Cylindric Kleene Lattices for Program Construction*

Brijesh Dongol[1], Ian Hayes[2], Larissa Meinicke[2], and Georg Struth[3]

[1] University of Surrey, UK
[2] University of Queensland, Australia
[3] University of Sheffield, UK

Abstract. Cylindric algebras have been developed as an algebraisation of equational first order logic. We adapt them to cylindric Kleene lattices and their variants and present relational and relational fault models for these. This allows us to encode frames and local variable blocks, and to derive Morgan's refinement calculus as well as an algebraic Hoare logic for while programs with assignment laws. Our approach thus opens the door for algebraic calculations with program and logical variables instead of domain-specific reasoning over concrete models of the program store. A refinement proof for a small program is presented as an example.

1 Introduction

Kleene algebras and similar formalisms have found their place in program construction and verification. Kleene algebras with tests [19] have been used for calculating complex program equivalences; the rules of propositional Hoare logic—Hoare logic without assignments laws—can be derived from their axioms [20]. Demonic refinement algebras [29] have been applied to non-trivial program transformations in the refinement calculus [3]. Modal Kleene algebras [7,8] have been linked with predicate transformer semantics and found applications in program correctness. More recently, links between Kleene algebras and Morgan-style refinement calculi [23] have been established; program construction and verification components based on Kleene algebras have been formalised in proof assistants such as Coq [9,25] or Isabelle/HOL [2,15,28].

The Isabelle components are based on shallow embeddings of while programs, Hoare logic and refinement calculi. Programs, assertions and correctness specifications are modelled as semantic objects directly within Isabelle's higher-order logic. Explicit data types for the syntax of programs, assertions or logics of programs, and explicit semantic maps for their interpretation can thus be avoided. Kleene algebras, as abstract semantics for while programs, propositional Hoare logics or propositional refinement calculi, fit very naturally into this approach. Yet assignments and their laws are currently formalised in concrete program store semantics that form models of the algebras. With a shallow embedding,

* Dongol and Struth are supported by EPSRC Grant EP/R032556/2; Hayes, Meinicke and Dongol are supported by ARC Discovery Grant DP19010214.

program construction and verification is thus performed in these concrete semantics. Other familiar features of refinement calculi, such as variable frames or local variable blocks, cannot be expressed in Kleene algebras either. How algebra could handle such important features remains open.

Yet algebra *can* deal with bindings, scopes and variables. Nominal Kleene algebras [13] can model the first two features, and cylindric algebras of Henkin, Monk and Tarski [17] the third, albeit in the setting of boolean algebras, where notions of variables and quantification are added in an algebratisation of first-order equational logic. They introduce a family of cylindrification operators $c_\kappa\, x$ that abstract existential quantification $\exists_\kappa x$ of first-order formulas.

Henkin, Monk and Tarski give a standard interpretation of c_κ in *cylindric set algebras* [17, p.166]. In this setting, cylindrification is defined over $\mathcal{P}\, X^\alpha$ for some set X and ordinal α.[4] Elements of a cylindric set algebra are therefore functions of type $\alpha \to X$, or sequences $x = (x_0, x_1, \dots)$ of "length" α. In logic, if α is a set of logical variables and X the carrier set of a structure, these correspond to valuations. Geometrically, X^α corresponds to an α-dimensional Cartesian space with base X where x_κ represents the κth coordinate. Apart from the usual boolean operations on sets, cylindric set algebras use a family of cylindrification operators $C^c_\kappa : \mathcal{P}\, X^\alpha \to \mathcal{P}\, X^\alpha$ for $\kappa < \alpha$—the superscript c stannds for "classical". For each $A \subseteq X^\alpha$,

$$C^c_\kappa A = \{y \in X^\alpha \mid \exists x \in A.\ x \approx_\kappa y\},$$

where $x \approx_\kappa y$ if x and y are equal, except at κ, (i.e. $\forall \lambda \neq \kappa.\ x_\lambda = y_\lambda$). Geometrically, $C^c_\kappa A$ thus translates A along the κ-axis and constructs a cylinder in some hyperspace.

Our main idea is to generalise cylindrification from boolean algebras to Kleene lattices (thus foregoing the complement operator of boolean algebra, while adding a monoidal composition and a star). We explain it through *relational cylindrification* C_κ, which acts on programs modelled by relations in $\mathcal{P}\,(X^\alpha \times X^\alpha)$, where X^α represents program stores as functions from variables in α to values in X. Cylindrifying relation R in variable κ by $C_\kappa\, R$ means adding any combination of values for κ to elements of R. We therefore say that C_κ *liberates* variable κ in program R,

$$C_\kappa R = \{(a, b) \in X^\alpha \times X^\alpha \mid \exists (c, d) \in R.\ a \approx_\kappa c \wedge b \approx_\kappa d\}.$$

Note that κ is liberated (can take on any value) independently in both the first and second coordinates of R.[5]

The cylindrification of the identity relation, $C_\kappa\, Id_{X^\alpha}$, in particular, liberates variable κ while constraining all other variables to satisfy the identity relation on

[4] In applying Henkin, Monk and Tarski's work to program algebra we do not rely much on the use of ordinals; sets usually suffice.

[5] Expressing the relation as a predicate in the Z style [27,16], i.e. representing before values of a variable by x and after values by x', relational cylindrification corresponds to the predicate $\exists_{x,x'} R$.

X^α. Henkin, Monk and Tarski [17, §1.7] have generalised cylindrification to finite sets of variables so that $c_{(\{\kappa_0,\ldots,\kappa_{n-1}\})}x = c_{\kappa_0}\ldots c_{\kappa_{n-1}}x$, where the parentheses on the left are part of their syntax. For a set of variables Γ, a program R may be restricted to only change variables in Γ by conjoining it with $C_{(\Gamma)}Id_{X^\alpha}$, i.e. $R \cap C_{(\Gamma)}Id_{X^\alpha}$, which we abbreviate to $\Gamma : x$ to match the syntax of frames in Morgan's refinement calculus [24]. A local variable κ with a scope over some program R is obtained by first liberating the local κ over the program and then constraining any non-local κ to not change, i.e. $(C_\kappa R) \cap C_{(\overline{\{\kappa\}})}Id_{X^\alpha}$, which we abbreviate as $(\mathbf{var}\,\kappa.R)$. Finally, assignment statements are encoded by framed specification statements, where tests are used to abstract from expressions, and variable substitutions are handled using another concept from cylindric algebras, namely diagonal elements.

Our main contribution lies in the formal development of this new extension and application of cylindrification. This opens the door to algebraic calculations with variables in imperative programs where set-theoretic reasoning in concrete store semantics is so far required. Our technical contributions are as follows.

- We extend Kleene algebras (§2) to cylindric Kleene lattices (§4), explore their basic properties and prove their soundness with respect to a relational (fault) semantics for imperative programs (§3 and §5).
- Generalised cylindrification liberates a set of variables, rather than a single variable (§6). It is used to show that the frames of Morgan's refinement calculus (§8) and local variable blocks (§9) can be expressed in cylindric Kleene lattices. Based on these encodings we derive the laws of Morgan's refinement calculus with frames and those of Hoare logic (§7), both with assignment laws.
- Synchronous cylindrification (§10) supports the cylindrification of tests in the relational model. It is used in combination with diagonal elements (representing equality in equational logic) to define substitutions algebraically (§11). These are then used to define variable assignments (§12).
- We explain how simple refinement proofs can be performed in our framework by purely algebraic and symbolic reasoning.
- We propose liberation Kleene lattices (§13) as a conceptually simpler and more fine-grained variant, and prove that the axioms of cylindric Kleene lattices are derivable from those of liberation Kleene lattices.

Many of our results have been verified with Isabelle/HOL, but verification and refinement components based on cylindric Kleene algebras remain work in progress. All Isabelle proofs are accessible online[6].

Overall, many of the concepts needed for our development could be readily adapted from cylindric algebra. Henkin, Monk and Tarski's textbook [17] has been a surprising source of insights from a seemingly unrelated area. We follow their notational conventions closely.

[6] https://github.com/gstruth/liberation

2 l-Monoids and Kleene Lattices

This section briefly recalls the basic algebraic structures used in this article. Cylindric variants are presented in Section 4, liberation algebras are introduced in Section 13. We work with l-monoids instead of dioids and Kleene lattices instead of Kleene algebras because a meet operation is crucial for defining the concepts we care about: frames, local variables and variable assignments.

Definition 1 (l-monoid). *A* lattice-ordered monoid *(*l-monoid*) [4] is a structure* $(L, +, \cdot, ;, 0, 1)$ *such that* $(L, +, \cdot, 0)$ *is a lattice with join operation* $+$*, meet operation* $\cdot$*, and least element* 0*;* $(L, ;, 1)$ *is a monoid and the distributivity axioms* $x ; (y + z) = x ; y + x ; z$ *and* $(x + y) ; z = x ; y + x ; z$ *and annihilation axioms* $0 ; x = 0$ *and* $x ; 0 = 0$ *hold for all* $x, y, z \in L$*. An l-monoid is* weak *if the axiom* $x ; 0 = 0$ *is absent.*

Definition 2 (Kleene lattice). *A (weak)* Kleene lattice *[18,1] is a (weak) l-monoid,* K*, equipped with a star operation* $^* : K \to K$*, that satisfies the unfold and induction axioms*

$$1 + x ; x^* \leq x^*, \qquad z + x ; y \leq y \Rightarrow x^* ; z \leq y,$$
$$1 + x^* ; x \leq x^*, \qquad z + y ; x \leq y \Rightarrow z ; x^* \leq y.$$

The unfold and induction laws in the first line and those in the second line above are opposites: the order of composition has been swapped.

Forgetting the meet operation in l-monoids yields dioids (i.e., semirings with idempotent addition); forgetting meet in Kleene lattices yields Kleene algebras.

Definition 3 (l-monoid with tests). *A (weak)* l-monoid with tests *is a structure* $(L, B, +, \cdot, ;, 0, 1, \neg)$ *where* $B \subseteq L$*,* $\neg$ *is a partial operation defined on* B *such that* $(B, +, \cdot, 0, 1, \neg)$ *is a boolean algebra in which* $;$ *and* $\cdot$ *coincide and* $(L, +, \cdot, ;, 0, 1)$ *is a (weak) l-monoid. In addition, for all* $p \in B$ *and* $x, y \in K$*,*

$$p ; (x \cdot y) = (p ; x) \cdot (p ; y), \qquad \text{and} \qquad (x \cdot y) ; p = (x ; p) \cdot (y ; p).$$

Definition 4 (Kleene lattice with tests). *A (weak)* Kleene lattice with tests *is a (weak) l-monoid with tests that is also a (weak) Kleene lattice.*

Alternatively, Kleene lattices can be based on the operation $^+ : K \to K$ that satisfies the following unfold and induction axioms

$$x + x ; x^+ = x^+, \qquad z + x ; y \leq y \Rightarrow z + x^+ ; z \leq y,$$

and their opposites $x + x^+ ; x = x^+$ and $z + y ; x \leq y \Rightarrow z + z ; x^+ \leq y$, even when the unit 1 is absent. In the presence of this unit, the identities $x^+ = x ; x^*$ and $x^* = 1 + x^+$ make the two variants interderivable.

3 Relation Kleene Lattices

Before cylindrifying l-monoids and Kleene lattices in the next section, we sketch the relational model and the relational fault model of these algebras. First of all, these form the basis of the standard relational program semantics to which we restrict our attention. Secondly, they are used in the soundness proofs of the cylindric and liberation algebras that we axiomatise. Last, but not least, they provide valuable intuitions for the algebraic development.

A standard model of Kleene algebra with tests is formed by the algebra of binary relations over a set X. In this model, $+$ is interpreted as union, $;$ as relational composition ($(a,b) \in R\,;S \Leftrightarrow \exists c \in X.\ (a,c) \in R \wedge (c,b) \in S$), 0 as $\emptyset$, 1 as the identity relation on X, ($(a,b) \in Id_X \Leftrightarrow a = b$), and * as the reflexive-transitive closure operation ($R^* = \bigcup_{i<\omega} R^i$, for $R^0 = Id_X$ and $R^{i+1} = R\,;R^i$). As our basis is a lattice, $\cdot$ is interpreted as intersection. Finally, tests are *subidentities*, that is, elements of $\mathcal{P}Id_X = \{R \subseteq X \times X \mid R \subseteq Id_X\}$. These distribute over infs in both arguments with respect to sequential composition. Test complementation is defined by $Id_X - (\_)$. The test algebra $\mathcal{P}Id_X$ forms a subalgebra of any algebra $\mathcal{P}(X \times X)$ of binary relations—in fact a complete atomic boolean algebra. The following result is therefore routine.

Proposition 5. *Let X be a set. Then $(\mathcal{P}(X \times X), \mathcal{P}Id_X, \cup, \cap, ;, \emptyset, Id_X, -, ^*)$ is a Kleene lattice with tests—the* full relation Kleene lattice with tests *over X.*

Weak Kleene lattices with tests are formed by relations that model faults or nontermination over $X \times X_\perp$, where $X_\perp = X \cup \{\perp\}$ and $\perp \notin X$ is an element that represents a fault or non-termination. We refer to this model as the *relational fault model*. We partition each $R \subseteq X \times X_\perp$ into its *proper part* $R_p \subseteq X \times X$ and its *faulting part* $R_f \subseteq X \times \{\perp\}$, that is, $R = R_p \cup R_f$ and $R_p \cap R_f = \emptyset$. Redefining $R\,;S = R_f \cup R_p;S$ then makes faults override compositions, representing R as (R_p, R_f) and S by (S_p, S_f) yields the semidirect product, which is well known in semigroup theory:

$$(R_p, R_f)\,;(S_p, S_f) = (R_p\,;S_p, R_f \cup R_p\,;S_f). \tag{1}$$

With $(R_p, R_f)^0 = (Id_X, \emptyset)$ and $(R_p, R_f)^{i+1} = (R_p, R_f)\,;(R_p, R_f)^i$ we define

$$(R_p, R_f)^* = \bigcup_{i<\omega} (R_p, R_f)^i. \tag{2}$$

An inductive argument shows that * satisfies the Kleene algebra axioms and that

$$(R_p, R_f)^* = (R_p^*, R_p^*\,;R_f).$$

Proposition 6. *Let X be a set. Then $(\mathcal{P}(X \times X_\perp), \mathcal{P}Id_X, \cup, \cap, ;, \emptyset, Id_X, -, ^*)$, with composition (1) and star (2), forms a weak Kleene lattice with tests—the* full weak relation Kleene lattice with tests *over X.*

The identity of the pair representation with respect to ; is $(Id_X, \emptyset)$; its left zero is $(\emptyset, \emptyset)$. All tests are proper and test complementation is restricted to the proper part. Right annihilation fails because $(R_p, R_f) \,;\, (\emptyset, \emptyset) = (\emptyset, R_f) \neq (\emptyset, \emptyset)$ whenever $R_f \neq \emptyset$. Algebraic proofs for this development can be found in Appendix A; it has been formalised with Isabelle.

Each subalgebra (K, B), with $K \subseteq \mathcal{P}(X \times X)$ and $B \subseteq \mathcal{P} Id_X$, of a full (weak) relation Kleene lattice with tests over X is a *(weak) relation Kleene lattice with tests* over X.

The relation algebras described in this section have of course a much richer structure. Firstly, we ignore the fact that relations have converses and can be complemented, yet this only means that we focus on the programming concepts that matter. Secondly, relational composition preserves sups in both arguments, whereas the redefined composition (1) preserves sups in its first and non-empty sups in its second argument. Non-preservation of empty sups in the second argument is of course due to the absence of right annihilation.

4 Cylindric l-Monoids and Kleene Lattices

This section extends l-monoids and Kleene lattices from Section 2 by a family of cylindrification operators. In other words, we generalise the classical cylindric algebras of Henkin, Monk and Tarski [17] from boolean algebras to Kleene algebras. The axiomatisations have been developed, minimised and proved to be independent using Isabelle/HOL. Apart from the axioms, we present some simple algebraic properties, all of which have been verified with Isabelle. The relational models from Section 3 are extended to models for cylindric l-monoids and Kleene lattices in Section 5. In reading the following definition a suitable intuition is that $c_\kappa x$ represents an abstraction of existential quantification $\exists_\kappa x$.

Definition 7 (cylindric l-monoid). *Let α be an ordinal. A (*weak*)* cylindric l-monoid *(*CLM*) of dimension α is a structure $(L, +, \cdot, ;, 0, 1, c_\kappa)_{\kappa<\alpha}$ such that $(L, +, \cdot, ;, 0, 1)$ is a (weak) l-monoid and $c_\kappa : L \to L$ satisfies:*

$$c_\kappa\, 0 = 0, \tag{C1}$$
$$x \leq c_\kappa\, x, \tag{C2}$$
$$c_\kappa\,(x \cdot c_\kappa\, y) = c_\kappa\, x \cdot c_\kappa\, y, \tag{C3}$$
$$c_\kappa c_\lambda\, x = c_\lambda c_\kappa\, x, \tag{C4}$$
$$c_\kappa\,(x + y) = c_\kappa\, x + c_\kappa\, y, \tag{C5}$$
$$c_\kappa\,(x \,;\, c_\kappa\, y) = c_\kappa\, x \,;\, c_\kappa\, y, \tag{C6}$$
$$c_\kappa\,(c_\kappa\, x \,;\, y) = c_\kappa\, x \,;\, c_\kappa\, y, \tag{C7}$$
$$\kappa \neq \lambda \;\Rightarrow\; c_\kappa\, 1 \cdot c_\lambda\, 1 = 1, \tag{C8}$$
$$(c_\kappa\, 1 \,;\, c_\lambda\, 1) \cdot (c_\kappa\, 1 \,;\, c_\mu\, 1) = c_\kappa\,(c_\lambda\, 1 \cdot c_\mu\, 1), \tag{C9}$$
$$c_\kappa\,(c_\lambda\, 1) = c_\kappa\, 1 \,;\, c_\lambda\, 1. \tag{C10}$$

Classical cylindric algebra is axiomatised over a boolean algebra instead of a Kleene lattice; a monoidal structure is absent. Cylindric algebras usually consider diagonal elements $d_{\kappa\lambda}$ as well [17]. In this sense CLM is *diagonal free*. CLMs with diagonals are introduced in Section 11.

Axioms (C1), (C2), (C3) and (C4) are those of classical cylindric algebra [17, p.162]; (C5) is derivable in that context because it is based on a boolean algebra. The axioms (C6) and (C7) appear in a previous abelian-semiring-based approach to cylindrification by Giacobazzi, Debray and Levi [14]. Axioms (C8)-(C10) are new. In axioms (C1), (C5), (C9) and (C10), $=$ could have been weakened to $\leq$. Isabelle's counterexample generators show that the axioms are independent. We write 1_κ instead of $c_\kappa\, 1$. Intuitively, such elements are identities of ; except for κ. The next lemmas establish basic facts about cylindrification. The properties in the first one are known from classical cylindric algebras.

Lemma 8. *[17, §1.2] In every weak* CLM*,*

1. $c_\kappa\, c_\kappa\, x = c_\kappa\, x$, *(HMT1.2.3)*
2. $c_\kappa\, x = 0 \Leftrightarrow x = 0$, *(HMT1.2.1)*
3. $c_\kappa\, x = x \Leftrightarrow \exists y.\ c_\kappa\, y = x$, *(HMT1.2.4)*
4. $x \cdot c_\kappa\, y = 0 \Leftrightarrow y \cdot c_\kappa\, x = 0$, *(HMT1.2.5)*
5. $x \leq y \Rightarrow c_\kappa\, x \leq c_\kappa\, y$, *(HMT1.2.7)*
6. $x \leq c_\kappa\, y \Leftrightarrow c_\kappa\, x \leq c_\kappa\, y$, *(HMT1.2.9)*
7. $c_\kappa\, x \cdot c_\lambda\, y = 0 \Leftrightarrow c_\lambda\, x \cdot c_\kappa\, y = 0$. *(HMT1.2.15)*

Axiom (C2) and Lemma 8(1) and (5) can be summarised as follows.

Lemma 9. *In every weak* CLM*, c_κ is a closure operator.*

The next lemma collects properties beyond classical cylindrical algebra.

Lemma 10. *In every weak* CLM*,*

1. $c_\kappa\, (x\,;\, y) \leq c_\kappa\, x\,;\, c_\kappa\, y$,
2. $1_\kappa\,;\, x\,;\, 1_\kappa \leq c_\kappa\, x$,
3. $1 \leq 1_\kappa$,
4. $1_\kappa\,;\, 0 = 0$,
5. $1_\kappa\,;\, 1_\kappa = 1_\kappa$,
6. $1_\kappa\,;\, 1_\lambda = 1_\lambda\,;\, 1_\kappa$,
7. $c_\kappa\, (1_\lambda \cdot 1_\mu) = 1_\kappa\,;\, (1_\lambda \cdot 1_\mu)$,
8. $1_\kappa + 1_\lambda \leq 1_\kappa\,;\, 1_\lambda$.

Lemma 10(2) may be strengthened to an equality in the relational model of CLM, but neither in trace models [21] nor in the algebra; the following lemma gives a counterexample.

Lemma 11. *There is a* CLM *in which $c_\kappa\, x \not\leq 1_\kappa\,;\, x\,;\, 1_\kappa$.*

Proof. Consider the CLM with $L = \{0, 1, a\}$, join and meet defined by $0 < a < 1$ and composition by $a;a = a$. It can be checked that $c_\kappa : 0 \mapsto 0, a \mapsto 1, 1 \mapsto 1$ satisfies (C1)-(C7). Yet $c_\kappa\, a = 1 \neq a = 1\,;\, a\,;\, a = 1_\kappa\,;\, a\,;\, 1_\kappa$. □

In any (weak) CLM L, let

$$L_{c_\kappa} = \{x \in L \mid c_\kappa\, x = x\},$$

denote the set of cylindrified elements in dimension κ. Similarly, we define $L^l_{1_\kappa}$, $L^r_{1_\kappa}$ and L_{1_κ} as the sets of fixpoints of $1_\kappa\,;(\_)$, $(\_)\,;1_\kappa$ and $1_\kappa\,;(\_)\,;1_\kappa$, respectively. Lemma 8(3) implies that L_{c_κ} is equal to the image of L under c_κ. Analogous facts hold for the other three functions.

Proposition 12. *Let L be a (weak)* CLM *and let $\kappa < \alpha$. Then*

1. $(L^l_{1_\kappa}, +, \cdot, ;, 0, 1_\kappa)$ *forms a (weak) sub-l-semigroup of L with left unit 1_κ,*
2. $(L^r_{1_\kappa}, +, \cdot, ;, 0, 1_\kappa)$ *forms a sub-l-semigroup of L with right unit 1_κ and if L is a strong* CLM*, then $L^l_{1_\kappa}$ and $L^r_{1_\kappa}$ are isomorphic,*
3. $(L_{1_\kappa}, +, \cdot, ;, 0, 1_\kappa)$ *forms a sub-l-monoid of $L^l_{1_\kappa}$ and $L^r_{1_\kappa}$,*
4. $(L_{c_\kappa}, +, \cdot, ;, 0, 1_\kappa)$ *forms a (weak) sub-l-monoid of L_{1_κ}.*

Proof.

1. For $L^l_{1_\kappa}$, it is well known that any principal right-ideal of an idempotent in a monoid forms a subsemigroup with the idempotent as left unit. By Lemma 10(5), 1_κ is an idempotent; $L^l_{1_\kappa}$ is the principal right-ideal generated by 1_κ by definition. Closure with respect to sups follows from the dioid axioms in L and idempotence of 1_κ; inf-closure from $1_\kappa\,;(1_\kappa\,;x\cdot 1_\kappa\,;y) = 1_\kappa\,;x\cdot 1_\kappa\,;y$, which has been checked with Isabelle.
2. The proof for $L^r_{1_\kappa}$ follows from that of $L^l_{1_\kappa}$ by opposition, using the dual identity $(x\,;1_\kappa\cdot y\,;1_\kappa)\,;1_\kappa = x\,;1_\kappa\cdot y\,;1_\kappa$ for inf-closure. Right annihilation in $L^r_{1_\kappa}$ follows from Lemma 10(4). In the strong case, the isomorphism is given by opposition.
3. The subalgebra proof for L_{1_κ} follows from (1) and (2). Checking that L_{1_κ} is a subalgebra of both $L^l_{1_\kappa}$ and $L^r_{1_\kappa}$ is straightforward: by idempotence of 1_κ, every fixpoint of L_{1_κ} is a fixpoint of $L^l_{1_\kappa}$ and $L^r_{1_\kappa}$.
4. For L_{c_κ}, closure with respect to $+$, $\cdot$ and $;$ is immediate from the axioms. Sup-closure, for instance, means checking that $c_\kappa\,(c_\kappa\,x + c_\kappa\,y) = c_\kappa\,x + c_\kappa\,y$. Finally, 1_κ is the unit in the subalgebra because $1_\kappa\,;c_\kappa\,x = c_\kappa\,x = c_\kappa\,x\,;1_\kappa$. This property, which also establishes that L_{c_κ} is a subalgebra of L_{1_κ}, has been confirmed by Isabelle.

□

By Lemma 11, the sets of fixpoints of L_{c_κ} and L_{1_κ} need not coincide. Separating the remaining sets of fixpoints with Isabelle's counterexample generators is a simple exercise and need not be expanded.

Definition 13 (cylindric Kleene lattice). *A (*weak*)* cylindric Kleene lattice *(*CKL*) of dimension α is a (weak) cylindric l-monoid of dimension α that is also a (weak) Kleene lattice, and in which*

$$c_\kappa\, x^+ \leq (c_\kappa\, x)^+. \tag{C11}$$

Isabelle's counterexample generators show that 1 need not be in K_{c_κ} for any κ, in particular not in the relational models described in Section 5. Together with Proposition 12 this explains why a $+$-axiom appears in CKL, and not a $*$-axiom. Next we list properties of cylindric Kleene lattices.

Lemma 14. *In every weak* CKL*,*

1. $c_\kappa\, x^* \leq 1_\kappa\,;\,(c_\kappa\, x)^*$,
2. $c_\kappa\,(c_\kappa\, x)^+ = (c_\kappa\, x)^+$,
3. $1_\kappa\,;\,(c_\kappa\, x)^+ = (c_\kappa\, x)^+$,
4. $1_\kappa\,;\,(c_\kappa\, x)^+ = (c_\kappa\, x)^+\,;\,1_\kappa$,
5. $c_\kappa\,(c_\kappa\, x)^* = 1_\kappa\,;\,(c_\kappa\, x)^*$,
6. $1_\kappa\,;\,(c_\kappa\, x)^* = (c_\kappa\, x)^*\,;\,1_\kappa$,
7. $(1_\kappa + 1_\lambda)^+ = 1_\kappa\,;\,1_\lambda = (1_\kappa + 1_\lambda)^*$,
8. $1_\kappa^+ = 1_\kappa = 1_\kappa^*$.

Finally, Proposition 12 extends to CKL.

Proposition 15. $(K_{c_\kappa}, +, \cdot, ;, 0, 1_\kappa, (\_)^+)$ *is a (weak) sub-Kleene lattice of the (weak)* CKL *K for each $\kappa < \alpha$.*

The cases of $K^l_{1_\kappa}$, $K^r_{1_\kappa}$ and K_{1_κ} are analogous. The first two benefit from the fact that $(\_)^+$ can be used to define sub-Kleene lattices $(K^l_{1_\kappa}, +, \cdot, ;, 0, 1_\kappa, (\_)^+)$ and its opposite $(K^r_{1_\kappa}, +, \cdot, ;, 0, 1_\kappa, (\_)^+)$ that do not require 1_κ.

5 Relational Cylindrification

In constructions of cylindric algebras of formulas of predicate logic, sequences in X^α correspond to valuations [17]. They associate variables of first-order formulas with values in their models. In imperative programming languages, functions from variables in α to values in X form the standard model of program stores, and the standard denotational semantics interprets programs as relations between these. Our aim is to model cylindrifications over such relations.

Hence we consider relations $R \subseteq X^\alpha \times X^\alpha$ and *relational cylindrifications* $C_\kappa : \mathcal{P}\,(X^\alpha \times X^\alpha) \to \mathcal{P}\,(X^\alpha \times X^\alpha)$ that liberate the value of variable κ in both coordinates of ordered pairs. Formally, we therefore define

$$C_\kappa\, R = \{(a,b) \in X^\alpha \times X^\alpha \mid \exists c, d \in X^\alpha.\ (a,b) \approx_\kappa (c,d) \wedge (c,d) \in R\},$$

where $\approx_\kappa$ has been extended pointwise to an equivalence on pairs: $(a,b) \approx_\kappa (c,d)$ if and only if $a \approx_\kappa c$ and $b \approx_\kappa d$.

Operationally, therefore, $C_\kappa\, R$ is constructed from R by adding all those pairs to R that are equal to some element of R, except at κ, in both their first and their second coordinate. In particular,

$$(a,b) \in (Id_{X^\alpha})_\kappa \Leftrightarrow a \approx_\kappa b.$$

On pairs, $(a,b) \approx_\kappa (c,d) \Leftrightarrow \exists e, f.\ (a,b) = (c[\kappa \leftarrow e], d[\kappa \leftarrow f])$. Thus

$$C_\kappa\, R = \{(a,b) \mid \exists e, f.\ (a[\kappa \leftarrow e], b[\kappa \leftarrow f]) \in R\}$$

presents relational cylindrification in a way that is particularly suggestive for programming: $C_\kappa\, R$ is obtained from R by updating variable κ "asynchronously" in the pre-state and post-state of R in all possible ways.

We henceforth write Id and Id_κ when the underlying set X^α is obvious. An important property is that the relational cylindrification of Id suffices to express all other relational cylindrifications.

Lemma 16. *Let $R \subseteq X^\alpha \times X^\alpha$. Then*

$$C_\kappa\, R = Id_\kappa\, ;\, R\, ;\, Id_\kappa.$$

We have proved this fact with Isabelle. By Lemma 11, CKL is too weak to capture this property, but we expect it to fail, for instance, in trace models for which cylindrification by κ liberates κ in every state in the trace [21], not just the first and last states, i.e. $\approx_\kappa$ is lifted to apply to every state in the traces.

Some rewriting may be helpful to understand the actions of $Id_\kappa\, ;\, (\_)$ and $(\_)\, ;\, Id_\kappa$ on relations: $Id_\kappa\, ;\, R = \{(a,b) \mid \exists c \in X.\ (a[\kappa \leftarrow c], b) \in R\}$ and $R\, ;\, Id_\kappa$ acts similarly on second coordinates. Thus $Id_\kappa ; R$ models a left-handed relational cylindrification of first coordinates and $R\, ;\, Id_\kappa$ its right-handed opposite.

For faulting relations, $C_\kappa : \mathcal{P}\,(X^\alpha \times X^\alpha_\perp) \to \mathcal{P}\,(X^\alpha \times X^\alpha_\perp)$ is determined by Lemma 16 as $(Id_\kappa, \emptyset)\, ;\, (R_p, R_f); (Id_\kappa, \emptyset)$, which yields

$$C_\kappa\, R = (C_\kappa\, R_p, Id_\kappa\, ;\, R_f).$$

Hence we cylindrify the proper part of R and the first coordinate of its faulting part. This prevents the leakage of faults into proper parts of relations. We recall that $\mathcal{P}Id_{X^\alpha}$ is the set of subidentities over X^α.

Proposition 17. *For every ordinal α and set X,*

1. $(\mathcal{P}\,(X^\alpha \times X^\alpha), \mathcal{P}\,Id_{X^\alpha}, \cup, \cap, ;, \emptyset, Id_{X^\alpha}, -, ^*, C_\kappa)_{\kappa<\alpha}$ *is a* CKL *with tests;*
2. $(\mathcal{P}(X^\alpha \times X^\alpha_\perp), \mathcal{P}\,Id_{X^\alpha}, \cup, \cap, ;, \emptyset, Id_{X^\alpha}, -, ^*, C_\kappa)_{\kappa<\alpha}$, *with composition (1) and star (2), is a weak* CKL *with tests.*

Proof. Liberation Kleene lattices and their weak variants are introduced in Section 13. Proposition 43 in that section shows that every (weak) liberation Kleene lattice is a (weak) CKL. Lemma 44 in the same section shows that the liberation Kleene lattice axioms hold in $\mathcal{P}\,(X^\alpha \times X^\alpha)$ while $\mathcal{P}\,(X^\alpha \times X^\alpha_\perp)$ satisfies the weak liberation Kleene lattice axioms. □

We call $\mathcal{P}\,(X^\alpha \times X^\alpha)$ the (full) *relation* CKL *with tests* over X^α and $\mathcal{P}\,(X^\alpha \times X^\alpha_\perp)$ the (full) *weak relation* CKL *with tests* over X^α.

Henkin, Monk and Tarski show that classical cylindric algebras are closed under direct products. Yet $\mathcal{P}\,X^\alpha \times \mathcal{P}\,X^\alpha$ and $\mathcal{P}\,(X^\alpha \times X^\alpha)$ are not isomorphic and thus our axiomatisation of CKL cannot be explained in terms of a simple

pair construction on classical cylindric algebras. Nevertheless, many properties, for instance in Lemmas 8 and 14, translate from their setting into ours, and relations in $\mathcal{P}(X^\alpha \times X^\alpha)$ can of course be encoded as predicates in $\mathcal{P}\, X^{2\alpha}$ or higher dimensions.[7] As the elementary theory of binary relations is captured by classical cylindric algebra, it can be expected that at least relation CLM can be expressed in this setting, yet rather indirectly.[8]

6 Generalised Cylindrification

Modelling frames in Morgan's refinement calculus through cylindrification requires the consideration of sets of variables, at least finite ones, and the liberation of these. Henkin, Monk and Tarski [17, §1.7] have already generalised cylindrification from single variables to finite sets. We merely need to translate their approach into CKL, and this is the purpose of this section. Once again, all properties in this section have been verified with Isabelle.

For a finite subset Γ of an ordinal α, we follow Henkin, Monk and Tarski in defining

$$c_{(\emptyset)} = id \qquad \text{and} \qquad c_{(\kappa,\Gamma)} = c_\kappa \circ c_{(\Gamma)},$$

where id is the identity function on X^α, $\circ$ is function composition, and $c_{(\kappa,\Gamma)}$ abbreviates $c_{(\{\kappa\}\cup\Gamma)}$. A simple proof by induction shows that

$$c_{(\Gamma)} \circ c_{(\Delta)} = c_{(\Gamma\cup\Delta)} \tag{HMT1.7.3}$$

holds for all finite subsets Γ and Δ of α.

Henkin, Monk and Tarski call an element x of a CKL *rectangular* if

$$c_{(\Gamma)}\, x \cdot c_{(\Delta)}\, x = c_{(\Gamma\cap\Delta)}\, x \tag{HMT1.10.6}$$

holds for all finite sets Γ and Δ. They show in the classical setting that x is rectangular if and only if $c_{(\kappa,\Gamma)}\, x \cdot c_{(\lambda,\Gamma)}\, x = c_{(\Gamma)}\, x$ holds for all κ, λ and finite Γ, such that $\kappa \neq \lambda$. By defining rectangular elements of a CKL in the same way, their proof transfers to CKL. We henceforth abbreviate $c_{(\Gamma)}\, 1$ as $1_{(\Gamma)}$. Our main interest in rectangularity lies in the following inf-closure property.

Lemma 18. *In every relation* CKL*, Id is rectangular; for all finite* Γ *and* Δ*,*

$$Id_{(\Gamma)} \cap Id_{(\Delta)} = Id_{(\Gamma\cap\Delta)}.$$

[7] This is similar to the predicative encoding of relations in the Z style [27,16], in which the value of a variable κ in the initial state is represented by κ and its value in the final state is represented by κ'; relational cylindrification in Z is represented by $\exists_{\kappa,\kappa'} R$, i.e. $C_\kappa C_{\kappa'} R$ in the relational model. That is, relations are encoded using a set of variables, which for each program variable κ also contains κ'.

[8] We are grateful to an anonymous referee for pointing out an encoding.

Proof. Defining the equivalence $a \approx_\Gamma b$ as $\forall \lambda \notin \Gamma.\ a_\lambda = b_\lambda$, it is easy to check that $(a,b) \in Id_{(\Gamma)} \Leftrightarrow a \approx_\Gamma b$. Hence

$$\begin{aligned}(a,b) \in Id_{(\Gamma)} \cap Id_{(\Delta)} &\Leftrightarrow a \approx_\Gamma b \wedge a \approx_\Delta b\\ &\Leftrightarrow \forall \lambda.\ (\lambda \notin \Gamma \Rightarrow a_\lambda = b_\lambda) \wedge (\lambda \notin \Delta \Rightarrow a_\lambda = b_\lambda)\\ &\Leftrightarrow \forall \lambda.\ \lambda \notin (\Gamma \cap \Delta) \Rightarrow a_\lambda = b_\lambda\\ &\Leftrightarrow a \approx_{\Gamma\cap\Delta} b\\ &\Leftrightarrow (a,b) \in Id_{(\Gamma\cup\Delta)}.\end{aligned}$$

□

At the moment, we are nevertheless neither able to derive rectangularity of 1 from the CKL axioms nor to refute its derivability.

Question 19. Do the CKL axioms imply that 1 is rectangular? Otherwise, is there any finitary extension of these axioms that implies this fact?

We henceforth indicate explicitly, whenever rectangularity of 1 is assumed.

Henkin, Monk and Tarski have also shown that the axioms of classical cylindric algebras generalise to finite sets. This fact extends to CKL as well.

Lemma 20. *In every* CKL *the following generalisations of axioms (C1)-(C11) hold. For all finite* $\Gamma, \Delta, E \subseteq \alpha$,

1. $c_{(\Gamma)}\, 0 = 0$,
2. $x \leq c_{(\Gamma)}\, x$,
3. $c_{(\Gamma)}\,(x \cdot c_{(\Gamma)}\, y) = c_{(\Gamma)}\, x \cdot c_{(\Gamma)}\, y$,
4. $c_{(\Gamma)} c_{(\Delta)}\, x = c_{(\Delta)} c_{(\Gamma)}\, x$,
5. $c_{(\Gamma)}\,(x + y) = c_{(\Gamma)}\, x + c_{(\Gamma)}\, y$,
6. $c_{(\Gamma)}\,(x\,;\, c_{(\Gamma)}\, y) = c_{(\Gamma)}\, x\,;\, c_{(\Gamma)}\, y$,
7. $c_{(\Gamma)}\,(c_{(\Gamma)}\, x\,;\, y) = c_{(\Gamma)}\, x\,;\, c_{(\Gamma)}\, y$,
8. $\Gamma \cap \Delta = \emptyset \Rightarrow 1_{(\Gamma)} \cdot 1_{(\Delta)} = 1$, *assuming* 1 *is rectangular,*
9. $(1_{(\Gamma)}\,;\, 1_{(\Delta)}) \cdot (1_{(\Gamma)}\,;\, 1_{(E)}) = 1_{(\Gamma)}; (1_{(\Delta)} \cdot 1_{(E)})$, *assuming* 1 *is rectangular,*
10. $c_{(\Gamma)}\, 1_{(\Delta)} = 1_{(\Gamma)}\,;\, 1_{(\Delta)}$,
11. $c_{(\Gamma)}\, x^+ \leq (c_{(\Gamma)}\, x)^+$.

In addition,

12. $\Gamma \subseteq \Delta \Rightarrow c_{(\Gamma)}\, x \leq c_{(\Delta)}\, x$,
13. $1_{(\Gamma)}\,;\, 1_{(\Delta)} = 1_{(\Gamma\cup\Delta)}$,
14. $(1_{(\Gamma)})^* = 1_{(\Gamma)} = (1_{(\Gamma)})^+$.

These properties, plus rectangularity of 1, could be used for a set-based axiomatisation of cylindrification, in which the c_k appear as special cases.

At the end of this section we study the algebra of generalised cylindrified units $1_{(\Gamma)}$. First of all, these units need not be closed under sups.

Lemma 21. *In some (relation)* CKL*, generalised cylindrified units need not be closed under sups.*

Proof. Let $X = \{a, b\}$ and $\alpha = 2$. Then, for $\kappa < \alpha$,

$$Id_\kappa = \left\{ \left(\begin{pmatrix} x_0 \\ x_1 \end{pmatrix}, \begin{pmatrix} y_0 \\ y_1 \end{pmatrix} \right) \in X^2 \times X^2 \;\middle|\; x_{1-\kappa} = y_{1-\kappa} \right\}.$$

It is easy to check that $Id \neq Id_k \neq Id_0 \cup Id_1$. In addition,

$$\left(\begin{pmatrix} a \\ b \end{pmatrix}, \begin{pmatrix} b \\ a \end{pmatrix} \right) \notin Id_0 \cup Id_1$$

and hence $Id_0 \cup Id_1 \neq Id_0 \,;\, Id_1 = Id_{(\{0,1\})} = X^2 \times X^2$. Therefore $Id_0 \cup Id_1$ is none of the generalised cylindrified units Id, Id_0, Id_1, $Id_{(\{0,1\})}$ in $X^2 \times X^2$. □

Proposition 22. *Let K be a (weak)* CKL *and suppose that* 1 *is rectangular. Let*

$$\mathbb{1} = \{1_{(\Gamma)} \mid \Gamma \text{ is a finite subset of } \alpha\}.$$

1. *Then $(\mathbb{1}, ;, \cdot)$ forms a distributive lattice with sup ;, inf $\cdot$ and least element* 1*;*
2. *if α is finite, then $\mathbb{1}$ forms a finite boolean algebra with greatest element $1_{(\alpha)}$;*
3. *the map $1_{(\_)}$ from the set of finite subsets of α into $\mathbb{1}$ is a surjective lattice morphism that preserves minimal and (existing) maximal elements.*

Proof.

1. Composition in $\mathbb{1}$ is clearly associative, commutative and idempotent by Lemma 20. The distributivity laws between ; and $\cdot$ follow from Lemma 20(9), (10) and identity (HMT1.7.3). The absorption laws $1_{(\Gamma)} \,;\, (1_{(\Gamma)} \cdot 1_{(\Delta)}) = 1_{(\Gamma)}$ and $1_{(\Gamma)} \cdot (1_{(\Gamma)} \,;\, 1_{(\Delta)}) = 1_{(\Gamma)}$ have been verified with Isabelle. By rectangularity, $\mathbb{1}$ is closed under infs; by Lemma 20(13), the set is closed under composition. By definition, $1_{(0)} = 1$.
2. For finite α, Lemma 20(12) implies that $1_{(\alpha)}$ is the greatest element in $\mathbb{1}$.
3. The map $1_{(\_)}$ preserves sups by Lemma 20(13), infs by rectangularity of 1, least elements by (1) and greatest elements by (2), whenever α is finite. Surjectivity is obvious.

□

Isabelle's counterexample generators show that $1_{(\_)}$ need not be injective in CKL. Hence the lattice of these finite sets need not to be isomorphic to the lattice $\mathbb{1}$.

Lemma 23. *Let $\mathcal{P}(X^\alpha \times X^\alpha)$ by a relation* CKL *with $|X| > 1$. Then $Id_{(\_)}$ is a lattice isomorphism.*

Proof. Relative to Proposition 22, it remains to show that $Id_{(\_)}$ is injective. First we consider singleton sets. For $|X| > 1$, Id is obviously a strict subset of any Id_κ. Hence $\kappa \neq \lambda$ implies $Id_\kappa \cap Id_\lambda \neq Id_\kappa$ by (C8) and therefore $Id_\kappa \neq Id_\lambda$.

Next, suppose $\Gamma \neq \Delta = \{\lambda_1, \ldots, \lambda_n\}$ and, without loss of generality, that $\kappa \in \Gamma$, but $\kappa \notin \Delta$. Then $Id_{(\Delta)} = Id_{\lambda_1} ; \cdots ; Id_{\lambda_n}$ by Lemma 20(13) and $Id_\kappa \neq Id_{\lambda_i}$ for all $\lambda_i \in \Delta$ by injectivity on singleton sets. Thus $Id_\kappa \not\leq Id_{(\Delta)}$, because Id_κ and the Id_{λ_i} are all atoms, and therefore $1_{(\Gamma)} \neq 1_{(\Delta)}$. □

Injectivity of $1_{(\_)}$ can therefore be assumed safely relation CKL, but other models require additional investigations. Whether this property should be turned into another CKL axiom is left for future work.

7 Propositional Refinement Calculus

Armstrong, Gomes and Struth have extended Kleene algebras with tests to refinement Kleene algebras with tests and derived the rules of a propositional variant of Morgan's refinement calculus—no frames, no local variables, no assignment laws—in this setting [2]. In the next section we show how the rules of a propositional refinement calculus with frames can be derived from the CKL axioms. Assignment laws are derived from the axioms of CKL with diagonals in Section 12. In this section we merely adapt the definition of refinement Kleene algebras with tests to our purposes.

Kleene algebra with tests captures propositional Hoare logic in a partial correctness setting. For a program $x \in K$ and tests $p, q \in B$, validity of the Hoare triple can be encoded as

$$\{p\}x\{q\} \Leftrightarrow p\,;x \leq x\,;q \Leftrightarrow p\,;x\,;\neg q = 0.$$

By the right-hand identity, the Hoare triple for precondition p, program x and postcondition q holds if it is impossible to execute x from states where p holds and, if the program terminates, end up in states where q does not hold. This intuition for partial correctness is easily backed up by the relational model.

In a refinement Kleene algebra [2], a specification statement $[p, q]$, where $p, q \in B$, is modelled as the largest program that satisfies $\{p\}(\_)\{q\}$. We adapt this definition to CKL.

Definition 24. *A* refinement cylindric Kleene lattice with tests *is a distributive* CKL *with tests expanded by an operation* $[\_,\_] : B \times B \to K$ *that satisfies*

$$p\,;x\,;\neg q = 0 \Leftrightarrow x \leq [p, q]. \tag{3}$$

It follows that $[p, q]$ satisfies $\{p\}[p, q]\{q\}$ and that it is indeed the greatest program that does so. It is also easy to check that in relation CKL,

$$[P, Q] = \bigcup\{R \subseteq X^\alpha \times X^\alpha \mid \{P\}R\{Q\}\},$$

which further confirms this programming intuition.

In addition, CKL with tests—like Kleene algebra with tests—provides an algebraic semantics of conditionals and while-loops that is consistent with the relational one.

$$\textbf{if } b \textbf{ then } x \textbf{ else } y = b\,;x + \neg b\,;y, \tag{4}$$

$$\textbf{while } b \textbf{ do } x = (b; x)^*; \neg b. \tag{5}$$

8 Variable frames

Our first application to program construction shows that CKL is expressive enough to capture the variable frames of Morgan's refinement calculus [23]. For

the sake of simplicity, we restrict our attention to a partial correctness setting. In contrast to standard notations for the refinement calculus [3,23], our lattice is the dual of the refinement lattice; the standard refinement ordering $\sqsubseteq$ is the opposite of $\leq$. Hence y is a refinement of x, denoted $x \sqsubseteq y$ if and only if $x \geq y$.

In this context, we fix a CKL with tests K. We call elements of K *programs* and finite subsets of α *frames*. A frame represents the set of variables a program may modify. The program $x \cdot 1_{(\Gamma)}$ restricts x so that it may only modify variables in Γ. Using Morgan's refinement calculus notation, we define

$$\Gamma : x = x \cdot 1_{(\Gamma)} \tag{6}$$

for a program x restricted to frame Γ. This is consistent with relation CKL, where for a relation R and variable κ,

$$\kappa : R = \{(a, b) \mid (a, b) \in R \wedge \exists c.\ a = b[\kappa \leftarrow c]\}.$$

This constrains the values of all variables other than κ to remain unchanged by R, while κ is liberated and may be modified ad libitum. The generalisation to finite sets is straightforward. The following framing laws are helpful for the derivation of the laws of Morgan's refinement calculus in Proposition 26 below. They have been verified with Isabelle.

Lemma 25. *In any* CKL*,*

1. $\Gamma : x \leq x$,
2. $\Gamma \subseteq \Delta \Rightarrow \Gamma : x \leq \Delta : x$,
3. $x \leq y \Rightarrow \Gamma : x \leq \Gamma : y$,
4. $(\Gamma : x); (\Gamma : y) \leq \Gamma : (x; y)$,
5. $(\Gamma : x)^* \leq \Gamma : (x^*)$,
6. $\Gamma : x = x$, *if* $x \leq 1$.

By Lemma 25, it is a refinement to add or restrict a frame by (1) and (2). By (3), framing is isotone with respect to refinement. Equivalently to frame isotonicity, $(\Gamma : x) + (\Gamma : y) \leq \Gamma : (x + y)$. Framing distributes over sequential composition and iteration by (4) and (5). A frame has no effect on a test by (6). The distribution over sequential composition in (4) is only a refinement because the right-hand side constrains variables outside Γ to be unchanged from the initial state to the middle state and the middle state to the final state, whereas the left-hand side only has an initial-to-final constraint.

This prepares us for the main result of this section, which adapts the refinement laws derived by Armstrong, Gomes and Struth [2] to framed specifications.

Proposition 26. *The following refinement laws are derivable in any refinement* CKL *with tests.*

1. $\Gamma : [p, p] \geq 1$,
2. $\Gamma : [p, q] \geq \Gamma : [p', q']$ *if* $p' \geq p \wedge q \geq q'$,
3. $\Gamma : [0, 1] \geq \Gamma : x$,

4. $x \geq \Gamma : [1, 0]$,
5. $\Gamma : [p, q] \geq \Gamma : [p, r] ; \Gamma : [r, q]$,
6. $\Gamma : [p, q] \geq \textbf{if } b \textbf{ then } \Gamma : [b \cdot p, q] \textbf{ else } \Gamma : [\neg b \cdot p, q]$,
7. $\Gamma : [p, \neg b \cdot p] \geq \textbf{while } b \textbf{ do } \Gamma : [b \cdot p, p]$.

We have verified this result with Isabelle relative to Armstrong, Gomes and Struth's proof. Assuming that the refinement laws obtained by deleting all occurrences of frames from (1)-(7) hold, we have shown that the corresponding laws with frames are derivable using a simple formalisation within CKL without tests and refinement statements. For (1), we have shown that $1 \leq x$ implies $\Gamma : 1 \leq \Gamma : x$, which is an instance of Lemma 25(3). Similarly, (2) and (3) are instances of frame isotonicity. For (4), we have verified that $x \leq y$ implies $\Gamma : x \leq y$, for (5) that $x ; y \leq z$ implies $\Gamma : x ; \Gamma : y \leq \Gamma : z$, for (6) that $v ; x + w ; y \leq z$ implies $v ; \Gamma : x + w ; \Gamma : y \leq \Gamma : z$ whenever $v, w \leq 1$, and for (7) that $(v ; x)^* ; w \leq y$ implies $(v ; \Gamma : x)^* ; w \leq \Gamma : y$ whenever $v, w \leq 1$. All proofs use properties from Lemma 25. None of them depends on rectangularity of generalised cylindrified units.

9 Local variable blocks

Next we show how local variable blocks can be expressed in CKL for which 1 is rectangular. Intuitively, a local variable block introduces a variable κ having as scope a program x. The definition allows for the fact that outside the local variable block κ may (or may not) be in use as a program variable. The outer κ is unmodified by the local variable block (as represented in the definition by the conjunction of $1_{(\overline{\kappa})}$) but the body of the block is free to update the local κ as it sees fit (as represented by the cylindrification $c_\kappa\, x$). We define a local variable block ($\textbf{var } \kappa.\ x$) that introduces a local variable κ with scope x as

$$\textbf{var } \kappa.\ x = (c_\kappa\, x) \cdot 1_{(\overline{\kappa})}. \tag{7}$$

It requires α to be a finite ordinal, so that the set $\overline{\kappa} = \alpha - \{\kappa\}$ is finite and hence $1_{(\overline{\kappa})}$ well defined. The following law allows a local variable κ to be introduced so that κ can be used to hold intermediate results of a computation.

Lemma 27. *Let K be a* CKL *for a finite ordinal α and in which 1 is rectangular. For all $\kappa < \alpha$ and $\Gamma \subseteq \alpha$, if $\kappa \notin \Gamma$ and $x \in K_\kappa$, that is, $c_\kappa x = x$, then*

$$\Gamma : x = \textbf{var } \kappa.\ (\kappa, \Gamma) : x.$$

Proof.

$$\begin{aligned}
\textbf{var } \kappa.\ (\kappa, \Gamma) : x &= (c_\kappa\, (x \cdot c_\kappa\, 1_{(\Gamma)}) \cdot 1_{(\overline{\kappa})} && \text{by definitions (6) and (7)} \\
&= x \cdot 1_{(\kappa,\Gamma)} \cdot 1_{(\overline{\kappa})} && \text{by (C3) and } c_\kappa x = x \\
&= x \cdot 1_{((\kappa,\Gamma) \cap \overline{\kappa})} && \text{as 1 is rectangular} \\
&= x \cdot 1_{(\Gamma)} && \text{as } \kappa \notin \Gamma,\ (\{\kappa\} \cup \Gamma) \cap \overline{\{\kappa\}} = \Gamma \\
&= \Gamma : x.
\end{aligned}$$

□

Because both cylindrification and meet are isotone so is a local variable block.

Lemma 28. *For any $\kappa < \alpha$, if $x \leq y$, then* **var** κ. $x \leq$ **var** $\kappa.y$.

Introducing a local variable in a refinement is facilitated by Morgan's law (6.1) [23]. An algebraic variant of this refinement can be derived as follows.

Lemma 29. *Let (K, B) be a* CKL *for a finite ordinal α and in which 1 is rectangular. For all $\kappa < \alpha$ and $\Gamma \subseteq \alpha$, if $\kappa \notin \Gamma$ and $p, q \in B_\kappa$, i.e. $c_\kappa p = p$ and $c_\kappa q = q$,*

$$\Gamma : [p, q] = \textbf{var}\ \kappa.\ (\kappa, \Gamma) : [p, q].$$

Proof. From Lemma 27 it suffices to show $[p, q] \in K_\kappa$ given that $p, q \in B_\kappa$, hence $c_\kappa[p, q] = [p, q]$. From (C2) it then suffices to show $c_\kappa[p, q] \leq [p, q]$.

$$\begin{aligned}
c_\kappa[p, q] \leq [p, q] &\Leftrightarrow p; c_\kappa[p, q]; \neg q = 0 && \text{by (3)}\\
&\Leftrightarrow c_\kappa p; c_\kappa[p, q]; c_\kappa \neg q = 0 && \text{as } p, q \in B_\kappa\\
&\Leftrightarrow c_\kappa(p; [p, q]; \neg q) = 0 && \text{by (C6) and (C7)}\\
&\Leftarrow p; [p, q]; \neg q = 0 && \text{by (C1)}\\
&\Leftrightarrow [p, q] \leq [p, q]. && \text{by (3)}
\end{aligned}$$

□

This law extends the refinement laws from Proposition 26 to local variable blocks.

10 Synchronous Cylindrification

Next we turn to the definition of variable assignments in CKL. This, however requires some preparation. In this section, we set up the link between CKL-style cylindrification and the classical one, which we need to apply to the tests in specification statements to model assignments. Section 11 introduces diagonal elements and substitutions as additional ingredients that are definable in CKL and needed for assignments, which are finally discussed in Section 12.

We have already emphasised in Section 5 that relational cylindrification liberates the variables in the first and second coordinates of pairs asynchronously, and this is in particular the case for subidentities, which correspond to predicates or sets. As an undesirable side effect, by Lemma 8(3), tests in CKL are not closed with respect to cylindrification: an element x of a (weak) CKL is a fixpoint of c_κ if and only if x itself has already been cylindrified by c_κ. In relational CKL, therefore, no test except $\emptyset$ is a fixpoint of any C_κ, no cylindrification of any test except $\emptyset$ is a test and $C_\kappa[\mathcal{P}Id_X] \cap \mathcal{P}Id_X = \{\emptyset\}$.

Hence if $\lceil \_ \rceil$ denotes the bijection from sets into relational subidentities, and C^c_κ denotes classical cylindrification, then $\lceil C^c_\kappa P \rceil \neq C_\kappa \lceil P \rceil$ except when predicate P is $\emptyset$.

Equality of $\lceil C^c_\kappa P \rceil$ and $C_\kappa \lceil P \rceil$ requires "synchronising" relational cylindrifications to ensure that the values of the cylindrified variable κ match in the

first and the second coordinate. Synchronised relational cylindrification can be expressed in CKL as

$$\widehat{c}_\kappa x = c_\kappa x \cdot 1,$$

so that $\widehat{C}_\kappa R = C_\kappa R \cap Id_X$ and therefore, for any set P,

$$\widehat{C}_\kappa \lceil P \rceil = \{(a, a) \mid \exists c \in X.\ (a[\kappa \leftarrow c], a[\kappa \leftarrow c]) \in \lceil P \rceil\}.$$

It is then easy to check that

$$\lceil \widehat{C}_\kappa P \rceil = \widehat{C}_\kappa \lceil P \rceil$$

for any set P and hence $\lfloor \widehat{C}_\kappa P \rfloor = C^c_\kappa \lfloor P \rfloor$ for any relational subidentity P and the inverse bijection $\lfloor \_ \rfloor$.

The definition of $\widehat{c}_\kappa$ and its relational instance $\widehat{C}_\kappa$ implies that $\widehat{C}\kappa[\mathcal{P}Id_X] \subseteq \mathcal{P}Id_X$. Thus relational subidentities are closed under $\widehat{C}_\kappa$. Yet, for a general CKL with tests $B \neq 1{\downarrow} = \{x \in K \mid x \leq 1\}$ it cannot be guaranteed that $\widehat{c}_\kappa[B] \subseteq B$. Yet we may require that $B = 1{\downarrow}$, which is consistent with relational models and many others. In fact, all applications of $\widehat{c}_\kappa$ in this article are restricted to tests that satisfy this property.

The current axiomatisation of the relationship between tests and the cylindrifications is not sufficient to prove some properties that we know to be true for the relational model. For example, for relations, we must add the following additional axiom relating the two notions of cylindrification for $p \in B$, where $B = 1{\downarrow}$:

$$c_\kappa\, p = 1_\kappa \,;\, \widehat{c}_\kappa\, p = \widehat{c}_\kappa\, p \,;\, 1_\kappa. \tag{8}$$

From this assumption, we have that test $\widehat{c}_\kappa\, p$ commutes over 1_κ for any test $p \in B$, i.e. $\widehat{c}_\kappa\, p \,;\, 1_\kappa = \widehat{c}_\kappa\, p \,;\, 1_\kappa \,;\, \widehat{c}_\kappa\, p$, giving us the following lemma, which is an important property used in Section 12 to derive properties of assignment statements.

Lemma 30. *If $p = \widehat{c}_{(\Gamma)} p$ then, $\Gamma : [p \cdot q, r] = \Gamma : [p \cdot q, p \cdot r]$.*

Proof. Refinement from left to right follows from Proposition 26(2). For the reverse direction we begin the proof by expanding using the definition of a frame.

$$\begin{array}{lll}
 & [p \cdot q,\ r] \cdot 1_{(\Gamma)} \leq [p \cdot q,\ p \cdot r] \cdot 1_{(\Gamma)} & \\
\Leftrightarrow & [p \cdot q,\ r] \cdot 1_{(\Gamma)} \leq [p \cdot q,\ p \cdot r] & \text{property of meet} \\
\Leftrightarrow & (p \cdot q); [p \cdot q,\ r] \cdot 1_{(\Gamma)}; (\neg p + \neg r) = 0 & \text{by (3) and De Morgan} \\
\Leftarrow & (p; 1_{(\Gamma)}; \neg p = 0) \wedge ((p \cdot q); [p \cdot q,\ r]; \neg r = 0) & \text{distributing and simplifying} \\
\Leftrightarrow & \widehat{c}_{(\Gamma)} p; 1_{(\Gamma)}; \neg p = 0 & \text{assumption } p = \widehat{c}_{(\Gamma)} p \text{ and (3)} \\
\Leftrightarrow & \widehat{c}_{(\Gamma)} p; 1_{(\Gamma)}; \widehat{c}_{(\Gamma)} p; \neg p = 0 & \text{commutativity assumption} \\
\Leftrightarrow & p; 1_{(\Gamma)}; 0 = 0 & \text{assumption and } p; \neg p = 0
\end{array}$$

The later holds because 0 is an annihilator for tests.

Lemma 31. *If* $p = \widehat{c}_{(\Gamma)}p$ *and* $p \cdot r_2 \leq r_1$ *then,* $\Gamma\!:\![p \cdot q, r_1] \geq \Gamma\!:\![p \cdot q, r_2]$.

Proof.

$$\begin{aligned} \Gamma\!:\![p \cdot q, r_2] &= \Gamma\!:\![p \cdot q, p \cdot r_2] && \text{by Lemma 30} \\ &\leq \Gamma\!:\![p \cdot q, r_1]. && \text{by Proposition 26(2)} \end{aligned}$$

□

11 Diagonals and Substitution

Our next step toward modelling assignments algebraically requires capturing substitutions algebraically. Once again, Henkin Monk and Tarski have paved the way for us [17, §1.5]. Yet their concept of variable substitution in classical cylindric algebra depends on another concept, which is integral to their approach, and we have so far neglected: that of diagonal elements, which abstract equality in equational logic.

In standard cylindric set algebras, *diagonal elements* [17] are defined, for each $\kappa, \lambda < \alpha$, as

$$D_{\kappa\lambda} = \{x \in X^\alpha \mid x_\kappa = x_\lambda\}.$$

Henkin, Monk and Tarski [17] give a geometric interpretation of $D_{\kappa\lambda}$ as a hyperplane in X^α that is described by the equation $x_\kappa = x_\lambda$. For instance, for $\alpha = 2$, D_{01} corresponds to the diagonal line between the coordinate axes 0 and 1; for $\alpha = 3$, D_{01} is the plane spanned by that diagonal and 3-axis.

While diagonalisation could be generalised to relational diagonalisation, we only require diagonal elements on the boolean subalgebra of tests, which is captured by the standard approach, in combination with synchronised cylindrification $\widehat{c}_\kappa$. Henkin, Monk and Tarski's axioms for classical cylindric algebra therefore lead us to the following definition.

Definition 32. *A* cylindric Kleene lattice with enriched tests *is a* CKL *equipped with a family of elements* $(d_{\kappa\lambda})_{\kappa,\lambda<\alpha} \subseteq B = 1{\downarrow}$ *that satisfy*

$$d_{\kappa\kappa} = 1, \tag{D1}$$

$$d_{\lambda\mu} = \widehat{c}_\kappa(d_{\lambda\kappa} \cdot d_{\kappa\mu}), \qquad \text{if } \kappa \notin \{\lambda, \mu\}, \tag{D2}$$

$$\widehat{c}_\kappa(d_{\kappa\lambda} \cdot p) \cdot \widehat{c}_\kappa(d_{\kappa\lambda} \cdot \neg p) = 0, \qquad \text{if } \kappa \neq \lambda. \tag{D3}$$

The axioms (D1)-(D3) are precisely the diagonal axioms of classical cylindric algebras [17]. They are applied to tests only and use $\widehat{c}_\kappa$ instead of c_κ. Axiom (D3) captures a notion of variable substitution. In fact, Henkin, Monk and Tarski define

$$s^\kappa_\lambda\, p = \begin{cases} p, & \text{if } \kappa = \lambda, \\ \widehat{c}_\kappa(d_{\kappa\lambda} \cdot p), & \text{if } \kappa \neq \lambda. \end{cases} \tag{9}$$

to indicate that λ is substituted for κ in p. Axiom (D3) can then be rewritten as $s^\kappa_\lambda\, p \cdot s^\kappa_\lambda\, \neg p = 0$. The substitution operator s^κ_λ satisfies the following properties, which have been verified with Isabelle, and turn out to be useful in the following sections.

Lemma 33. *Let (K, B) be a* CKL *with enriched tests. If $p, q \in B$ and $\kappa, \lambda, \mu < \alpha$, then*

1. $s^\kappa_\lambda\,(p+q) = s^\kappa_\lambda\, p + s^\kappa_\lambda\, q$, *(HMT1.5.3(i))*
2. $\neg s^\kappa_\lambda\, p = s^\kappa_\lambda\, \neg p$, *(HMT1.5.3(ii))*
3. $s^\kappa_\lambda\, 1 = 1$,
4. $s^\kappa_\lambda\,(p \cdot q) = s^\kappa_\lambda\, p \cdot s^\kappa_\lambda\, q$,
5. $s^\kappa_\lambda\,(d_{\kappa\mu}) = d_{\lambda\mu}$ *if* $\kappa \neq \mu$, *(HMT1.5.4(i))*
6. $s^\kappa_\lambda\,(d_{\mu\nu}) = d_{\mu\nu}$ *if* $\kappa \notin \{\mu, \nu\}$, *(HMT1.5.4(ii))*
7. $s^\kappa_\tau\,(d_{\kappa\lambda} \cdot d_{\mu\nu}) = d_{\tau\lambda} \cdot d_{\mu\nu}$ *for distinct* $\kappa, \lambda, \mu, \nu, \tau$.

12 Assignments

Assignment statements are usually of the form $\kappa := e$, where e is an expression on the programming variables. Expressions are not available in CKL with enriched tests, however we can use framed specification statements to abstract the behaviour of assignments. For any $p \in B$ we write $\kappa :\in p$ to denote a non-deterministic assignment of variable κ to a value such that the final state of the command satisfies test p. It is defined as

$$\kappa :\in p = \kappa : [1, p].$$

A special case of this is the direct assignment of one variable to another, written $\kappa := \lambda$, which is defined by taking predicate p to be the diagonal $d_{\kappa\lambda}$:

$$\kappa := \lambda = \kappa : [1, d_{\kappa\lambda}]$$

For example, if κ is fresh in expression e, the assignment $\kappa := e$ can be encoded using the non-deterministic assignment command as $\kappa :\in (\kappa = e)$, where $(\kappa = e)$ is abstracted to a test in the algebra. For the more general case we can choose a variable λ that is fresh in e and write

$$(\textbf{var}\ \lambda.\ \lambda := \kappa\ ;\ \kappa :\in (\kappa = e[\kappa \backslash \lambda]))$$

where, in the program model, $e[\kappa\backslash\lambda]$ is the expression e with λ substituted for κ, but in the algebra $(\kappa = e[\kappa\backslash\lambda])$ is simply abstracted as a test.

The following propositions are used to verify the algebraic equivalent of the assignment law defined by Morgan [23, p.8]. In order to more simply represent the precondition, we introduce two notations on tests: the *inner cylindrification* $c^\partial_\kappa p$ is the De Morgan dual of $\widehat{c}_\kappa$ and corresponds to universal quantification in first order logic [17, §1.4]; and $p \to q$ is a shorthand for implication in the boolean algebra of tests.

$$c^\partial_\kappa p = \neg \widehat{c}_\kappa \neg p, \tag{10}$$

$$p \to q = \neg p + q. \tag{11}$$

In the proposition below the test $c^\partial_\kappa (r \to q)$ can be interpreted as saying that for all values of κ, test r implies q, i.e. it is a test describing the states from which substituting κ for any value satisfying r will certainly result in a post-state q.

Proposition 34. *Suppose B is the test subalgebra of a* CKL *with enriched tests. If $q, r \in B$ and $\kappa < \alpha$ and $p \leq c^\partial_\kappa(r \to q)$,*

$$\kappa{:}[p,\ q] \geq \kappa{:}\in r. \tag{12}$$

Proof. The application of Lemma 31 requires $c^\partial_\kappa(r \to q) \cdot r \leq q$, which can be shown as follows.

$$\begin{array}{lll}
 & \widehat{c}_\kappa(r \cdot \neg q) \geq r \cdot \neg q & \text{by (C2)} \\
\Leftrightarrow & \neg\widehat{c}_\kappa(r \cdot \neg q) \leq \neg(r \cdot \neg q) & \text{negating both sides and reversing the order} \\
\Rightarrow & c^\partial_\kappa(r \to q) \cdot r \leq (\neg r + q) \cdot r & \text{by definition of } c^\partial \text{ and conjoin } r \text{ to both sides} \\
\Leftrightarrow & c^\partial_\kappa(r \to q) \cdot r \leq q \cdot r & \text{boolean simplification} \\
\Rightarrow & c^\partial_\kappa(r \to q) \cdot r \leq q & \text{as } q \cdot r \leq q.
\end{array}$$

It also requires that $\widehat{c}_{(\Gamma)} c^\partial_{(\Gamma)} p = c^\partial_{(\Gamma)} p$, which has been shown in [17, Theorem 1.4.4(ii)].

$$\begin{array}{lll}
\kappa{:}[p,\ q] \geq \kappa{:}[c^\partial_\kappa(r \to q),\ q] & & \text{by Lemma 26(2)} \\
\geq \kappa{:}[c^\partial_\kappa(r \to q),\ r] & & \text{by Lemma 31 as } c^\partial_\kappa(r \to q) \cdot r \leq q \\
\geq \kappa{:}[1,\ r] & & \text{by Lemma 26(2)} \\
= \kappa{:}\in r & & \text{From (12).}
\end{array}$$

□

When r is $d_{\kappa\lambda}$, the test $c^\partial_\kappa(r \to q)$ simplifies to $s^\kappa_\lambda\, q$, the substitution of λ for κ in test q.

Proposition 35. *Suppose B is the test subalgebra of a* CKL *with enriched tests. If $q \in B$ and $\kappa, \lambda < \alpha$ and $p \leq s^\kappa_\lambda\, q$,*

$$\kappa{:}[p,\ q] \geq \kappa := \lambda.$$

Proof. We have $c^\partial_\kappa(d_{\kappa\lambda} \to q) = \neg c_\kappa(d_{\kappa\lambda} \cdot \neg q) = \neg s^\kappa_\lambda \neg q = s^\kappa_\lambda\, q$ by Lemma 33(2).

$$\begin{array}{lll}
\kappa{:}[p,\ q] \geq \kappa{:}[s^\kappa_\lambda\, q,\ q] & & \text{by Proposition 26(2)} \\
= \kappa{:}[c^\partial_\kappa(d_{\kappa\lambda} \to q),\ q] & & \text{by above reasoning} \\
\geq \kappa{:}\in d_{\kappa\lambda} & & \text{taking } r \text{ to be } d_{\kappa\lambda} \text{ in Proposition 34} \\
= \kappa := \lambda & & \text{by definition (12).}
\end{array}$$

Proposition 34 and Proposition 35 encode the assignment law defined by Morgan [23, p.8] for our non-deterministic assignment statement, and for the special case where we assign one variable directly to another. These propositions can be equivalently expressed using Hoare logic using the specification statement definition (3) and the Hoare logic encoding.

Proposition 36. *Suppose B is the test subalgebra of a* CKL *with enriched tests. If $p \in B$ and $\kappa, \lambda < \alpha$ then*

$$\{p\}\ \kappa :\in r\ \{q\}, \qquad \text{if } p \leq c^{\partial}_{\kappa}(r \to q),$$
$$\{p\}\ \kappa := \lambda\ \{q\}, \qquad \text{if } p \leq s^{\kappa}_{\lambda}\, q.$$

Frames and diagonals together allow one to make use of *logical variables* and constants (e.g., natural numbers) within a specification. In Example 38, we consider a derivation of a program that swaps the values of at indices λ and κ. This example is given by Morgan [23]; the difference here is that the derivation is purely algebraic. The example uses Morgan's *following assignment* law, in which a specification statement is refined to another specification statement followed by an assignment command. The next lemma derives this in the algebra.

Lemma 37. *Suppose B is the test subalgebra of a* CKL *with enriched tests. If $p, q \in B$ and $\kappa, \lambda, \mu < \alpha$, then*

$$\lambda, \kappa : [p, q] \geq \lambda, \kappa : [p, s^{\kappa}_{\mu}\, q]; \kappa := \mu.$$

Proof.

$$\begin{aligned}
\lambda, \kappa : [p, q] &\geq \lambda, \kappa : [p, s^{\kappa}_{\mu}\, q]; \lambda, \kappa : [s^{\kappa}_{\mu}\, q, q] && \text{by Proposition 26(5)}\\
&\geq \lambda, \kappa : [p, s^{\kappa}_{\mu}\, q]; \kappa : [s^{\kappa}_{\mu}\, q, q] && \text{by Lemma 25(2)}\\
&\geq \lambda, \kappa : [p, s^{\kappa}_{\mu}\, q]; \kappa := \mu && \text{by Proposition 35.}
\end{aligned}$$

□

Example 38. The swapping variables example can be handled entirely within the algebra. Suppose $\kappa_1, \kappa_2, \lambda_1, \lambda_2, \tau < \alpha$ are distinct. As in many refinement proofs (see [23]), λ_1 and λ_2 are logical indices used to specify the initial values of program variables κ_1 and κ_2. The first step uses Lemma 27 to introduce local variable τ that we use to temporarily store the value of κ_2:

$$\begin{aligned}
&\kappa_1, \kappa_2 : [d_{\kappa_1\lambda_1} \cdot d_{\kappa_2\lambda_2}, d_{\kappa_1\lambda_2} \cdot d_{\kappa_2\lambda_1}]\\
&= \mathbf{var}\ \tau.\ \tau, \kappa_1, \kappa_2 : [d_{\kappa_1\lambda_1} \cdot d_{\kappa_2\lambda_2}, d_{\kappa_1\lambda_2} \cdot d_{\kappa_2\lambda_1}].
\end{aligned}$$

Using Lemma 28 this can be refined by refining the body of the local variable block as follows.

$$\begin{aligned}
&\tau, \kappa_1, \kappa_2 : [d_{\kappa_1\lambda_1} \cdot d_{\kappa_2\lambda_2}, d_{\kappa_1\lambda_2} \cdot d_{\kappa_2\lambda_1}]\\
&\geq \tau, \kappa_1, \kappa_2 : [d_{\kappa_1\lambda_1} \cdot d_{\kappa_2\lambda_2}, s^{\kappa_1}_{\tau}\, (d_{\kappa_1\lambda_2} \cdot d_{\kappa_2\lambda_1})]; \kappa_1 := \tau && \text{by Lemma 37}\\
&\geq \tau, \kappa_1, \kappa_2 : [d_{\kappa_1\lambda_1} \cdot d_{\kappa_2\lambda_2}, d_{\tau\lambda_2} \cdot d_{\kappa_2\lambda_1}]; \kappa_1 := \tau && \text{by Lemma 33(7).}
\end{aligned}$$

Applying this pattern twice yields

$$\begin{aligned}
&\tau, \kappa_1, \kappa_2 : [d_{\kappa_1\lambda_1} \cdot d_{\kappa_2\lambda_2}, d_{\tau\lambda_2} \cdot d_{\kappa_2\lambda_1}]; \kappa_1 := \tau\\
&\geq \tau, \kappa_1, \kappa_2 : [d_{\kappa_1\lambda_1} \cdot d_{\kappa_2\lambda_2}, d_{\tau\lambda_2} \cdot d_{\kappa_1\lambda_1}]; \kappa_2 := \kappa_1; \kappa_1 := \tau\\
&\geq \tau, \kappa_1, \kappa_2 : [d_{\kappa_1\lambda_1} \cdot d_{\kappa_2\lambda_2}, d_{\kappa_2\lambda_2} \cdot d_{\kappa_1\lambda_1}]; \tau := \kappa_2; \kappa_2 := \kappa_1; \kappa_1 := \tau\\
&\geq 1; \tau := \kappa_2; \kappa_2 := \kappa_1; \kappa_1 := \tau && \text{by Proposition 26(1).}
\end{aligned}$$

Eliminating the identity 1 and substituting the refined body back in the local variable block, the final code is

$$\textbf{var}\ \tau.\ (\tau := \kappa_2; \kappa_2 := \kappa_1; \kappa_1 := \tau).$$

13 Beyond Cylindrification: Liberation Algebras

An interesting axiomatic question arises from the fact that, by Lemmas 16 and 49, the identity

$$c_\kappa\, x = 1_\kappa\, ;\, x\, ;\, 1_\kappa$$

holds in (weak) relational CKL, whereas, by Lemma 11, it is not derivable in CKL. On the one hand, non-derivability is desirable, because the identity fails in program trace models of CKL [21]. On the other hand, it shifts the focus from cylindrification to identities 1_κ and raises the question of directly axiomatising elements 1_κ for $\kappa < \alpha$ directly over (weak) Kleene lattices and defining the cylindrification operators c_κ explicitly via the identity above. This section describes the initial steps for such an approach. The elements 1_κ are now written more simply as κ for $\kappa < \alpha$.

Definition 39 (LLM). *A* (weak) liberation l-monoid *is a (weak) l-monoid L that is equipped with a family $(\kappa)_{\kappa<\alpha}$ of elements that satisfy*

$$\kappa\, ;\, 0 = 0, \tag{L1}$$
$$1 \leq \kappa, \tag{L2}$$
$$\kappa\, ;\, (x \cdot (\kappa\, ;\, y)) = (\kappa\, ;\, x) \cdot (\kappa\, ;\, y), \tag{L3}$$
$$(x \cdot (y\, ;\, \kappa))\, ;\, \kappa = (x\, ;\, \kappa) \cdot (y\, ;\, \kappa), \tag{L4}$$
$$\kappa\, ;\, \lambda = \lambda\, ;\, \kappa, \tag{L5}$$
$$\kappa \neq \lambda\ \Rightarrow\ \kappa \cdot \lambda = 1, \tag{L6}$$
$$(\kappa\, ;\, \lambda) \cdot (\kappa\, ;\, \mu) = \kappa\, ;\, (\lambda \cdot \mu), \tag{L7}$$
$$(\kappa\, ;\, \mu) \cdot (\lambda\, ;\, \mu) = (\kappa \cdot \lambda)\, ;\, \mu. \tag{L8}$$

As expected, there is a close correspondence between these axioms and axioms (C1)-(C10), although analogues of (C5)-(C7) and (C10) are derivable in this context, and therefore redundant.

Definition 40 (LKL). *A* (weak) liberation Kleene lattice *is a (weak) Kleene lattice with a family $(\kappa)_{\kappa<\alpha}$ of elements that satisfy (L1)-(L8).*

We have checked independence of these axioms in Isabelle. Extensions to (weak) liberation Kleene lattices with tests are straightforward.

Proposition 41. *Every weak* LLM *is a weak* CLM *with $c_\kappa x = \kappa\, ;\, x\, ;\, \kappa$.*

Rewriting the CLM axioms with $c_\kappa x = \kappa\,;x\,;\kappa$ and deriving the results from the LLM axioms is straightforward with Isabelle. Axiom (C6), for instance, becomes $\kappa\,;x\,;\kappa\,;y\,;\kappa\,;\kappa = \kappa\,;x\,;\kappa\,;\kappa\,;y\,;\kappa$, which is derivable because any κ can be shown to be an idempotent with respect to ; in LLM by taking x and y to both be 1 in (L3). Axiom (C3) becomes $\kappa\,;(x \cdot (\kappa\,;y\,;\kappa))\,;\kappa = (\kappa\,;x\,;\kappa) \cdot (\kappa\,;y\,;\kappa)$, which can be obtained from (L3) and (L4).

Unlike for CKL, a special star axiom is not needed for liberation algebras. The following lemma has been obtained with Isabelle.

Lemma 42. *In every weak* LKL*,*

1. $x^+\,;\kappa \leq (x\,;\kappa)^+$,
2. $\kappa\,;x^+ \leq (\kappa\,;x)^+$,
3. $\kappa\,;x^+\,;\kappa \leq (\kappa\,;x\,;\kappa)^+$.

The proof of (1) is very simple: $x^+\,;\kappa = x^*\,;x\,;\kappa \leq (x\,;\kappa)^*\,;x\,;\kappa = (x\,;\kappa)^+$. Using the last identity then yields the following result.

Proposition 43. *Every weak* LKL *is a weak* CKL *with* $c_\kappa x = \kappa\,;x\,;\kappa$.

In addition, the LKL axioms are sound with respect to relational models.

Proposition 44. *The (weak)* LKL *axioms hold in the relational (fault) model with* κ *interpreted as* Id_κ *for all* $\kappa < \alpha$.

Proof. The relational variants of the LKL axioms have been verified with Isabelle. An algebraic proof for the weak case is given in Proposition 50, Appendix B. It has been checked with Isabelle. □

The next two facts generalise Lemma 9 and Proposition 15 from Section 4.

Lemma 45. *In every weak* LLM*,* $\kappa\,;(\_)$ *and* $(\_)\,;\kappa$ *are closure operators.*

Writing K^l_κ for the set of fixpoints of $\kappa\,;(\_)$, K^r_κ for those of $(\_)\,;\kappa$ and K_κ for those of $\kappa\,;(\_)\,;\kappa$ yields the following result.

Proposition 46. *Let* K *be a (weak)* LKL *and let* $\kappa < \alpha$*. Then*

1. $(K^l_\kappa, +, \cdot, ;, 0, \kappa, (\_)^+)$ *is a (weak) sub-Kleene lattice of* K *with left unit* κ*,*
2. $(K^r_\kappa, +, \cdot, ;, 0, \kappa, (\_)^+)$ *is a sub-Kleene lattice of* L *with right unit* κ *and if* K *is a strong* LKL*, then* K^l_κ *and* K^r_κ *are isomorphic,*
3. $(K_\kappa, +, \cdot, ;, 0, \kappa, (\_)^+)$ *is a sub-Kleene lattice of* K^l_κ *and* K^r_κ.

The proofs are very similar to those for c_κ.

The results for $\kappa\,;(\_)$ and $(\_)\,;\kappa$ reveal a duality in relational cylindrification without faults that is not present in the traditional approach. We already pointed out in Section 5 that, in the relational model, $Id_\kappa\,;R$ corresponds to a left-handed cylidrification of R and $R\,;Id_\kappa$ to its right-handed opposite. One can therefore introduce handedness via opposition to cylindrification over l-monoids by axiomatising left-handed cylindrification c^l_κ and right-handed cylindrification

c^r_κ and split the axioms (C6) and (C7) accordingly. This yields a more fine-grained view on cylindrification in models with opposition duality. In addition, left-handed and right-handed cylindrifications commute (i.e. $c^l_\kappa \circ c^r_\lambda = c^r_\lambda \circ c^l_\kappa$) and $c_\kappa = c^l_\kappa \circ c^r_\kappa$ holds in the relational model but not in general. Details have been worked out in a companion article [21]. The handed cylindrifications are akin to forward and backward modal operators, yet defined over Kleene lattices instead of boolean algebras.

14 Conclusion

We have shown that cylindrification can be adapted to Kleene lattices and their relational models in such a way that variable assignments, frames and local variable blocks can be modelled. Based on this, we have derived the laws of Morgan's refinement calculus and the rules of Hoare logic, including those for assignments. The scope of algebraic approaches to program construction has therefore been extended, with the potential of fully algebraic reasoning about imperative programs.

Nevertheless, many questions about cylindric Kleene lattices and their relatives remain open and deserve further investigation. Instead of the obvious questions on completeness or decidability, we focus on conceptual ones.

First, it is easy to check that relational cylindrifications preserves arbitrary sups and hence have upper adjoints. This situation is well known from classical cylindric algebra, where the standard *outer cylindrifications* c_κ are accompanied by *inner cylindrifications* c^∂_κ that are related by De Morgan duality. Geometrically, these describe greatest cylinders with respect to κ that are contained in a given set. In cylindric algebras of formulas, inner cylindrification gives the algebra of universal quantification. In an extension of CKL, where lattices need not be complemented, dual cylindrifications can be axiomatised by adjunction. In extensions of LKL, they can be defined explicitly as $c^\partial_\kappa\ x = 1_\kappa \backslash x / 1_\kappa$, where $\backslash$ and / are residuals, as they appear in action algebras [18] and action logic [26]. Our Isabelle components already contain axiomatisations for these structures, but so far we do not have any use for them.

Second, our refinement calculus and Hoare logic are restricted to partial program correctness for the sake of simplicity; yet the relational fault model is relevant to total correctness and our Isabelle components are based on weak cylindric Conway lattices and weak liberation Conway lattices, in which iteration is weak enough to be either finite, as in Kleene lattices, or possibly infinite, as in demonic refinement algebra [29]. Almost all properties presented in our paper hold in fact in this more general setting, and our relational models are a fortiori models of these generalisations. The relevance of these algebras to models with finite or possibly infinite traces, and the derivation of while rules and refinement laws for total program correctness remain to be explored.

For concurrent programs with a semantics based on a set of traces, cylindrification can be applied to liberate a variable κ in every state of each trace, in the same way that liberation of a relation liberates κ in both the initial and final

states. In that setting liberation can be used in the definition of a local variable block in a similar fashion to the way it is used here [21]. A trace-based semantics for the liberation operator was given in [5, §4.6 and §5.6].[9] The generalisation of cylindric algebra presented in this paper applies directly to the trace-based model used for concurrency. That model also uses sets, binary relations, a subset of commands that form instantaneous tests (isomorphic to sets of states), subsets of commands representing program steps and environment steps (each of which is isomorphic to binary relations on states). Factoring out the cylindric algebra and applying it in each of these contexts allows one to reuse the properties of cylindric algebra in each of these contexts, thus simplifying the mechanisation of the theory.

Finally, while part of the theory and many of the proofs in this article have been formalised with Isabelle/HOL, the question whether our approach may lead to program construction and verification components that support an algebraic treatment of variable assignments requires further exploration. This seems a particularly promising avenue for future research.

Acknowledgements. We thank Simon Doherty for discussions on earlier versions of this work.

References

1. H. Andréka, S. Mikulás, and I. Németi. The equational theory of Kleene lattices. *Theoretical Computer Science*, 412(52):7099–7108, 2011.
2. A. Armstrong, V. B. F. Gomes, and G. Struth. Building program construction and verification tools from algebraic principles. *Formal Aspects of Computing*, 28(2):265–293, 2016.
3. R.-J. Back and J. von Wright. *Refinement calculus - a systematic introduction.* Springer, 1999.
4. G. Birkhoff. *Lattice Theory.* American Mathematical Society, 1940.
5. R. J. Colvin, I. J. Hayes, and L. A. Meinicke. Designing a semantic model for a wide-spectrum language with concurrency. *Formal Aspects of Computing*, 29:853–875, 2016.
6. J. Cranch, M. R. Laurence, and G. Struth. Completeness results for omega-regular algebras. *J Logical and Algebric Methods in Programming*, 84(3):402–425, 2015.
7. J. Desharnais, B. Möller, and G. Struth. Kleene algebra with domain. *ACM TOCL*, 7(4):798–833, 2006.
8. J. Desharnais and G. Struth. Internal axioms for domain semirings. *Science of Computer Programming*, 76(3):181–203, 2011.
9. The Coq development team. *The Coq proof assistant reference manual.* LogiCal Project, 2004. Version 8.0.
10. B. Dongol, V. F. B. Gomes, I. J. Hayes, and G. Struth. Partial semigroups and convolution algebras. *Archive of Formal Proofs*, 2017, 2017.
11. B. Dongol, I. J. Hayes, and G. Struth. Relational convolution, generalised modalities and incidence algebras. *CoRR*, abs/1702.04603, 2017.

[9] In that paper $c_\kappa x$ is written $x\backslash\kappa$.

12. T. Ehm, B. Möller, and G. Struth. Kleene modules. In R. Berghammer, B. Möller, and G. Struth, editors, *RAMiCS 2003*, volume 3051 of *LNCS*, pages 112–124. Springer, 2004.
13. M. J. Gabbay and V. Ciancia. Freshness and name-restriction in sets of traces with names. In M. Hofmann, editor, *FOSSACS 2011*, volume 6604 of *LNCS*, pages 365–380. Springer, 2011.
14. R. Giacobazzi, S. K. Debray, and G. Levi. A generalized semantics for constraint logic programs. In *FGCS*, pages 581–591, 1992.
15. V. B. F. Gomes and G. Struth. Modal Kleene algebra applied to program correctness. In J. S. Fitzgerald, C. L. Heitmeyer, S. Gnesi, and A. Philippou, editors, *FM 2016*, volume 9995 of *LNCS*, pages 310–325, 2016.
16. Ian Hayes, editor. *Specification Case Studies*. Prentice Hall International, second edition, 1993.
17. Leon Henkin, James Donald Monk, and Alfred Tarski. *Cylindric Algebras, Part I*, volume 64 of *Studies in logic and the foundations of mathematics*. North-Holland Pub. Co., 1971.
18. D. Kozen. On action algebras. In J. van Eijk and A. Visser, editors, *Logic and Information Flow*, pages 78–88. MIT Press, 1994.
19. D. Kozen. Kleene algebra with tests. *ACM Trans. Program. Lang. Syst.*, 19(3):427–443, 1997.
20. D. Kozen. On Hoare logic and Kleene algebra with tests. *ACM Trans. Comput. Log.*, 1(1):60–76, 2000.
21. L. A. Meinicke and I. J. Hayes. Handling localisation in rely/guarantee concurrency: An algebraic approach. arXiv:?? [cs.LO], 2019.
22. B. Möller and G. Struth. wp is wlp. In W. MacCaull, M. Winter, and I. Düntsch, editors, *RelMiCS/AKA 2005*, volume 3929 of *LNCS*, pages 200–211. Springer, 2006.
23. C. Morgan. *Programming from Specifications*. Prentice-Hall, 1990.
24. C. C. Morgan. *Programming from Specifications*. Prentice Hall, second edition, 1994.
25. D. Pous. Kleene algebra with tests and Coq tools for while programs. In S. Blazy, C. Paulin-Mohring, and D. Pichardie, editors, *ITP 2013*, volume 7998 of *LNCS*, pages 180–196. Springer, 2013.
26. V. A. Pratt. Action logic and pure induction. In J. van Eijck, editor, *JELIA '90*, volume 478 of *LNCS*, pages 97–120. Springer, 1991.
27. J.M. Spivey. *The Z Notation: A Reference Manual*. Prentice Hall International, second edition, 1992.
28. G. Struth. Hoare semigroups. *Mathematical Structures in Computer Science*, 28(6):775–799, 2018.
29. J. von Wright. Towards a refinement algebra. *Science of Computer Programming*, 51(1-2):23–45, 2004.
30. C. Wells. Some applications of the wreath product construction. *The American Mathematical Monthly*, 83(5):317–338, 1976.

A Construction of Weak Kleene Lattices

Instead of proving Proposition 6, we show that it is a corollary to a standard semidirect product construction, which is well known from semigroup theory. All proofs in this appendix have been verified with Isabelle.

An *l-monoid module* of an l-monoid L and a semilattice S with least element 0 is an action $\circ : L \to S \to S$ that satisfies

$$\begin{aligned}(p\,;q)\circ x &= p\circ(q\circ x),\\ (p+q)\circ x &= p\circ x + q\circ x,\\ p\circ(x+y) &= p\circ x + p\circ y,\\ 1\circ x &= x,\\ 0\circ x &= 0.\end{aligned}$$

It follows that $p\circ 0 = 0$.

The *semidirect product* $L \ltimes S$ on $L \times S$ is defined by

$$(p,x)\ltimes(q,y) = (p\,;q, x + p\circ y).$$

The relational redefinition of composition in Section 3 is a simple instance of this standard algebraic concept. It is easy to check that $(1,0)$ is the unit of $\ltimes$ and $(0,0)$ a left annihilator. In addition, we define join and meet pointwise on pairs as $(p,x)+(q,y) = (p+q, x+y)$ and $(p,x)\cdot(q,y) = (p\cdot q, x\cdot y)$. The following fact is routine. Most axioms have already been checked elsewhere [6,11].

Proposition 47. *Let L be an l-monoid and S a semilattice with 0. Then $L \ltimes S$ forms a weak l-monoid.*

If K is a Kleene lattice, we define a *Kleene lattice module* by adding the axiom

$$x + p\circ y \leq y \Rightarrow p^*\circ x \leq y.$$

Hence the action axiom for Kleene lattice modules are essentially those for Kleene modules [12]. Finally, we define the star on products as

$$(p,x)^* = (p^*, p^*\circ x).$$

It follows that $(p,x)^+ = (p^+, p^*\circ x)$.

Proposition 47 then extends as follows.

Theorem 48. *Let K be a Kleene lattice and S a semilattice with 0. Then $K \ltimes S$ forms a weak Kleene lattice.*

Dongol, Hayes and Struth [11] present a similar result in the less general setting of quantale modules, which however captures the relational fault model in Section 3. A formalisation with Isabelle can be found in the Archive of Formal Proofs [10], including a verification of the properties of the relational star presented in Section 3. Cranch, Laurence and Struth [6] present a second proof in the more general setting of regular algebras that satisfy strictly weaker induction axioms. It gives a good impression of the manipulations needed in our present proof. Möller and Struth [22] present a third proof for total correctness in the setting of modal Kleene algebras. Instead of semidirect products, it is based on wreath products (cf. [30]).

B Construction of Weak Liberation Kleene Lattices

Instead of proving Proposition 44 for relational cylindrification we give an algebraic proof based on a new algebraic definition. This proof also supports an indirect proof of Proposition 6. All proofs in this appendix have once again been checked with Isabelle.

A *cylindric Kleene lattice module* is a Kleene lattice module over a cylindric Kleene lattice with cylindrification defined by

$$\widetilde{c}_\kappa\,(p,x) = (c_\kappa\,p, 1_\kappa \circ x).$$

By this definition, $\widetilde{c}_\kappa\,(1,0) = (1_\kappa, 0)$ and $(p,x) \ltimes (1_\kappa, 0) = (p\,;1_\kappa, x)$.

First we derive an algebraic variant of Lemma 16 that is suitable for the relational fault model.

Lemma 49. *Let L be a* CKL *and S a semilattice with* 0*. Then*

$$c_\kappa\,p = 1_\kappa \circ p \circ 1_\kappa \;\Rightarrow\; \widetilde{c}_\kappa\,(p,x) = \widetilde{c}_\kappa\,(1,0) \ltimes (p,x) \ltimes \widetilde{c}_\kappa\,(1,0).$$

Next we turn to the algebraic proof that subsumes Proposition 44.

A *Liberation Kleene lattice module* is a Kleene lattice module defined over a liberation Kleene lattice.

Proposition 50. *Let K be a* LKL *and S a semilattice with* 0*, such that*

$$1_\kappa \circ (x \cdot (1_\kappa \circ y)) = (1_\kappa \circ x) \cdot (1_\kappa \circ y)$$

holds for all $x, y \in S$. Then $K \ltimes S$ forms a weak LKL*.*