient algorithms for inference, static analysis, model checking by extending the classical local computation framework of valuation algebras to CVAs.
- We have a computer formalisation underway in Isabelle/HOL.

Plan.

- Complete the formalisation of CVAs in Isabelle/HOL.
- More models: Aczel trace models with alternative extension operators, graph models, simplicial model, sheaf based models that better capture local properties. . .
- Better understand the implications of nontrivial topologies.
- Generalise the trace model constructions with multioperations.
- Categorical structure of CVAs; connection to monoidal Grothendieck fibrations.
- Push-forward/pullback of OVAs/CVAs on different spaces.

-  Evangelou-Oost, Nasos, Callum Bannister and Ian J. Hayes (2023). 'Contextuality in Distributed Systems'. In: *Relational and Algebraic Methods in Computer Science*. Springer International Publishing, pp. 52–68.
-  Evangelou-Oost, Nasos, Callum Bannister, Larissa Meinicke et al. (2023). *Trace models of concurrent valuation algebras*. arXiv: 2305.18017 [cs.LO].
-  Hayes, Ian J. (2016). 'Generalised rely-guarantee concurrency: an algebraic foundation'. In: *Formal Aspects Comput.* 28.6, pp. 1057–1078. DOI: 10.1007/s00165-016-0384-0. URL: <https://doi.org/10.1007/s00165-016-0384-0>.
-  Hoare, C. A. R. et al. (2009). 'Concurrent Kleene Algebra'. In: *CONCUR 2009 - Concurrency Theory, 20th International Conference, CONCUR 2009, Bologna, Italy, September 1-4, 2009. Proceedings*. Ed. by Mario Bravetti and Gianluigi Zavattaro. Vol. 5710. Lecture Notes in Computer Science. Springer, pp. 399–414. DOI: 10.1007/978-3-642-04081-8_27. URL: https://doi.org/10.1007/978-3-642-04081-8%5C_27.
-  Pouly, Marc and Jürg Kohlas (2012). *Generic inference: a unifying theory for automated reasoning*. John Wiley & Sons.

Contact: nasoev@proton.me, [nasosev.github.io](https://github.com/nasosev)

THANK YOU!