Initial Algebras of Terms, with binding and algebraic structure

Alexandra Silva and Bart Jacobs

COIN, March 2013

Motivation

$$t := x \mid (t\ t) \mid \lambda x.t \qquad P := 0 \mid P + P \mid a.P \mid x \mid \mu x.P$$

- Terms and formulas

Motivation

$$t := x \mid (t\ t) \mid \lambda x.t \qquad P := 0 \mid P + P \mid a.P \mid x \mid \mu x.P$$

- Terms and formulas
- Unifying perspective on syntax (and semantics).

Motivation

$$t := x \mid (t\ t) \mid \lambda x.t \qquad P := 0 \mid P + P \mid a.P \mid x \mid \mu x.P$$

- Terms and formulas
- Unifying perspective on syntax (and semantics).
- Syntax = Algebra   Semantics = Coalgebra

Motivation

$$t := x \mid (t\ t) \mid \lambda x.t \qquad P := 0 \mid P + P \mid a.P \mid x \mid \mu x.P$$

- Terms and formulas
- Unifying perspective on syntax (and semantics).
- Syntax = Algebra   Semantics = Coalgebra
- Combination = ultimate goal

Motivation

$$t ::= x \mid (t\ t) \mid \lambda x.t \qquad P ::= 0 \mid P + P \mid a.P \mid x \mid \mu x.P$$

- Terms and formulas
- Unifying perspective on syntax (and semantics).
- Syntax = Algebra   Semantics = Coalgebra
- Combination = ultimate goal

This talk: Algebra

Motivation (c'd)

$$t ::= x \mid (t\ t) \mid \lambda x.t \qquad P ::= 0 \mid P + P \mid a.P \mid x \mid \mu x.P$$

- Many calculi have binding operators.

Motivation (c'd)

$$t := x \mid (t\ t) \mid \lambda x.t \qquad P := 0 \mid P + P \mid a.P \mid x \mid \mu x.P$$

- Many calculi have binding operators.
- Challenge: how to keep track of variables in a modular uniform way?

Motivation (c'd)

$$t ::= x \mid (t\ t) \mid \lambda x.t \qquad P ::= 0 \mid P + P \mid a.P \mid x \mid \mu x.P$$

- Many calculi have algebraic operators.
- Challenge: how to derive syntax where this algebraic operators arise for free.

Motivation (c'd)

$$t := x \mid (t\ t) \mid \lambda x.t \qquad P := 0 \mid P{+}P \mid a.P \mid x \mid \mu x.P$$

- Many calculi have algebraic operators.
- Challenge: how to derive syntax where this algebraic operators arise for free.

This talk: How to combine both!

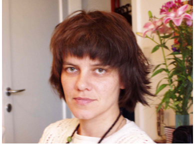
History and inspiration: binding

$t := x \mid (t\ t) \mid \lambda x.t \qquad P := 0 \mid P{+}P \mid a.P \mid x \mid \mu x.P$

- Expressions with variable binding can be described via initiality in presheaf categories.
- Initial algebra semantics: modern perspective on expressions.
- Denotational semantics of expressions is provided by the initial algebra map.
- By construction “compositional”.

History and inspiration: expressions w/ algebraic ops




...

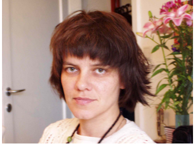
$P := 0 \mid P + P \mid a.P \mid x \mid \mu x.P$ $\quad$ $P := 0 \mid P + P \mid s \bullet P \mid a.\sum_I P_i \mid \cdots$

non-deterministic = **JSL** $\quad$ weighted = **Vect** ($s \in \mathbb{F}$)

- Expressions given and Kleene theorem proved.

History and inspiration: expressions w/ algebraic ops




...

$P ::= 0 \mid P + P \mid a.P \mid x \mid \mu x.P$ $\quad$ $P ::= 0 \mid P + P \mid s \bullet P \mid a.\sum_I P_i \mid \cdots$

non-deterministic = **JSL** $\quad$ weighted = **Vect** ($s \in \mathbb{F}$)

- Expressions given and Kleene theorem proved.
- vs. expressions derived (as initial algebra) and Kleene theorem (almost) for free.

This talk

- We want expressions with a binding operator and algebraic operations.

This talk

- We want expressions with a binding operator and algebraic operations.
- Fiore/Plotkin/Turi used set-valued presheaves $\mathbb{N} \to$ **Sets**.
- We use algebra-valued presheaves $\mathbb{N} \to \mathcal{EM}(T)$.
- We derive expressions as initial algebras.
- Examples with $\mathcal{EM}(\mathcal{P}_{\mathrm{fin}}) =$ **JSL** and $\mathcal{EM}(\mathcal{M}_S) =$ **SMod**.
- Algebraic effects: non-determinism and resource-sensitivity.

The subtlety in a nutshell

$$((x\ y)\ y)[t/x] \quad \mapsto \quad ((t\ y)\ y)$$

The subtlety in a nutshell

$$((x\ y)\ y)[t/x] \quad \mapsto \quad ((t\ y)\ y)$$

$$((x\ y)\ y)[t/y] \quad \mapsto \quad ((x\ t)\ t)$$

The subtlety in a nutshell

$$((x\ y)\ y)[t/x] \quad \mapsto \quad ((t\ y)\ y)$$

$$((x\ y)\ y)[t/y] \quad \mapsto \quad ((x\ t)\ t)$$

- what is the type of substitution?
- $-[-/-]\colon \mathit{Term} \times \mathit{Term} \times \mathit{Var} \to \mathit{Term}$

The subtlety in a nutshell

$$((x\ y)\ y)[t/x] \quad \mapsto \quad ((t\ y)\ y)$$

$$((x\ y)\ y)[t/y] \quad \mapsto \quad ((x\ t)\ t)$$

- what is the type of substitution?
- $-[-/-] \colon \mathit{Term} \times \mathit{Term} \times \mathit{Var} \to \mathit{Term}$
- $-[-/-] \colon \mathit{Term} \times\, !\mathit{Term} \times \mathit{Var} \to \mathit{Term}$
- Need to explicitly model replication !

Variables and terms

Setup:

- Categories of the form $\mathbf{A}^{\mathbb{N}}$ (actually, $\mathcal{EM}(T)^{\mathbb{N}}$)

Variables and terms

Setup:

- Categories of the form $\mathbf{A}^{\mathbb{N}}$ (actually, $\mathcal{EM}(T)^{\mathbb{N}}$)

  Intuition: the exponent in $\mathbf{A}^{\mathbb{N}}$ allows to model the number of free variables.

Variables and terms

Setup:

- Categories of the form $\mathbf{A}^{\mathbb{N}}$ (actually, $\mathcal{EM}(T)^{\mathbb{N}}$)

  Intuition: the exponent in $\mathbf{A}^{\mathbb{N}}$ allows to model the number of free variables.
- We have the free functor $\mathcal{F} \colon \mathbf{Sets} \to \mathcal{EM}(T)$;
- The free algebra adjunction $\mathcal{F} \dashv \mathcal{U}$ induces a comonad $\mathcal{FU} \colon \mathcal{EM}(T) \to \mathcal{EM}(T)$ that we write as !.
- ! is relevant in e.g. linear logic.

Variables and terms

Setup (c'd):

- Weakening monad $\mathcal{W}$: context extension, number of free variables changes

Variables and terms

Setup (c'd):

- Weakening monad $\mathcal{W}$: context extension, number of free variables changes

Lemma

We define a monad $\mathcal{W} = \mathbf{A}^{(-)+1} \colon \mathbf{A}^{\mathbb{N}} \to \mathbf{A}^{\mathbb{N}}$*, so that:*

$$\mathcal{W}(P)(n) \;=\; P(n+1) \qquad \mathcal{W}(P)(f) \;=\; P(f + \mathrm{id}_1)$$

The unit $\mathrm{up} \colon \mathrm{id} \Rightarrow \mathcal{W}$ *and multiplication* $\mathrm{ctt} \colon \mathcal{W}^2 \Rightarrow \mathcal{W}$*:*

$$P(n) \xrightarrow{\mathrm{up}_{P,n} = P(\kappa_1)} P(n+1) \xleftarrow{\mathrm{ctt}_{P,n} = P([\mathrm{id}, \kappa_2])} P(n+2). \qquad \square$$

Variables and terms

Setup (c'd):

- Weakening monad $\mathcal{W}$: context extension, number of free variables changes

Lemma

We define a monad $\mathcal{W} = \mathbf{A}^{(-)+1} : \mathbf{A}^{\mathbb{N}} \to \mathbf{A}^{\mathbb{N}}$*, so that:*

$$\mathcal{W}(P)(n) \;=\; P(n+1) \qquad \mathcal{W}(P)(f) \;=\; P(f + \mathrm{id}_1)$$

The unit $\mathrm{up} : \mathrm{id} \Rightarrow \mathcal{W}$ *and multiplication* $\mathrm{ctt} : \mathcal{W}^2 \Rightarrow \mathcal{W}$*:*

$$P(n) \xrightarrow{\mathrm{up}_{P,n} = P(\kappa_1)} P(n+1) \xleftarrow{\mathrm{ctt}_{P,n} = P([\mathrm{id}, \kappa_2])} P(n+2). \qquad \square$$

up: add a fresh variable; ctt: remove the last variable

Variables and terms

Setup (c'd):

- Weakening monad $\mathcal{W}$: context extension, number of free variables changes

Variables and terms

Setup (c'd):

- Weakening monad $\mathcal{W}$: context extension, number of free variables changes

Intuition/usage: a binding operator b, like λ or μ, has type

$$b \colon \mathcal{W}(P) \to P$$

Technical intermezzo: Substitution for algebra-valued presheaves.

Goal: to define

$$\mathrm{sbs}\colon \mathcal{W}T(\mathcal{F}) \otimes !\,T(\mathcal{F}) \to T(\mathcal{F})$$

Technical intermezzo: Substitution for algebra-valued presheaves.

Goal: to define

$$\mathrm{sbs}\colon \mathcal{W}T(\mathcal{F}) \otimes {!}\,T(\mathcal{F}) \to T(\mathcal{F})$$

How? By induction with parameters (next slide).

Technical intermezzo: Substitution for algebra-valued presheaves.

Goal: to define

$$\mathrm{sbs}\colon \mathcal{W}T(\mathcal{F}) \otimes {!T(\mathcal{F})} \to T(\mathcal{F})$$

How? By induction with parameters (next slide).

One may read $\mathrm{sbs}_n(s \otimes U) = s[U/v_{n+1}]$ where U is of !-type.
The type $\mathcal{W}T(\mathcal{F}) \otimes {!T(\mathcal{F})} \to T(\mathcal{F})$ is rich:

- First argument $\mathcal{W}T(\mathcal{F})$: term s in an augmented context (variable v_{n+1} to be substituted)
- The second argument U of replication type $!T(\mathcal{F})$ is going to be substituted for the variable v_{n+1}.
- Number of times U needs to be used in substitution taken into account (main diff. with Fiore/Plotkin/Turi).

Induction with parameters

$H\colon \mathcal{EM}(T) \to \mathcal{EM}(T)$ endofunctor on $\mathcal{EM}(T)$ of a commutative monad T on **Sets**. If H has an initial algebra $a\colon H(A) \xrightarrow{\cong} A$ then:

$$\begin{array}{ccccccc}
H(A) \otimes !B & \xrightarrow{\mathrm{id}\otimes\Delta} & H(A) \otimes !B \otimes !B & \xrightarrow{\mathrm{st}\otimes\mathrm{id}} & H(A \otimes !B) \otimes !B & \xrightarrow{H(h)\otimes\mathrm{id}} & H(C) \otimes !B \\
{\scriptstyle a\otimes\mathrm{id}}\downarrow{\scriptstyle\cong} & & & & & & \downarrow{\scriptstyle c} \\
A \otimes !B & & & \xrightarrow[h]{\qquad\qquad} & & & C
\end{array}$$

Induction with parameters

$H\colon \mathcal{EM}(T) \to \mathcal{EM}(T)$ endofunctor on $\mathcal{EM}(T)$ of a commutative monad T on **Sets**. If H has an initial algebra $a\colon H(A) \xrightarrow{\cong} A$ then:

$$\begin{array}{ccccccc}
H(A) \otimes !B & \xrightarrow{\mathrm{id}\otimes\Delta} & H(A) \otimes !B \otimes !B & \xrightarrow{\mathrm{st}\otimes\mathrm{id}} & H(A \otimes !B) \otimes !B & \xrightarrow{H(h)\otimes\mathrm{id}} & H(C) \otimes !B \\
{\scriptstyle a\otimes\mathrm{id}}\big\downarrow{\scriptstyle\cong} & & & & & & \big\downarrow{\scriptstyle c} \\
A \otimes !B & & & \xrightarrow[h]{\qquad\qquad\qquad} & & & C
\end{array}$$

Our goal was: $\mathrm{sbs}\colon \mathcal{W}T(\mathcal{F}) \otimes !T(\mathcal{F}) \to T(\mathcal{F})$.
Missing: the st map and $\mathcal{W}T(\mathcal{F})$ as initial algebra.

The st map

Proposition

Let T be a commutative monad on **Sets***, and $H\colon \mathcal{EM}(T) \to \mathcal{EM}(T)$ be an arbitrary functor. For algebras A, B there is a "non-linear" strength map:*

$$H(A) \otimes !B \xrightarrow{\quad \text{st} \quad} H(A \otimes !B). \tag{1}$$

$\mathcal{W}T(\mathcal{F})$ as initial algebra

- $\mathcal{W}T(\mathcal{F})$ is the free H-algebra on $\mathcal{W}(\mathcal{F})$ (technical lemma);
- Hence, if we have an isomorphism $\phi\colon H\mathcal{W} \overset{\cong}{\Rightarrow} \mathcal{W}H$, it is an initial algebra of the functor $\mathcal{W}(\mathcal{F}) + H(-)\colon \mathcal{EM}(T)^{\mathbb{N}} \to \mathcal{EM}(T)^{\mathbb{N}}$, via:

$$\mathcal{W}(\mathcal{F}) + H\big(\mathcal{W}T(\mathcal{F})\big) \xrightarrow[\cong]{\mathrm{id}+\phi} \mathcal{W}(\mathcal{F}) + \mathcal{W}H\big(T(\mathcal{F})\big) = \mathcal{W}\Big(\mathcal{F} + H\big(T(\mathcal{F})\big)\Big)$$

$$\cong \Big\downarrow \mathcal{W}([\eta_{\mathcal{F}}, \theta_{\mathcal{F}}])$$

$$\mathcal{W}T(\mathcal{F}).$$

- This is enough to define $\mathrm{sbs}\colon \mathcal{W}T(\mathcal{F}) \otimes !\,T(\mathcal{F}) \to T(\mathcal{F})$ by induction with parameters.

Non-deterministic lambda calculus

$\Lambda \in \mathbf{JSL}^{\mathbb{N}}$ initial algebra of

$$P \quad \longmapsto \quad \mathcal{F} + \mathcal{W}(P) + (P \otimes !P),$$

$\mathcal{F}(n) = \mathcal{P}_{\mathrm{fin}}(n)$ and $!P(n) = \mathcal{P}_{\mathrm{fin}}(P(n))$.

Non-deterministic lambda calculus

$\Lambda \in \mathbf{JSL}^{\mathbb{N}}$ initial algebra of

$$P \quad \longmapsto \quad \mathcal{F} + \mathcal{W}(P) + (P \otimes !P),$$

$\mathcal{F}(n) = \mathcal{P}_{\text{fin}}(n)$ and $!P(n) = \mathcal{P}_{\text{fin}}(P(n))$.

Described by:

$$\mathcal{F} + \mathcal{W}(\Lambda) + (\Lambda \otimes !\Lambda) \xrightarrow[\cong]{[\text{var},\text{lam},\text{app}]} \Lambda$$

Non-deterministic lambda calculus

Elements of the set of terms $\Lambda(n) \in \mathbf{JSL}$ with variables from $\{v_1, \ldots, v_n\}$ are inductively given by:

- $\mathrm{var}_n(V)$, where $V \subseteq n = \{v_1, v_2, \ldots, v_n\}$;
- $\mathrm{lam}_n(N) = \lambda v_{n+1}.\, N$, where $N \in \Lambda(n+1)$;
- $\mathrm{app}(M, \{N_1, \ldots, N_k\}) = M \cdot \{N_1, \ldots, N_k\}$, where $M, N_1, \ldots, N_k \in \Lambda(n)$;
- $\bot \in \Lambda(n)$, and $M \vee N \in \Lambda(n)$, for $M, N \in \Lambda(n)$.

Non-deterministic lambda calculus

Elements of the set of terms $\Lambda(n) \in \mathbf{JSL}$ with variables from $\{v_1, \ldots, v_n\}$ are inductively given by:

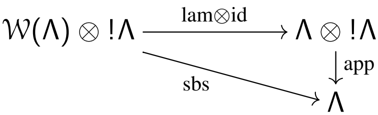
- $\mathrm{var}_n(V)$, where $V \subseteq n = \{v_1, v_2, \ldots, v_n\}$;
- $\mathrm{lam}_n(N) = \lambda v_{n+1}.\, N$, where $N \in \Lambda(n+1)$;
- $\mathrm{app}(M, \{N_1, \ldots, N_k\}) = M \cdot \{N_1, \ldots, N_k\}$, where $M, N_1, \ldots, N_k \in \Lambda(n)$;
- $\bot \in \Lambda(n)$, and $M \vee N \in \Lambda(n)$, for $M, N \in \Lambda(n)$.

Non-standard features

Sets of variables in var_n and second arg of application. Using linearity of the operations these can given in terms of single variables.

Equations

(β)-rule: $(\lambda x.\, M)N = M[N/x]$



Weighted lambda calculus

Initial algebra of the *same* functor.

$\Lambda_w \in \mathbf{SMod}^{\mathbb{N}}$ initial algebra of

$$P \quad \longmapsto \quad \mathcal{F} + \mathcal{W}(P) + (P \otimes !P),$$

with $\mathcal{F}(n) = \mathcal{M}(n)$ and $!P(n) = \mathcal{M}(P(n))$.

Weighted lambda calculus

Initial algebra of the *same* functor.

$\Lambda_w \in \mathbf{SMod}^{\mathbb{N}}$ initial algebra of

$$P \quad \longmapsto \quad \mathcal{F} + \mathcal{W}(P) + (P \otimes !P),$$

with $\mathcal{F}(n) = \mathcal{M}(n)$ and $!P(n) = \mathcal{M}(P(n))$.

Non-standard features

Variables and second argument of application are linear combinations of terms.

Non-deterministic automata

The presheaf of expressions $\mathsf{E} \in \mathbf{JSL}^{\mathbb{N}}$ is the initial algebra of the functor on $\mathbf{JSL}^{\mathbb{N}}$ given by:

$$P \quad \longmapsto \quad \mathcal{F} + \mathcal{W}(P) + 2 + A \cdot {!P},$$

where $2 = \{\bot, \top\}$, $\mathcal{F}(n) = \mathcal{P}_{\text{fin}}(\{v_1, \ldots, v_n\})$, and $!P = \mathcal{P}_{\text{fin}}(P)$.

Non-deterministic automata

The presheaf of expressions $\mathsf{E} \in \mathbf{JSL}^{\mathbb{N}}$ is the initial algebra of the functor on $\mathbf{JSL}^{\mathbb{N}}$ given by:

$$P \longmapsto \mathcal{F} + \mathcal{W}(P) + 2 + A \cdot !P,$$

where $2 = \{\bot, \top\}$, $\mathcal{F}(n) = \mathcal{P}_{\text{fin}}(\{v_1, \ldots, v_n\})$, and $!P = \mathcal{P}_{\text{fin}}(P)$.

We can describe E as

$$\mathcal{F} + \mathcal{W}(\mathsf{E}) + 2 + A \cdot !\mathsf{E} \xrightarrow[\cong]{[\text{var},\text{fix},\text{ops},\text{pre}]} \mathsf{E}$$

Non-deterministic automata

The presheaf of expressions $\mathsf{E} \in \mathbf{JSL}^{\mathbb{N}}$ is the initial algebra of the functor on $\mathbf{JSL}^{\mathbb{N}}$ given by:

$$P \quad \longmapsto \quad \mathcal{F} + \mathcal{W}(P) + 2 + A \cdot !P,$$

where $2 = \{\bot, \top\}$, $\mathcal{F}(n) = \mathcal{P}_{\text{fin}}(\{v_1, \ldots, v_n\})$, and $!P = \mathcal{P}_{\text{fin}}(P)$.

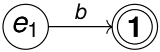
We can describe E as

$$\mathcal{F} + \mathcal{W}(\mathsf{E}) + 2 + A \cdot !\mathsf{E} \xrightarrow[\cong]{[\text{var},\text{fix},\text{ops},\text{pre}]} \mathsf{E}$$

- $\mu v_{n+1}.e = \text{fix}_n(e)$;
- $\mathbf{0} = \text{ops}(\bot) \qquad \mathbf{1} = \text{ops}(\top)$;
- $a(\{e_1, \ldots e_k\}) = \text{pre}(\kappa_a(\{e_1, \ldots e_k\}))$
- $\bot$ and $e \vee e'$ for any $e, e' \in \mathsf{E}(n)$.

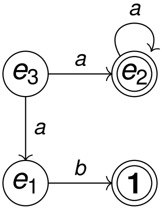
Examples

$e_1 = b(\mathbf{1})$

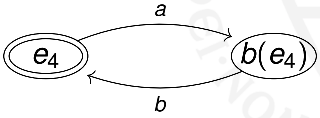


$e_2 = \mu x.a(\{x\}) \vee \mathbf{1}$

$e_3 = a(\{e_1, e_2\})$

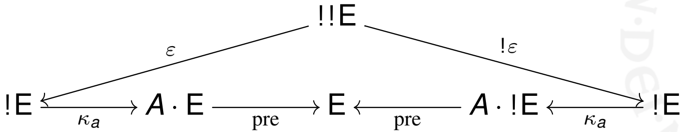


$e_4 = \mu x.\mathbf{1} \vee a(b(x))$



Equations

$$a(\{e \vee e'\}) = a(\{e\}) \vee a(\{e'\}).$$



(2)

Weighted automata

Initial algebra E of the functor on **SMod**$^{\mathbb{N}}$:

$$P \longmapsto \mathcal{F} + \mathcal{W}(P) + S + A \cdot !P,$$

where $\mathcal{F}(n) = \mathcal{M}(\{v_1, \ldots, v_n\})$ and $!P = \mathcal{M}(P)$.

Weighted automata

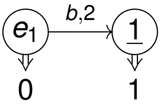
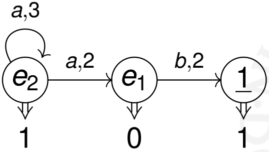
Initial algebra E of the functor on **SMod**$^{\mathbb{N}}$:

$$P \quad \longmapsto \quad \mathcal{F} + \mathcal{W}(P) + S + A \cdot {!P},$$

where $\mathcal{F}(n) = \mathcal{M}(\{v_1, \ldots, v_n\})$ and $!P = \mathcal{M}(P)$.

- $\underline{s} = \mathrm{val}(s)$, for any $s \in S$;
- $a(\sum_i s_i e_i) = \mathrm{pre}(\kappa_a(\sum_i s_i e_i))$;
- 0, $s \bullet e$, and $e + e'$ for any $e, e' \in \mathrm{E}(n)$ and $s \in S$.

Examples

$e_1 = b(2 \cdot \underline{1})$	$e_2 = \mu x.a(3x + 2e_1) + \underline{1}$
	

Weighted automata: equations

How do we axiomatize trace semantics for weighted automata?

Weighted automata: equations

How do we axiomatize trace semantics for weighted automata?

We take the same diagram as before

$$
\begin{array}{ccccccccc}
 & & & & !!\mathsf{E} & & & & \\
 & \swarrow^{\varepsilon} & & & & & & {}^{!\varepsilon}\searrow & \\
!\mathsf{E} & \xrightarrow[\kappa_a]{} & A \cdot \mathsf{E} & \xrightarrow[\text{pre}]{} & \mathsf{E} & \xleftarrow[\text{pre}]{} & A \cdot !\mathsf{E} & \xleftarrow[\kappa_a]{} & !\mathsf{E}
\end{array}
\qquad (3)
$$

Weighted automata: equations

How do we axiomatize trace semantics for weighted automata?

We take the same diagram as before

$$\begin{array}{ccccccccc} & & & & !!\mathrm{E} & & & & \\ & \varepsilon \swarrow & & & & & & \searrow !\varepsilon & \\ !\mathrm{E} & \xrightarrow{\kappa_a} & A \cdot \mathrm{E} & \xrightarrow{\text{pre}} & \mathrm{E} & \xleftarrow{\text{pre}} & A \cdot !\mathrm{E} & \xleftarrow{\kappa_a} & !\mathrm{E} \end{array} \qquad (3)$$

and write down the equation:

$$a(s(e_1 + e_2)) \quad = \quad a(se_1) + a(se_2).$$

Weighted automata: equations

How do we axiomatize trace semantics for weighted automata?

We take the same diagram as before

$$\begin{array}{ccccccccc} & & & & !!\mathrm{E} & & & & \\ & \swarrow^{\varepsilon} & & & & & & \searrow^{!\varepsilon} & \\ !\mathrm{E} & \xrightarrow[\kappa_a]{} & A \cdot \mathrm{E} & \xrightarrow[\text{pre}]{} & \mathrm{E} & \xleftarrow[\text{pre}]{} & A \cdot !\mathrm{E} & \xleftarrow[\kappa_a]{} & !\mathrm{E} \end{array} \qquad (3)$$

and write down the equation:

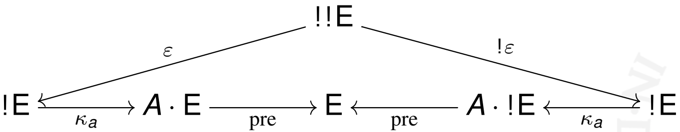
$$a(s(e_1 + e_2)) \quad = \quad a(se_1) + a(se_2).$$

same equation as in [Bonsangue, Milius, Silva 2012].

Weighted automata: equations

How do we axiomatize trace semantics for weighted automata?

We take the same diagram as before



(3)

and write down the equation:

$$a(s(e_1 + e_2)) \quad = \quad a(se_1) + a(se_2).$$

same equation as in [Bonsangue, Milius, Silva 2012]. The difference wrt NDA is in the interpretation of !.

Conclusions and Future Work

Conclusions

- Exploited initiality in algebra-valued pre-sheaves
- Systematic description of several examples: lambda-calculi and automata.

Conclusions and Future Work

Conclusions

- Exploited initiality in algebra-valued pre-sheaves
- Systematic description of several examples: lambda-calculi and automata.

Future Work

- Develop the coalgebraic/semantic side of the framework.
- Explore formalization of equations: systematic account of axiomatizations for different equivalences.

Conclusions and Future Work

Conclusions

- Exploited initiality in algebra-valued pre-sheaves
- Systematic description of several examples: lambda-calculi and automata.

Future Work

- Develop the coalgebraic/semantic side of the framework.
- Explore formalization of equations: systematic account of axiomatizations for different equivalences.

Thanks!