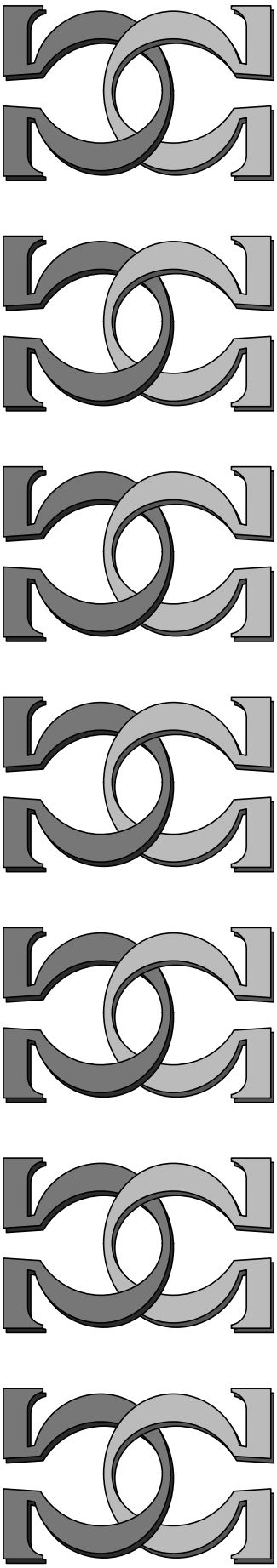


CDMTCS
Research
Report
Series

Computable Isomorphism of Boolean Algebras with Operators

Bakhadyr Khoussainov
Department of Computer Science
University of Auckland
Auckland, New Zealand

Tomasz Kowalski
JAIST, Japan

CDMTCS-181
March 2002

Centre for Discrete Mathematics and
Theoretical Computer Science

Bakhadyr Khoussainov, Tomasz Kowalski

Computable isomorphisms of Boolean Algebras with Operators

1. Introduction

One of the central topics in computable algebra and model theory is concerned with the study of computable isomorphisms. Classically, we do not distinguish between isomorphic structures, however, from computability point of view, isomorphic structures can differ quite dramatically. A typical example is provided by the linear order of type ω. It has two computable copies such that in one the successor function is computable, but it is not computable in the other. These are clearly classically isomorphic, but they are not computably isomorphic.

Computable isomorphisms of structures have been studied intensely for at least three decades. A number of natural structures arising in universal algebra, such as Boolean algebras, vector spaces, and Abelian groups have been considered. In the present paper we continue this line of research and study computable isomorphisms of Boolean algebras with operators (BAOs). Our interest in these particular structures is twofold. On the one hand, BAOs are frequently encountered in universal algebra, not least in their connection to modal logic. On the other, there is a considerable body of results about computable isomorphisms of Boolean algebras, so one may wonder how expanding the language of Boolean algebras affects computable isomorphisms. Similar investigations have been carried out quite recently in connection with computable isomorphisms of Abelian groups and ordered Abelian groups. It has been shown in [5] that the number of computable isomorphism types of any computable linearly ordered Abelian group is either 1 or ω, a fact true also in the class of computable Abelian groups.

Here is an overview of the paper. In the next two sections we introduce basic concepts from computable model theory and the theory of BAOs. Section 4 will be devoted to computable enumerations of families of sets. These will play a central role in our results. In Section 5 we construct a BAO with exactly n computable isomorphism types, for any given $n \in \omega$. This shows how the number of computable isomorphism types changes when we pass from Boolean algebras (where it can only be 1 or ω, see e.g., [2] or [9]) to Boolean algebras with operators. Finally, we demonstrate that adding a single constant to a computably categorical BAO can lift the number of isomorphism types of the expanded BAO to 2, and in fact to any finite k.

2. Basics

We begin with presenting basic definitions from computable model theory.

Definition 1. An algebra $\mathbf{A} = \langle A; f_0^{n_0}, f_1^{n_1}, \ldots \rangle$ is *computable* if $A = \omega$ and the function $f(i, x_1, \ldots, x_{n_i}) = f_i^{n_i}(x_1, \ldots, x_{n_i})$ is computable. An algebra $\mathbf{B}$ is computably presentable if $\mathbf{B}$ is (classically) isomorphic to

a computable algebra $\mathbf{A}$. In this case, any isomorphism from $\mathbf{A}$ onto $\mathbf{B}$ is called a *computable presentation* (or a *computable copy*) of $\mathbf{B}$.

For instance, the structure $\langle \omega; s \rangle$, with s being the successor function, is a computable algebra, and $\mathbf{B}_\omega$—the Boolean algebra generated by left-closed, right-open intervals of ω under its set-theoretical order—has a computable presentation.

We are interested in those computable algebras which have the same computable isomorphism types. We formalise this as follows.

Definition 2. An isomorphism f from a computable algebra $\mathbf{B}$ to a computable algebra $\mathbf{A}$ is a *computable isomorphism* if f itself is a computable function. In this case we say that $\mathbf{A}$ and $\mathbf{B}$ have the same *computable isomorphism type*.

For instance, any two computable copies of ω with successor are computably isomorphic. In general, two classically isomorphic computable algebras need not be computably isomorphic. The following definition makes this simple observation precise.

Definition 3. The *computable dimension* of an algebra $\mathbf{A}$ is the number of its computable isomorphism types. The algebra $\mathbf{A}$ is *computably categorical* if its computable dimension is 1.

A typical example of a computably categorical algebra is any computable finitely generated algebra. The reason is that any suitable mapping from generators in one computable copy to generators in another can be extended to a computable isomorphism. As another example we want to mention the following result obtained independently in [2] and [9] that charcterises computably categorical Boolean algebras.

Theorem 1. *A Boolean algebra $\mathbf{A}$ is computably categorical if and only if $\mathbf{A}$ has finitely many atoms. Moreover, if $\mathbf{A}$ is not computably categorical then its computable dimension is ω.*

Usually it is not difficult to find algebras of computable dimension ω or 1. Algebras of finite computable dimension greater than 1 are much more difficult to come by. Goncharov was the first to construct such algebras for any given n (cf. [3]).

3. Boolean algebras with operators

Since Theorem 1 above solves completely the question of computable isomorphism types of Boolean algebras, it seems natural to investigate the same question with respect to algebras which have a Boolean algebra reduct. A class of such algebras that suggests itself is the class of BAOs. These are Boolean algebras with additional operations which distribute over join and preserve zero in each argument. More formally, we have:

Definition 4. A Boolean algebra with operators (BAO) is an algebra $\mathbf{A} = \langle A; \wedge, \vee, ^-, f_i (i \in I), 0, 1 \rangle$ such that the reduct $\langle A; \wedge, \vee, ^-, 0, 1 \rangle$ is a Boolean algebra and each operation f_i, $(i \in I)$, say of arity k, satisfies the following:

(i) $f_i(a_0, \ldots, 0, \ldots, a_{k-1}) = 0$,

(ii) $f_i(a_0, \ldots, b \vee c, \ldots, a_{k-1}) = f_i(a_0, \ldots, b, \ldots, a_{k-1}) \vee f_i(a_0, \ldots, c, \ldots, a_{k-1})$, in each argument $j < k$.

In the terminology of [6] the first condition is called *normality* and the second *additivity*. Operations that satisfy (i) and (ii) are called *operators*. BAOs with (many) unary normal operators are often called *(poly-) modal algebras* because of the connection with modal logic. Such are going to be the BAOs we construct. In the sequel we will adopt the following notational convention. If $\mathbf{A}$ is a Boolean algebra, and f an operator on A, we will write $\langle \mathbf{A}; f \rangle$, for the algebra $\langle A; \wedge, \vee, ^-, f, 0, 1 \rangle$.

We will make a particular use of BAOs that arise from certain relational structures akin to unary algebras, by means of the construction we describe below. Let $\mathbf{U} = \langle U; f_0 \rangle$ be an infinite structure with f_0 a binary relation such that for each $u \in U$ the f_0-image of the singleton $\{u\}$ is a finite nonempty set. It will be convenient to view f_0 as a multi-valued function on U and for that reason we will call structures of this kind *multi-valued unary algebras*, although we are ready to give up this mouthful in favour of any better alternative. For a subset x of U we will write $f_0[x]$ for the f_0-image of x. Clearly, $\mathbf{U}$ is a unary algebra iff $f_0[\{u\}]$ is a singleton for all $u \in U$. Now, let $\mathbf{B}$ be the Boolean algebra $\langle B; \cup, \cap, ^-, U, \emptyset \rangle$, where B is the set of all finite and cofinite subsets of U. Then, define a unary operation f on B by putting:

$$fx = \begin{cases} f_0[x], & \text{if } x \text{ is a finite subset of } U, \\ U, & \text{otherwise.} \end{cases}$$

Further, define another unary operation g on B as follows:

$$gx = \begin{cases} \emptyset, & \text{if } x = \emptyset, \\ x^-, & \text{if } x = \{u\} \text{ for some } u \in U, \\ U, & \text{otherwise.} \end{cases}$$

So defined g will serve as a device for recognising atoms, since it sends every atom into its complement, zero to zero, and everything else to 1. Let $\mathbf{A}$ stand for the algebra $\langle \mathbf{B}; f, g \rangle$.

Lemma 1. *The algebra* $\mathbf{A}$ *is a BAO.*

Proof. We need to show that f and g are operators. To see that for f notice first that $f_0[\emptyset] = \emptyset$. Then, take $x, y \subseteq U$. If at least one of x, y is cofinite, $f(x \cup y) = U = fx \cup fy$. If both x and y are finite, $f(x \cup y) = f_0[x \cup y] = f_0[x] \cup f_0[y] = fx \cup fy$. Then, consider $g(x \cup y)$. Only the case where x and y are atoms is not immediate. Suppose first $x \neq y$. Then $x \cup y$ is not an atom and $x^- \cup y^- = U$. Thus, $g(x \cup y) = U = x^- \cup y^- = gx \cup gy$ as needed. If $x = y$, the claim holds trivially.

4. Computable enumerations

A standard method of constructing structures of finite computable dimension is to code certain families of sets into these structures so that computability theoretic properties of the families are reflected in computable isomorphism types. We need another definition.

Definition 5. A *computable enumeration* of a family F of computably enumerable subsets of ω is a bijective mapping $\nu : \omega \longrightarrow F$ such that the set $\{(i, x) : x \in \nu(i)\}$ is computably enumerable.

Two computable enumerations ν and μ of F are equivalent if there is a computable function f such that $\nu(i) = \mu(f(i))$. The idea behind is that from the ν-code of a set in F we can find its μ-code. One of the instrumental results in constructing structures of finite computable dimensions is the following theorem from [3].

Theorem 2. *For any $n \leq \omega$ there exists a family F that has exactly n pairwise non-equivalent computable enumerations.*

It is known that any family with finitely many infinite sets (and possibly infinitely many finite ones) has either exactly one or ω non-equivalent computable enumerations. The construction of an F satisfying the theorem above requires sophisticated methods from computability theory and is rather complicated. Fortunately, for the purpose at hand, all we need to know is that such a family exists.

5. BAOs with finite computable dimensions

We start with a countable multi-valued unary algebra $\mathbf{U}$ and construct the associated BAO $\mathbf{A}$, as described at the end of section 3.

Definition 6. A (presentation of) computable multi-valued unary algebra $\mathbf{U}$ is locally effective if there exists an algorithm that for any given $u \in U$ computes the cardinality of the set $f_0[u]$.

Note that any computable unary algebra is locally effective, as then by definition $|f_0[u]| = 1$, for all $u \in U$. However it is not at all hard to give examples of computable multi-valued unary algebras that are not locally effective. Our next lemma exhibits the relationship between computable presentations of $\mathbf{U}$ and the associated BAO $\mathbf{A}$.

Lemma 2. *The structure $\mathbf{U}$ has a locally effective computable presentation if and only if the algebra $\mathbf{A}$ has a computable presentation.*

Proof. Suppose that $\mathbf{U}$ has a locally effective computable presentation. We can assume that this presentation is $\mathbf{U}$ itself, i.e., in particular, $U = \omega$. Recall that $\mathbf{B}_\omega$ is the Boolean algebra generated by left-closed right-open intervals of ω under the natural ordering, and that $\mathbf{B}_\omega$ is isomorphic to the algebra of finite and cofinite subsets of ω. Let $\mathbf{B}'_\omega$ be a computable copy of $\mathbf{B}_\omega$ such that the set of atoms of $\mathbf{B}_\omega$—denoted $\mathrm{At}(\mathbf{B}'_\omega)$ later on—is a computable set. Let $\varphi : \omega \longrightarrow \mathrm{At}(\mathbf{B}'_\omega)$ be a computable bijection. The unary multi-valued function f_0 on U naturally induces a computable multi-valued function f'_0 on $\mathrm{At}(\mathbf{B}'_\omega)$ via $\varphi f_0 \varphi^{-1}$. Now we can computably extend f'_0 to a function f' as follows:

$$f'x = \begin{cases} 0, & \text{if } x = 0, \\ f_0[a_1] \vee \ldots \vee f_0[a_n], & \text{if } x = a_1 \vee \ldots \vee a_n, \text{ and } a_i \in \mathrm{At}(\mathbf{B}'_\omega),\ 1 \leq i \leq n, \\ 1, & \text{otherwise.} \end{cases}$$

Note that f' is a computable function, because $\mathbf{U}$ is locally effective. Further, define the function g' by:

$$g'x = \begin{cases} 0, & \text{if } x = 0, \\ x^-, & \text{if } x \in \mathrm{At}(\mathbf{B}'_\omega), \\ 1, & \text{otherwise.} \end{cases}$$

Clearly, this function is computable as well. Thus, $\langle \mathbf{B}'_\omega; f', g' \rangle$ is a computable copy of $\mathbf{A}$, with the isomorphism established by naturally extending φ from atoms onto the whole universe.

For the other direction, assume that $\mathbf{A}$ has a computable copy $\mathbf{A}'$. The set U' of atoms of $\mathbf{A}'$ is definable by the formula $gx = x^-$, and therefore computable. On U' define a binary relation f'_0 putting xf'_0y iff $y \leq f'x$ in $\mathbf{A}'$. Clearly, $\langle U'; f'_0 \rangle$ is a computable structure isomorphic to $\mathbf{U}$. Now, by computability of $\mathbf{A}'$ and the way f' behaves, given an $x \in U'$ one can compute the number of atoms $a_1, \ldots, a_n \in U'$ with $f'x = y_1 \vee \ldots \vee y_n$. By definition we have xf'_0y iff $y \in \{y_1, \ldots, y_n\}$, and therefore $\langle U'; f'_0 \rangle$ is also locally effective.

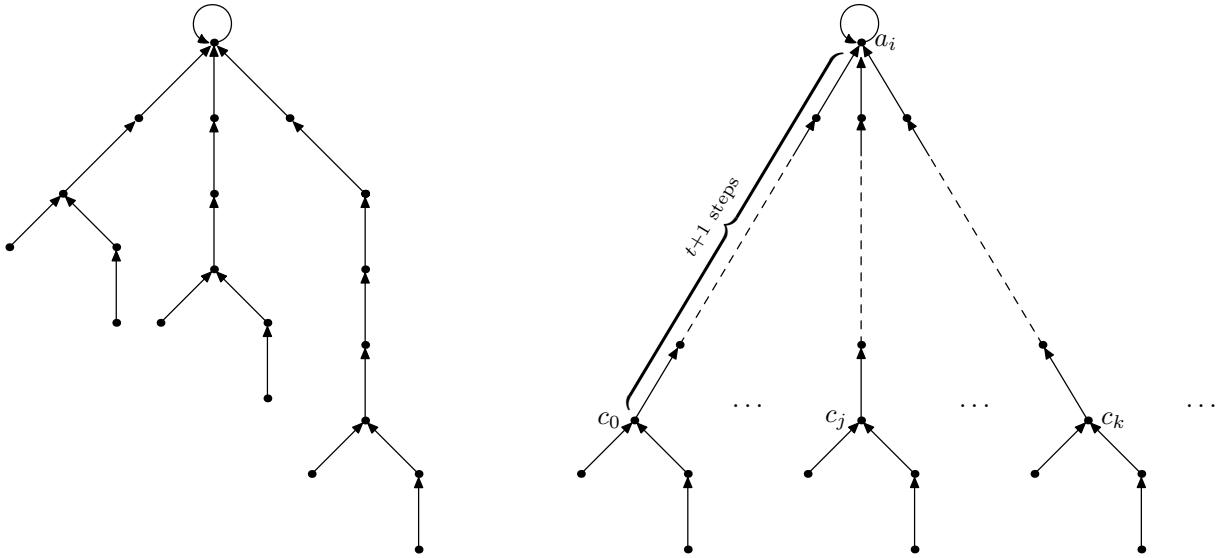
The proof of the lemma above implicitly defines two transformations: τ_B and τ_U. The former transforms a given computable copy of a multi-valued unary algebra into a computable BAO; the latter transforms a given computable copy of a BAO into a computable multi-valued unary algebra. It turns out that these transformations preserve computable isomorphism types. We state this in the next lemma whose proof we leave to the reader.

Lemma 3. *Multi-valued unary algebras $\mathbf{U}_1$ and $\mathbf{U}_2$ are computably isomorphic if and only if $\tau_B(\mathbf{U}_1)$ is computably isomorphic to $\tau_B(\mathbf{U}_2)$. Similarly, BAOs $\mathbf{A}_1$ and $\mathbf{A}_2$ are computably isomorphic if and only if $\tau_U(\mathbf{A}_1)$ is computably isomorphic to $\tau_U(\mathbf{A}_2)$.*

The two lemmas above show that in order to define BAOs with finite computable dimensions is suffices to produce unary algebras of finite computable dimensions. The next lemma takes care of this. The reader will notice that the unary algebras we construct here are not multi-valued, in other words, they are indeed algebras.

Lemma 4. *For each $n \in \omega$ there exists a unary algebra $\mathbf{U}$ of computable dimension n.*

Proof. In the proof we use the family F that has exactly n nonequivalent computable enumerations (see [3]). Let $\nu : \omega \longrightarrow F$ be a computable enumeration of F. For each $i \in \omega$ construct a unary algebra $\mathbf{U}_i$ that codes the set $\nu(i)$, as in the picture below.



Picture 1. Left: code of the set $\{1, 2, 4\}$; right: algebra $\mathbf{U}_i$.

As it is easy to see from Picture 1, the construction guarantees that the set $\nu(i)$ coincides with $\{t : \exists c_j : f^{t+1}(c_j) = a_i\}$. Informally, we view the element a_i as the code of $\nu(i)$. Clearly, $\mathbf{U}_i$ can be constructed from $\nu(i)$. Notice that $\mathbf{U}_i$ does not have any nontrivial automorphisms.

Let $\mathbf{U}_\nu$ be an effective disjoint union of all $\mathbf{U}_i$. The following list of properties shows that $\mathbf{U}_\nu$ is the desired algebra.

(1) For all computable enumerations ν and μ of F, the algebras $\mathbf{U}_\nu$ and $\mathbf{U}_\mu$ are isomorphic.

(2) For all computable enumerations ν and μ of F, the algebras $\mathbf{U}_\nu$ and $\mathbf{U}_\mu$ are computably isomorphic iff ν is equivalent to μ.

(3) From any computable copy $\mathbf{U}$ of $\mathbf{U}_\nu$ one can construct a computable enumeration ν_U of F such that ν_U and μ are equivalent iff $\mathbf{U}_{\nu_U}$ and $\mathbf{U}_\mu$ are computably isomorphic.

Property (1) is clear from the construction, as is the 'if' part of property (2). For its 'only if' part suppose $\mathbf{U}_\nu$ is computably isomorphic to $\mathbf{U}_\mu$ via φ. For an $a_i \in U_\nu$ with $fa_i = a_i$, consider $\varphi(a_i)$. Since φ is an isomorphism, $f\varphi(a_i) = \varphi(a_i)$ and the set $\{x \in U_\nu : f^k x = a \text{ for some } k \in \omega\}$, which in fact is the subalgebra $\mathbf{U}_i$ of $\mathbf{U}_\nu$, is isomorphic to $\{\varphi(x) \in U_\mu : f^k\varphi(x) = a_i \text{ for some } k \in \omega\}$. Again, speaking informally, $\varphi(a_i)$ codes the same set in $\mathbf{U}_\mu$ as a_i in $\mathbf{U}_\nu$. This proves that ν and μ are equivalent by means of φ composed with the computable bijection $a_i \mapsto i$. Now, the second part of property (3) follows from property (2). For its first part, here is the procedure. Pick up any element of U; iterate f until you reach an element a with $fa = a$; this is a code of some set from family F; then keep enumerating U until the complete path from a back to the fork-shaped quadruple of elements has appeared. Now you know that the number t—the length of the path from the top of the fork to a, minus 1—belongs to the set coded by a. Repeating this *ad infinitum* you will have enumerated all pairs (a, t) such that t belongs to the set coded by a, precisely what a computable enumeration of F requires.

The lemmas above amount to a proof of the following theorem.

Theorem 3. *For each $n \in \omega$ there exists a BAO $\mathbf{A}$ of computable dimension n.*

6. Adding constants

To finish off we will address the question of how constants affect computable dimensions of BAOs. Notice first that adding a constant to a BAO results in another BAO, so such expansions seem rather natural. The following theorem proved by Millar in [8] shows that certain amount of decidability makes constants irrelevant to the issue.

Theorem 4. *If a structure $\mathbf{S}$ is computably categorical and its existential theory is decidable, then any expansion of $\mathbf{S}$ by finitely many constants is also computably categorical.*

However, without the decidability assumption the picture changes quite dramatically, as Cholak, Goncharov, Khoussainov and Shore proved in [1]. Namely:

Theorem 5. *For each $k \in \omega$ there is a computably categorical structure* $\mathbf{S}$ *such that the expansion of* $\mathbf{S}$ *by a single constant has computable dimension k.*

The proof of the above theorem proceeds by coding certain computably enumerable families of k-tuples of sets. We will show that the theorem remains true if $\mathbf{S}$ is assumed to be a BAO. In fact, we will confine ourselves to the case $k = 2$, the extension to any positive k being rather straightforward given the constructions in [1]. We will need yet another definition.

Definition 7. Let F be a family of nonempty subsets of ω. We call F symmetric, if $(A, B) \in F$ implies $(B, A) \in F$. The family F is computably enumerable, if there is a bijective mapping $\nu : \omega \longrightarrow F$ such that the set $\{(i, x, y) : x \in l\nu(i),\ y \in r\nu(i)\}$ is computably enumerable, where l and r stand for the left and right projections of pairs.

If $\nu : \omega \longrightarrow F$ is a computable enumeration of a symmetric family F, then ν^d defined as $\nu^d(i) = (r\nu(i), l\nu(i))$ is another computable enumeration of F. To prove the result we announced, we will need a symmetric family F for which ν and ν^d are the only computable enumerations possible. That is the content of another theorem from [1].

Theorem 6. *There is a symmetric family F and its computable enumeration ν such that ν is not equivalent to ν^d and every computable enumeration of F is equivalent to either ν of ν^d.*

The existence of the family F granted, and given a computable enumeration ν of F, we will construct a multi-valued unary algebra $\mathbf{M}_\nu$, whose fragment is shown in Picture 2. It shows how to code two pairs from F. Again, informally we say that the element a_i codes the pair (A, B) under enumeration ν, the element b_i codes the same pair under ν^d; by symmetry the pair (B, A) must also appear somewhere, we have drawn it as coded by a_j and b_j under ν and ν^d, respectively. The whole $\mathbf{M}_\nu$ is obtained by repeating this for all pairs $(A, B) \in F$, so it can be visualised as a daisy-wheel-like structure around the elements c and d.

Given a structure $\mathbf{S}$ and an element $s \in S$, we will write $\langle \mathbf{S}, s \rangle$ for the structure of the type of $\mathbf{S}$ expanded by a single constant interpreted as the element s. For instance, $\langle \mathbf{M}_\nu, c \rangle$ and $\langle \mathbf{M}_\nu, d \rangle$ arise from $\mathbf{M}_\nu$ by adding a name for the element c and d, respectively.

Lemma 5. *The structure $\mathbf{M}_\nu$ has the following properties:*

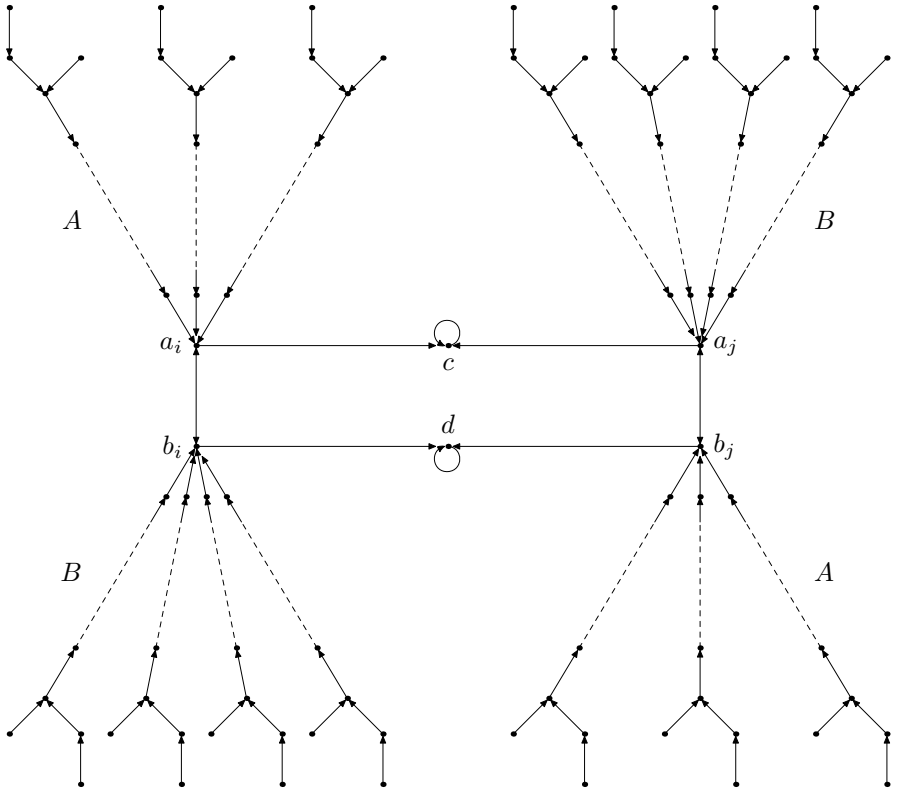
(i) *there is a single nontrivial automorphism α of $\mathbf{M}_\nu$, and $\alpha(a_i) = b_i$;*

(ii) *if μ is a computable enumeration of F, then $\mathbf{M}_\mu$ is isomorphic to $\mathbf{M}_\nu$;*

(iii) *the structures $\langle \mathbf{M}_\nu, c \rangle$ and $\langle \mathbf{M}_\nu, d \rangle$ are classically isomorphic, but not computably so;*

(iv) *the computable dimension of $\langle \mathbf{M}_\nu, c \rangle$ is 2;*

(v) *the computable dimension of $\mathbf{M}_\nu$ is 1.*



Picture 2. Fragment of $\mathbf{M}_\nu$.

Proof. Part (i) is clear from the picture; the automorphism in question is just the flipping over. For part (ii) notice that μ is equivalent to either ν of ν^d. Therefore, $\mathbf{M}_\mu$ is isomorphic (computably) to either $\mathbf{M}_\nu$ or $\mathbf{M}_{\nu^d}$. These are in turn isomorphic by (i). For (iii), classical isomorphism is clear. Suppose $\langle \mathbf{M}_\nu, c\rangle$ and $\langle \mathbf{M}_\nu, d\rangle$ are computably isomorphic. The isomorphism then induces a computable function $a_i \mapsto b_i$, which makes enumerations ν and ν^d equivalent. This contradiction shows that such an isomorphism cannot be computable. For part (iv) we have by construction that from a computable copy $\langle \mathbf{M}, e\rangle$ of $\langle \mathbf{M}_\nu, c\rangle$ a computable enumeration μ of F can be effectively obtained, for instance as a listing of members of the computable set $\{x \in M : x \neq e,\ e \in f_0[x],\ x \in f_0^2[x]\}$. If μ is equivalent to ν, then $\langle \mathbf{M}, e\rangle$ is computably isomorphic to $\langle \mathbf{M}_\nu, c\rangle$; otherwise μ is equivalent to ν^d, and then $\langle \mathbf{M}, e\rangle$ is computably isomorphic to $\langle \mathbf{M}_\nu, d\rangle$. Finally, to prove (v) let $\mathbf{M}$ be a computable copy of $\mathbf{M}_\nu$. Enumerate M until an element e with $e \in f_0[e]$ appears. Then send e to either c or d and then proceed as in the case with the additional constant. This establishes a computable isomorphism.

Theorem 7. *There is a computably categorical BAO* $\mathbf{A}$*, such that for a certain element* $a \in A$*, the expanded algebra* $\langle \mathbf{A}, a\rangle$ *has computable dimension 2.*

Proof. Take the multi-valued unary algebra $\mathbf{M}_\nu$ and produce the BAO $\mathbf{A}$ as in section 3. Then pick the element $c \in M_\nu$; clearly $a = \{c\}$ is an element (in fact, an atom) of A. Repeating the argument from section 4 we see that computable dimensions of $\mathbf{A}$ and $\langle \mathbf{A}, a\rangle$ are respectively the same as the dimensions of $\mathbf{M}_\nu$ and $\langle \mathbf{M}_\nu, c\rangle$.

By coding "families of dimension k", as they are called in [1], we can also obtain the following generalisation of Theorem 7, which we will state without proof.

Theorem 8. *For any $k \in \omega$, there is a computably categorical BAO $\mathbf{A}$, such that for a certain element $a \in A$, the expanded algebra $\langle \mathbf{A}, a \rangle$ has computable dimension k.*

Closing remarks

As we mentioned in the introduction, the computable dimension of any computable Boolen algebra is either 1 or ω. In this paper we produced BAOs with computable dimension k, for any $k \in \omega$. Although our construction employed two unary operators, there are ways of achieving the same with only one. It suffices to say that the simulation technique from [7], enabling one to mimic the action of two operators with only one, preserves computability. It would be therefore interesting to find a natural variety $\mathcal{V}$ of modal algebras which is not term-equivalent to Boolean algebras and yet the computable dimension of any algebra from $\mathcal{V}$ is either 1 or ω, thus extending the result for Boolean algebras. In the variety of monadic algebras all subdirectly irreducible members are such, so this variety may well be the first candidate to consider.

References

[1] P. Cholak, S. Goncharov, B. Khoussainov, R. Shore, Computably categorical structures and expansions by constants, *Journal of Symbolic Logic* 64, 1999, 13–37.

[2] V. Dzgoev, S. Goncharov, Autostability of models, *Algebra and Logic* 19, 1980, 45–53.

[3] S. Goncharov, The problem of the number of non-self-equivalent constructivizations, *Algebra and Logic* 19, 1988, 621–639.

[4] S. Goncharov, A. Nerode, J. Remmel (eds) *Handbook of Recursive Mathematics*, Studies in Logic and the Foundations of Mathematics, vols. 138, 139, Elsevier, 1998.

[5] S. Goncharov, S. Lempp, R. Solomon, The computable dimension of ordered Abelian groups, to appear in *Advances of Mathematics.*

[6] B. Jónsson, A. Tarski, Boolean algebras with operators. Part I, *American Journal of Mathematics* 73, 1951, 891–939.

[7] M. Kracht, F. Wolter, Normal modal logics can simulate all others, *Journal of Symbolic Logic* 64, 1999, 99–138.

[8] T. Millar, Recursive categoricity and persistence, *Journal of Symbolic Logic* 51, 1986, 430–434.

[9] J. Remmel, Recursive isomorphism types of recursive Boolean algebras, *Journal of Symbolic Logic* 46, 1981, 572–593.