# Towards a Coalgebraic Chomsky Hierarchy

**Sergey Goncharov**, Stefan Milius, Alexandra Silva

IFIP WG 1.3, Sinaia, 2–3.09.2014

FRIEDRICH-ALEXANDER
UNIVERSITÄT
ERLANGEN-NÜRNBERG
TECHNISCHE FAKULTÄT

# Short Histrory of Coalgebraic Invasion to Automata Theory

- Deterministic automata as coalgebras **[Rutten, 1998]**.
- Generalized regular expressions and Kleene's theorem for (Kripke-)polynomial functors **[Silva, 2010]**.
- Generalized powerset construction **[Silva et al., 2010]**.
- Regular expressions for equationally presented functors and monads **[Myers, 2013]**.
- Context-free languages, coalgebraically **[Winter et al., 2013]**.

# Moore Automata, Coalgebraically

Moore automaton with input alphabet $A$ and output alphabet $B$ is given by

$$t^{m} : X \times A \to X \quad \text{(transition)} \qquad \text{and} \qquad o^{m} : X \to B \quad \text{(output)}$$

Thus, a Moore automaton is a coalgebra $m : X \to B \times X^A$ on **Set**.

We shall assume these finite

Moore automaton with input alphabet $A$ and output alphabet $B$ is given by

$$t^{m} : X \times A \to X \quad \text{(transition)} \qquad \text{and} \qquad o^{m} : X \to B \quad \text{(output)}$$

Thus, a Moore automaton is a coalgebra $m : X \to B \times X^{A}$ on **Set**.

We shall assume these finite

Moore automaton with input alphabet $A$ and output alphabet $B$ is given by

$$t^{m} : X \times A \to X \quad \text{(transition)} \qquad \text{and} \qquad o^{m} : X \to B \quad \text{(output)}$$

Thus, a Moore automaton is a coalgebra $m : X \to B \times X^A$ on **Set**.

A final $L_{A,B}$ -coalgebra is carried by the set $B^{A^*}$ of formal power series on $B$ with coalgebra structure $\langle o, t \rangle : B^{A^*} \to B \times (B^{A^*})^A$

$$o(\sigma : A^* \to B) = \sigma(\varepsilon) \qquad \text{and} \qquad t(\sigma : A^* \to B, a) = \lambda w.\sigma(a \cdot w).$$

$L_{A,B}X := B \times X^A$

We shall assume these finite

Moore automaton with input alphabet $A$ and output alphabet $B$ is given by

$$t^{m} : X \times A \to X \quad \text{(transition)} \qquad \text{and} \qquad o^{m} : X \to B \quad \text{(output)}$$

Thus, a Moore automaton is a coalgebra $m : X \to B \times X^A$ on **Set**.

A final $L_{A,B}$-coalgebra is carried by the set $B^{A^*}$ of formal power series on $B$ with coalgebra structure $\langle o, t \rangle : B^{A^*} \to B \times (B^{A^*})^A$

$$o(\sigma : A^* \to B) = \sigma(\varepsilon) \qquad \text{and} \qquad t(\sigma : A^* \to B, a) = \lambda w.\sigma(a \cdot w).$$

$L_{A,B}X := B \times X^A$

We shall assume these finite

Moore automaton with input alphabet $A$ and output alphabet $B$ is given by

$$t^{m} : X \times A \to X \quad \text{(transition)} \qquad \text{and} \qquad o^{m} : X \to B \quad \text{(output)}$$

Thus, a Moore automaton is a coalgebra $m : X \to B \times X^A$ on **Set**.

A final $L_{A,B}$-coalgebra is carried by the set $B^{A^*}$ of formal power series on $B$ with coalgebra structure $\langle o, t \rangle : B^{A^*} \to B \times (B^{A^*})^A$

$$o(\sigma : A^* \to B) = \sigma(\varepsilon) \qquad \text{and} \qquad t(\sigma : A^* \to B, a) = \lambda w.\sigma(a \cdot w).$$

(Brzozowski) Derivatives: given $w \in A^*$,

$$\partial_\varepsilon(\sigma) = \sigma \qquad \text{and} \qquad \partial_{a \cdot w}(\sigma) = t(\partial_w(\sigma), a)$$

If $B = 2$ then $B^{A^*} \simeq \mathcal{P}(A^*)$ is the set of all formal languages over $A$.

# Regularity of Formal Power Series

By the the universal property of the final coalgebra for any $m : X \to B \times X^A$ there exists a unique $L_{A,B}$-coalgebra homomorphism $\widehat{m}$ such that:

$$\begin{array}{ccc} X & \xrightarrow{\ \widehat{m}\ } & B^{A^*} \\ {\scriptstyle m}\downarrow & & \downarrow{\scriptstyle \iota} \\ B \times X^A & \xrightarrow[\ \mathrm{id} \times \widehat{m}^A\ ]{} & B \times (B^{A^*})^A \end{array}$$

Given $x \in X$, $[\![x]\!]_m := \widehat{m}(x)$ is the "language" recognized by $m$ at $x$.

A formal power series $\sigma : A^* \to B$ is regular if $\{\partial_w(\sigma) \mid w \in A^*\}$ is finite.

# Regularity of Formal Power Series

By the the universal property of the final coalgebra for any $m : X \to B \times X^A$ there exists a unique $L_{A,B}$-coalgebra homomorphism $\widehat{m}$ such that:

$$\begin{array}{ccc} X & \xrightarrow{\ \widehat{m}\ } & B^{A^*} \\ {\scriptstyle m}\downarrow & & \downarrow{\scriptstyle \iota} \\ B \times X^A & \xrightarrow[\ \mathrm{id} \times \widehat{m}^A\ ]{} & B \times (B^{A^*})^A \end{array}$$

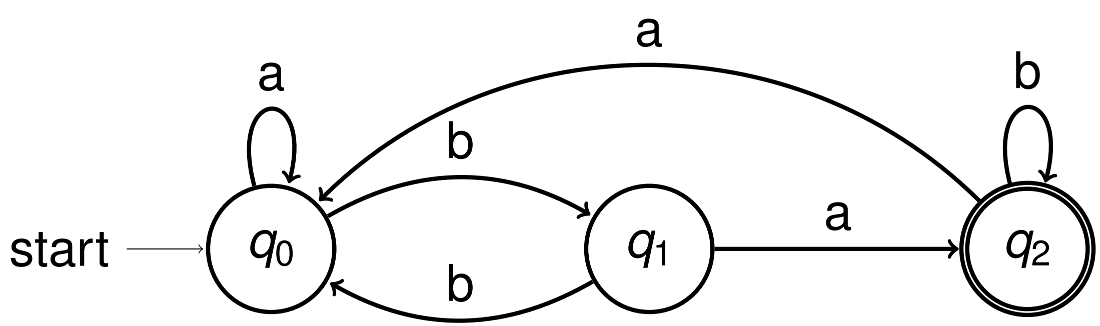
Given $x \in X$, $[\![x]\!]_m := \widehat{m}(x)$ is the “language” recognized by $m$ at $x$.

A formal power series $\sigma : A^* \to B$ is regular if $\{\partial_w(\sigma) \mid w \in A^*\}$ is finite.

**Theorem.** A formal power series $\sigma$ is regular iff $\sigma = [\![x]\!]_{\widehat{m}}$ for some $m : X \to B \times X^A$ and $x \in X$.

# Regular Expressions, Coalgebraically

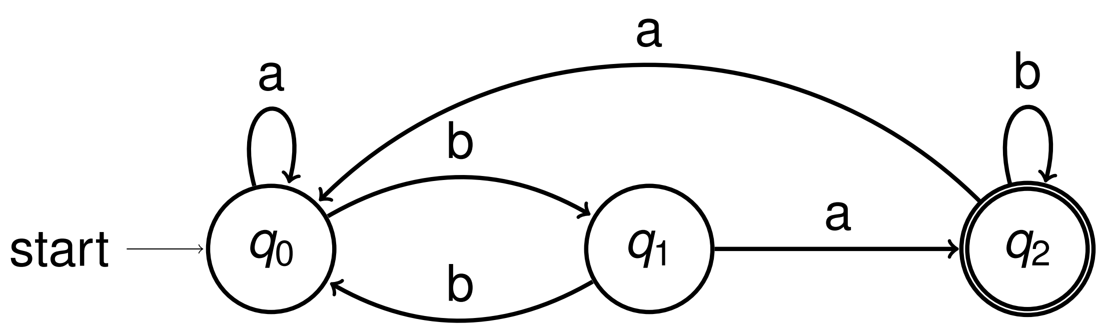
$B = \{\top, \bot\}$




$$\begin{cases} e_0 = a.e_0 \pitchfork b.e_1 \pitchfork \bot \\ e_1 = a.e_2 \pitchfork b.e_0 \pitchfork \bot \\ e_2 = a.e_0 \pitchfork b.e_2 \pitchfork \top \end{cases}$$

# Regular Expressions, Coalgebraically

$B = \{\top, \bot\}$



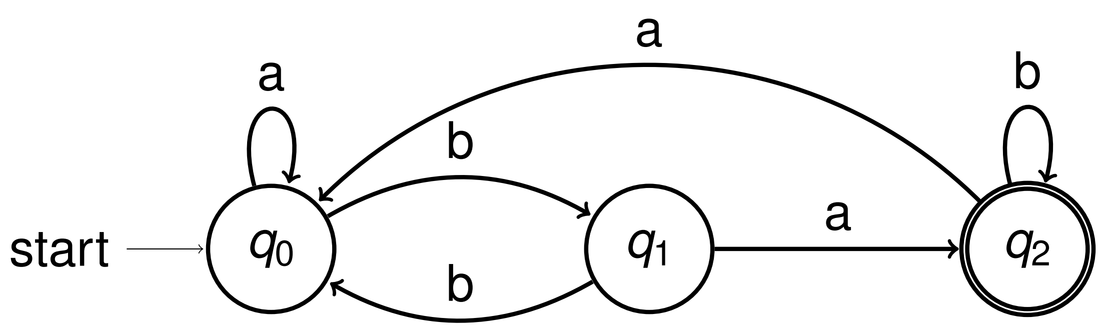

$$\begin{cases} e_0 = a.e_0 \pitchfork b.e_1 \pitchfork \bot \\ e_1 = a.e_2 \pitchfork b.e_0 \pitchfork \bot \\ e_2 = a.e_0 \pitchfork b.e_2 \pitchfork \top \end{cases}$$

the same
a la Kleene:

$$\begin{cases} e_0 = ae_0 + be_1 \\ e_1 = ae_2 + be_0 \\ e_2 = ae_0 + be_2 + 1 \end{cases}$$

# Regular Expressions, Coalgebraically

$B = \{\top, \bot\}$




$$\begin{cases} e_0 = a.e_0 \pitchfork b.e_1 \pitchfork \bot \\ e_1 = a.e_2 \pitchfork b.e_0 \pitchfork \bot \\ e_2 = a.e_0 \pitchfork b.e_2 \pitchfork \top \end{cases}$$

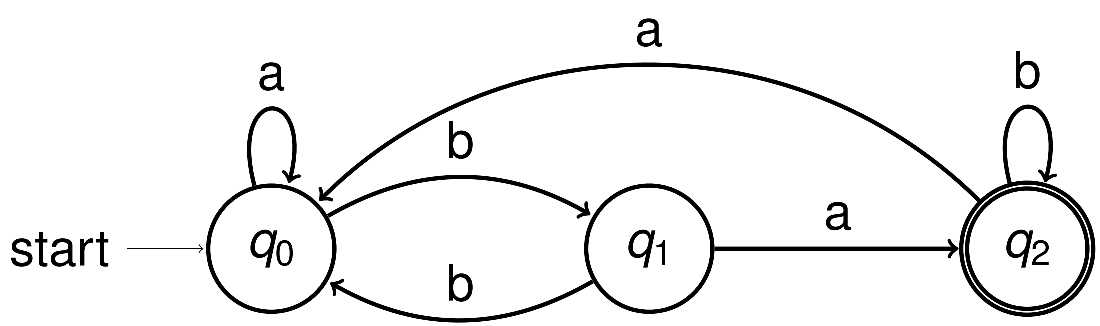
the same
a la Kleene:

$$\begin{cases} e_0 = ae_0 + be_1 \\ e_1 = ae_2 + be_0 \\ e_2 = ae_0 + be_2 + 1 \end{cases}$$

Equivalently, $e_0 = \mu x.\,(a.x \pitchfork b.\mu y.\,(a.\mu z.\,(a.x \pitchfork b.z \pitchfork \top) \pitchfork b.x \pitchfork \bot) \pitchfork \bot)$

# Regular Expressions, Coalgebraically

$B = \{\top, \bot\}$




$$\begin{cases} e_0 = a.e_0 \pitchfork b.e_1 \pitchfork \bot \\ e_1 = a.e_2 \pitchfork b.e_0 \pitchfork \bot \\ e_2 = a.e_0 \pitchfork b.e_2 \pitchfork \top \end{cases}$$

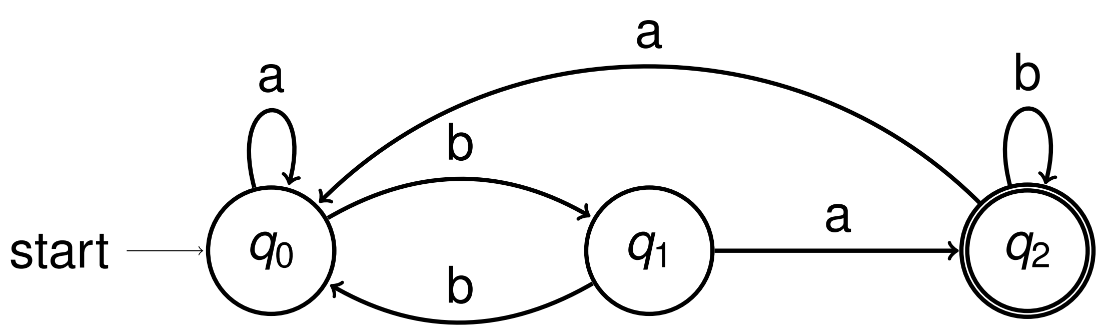
the same
a la Kleene:

$$\begin{cases} e_0 = ae_0 + be_1 \\ e_1 = ae_2 + be_0 \\ e_2 = ae_0 + be_2 + 1 \end{cases}$$

Equivalently, $e_0 = \mu x. \Big(ax + b\mu y. \Big(a\ \mu z. (ax + bz + 1)\ + bx\Big)\Big)$

# Regular Expressions, Coalgebraically

$B = \{\top, \bot\}$




$$\begin{cases} e_0 = a.e_0 \pitchfork b.e_1 \pitchfork \bot \\ e_1 = a.e_2 \pitchfork b.e_0 \pitchfork \bot \\ e_2 = a.e_0 \pitchfork b.e_2 \pitchfork \top \end{cases}$$

the same a la Kleene:

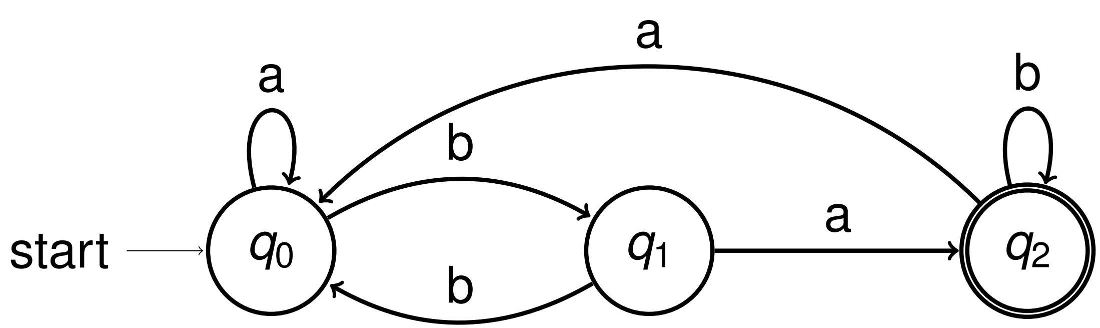
$$\begin{cases} e_0 = ae_0 + be_1 \\ e_1 = ae_2 + be_0 \\ e_2 = ae_0 + be_2 + 1 \end{cases}$$

Equivalently, $e_0 = \mu x.\, \Big(ax + b\mu y.\, \big(a\, \mu z.\, (ax + bz + 1) + bx\big)\Big)$

Using $\mu x.\, (tx + s) \mapsto t^* s$

# Regular Expressions, Coalgebraically

$B = \{\top, \bot\}$




$$\begin{cases} e_0 = a.e_0 \pitchfork b.e_1 \pitchfork \bot \\ e_1 = a.e_2 \pitchfork b.e_0 \pitchfork \bot \\ e_2 = a.e_0 \pitchfork b.e_2 \pitchfork \top \end{cases}$$

the same
a la Kleene:

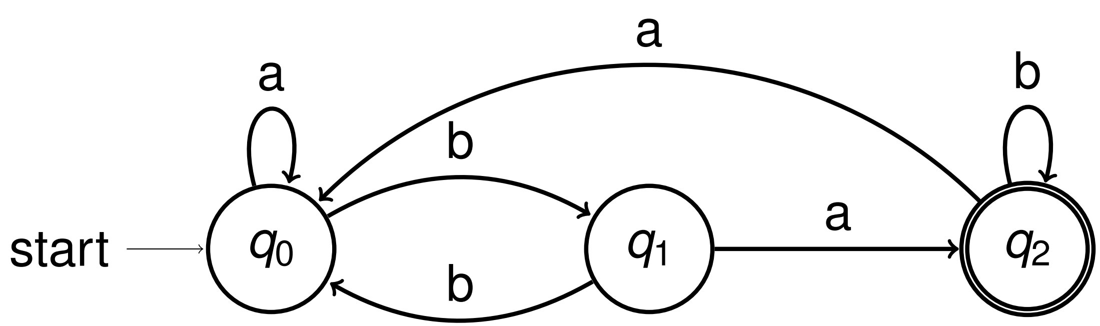
$$\begin{cases} e_0 = ae_0 + be_1 \\ e_1 = ae_2 + be_0 \\ e_2 = ae_0 + be_2 + 1 \end{cases}$$

Equivalently, $e_0 = \mu x.\, \Big( ax + b\ \mu y.\, (ab^*(ax + 1) + bx) \Big)$

Using $\mu x.\, (tx + s) \mapsto t^* s$

# Regular Expressions, Coalgebraically

$B = \{\top, \bot\}$




$$\begin{cases} e_0 = a.e_0 \pitchfork b.e_1 \pitchfork \bot \\ e_1 = a.e_2 \pitchfork b.e_0 \pitchfork \bot \\ e_2 = a.e_0 \pitchfork b.e_2 \pitchfork \top \end{cases}$$

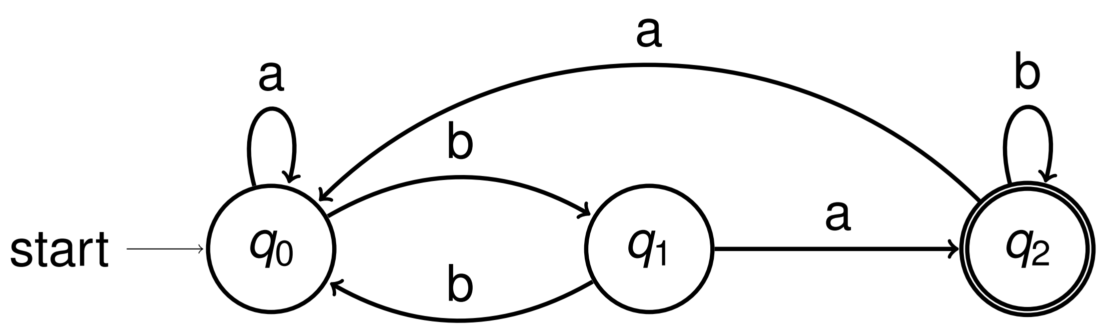
the same
a la Kleene:

$$\begin{cases} e_0 = ae_0 + be_1 \\ e_1 = ae_2 + be_0 \\ e_2 = ae_0 + be_2 + 1 \end{cases}$$

Equivalently, $e_0 = \mu x. \Big( ax + b\, \mu y. \big(ab^*(ax + 1) + bx\big) \Big)$

# Regular Expressions, Coalgebraically

$B = \{\top, \bot\}$




$$\begin{cases} e_0 = a.e_0 \pitchfork b.e_1 \pitchfork \bot \\ e_1 = a.e_2 \pitchfork b.e_0 \pitchfork \bot \\ e_2 = a.e_0 \pitchfork b.e_2 \pitchfork \top \end{cases}$$

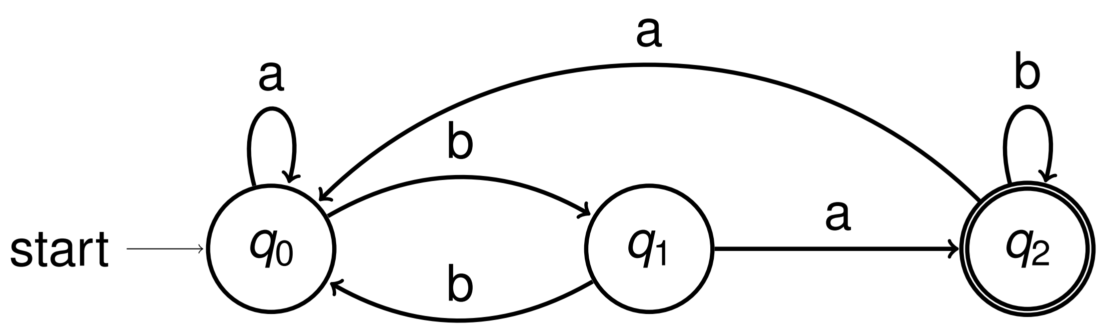
the same
a la Kleene:

$$\begin{cases} e_0 = ae_0 + be_1 \\ e_1 = ae_2 + be_0 \\ e_2 = ae_0 + be_2 + 1 \end{cases}$$

Equivalently, $e_0 = \mu x.\,(ax + b(ab^*(ax + 1) + bx))$

# Regular Expressions, Coalgebraically

$B = \{\top, \bot\}$




$$\begin{cases} e_0 = a.e_0 \pitchfork b.e_1 \pitchfork \bot \\ e_1 = a.e_2 \pitchfork b.e_0 \pitchfork \bot \\ e_2 = a.e_0 \pitchfork b.e_2 \pitchfork \top \end{cases}$$

the same
a la Kleene:

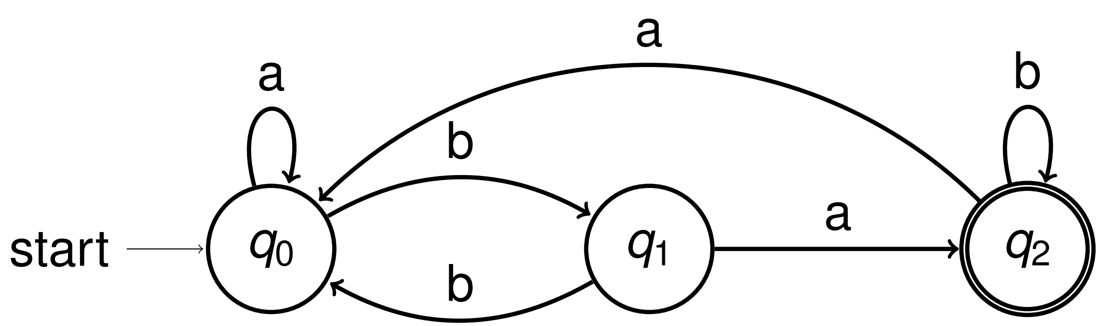
$$\begin{cases} e_0 = ae_0 + be_1 \\ e_1 = ae_2 + be_0 \\ e_2 = ae_0 + be_2 + 1 \end{cases}$$

Equivalently, $e_0 = \mu x.\,(ax + b(ab^*(ax + 1) + bx))$

Using $\mu x.\,(tx + s) \mapsto t^* s$

# Regular Expressions, Coalgebraically

$B = \{\top, \bot\}$




$$\begin{cases} e_0 = a.e_0 \pitchfork b.e_1 \pitchfork \bot \\ e_1 = a.e_2 \pitchfork b.e_0 \pitchfork \bot \\ e_2 = a.e_0 \pitchfork b.e_2 \pitchfork \top \end{cases}$$

the same
a la Kleene:

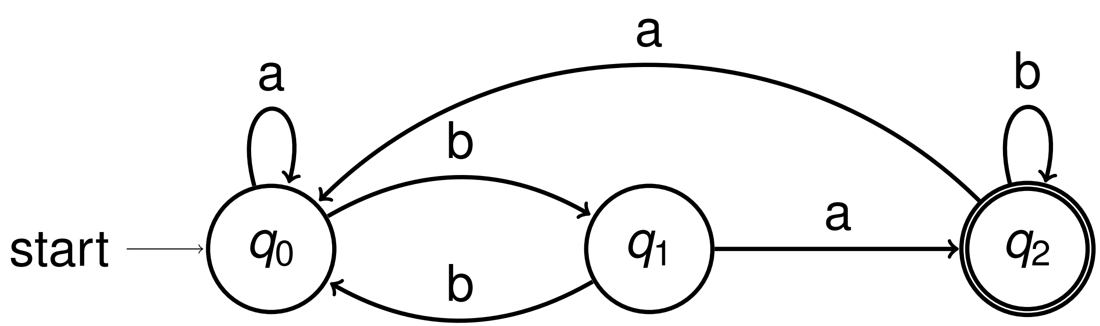
$$\begin{cases} e_0 = ae_0 + be_1 \\ e_1 = ae_2 + be_0 \\ e_2 = ae_0 + be_2 + 1 \end{cases}$$

Equivalently, $e_0 = (a + bab^*a + b)^*bab^*$

Using $\mu x.\,(tx + s) \mapsto t^*s$

# Regular Expressions, Coalgebraically

$B = \{\top, \bot\}$




$$\begin{cases} e_0 = a.e_0 \pitchfork b.e_1 \pitchfork \bot \\ e_1 = a.e_2 \pitchfork b.e_0 \pitchfork \bot \\ e_2 = a.e_0 \pitchfork b.e_2 \pitchfork \top \end{cases}$$

the same
a la Kleene:

$$\begin{cases} e_0 = ae_0 + be_1 \\ e_1 = ae_2 + be_0 \\ e_2 = ae_0 + be_2 + 1 \end{cases}$$

Equivalently, $e_0 = (a + bab^*a + b)^*bab^*$

# Monads

**Mac Lane:** A monad $\mathbb{T}$ is just a monoid in the category of endofunctors

**Mac Lane:** A monad $\mathbb{T}$ is just a monoid in the category of endofunctors

**Moggi:** A monad $\mathbb{T}$ is a (generalized) computational effect

**Moggi:** A monad $\mathbb{T}$ is a (generalized) computational effect

**Examples:**

- (finitary) nondeterminism: $TX = \mathcal{P}_\omega X$; "nondeterministic functions" $A \to \mathcal{P}_\omega B$ are relations
- probabilistic nondeterminism: $TX = \{\rho : X \to [0,1] \mid \sum \rho = 1\}$; "probabilistic functions" $A \to TB$ are "probabilistic relations"
- (finite) background store: $TX = (X \times S)^S$; side-effecting functions $A \to TB$ are functions $A \times S \to B \times S$

# Computational Metalanguage

Any monad $\mathbb{T}$ supports

- inclusion of a value into a computation $\eta : X \to TX$ (unit)
- Kleisli lifting $(f : X \to TY) \mapsto (f^* : TX \to TY)$

# Computational Metalanguage

Any monad $\mathbb{T}$ supports

- inclusion of a value into a computation $\eta : X \to TX$ (unit)
- Kleisli lifting $(f : X \to TY) \mapsto (f^* : TX \to TY)$

Alternatively, a monad is a type constructor supporting

$$\frac{\Gamma \vdash x : A}{\Gamma \vdash \mathrm{ret}\, x : TA} \qquad \frac{\Gamma \vdash p : TA \quad \Gamma, x : A \vdash TB}{\Gamma \vdash \mathrm{do}\; x \leftarrow p; q : TB}$$

# Computational Metalanguage

Any monad $\mathbb{T}$ supports

- inclusion of a value into a computation $\eta : X \to TX$ (unit)
- Kleisli lifting $(f : X \to TY) \mapsto (f^* : TX \to TY)$

Alternatively, a monad is a type constructor supporting

$$\frac{\Gamma \vdash x : A}{\Gamma \vdash \mathsf{ret}\, x : TA} \qquad \frac{\Gamma \vdash p : TA \quad \Gamma, x : A \vdash TB}{\Gamma \vdash \mathsf{do}\; x \leftarrow p; q : TB}$$

This is useful for

- writing programs, e.g. do $x \leftarrow$ **toss**; *if* $(x > 0)$ *then* $y \leftarrow$ **toss**; ret $y$ *else* ret $x$
- expressing metaproperties, such as commutativity

$$\mathsf{do}\; x \leftarrow p; y \leftarrow q; \mathsf{ret}\langle x, y\rangle = \mathsf{do}\; y \leftarrow q; x \leftarrow p; \mathsf{ret}\langle x, y\rangle$$

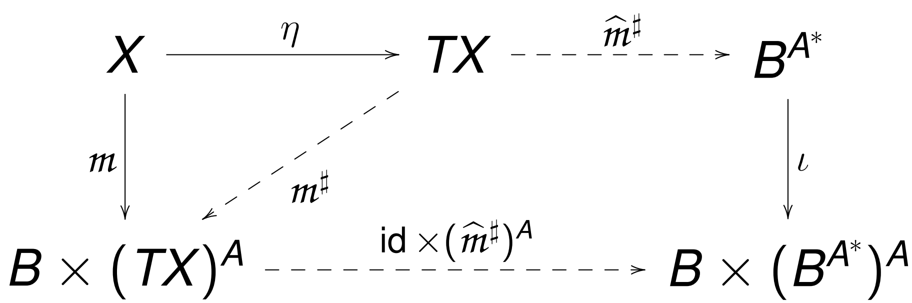
# Generalized Powerset Construction and $\mathbb{T}$-automata

**Definition:** Given a monad $\mathbb{T}$, a $\mathbb{T}$-automaton is a triple of maps

$$o^{m} : X \to B, \qquad t^{m} : X \times A \to TX, \qquad a^{m} : TB \to B$$

where $a^{m}$ is a $\mathbb{T}$-algebra.

Essentially, a $\mathbb{T}$-automaton is a coalgebra $m : X \to B \times (TX)^A$.




Factorization $m = m^\sharp \eta$ is unique and we put $[\![x]\!]_{m} = [\![\eta(x)]\!]_{m^\sharp}$.

**Theorem:** If $B$ is finite then $[\![x]\!]_{m}$ is regular.

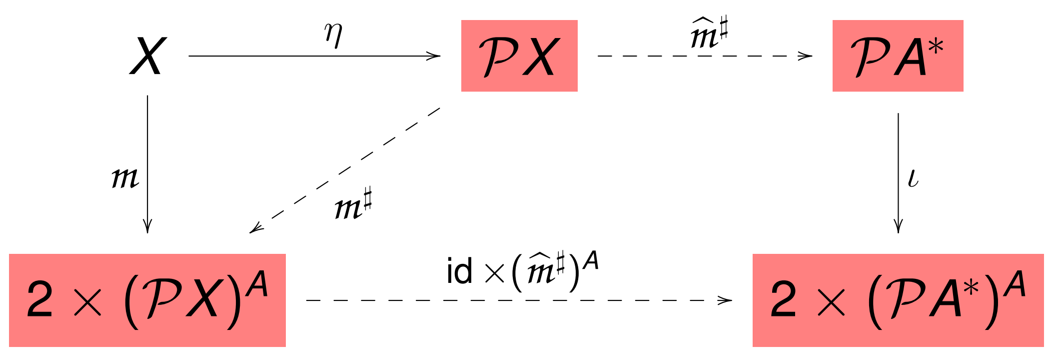
# Generalized Powerset Construction and $\mathbb{T}$-automata

**Definition:** Given a monad $\mathbb{T}$, a $\mathbb{T}$-automaton is a triple of maps

$$o^m : X \to B, \qquad t^m : X \times A \to TX, \qquad a^m : TB \to B$$

where $a^m$ is a $\mathbb{T}$-algebra.

Essentially, a $\mathbb{T}$-automaton is a coalgebra $m : X \to B \times (TX)^A$.




Factorization $m = m^\sharp \eta$ is unique and we put $[\![x]\!]_m = [\![\eta(x)]\!]_{m^\sharp}$.

**Theorem:** If $B$ is finite then $[\![x]\!]_m$ is regular.

# How $\mathbb{T}$-automata Differ from Classical Automata?

- in $m : X \to B \times (TX)^A$, $A$ is reserved for the input tape
- $\mathbb{T}$-automata are real-time, that is no internal transitions
- $\mathbb{T}$ enforces compositionality of computation steps: single steps are indistinguishable from finite sequences of steps

# Algebraic Theories

**Moggi:** A monad is a (generalized) computational effect

**Moggi:** A monad is a (generalized) computational effect

**Plotkin & Power:** A monad is a (generalized) algebraic theory

## **Plotkin & Power:** A monad is a (generalized) algebraic theory

An algebraic theory $\mathcal{E}$ is given by a signature $\Sigma$ and a set of equations.

Any $\mathcal{E}$ defines a monad (converse is true for finitary monads):

- $T_{\mathcal{E}}X =$ 'set of $\Sigma$-terms over $X$ modulo $\mathcal{E}$';
- $\eta$ coerces a variable to a term;
- $\sigma^*(t)$ applies substitution $\sigma : X \to T_{\mathcal{E}}Y$ to $t : T_{\mathcal{E}}X$.

**Example:** Finite powerset monad $\mathcal{P}_\omega \iff$ join semilattices with bottom.

**Example:** Finite probability distributions $\mathcal{D}_\omega \iff$ barycentric algebras; $\Sigma = \{+_p \mid p \in [0,1]\}$, satisfying e.g. "associativity":

$$(x +_p y) +_q z = x +_{p/(p+q-pq)} (y +_{p+q-pq} z)$$

# Stack Theory

Stack theory is given by **pop** : $X^{n+1} \to X$ and **push**$_i$ : $X \to X$ ($i \leq n$):

$$\begin{aligned}
\mathbf{push}_i(\mathbf{pop}(x_1, \ldots, x_n, y)) &= x_i \\
\mathbf{pop}(\mathbf{push}_1(x), \ldots, \mathbf{push}_n(x), x) &= x \\
\mathbf{pop}(x_1, \ldots, x_n, \mathbf{pop}(y_1, \ldots, y_n, z)) &= \mathbf{pop}(x_1, \ldots, x_n, z)
\end{aligned}$$

# Stack Theory

Stack theory is given by $\overline{\textbf{pop}} : 1 \to T(\lfloor n \rfloor + 1)$ and $\overline{\textbf{push}} : n \to T1$:

$$\begin{aligned}
\text{do } \overline{\textbf{push}}(\gamma_i); \overline{\textbf{pop}} &= \text{ret inl } \gamma_i \\
\text{do } x \leftarrow \overline{\textbf{pop}}; \text{case } x \text{ of inl } \gamma_i \mapsto \overline{\textbf{push}}(\gamma_i); \text{inr} \star \mapsto \text{ret} \star &= \text{ret} \star \\
\text{do } x \leftarrow \overline{\textbf{pop}}; \text{case } x \text{ of inl } \gamma_i \mapsto \text{ret inl } \gamma_i; \text{inr} \star \mapsto \overline{\textbf{pop}} &= \overline{\textbf{pop}}
\end{aligned}$$

$\Gamma = \{\gamma_1, \dots, \gamma_n\}$ is stack alphabet

# Stack Theory

Stack theory is given by $\overline{\textbf{pop}} : 1 \to T(\lfloor n \rfloor + 1)$ and $\overline{\textbf{push}} : n \to T1$:

$$\begin{aligned} \text{do } \overline{\textbf{push}}(\gamma_i); \overline{\textbf{pop}} &= \text{ret inl } \gamma_i \\ \text{do } x \leftarrow \overline{\textbf{pop}}; \text{case } x \text{ of inl } \gamma_i \mapsto \overline{\textbf{push}}(\gamma_i); \text{inr} \star \mapsto \text{ret} \star &= \text{ret} \star \\ \text{do } x \leftarrow \overline{\textbf{pop}}; \text{case } x \text{ of inl } \gamma_i \mapsto \text{ret inl } \gamma_i; \text{inr} \star \mapsto \overline{\textbf{pop}} &= \overline{\textbf{pop}} \end{aligned}$$

**Theorem:** stack theory induces the stack monad, a submonad of the store monad $\Gamma^* \to (X \times \Gamma^*)$: $\langle r, t \rangle : \Gamma^* \to (X \times \Gamma^*)$ is in $TX$ iff

$$r(u \cdot w) = r(u) \qquad\qquad t(u \cdot w) = t(u) \cdot w$$

whenever $|u| > k$ for some $k$.

# Nondeterministic Stacks

Nondeterministic stack theory is the tensor product of the stack theory with $\mathcal{P}_\omega$. It consists of

*stack equations:*

$$\begin{aligned} \mathbf{push}_i(\mathbf{pop}(x_1,\dots,x_n,y)) &= x_i \\ \mathbf{pop}(\mathbf{push}_1(x),\dots,\mathbf{push}_n(x),x) &= x \\ \mathbf{pop}(x_1,\dots,x_n,\mathbf{pop}(y_1,\dots,y_n,z)) &= \mathbf{pop}(x_1,\dots,x_n,z) \end{aligned}$$

*semilattice equations:*

$$(x+y)+z = x+(y+z) \qquad x+y = y+x \qquad x+\varnothing = x+x = x$$

*tensor laws:*

$$\mathbf{pop}(\varnothing) = \varnothing \qquad \mathbf{push}(\varnothing,\dots,\varnothing,\varnothing) = \varnothing \qquad \mathbf{pop}(x+x') = \mathbf{pop}(x)+\mathbf{pop}(x')$$

$$\mathbf{push}(x_1+x_1',\dots,x_n+x_n',y+y') = \mathbf{push}(x_1,\dots,x_n,y)+\mathbf{push}(x_1',\dots,x_n',y')$$

# Reactive Expressions

Reactive expressions $\mathsf{E}_{\Sigma,B_0}$ are closed $\delta$-expressions generated by the grammar

$$\begin{aligned} \delta &::= x \mid \gamma \mid f(\delta,\ldots,\delta) && (x \in X, f \in \Sigma) \\ \gamma &::= \mu x.\ (a_1.\delta \pitchfork \cdots \pitchfork a_n.\delta \pitchfork \beta) && (x \in X, a_i \in A) \\ \beta &::= b \mid f(\beta,\ldots,\beta) && (b \in B_0, f \in \Sigma) \end{aligned}$$

Given a monad $\mathbb{T}$ whose algebraic theory is axiomatized in terms of $\Sigma$ and a $\mathbb{T}$-algebra $B$ generated by $B_0$, we can define $o$ and $\partial_a$ over $\mathsf{E}_{\Sigma,B_0}$.

# Reactive Expressions

Reactive expressions $\mathsf{E}_{\Sigma,B_0}$ are closed $\delta$-expressions generated by the grammar

$$\begin{aligned} \delta &::= x \mid \gamma \mid f(\delta,\dots,\delta) && (x \in X, f \in \Sigma) \\ \gamma &::= \mu x.\ (a_1.\delta \pitchfork \cdots \pitchfork a_n.\delta \pitchfork \beta) && (x \in X, a_i \in A) \\ \beta &::= b \mid f(\beta,\dots,\beta) && (b \in B_0, f \in \Sigma) \end{aligned}$$

Given a monad $\mathbb{T}$ whose algebraic theory is axiomatized in terms of $\Sigma$ and a $\mathbb{T}$-algebra $B$ generated by $B_0$, we can define $o$ and $\partial_a$ over $\mathsf{E}_{\Sigma,B_0}$.

This induces the semantics

$$[\![e]\!](w) = o(\partial_w(e)) \qquad (w \in A^*)$$

# Reactive Expressions

Reactive expressions $\mathsf{E}_{\Sigma,B_0}$ are closed $\delta$-expressions generated by the grammar

$$\delta ::= x \mid \gamma \mid f(\delta,\dots,\delta) \qquad (x \in X, f \in \Sigma)$$

$$\gamma ::= \mu x.\ (a_1.\delta \pitchfork \cdots \pitchfork a_n.\delta \pitchfork \beta) \qquad (x \in X, a_i \in A)$$

$$\beta ::= b \mid f(\beta,\dots,\beta) \qquad (b \in B_0, f \in \Sigma)$$

Given a monad $\mathbb{T}$ whose algebraic theory is axiomatized in terms of $\Sigma$ and a $\mathbb{T}$-algebra $B$ generated by $B_0$, we can define $o$ and $\partial_a$ over $\mathsf{E}_{\Sigma,B_0}$.

This induces the semantics

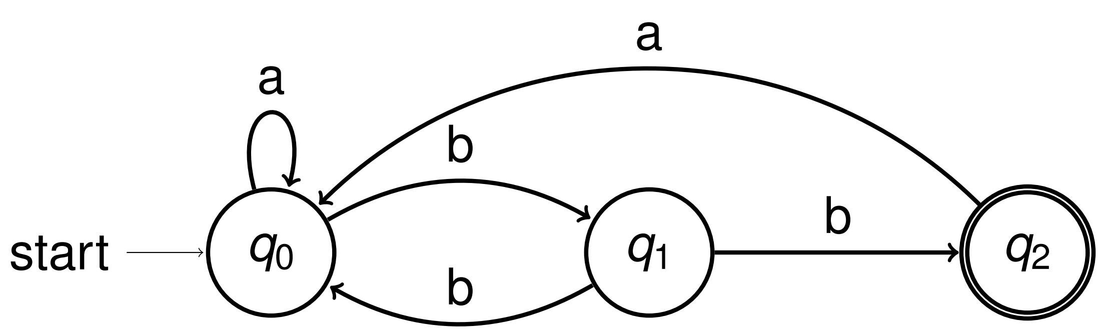
$$[\![e]\!](w) = o(\partial_w(e)) \qquad (w \in A^*)$$

**Kleene Theorem:** for any $e \in \mathsf{E}_{\Sigma,B_0}$ there is a $\mathbb{T}$-automaton $m$ over $X$ and a state $x \in X$ such that $[\![e]\!] = [\![x]\!]_m$ and vice versa.

# Example: Nondeterministic Automata

$B = \{\top, \bot\}$

$\mathbb{T} = \mathcal{P}_\omega$




$$\begin{cases} e_0 = a.e_0 \pitchfork b.e_1 \pitchfork \bot \\ e_1 = a.\varnothing \pitchfork b.(e_0 + e_2) \pitchfork \bot \\ e_2 = a.e_0 \pitchfork b.\varnothing \pitchfork \top \end{cases}$$

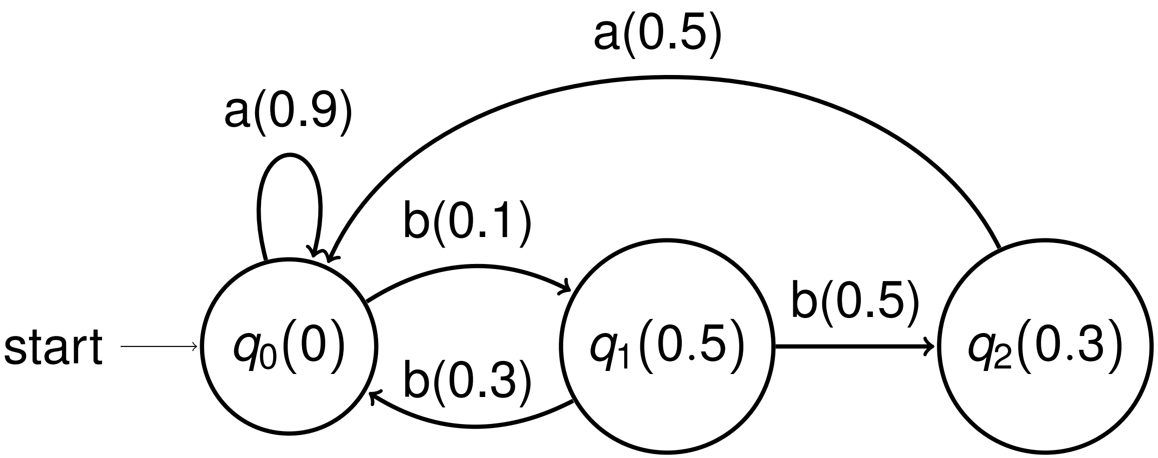
Equivalently,

$e_0 = \mu x.\, (a.x \pitchfork b.\mu y.\, (a.\varnothing \pitchfork b.(x + \mu z.\, (a.x \pitchfork b.\varnothing \pitchfork \top)) \pitchfork \bot) \pitchfork \bot)$.

# Example: Probabilistic Automata

$B = [0, 1]$

$\mathbb{T} = \mathcal{D}_{\omega}^{\leq 1}$



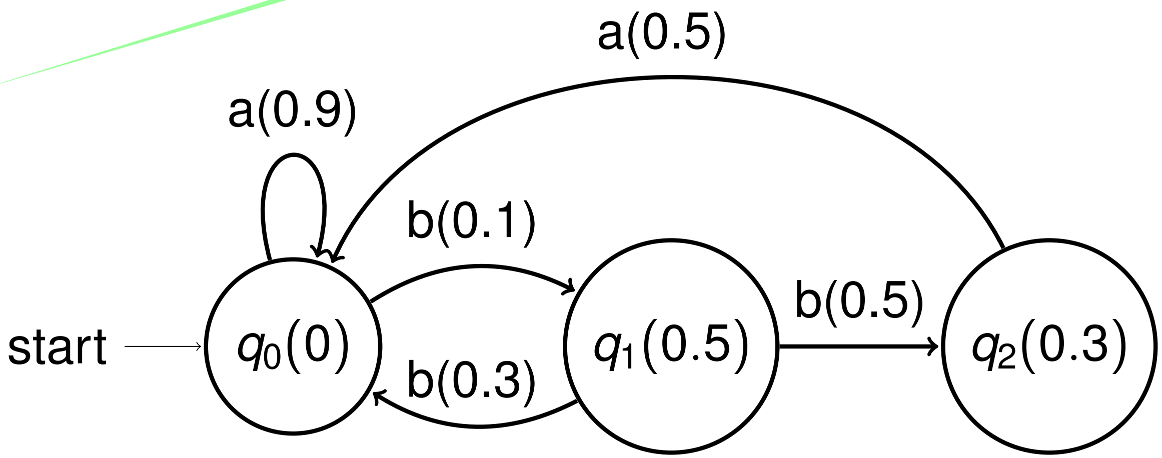

$$\begin{cases} e_0 = a.(e_0 +_{0.9} 0) \pitchfork b.(e_1 +_{0.1} 0) \pitchfork 0 \\ e_1 = a.0 \pitchfork b.(e_0 +_{0.3} e_2 +_{0.5} 0) \pitchfork 0.5 \\ e_2 = a.(e_0 +_{0.5} 0) \pitchfork b.0 \pitchfork 0.3 \end{cases}$$

## Example: Probabilistic Automata

$TX = \{\rho : X \to [0,1] \mid \sum \rho = 1\}$

$B = [0,1]$

$\mathbb{T} = \mathcal{D}_{\omega}^{\leq 1}$




$$\begin{cases} e_0 = a.(e_0 +_{0.9} 0) \pitchfork b.(e_1 +_{0.1} 0) \pitchfork 0 \\ e_1 = a.0 \pitchfork b.(e_0 +_{0.3} e_2 +_{0.5} 0) \pitchfork 0.5 \\ e_2 = a.(e_0 +_{0.5} 0) \pitchfork b.0 \pitchfork 0.3 \end{cases}$$

## Stack $\mathbb{T}$-automata and CFL

(Nondeterministic 1-)stack $\mathbb{T}$-automaton is such $m : X \to \mathcal{B}(\Gamma) \times (TX)^A$ that

- $\mathbb{T}$ is the non-deterministic stack monad;
- $\mathcal{B}(\Gamma)$ are predicates over $\Gamma^*$ such that $p \in \mathcal{B}(\Gamma)$ iff $p(u \cdot w) \iff p(u)$ once $|u| > k$ with some $k$.

# Stack $\mathbb{T}$-automata and CFL

(Nondeterministic 1-)stack $\mathbb{T}$-automaton is such $m : X \to \mathcal{B}(\Gamma) \times (TX)^A$ that

- $\mathbb{T}$ is the non-deterministic stack monad;
- $\mathcal{B}(\Gamma)$ are predicates over $\Gamma^*$ such that $p \in \mathcal{B}(\Gamma)$ iff $p(u \cdot w) \iff p(u)$ once $|u| > k$ with some $k$.

Equivalently, stack $\mathbb{T}$-automata can be specified by reactive expressions, e.g.

$$\mu x.\, (a.\mathbf{push}(x) \pitchfork b.\mathbf{pop}(x, \top) \pitchfork \bot)$$

corresponds to context-free grammar: $X \to aXX$, $X \to b$.

# Stack $\mathbb{T}$-automata and CFL

(Nondeterministic 1-)stack $\mathbb{T}$-automaton is such $m : X \to \mathcal{B}(\Gamma) \times (TX)^A$ that

- $\mathbb{T}$ is the non-deterministic stack monad;
- $\mathcal{B}(\Gamma)$ are predicates over $\Gamma^*$ such that $p \in \mathcal{B}(\Gamma)$ iff $p(u \cdot w) \iff p(u)$ once $|u| > k$ with some $k$.

Equivalently, stack $\mathbb{T}$-automata can be specified by reactive expressions, e.g.

$$\mu x.\, (a.\mathbf{push}(x) \pitchfork b.\mathbf{pop}(x, \top) \pitchfork \bot)$$

corresponds to context-free grammar: $X \to aXX$, $X \to b$.

**Theorem.** If $m$ is an $n$-stack $\mathbb{T}$-automaton, $x \in X$, $s \in \Gamma^*$ then

$$\{w \in A^* \mid [\![x]\!]_m(w)(s)\}$$

ranges over CFL if $n = 1$ and over NTIME if $n > 2$.

# Tape Monad

Tape monad is a submonad of the store monad $\mathbb{Z} \times \Gamma^{\mathbb{Z}} \to (X \times \mathbb{Z} \times \Gamma^{\mathbb{Z}})$ formed by those $r : \mathbb{Z} \times \Gamma^{\mathbb{Z}} \to X$, $z : \mathbb{Z} \times \Gamma^{\mathbb{Z}} \to \mathbb{Z}$, $t : \mathbb{Z} \times \Gamma^{\mathbb{Z}} \to \Gamma^{\mathbb{Z}}$ which satisfy coherence conditions with some $k \geq 0$:

$$t(i,\sigma) =_{i\pm k} t(i,\sigma') \quad z(i,\sigma) = z(i,\sigma') \quad r(i,\sigma) = r(i,\sigma') \qquad (\sigma =_{i\pm k} \sigma')$$

$$t(i,\sigma_{+j}) = t(i+j,\sigma)_{+j} \quad z(i,\sigma_{+j}) = z(i+j,\sigma) - j \quad r(i,\sigma_{+j}) = r(i+j,\sigma)$$

$$t(i,\sigma) =^{i\pm k} \sigma \qquad |z(i,\sigma) - i| \leq k$$

$$\left( \sigma_{+j}(i) = \sigma(i+j) \qquad \begin{array}{l} \sigma =_{i\pm k} \sigma' \text{ iff } \sigma(j) = \sigma'(j) \text{ whenever } |i-j| \leq k \\ \sigma =^{i\pm k} \sigma' \text{ iff } \sigma(j) = \sigma'(j) \text{ whenever } |i-j| > k \end{array} \right)$$

# Tape Monad

head position

tape content

Tape monad is a submonad of the store monad $\mathbb{Z} \times \Gamma^{\mathbb{Z}} \to (X \times \mathbb{Z} \times \Gamma^{\mathbb{Z}})$ formed by those $r : \mathbb{Z} \times \Gamma^{\mathbb{Z}} \to X$, $z : \mathbb{Z} \times \Gamma^{\mathbb{Z}} \to \mathbb{Z}$, $t : \mathbb{Z} \times \Gamma^{\mathbb{Z}} \to \Gamma^{\mathbb{Z}}$ which satisfy coherence conditions with some $k \geq 0$:

$$t(i,\sigma) =_{i\pm k} t(i,\sigma') \quad z(i,\sigma) = z(i,\sigma') \quad r(i,\sigma) = r(i,\sigma') \qquad (\sigma =_{i\pm k} \sigma')$$

$$t(i,\sigma_{+j}) = t(i+j,\sigma)_{+j} \quad z(i,\sigma_{+j}) = z(i+j,\sigma) - j \quad r(i,\sigma_{+j}) = r(i+j,\sigma)$$

$$t(i,\sigma) =^{i\pm k} \sigma \qquad |z(i,\sigma) - i| \leq k$$

$$\left( \sigma_{+j}(i) = \sigma(i+j) \qquad \begin{array}{llll} \sigma =_{i\pm k} \sigma' & \text{iff} & \sigma(j) = \sigma'(j) & \text{whenever } |i-j| \leq k \\ \sigma =^{i\pm k} \sigma' & \text{iff} & \sigma(j) = \sigma'(j) & \text{whenever } |i-j| > k \end{array} \right)$$

# Tape Theory

Tape theory is defined over signature **read** : $n \to 1$, **write**$_i$ : $n \to 1$ $(1 \leq i \leq n)$, **lmove** : $1 \to 1$, **rmove** : $1 \to 1$

Let

$$\begin{aligned} [\![\mathbf{read}]\!](p_1, \ldots, p_n)(z, \sigma) &= p_{\sigma(z)}(z, \sigma) \\ [\![\mathbf{lmove}]\!](p)(z, \sigma) &= p(z - 1, \sigma) \\ [\![\mathbf{write_i}]\!](p)(z, \sigma) &= p(z, \sigma[z \mapsto \gamma_i]) \\ [\![\mathbf{rmove}]\!](p)(z, \sigma) &= p(z + 1, \sigma) \end{aligned}$$

An equation $p = q$ belongs to the tape theory iff $[\![p]\!] = [\![q]\!]$

**Theorem:** Tape theory is not finitely axiomatizable

# Tape $\mathbb{T}$-automata

Tape $\mathbb{T}$-automaton is a $\mathbb{T}$-automaton $m : X \to \mathcal{C}(\Gamma) \times (TX)^A$ where

- $\mathbb{T}$ is the tape monad over $\Gamma$;
- $\mathcal{C}(\Gamma)$ is the set of predicates over $\mathbb{Z} \times \Gamma^{\mathbb{Z}}$ such that $p \in \mathcal{C}(\Gamma)$ iff there is a $k$ such that $p(i, \sigma) = p(i, \sigma')$ and $p(i, \sigma_{+j}) = p(i + j, \sigma)$ if $\sigma =_{i \pm k} \sigma'$.

**Conjecture:** Tape $\mathbb{T}$-automata capture precisely linear-time languages.

**Theorem:** Let $\tau \in A$ and let $\mathcal{L}$ be the class of languages over $A$ captured by $\mathbb{T}$-automata. Then $\{L \backslash \tau \mid \mathcal{L}\}$ are exactly all r.e. languages where $L \backslash \tau$ is the result of removing $\tau$ from $L$.

# A Closer Look at Tensors

The general form of tensor laws:

$$f(g(x_1^1,\dots,x_m^1),\dots,g(x_1^n,\dots,x_m^n)) = g(f(x_1^1,\dots,x_1^n),\dots,f(x_m^1,\dots,x_m^n))$$

**Theorem [Freyd]:** Tensor with $\mathcal{P}_\omega$ yields an idempotent semimodule monad, i.e. $TX = R^X$ where $R$ is an idempotent semiring

In terms of (guarded) reactive expressions:

$$\delta ::= x \mid \varnothing \mid \delta + \delta \mid a.\delta \mid r \cdot \delta \mid \mu x.\,\delta \mid b \qquad (r \in R)$$

**Corolary:** A multistack nondeterministic $\mathbb{T}$-automaton is isomorphic to a weighted $\mathbb{T}$-automaton, i.e. an automaton over some $TX = X^R$

# A Closer Look at Tensors (cntd.)

Tensor of $\mathcal{P}_\omega$ with a finitary submonad of the store monad is isomorphic to a semimodule monad. Intuitively, for

$$\mathcal{P}(X \times \Gamma^*)^{\Gamma^*} \cong \mathcal{P}(\Gamma^* \times \Gamma^*)^X$$

For stacks:

$$\mathbf{pop}(x_1, \ldots, x_n, y) = \mathbf{pop}_1(x_1) + \ldots + \mathbf{pop}_n(x_n) + \mathbf{empty}(y)$$

where

$$\mathbf{pop}_i(x) = \mathbf{pop}(\varnothing, \ldots, x, \ldots, \varnothing, \varnothing)$$
$$\mathbf{empty}(x) = \mathbf{pop}(\varnothing, \ldots, \varnothing, x)$$

Hence, reactive expression are those guarded (!) expressions given by the grammar:

$$\delta ::= x \mid \varnothing \mid \delta + \delta \mid a.\delta \mid r \cdot \delta \mid \mu x.\, \delta \mid 1 \qquad (r \in \{\mathbf{pop}_i, \mathbf{push}_i, \mathbf{empty}\})$$

# CFL revisited

$\mathbb{T}$-automata over $\mathcal{P}_\omega \otimes (- \times M)$ where $M$ is a monoid are called valence automata.

**Example (Polycyclic monoids):** $M$ is the monoid over a set of generators $0, g_1, \ldots, g_k, \ldots g_1^{-1}, \ldots, g_k^{-1}$ satisfying identities

$$0g_i = g_i0 = 0, \qquad g_ig_i^{-1} = 1, \qquad g_ig_j^{-1} = 0 \quad (i \neq j).$$

**Theorem:** If an idempotent semiring $R$ can encode Dyck languages over $\{(_1, )_1, \ldots, (_n, )_n\}^*$ then $\mathbb{T}$-automata recognize CFL.

This applies both to nondeterministic 1-stack automata and to polycyclic valence automata.

# Bonus: Eliminating Internal Actions

Suppose, $\mathbb{T}$ admits infinite summation. Then we can derive from any $\mathcal{m} : X \to B \times TX^{A_\tau}$, $\mathcal{m}_v : X \to B \times TX^{A}$:

$$t^{\mathcal{m}_v}(x_0, a) = \sum_{i=1}^{\infty} \text{do } x_1 \leftarrow t^{\mathcal{m}}(x_0, \tau); \ldots; x_{i-1} \leftarrow t^{\mathcal{m}}(x_{i-2}, \tau); t^{\mathcal{m}}(x_{i-1}, a),$$

$$o^{\mathcal{m}_v}(x_0) = o^{\mathcal{m}}(x_0) + \sum_{i=1}^{\infty} \big(\text{do } x_1 \leftarrow t^{\mathcal{m}}(x_0, \tau); \ldots; t^{\mathcal{m}}(x_{i-1}, \tau)\big)(o^{\mathcal{m}}).$$

**Definition (Observational Semantics):**

$$[\![x]\!]^{v}_{\mathcal{m}} := [\![x]\!]_{\mathcal{m}_v}$$

what if it does not?

# Bonus: Eliminating Internal Actions

Suppose, $\mathbb{T}$ admits infinite summation. Then we can derive from any $\mathfrak{m} : X \to B \times TX^{A_\tau}$, $\mathfrak{m}_v : X \to B \times TX^A$:

$$t^{\mathfrak{m}_v}(x_0, a) = \sum_{i=1}^{\infty} \text{do } x_1 \leftarrow t^{\mathfrak{m}}(x_0, \tau); \ldots ; x_{i-1} \leftarrow t^{\mathfrak{m}}(x_{i-2}, \tau); t^{\mathfrak{m}}(x_{i-1}, a),$$

$$o^{\mathfrak{m}_v}(x_0) = o^{\mathfrak{m}}(x_0) + \sum_{i=1}^{\infty} \big(\text{do } x_1 \leftarrow t^{\mathfrak{m}}(x_0, \tau); \ldots ; t^{\mathfrak{m}}(x_{i-1}, \tau)\big)(o^{\mathfrak{m}}).$$

**Definition (Observational Semantics):**

$$[\![x]\!]^{v}_{\mathfrak{m}} := [\![x]\!]_{\mathfrak{m}_v}$$

# Bonus: Eliminating Internal Actions (cntd.)

It is known (e.g. [Kock, 1970]) that $\mathbb{T}$-algebra structures on $B$ are in one-to-one correspondence with monad morphisms $\mathbb{T} \to \mathbb{T}_B$ where $\mathbb{T}_B$ is the continuation monad

$$T_B X = (X \to B) \to B$$

This induces a transformation of a $\mathbb{T}$-automaton $m$ to a $\mathbb{T}_B$-automaton $m^*$

**Lemma:** $[\![x]\!]_m = [\![x]\!]_{m^*}$

Provided $B$ is a commutative monoid with countable summation, we can introduce observational semantics for $m$:

$$[\![x]\!]^v_m := [\![x]\!]_{m^*_v}$$

# Conclusions: So Many Things to Do Next

- General notions of nondeterminism/memory in terms of theories/monads
- The relation between determinism, nondeterminism and real-timeness
- Compexity of recognized languages as functions of $\mathbb{T}$
- Chomsky-Schützenberger theorem for $\mathbb{T}$-automata
- (Complete) equational calculi of fixpoint expressions
- Identifying further language and complexity classes by $\mathbb{T}$-automata
- $\mathbb{T}$-automata over trees

# Thank You for Your Attention!

FRIEDRICH-ALEXANDER
UNIVERSITÄT
ERLANGEN-NÜRNBERG

TECHNISCHE FAKULTÄT

- Kock, A. (1970).

  On double dualization monads.

  *Math. Scand.*, 27:151–165.

- Myers, R. (2013).

  *Rational Coalgebraic Machines in Varieties: Languages, Completeness and Automatic Proofs*.

  PhD thesis, Imperial College London.

- Rutten, J. J. M. M. (1998).

  Automata and coinduction (an exercise in coalgebra).

  In Sangiorgi, D. and de Simone, R., editors, *CONCUR*, volume 1466 of *Lecture Notes in Computer Science*, pages 194–218. Springer.

- Silva, A. (2010).

  *Kleene coalgebra*.

  PhD thesis, Radboud Univ. Nijmegen.

- Silva, A., Bonchi, F., Bonsangue, M. M., and Rutten, J. J. M. M. (2010).

  Generalizing the powerset construction, coalgebraically.

  In *FSTTCS*, volume 8 of *LIPIcs*, pages 272–283.

- Winter, J., Bonsangue, M. M., and Rutten, J. J. M. M. (2013).

  Coalgebraic characterizations of context-free languages.

  *LMCS*, 9(3).