From Regular Expressions to Star Fragments

Todd Schmid

St. Mary’s College of California (Bucknell University starting in July)

Based on Coalgebraic Completeness Theorems for Effectful Process Calculi, UCL, 2023 and joint work with

Wojciech Rozowski (UCL)  Tobias Kappé (Open Universiteit)

Dexter Kozen (Cornell University)  Jurriaan Rot (Radboud University)  Alexandra Silva (Cornell University)

LLAMA Seminar
This Talk

1. Regular expressions and regular languages
2. Axioms for language equivalence \(á \text{ la} \) Salomaa
3. Process (bisimilarity) semantics of regular expressions
4. Guarded Kleene Algebra with Tests mod bisimilarity
5. What these process algebras have in common
6. Star Fragments
7. Open Problems
Regular Languages

\[ X \to \{\bot, \top\} \times X^A \]

(Kleene, 1956)
Regular Languages

\[ X \to \{\bot, T\} \times X^A \]
Regular Languages

\[ X \rightarrow \{\bot, T\} \times X^A \]

(Kleene, 1956)
Regular Languages

\[ X \to \{\bot, T\} \times X^A \]

\[ a, b, aab \]

(Kleene, 1956)
Regular Languages

\[ X \to \{ \bot, T \} \times X^A \]

\( ab, aaab, bb \)

(Kleene, 1956)
Regular Languages

\[ X \rightarrow \{\perp, T\} \times X^A \]

\[ ab, aaab, bb, babb \]

(Kleene, 1956)
Regular Languages

\[ X \rightarrow \{\bot, \top\} \times X^A \]

\[ L = \{ab, aaab, bb, babb, \ldots\} \]

(Kleene, 1956)
Regular Languages

\[ X \rightarrow \{\bot, T\} \times X^A \]

Regular expressions: syntax for regular languages

\[ L = \{ab, aaab, bb, babb, \ldots\} = (aa + ba)^*(ab + bb) \]

(Kleene, 1956)
Regular Expressions and Regular Languages

RegEx \( \ni e, f ::= 0 \mid 1 \mid p \in \Sigma \mid e + f \mid ef \mid e^* \)

\[ L : \text{RegEx} \rightarrow \mathcal{P}(\Sigma^*) \]

\[ L(0) = \emptyset \quad L(1) = \{\varepsilon\} \quad L(p) = \{p\} \]

\[ L(e + f) = L(e) \cup L(f) \quad L(ef) = L(e)L(f) \quad L(e^*) = \bigcup_{n \in \omega} L(e)^n \]

(Kleene, 1956)

\[ L = L(r) \] iff \( L \) is recognized by a deterministic finite automaton.
Regular Expressions and Regular Languages

\[ q_0 \xrightarrow{a} q_1 \xrightarrow{a} q_2 \xrightarrow{b} q_3 \]

\[ p_0 \xrightarrow{a} p_1 \xrightarrow{a,b} p_2 \xrightarrow{b} \]
Regular Expressions and Regular Languages
Regular Expressions and Regular Languages
Regular Expressions and Regular Languages
Regular Expressions and Regular Languages
Regular Expressions and Regular Languages

bisimulation

Regular Expressions and Regular Languages
Regular Expressions and Regular Languages

For DFAs,
For DFAs,
- bisimilarity = language equivalence
Regular Expressions and Regular Languages

For DFAs,
- bisimilarity = language equivalence
- Using (Hopcroft, Karp, 1971), bisimilarity is checked in almost linear time
Regular Expressions and Regular Languages

For DFAs,
- bisimilarity = language equivalence
- Using (Hopcroft, Karp, 1971), bisimilarity is checked in almost linear time

\[ L((aa + ba)^*(ab + bb)) = L(((a + b)a)^*b) \]
For DFAs,
- bisimilarity = language equivalence
- (Hopcroft, Karp, 1971) Bisimilarity is checked in nearly linear time

(Kleene, 1956)
Give a complete axiomatization of language equivalence of regular expressions
**Axiomatizing Language Equivalence**

(Salomaa, 1964) A complete axiomatization of language equivalence of regular expressions:

<p>| | | | | | | | | | | | | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>$A_1$</td>
<td>$\alpha + (\beta + \gamma) = (\alpha + \beta) + \gamma$,</td>
<td>$A_7$</td>
<td>$\phi^* \alpha = \alpha$,</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>$A_2$</td>
<td>$\alpha(\beta\gamma) = (\alpha\beta)\gamma$,</td>
<td>$A_8$</td>
<td>$\phi \alpha = \phi$,</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>$A_3$</td>
<td>$\alpha + \beta = \beta + \alpha$,</td>
<td>$A_9$</td>
<td>$\alpha + \phi = \alpha$,</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>$A_4$</td>
<td>$\alpha(\beta + \gamma) = \alpha\beta + \alpha\gamma$,</td>
<td>$A_{10}$</td>
<td>$\alpha^* = \phi^* + \alpha^* \alpha$,</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>$A_5$</td>
<td>$(\alpha + \beta)\gamma = \alpha\gamma + \beta\gamma$,</td>
<td>$A_{11}$</td>
<td>$\alpha^* = (\phi^* + \alpha)^*$,</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>$A_6$</td>
<td>$\alpha + \alpha = \alpha$,</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

**R1 (Substitution).** Assume that $\gamma'$ is the result of replacing an occurrence of $\alpha$ by $\beta$ in $\gamma$. Then from the equations $\alpha = \beta$ and $\gamma = \delta$ one may infer the equation $\gamma' = \delta$ and the equation $\gamma' = \gamma$.

**R2 (Solution of equations).** Assume that $\beta$ does not possess e.w.p. Then from the equation $\alpha = \alpha\beta + \gamma$ one may infer the equation $\alpha = \gamma\beta^*$. 
Axiomatizing Language Equivalence

(Milner, 1984) Rephrased Salomaa’s rules as follows:

Salomaa [9] provides a complete inference system for star expressions under standard interpretation. When we dualise it, by writing $f \circ e$ for $e \circ f$ everywhere in Salomaa’s rules (which gives an equipotent system), it has the following rules:

\begin{align*}
A_1 & \quad e + (f + g) = (e + f) + g \\
A_2 & \quad (e \circ f) \circ g = e \circ (f \circ g) \\
A_3 & \quad e + f = f + e \\
A_4 & \quad (e + f) \circ g = e \circ g + f \circ g \\
A_5 & \quad e \circ (f + g) = e \circ f + e \circ g \\
A_6 & \quad e + e = e \\
R_2 & \quad \text{If } f \text{ does not possess e.w.p. then}
\end{align*}

from $e = f \circ e + h$ infer $e = f^* \circ h$.

(We have omitted $R_1$, the substitution rule.)
Salomaa [9] provides a complete inference system for star expressions under standard interpretation. When we dualise it, by writing \( f \circ e \) for \( e \circ f \) everywhere in Salomaa’s rules (which gives an equipotent system), it has the following rules:

<table>
<thead>
<tr>
<th>Rule</th>
<th>Equation</th>
</tr>
</thead>
<tbody>
<tr>
<td>A₁</td>
<td>( e + (f + g) = (e + f) + g )</td>
</tr>
<tr>
<td>A₂</td>
<td>( (e \circ f) \circ g = e \circ (f \circ g) )</td>
</tr>
<tr>
<td>A₃</td>
<td>( e + f = f + e )</td>
</tr>
<tr>
<td>A₄</td>
<td>( (e + f) \circ g = e \circ g + f \circ g )</td>
</tr>
<tr>
<td>A₅</td>
<td>( e \circ (f + g) = e \circ f + e \circ g )</td>
</tr>
<tr>
<td>A₆</td>
<td>( e + e = e )</td>
</tr>
<tr>
<td>A₇</td>
<td>( e \circ \phi^* = e )</td>
</tr>
<tr>
<td>A₈</td>
<td>( e \circ \phi = \phi )</td>
</tr>
<tr>
<td>A₉</td>
<td>( e + \phi = e )</td>
</tr>
<tr>
<td>A₁₀</td>
<td>( e^* = \phi^* + e \circ e^* )</td>
</tr>
<tr>
<td>A₁₁</td>
<td>( e^* = (\phi^* + e)^* )</td>
</tr>
</tbody>
</table>

If \( f \) does not possess e.w.p. then

from \( e = f \circ e + h \) infer \( e = f^* \circ h \).

(We have omitted \( R₁ \), the substitution rule.)
Deciding language equivalence

$$(aa + ba)^*(ab + bb)$$
Deciding language equivalence

\[(aa + ba)^*(ab + bb)\]

Regular Expressions

Thompson Construction,
SOS,
Antimirov Derivatives
Deciding language equivalence

Regular Expressions $\rightarrow$ Nondeterministic FAs

$$(aa + ba)^*(ab + bb)$$

$X \rightarrow \{\bot, T\} \times \mathcal{P}(X)^A$
Deciding language equivalence

\[(aa + ba)^* (ab + bb)\]

\[X \rightarrow \{\bot, T\} \times \mathcal{P}(X)^A\]

Regular Expressions \rightarrow \text{Nondeterministic FAs} \rightarrow \text{Determinize}

Thompson Construction, SOS, Antimirov Derivatives
Deciding language equivalence

\[(aa + ba^*) (ab + bb)\]

\[X \rightarrow \{\bot, T\} \times \mathcal{P}(X)^A\]

Regular Expressions  \[\rightarrow\]  Nondeterministic FAs  \[\rightarrow\]  DFAs

Thompson Construction, SOS, Antimirov Derivatives
Deciding language equivalence

\[(aa + ba)^*(ab + bb)\]

\[X \rightarrow \{\perp, T\} \times \mathcal{P}(X)^A\]

Regular Expressions \rightarrow Nondeterministic FAs \rightarrow DFAs \rightarrow Check for Bisimilarity

Thompson Construction, SOS, Antimirov Derivatives

Determinize
Deciding language equivalence

\[(aa + ba)^* (ab + bb)\]

\[X \rightarrow \{\bot, \top\} \times \mathcal{P}(X)^A\]

- Regular Expression
- Nondeterministic FAs
- DFAs
- Check for Bisimilarity

Thompson Construction,
Operational Semantics,
Antimirov Derivatives

(Milner, 1984)
Bisimilarity for NFAs is Finer than Language Equivalence

(Milner, 1984)
Bisimilarity for NFAs is Finer than Language Equivalence

(Milner, 1984)
Bisimilarity for NFAs is Finer than Language Equivalence

(Milner, 1984)
Bisimilarity for NFAs is Finer than Language Equivalence

(Milner, 1984)
Bisimilarity for NFAs is Finer than Language Equivalence

(Milner, 1984)
Bisimilarity for NFAs is Finer than Language Equivalence

(Milner, 1984)
Bisimilarity for NFAs is Finer than Language Equivalence

Not all axioms are sound!

(Milner, 1984)
Axiomatizing Bisimilarity of Regular Expressions

Salomaa [9] provides a complete inference system for star expressions under standard interpretation. When we dualise it, by writing $f \circ e$ for $e \circ f$ everywhere in Salomaa’s rules (which gives an equipotent system), it has the following rules:

$$
\begin{align*}
A_1 & \quad e + (f + g) = (e + f) + g \\
A_2 & \quad (e \circ f) \circ g = e \circ (f \circ g) \\
A_3 & \quad e + f = f + e \\
A_4 & \quad (e + f) \circ g = e \circ g + f \circ g \\
A_5 & \quad e \circ (f + g) = e \circ f + e \circ g \\
A_6 & \quad e + e = e \\
R_2 & \quad \text{If } f \text{ does not possess e.w.p. then}
\end{align*}
$$

from $e = f \circ e + h$ infer $e = f^* \circ h$.

(We have omitted $R_1$, the substitution rule.)

(Milner, 1984)
Salomaa [9] provides a complete inference system for star expressions under standard interpretation. When we dualise it, by writing $f \circ e$ for $e \circ f$ everywhere in Salomaa’s rules (which gives an equipotent system), it has the following rules:

\[
\begin{align*}
A_1 & \quad e + (f + g) = (e + f) + g \\
A_2 & \quad (e \circ f) \circ g = e \circ (f \circ g) \\
A_3 & \quad e + f = f + e \\
A_4 & \quad (e + f) \circ g = e \circ g + f \circ g \\
A_5 & \quad e \circ (f + g) = e \circ f + e \circ g \\
A_6 & \quad e + e = e \\
A_7 & \quad e \circ \phi^* = e \\
A_8 & \quad e \circ \phi = \phi \\
A_9 & \quad e + \phi = e \\
A_{10} & \quad e^* = \phi^* + e \circ e^* \\
A_{11} & \quad e^* = (\phi^* + e)^* \\
R_2 & \quad If \ f \ does \ not \ possess \ e.w.p. \ then \\
& \quad \quad from \ e = f \circ e + h \ infer \ e = f^* \circ h.
\end{align*}
\]

(We have omitted $R_1$, the substitution rule.)
Axiomatizing Bisimilarity of Regular Expressions

Salomaa [9] provides a complete inference system for star e, under the standard interpretation. When we dualise it, by writing $f \circ e$ for $e$, we get Salomaa's rules (which gives an equipotent system), it has the following:

$$
\begin{align*}
A_1 & \quad e + (f + g) = (e + f) + g \\
A_2 & \quad (e \circ f) \circ g = e \circ (f \circ g) \\
A_3 & \quad e + f = f + e \\
A_4 & \quad (e + f) \circ g = e \circ g + f \circ g \\
A_5 & \quad e \circ (f + g) = e \circ f + e \circ g \\
A_6 & \quad e + e = e \\
R_2 & \quad \text{If } f \text{ does not possess e.w.p. then} \\
& \quad \text{from } e = f \circ e + h \text{ infer } e = f^* \circ h.
\end{align*}
$$

(We have omitted $R_1$, the substitution rule.)

By deleting these axioms, Milner obtains a sound axiomatization of bisimilarity.

(Milner, 1984)
Salomaa [9] provides a complete inference system for star equivalence in the standard interpretation. When we dualise it, by writing \( f \circ e \) for \( e \circ f \), we find that Salomaa’s rules (which gives an equipotent system), it has the following axioms:

\[
\begin{align*}
A_1 & \quad e + (f + g) = (e + f) + g \\
A_2 & \quad (e \circ f) \circ g = e \circ (f \circ g) \\
A_3 & \quad e + f = f + e \\
A_4 & \quad (e + f) \circ g = e \circ g + f \circ g \\
A_5 & \quad e \circ (f + g) = e \circ f + e \circ g \\
A_6 & \quad e + e = e \\
R_2 & \quad \text{If } f \text{ does not possess e.w.p. then}
\end{align*}
\]

By deleting these axioms, Milner obtains a sound axiomatization of bisimilarity.

\[
\begin{align*}
A_7 & \quad e \circ \emptyset^* = e \\
A_8 & \quad e \circ \emptyset = \emptyset \\
A_9 & \quad e + \emptyset = e \\
A_{10} & \quad e^* = \emptyset^* + e \circ e^* \\
A_{11} & \quad e^* = (\emptyset^* + e)^* \\
A_8' & \quad \emptyset \circ e = \emptyset
\end{align*}
\]

from \( e = f \circ e + h \) infer \( e = f^* \circ h \).

(We have omitted \( R_1 \), the substitution rule.)

(Milner, 1984)
Salomaa [9] provides a complete inference system for star equations under the standard interpretation. When we dualise it, by writing $f \circ e$ for $e \circ f$, Salomaa’s rules (which gives an equipotent system), it has the following form:

- **Axiom 1**: $e + (f + g) = (e + f) + g$
- **Axiom 2**: $(e \circ f) \circ g = e \circ (f \circ g)$
- **Axiom 3**: $e + f = f + e$
- **Axiom 4**: $(e + f) \circ g = e \circ g + f \circ g$
- **Axiom 5**: $e \circ (f + g) = e \circ f + e \circ g$
- **Axiom 6**: $e + e = e$
- **Axiom 7**: $e \circ \phi^* = e$
- **Axiom 8**: $e \circ \phi = \phi$
- **Axiom 9**: $e + \phi = e$
- **Axiom 10**: $e^* = \phi^* + e \circ e^*$
- **Axiom 11**: $e^* = (\phi^* + e)^*$
- **Axiom 12**: $\phi \circ e = \phi$

By deleting these axioms, Milner obtains a sound axiomatization of bisimilarity.

(Milner, 1984)

Is this axiomatization complete?

(Milner, 1984)

(We have omitted R₁, the substitution rule.)
Axiomatizing Bisimilarity of Regular Expressions

Salomaa [9] provides a complete inference system for star expression via standard interpretation. When we dualise it, by writing \( f \circ e \) for \( e \), we get Salomaa’s rules (which gives an equipotent system), it has the following:

\[
\begin{align*}
A_1 & \quad e + (f + g) = (e + f) + g & A_7 & \quad e \circ \phi^* = e \\
A_2 & \quad (e \circ f) \circ g = e \circ (f \circ g) & A_8 & \quad e \circ \phi = \phi \\
A_3 & \quad e + f = f + e & A_9 & \quad e + \phi = e \\
A_4 & \quad (e + f) \circ g = e \circ g + f \circ g & A_{10} & \quad e^* = \phi^* + e \circ e^* \\
A_5 & \quad e \circ (f + g) = e \circ f + e \circ g & A_{11} & \quad e^* = (\phi^* + e)^* \\
A_6 & \quad e + e = e & A'_8 & \quad \phi \circ e = \phi \\
R_2 & \quad \text{If } f \text{ does not possess e.w.p. then} & & \\
& \quad \text{from } e = f \circ e + h \text{ infer } e = f^* \circ h. & & \\
\end{align*}
\]

(We have omitted \( R_1 \), the substitution rule.)

By deleting these axioms, Milner obtains a sound axiomatization of bisimilarity.

(Milner, 1984)

Is this axiomatization complete?

(Grabmayer, 2022)

Yes!
Axiomatizing Bisimilarity of Regular Expressions

An equivalent rendering of Milner’s axioms for regular expressions modulo bisimilarity:

\[
\begin{align*}
    e &= e + 0 \\
    e &= e + e \\
    f + e &= e + f \\
    e + (f + g) &= (e + f) + g \\
    0e &= 0 \\
    1e &= e \\
    e &= e1 \\
    e(fg) &= (ef)g \\
    (e + f)g &= eg + fg \\
    e^* &= (1 + e)^* \\
    e^* &= ee^* + 1 \\
    g &= eg + f \quad e \text{ guarded} \\
    g &= e^*f
\end{align*}
\]
Axiomatizing Bisimilarity of Regular Expressions

An equivalent rendering of Milner’s axioms for regular expressions modulo bisimilarity:

**Equational Branching Axioms**

\[
\begin{align*}
e &= e + 0 \\
e &= e + e \\
f + e &= e + f \\
e + (f + g) &= (e + f) + g
\end{align*}
\]

\[
\begin{align*}
0e &= 0 \\
1e &= e \\
e &= e1 \\
e(fg) &= (ef)g \\
(e + f)g &= eg + fg
\end{align*}
\]

\[
\begin{align*}
e^* &= (1 + e)^* \\
e^* &= ee^* + 1 \\
g &= eg + f \quad \text{e guarded} \\
g &= e^*f
\end{align*}
\]
Axiomatizing Bisimilarity of Regular Expressions

An equivalent rendering of Milner’s axioms for regular expressions modulo bisimilarity:

Equational Branching Axioms

\[
\begin{align*}
    e &= e + 0 \\
    e &= e + e \\
    f + e &= e + f \\
    e + (f + g) &= (e + f) + g
\end{align*}
\]

Sequencing Axioms

\[
\begin{align*}
    0e &= 0 \\
    1e &= e \\
    e &= e1 \\
    e(fg) &= (ef)g \\
    (e + f)g &= eg + fg
\end{align*}
\]

\[
\begin{align*}
    e^* &= (1 + e)^* \\
    e^* &= ee^* + 1 \\
    g &= eg + f 	ext{ guarded} \\
    g &= e^*f
\end{align*}
\]
Axiomatizing Bisimilarity of Regular Expressions

An equivalent rendering of Milner’s axioms for regular expressions modulo bisimilarity:

**Equational Branching Axioms**

\[
\begin{align*}
  e &= e + 0 \\
  e &= e + e \\
  f + e &= e + f \\
  e + (f + g) &= (e + f) + g
\end{align*}
\]

**Sequencing Axioms**

\[
\begin{align*}
  0e &= 0 \\
  1e &= e \\
  e &= e1 \\
  e(fg) &= (ef)g \\
  (e + f)g &= eg + fg
\end{align*}
\]

**Unique Guarded Fixed-point Axioms**

\[
\begin{align*}
  e^* &= (1 + e)^* \\
  e^* &= ee^* + 1 \\
  g &= eg + f \quad e \text{ guarded} \\
  g &= e^*f
\end{align*}
\]
Axiomatizing Bisimilarity of Regular Expressions

An equivalent rendering of Milner’s axioms for regular expressions modulo bisimilarity:

**Equational Branching Axioms**

\[
\begin{align*}
  e &= e + 0 \\
  e &= e + e \\
  f + e &= e + f \\
  e + (f + g) &= (e + f) + g 
\end{align*}
\]

**Sequencing Axioms**

\[
\begin{align*}
  0e &= 0 \\
  1e &= e \\
  e &= e1 \\
  e(fg) &= (ef)g \\
  (e + f)g &= eg + fg 
\end{align*}
\]

**Unguarded Fixed-point Axiom**

\[
e^* = (1 + e)^*
\]

**Unique Guarded Fixed-point Axioms**

\[
ge = eg + f \quad e \text{ guarded} \\
g = e^*f
\]
A Similar Situation: Guarded Kleene Algebra with Tests

\[ \alpha \mid p \quad \bar{\alpha} \mid q \quad \top \mid r \quad \alpha \lor \beta \mid p \]

\[ \beta \]
A Similar Situation: Guarded Kleene Algebra with Tests

- An algebra of propositional WHILE programs
A Similar Situation: Guarded Kleene Algebra with Tests

- An algebra of propositional WHILE programs
- (Kozen, Tseng, 2008) Syntax and language semantics from Kleene Algebra with Tests
A Similar Situation: Guarded Kleene Algebra with Tests

- An algebra of propositional WHILE programs
- (Kozen, Tseng, 2008) Syntax and language semantics from Kleene Algebra with Tests
- (Smolka, Foster, Hsu, Kappé, Kozen, Silva, 2019)
  - Operational semantics, almost linear decision procedure
  - Propose a Salomaa-like axiomatization of language equivalence
A Similar Situation: Guarded Kleene Algebra with Tests

- An algebra of propositional WHILE programs

- (Kozen, Tseng, 2008) Syntax and language semantics from Kleene Algebra with Tests

- (Smolka, Foster, Hsu, Kappé, Kozen, Silva, 2019)
  - Operational semantics, almost linear decision procedure
  - Propose a Salomaa-like axiomatization of language equivalence

- (S., Kappé, Kozen, Silva, 2021)
  - Infinite tree semantics = bisimilarity
  - Propose a Salomaa-like axiomatization of bisimilarity
Guarded Kleene Algebra with Tests

\[
\text{BExp} \ni b, c ::= 0 \mid 1 \mid t \in T \mid b \lor c \mid b \land c \mid \bar{b}
\]
Guarded Kleene Algebra with Tests

Generates an atomic Boolean algebra with atoms $At = 2^T$.

$\text{BExp} \ni b, c ::= 0 | 1 | t \in T | b \lor c | b \land c | \overline{b}$

$\text{BExp}/ =_{\text{BA}} \cong \mathcal{P}(2^T)$
Generates an atomic Boolean algebra with atoms $At = 2^T$. 
$\text{BExp} / =_{\text{BA}} \cong \mathcal{P}(2^T)$

$\exists b, c ::= 0 \mid 1 \mid t \in T \mid b \lor c \mid b \land c \mid \overline{b}$

$\exists e, f ::= b \in \text{BExp} \mid p \in \Sigma \mid e \lor_b f \mid ef \mid e^{(b)}$

(Smolka, Foster, Hsu, Kappé, Kozen, Silva, 2019)
Generates an atomic Boolean algebra with atoms $At = 2^T$.

$\mathcal{BExp} \ni b, c ::= 0 \mid 1 \mid t \in T \mid b \lor c \mid b \land c \mid \bar{b}$

$\mathcal{GExp} \ni e, f ::= b \in \mathcal{BExp} \mid p \in \Sigma \mid e +_b f \mid ef \mid e^{(b)}$

assert $b$
Guaranted Kleene Algebra with Tests

Generates an atomic Boolean algebra with atoms $At = 2^T$.
BExp/ =$_{BA} \cong \mathcal{P}(2^T)$

$BExp \ni b, c ::= 0 \mid 1 \mid t \in T \mid b \lor c \mid b \land c \mid \overline{b}$

$GExp \ni e, f ::= b \in BExp \mid p \in \Sigma \mid e +_b f \mid ef \mid e^{(b)}$

assert $b$  \hspace{1cm} do $p$

(Grumka, Foster, Hsu, Kappé, Kozen, Silva, 2019)
Generates an atomic Boolean algebra with atoms $At = 2^T$.

$$BExp \ni b, c ::= 0 | 1 | t \in T | b \lor c | b \land c | \bar{b}$$

$$GExp \ni e, f ::= b \in BExp | p \in \Sigma | e \cdot f | ef | e^{(b)}$$

assert $b$  
do $p$  
if $b$ then $e$ else $f$  

(Smolka, Foster, Hsu, Kappé, Kozen, Silva, 2019)
Guarded Kleene Algebra with Tests

Generates an atomic Boolean algebra with atoms $A_t = 2^T$.

$\text{BExp} \ni b, c ::= 0 \mid 1 \mid t \in T \mid b \lor c \mid b \land c \mid \bar{b}$

$\text{GExp} \ni e, f ::= b \in \text{BExp} \mid p \in \Sigma \mid e +_b f \mid ef \mid e^{(b)}$

assert $b$  do $p$  if $b$ then $e$ else $f$

$e$

$(\text{Smolka, Foster, Hsu, Kappé, Kozen, Silva, 2019})$
**Guarded Kleene Algebra with Tests**

Generates an atomic Boolean algebra with atoms $At = 2^T$.

$BExp ⊨ b, c ::= 0 \mid 1 \mid t \in T \mid b \lor c \mid b \land c \mid \bar{b}$

$GExp ⊨ e, f ::= b \in BExp \mid p \in \Sigma \mid e +_b f \mid ef \mid e^{(b)}$

- **assert** $b$
- **do** $p$
- **if** $b$ then $e$ else $f$
- $ef$
- **while** $b$ do $e$

$BExp/_{BA} \cong \mathcal{P}(2^T)$

(Smolka, Foster, Hsu, Kappé, Kozen, Silva, 2019)
Example of a GKAT Automaton

\[ (pr)^{(\alpha)} q (p \beta + \alpha \lor \beta) 0 \]

while \( \alpha \) do
  \[ p \]
  \[ r \]
  \[ q \]
  if \( \alpha \lor \beta \) then
    \[ p \]
    assert \( \beta \)
  else
    assert False

\( X \longrightarrow (\{\bot, \top\} + \Sigma \times X)^{At} \)

(Smolka, Foster, Hsu, Kappé, Kozen, Silva, 2019)
Axiomatizing GKAT Programs up to Language Equivalence

(Smolka et al., 2019) Proposed the following axiomatization of GKAT

<table>
<thead>
<tr>
<th>Guarded Union Axioms</th>
<th>Sequence Axioms (inherited from KA)</th>
</tr>
</thead>
<tbody>
<tr>
<td>U1. ( e+b e \equiv e )</td>
<td>S1. ( (e \cdot f) \cdot g \equiv e \cdot (f \cdot g) )</td>
</tr>
<tr>
<td>U2. ( e+b f \equiv f +_b e )</td>
<td>S2. ( 0 \cdot e \equiv 0 )</td>
</tr>
<tr>
<td>U3. ( (e+b f) +_c g \equiv e +_b (f +_c g) )</td>
<td>S3. ( e \cdot 0 \equiv 0 )</td>
</tr>
<tr>
<td>U4. ( e+b f \equiv be +_b f )</td>
<td>S4. ( 1 \cdot e \equiv e )</td>
</tr>
<tr>
<td>U5. ( eg +_b fg \equiv (e +_b f) \cdot g )</td>
<td>S5. ( e \cdot 1 \equiv e )</td>
</tr>
</tbody>
</table>

Guarded Loop Axioms

W1. \( e^{(b)} \equiv ee^{(b)} +_b 1 \) | (unrolling) |
W2. \( (e+_c 1)^{(b)} \equiv (ce)^{(b)} \) | (tightening) |
W3. \( g \equiv eg +_b f \) \[ g \equiv e^{(b)} f \] if \( E(e) \equiv 0 \) | (fixpoint) |

(Smolka, Foster, Hsu, Kappé, Kozen, Silva, 2019)
Axiomatizing GKAT Programs up to Language Equivalence

(Smolka et al., 2019) Proposed the following axiomatization of GKAT:

### Guarded Union Axioms

| U1. | \( e +_b e \equiv e \)  | (idempotence) |
| U2. | \( e +_b f \equiv f +_b e \)  | (skew commut.) |
| U3. | \( (e +_b f) +_c g \equiv e +_b (f +_c g) \)  | (skew assoc.) |
| U4. | \( e +_b f \equiv be +_b f \)  | (guardedness) |
| U5. | \( eg +_b fg \equiv (e +_b f) \cdot g \)  | (right distrib.) |

### Guarded Loop Axioms

| W1. | \( e^{(b)} \equiv ee^{(b)} +_b 1 \)  | (unrolling) |
| W2. | \( (e +_c 1)^{(b)} \equiv (ce)^{(b)} \)  | (tightening) |

### Sequence Axioms (inherited from KA)

| S1. | \((e \cdot f) \cdot g \equiv e \cdot (f \cdot g)\)  | (associativity) |
| S2. | \(0 \cdot e \equiv 0\)  | (absorbing left) |
| S3. | \(e \cdot 0 \equiv 0\)  | (absorbing right) |
| S4. | \(1 \cdot e \equiv e\)  | (neutral left) |
| S5. | \(e \cdot 1 \equiv e\)  | (neutral right) |

### Open Problem:

Are these axioms complete for language equivalence?

(Smolka, Foster, Hsu, Kappé, Kozen, Silva, 2019)
Axiomatizing GKAT Programs up to Bisimilarity

(S., Kappé, Kozen, Silva, 2021) Proposed the following axiomatization of GKAT/bisimilarity:

<table>
<thead>
<tr>
<th>Guarded Union Axioms</th>
<th>Sequence Axioms (inherited from KA)</th>
</tr>
</thead>
<tbody>
<tr>
<td>U1. ( e \mathbin{+}_b e \equiv e ) (idempotence)</td>
<td>S1. ( (e \cdot f) \cdot g \equiv e \cdot (f \cdot g) ) (associativity)</td>
</tr>
<tr>
<td>U2. ( e \mathbin{+}<em>b f \equiv f \mathbin{+}</em>{\bar{b}} e ) (skew commut.)</td>
<td>S2. ( 0 \cdot e \equiv 0 ) (absorbing left)</td>
</tr>
<tr>
<td>U3. ( (e \mathbin{+}_b f) \mathbin{+}<em>c g \equiv e \mathbin{+}</em>{b+c} (f \mathbin{+}_c g) ) (skew assoc.)</td>
<td>S3. ( e \cdot 0 \equiv 0 ) (absorbing right)</td>
</tr>
<tr>
<td>U4. ( e \mathbin{+}_b f \equiv be \mathbin{+}_b f ) (guardedness)</td>
<td>S4. ( 1 \cdot e \equiv e ) (neutral left)</td>
</tr>
<tr>
<td>U5. ( eg \mathbin{+}_b fg \equiv (e \mathbin{+}_b f) \cdot g ) (right distrib.)</td>
<td>S5. ( e \cdot 1 \equiv e ) (neutral right)</td>
</tr>
</tbody>
</table>

Guarded Loop Axioms

<table>
<thead>
<tr>
<th>W1. ( e^{(b)} \equiv ee^{(b)} \mathbin{+}_b 1 ) (unrolling)</th>
<th>W3. ( g \equiv eg \mathbin{+}_b f ) if ( E(e) \equiv 0 ) (fixpoint)</th>
</tr>
</thead>
<tbody>
<tr>
<td>W2. ( (e + c 1)^{(b)} \equiv (ce)^{(b)} ) (tightening)</td>
<td></td>
</tr>
</tbody>
</table>

\( g \equiv e^{(b)}f \)
## Axiomatizing GKAT Programs up to Bisimilarity

(S., Kappé, Kozen, Silva, 2021) Proposed the following axiomatization of GKAT/bisimilarity

<table>
<thead>
<tr>
<th>Guarded Union Axioms</th>
<th>Sequence Axioms (inherited from KA)</th>
</tr>
</thead>
<tbody>
<tr>
<td>U1. $e +_b e \equiv e$ (idempotence)</td>
<td>S1. $(e \cdot f) \cdot g \equiv e \cdot (f \cdot g)$ (associativity)</td>
</tr>
<tr>
<td>U2. $e +_b f \equiv f +_b e$ (skew commut.)</td>
<td>S2. $0 \cdot e \equiv 0$ (absorbing left)</td>
</tr>
<tr>
<td>U3. $(e +<em>b f) + c g \equiv e +</em>{bc} (f + c g)$ (skew assoc.)</td>
<td>S3. $e \cdot 0 \equiv 0$ (absorbing right)</td>
</tr>
<tr>
<td>U4. $e +_b f \equiv be +_b f$ (guardedness)</td>
<td>S4. $1 \cdot e \equiv e$ (neutral left)</td>
</tr>
<tr>
<td>U5. $eg +_b fg \equiv (e +_b f) \cdot g$ (right distrib.)</td>
<td>S5. $e \cdot 1 \equiv e$ (neutral right)</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Guarded Loop Axioms</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>W1. $e^{(b)} \equiv ee^{(b)} +_b 1$ (unrolling)</td>
<td>W3. $g \equiv eg +_b f$ if $E(e) \equiv 0$ (fixpoint)</td>
</tr>
<tr>
<td>W2. $(e + c 1)^{(b)} \equiv (ce)^{(b)}$ (tightening)</td>
<td></td>
</tr>
</tbody>
</table>

**Open Problem:** Are these axioms complete for bisimilarity?
### Axiomatizing GKAT Programs up to Bisimilarity

(S., Kappé, Kozen, Silva, 2021) Proposed the following axiomatization of GKAT/bisimilarity

<table>
<thead>
<tr>
<th>Guarded Union Axioms</th>
<th>Sequence Axioms (inherited from KA)</th>
</tr>
</thead>
<tbody>
<tr>
<td>U1. ( e +_b e \equiv e ) (idempotence)</td>
<td>S1. ( (e \cdot f) \cdot g \equiv e \cdot (f \cdot g) ) (associativity)</td>
</tr>
<tr>
<td>U2. ( e +_b f \equiv f +_b e ) (skew commut.)</td>
<td>S2. ( 0 \cdot e \equiv 0 ) (absorbing left)</td>
</tr>
<tr>
<td>U3. ( (e +_b f) +_c g \equiv e +_b (f +_c g) ) (skew assoc.)</td>
<td>S3. ( e \cdot 0 \equiv 0 ) (absorbing right)</td>
</tr>
<tr>
<td>U4. ( e +_b f \equiv be +_b f ) (guardedness)</td>
<td>S4. ( 1 \cdot e \equiv e ) (neutral left)</td>
</tr>
<tr>
<td>U5. ( eg +_b fg \equiv (e +_b f) \cdot g ) (right distrib.)</td>
<td>S5. ( e \cdot 1 \equiv e ) (neutral right)</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Guarded Loop Axioms</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>W1. ( e^{(b)} \equiv ee^{(b)} +_b 1 ) (unrolling)</td>
<td>W3. ( g \equiv eg +_b f ) if ( E(e) \equiv 0 ) (fixpoint)</td>
</tr>
<tr>
<td>W2. ( (e +_c 1)^{(b)} \equiv (ce)^{(b)} ) (tightening)</td>
<td></td>
</tr>
</tbody>
</table>

**Open Problem:** Are these axioms complete for bisimilarity?

Completeness here implies completeness for language equivalence.
Massaging the Syntax to Fit the Mould

\[ b \in \text{BExp} \quad \text{interpreted as} \quad \texttt{assert } b \]
Massaging the Syntax to Fit the Mould

\[ b \in \text{BExp} \quad \text{interpreted as} \quad \text{assert } b \]

The test 1 is interpreted as \text{assert True}
Massaging the Syntax to Fit the Mould

\[ b \in \text{BExp} \quad \text{interpreted as} \quad \text{assert } \ b \]

The test 1 is interpreted as \text{assert True}
and the test 0 is interpreted as \text{assert False}
Massaging the Syntax to Fit the Mould

\( b \in \text{BExp} \) interpreted as \texttt{assert } \( b \)

The test \texttt{1} is interpreted as \texttt{assert True}
and the test \texttt{0} is interpreted as \texttt{assert False}
Massaging the Syntax to Fit the Mould

\[ b \in \text{BExp} \quad \text{interpreted as} \quad \text{assert } b \]

The test 1 is interpreted as \text{assert True}
and the test 0 is interpreted as \text{assert False}

\text{assert True} \text{ is equivalent to simply skip}
Massaging the Syntax to Fit the Mould

\[
b \in \text{BExp} \quad \text{interpreted as} \quad \text{assert } b
\]

The test \(1\) is interpreted as \text{assert } \text{True}
and the test \(0\) is interpreted as \text{assert } \text{False}

\text{assert } \text{True} \text{ is equivalent to simply } \text{skip}
\text{assert } \text{False} \text{ is equivalent to simply } \text{crash}
Massaging the Syntax to Fit the Mould

\[ b \in \text{BExp} \quad \text{interpreted as} \quad \text{assert } b \]

The test 1 is interpreted as \text{assert } \text{True}
and the test 0 is interpreted as \text{assert } \text{False}

\text{assert } \text{True} \text{ is equivalent to simply skip}
\text{assert } \text{False} \text{ is equivalent to simply crash}

\[ \text{GKAT} \vdash b = 1 + b \ 0 \quad \text{or} \quad \text{if } b \text{ then skip else crash} \]
Guarded Kleene Algebra with Tests modulo Bisimulation

\[ \text{GExp}_{ts} \ni e,f ::= 0 \mid 1 \mid p \in \Sigma \mid e +_b f \mid ef \mid e^{(b)} \]

\[
\begin{align*}
e &= e +_\top f \\
e &= e +_b e \\
e +_b f &= f +_b e \\
e +_b (f +_c g) &= (e +_b f) +_{b \lor c} g
\end{align*}
\]

\[
\begin{align*}
0e &= 0 \\
1e &= e \\
e &= e1 \\
e(fg) &= (ef)g \\
(e +_b f)g &= eg +_b fg
\end{align*}
\]

\[
\begin{align*}
(1 +_c e)^{(b)} &= (0 +_c e)^{(b)} \\
e^{(b)} &= ee^{(b)} +^{(b)} 1 \\
g &= eg +^{(b)} f & \text{e guarded} \\
g &= e^{(b)} f
\end{align*}
\]
Guarded Kleene Algebra with Tests modulo Bisimulation

\[ \text{GExp}_{ts} \ni e,f ::= 0 \mid 1 \mid p \in \Sigma \mid e +_b f \mid ef \mid e^{(b)} \]

\begin{align*}
e &= e +_T f \\
e &= e +_b e \\
e +_b f &= f +_b e \\
e +_b (f +_c g) &= (e +_b f) +_b v e \ g
\end{align*}

\begin{align*}
0e &= 0 \\
1e &= e \\
e &= e1 \\
e(fg) &= (ef)g \\
(e +_b f)g &= eg +_b fg \\
(1 +_c e)^{(b)} &= (0 +_c e)^{(b)} \\
e^{(b)} &= ee^{(b)} +^{(b)} 1 \\
g &= eg +^{(b)} f & e \text{ guarded} \\
g &= e^{(b)} f
\end{align*}

Equational Branching Axioms
Guarded Kleene Algebra with Tests modulo Bisimulation

\[ \text{GExp}_{ts} \ni e,f ::= 0 \mid 1 \mid p \in \Sigma \mid e +_b f \mid ef \mid e^{(b)} \]

Equational Branching Axioms

\[
\begin{align*}
e &= e +_b f \\
e &= e +_b e \\
e +_b f &= f +_b e \\
e +_b (f +_c g) &= (e +_b f) +_b \vee e g
\end{align*}
\]

Sequencing Axioms

\[
\begin{align*}
0e &= 0 \\
1e &= e \\
e &= e1 \\
e(fg) &= (ef)g \\
(e +_b f)g &= eg +_b fg
\end{align*}
\]

\[
\begin{align*}
(1 + c e)^{(b)} &= (0 + c e)^{(b)} \\
\text{e}^{(b)} &= \text{ee}^{(b)} +^{(b)} 1 \\
g &= eg +^{(b)} f \\
e \text{ guarded} \\
g &= e^{(b)} f
\end{align*}
\]
Guarded Kleene Algebra with Tests modulo Bisimulation

GExp_{ts} \ni e,f ::= 0 \mid 1 \mid p \in \Sigma \mid e +_b f \mid ef \mid e^{(b)}

Equational Branching Axioms

\[
e = e + f \tag{1}
\]
\[
e = e +_b e \tag{2}
\]
\[
e +_b f = f +_b e \tag{3}
\]
\[
e +_b (f +_c g) = (e +_b f) +_b e +_c g \tag{4}
\]

Sequencing Axioms

\[
0e = 0 \tag{5}
\]
\[
1e = e \tag{6}
\]
\[
e = e1 \tag{7}
\]
\[
e(fg) = (ef)g \tag{8}
\]
\[
(e +_b f)g = eg +_b fg \tag{9}
\]

Unique Guarded Fixed-point Axioms

\[
(1 +_c e)^{(b)} = (0 +_c e)^{(b)} \tag{10}
\]
\[
e^{(b)} = ee^{(b)} +^{(b)} 1 \tag{11}
\]
\[
g = eg +^{(b)} f \quad \text{e guarded} \tag{12}
\]
\[
g = e^{(b)} f \tag{13}
\]
Guarded Kleene Algebra with Tests modulo Bisimulation

\[
\text{GExp}_{ts} \ni e, f ::= 0 \mid 1 \mid p \in \Sigma \mid e +_b f \mid ef \mid e^{(b)}
\]

Equational Branching Axioms

\[
\begin{align*}
    e &= e +^T f \\
    e &= e +_b e \\
    e +_b f &= f +_b e \\
    e +_b (f + c g) &= (e +_b f) +_{b \vee c} g
\end{align*}
\]

Sequencing Axioms

\[
\begin{align*}
    0e &= 0 \\
    1e &= e \\
    e &= e1 \\
    e(fg) &= (ef)g \\
    (e +_b f)g &= eg +_b fg
\end{align*}
\]

Unguarded Fixed-point Axiom

\[
(1 +_c e)^{(b)} = (0 +_c e)^{(b)}
\]

Unique Guarded Fixed-point Axioms

\[
\begin{align*}
    e^{(b)} &= ee^{(b)} +^{(b)} 1 \\
    g &= eg +^{(b)} f \quad e \text{ guarded} \\
    g &= e^{(b)} f
\end{align*}
\]
How to distinguish the examples!

**EQUATIONAL THEORY**

- $e = e +_T f$
- $e = e +_E e$
- $e +_b f = f +_b e$
- $e +_b (f +_c g) = (e +_b f) +_b v_c g$

**Equational Branching Axioms**

**Sequencing Axioms**

- $0e = 0$
- $1e = e$
- $e = e1$
- $e(fg) = (ef)g$
- $(e +_b f)g = eg +_b fg$

**FIXED POINT EQUATIONS**

- $(1e)_e$

**Unguarded Fixed-point Axioms**

- $e^{(b)} = ee^{(b)} +^{(b)} 1$
- $g = eg +^{(b)} f$
- $e$ guarded
- $g = e^{(b)} f$

**Unique Guarded Fixed-point Axioms**
How to distinguish the examples!

**Equational Theory**

- Equational Branching Axioms
  
  \[ e = e + \top \cdot f \]
  
  \[ e = e + \bot \cdot e \]
  
  \[ e +_b (f +_c g) = (e +_b f) +_b v_c g \]

- Sequencing Axioms
  
  \[ 0e = 0 \]
  
  \[ 1e = e \]
  
  \[ e = e1 \]
  
  \[ e(fg) = (ef)g \]
  
  \[ (e +_b f)g = eg +_b fg \]

- Unique Guarded Fixed-point Axioms
  
  \[ e^{(b)} = ee^{(b)} +^{(b)} 1 \]
  
  \[ g = eg +^{(b)} f \quad e \text{ guarded} \]
  
  \[ g = e^{(b)} f \]

Together, this data comprises a branching theory.
A Recipe

**Definition.** A *branching theory* consists of a
Definition. A branching theory consists of a
1. An algebraic signature $S = S_0 + S_2 \times \text{Id}^2$ consisting of constants and binary operations
Definition. A branching theory consists of a

1. An algebraic signature $S = S_0 + S_2 \times \text{Id}^2$ consisting of constants and binary operations
2. A set $T \subseteq S^*(Var) \times S^*(Var)$ of equations between $S$-terms
A Recipe

Definition. A branching theory consists of a
1. An algebraic signature $S = S_0 + S_2 \times \text{Id}^2$ consisting of constants and binary operations
2. A set $T \subseteq S^*(\text{Var}) \times S^*(\text{Var})$ of equations between $S$-terms
3. A fixed-point operator on $S$-terms $\text{fp } x : S^*(\{x\} + Y) \to S^*(Y)$ (natural in $Y$) satisfying
A Recipe

**Definition.** A *branching theory* consists of a

1. An algebraic signature $S = S_0 + S_2 \times \text{Id}^2$ consisting of **constants** and **binary operations**
2. A set $T \subseteq S^*(\text{Var}) \times S^*(\text{Var})$ of equations between $S$-terms
3. A **fixed-point operator** on $S$-terms $\text{fp } x : S^*(\{x\} + Y) \rightarrow S^*(Y)$ (natural in $Y$) satisfying

$$T \vdash \text{fp } x \ t(x, \overline{y}) = t(\text{fp } x \ t(x, \overline{y}), \overline{y})$$
Introducing: *Star Fragments*!

**Definition.** For a given branching theory \((S, T, fp)\), the set of *star expressions* is given by

\[
S_0 = \{0\}
\]

\[
S_2 = \{ +b \mid b \in \text{BExp}\}
\]
Introducing: Star Fragments!

**Definition.** For a given branching theory \((S, T, \text{fp})\), the set of star expressions is given by

\[
\text{StExp} \ni e, f ::= c \in S_0 \quad \text{raise } c
\]

**Eg.**

\[
S_0 = \{0\}
\]

\[
S_2 = \{+b \mid b \in \text{BExp}\}
\]
Introducing: Star Fragments!

**Definition.** For a given branching theory \((S, T, \text{fp})\), the set of *star expressions* is given by

\[
\text{StExp} \ni e, f ::= \begin{array}{l}
\text{raise } c \\
| \text{skip}
\end{array}
\]

\[
\begin{align*}
S_0 &= \{0\} \\
S_2 &= \{ +b \mid b \in \text{BExp} \}
\end{align*}
\]
Introducing: *Star Fragments!*

**Definition.** For a given branching theory \((S, T, fp)\), the set of *star expressions* is given by

\[
\text{StExp} \ni e, f ::= \begin{array}{l}
\text{raise } c \\
\mid 1 \\
\mid e +_\sigma f
\end{array}
\]

raise \(c\)

skip

branch into \(\sigma(e, f)\), where \(\sigma \in S_2\)

**Eg.**

\(S_0 = \{0\}\)

\(S_2 = \{ +_b \mid b \in \text{BExp}\}\)
Introducing: **Star Fragments!**

**Definition.** For a given branching theory \((S, T, \text{fp})\), the set of *star expressions* is given by

\[
\text{StExp} \ni e, f ::= c \in S_0 \quad \text{raise } c \\
| 1 \quad \text{skip} \\
| e +_\sigma f \quad \text{branch into } \sigma(e, f), \text{ where } \sigma \in S_2 \\
| ef \quad e; f
\]

**Eg.**

\[
S_0 = \{0\} \\
S_2 = \{ +_b | b \in \text{BExp}\}
\]
Introducing: Star Fragments!

**Definition.** For a given branching theory \((S, T, fp)\), the set of *star expressions* is given by

\[
\text{StExp} \ni e, f ::= c \in S_0 \\
\quad \mid 1 \\
\quad \mid e +_\sigma f \\
\quad \mid ef \\
\quad \mid e^{(\sigma)}
\]

- **raise** \(c\)
- **skip**
- **branch into** \(\sigma(e, f)\), where \(\sigma \in S_2\)
- **e;f**
- **recurse in** \(x = \sigma(e; x, T)\)

**Eg.**

\[ S_0 = \{0\} \]

\[ S_2 = \{+b \mid b \in \text{BExp}\} \]
Introducing: Star Fragments!

**Definition.** For a given branching theory \((S, T, fp)\), the set of *star expressions* is given by

\[
\text{StExp} \ni e, f ::= \begin{array}{ll}
    c & \text{raise } c \\
    1 & \text{skip} \\
    e + \sigma f & \text{branch into } \sigma(e, f), \text{ where } \sigma \in S_2 \\
    ef & e; f \\
    e^{(\sigma)} & \text{recurse in } x = \sigma(e; x, \top)
\end{array}
\]

**Eg.**

\[
S_0 = \{0\} \\
S_2 = \{ +_b \mid b \in \text{BExp}\}
\]
Introducing: Star Fragments!

Definition. For a given branching theory \((S, T, fp)\), the set of star expressions is given by

\[
\text{StExp} \ni e, f ::= c \in S_0 \\
| \quad 1 \\
| \quad e +_\sigma f \\
| \quad ef \\
| \quad e^{(\sigma)}
\]

- `raise c`  
- `skip`  
- `branch into \(\sigma(e, f)\)`, where \(\sigma \in S_2\)  
- `e; f`  
- `recurse in x = \sigma(e; x, T)`

Eg. \(GExp_{ts} \ni e, f ::= 0\)  

- `crash`  

\(S_0 = \{0\}\)  
\(S_2 = \{+_b | b \in BExp\}\)
Introducing: Star Fragments!

**Definition.** For a given branching theory \((S, T, \text{fp})\), the set of *star expressions* is given by

\[
\text{StExp} \ni e,f ::= c \in S_0 \\
| \ 1 \quad \text{raise } c \\
| e +_\sigma f \quad \text{skip} \\
| ef \quad \text{branch into } \sigma(e,f), \text{ where } \sigma \in S_2 \\
| e^\sigma \quad e;f \\
| e^{(\sigma)} \quad \text{recurse in } x = \sigma(e;x, T )
\]

**Eg.** \(\text{GExp}_{ts} \ni e,f ::= 0 \quad \text{crash} \quad S_0 = \{0\} \\
| 1 \quad \text{skip} \quad S_2 = \{ +_b | b \in \text{BExp} \} \)
Introducing: Star Fragments!

Definition. For a given branching theory \((S, T, fp)\), the set of star expressions is given by

\[
\text{StExp} \ni e, f ::= c \in S_0 \quad \text{raise } c
\]

\[
| 1 \quad \text{skip}
\]

\[
| e +_\sigma f \quad \text{branch into } \sigma(e, f), \text{ where } \sigma \in S_2
\]

\[
| ef \quad e; f
\]

\[
| e^{(\sigma)} \quad \text{recurse in } x = \sigma(e; x, T)
\]

Eg. \(\text{GExp}_{ts} \ni e, f ::= 0 \quad \text{crash}\)

\[
| 1 \quad \text{skip}
\]

\[
| e +_b f \quad \text{if } b \text{ then } e \text{ else } f \quad S_2 = \{+_b \mid b \in \text{BExp}\}
\]

\(S_0 = \{0\}\)
Introducing: *Star Fragments!*

**Definition.** For a given branching theory \((S, T, fp)\), the set of *star expressions* is given by

\[
\text{StExp} \ni e, f ::= \begin{cases} 
    c & \text{raise } c \\
    1 & \text{skip} \\
    e + \sigma f & \text{branch into } \sigma(e, f), \text{ where } \sigma \in S_2 \\
    ef & e; f \\
    e^{(\sigma)} & \text{recurse in } x = \sigma(e; x, \top)
\end{cases}
\]

**Eg.** \(\text{GExp}_{ts} \ni e, f ::= 0\)

\[
\begin{cases} 
    1 & \text{skip} \\
    e +_b f & \text{if } b \text{ then } e \text{ else } f \\
    ef & e; f
\end{cases}
\]
# Introducing: Star Fragments!

**Definition.** For a given branching theory \((S, T, \mathsf{fp})\), the set of *star expressions* is given by

\[
\text{StExp} \ni e, f ::= c \in S_0 \\
| 1 \\
| e + \sigma f \\
| ef \\
| e^{(\sigma)}
\]

*raise* \(c\)

*skip*

*branch into* \(\sigma(e, f)\), where \(\sigma \in S_2\)

*e;f*

*recurse in* \(x = \sigma(e; x, \top)\)

---

**Eg.** \(\text{GExp}_{ts} \ni e, f ::= 0 \)

*crash*

\(S_0 = \{0\}\)

| 1 \\
| \(e + \sigma f\) \\
| ef \\
| \(e^{(\sigma)}\)

*skip*

*if* \(b \text{ then } e \text{ else } f\) \(S_2 = \{ +_b \mid b \in \text{BExp}\}\)

*e;f*

*while* \(b \text{ do } e\)
Star Fragment Semantics

Operational semantics of regular expressions modulo bisimilarity:

\[ \text{Exp} \rightarrow \{ \bot, \top \} \times \mathcal{P}_{\text{fin}}(\text{Exp})^A \]

![Diagram of a state transition graph with states labeled with \(a\) and \(b\).]
Star Fragment Semantics

Operational semantics of regular expressions modulo bisimilarity:

\[ \ell : \text{Exp} \rightarrow \mathcal{P}_{\text{fin}}( \mathcal{T} + \mathcal{A} \times \text{Exp} ) \]
Star Fragment Semantics

Operational semantics of regular expressions modulo bisimilarity:

\[ \ell : \text{Exp} \longrightarrow \mathcal{P}_{\text{fin}}( T + A \times \text{Exp}) \]

\[ \ell(0) = \emptyset \quad \ell(1) = \{ T \} \quad \ell(a) = \{(a,1)\} \quad \ell(e + f) = \ell(e) \cup \ell(f) \]
Star Fragment Semantics

Operational semantics of regular expressions modulo bisimilarity:

\[ \ell : \text{Exp} \longrightarrow \mathcal{P}_{\text{fin}}(T + A \times \text{Exp}) \]

\[ \ell(0) = \emptyset \quad \ell(1) = \{T\} \quad \ell(a) = \{(a,1)\} \quad \ell(e + f) = \ell(e) \cup \ell(f) \]

and if \( \ell(e) = \{T, (a_1, e_1), \ldots, (a_n, e_n)\} \), then
Star Fragment Semantics

Operational semantics of regular expressions modulo bisimilarity:

\[ \ell : \text{Exp} \longrightarrow \mathcal{P}_{\text{fin}}( T + A \times \text{Exp}) \]

\[ \ell(0) = \emptyset \quad \ell(1) = \{ T \} \quad \ell(a) = \{(a,1)\} \quad \ell(e + f) = \ell(e) \cup \ell(f) \]

and if \( \ell(e) = \{ T, (a_1, e_1), \ldots, (a_n, e_n) \} \), then

\[ \ell(ef) = \ell(f) \cup \{(a_1, e_1f), \ldots, (a_n, e_nf)\} \quad \text{and} \quad \ell(e^*) = \{ T, (a_1, e_1e^*), \ldots, (a_n, e_ne^*) \} \]
Star Fragment Semantics

Operational semantics of regular expressions modulo bisimilarity:

\[ \text{Exp} \rightarrow \mathcal{P}_{\text{fin}}(\mathcal{T} + A \times \text{Exp}) \]

Operational semantics of GKAT expressions modulo bisimilarity:

\[ \text{GExp} \rightarrow ((\{\bot, \top\} + \Sigma \times \text{GExp})^{\mathcal{A}_t}) \]
Star Fragment Semantics

Operational semantics of regular expressions modulo bisimilarity:

\[
\text{Exp} \rightarrow \mathcal{P}_{\text{fin}}(\top + A \times \text{Exp})
\]

Operational semantics of GKAT expressions modulo bisimilarity:

\[
\text{GExp} \rightarrow (\bot + (\top + \Sigma \times \text{GExp}))^{At}
\]
Star Fragment Semantics

Operational semantics of regular expressions modulo bisimilarity:

\[
\text{Exp} \rightarrow \mathcal{P}_{\text{fin}}(\top + \text{Act} \times \text{Exp})
\]

Operational semantics of GKAT expressions modulo bisimilarity:

\[
\text{GExp} \rightarrow (\bot + (\top + \text{Act} \times \text{GExp}))^{At}
\]
Star Fragment Semantics

Operational semantics of regular expressions modulo bisimilarity:

\[
\text{Exp} \rightarrow \mathcal{P}_{\text{fin}}( \top + \text{Act} \times \text{Exp})
\]

Operational semantics of GKAT expressions modulo bisimilarity:

\[
\text{GExp} \rightarrow (\bot + (\top + \text{Act} \times \text{GExp}))^{\text{At}}
\]

**Observe:** Format is \( \top + \text{Act} \times (\_\_) \) wrapped in \( M(\_\_) \).

\[\mathcal{P}_{\text{fin}}(\_\_) \quad \text{— the finite powerset monad}\]

\[(\bot + (\_\_))^{\text{At}} \quad \text{— the partial functions monad}\]
Star Fragment Semantics: Branching Types

Fix an algebraic signature $S = S_0 + S_2 \times \text{Id}^2$ and a set of equations $T \subseteq S^*(V) \times S^*(V)$. 
Star Fragment Semantics: Branching Types

Fix an algebraic signature $S = S_0 + S_2 \times \text{Id}^2$ and a set of equations $T \subseteq S^*(V) \times S^*(V)$.

**Definition.** A monad is $M$ presented by the equational theory $(S, T)$ if there is an isomorphism

$$M \cong S^*(-)\mathcal{E}_T$$

i.e., the monad $M$ is a free-algebra construction for $(S, T)$. 
Star Fragment Semantics: Branching Types

Fix an algebraic signature \( S = S_0 + S_2 \times \text{Id}^2 \) and a set of equations \( T \subseteq S^*(V) \times S^*(V) \).

**Definition.** A monad is \( M \) presented by the equational theory \((S, T)\) if there is an isomorphism

\[
M \cong S^*( - )/ \vdash_T
\]

i.e., the monad \( M \) is a free-algebra construction for \((S, T)\).

**Example.** The equational theory in Salomaa/Milner’s axioms captures semilattices with bottom.

\[
\begin{align*}
e &= e + 0 \\
e &= e + e \\
f + e &= e + f \\
e + (f + g) &= (e + f) + g
\end{align*}
\]
Star Fragment Semantics: Branching Types

Fix an algebraic signature $S = S_0 + S_2 \times \text{Id}^2$ and a set of equations $T \subseteq S^*(V) \times S^*(V)$.

**Definition.** A monad is *$M$ presented* by the equational theory $(S, T)$ if there is an isomorphism

$$M \cong S^*(\cdot) / \models T$$

i.e., the monad $M$ is a *free-algebra construction* for $(S, T)$.

**Example.** The equational theory in Salomaa/Milner’s axioms captures *semilattices with bottom*.

\[
\begin{align*}
e &= e + 0 \\
  e &= e + e \\
  f + e &= e + f \\
  e + (f + g) &= (e + f) + g
\end{align*}
\]
Star Fragment Semantics: Branching Types

Fix an algebraic signature $S = S_0 + S_2 \times \text{Id}^2$ and a set of equations $T \subseteq S^*(V) \times S^*(V)$.

**Definition.** A monad is $M$ presented by the equational theory $(S, T)$ if there is an isomorphism

$$M \cong S^*(\_)/\equiv_T$$

i.e., the monad $M$ is a free-algebra construction for $(S, T)$.

**Example.** The equational theory in Salomaa/Milner’s axioms captures semilattices with bottom.

\[
\begin{align*}
e &= e + 0 \\
e &= e + e \\
f + e &= e + f \\
e + (f + g) &= (e + f) + g
\end{align*}
\]

presents

**Finite Powerset Monad** $\mathcal{P}_{\text{fin}}$
Star Fragment Semantics: Branching Types

Fix an algebraic signature $S = S_0 + S_2 \times \text{Id}^2$ and a set of equations $T \subseteq S^*(V) \times S^*(V)$.

**Definition.** A monad is $M$ presented by the equational theory $(S, T)$ if there is an isomorphism

$$M \cong S^*(-) / =_T$$

i.e., the monad $M$ is a free-algebra construction for $(S, T)$.

**Example.** The equational theory in Salomaa/Milner’s axioms captures semilattices with bottom.

\[
\begin{align*}
e &= e + 0 \\
e &= e + e \\
f + e &= e + f \\
e + (f + g) &= (e + f) + g
\end{align*}
\]

presents

Finite Powerset Monad $\mathcal{P}_{\text{fin}}$

$$U_1, U_2 \in \mathcal{P}_{\text{fin}}(X)$$
Star Fragment Semantics: Branching Types

Fix an algebraic signature $S = S_0 + S_2 \times \text{Id}^2$ and a set of equations $T \subseteq S^*(V) \times S^*(V)$.

**Definition.** A monad is $M$ presented by the equational theory $(S, T)$ if there is an isomorphism

$$M \cong S^*(-) / =_T$$

i.e., the monad $M$ is a free-algebra construction for $(S, T)$.

**Example.** The equational theory in Salomaa/Milner’s axioms captures semilattices with bottom.

\[
\begin{align*}
e &= e + 0 \\
e &= e + e \\
f + e &= e + f \\
e + (f + g) &= (e + f) + g
\end{align*}
\]

presents

**Finite Powerset Monad** $\mathcal{P}_{\text{fin}}$

\[
\begin{align*}
U_1, U_2 &\in \mathcal{P}_{\text{fin}}(X) \\
U_1 + U_2 &= U_1 \cup U_2
\end{align*}
\]
Star Fragment Semantics: Branching Types

Fix an algebraic signature \( S = S_0 + S_2 \times \text{Id}^2 \) and a set of equations \( T \subseteq S^*(V) \times S^*(V) \).

**Definition.** A monad is \( M \) presented by the equational theory \((S, T)\) if there is an isomorphism
\[
M \cong S^*(\cdot) / \ =_T
\]
i.e., the monad \( M \) is a free-algebra construction for \((S, T)\).

**Example.** The equational theory in Salomaa/Milner’s axioms captures *semilattices with bottom*.

\[
\begin{align*}
e &= e + 0 \\
e &= e + e \\
f + e &= e + f \\
e + (f + g) &= (e + f) + g
\end{align*}
\]

**Finite Powerset Monad** \( \mathcal{P}_{\text{fin}} \)

\[
\begin{align*}
U_1, U_2 &\in \mathcal{P}_{\text{fin}}(X) \\
U_1 + U_2 &= U_1 \cup U_2 \\
0 &= \emptyset
\end{align*}
\]
Star Fragment Semantics: Branching Types

Fix an algebraic signature \( S = S_0 + S_2 \times \text{Id}^2 \) and a set of equations \( T \subseteq S^*(V) \times S^*(V) \).

**Definition.** A monad is \( M \) *presented* by the equational theory \((S, T)\) if there is an isomorphism
\[
M \cong S^*(\_)/\equiv_T
\]
i.e., the monad \( M \) is a *free-algebra construction* for \((S, T)\).

**Example.** The equational theory in Salomaa/Milner’s axioms captures *semilattices with bottom*.

\[
\begin{align*}
e &= e + 0 \\
e &= e + e \\
f + e &= e + f \\
e + (f + g) &= (e + f) + g
\end{align*}
\]

presents

**Finite Powerset Monad** \( P_{\text{fin}} \)

\[
\begin{align*}
U_1, U_2 &\in P_{\text{fin}}(X) \\
U_1 + U_2 &= U_1 \cup U_2 \\
0 &= \emptyset
\end{align*}
\]

**Definition.** A monad that is presented by \((S, T)\) is a *branching type* of the branching theory.
Star Fragment Semantics: Unguarded Fixed-points

Last ingredient of a branching theory is the **fixed-point operator** $\text{fp } x: S^*({x} + Y) \rightarrow S^*(Y)$

\[
T \vdash \text{fp } x \ t(x, \vec{y}) = t( \text{fp } x \ t(x, \vec{y}), \vec{y})
\]
Star Fragment Semantics: Unguarded Fixed-points

Last ingredient of a branching theory is the **fixed-point operator** \( \text{fp} \ x : S^*(\{x\} + Y) \rightarrow S^*(Y) \)

\[
T \vdash \text{fp} \ x \ t(x, \vec{y}) = t(\text{fp} \ x \ t(x, \vec{y}), \vec{y})
\]

We obtain an operator on \( M \) that performs a type of iteration determined by \( \text{fp} \ x \)

\[
\begin{align*}
S^*(\{x\} + Y)/=_{T} & \xrightarrow{\text{fp} \ x} S^*(Y)/=_{T} \\
\cong & \uparrow \\
M(\{x\} + Y) & \xrightarrow{\text{fp} \ x} M(Y)
\end{align*}
\]
Star Fragment Semantics: Unguarded Fixed-points

Last ingredient of a branching theory is the **fixed-point operator** \( \text{fp } x : S^*(\{x\} + Y) \rightarrow S^*(Y) \)

\[
T \vdash \text{fp } x \ t(x, \vec{y}) = t( \text{fp } x \ t(x, \vec{y}), \vec{y})
\]

We obtain an operator on \( M \) that performs a type of iteration determined by \( \text{fp } x \)

**Example.** The operator \( \text{fp } x \ t(x, \vec{y}) = t(0, \vec{y}) \) on semilattice terms is a fixed-point operator:
Star Fragment Semantics: Unguarded Fixed-points

Last ingredient of a branching theory is the **fixed-point operator** $\text{fp } x : S^*({x} + Y) \rightarrow S^*(Y)$

$$
T \vdash \text{fp } x \ t(x, \bar{y}) = t(\text{fp } x \ t(x, \bar{y}),\bar{y})
$$

We obtain an operator on $M$ that performs a type of iteration determined by $\text{fp } x$

Example. The operator $\text{fp } x \ t(x, \bar{y}) = t(0, \bar{y})$ on semilattice terms is a fixed-point operator:

$$
T_{SL} \vdash \text{fp } x \ (x + y) = 0 + y = y = y + y = (\text{fp } x \ (x + y)) + y
$$
Star Fragment Semantics: Unguarded Fixed-points

Last ingredient of a branching theory is the **fixed-point operator** \( \text{fp } x : S^*(\{x\} + Y) \rightarrow S^*(Y) \)

\[
T \vdash \text{fp } x \ t(x, \vec{y}) = t( \text{fp } x \ t(x, \vec{y}), \vec{y})
\]

We obtain an operator on \( M \) that performs a type of iteration determined by \( \text{fp } x \)

\[
\begin{align*}
S^*(\{x\} + Y) / =_T & \xrightarrow{\text{fp } x} S^*(Y) / =_T \\
M(\{x\} + Y) & \xrightarrow{\text{fp } x} M(Y)
\end{align*}
\]

**Example.** The operator \( \text{fp } x \ t(x, \vec{y}) = t(0, \vec{y}) \) on semilattice terms is a fixed-point operator:

\[
T_{SL} \vdash \text{fp } x \ (x + y) = 0 + y = y = y + y = (\text{fp } x \ (x + y)) + y
\]

Given \( U \subseteq \{x\} + Y \), this corresponds to
Star Fragment Semantics: Unguarded Fixed-points

Last ingredient of a branching theory is the fixed-point operator \( \text{fp } x : S^*({x} + Y) \rightarrow S^*(Y) \)

\[
T \vdash \text{fp } x \ t(x, \vec{y}) = t(\text{fp } x \ t(x, \vec{y}), \vec{y})
\]

We obtain an operator on \( M \) that performs a type of iteration determined by \( \text{fp } x \)

**Example.** The operator \( \text{fp } x \ t(x, \vec{y}) = t(0, \vec{y}) \) on semilattice terms is a fixed-point operator:

\[
S^*({x} + Y) \simeq_T \text{fp } x \rightarrow S^*(Y) \simeq_T \\
\uparrow \sim \downarrow \sim \\
M({x} + Y) \xrightarrow{\text{fp } x} M(Y)
\]

Given \( U \subseteq {x} + Y \), this corresponds to

\[
\text{fp } x \ (U) = U - {x}
\]
Star Fragment Semantics

Operational semantics is given by a map

\[ \ell : \text{StExp} \longrightarrow M( T + Act \times \text{StExp}) \]
Star Fragment Semantics

Operational semantics is given by a map

$$\ell : \text{StExp} \rightarrow M(\mathcal{T} + \text{Act} \times \text{StExp})$$

$$\ell(c) = c$$
Star Fragment Semantics

Operational semantics is given by a map

\[ \ell : \text{StExp} \longrightarrow M( T + Act \times \text{StExp}) \]

\[ \ell(c) = c \quad \ell(1) = T \]

\[
\begin{array}{c}
\bigcirc \quad \ell(c) = c \\
1 \quad \ell(1) = T
\end{array}
\]
Star Fragment Semantics

Operational semantics is given by a map

\[ \ell : \text{StExp} \rightarrow M(\mathbb{T} + \text{Act} \times \text{StExp}) \]

\[ \ell(c) = c \quad \ell(1) = \top \quad \ell(p) = (p, 1) \]

Diagram:

- \( c \rightarrow c \)
- \( 1 \rightarrow \)
- \( p \rightarrow 1 \)
Star Fragment Semantics

Operational semantics is given by a map

\[ \ell : \text{StExp} \rightarrow M(\ T + \text{Act} \times \text{StExp}) \]

\[ \ell(c) = c \quad \ell(1) = T \quad \ell(p) = (p, 1) \]

\[ \ell(e +_\sigma f) = \sigma(\ell(e), \ell(f)) \]

\[ \ell(e) \]

\[ e +_\sigma f \]

\[ \ell(f) \]

\[ \sigma \]
Star Fragment Semantics

Operational semantics is given by a map

\[ \ell : \text{StExp} \rightarrow M(\ T + \text{Act} \times \text{StExp}) \]

- \( \ell(c) = c \)
- \( \ell(1) = T \)
- \( \ell(p) = (p,1) \)
- \( \ell(e +_\sigma f) = \sigma(\ell(e), \ell(f)) \)
- \( \ell(e_1 f) = t(\ell(f), (p_1, e_1 f), \ldots, (p_n, e_n f)) \)

If \( \ell(e) = t(\ T, (p_1, e_1), \ldots, (p_n, e_n)) \), then

\[ \ell(ef) = t(\ell(f), (p_1, e_1 f), \ldots, (p_n, e_n f)) \]
Star Fragment Semantics

If $\ell(e) = t(\top, (p_1, e_1), \ldots, (p_1, e_1))$, then

$$\ell(e^{(\sigma)}) = \text{fp } x \sigma(t(x, (p_1, e_1^{(\sigma)}), \ldots, (p_1, e_1^{(\sigma)})), \top)$$
Star Fragment Semantics

If $\ell(e) = t(\top, (p_1, e_1), \ldots, (p_1, e_1))$, then

$$\ell(e^{(\sigma)}) = \text{fp } x \sigma(t(x, (p_1, e_1^{(\sigma)}), \ldots, (p_1, e_1^{(\sigma)})), \top)$$
Star Fragment Semantics

If $\ell(e) = t(\top, (p_1, e_1), \ldots, (p_1, e_1))$, then

$$\ell(e^{(\sigma)}) = \text{fp } x \sigma(t(x, (p_1, e_1^{(\sigma)}), \ldots, (p_1, e_1^{(\sigma)})), \top)$$

**Example.** For regular expressions, if $p \in Act$, then
Star Fragment Semantics

If \( \ell(e) = t(\top, (p_1, e_1), \ldots, (p_1, e_1)) \), then

\[
\ell(e^{(\sigma)}) = \bigvee p \ x \ \sigma(t(x, (p_1, e_1^{(\sigma)}), \ldots, (p_1, e_1^{(\sigma)})), \top)
\]

**Example.** For regular expressions, if \( p \in Act \), then
Star Fragment Semantics

If \( \ell(e) = t(\top, (p_1, e_1), \ldots, (p_1, e_1)) \), then

\[
\ell(e^{(\sigma)}) = \text{fp} \ x \ \sigma(t(x, (p_1, e_1^{(\sigma)}), \ldots, (p_1, e_1^{(\sigma)})), \top)
\]

**Example.** For regular expressions, if \( p \in \text{Act} \), then

\[
\ell(1 + p) = \{ \top, (p, 1) \}
\]
Star Fragment Semantics

If \( \ell(e) = t(\top, (p_1, e_1), \ldots, (p_1, e_1)) \), then

\[
\ell(e^{(\sigma)}) = \text{fp } x \ \sigma(t(x, (p_1, e_1^{(\sigma)}), \ldots, (p_1, e_1^{(\sigma)})), \top)
\]

**Example.** For regular expressions, if \( p \in \text{Act} \), then

\[
\ell(1 + p) = \{ \top, (p, 1) \}
\]
Star Fragment Semantics

If $\ell(e) = t(\top, (p_1, e_1), \ldots, (p_1, e_1))$, then

$$\ell(e^{(\sigma)}) = \text{fp } x \sigma(t(x, (p_1, e_1^{(\sigma)}), \ldots, (p_1, e_1^{(\sigma)})), \top)$$

**Example.** For regular expressions, if $p \in \text{Act}$, then

$$\ell(1 + p) = \{ \top, (p, 1) \}$$

$$\ell((1 + p)^*) = \text{fp } x \{ x, (p, 1e^{(\sigma)}) \} \cup \{ \top \}$$
Star Fragment Semantics

If $\ell(e) = t(\top, (p_1, e_1), \ldots, (p_1, e_1))$, then

$$\ell(e^{(\sigma)}) = \text{fp } x \text{ } \sigma(t(x, (p_1, e_1^{(\sigma)}), \ldots, (p_1, e_1^{(\sigma)})), \top)$$

**Example.** For regular expressions, if $p \in Act$, then

$$\ell(1 + p) = \{ \top , (p,1) \}$$

$$\ell((1 + p)*) = \text{fp } x \{x, (p,1e^{(\sigma)})\} \cup \{ \top \}$$

$$= \{(p,1e^{(\sigma)}) , \top \}$$
Star Fragment Semantics

If $\ell(e) = t(\top, (p_1, e_1), \ldots, (p_1, e_1))$, then

$$\ell(e^{(\sigma)}) = \text{fp } x \sigma(t(x, (p_1, e_1^{(\sigma)}), \ldots, (p_1, e_1^{(\sigma)})), \top)$$

**Example.** For regular expressions, if $p \in \text{Act}$, then

$$\ell(1 + p) = \{ \top, (p, 1) \}$$

$$\ell((1 + p)^*) = \text{fp } x \{ x, (p, 1e^{(\sigma)}) \} \cup \{ \top \}$$

$$= \{ (p, 1e^{(\sigma)}), \top \}$$
An Axiomatization of Star Fragments modulo Bisimilarity?

**Equational Branching Axioms**

\[
T
\]

\[
\begin{align*}
ce &= c \\
1e &= e \\
e &= e1 \\
e(fg) &= (ef)g \\
(e +_\sigma f)g &= eg +_\sigma fg
\end{align*}
\]

**Sequencing Axioms**

**General Unguarded Fixed-point Axiom**

\[
t(1, g)^{(\sigma)} = \text{fp } x \ (t(x, gt(1, g)^{(\sigma)}) +_\sigma 1)
\]

(Above, \(g = (g_1, \ldots, g_n)\) are guarded)

**Unique Guarded Fixed-point Axioms**

\[
e^{(\sigma)} = ee^{(\sigma)} +_\sigma 1 \\
g = eg +_\sigma f \quad e \text{ guarded}
\]

\[
g = e^{(\sigma)} f
\]
An Axiomatization of Star Fragments modulo Bisimilarity?

Generalized Milner’s Completeness Problem:
Is this axiomatization of bisimulation complete for every star fragment?

Equational Branching Axioms

\[
T
\]

Sequencing Axioms

\[
\begin{align*}
ce &= c \\
1e &= e \\
e &= e1 \\
e(fg) &= (ef)g \\
(e +_\sigma f)g &= eg +_\sigma fg
\end{align*}
\]

General Unguarded Fixed-point Axiom

\[
t(1, \vec{g})^{(\sigma)} = \text{fp } x \left( t(x, \vec{g}t(1, \vec{g})^{(\sigma)}) +_\sigma 1 \right)
\]
(Above, $\vec{g} = (g_1, \ldots, g_n)$ are guarded)

Unique Guarded Fixed-point Axioms

\[
e^{(\sigma)} = ee^{(\sigma)} +_\sigma 1
\]
\[
\frac{g = eg +_\sigma f \quad e \text{ guarded}}{g = e^{(\sigma)}f}
\]
**Known & Unknown Completeness Theorems**

<table>
<thead>
<tr>
<th></th>
<th>Regex mod bisimilarity</th>
<th>GKAT mod bisimilarity</th>
<th>ProbRegex mod bisim.</th>
<th>ProbGKAT mod bisim.</th>
</tr>
</thead>
<tbody>
<tr>
<td>μ-exp</td>
<td>complete</td>
<td>complete</td>
<td>complete</td>
<td>complete</td>
</tr>
<tr>
<td>star fragment</td>
<td>complete (Grabmayer, 2022)</td>
<td>Unknown</td>
<td>Unknown</td>
<td>Unknown</td>
</tr>
<tr>
<td>1-free star fragment</td>
<td>complete (Grabmayer, Fokkink, 2019)</td>
<td>complete (Kappé, S., Silva, 2023)</td>
<td>complete (unpublished)</td>
<td>Unknown</td>
</tr>
<tr>
<td>recursion-free</td>
<td>complete</td>
<td>complete</td>
<td>complete</td>
<td>complete</td>
</tr>
</tbody>
</table>
Summary

- Star fragments arise from *branching theories*, \((S, T, \text{fp})\) consisting of an algebraic theory and a fixed-point operator that determines behaviour of unguarded fixed-points
- Milner’s regular expressions mod bisimilarity = *semilattices with bottom* star fragment
- GKAT/bisimilarity = **if-then-else** with **crash** star fragment
- Further examples:
  - (Rozowski, Kappé, Kozen, Schmid, Silva, 2023) ProbGKAT mod bisimilarity = GKAT + \(\bigoplus_p\)
  - Probabilistic regular expressions mod bisimilarity = \(\bigoplus_p\) instead of +
  - Regex mixing nondeterminism and probability = Regular expressions + \(\bigoplus_p\)

---

**Generalized Milner’s Completeness Problem:**
Is this axiomatization of bisimulation complete for every star fragment?

**Equational Branching Axioms**
- \(ce = c\)
- \(1e = e\)
- \(e = e1\)
- \(e(fg) = (ef)g\)
- \((e +_\sigma f)g = eg +_\sigma fg\)

**Sequencing Axioms**
- \(t(1, \bar{g})^{(\sigma)} = \text{fp } x \ (t(x, \bar{g}t(1, \bar{g})^{(\sigma)}) +_\sigma 1)\)
  (Above, \(\bar{z} = \{\bar{g}, \ldots, \bar{g}\}\) are guarded)
- \(e^{(\sigma)} = ee^{(\sigma)} +_\sigma 1\)
- \(g = eg +_\sigma f\) \(e\) guarded
- \(g = e^{(\sigma)}f\)

**Unique Guarded Fixed-point Axioms**