Isomorphisms between low$_n$ and computable Boolean algebras

Jonny Stephenson

UChicago

October 1, 2013

Computability of low$_n$ Boolean algebras

Theorem (Downey, Jockusch 1994)

Every low Boolean algebra has a computable copy.

Computability of low$_n$ Boolean algebras

Theorem (Downey, Jockusch 1994)

Every low Boolean algebra has a computable copy.

Theorem (Thurber 1995)

Every low$_2$ Boolean algebra has a computable copy.

Computability of low$_n$ Boolean algebras

Theorem (Downey, Jockusch 1994)

Every low Boolean algebra has a computable copy.

Theorem (Thurber 1995)

Every low$_2$ Boolean algebra has a computable copy.

Theorem (Knight, Stob 2000)

Every low$_4$ Boolean algebra has a computable copy.

Computability of low$_n$ Boolean algebras

Theorem (Downey, Jockusch 1994)

Every low Boolean algebra has a computable copy.

Theorem (Thurber 1995)

Every low$_2$ Boolean algebra has a computable copy.

Theorem (Knight, Stob 2000)

Every low$_4$ Boolean algebra has a computable copy.

In each case the isomorphism from the low$_n$ copy to the computable copy is $\emptyset^{(n+2)}$-computable.

Questions

Question

Is it the case that every low$_n$ Boolean algebra has a computable copy?

Questions

Question

Is it the case that every low$_n$ Boolean algebra has a computable copy?

This question is open.

Questions

Question

Is it the case that every low_n *Boolean algebra has a computable copy?*

This question is open.

Question

Is it the case that every low_n *Boolean algebra has a computable copy with a* $\emptyset^{(n+2)}$*-computable isomorphism?*

Questions

Question

Is it the case that every low$_n$ Boolean algebra has a computable copy?

This question is open.

Question

Is it the case that every low$_n$ Boolean algebra has a computable copy with a $\emptyset^{(n+2)}$-computable isomorphism?

No!

Questions

Question

Is it the case that every low_n Boolean algebra has a computable copy?

This question is open.

Question

Is it the case that every low_n Boolean algebra has a computable copy with a $\emptyset^{(n+2)}$-computable isomorphism?

No!

Theorem (Harris, Montalbán 2010)

There is a low_5 Boolean algebra with no $\emptyset^{(7)}$-computable isomorphism to any computable copy.

Questions

Question

Is it the case that every low$_n$ Boolean algebra has a computable copy?

This question is open.

Question

Is it the case that every low$_n$ Boolean algebra has a computable copy with a $\emptyset^{(n+2)}$-computable isomorphism?

No!

Theorem (Harris, Montalbán 2010)

There is a low$_5$ Boolean algebra with no $\emptyset^{(7)}$-computable isomorphism to any computable copy.

The algebra has a computable copy, and a $\emptyset^{(8)}$-computable isomorphism to it.

The main result

So the behaviour at $n = 5$ is different. Is this an instance of a more general phenomenon?

The main result

So the behaviour at $n = 5$ is different. Is this an instance of a more general phenomenon?

Question

For which values of n is there a low_n boolean algebra that has no $\emptyset^{(n+2)}$-computable isomorphism to a computable copy?

Theorem (Stephenson 2013)

For each $n \in \omega$ there is a low_{2n+5} Boolean algebra with no $\emptyset^{(2n+7)}$-computable isomorphism to any computable copy.

The main result

So the behaviour at $n = 5$ is different. Is this an instance of a more general phenomenon?

Question

For which values of n is there a low_n *boolean algebra that has no* $\emptyset^{(n+2)}$*-computable isomorphism to a computable copy?*

Theorem (Stephenson 2013)

For each $n \in \omega$ *there is a* low_{2n+5} *Boolean algebra with no* $\emptyset^{(2n+7)}$*-computable isomorphism to any computable copy.*

The plan:

The main result

So the behaviour at $n = 5$ is different. Is this an instance of a more general phenomenon?

Question

For which values of n is there a low$_n$ boolean algebra that has no $\emptyset^{(n+2)}$-computable isomorphism to a computable copy?

Theorem (Stephenson 2013)

For each $n \in \omega$ there is a low$_{2n+5}$ Boolean algebra with no $\emptyset^{(2n+7)}$-computable isomorphism to any computable copy.

The plan:

- Use Montalbán's copy/diagonalise game.

The main result

So the behaviour at $n = 5$ is different. Is this an instance of a more general phenomenon?

Question

For which values of n is there a low_n boolean algebra that has no $\emptyset^{(n+2)}$-computable isomorphism to a computable copy?

Theorem (Stephenson 2013)

For each $n \in \omega$ there is a low_{2n+5} Boolean algebra with no $\emptyset^{(2n+7)}$-computable isomorphism to any computable copy.

The plan:

- Use Montalbán's copy/diagonalise game.
- Place the low_5 construction of Harris and Montalbán into this context.

The main result

So the behaviour at $n = 5$ is different. Is this an instance of a more general phenomenon?

Question

For which values of n is there a low_n boolean algebra that has no $\emptyset^{(n+2)}$-computable isomorphism to a computable copy?

Theorem (Stephenson 2013)

For each $n \in \omega$ there is a low_{2n+5} Boolean algebra with no $\emptyset^{(2n+7)}$-computable isomorphism to any computable copy.

The plan:

- Use Montalbán's copy/diagonalise game.
- Place the low_5 construction of Harris and Montalbán into this context.
- Perform an inductive argument to get the theorem.

Montalbán's Copy/Diagonalise Game

If $\mathbb{K}$ is a class of structures over a language L, and α is some a computable ordinal, then $\mathsf{G}^{\alpha}(\mathbb{K})$ is a game played by a *copying player* C and a *diagonalising player* D.

Montalbán's Copy/Diagonalise Game

If $\mathbb{K}$ is a class of structures over a language L, and α is some a computable ordinal, then $\mathsf{G}^{\alpha}(\mathbb{K})$ is a game played by a *copying player* C and a *diagonalising player* D.

C constructs infinitely many L-structures $(\mathcal{C}^i)_{i\in\omega}$.
D constructs one structure $\mathcal{D}$.

Montalbán's Copy/Diagonalise Game

If $\mathbb{K}$ is a class of structures over a language L, and α is some a computable ordinal, then $\mathsf{G}^\alpha(\mathbb{K})$ is a game played by a *copying player* C and a *diagonalising player* D.

C constructs infinitely many L-structures $(\mathcal{C}^i)_{i\in\omega}$.
D constructs one structure $\mathcal{D}$.

The players play finite approximations to L-structures, extending at each stage.
At stage s, C builds $\mathcal{C}^i[s]$ for each i, and D responds with $\mathcal{D}[s]$.

Montalbán's Copy/Diagonalise Game

C computes some i from the α^{th} jump of D's sequence of moves, and wants $\mathcal{D} \cong \mathcal{C}^i$. This is the main win condition.

Montalbán's Copy/Diagonalise Game

C computes some i from the α^{th} jump of D's sequence of moves, and wants $\mathcal{D} \cong \mathcal{C}^i$. This is the main win condition.

If C fails to construct structures in $\mathbb{K}$, then D wins.

Montalbán's Copy/Diagonalise Game

C computes some i from the α^{th} jump of D's sequence of moves, and wants $\mathcal{D} \cong \mathcal{C}^i$. This is the main win condition.

If C fails to construct structures in $\mathbb{K}$, then D wins.

If D fails to construct structures in $\mathbb{K}$ but C does, then C wins.

Montalbán's Copy/Diagonalise Game

C computes some i from the α^{th} jump of D's sequence of moves, and wants $\mathcal{D} \cong \mathcal{C}^i$. This is the main win condition.

If C fails to construct structures in $\mathbb{K}$, then D wins.

If D fails to construct structures in $\mathbb{K}$ but C does, then C wins.

If both players construct structures in $\mathbb{K}$, then the winner is determined by the main win condition.

Montalbán's Copy/Diagonalise Game

C computes some i from the α^{th} jump of D's sequence of moves, and wants $\mathcal{D} \cong \mathcal{C}^i$. This is the main win condition.

If C fails to construct structures in $\mathbb{K}$, then D wins.

If D fails to construct structures in $\mathbb{K}$ but C does, then C wins.

If both players construct structures in $\mathbb{K}$, then the winner is determined by the main win condition.

We want to know when the diagonalising player has a winning strategy, and how effective it is.

$\Sigma_n^{c,Y}$ formulas

Definition

Let L be a computable language, and $Y \subseteq \omega$. $\Sigma_n^{c,Y}$ denotes all $L_{\omega_1,\omega}$-formulas where the infinitary conjunctions and disjunctions are Y-c.e.

$\Sigma_n^{c,Y}$ formulas

Definition

Let L be a computable language, and $Y \subseteq \omega$. $\Sigma_n^{c,Y}$ denotes all $L_{\omega_1,\omega}$-formulas where the infinitary conjunctions and disjunctions are Y-c.e.

Definition

Let $\mathbb{B}^{n,Y}$ be the class of structures that are $\Sigma_n^{c,Y}$-presentations of Boolean algebras.

Diagonalising and low$_n$ Boolean algebras

Theorem

Suppose that $n \geq 1$, and the diagonalising player has a winning $\emptyset^{(n)}$-computable strategy in $G^\alpha(\mathbb{B}^{n,\emptyset})$.

Then there is a low$_n$ Boolean algebra $\mathcal{D}$ which is not isomorphic to any computable Boolean algebra by a $\emptyset^{(n+\alpha)}$-computable map.

Diagonalising and low_n Boolean algebras

The proof of Harris and Montalbán can be adapted to show that the diagonalising player has a $\emptyset^{(5)}$-computable winning strategy in the game $G^2(\mathbb{B}^{5,\emptyset})$.

Combining with the previous result essentially gives their construction of a low_5 Boolean algebra.

Diagonalising and low$_n$ Boolean algebras

The proof of Harris and Montalbán can be adapted to show that the diagonalising player has a $\emptyset^{(5)}$-computable winning strategy in the game $G^2(\mathbb{B}^{5,\emptyset})$.
Combining with the previous result essentially gives their construction of a low$_5$ Boolean algebra.

The argument can be relativised to show:

Theorem

The diagonalising player has a winning $Y^{(5)}$-computable strategy in $G^2(\mathbb{B}^{5,Y})$, for each $Y \subseteq \omega$.

Translating between families of structures

We now want to consider how a strategy for winning a game $\mathsf{G}^\alpha(\mathbb{K})$ in one class of structures might be used to get a strategy that wins a game for a different class of structures.

Definition

Suppose that $\mathbb{K}$ and $\mathbb{L}$ are two families of structures.
Let Φ and Ψ be Turing functionals with the following properties:

- *If $\mathcal{K}$ is a structure in $\mathbb{K}$ and $\mathcal{L}$ is a structure in $\mathbb{L}$, then $\Phi(\mathcal{L}) \in \mathbb{K}$ and $\Psi(\mathcal{K}) \in \mathbb{L}$.*
- *If $\mathcal{K} \in \mathbb{K}$, then $\Phi(\Psi(\mathcal{K}))$ and $\mathcal{K}$ are isomorphic.*
- *If $\mathcal{L}_1, \mathcal{L}_2 \in \mathbb{L}$ are isomorphic to the same structure in the image of $\Psi(\mathbb{K})$, then $\Phi(\mathcal{K}_1)$ and $\Phi(\mathcal{K}_2)$ are isomorphic.*

Then we say Φ and Ψ translate diagonalisation from $\mathbb{K}$ to $\mathbb{L}$.

Translating between families of structures

Theorem (vandenDriessche, Gonzáles — in $G^{\infty}(\mathbb{K})$)

Suppose that $\mathbb{K}$ and $\mathbb{L}$ are two families of structures, and let Φ and Ψ translate diagonalisation from $\mathbb{K}$ to $\mathbb{L}$.
Suppose that the diagonalising player has a computable strategy to win in $G^{\alpha}(\mathbb{K})$.
The the diagonalising player has a computable strategy to win in $G^{\alpha}(\mathbb{L})$.

$\omega\mathcal{B}$

Definition

Let $\mathcal{B}$ be a Boolean algebra, and let $L(\mathcal{B})$ be a linearisation of $\mathcal{B}$. Let $\omega\mathcal{B}$ be the Boolean algebra $Int(\omega L(\mathcal{B}))$.

Definition

Let $\mathcal{B}$ be a Boolean algebra, and let $L(\mathcal{B})$ be a linearisation of $\mathcal{B}$. Let $\omega\mathcal{B}$ be the Boolean algebra $Int(\omega L(\mathcal{B}))$.

Definition

If $\mathcal{B}$ is an infinite Boolean algebra, let $F_{\mathcal{B}}$ be the collection of elements of $\mathcal{B}$ which consist of finitely many atoms.

We can find $\omega\mathcal{B}$ effectively from $\mathcal{B}$, and arrange that $F_{\omega\mathcal{B}}$ is some fixed set, and thus we know which elements are atoms or finite collections of atoms.

$\mathcal{B}/F_{\mathcal{B}}$

Definition

Let $\mathcal{B}/F_{\mathcal{B}}$ be the Boolean algebra obtained by identifying any two elements of $\mathcal{B}$ that differ by a member of $F_{\mathcal{B}}$.

$\mathcal{B}/F_{\mathcal{B}}$

Definition

Let $\mathcal{B}/F_{\mathcal{B}}$ be the Boolean algebra obtained by identifying any two elements of $\mathcal{B}$ that differ by a member of $F_{\mathcal{B}}$.

Note that taking this quotient preserves isomorphism, and in addition that $\omega\mathcal{B}/F_{\omega\mathcal{B}} \cong \mathcal{B}$.

The plan

We want to show that for every $m \in \omega$ there is a low$_{2m+5}$ Boolean algebra not $\emptyset^{(2m+7)}$-computably isomorphic to any computable Boolean algebra.

The plan

We want to show that for every $m \in \omega$ there is a low$_{2m+5}$ Boolean algebra not $\emptyset^{(2m+7)}$-computably isomorphic to any computable Boolean algebra.

To do so, let $Y \subseteq \omega$ and $n \in \omega$. Consider the families $\mathbb{B}^{n,Y''}$ and $\mathbb{B}^{n+2,Y}$.

We will show:

Theorem

If D *has a $Y^{(n+2)}$-computable strategy for winning $G^\alpha(\mathbb{B}^{n,Y''})$, then* D *has a $Y^{(n+2)}$-computable strategy for winning $G^\alpha(\mathbb{B}^{n+2,Y})$.*

The plan

We want to show that for every $m \in \omega$ there is a low$_{2m+5}$ Boolean algebra not $\emptyset^{(2m+7)}$-computably isomorphic to any computable Boolean algebra.

To do so, let $Y \subseteq \omega$ and $n \in \omega$. Consider the families $\mathbb{B}^{n,Y''}$ and $\mathbb{B}^{n+2,Y}$.

We will show:

Theorem

If D *has a $Y^{(n+2)}$-computable strategy for winning $G^{\alpha}(\mathbb{B}^{n,Y''})$, then* D *has a $Y^{(n+2)}$-computable strategy for winning $G^{\alpha}(\mathbb{B}^{n+2,Y})$.*

To do so, we'll want to establish Φ and Ψ to translate diagonalisation. Choose

- $\Psi : \mathbb{B}^{n,Y''} \to \mathbb{B}^{n+2,Y}$ to be $\mathcal{B} \mapsto \omega\mathcal{B}$
- $\Phi : \mathbb{B}^{n+2,Y} \to \mathbb{B}^{n,Y''}$ to be $\mathcal{B} \mapsto \mathcal{B}/F_{\mathcal{B}}$

The plan

It suffices to show that

- We can effectively evaluate $\Sigma_n^{c,Y''}$ L-formulas in $\mathcal{B}/F_{\mathcal{B}}$ by consulting $\Sigma_{n+2}^{c,Y}$ L-formulas in $\mathcal{B}$.
- We can effectively evaluate $\Sigma_{n+2}^{c,Y}$ L-formulas in $\omega\mathcal{B}$ by consulting $\Sigma_n^{c,Y''}$ L-formulas in $\mathcal{B}$.

Constructing the map Φ is easy: two elements of $\mathcal{B}/F_{\mathcal{B}}$ are equal when their preimages in $\mathcal{B}$ differ by a member of $F_{\mathcal{B}}$, which is Π_2^c to check.

L_2-formulas in $\omega\mathcal{B}$

Now, we'll construct Ψ.

L_2-formulas in $\omega\mathcal{B}$

Now, we'll construct Ψ.
Let L_2 be the language of boolean algebras together with two additional predicates Inf and Atom. Knowing these predicates is equivalent to having the second structural jump of a boolean algebra, provided that it is of the form $\omega\mathcal{B}$.

L_2-formulas in $\omega\mathcal{B}$

Now, we'll construct Ψ.

Let L_2 be the language of boolean algebras together with two additional predicates Inf and Atom. Knowing these predicates is equivalent to having the second structural jump of a boolean algebra, provided that it is of the form $\omega\mathcal{B}$.

Lemma

Given an index for a $\Sigma_{n+2}^{c,Y}$ L-formula $\varphi(x_1, \cdots, x_n)$ which implies that $x_1 \dot{\vee} \cdots \dot{\vee} x_n = 1$, we can uniformly compute an index for a $\Sigma_n^{c,Y''}$ L_2-formula which is equivalent to φ in Boolean algebras of the form $\omega\mathcal{B}$.

L_2-formulas in $\omega\mathcal{B}$

We can express any $\Sigma_n^{c,Y''}$ L_2-fact in an equivalent way which describes how a Boolean algebra can be partitioned into successively finer pieces. Use this partition refinement structure to split the information expressed by $\varphi(\bar{a})$ into two kinds:

L_2-formulas in $\omega\mathcal{B}$

We can express any $\Sigma_n^{c,Y''}$ L_2-fact in an equivalent way which describes how a Boolean algebra can be partitioned into successively finer pieces. Use this partition refinement structure to split the information expressed by $\varphi(\bar{a})$ into two kinds:

- Computable information about whether the elements in $\bar{a}$ are finite collections of atoms (and if so, how large).

L_2-formulas in $\omega\mathcal{B}$

We can express any $\Sigma_n^{c,Y''}$ L_2-fact in an equivalent way which describes how a Boolean algebra can be partitioned into successively finer pieces. Use this partition refinement structure to split the information expressed by $\varphi(\bar{a})$ into two kinds:

- Computable information about whether the elements in $\bar{a}$ are finite collections of atoms (and if so, how large).
- $\Sigma_n^{c,Y''}$ information which comes from $\mathcal{B}$ (telling us how the corresponding tuple there can be partitioned).

L_2-formulas in $\omega\mathcal{B}$

We can express any $\Sigma_n^{c,Y''}$ L_2-fact in an equivalent way which describes how a Boolean algebra can be partitioned into successively finer pieces. Use this partition refinement structure to split the information expressed by $\varphi(\bar{a})$ into two kinds:

- Computable information about whether the elements in $\bar{a}$ are finite collections of atoms (and if so, how large).
- $\Sigma_n^{c,Y''}$ information which comes from $\mathcal{B}$ (telling us how the corresponding tuple there can be partitioned).

Recombine these two kinds of information to build a $\Sigma_n^{c,Y''}$ L-formula to evaluate in $\mathcal{B}$.

Results

This suffices to show:

Theorem

If D *has a* $Y^{(n+2)}$*-computable strategy for winning* $G^{\alpha}(\mathbb{B}^{n,Y''})$*, then* D *has a* $Y^{(n+2)}$*-computable strategy for winning* $G^{\alpha}(\mathbb{B}^{n+2,Y})$*.*

Now we recall that:

Results

This suffices to show:

Theorem

If D *has a* $Y^{(n+2)}$*-computable strategy for winning* $G^{\alpha}(\mathbb{B}^{n,Y''})$*, then* D *has a* $Y^{(n+2)}$*-computable strategy for winning* $G^{\alpha}(\mathbb{B}^{n+2,Y})$*.*

Now we recall that:

Lemma

For each $Y \subseteq \omega$*, the diagonalising player has a winning* $Y^{(5)}$*-computable strategy in the game* $G^{2}(\mathbb{B}^{5,Y})$*.*

Results

This suffices to show:

Theorem

If D *has a* $Y^{(n+2)}$*-computable strategy for winning* $G^{\alpha}(\mathbb{B}^{n,Y''})$*, then* D *has a* $Y^{(n+2)}$*-computable strategy for winning* $G^{\alpha}(\mathbb{B}^{n+2,Y})$*.*

Now we recall that:

Lemma

For each $Y \subseteq \omega$*, the diagonalising player has a winning* $Y^{(5)}$*-computable strategy in the game* $G^{2}(\mathbb{B}^{5,Y})$*.*

In particular:

Results

This suffices to show:

Theorem

If D has a $Y^{(n+2)}$-computable strategy for winning $G^{\alpha}(\mathbb{B}^{n,Y''})$, then D has a $Y^{(n+2)}$-computable strategy for winning $G^{\alpha}(\mathbb{B}^{n+2,Y})$.

Now we recall that:

Lemma

For each $Y \subseteq \omega$, the diagonalising player has a winning $Y^{(5)}$-computable strategy in the game $G^2(\mathbb{B}^{5,Y})$.

In particular:

Lemma

D has a $\emptyset^{(2n+5)}$-computable strategy for winning $G^2(\mathbb{B}^{5,\emptyset^{(2n)}})$.

Main Result

Now apply our result to see that

Main Result

Now apply our result to see that
D has a $\emptyset^{(2n+5)}$-computable strategy for winning $G^2(\mathbb{B}^{7,\emptyset^{(2n-2)}})$

Main Result

Now apply our result to see that
D has a $\emptyset^{(2n+5)}$-computable strategy for winning $G^2(\mathbb{B}^{7,\emptyset^{(2n-2)}})$
$\vdots$
D has a $\emptyset^{(2n+5)}$-computable strategy for winning $G^2(\mathbb{B}^{2n+5,\emptyset})$

Main Result

Now apply our result to see that
D has a $\emptyset^{(2n+5)}$-computable strategy for winning $G^2(\mathbb{B}^{7,\emptyset^{(2n-2)}})$
$\vdots$
D has a $\emptyset^{(2n+5)}$-computable strategy for winning $G^2(\mathbb{B}^{2n+5,\emptyset})$
and hence

Main Result

Now apply our result to see that
D has a $\emptyset^{(2n+5)}$-computable strategy for winning $G^2(\mathbb{B}^{7,\emptyset^{(2n-2)}})$
$\vdots$
D has a $\emptyset^{(2n+5)}$-computable strategy for winning $G^2(\mathbb{B}^{2n+5,\emptyset})$
and hence

Theorem (Stephenson 2013)

There is a low_{2n+5} Boolean algebra with no $\emptyset^{(2n+7)}$-computable isomorphism between it and a computable Boolean algebra.

Future Directions

The machinery is very general, and the family of structures built will be improved by a stronger base case (e.g. arguments for the 6^{th} jump, or an argument showing no $\emptyset^{(n+3)}$-computable isomorphism between some low$_n$ boolean algebra and any computable copy).

Future Directions

The machinery is very general, and the family of structures built will be improved by a stronger base case (e.g. arguments for the 6^{th} jump, or an argument showing no $\emptyset^{(n+3)}$-computable isomorphism between some low$_n$ boolean algebra and any computable copy).

Is there a way to make a variant of this argument that steps from n to $n+1$, or to $n+3$?

Future Directions

The machinery is very general, and the family of structures built will be improved by a stronger base case (e.g. arguments for the 6^{th} jump, or an argument showing no $\emptyset^{(n+3)}$-computable isomorphism between some low$_n$ boolean algebra and any computable copy).

Is there a way to make a variant of this argument that steps from n to $n+1$, or to $n+3$?

Are there any other classes of structures for which translation of diagonalisation allows us to exploit structural similarities to uncover computational similarities?

Future Directions

The machinery is very general, and the family of structures built will be improved by a stronger base case (e.g. arguments for the 6^{th} jump, or an argument showing no $\emptyset^{(n+3)}$-computable isomorphism between some low$_n$ boolean algebra and any computable copy).

Is there a way to make a variant of this argument that steps from n to $n+1$, or to $n+3$?

Are there any other classes of structures for which translation of diagonalisation allows us to exploit structural similarities to uncover computational similarities?

Is there any specific property of Boolean algebras which causes the low$_n$ problem to be hard?
If such a property exists, are there other classes of structures with similar properties? Could the copy/diagonalise game be used to identify them?