Monoidal Categories, Bialgebras, and Automata

James Worthington
Mathematics Department
Cornell University

Binghamton University
Geometry/Topology Seminar
October 29, 2009

Background: Automata

Finite automata

- Model computation with finite memory
- Compute functions called **regular languages**
- Introduced by Kleene in 1951 to model neurons

Finite automata

- Model computation with finite memory
- Compute functions called **regular languages**
- Introduced by Kleene in 1951 to model neurons

Applications

- Logic, computer science
- Geometric group theory [Cannon et. al, 92]
- Number theory [Allouche & Shallit, 03]

Bialgebras

- Capture combination, decomposition
- Introduced by Hopf in 1930 (algebraic topology)

Background: Bialgebras

Bialgebras

- Capture combination, decomposition
- Introduced by Hopf in 1930 (algebraic topology)

Applications

- Hopf Algebras
- Physics (quantum groups) [Drinfel'd, '86]
- Combinatorics [Joni & Rota, '79]

Monoidal categories

- Category with associative operation, unit object
- Introduced Mac Lane, Bénabou in 1953 (independently)

Background: Monoidal Categories

Monoidal categories

- Category with associative operation, unit object
- Introduced Mac Lane, Bénabou in 1953 (independently)

Applications

- Categorical logic
- Quantum protocols [Abramsky & Coecke, '04]
- Thompson's group [Brin, '05]

Proof complexity

- Proof = “feasibly-verifiable witness to truth”
- Proof system = proof-verifying function
- Theorem hard for proof system = all proofs are long

Proof complexity

- Proof = “feasibly-verifiable witness to truth”
- Proof system = proof-verifying function
- Theorem hard for proof system = all proofs are long
- “Are there always hard theorems?” related to outstanding conjectures in complexity theory
- e.g., $NP = \mathrm{co}NP$ iff $\exists$ polynomially-bounded system for propositional tautologies

Background: Proof Complexity

- Examining different proof systems $\Rightarrow$ progress in complexity theory. But...

Background: Proof Complexity

- Examining different proof systems $\Rightarrow$ progress in complexity theory. But...
- Proving a theorem hard is hard
- Finding candidate hard theorems is hard

Background: Proof Complexity

- Examining different proof systems $\Rightarrow$ progress in complexity theory. But...
- Proving a theorem hard is hard
- Finding candidate hard theorems is hard
- One solution: work with systems in which proofs are encoded as well-known mathematical objects
- E.g., Nullstellensatz proof system for tautologies [Beame et. al, '96]

Overview — why category theory?

Main ideas:

- Automata, representations of bialgebras: same definition/constructions, different monoidal categories
- Monoidal categories: natural setting to talk about automata, languages
- Ongoing work: representation theory of bialgebras $\Rightarrow$ complexity-theoretic information about automata

(Symmetric) Monoidal Categories

Monoidal category $\mathcal{C}$:

- **Bifunctor** $\otimes : \mathcal{C} \times \mathcal{C} \to \mathcal{C}$
- **Associator**: natural isomorphism

$$a : X \otimes (Y \otimes Z) \cong (X \otimes Y) \otimes Z$$

- **Unit object** E and natural isomorphisms

$$l : E \otimes X \cong X \qquad r : X \otimes E \cong X$$

- **Symmetry**: natural isomorphism (involution)

$$\sigma : X \otimes Y \cong Y \otimes X$$

Pentagonal Diagram

Associator satisfies **pentagon condition**:

$$
\begin{array}{ccccc}
W \otimes (X \otimes (Y \otimes Z)) & \xrightarrow{a} & (W \otimes X) \otimes (Y \otimes Z) & \xrightarrow{a} & ((W \otimes X) \otimes Y) \otimes Z \\
\downarrow{\scriptstyle 1 \otimes a} & & & & \uparrow{\scriptstyle a \otimes 1} \\
W \otimes ((X \otimes Y) \otimes Z) & & \xrightarrow{\qquad\qquad a \qquad\qquad} & & (W \otimes (X \otimes Y)) \otimes Z
\end{array}
$$

- Associativity at level of **objects**
- in **Set**: $(X \times Y) \times Z \neq X \times (Y \times Z)$

$$\langle\langle x, y\rangle, z\rangle \neq \langle x, \langle y, z\rangle\rangle$$

Monoidal Categories: Examples

Examples

- **Set** (sets and functions), $\times$, $\star$
- K-**Mod** (K-semimodules and K-linear maps),$\otimes_K$, K (K a commutative semiring)
- K-**Mod**, $\oplus$, $\{0_K\}$

Monoidal Categories: Examples

Examples

- **Set** (sets and functions), $\times$, $\star$
- K-**Mod** (K-semimodules and K-linear maps),$\otimes_K$, K (K a commutative semiring)
- K-**Mod**, $\oplus$, $\{0_K\}$

Notes:

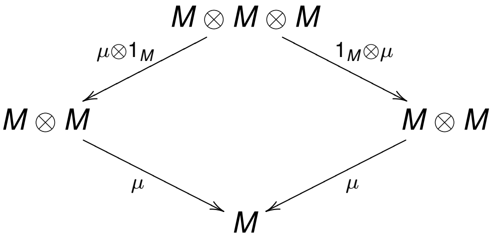
- $\otimes$ not necessarily categorical product
- Semiring = "ring without subtraction"

Monoids in Monoidal Categories

Definition

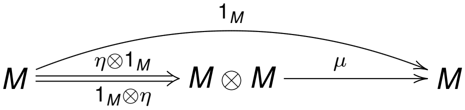
Let $\mathcal{C} = \langle \mathcal{C}, \otimes, E \rangle$ be a monoidal category. A **monoid** $\langle M, \mu, \eta \rangle$ in $\mathcal{C}$ consists of an object M of $\mathcal{C}$ and morphisms $\mu : M \otimes M \to M$, $\eta : E \to M$ satisfying the following diagrams:

Associative multiplication $\mu : M \otimes M \to M$



Monoids in Monoidal Categories

Unit diagram for $\eta : E \to M$



Recall:

$$(M \otimes E) \cong (E \otimes M) \cong M$$

Monoids in Monoidal Categories: Examples

Examples

- Monoids in $\langle \mathbf{Set}, \times, \star \rangle$ = "ordinary" monoids
- Monoids in $\langle \mathbf{Ab}, \otimes_{\mathbb{Z}}, \mathbb{Z} \rangle$ = rings
- Monoids in $\langle \text{K-}\mathbf{Mod}, \otimes_K, K \rangle$ = K-algebras

Note: collections of monoids are themselves categories

Important K-algebra

For remainder of talk:

- K = two-element idempotent semiring
- Underlying set of K = $\{0, 1\}$
- $1 + 1 = 1$

Important K-algebra

For remainder of talk:

- K = two-element idempotent semiring
- Underlying set of K = $\{0, 1\}$
- $1 + 1 = 1$
- P = polynomials over noncommuting variables x, y coefficients in K
- P = formal sums of words in letters x, y
- example element:

$$xyyxy + yx + x$$

Comonoids in Monoidal Categories

Definition

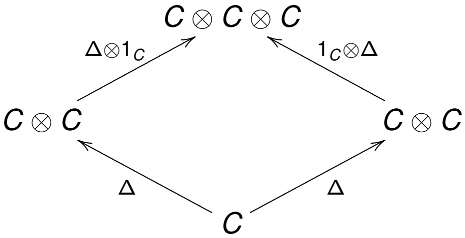
A **comonoid** C in a monoidal category $\mathcal{C}$ is a monoid in $\langle \mathcal{C}^{\mathrm{op}}, \otimes^{\mathrm{op}}, E \rangle$

Comonoids in Monoidal Categories

Definition

A **comonoid** C in a monoidal category $\mathcal{C}$ is a monoid in $\langle \mathcal{C}^{\mathrm{op}}, \otimes^{\mathrm{op}}, E \rangle$

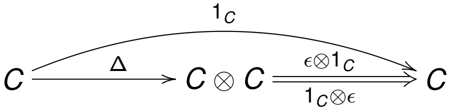
Coassociative comultiplication $\Delta : C \to C \otimes C$



Comultiplication: “splitting up” or “sharing out”
Called “duplicator” in some categorical logics

Comonoids in Monoidal Categories

Counit: map $C \to E$



Called “eraser” in some categorical logics

Comonoids in Monoidal Categories: Example

- $\Delta_P : P \to P \otimes P$ (as element of K-**Mod**)
- $\Delta_P(w) = w \otimes w$ for words w, extended K-linearly
- $\epsilon_C(w) = 1_K$, extended K-linearly

Comonoids in Monoidal Categories: Example

- $\Delta_P : P \to P \otimes P$ (as element of K-**Mod**)
- $\Delta_P(w) = w \otimes w$ for words w, extended K-linearly
- $\epsilon_C(w) = 1_K$, extended K-linearly

Coassociativity of Δ_P:

$$(1_P \otimes \Delta_P) \circ \Delta_P(w) = (1_P \otimes \Delta_P)(w \otimes w) = w \otimes (w \otimes w)$$

$$(\Delta_P \otimes 1_P) \circ \Delta_P(w) = (\Delta_P \otimes 1_P)(w \otimes w) = (w \otimes w) \otimes w$$

Lemma

Let $\mathcal{C}$ be a (locally small) monoidal category, C a comonoid in $\mathcal{C}$, and M a monoid in $\mathcal{C}$. Then $\mathrm{Hom}(C, M)$ is a monoid in **Set***.*

Multiplication: **convolution product**

$$f * g = \mu_M \circ (f \otimes g) \circ \Delta_C$$

- Coassociativity of Δ_C needed for associativity of $*$
- Identity for $*$ is $\eta_M \circ \epsilon_C$

Formal Languages

- Finite alphabet Σ
- Σ^* = set of all finite words over Σ
- Language = subset of Σ^*
- $f : \Sigma^* \to K(= \{0, 1\})$: **formal power series**
- Bijection between languages, formal power series

Operations on Languages

- Union
- Intersection
- Concatenation
- Shuffle

Hom(C, M): Formal Languages

How to describe operations on languages?

Quantify over words:

- $L_1 \cup L_2 = \{w \mid w \in L_1 \text{ or } w \in L_2\}$
- $L_1 \cap L_2 = \{w \mid w \in L_1 \text{ and } w \in L_2\}$
- $L_1 L_2 = \{w \mid w = w_1 w_2,\ w_1 \in L_1,\ w_2 \in L_2\}$

Hom(C, M): Formal Languages

How to describe operations on languages?

Quantify over words:

- $L_1 \cup L_2 = \{w \mid w \in L_1 \text{ or } w \in L_2\}$
- $L_1 \cap L_2 = \{w \mid w \in L_1 \text{ and } w \in L_2\}$
- $L_1 L_2 = \{w \mid w = w_1 w_2,\ w_1 \in L_1,\ w_2 \in L_2\}$

Operations on Formal Power Series:

- Union = pointwise addition
- Intersection = pointwise multiplication
- Concatenation = series product

Hom(C, M): Formal Languages

- $P = K$-algebra of formal sums of words $\in \{x, y\}^*$
- Elements of P^* in one-to-one correspondence with formal languages $\subseteq \{x, y\}^*$
- Intersection, shuffle determined by comultiplication on P (Duchamp et. al [01])

Hom(C, M): Formal Languages

- $P = K$-algebra of formal sums of words $\in \{x, y\}^*$
- Elements of P^* in one-to-one correspondence with formal languages $\subseteq \{x, y\}^*$
- Intersection, shuffle determined by comultiplication on P (Duchamp et. al [01])

- Union, intersection, shuffle, concatenation: convolution products
- Same definition with monoid, comonoid as parameters

Hom(C, M): Formal Languages

$$f * g = \mu_K \circ (f \otimes g) \circ \Delta_P$$

Intersection

- Monoidal Category: (K-**Mod**, $\otimes_K$, K)
- Comonoid: $\Delta_P : P \to P \otimes_K P$
- $\Delta_P(w) = w \otimes w$ extended K-linearly
- $\epsilon_P(w) = 1$, extended K-linearly
- Monoid: K as K-algebra
- $(f * g)(w) = f(w)g(w)$
- Identity = universal language

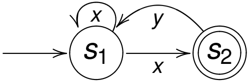
How to describe formal languages?

- Language is an arbitrary subset of Σ^*

How to describe formal languages?

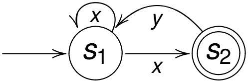
- Language is an arbitrary subset of Σ^*
- m.o. in c.s. — work with finite description of machine which computes (possibly) infinite object
- Machines to compute languages: automata
- Not all languages have finite machine

Nondeterministic Automaton: Example



- Start state = s_1
- Accept state = s_2
- Reads $w \in \{x, y\}^*$ letter by letter
- Each letter causes state transition
- Read y in state s_1: immediately fail

Nondeterministic Automaton: Example



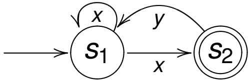
- Runs on xyx:

$$s_1 \xrightarrow{x} s_2 \xrightarrow{y} s_1 \xrightarrow{x} s_1$$

$$s_1 \xrightarrow{x} s_2 \xrightarrow{y} s_1 \xrightarrow{x} s_2$$

- Automaton **accepts** $w \Leftrightarrow$ there is some w-labelled path from a start state to an accept state

Nondeterministic Automaton: Example



- Runs on *xyx*:

$$s_1 \xrightarrow{x} s_2 \xrightarrow{y} s_1 \xrightarrow{x} s_1$$

$$s_1 \xrightarrow{x} s_2 \xrightarrow{y} s_1 \xrightarrow{x} s_2$$

- Automaton **accepts** $w \Leftrightarrow$ there is some w-labelled path from a start state to an accept state

  How to express in a monoidal category?

Actions of Monoids

Transitions = **actions**

Definition

Let $\mathcal{C}$ be a monoidal category and $\langle M, \mu, \eta \rangle$ a monoid in $\mathcal{C}$.

A **right action** of M on $X \in \mathcal{C}$ is an arrow

$$\triangleleft : X \otimes M \to X$$

satisfying:

$$\begin{array}{ccccccc}
(X \otimes M) \otimes M & \xrightarrow{a^{-1}} & X \otimes (M \otimes M) & \xrightarrow{1_X \otimes \mu} & X \otimes M & \xleftarrow{1_X \otimes \eta} & X \otimes E \\
\downarrow{\scriptstyle \triangleleft \otimes 1_M} & & & & \downarrow{\scriptstyle \triangleleft} & & \downarrow{\scriptstyle r} \\
X \otimes M & & \xrightarrow{\qquad\qquad \triangleleft \qquad\qquad} & & X & \xleftarrow{1_X} & X
\end{array}$$

Actions of Monoids

Transitions = **actions**

Definition

Let $\mathcal{C}$ be a monoidal category and $\langle M, \mu, \eta \rangle$ a monoid in $\mathcal{C}$.

A **right action** of M on $X \in \mathcal{C}$ is an arrow

$$\triangleleft : X \otimes M \to X$$

satisfying:

$$\begin{array}{ccccccc}
(X \otimes M) \otimes M & \xrightarrow{a^{-1}} & X \otimes (M \otimes M) & \xrightarrow{1_X \otimes \mu} & X \otimes M & \xleftarrow{1_X \otimes \eta} & X \otimes E \\
\downarrow{\scriptstyle \triangleleft \otimes 1_M} & & & & \downarrow{\scriptstyle \triangleleft} & & \downarrow{\scriptstyle r} \\
X \otimes M & & \xrightarrow{\qquad \triangleleft \qquad} & & X & \xleftarrow{1_X} & X
\end{array}$$

Also called a **representation** of M

K-linear Automata

K-linear automaton

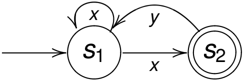
- Pointed, observable representation of K-algebra P
- Input: P
- States: K-semimodule N

K-linear Automata

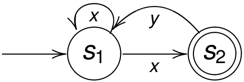
K-linear automaton

- Pointed, observable representation of K-algebra P
- Input: P
- States: K-semimodule N
- Transitions — action $\triangleleft : N \otimes P \to N$
- Pointing: distinguished start state $s \in N$
- Observation — K-linear map $\Omega : N \to K$

Automaton = pointed, observable representation of P:



Automaton = pointed, observable representation of P:



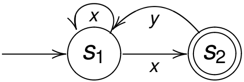
$$\begin{bmatrix} k_1 & k_2 \end{bmatrix} \triangleleft x = \begin{bmatrix} k_1 & k_2 \end{bmatrix} \begin{bmatrix} 1 & 1 \\ 0 & 0 \end{bmatrix}$$

$$\begin{bmatrix} k_1 & k_2 \end{bmatrix} \triangleleft y = \begin{bmatrix} k_1 & k_2 \end{bmatrix} \begin{bmatrix} 0 & 0 \\ 1 & 0 \end{bmatrix}$$

extend algebraically to right action $\begin{bmatrix} k_1 & k_2 \end{bmatrix} \triangleleft P$

Automata and K-algebras

Automaton = pointed, observable representation of P:



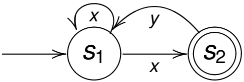
$$\begin{bmatrix} k_1 & k_2 \end{bmatrix} \triangleleft x = \begin{bmatrix} k_1 & k_2 \end{bmatrix} \begin{bmatrix} 1 & 1 \\ 0 & 0 \end{bmatrix}$$

$$\begin{bmatrix} k_1 & k_2 \end{bmatrix} \triangleleft y = \begin{bmatrix} k_1 & k_2 \end{bmatrix} \begin{bmatrix} 0 & 0 \\ 1 & 0 \end{bmatrix}$$

extend algebraically to right action $\begin{bmatrix} k_1 & k_2 \end{bmatrix} \triangleleft P$

Start vector: $\begin{bmatrix} 1 & 0 \end{bmatrix}$

Automaton = pointed, observable representation of P:



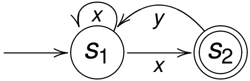
$$\begin{bmatrix} k_1 & k_2 \end{bmatrix} \triangleleft x = \begin{bmatrix} k_1 & k_2 \end{bmatrix} \begin{bmatrix} 1 & 1 \\ 0 & 0 \end{bmatrix}$$

$$\begin{bmatrix} k_1 & k_2 \end{bmatrix} \triangleleft y = \begin{bmatrix} k_1 & k_2 \end{bmatrix} \begin{bmatrix} 0 & 0 \\ 1 & 0 \end{bmatrix}$$

extend algebraically to right action $\begin{bmatrix} k_1 & k_2 \end{bmatrix} \triangleleft P$

Start vector: $\begin{bmatrix} 1 & 0 \end{bmatrix}$

$$\Omega\left(\begin{bmatrix} k_1 & k_2 \end{bmatrix}\right) = \begin{bmatrix} k_1 & k_2 \end{bmatrix} \begin{bmatrix} 0 \\ 1 \end{bmatrix}$$



Run of K-linear automaton on xyx

$$\begin{bmatrix} 1 & 0 \end{bmatrix} \begin{bmatrix} 1 & 1 \\ 0 & 0 \end{bmatrix} \begin{bmatrix} 0 & 0 \\ 1 & 0 \end{bmatrix} \begin{bmatrix} 1 & 1 \\ 0 & 0 \end{bmatrix} \begin{bmatrix} 0 \\ 1 \end{bmatrix} = 1$$

Language Accepted

Definition

Let $D = (N, P, s, \triangleleft, \Omega)$ be a K-linear automaton. The **language accepted** by D is the function

$$\rho_D : P \to K$$

$$\rho_D(p) = \Omega(s \triangleleft p)$$

Note: $\rho_D \in P^*$

Automata in Categories

- Definition can be formulated categorically
- K-**Mod** $\Rightarrow$ nondeterministic automata
- Deterministic automata as representation in $\mathbf{Vec}_F$ [Grossman & Larson, '04]

Putting it all together. . .

So far. . .

- Representations of K-algebra compute languages
- K-coalgebra defines language multiplication

Putting it all together...

So far...

- K-algebra compute languages — Representations of K-algebra compute languages
- K-coalgebra defines language multiplication

Next up...

- K-algebra and K-coalgebra play nice: K**-bialgebra**
- Can multiply representations of K-bialgebra
- Corresponds to running automata in parallel

Definition

A bimonoid B is a monoid in a category of comonoids, or equivalently, a comonoid in a category of monoids.

Definition

A K-bialgebra is a bimonoid “in” K-**Mod**.

Definition

A bimonoid B is a monoid in a category of comonoids, or equivalently, a comonoid in a category of monoids.

Definition

A K-bialgebra is a bimonoid "in" K-**Mod**.

Fact: Category of monoids of **symmetric** monoidal category is itself monoidal

$$A \otimes B \otimes A \otimes B \xrightarrow{1_A \otimes \sigma_{B,A} \otimes 1_B} A \otimes A \otimes B \otimes B \xrightarrow{\mu_A \otimes \mu_B} A \otimes B$$

Bialgebras, Bimonoids

Diagram relating Δ and μ in K-bialgebra B:

$$\begin{array}{ccccc} B \otimes B & \xrightarrow{\mu} & B & \xrightarrow{\Delta} & B \otimes B \\ \downarrow{\scriptstyle \Delta\otimes\Delta} & & & & \uparrow{\scriptstyle \mu\otimes\mu} \\ B \otimes B \otimes B \otimes B & & \xrightarrow{1_B \otimes \sigma \otimes 1_B} & & B \otimes B \otimes B \otimes B \end{array}$$

Multiplying Representations of K-Bialgebras

Have:

- Action $\triangleleft_{N_1} : N_1 \otimes B \to N_1$
- Action $\triangleleft_{N_2} : N_2 \otimes B \to N_2$

Want action $\triangleleft : N_1 \otimes N_2 \otimes B \to N_1 \otimes N_2$

Multiplying Representations of K-Bialgebras

Have:

- Action $\triangleleft_{N_1} : N_1 \otimes B \to N_1$
- Action $\triangleleft_{N_2} : N_2 \otimes B \to N_2$

Want action $\triangleleft : N_1 \otimes N_2 \otimes B \to N_1 \otimes N_2$

Definition ($\triangleleft : N_1 \otimes N_2 \otimes B \to N_1 \otimes N_2$)

$$N_1 \otimes N_2 \otimes B \xrightarrow{1 \otimes 1 \otimes \Delta_B} N_1 \otimes N_2 \otimes B \otimes B \xrightarrow{1 \otimes \sigma \otimes 1} N_1 \otimes B \otimes N_2 \otimes B \xrightarrow{\triangleleft_{N_1} \otimes \triangleleft_{N_2}} N_1 \otimes N_2$$

- Representations form monoidal category
- Unit object $\Rightarrow$ unit representation
- Instance of theorem about bimonoids

Multiplying Automata

Definition

Let D and E be K-linear automata. Then $D \otimes E$ is a K-linear automaton with:

- Transitions: multiply actions
- $s_{D \otimes E} = s_D \otimes s_E$
- $\Omega_{D \otimes E} = \Omega_D \otimes \Omega_E$

- "Run automata in parallel"
- Δ as parameter (intersection, shuffle)

Multiplying Automata

Definition

Let D and E be K-linear automata. Then $D \otimes E$ is a K-linear automaton with:

- Transitions: multiply actions
- $s_{D\otimes E} = s_D \otimes s_E$
- $\Omega_{D\otimes E} = \Omega_D \otimes \Omega_E$

- "Run automata in parallel"
- Δ as parameter (intersection, shuffle)

Theorem (W. '09)

$\rho_D * \rho_E = \rho_{D\otimes E}$

Morphisms of Actions

What about morphisms of actions?

Morphisms of Actions

What about morphisms of actions?

Definition

Let M be a monoid in $\mathcal{C}$ and let X, X' be objects of $\mathcal{C}$. Let $\triangleleft$ and $\triangleleft'$ be right actions of M on X, X', respectively.
A **morphism of right actions** is an arrow $f : X \to X'$ in $\mathcal{C}$ such that

$$\begin{array}{ccc} X \otimes M & \xrightarrow{f \otimes 1_M} & X' \otimes M \\ \downarrow{\scriptstyle \triangleleft} & & \downarrow{\scriptstyle \triangleleft'} \\ X & \xrightarrow{f} & X' \end{array}$$

Morphisms of Automata

Definition

Let D and E be K-linear automata. A K-linear map $\phi : D \to E$ is a **morphism of K-linear automata** if it satisfies:

$$
\begin{array}{ccc}
K & \xrightarrow{\alpha_D} & D \\
 & {\scriptstyle \alpha_E}\searrow & \downarrow{\scriptstyle \phi} \\
 & & E
\end{array}
\qquad
\begin{array}{ccc}
D & \xrightarrow{\triangleleft_D} & D \\
{\scriptstyle \phi}\downarrow & & \downarrow{\scriptstyle \phi} \\
E & \xrightarrow[\triangleleft_E]{} & E
\end{array}
\qquad
\begin{array}{ccc}
D & \xrightarrow{\Omega_D} & K \\
{\scriptstyle \phi}\downarrow & \nearrow{\scriptstyle \Omega_E} & \\
E & &
\end{array}
$$

- α_D, α_E: pointings
- Ω_D, Ω_E: observations

Morphisms as Proofs: Soundness

Definition

Automata D and E are **equivalent**: $\rho_D = \rho_E$.

Morphisms as Proofs: Soundness

Definition

Automata D and E are **equivalent**: $\rho_D = \rho_E$.

Theorem

Let D and E be K-linear automata. If there is a morphism of K-linear automata $\phi : D \to E$, then $\rho_D = \rho_E$.

For any $p \in P$,

$$\begin{aligned}\Omega_D(\alpha_D(1) \triangleleft_D p) &= \Omega_E(\phi(\alpha_D(1) \triangleleft_D p)) \\ &= \Omega_E(\phi(\alpha_D(1)) \triangleleft_E p) \\ &= \Omega_E(\alpha_E(1) \triangleleft_E p)\end{aligned}$$

Morphisms as Proofs: Completeness

Theorem (W. '09)

Let D and E be two equivalent K-linear automata. Then:

- *There is a sequence of K-linear automata and morphisms of K-linear automata which witnesses the equivalence.*
- *If D, E correspond to finite nfa, sequence can be constructed in PSPACE*

Proof uses:

- Adjunction between K-linear automata, “deterministic” automata
- Uniqueness of minimal deterministic automaton

Ongoing Work: Bialgebras $\Rightarrow$ Automata

- Equivalence of K-linear automata is *PSPACE*-complete
- Hard equivalences for proof system (unless $NP = PSPACE$)
- Find them, along with "useful" easy equivalences
- Use representation theory: understand how automata are put together to understand how proofs are put together

Ongoing Work: Bialgebras $\Rightarrow$ Automata

A: nfa with n^2 states

- Deterministic algorithm to decide whether A accepts every word requires n^4 many worktape cells of TM
- If $A = B \otimes C$ and B, C each have n states, only need n^2 cells (comultiplication for intersection)
- Can multiply proofs in certain instances

Thank You!