

(Abstract) GSOS for Trace Equivalence – Early Ideas

CALCO 2025 – Glasgow

Robin Jourde^{*}, Pouya Partow[†], Jonas Forster[‡], in collaboration with Stelios Tsampas[§], Sergey Goncharov[†], Henning Urbat[‡]

16 June 2025

^{*}ENS de Lyon, Université Savoie Mont Blanc

[†]University of Birmingham

[‡]**Friedrich-Alexander-Universität Erlangen-Nürnberg**

[§]Syddansk Universitet

1 (Abstract) GSOS

1.1 GSOS

- a **framework** for specifying reduction rules and semantics of systems \rightarrow rule format

1.1 GSOS

- a **framework** for specifying reduction rules and semantics of systems \rightarrow rule format
- **syntax**: operations with arities

1.1 GSOS

- a **framework** for specifying reduction rules and semantics of systems \rightarrow rule format
- **syntax**: operations with arities
- **behaviour** (LTS): $x \xrightarrow{a} x'$ ($a \in L$ for some fixed set L)

1.1 GSOS

- a **framework** for specifying reduction rules and semantics of systems \rightarrow rule format
- **syntax**: operations with arities
- **behaviour** (LTS): $x \xrightarrow{a} x'$ ($a \in L$ for some fixed set L)
- **GSOS rules**

$$\frac{x_1 \xrightarrow{a_1} y_1 \quad \dots \quad x_1 \xrightarrow{a_1} y_k \quad x_1 \xrightarrow{a_2} y_{k+1} \quad \dots \quad x_2 \xrightarrow{a_1} y_l \quad \dots \quad x_1 \xrightarrow{b_1} \dots}{\sigma(x_1 \dots x_n) \xrightarrow{c} u}$$

- ▶ x_i, y_j distinct **variables**
- ▶ u term with variables x_i, y_j

1.1 GSOS

- a **framework** for specifying reduction rules and semantics of systems \rightarrow rule format
- **syntax**: operations with arities
- **behaviour** (LTS): $x \xrightarrow{a} x'$ ($a \in L$ for some fixed set L)
- **GSOS rules**

$$\frac{x_1 \xrightarrow{a_1} y_1 \quad \dots \quad x_1 \xrightarrow{a_1} y_k \quad x_1 \xrightarrow{a_2} y_{k+1} \quad \dots \quad x_2 \xrightarrow{a_1} y_l \quad \dots \quad x_1 \xrightarrow{b_1} \dots}{\sigma(x_1 \dots x_n) \xrightarrow{c} u}$$

- ▶ x_i, y_j distinct **variables**
- ▶ u term with variables x_i, y_j
- set of GSOS rules \Rightarrow behaviour of terms

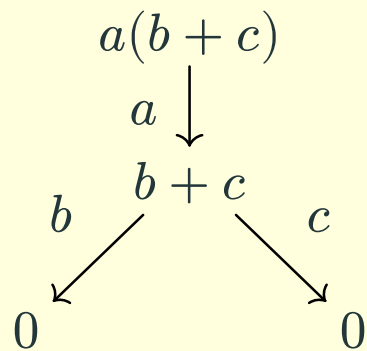
Example.

$$t ::= 0 \mid a.t \quad \forall a \in L \mid t + t$$
$$\frac{}{a.t \xrightarrow{a} t} \forall a \qquad \frac{t \xrightarrow{a} t'}{t + u \xrightarrow{a} t'} \forall a \qquad \frac{u \xrightarrow{a} u'}{t + u \xrightarrow{a} u'} \forall a$$

Example.

$$t ::= 0 \mid a.t \quad \forall a \in L \mid t + t$$

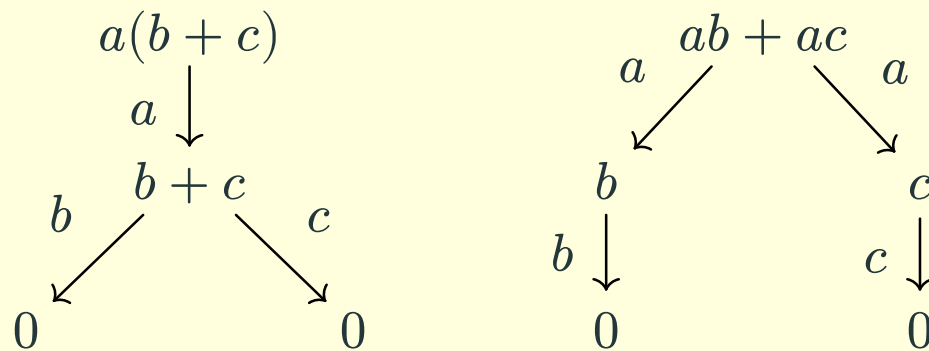
$$\frac{}{a.t \xrightarrow{a} t} \forall a \qquad \frac{t \xrightarrow{a} t'}{t + u \xrightarrow{a} t'} \forall a \qquad \frac{u \xrightarrow{a} u'}{t + u \xrightarrow{a} u'} \forall a$$



Example.

$$t ::= 0 \mid a.t \quad \forall a \in L \mid t + t$$

$$\frac{}{a.t \xrightarrow{a} t} \forall a \qquad \frac{t \xrightarrow{a} t'}{t + u \xrightarrow{a} t'} \forall a \qquad \frac{u \xrightarrow{a} u'}{t + u \xrightarrow{a} u'} \forall a$$



1.2 Abstract GSOS

- syntax functor Σ and behaviour functor H (on a category \mathbb{C})

1.2 Abstract GSOS

- syntax functor Σ and behaviour functor H (on a category \mathbb{C})

$$t ::= 0 \mid a.t \quad \forall a \in L \mid t + t \quad \rightsquigarrow \quad \Sigma X = 1 + A \times X + X^2$$

$$x \xrightarrow{a} x' \quad \rightsquigarrow \quad x' \in k(x)(a) \text{ with } k : X \rightarrow HX = \mathcal{P}(X)^L$$

- rules \rightsquigarrow a **natural transformation** $\rho_X : \Sigma(X \times HX) \rightarrow H\Sigma^*X$

1.2 Abstract GSOS

- syntax functor Σ and behaviour functor H (on a category \mathbb{C})

$$t ::= 0 \mid a.t \quad \forall a \in L \mid t + t \quad \rightsquigarrow \quad \Sigma X = 1 + A \times X + X^2$$

$$x \xrightarrow{a} x' \quad \rightsquigarrow \quad x' \in k(x)(a) \text{ with } k : X \rightarrow HX = \mathcal{P}(X)^L$$

- rules \rightsquigarrow a **natural transformation** $\rho_X : \Sigma(X \times HX) \rightarrow H\Sigma^*X$

Example.

$$\rho : 1 + A \times (X \times \mathcal{P}(X)^L) + (X \times \mathcal{P}(X)^L)^2 \rightarrow \mathcal{P}(\Sigma^*X)^L$$

1.2 Abstract GSOS

- syntax functor Σ and behaviour functor H (on a category \mathbb{C})

$$t ::= 0 \mid a.t \quad \forall a \in L \mid t + t \quad \rightsquigarrow \quad \Sigma X = 1 + A \times X + X^2$$

$$x \xrightarrow{a} x' \quad \rightsquigarrow \quad x' \in k(x)(a) \text{ with } k : X \rightarrow HX = \mathcal{P}(X)^L$$

- rules \rightsquigarrow a **natural transformation** $\rho_X : \Sigma(X \times HX) \rightarrow H\Sigma^*X$

Example.

$$\rho : 1 + A \times (X \times \mathcal{P}(X)^L) + (X \times \mathcal{P}(X)^L)^2 \rightarrow \mathcal{P}(\Sigma^*X)^L$$

- $\rho(*) (a) = \emptyset$

1.2 Abstract GSOS

- syntax functor Σ and behaviour functor H (on a category \mathbb{C})
- rules \rightsquigarrow a **natural transformation** $\rho_X : \Sigma(X \times HX) \rightarrow H\Sigma^*X$

Example.

$$\rho : 1 + A \times (X \times \mathcal{P}(X)^L) + (X \times \mathcal{P}(X)^L)^2 \rightarrow \mathcal{P}(\Sigma^*X)^L$$

- $\rho(*) (a) = \emptyset$
- $\rho((a', t, T))(a) = \{t\}$ if $a = a'$ and \emptyset otherwise

$$\frac{}{a.t \xrightarrow{a} t} \forall a$$

1.2 Abstract GSOS

- syntax functor Σ and behaviour functor H (on a category \mathbb{C})
- rules \rightsquigarrow a **natural transformation** $\rho_X : \Sigma(X \times HX) \rightarrow H\Sigma^*X$

Example.

$$\rho : 1 + A \times (X \times \mathcal{P}(X)^L) + (X \times \mathcal{P}(X)^L)^2 \rightarrow \mathcal{P}(\Sigma^*X)^L$$

- $\rho(*) (a) = \emptyset$
- $\rho((a', t, T))(a) = \{t\}$ if $a = a'$ and \emptyset otherwise
- $\rho((t, T), (u, U))(a) = \{t' \mid t' \in T(a)\} \cup \{u' \mid u' \in U(a)\} = T \cup U$

$$\frac{t \xrightarrow{a} t'}{t + u \xrightarrow{a} t'} \forall a$$

$$\frac{u \xrightarrow{a} u'}{t + u \xrightarrow{a} u'} \forall a$$

2 Trace equivalence

2.1 Program equivalence

- many different notions (linear-time branching-time spectrum), eg. bisimilarity, trace equivalence

2.1 Program equivalence

- many different notions (linear-time branching-time spectrum), eg. bisimilarity, trace equivalence
- important property: **congruence**

$$\forall \sigma \in \mathcal{O}, (\forall i, x_i \sim y_i) \Rightarrow \sigma(x_1..x_n) \sim \sigma(y_1..y_n)$$

2.1 Program equivalence

- many different notions (linear-time branching-time spectrum), eg. bisimilarity, trace equivalence
- important property: **congruence**

$$\forall \sigma \in \mathcal{O}, (\forall i, x_i \sim y_i) \Rightarrow \sigma(x_1..x_n) \sim \sigma(y_1..y_n)$$

Theorem. (D. Turi and G. Plotkin, 1997)

abstract GSOS \Rightarrow bisimilarity is a congruence

2.1 Program equivalence

- many different notions (linear-time branching-time spectrum), eg. bisimilarity, trace equivalence
- important property: **congruence**

$$\forall \sigma \in \mathcal{O}, (\forall i, x_i \sim y_i) \Rightarrow \sigma(x_1..x_n) \sim \sigma(y_1..y_n)$$

Theorem. (D. Turi and G. Plotkin, 1997)

abstract GSOS \Rightarrow bisimilarity is a congruence

- what about trace equivalence?

2.2 Trace and trace equivalence

- partial finite traces

$$\text{tr}(x) = \bigcup_{n \in \mathbb{N}} \left\{ w \in L^n \mid x \xrightarrow{w_1} x_1 \xrightarrow{w_2} \dots \xrightarrow{w_n} x_n \right\}$$

2.2 Trace and trace equivalence

- partial finite traces

$$\text{tr}(x) = \bigcup_{n \in \mathbb{N}} \left\{ w \in L^n \mid x \xrightarrow{w_1} x_1 \xrightarrow{w_2} \dots \xrightarrow{w_n} x_n \right\}$$

- $\text{tr}(x) \in \mathfrak{T} := \{ A \subseteq L^* \mid \varepsilon \in A \wedge A \text{ prefix-closed} \}$

2.2 Trace and trace equivalence

- partial finite traces

$$\text{tr}(x) = \bigcup_{n \in \mathbb{N}} \left\{ w \in L^n \mid x \xrightarrow{w_1} x_1 \xrightarrow{w_2} \dots \xrightarrow{w_n} x_n \right\}$$

- $\text{tr}(x) \in \mathfrak{T} := \{ A \subseteq L^* \mid \varepsilon \in A \wedge A \text{ prefix-closed} \}$

Remark. tr is the unique map $X \rightarrow \mathfrak{T}$ such that $a.w \in \text{tr}(x) \Leftrightarrow x \xrightarrow{a} y \wedge w \in \text{tr}(y)$
 (“coalgebra morphism”)

2.2 Trace and trace equivalence

- partial finite traces

$$\text{tr}(x) = \bigcup_{n \in \mathbb{N}} \left\{ w \in L^n \mid x \xrightarrow{w_1} x_1 \xrightarrow{w_2} \dots \xrightarrow{w_n} x_n \right\}$$

- $\text{tr}(x) \in \mathfrak{T} := \{ A \subseteq L^* \mid \varepsilon \in A \wedge A \text{ prefix-closed} \}$

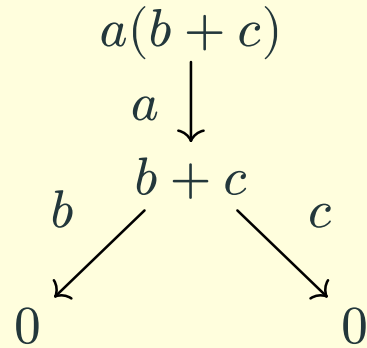
Remark. tr is the unique map $X \rightarrow \mathfrak{T}$ such that $a.w \in \text{tr}(x) \Leftrightarrow x \xrightarrow{a} y \wedge w \in \text{tr}(y)$
 (“coalgebra morphism”)

- trace equivalence: $x \sim_{\text{tr}} y \Leftrightarrow \text{tr}(x) = \text{tr}(y)$

2.2 Trace and trace equivalence

$$\text{tr}(x) = \bigcup_{n \in \mathbb{N}} \left\{ w \in L^n \mid x \xrightarrow{w_1} x_1 \xrightarrow{w_2} \dots \xrightarrow{w_n} x_n \right\}$$

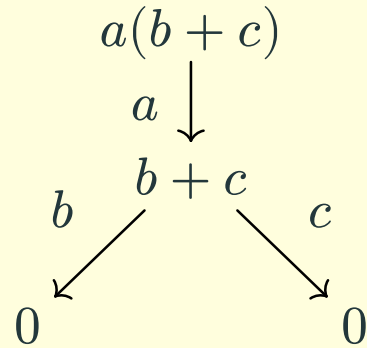
Example. $\text{tr } a(b + c) =$



2.2 Trace and trace equivalence

$$\text{tr}(x) = \bigcup_{n \in \mathbb{N}} \left\{ w \in L^n \mid x \xrightarrow{w_1} x_1 \xrightarrow{w_2} \dots \xrightarrow{w_n} x_n \right\}$$

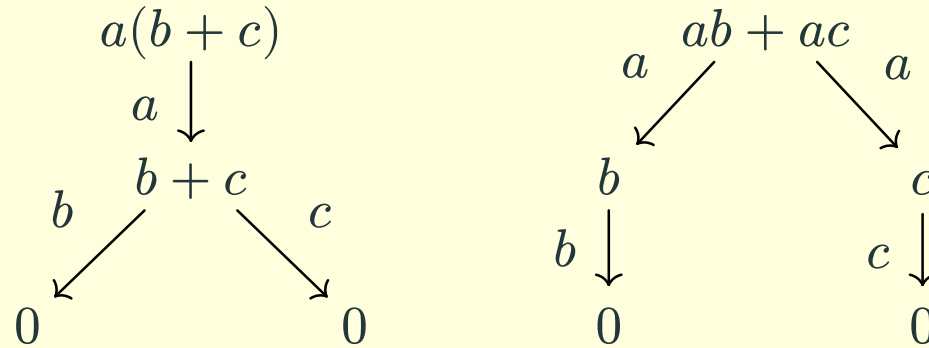
Example. $\text{tr } a(b + c) = \{\varepsilon, a, ab, ac\},$



2.2 Trace and trace equivalence

$$\text{tr}(x) = \bigcup_{n \in \mathbb{N}} \left\{ w \in L^n \mid x \xrightarrow{w_1} x_1 \xrightarrow{w_2} \dots \xrightarrow{w_n} x_n \right\}$$

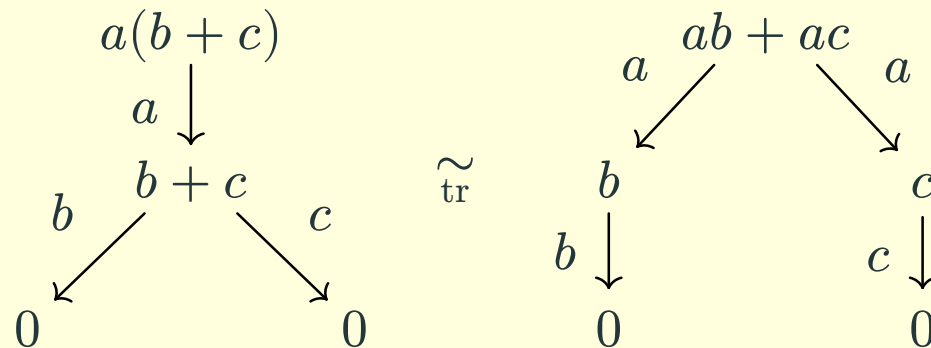
Example. $\text{tr } a(b + c) = \{\varepsilon, a, ab, ac\}$, $\text{tr } (ab + ac) =$



2.2 Trace and trace equivalence

$$\text{tr}(x) = \bigcup_{n \in \mathbb{N}} \left\{ w \in L^n \mid x \xrightarrow{w_1} x_1 \xrightarrow{w_2} \dots \xrightarrow{w_n} x_n \right\}$$

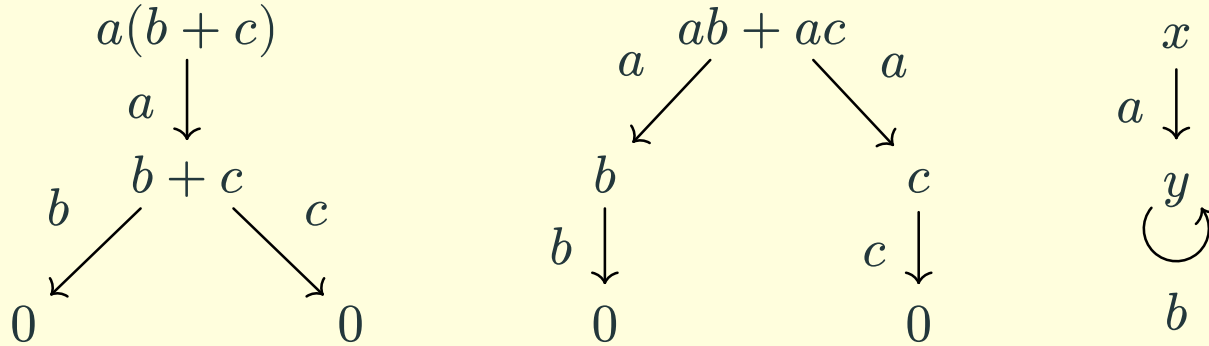
Example. $\text{tr } a(b + c) = \{\varepsilon, a, ab, ac\}$, $\text{tr } (ab + ac) = \{\varepsilon, a, ab, ac\}$,



2.2 Trace and trace equivalence

$$\text{tr}(x) = \bigcup_{n \in \mathbb{N}} \left\{ w \in L^n \mid x \xrightarrow{w_1} x_1 \xrightarrow{w_2} \dots \xrightarrow{w_n} x_n \right\}$$

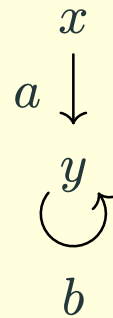
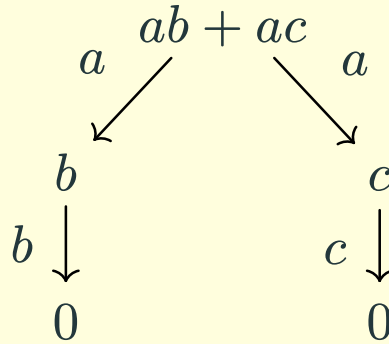
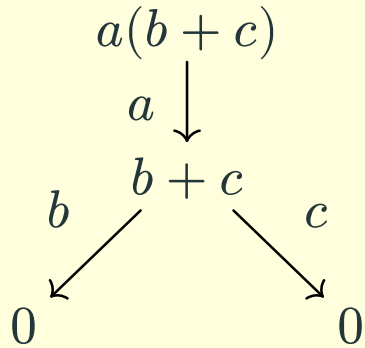
Example. $\text{tr } a(b + c) = \{\varepsilon, a, ab, ac\}$, $\text{tr } (ab + ac) = \{\varepsilon, a, ab, ac\}$, $\text{tr } x =$



2.2 Trace and trace equivalence

$$\text{tr}(x) = \bigcup_{n \in \mathbb{N}} \left\{ w \in L^n \mid x \xrightarrow{w_1} x_1 \xrightarrow{w_2} \dots \xrightarrow{w_n} x_n \right\}$$

Example. $\text{tr } a(b + c) = \{\varepsilon, a, ab, ac\}$, $\text{tr } (ab + ac) = \{\varepsilon, a, ab, ac\}$, $\text{tr } x = \{\varepsilon, a, ab, abb, abbb, \dots\}$



2.3 De Simone Format

- De Simone rule format (R. de Simone, 1985):

2.3 De Simone Format

- De Simone rule format (R. de Simone, 1985):
 - ▶ no negative premise
 - ▶ at most one premise per variable
 - ▶ linearity: each variable at most once in u + no x_i such that $x_i \xrightarrow{a_i} y_i$ in u

$$\frac{x_i \xrightarrow{a_i} y_i \quad \dots \quad x_j \xrightarrow{a_j} y_j}{\sigma(x_1 \dots x_n) \xrightarrow{c} u}$$

2.3 De Simone Format

- De Simone rule format (R. de Simone, 1985):
 - ▶ no negative premise
 - ▶ at most one premise per variable
 - ▶ linearity: each variable at most once in u + no x_i such that $x_i \xrightarrow{a_i} y_i$ in u

$$\frac{x_i \xrightarrow{a_i} y_i \quad \dots \quad x_j \xrightarrow{a_j} y_j}{\sigma(x_1 \dots x_n) \xrightarrow{c} u}$$

Theorem. (B. Bloom, 1994, restricted to GSOS)

De Simone \Rightarrow trace equivalence is a congruence

2.3 De Simone Format

Example. (With negative premises)

$$\dots + \frac{x \xrightarrow{a} y}{f(x) \xrightarrow{a} g(y)} \forall a \qquad \frac{x \not\xrightarrow{b}}{g(x) \xrightarrow{b} x}$$

2.3 De Simone Format

Example. (With negative premises)

$$\begin{array}{c}
 \dots + \frac{x \xrightarrow{a} y}{f(x) \xrightarrow{a} g(y)} \forall a \qquad \frac{x \not\xrightarrow{b}}{g(x) \xrightarrow{b} x} \\
 \\
 \begin{array}{c}
 f(a(b + c)) \\
 \downarrow a \\
 g(b + c) \\
 \downarrow \neg \\
 \downarrow
 \end{array}
 \qquad
 \begin{array}{c}
 \begin{array}{cc}
 a & f(ab + ac) & a \\
 \swarrow & & \searrow \\
 g(b) & & g(c) \\
 \downarrow & & \downarrow b \\
 \neg & & 0
 \end{array}
 \end{array}
 \end{array}$$

2.3 De Simone Format

Example. (With more than one premise per variable)

$$\dots + \frac{x \xrightarrow{a} y}{f'(x) \xrightarrow{a} g'(y)} \forall a \quad \frac{x \xrightarrow{b} x' \quad x \xrightarrow{c} x''}{g'(x) \xrightarrow{a} x' + x''}$$

2.3 De Simone Format

Example. (With more than one premise per variable)

$$\dots + \frac{x \xrightarrow{a} y}{f'(x) \xrightarrow{a} g'(y)} \forall a \qquad \frac{x \xrightarrow{b} x' \quad x \xrightarrow{c} x''}{g'(x) \xrightarrow{a} x' + x''}$$

$$f'(a(b + c))$$

$$\downarrow a$$

$$g'(b + c)$$

$$\downarrow a$$

$$0 + 0$$

$$\begin{array}{c} f'(ab + ac) \\ \swarrow a \quad \searrow a \\ g'(b) \quad g'(c) \\ \downarrow \quad \downarrow \end{array}$$

2.4 Abstract Trace Semantics

- abstract?

2.4 Abstract Trace Semantics

- abstract?

Remark. (B. Klin, 2005 and 2009) test suites and logical distributive laws

2.4 Abstract Trace Semantics

- abstract?

Remark. (B. Klin, 2005 and 2009) test suites and logical distributive laws

- recall behaviour $k : X \rightarrow HX = \mathcal{P}(X)^L$

2.4 Abstract Trace Semantics

- abstract?

Remark. (B. Klin, 2005 and 2009) test suites and logical distributive laws

- recall behaviour $k : X \rightarrow HX = \mathcal{P}(X)^L \cong (\mathcal{P}_{\text{ne}}(X) + 1)^L = BTX$
 - $T = \mathcal{P}_{\text{ne}}$: **effectful/branching** behaviour (non-determinism)
 - $B = (-+1)^L$: **pure** behaviour (labels and termination)

2.4 Abstract Trace Semantics

- abstract?

Remark. (B. Klin, 2005 and 2009) test suites and logical distributive laws

- recall behaviour $k : X \rightarrow HX = \mathcal{P}(X)^L \cong (\mathcal{P}_{\text{ne}}(X) + 1)^L = BTX$
 - $T = \mathcal{P}_{\text{ne}}$: **effectful/branching** behaviour (non-determinism)
 - $B = (-+1)^L$: **pure** behaviour (labels and termination)
- $\nu B = \mathfrak{T}$ set of (partial finite) traces: final B -coalgebra

2.4 Abstract Trace Semantics

- abstract?

Remark. (B. Klin, 2005 and 2009) test suites and logical distributive laws

- recall behaviour $k : X \rightarrow HX = \mathcal{P}(X)^L \cong (\mathcal{P}_{\text{ne}}(X) + 1)^L = BTX$
 - $T = \mathcal{P}_{\text{ne}}$: **effectful/branching** behaviour (non-determinism)
 - $B = (-+1)^L$: **pure** behaviour (labels and termination)
- $\nu B = \mathfrak{T}$ set of (partial finite) traces: final B -coalgebra
- categories of algebras (Eilenberg-Moore) $\mathbf{Set}^T = \mathbf{CSLat}$ category of unbounded complete semi-lattices

2.4 Abstract Trace Semantics

- abstract?

Remark. (B. Klin, 2005 and 2009) test suites and logical distributive laws

- recall behaviour $k : X \rightarrow HX = \mathcal{P}(X)^L \cong (\mathcal{P}_{\text{ne}}(X) + 1)^L = BTX$
 - $T = \mathcal{P}_{\text{ne}}$: **effectful/branching** behaviour (non-determinism)
 - $B = (-+1)^L$: **pure** behaviour (labels and termination)
- $\nu B = \mathfrak{T}$ set of (partial finite) traces: final B -coalgebra
- categories of algebras (Eilenberg-Moore) $\mathbf{Set}^T = \mathbf{CSLat}$ category of unbounded complete semi-lattices
- B lifts (with $\delta^B : TB \rightarrow BT$) and $\nu \overline{B} = \nu B$

2.4 Abstract Trace Semantics

- abstract GSOS theory in **CSLat**:

$$\bar{\rho} : \bar{\Sigma}(X \times \bar{B}X) \rightarrow \bar{B}(\bar{\Sigma}^*X)$$

2.4 Abstract Trace Semantics

- abstract GSOS theory in **CSLat**:

$$\bar{\rho} : \bar{\Sigma}(X \times \bar{B}X) \rightarrow \bar{B}(\bar{\Sigma}^* X)$$

- universal semantics $\tau : \mu\bar{\Sigma} \rightarrow \nu\bar{B}$

$$\begin{array}{ccccc} \bar{\Sigma}\mu\bar{\Sigma} & \longrightarrow & \mu\bar{\Sigma} & \longrightarrow & \bar{B}\mu\bar{\Sigma} \\ \bar{\Sigma}\tau \downarrow & & \tau \downarrow & & \bar{B}\tau \downarrow \\ \bar{\Sigma}\nu\bar{B} & \longrightarrow & \nu\bar{B} & \longrightarrow & \bar{B}\nu\bar{B} \end{array}$$

2.4 Abstract Trace Semantics

- abstract GSOS theory in **CSLat**:

$$\bar{\rho} : \bar{\Sigma}(X \times \bar{B}X) \rightarrow \bar{B}(\bar{\Sigma}^* X)$$

- universal semantics $\tau : \mu\bar{\Sigma} \rightarrow \nu\bar{B}$

$$\begin{array}{ccccc} \bar{\Sigma}\mu\bar{\Sigma} & \longrightarrow & \mu\bar{\Sigma} & \longrightarrow & \bar{B}\mu\bar{\Sigma} \\ \bar{\Sigma}\tau \downarrow & & \tau \downarrow & & \bar{B}\tau \downarrow \\ \bar{\Sigma}\nu\bar{B} & \longrightarrow & \nu\bar{B} & \longrightarrow & \bar{B}\nu\bar{B} \end{array}$$

- $\text{tr} : \mu\Sigma \xrightarrow{\xi} \mu\bar{\Sigma} \xrightarrow{\tau} \nu\bar{B}$

2.4 Abstract Trace Semantics

- abstract GSOS theory in **CSLat**:

$$\bar{\rho} : \bar{\Sigma}(X \times \bar{B}X) \rightarrow \bar{B}(\bar{\Sigma}^* X)$$

- universal semantics $\tau : \mu\bar{\Sigma} \rightarrow \nu\bar{B}$

$$\begin{array}{ccccc} \bar{\Sigma}\mu\bar{\Sigma} & \longrightarrow & \mu\bar{\Sigma} & \longrightarrow & \bar{B}\mu\bar{\Sigma} \\ \bar{\Sigma}\tau \downarrow & & \tau \downarrow & & \bar{B}\tau \downarrow \\ \bar{\Sigma}\nu\bar{B} & \longrightarrow & \nu\bar{B} & \longrightarrow & \bar{B}\nu\bar{B} \end{array}$$

- $\text{tr} : \mu\Sigma \xrightarrow{\xi} \mu\bar{\Sigma} \xrightarrow{\tau} \nu B$
- left square \Rightarrow tr is a congruence 😊

2.4 Abstract Trace Semantics

- abstract GSOS theory in **CSLat**:

$$\bar{\rho} : \bar{\Sigma}(X \times \bar{B}X) \rightarrow \bar{B}(\bar{\Sigma}^* X)$$

- universal semantics $\tau : \mu\bar{\Sigma} \rightarrow \nu\bar{B}$

$$\begin{array}{ccccc} \bar{\Sigma}\mu\bar{\Sigma} & \longrightarrow & \mu\bar{\Sigma} & \longrightarrow & \bar{B}\mu\bar{\Sigma} \\ \bar{\Sigma}\tau \downarrow & & \tau \downarrow & & \bar{B}\tau \downarrow \\ \bar{\Sigma}\nu\bar{B} & \longrightarrow & \nu\bar{B} & \longrightarrow & \bar{B}\nu\bar{B} \end{array}$$

- $\text{tr} : \mu\Sigma \xrightarrow{\xi} \mu\bar{\Sigma} \xrightarrow{\tau} \nu B$
- left square \Rightarrow tr is a congruence 😊
- how to get $\bar{\rho}$ natural transformation in **CSLat**?

2.5 Trace-GSOS

- rule format

- for each x_i and for each a ,
either $x_i \xrightarrow{a} y$ or $x_i \not\xrightarrow{a}$

$$\frac{x_1 \xrightarrow{a_1} y_1 \quad \dots \quad x_1 \xrightarrow{a_k} y_k \quad x_1 \not\xrightarrow{b_1} \dots \quad x_2 \dots \quad \dots}{\sigma(x_1 \dots x_n) \xrightarrow{c} u}$$

2.5 Trace-GSOS

- rule format

► for each x_i and for each a ,
either $x_i \xrightarrow{a} y$ or $x_i \not\xrightarrow{a}$

$$\frac{x_1 \xrightarrow{a_1} y_1 \quad \dots \quad x_1 \xrightarrow{a_k} y_k \quad x_1 \not\xrightarrow{b_1} \dots \quad x_2 \dots \quad \dots}{\sigma(x_1 \dots x_n) \xrightarrow{c} u}$$

Remark. Only **pure** premises: observe each variable at each label **once and only once**

2.5 Trace-GSOS

- rule format

▶ for each x_i and for each a ,
 either $x_i \xrightarrow{a} y$ or $x_i \not\xrightarrow{a}$

$$\frac{x_1 \xrightarrow{a_1} y_1 \quad \dots \quad x_1 \xrightarrow{a_k} y_k \quad x_1 \not\xrightarrow{b_1} \dots \quad x_2 \dots \quad \dots}{\sigma(x_1 \dots x_n) \xrightarrow{c} u}$$

Remark. Only **pure** premises: observe each variable at each label **once and only once**

- natural transformation

$$\rho : \Sigma(X \times HX) \rightarrow H\Sigma^*X$$

2.5 Trace-GSOS

- rule format

▶ for each x_i and for each a ,
 either $x_i \xrightarrow{a} y$ or $x_i \not\xrightarrow{a}$

$$\frac{x_1 \xrightarrow{a_1} y_1 \quad \dots \quad x_1 \xrightarrow{a_k} y_k \quad x_1 \not\xrightarrow{b_1} \dots \quad x_2 \dots \quad \dots}{\sigma(x_1 \dots x_n) \xrightarrow{c} u}$$

Remark. Only **pure** premises: observe each variable at each label **once and only once**

- natural transformation

$$\rho : \Sigma(X \times BTX) \rightarrow BT\Sigma^*X$$

2.5 Trace-GSOS

- rule format

► for each x_i and for each a ,
either $x_i \xrightarrow{a} y$ or $x_i \not\xrightarrow{a}$

$$\frac{x_1 \xrightarrow{a_1} y_1 \quad \dots \quad x_1 \xrightarrow{a_k} y_k \quad x_1 \not\xrightarrow{b_1} \dots \quad x_2 \dots \quad \dots}{\sigma(x_1 \dots x_n) \xrightarrow{c} u}$$

Remark. Only **pure** premises: observe each variable at each label **once and only once**

- natural transformation

$$\rho : \Sigma(X \times BX) \rightarrow BT\Sigma^*X$$

2.5 Trace-GSOS

- rule format

▶ for each x_i and for each a ,
 either $x_i \xrightarrow{a} y$ or $x_i \not\xrightarrow{a}$

$$\frac{x_1 \xrightarrow{a_1} y_1 \quad \dots \quad x_1 \xrightarrow{a_k} y_k \quad x_1 \not\xrightarrow{b_1} \dots \quad x_2 \dots \quad \dots}{\sigma(x_1 \dots x_n) \xrightarrow{c} u}$$

Remark. Only **pure** premises: observe each variable at each label **once and only once**

- natural transformation

$$\rho : \Sigma(X \times BX) \rightarrow BT\Sigma^*X$$

- De Simone rules \rightarrow Trace-GSOS rules:

$$\mathcal{S}(r) = \{r' \text{ Trace-GSOS rule} \mid \text{each premise of } r \text{ is a premise of } r'\}$$

2.5 Trace-GSOS

- characterize “good” Trace-GSOS specifications: smooth and minimally linear

2.5 Trace-GSOS

- characterize “good” Trace-GSOS specifications: smooth and minimally linear

Theorem. “good” Trace-GSOS specifications \iff De Simone specifications

2.5 Trace-GSOS

- characterize “good” Trace-GSOS specifications: smooth and minimally linear

Theorem. “good” Trace-GSOS specifications \iff De Simone specifications

Theorem. “good” Trace-GSOS specifications $\implies \rho$ yields a **natural** transformation $\bar{\rho}$ in CSLat

$$\bar{\rho} : \Sigma(X \times BX) \xrightarrow{\rho} BT\Sigma^*X \xrightarrow{BT\xi_X} BT\bar{\Sigma}^*X \xrightarrow{B\sigma_X} B\bar{\Sigma}^*X$$

2.5 Trace-GSOS

- characterize “good” Trace-GSOS specifications: smooth and minimally linear

Theorem. “good” Trace-GSOS specifications \iff De Simone specifications

Theorem. “good” Trace-GSOS specifications $\implies \rho$ yields a **natural** transformation $\bar{\rho}$ in CSLat

$$\bar{\rho} : \Sigma(X \times BX) \xrightarrow{\rho} BT\Sigma^*X \xrightarrow{BT\xi_X} BT\bar{\Sigma}^*X \xrightarrow{B\sigma_X} B\bar{\Sigma}^*X$$

-  find an easy/better abstract characterization of “good” Trace-GSOS

3 Conclusion...

3 Conclusion...

- Eilenberg-Moore trace semantics

3 Conclusion...

- Eilenberg-Moore trace semantics
- abstract GSOS in Eilenberg-Moore category

3 Conclusion...

- Eilenberg-Moore trace semantics
- abstract GSOS in Eilenberg-Moore category
- works for LTSs

3 Conclusion...

- Eilenberg-Moore trace semantics
- abstract GSOS in Eilenberg-Moore category
- works for LTSs
- 🚧 easier to check abstract condition

3 Conclusion...

- Eilenberg-Moore trace semantics
- abstract GSOS in Eilenberg-Moore category
- works for LTSs
- 🚧 easier to check abstract condition
- 🚧 other monads: probabilistic distributions

3 Conclusion...

- Eilenberg-Moore trace semantics
- abstract GSOS in Eilenberg-Moore category
- works for LTSs
- 🚧 easier to check abstract condition
- 🚧 other monads: probabilistic distributions
- 🚧 adequacy of operational models

3 Conclusion...

- Eilenberg-Moore trace semantics
- abstract GSOS in Eilenberg-Moore category
- works for LTSs
- 🚧 easier to check abstract condition
- 🚧 other monads: probabilistic distributions
- 🚧 adequacy of operational models

~ Thank you for your attention ~