

Formal Modelling of Software Intensive Systems

Formal Semantics of Regular Expressions

Francesco Tiezzi

University of Camerino
`francesco.tiezzi@unicam.it`

A.A. 2015/2016



Formal semantics

Three main approaches to formal semantics of programming languages:

- **Operational Semantics** (*How a program computes*) [Plotkin, Kahn]:
Sets of **computations** resulting from the **execution** of programs by an abstract machine
- **Denotational Semantics** (*What a program computes*) [Strachey, Scott]:
An input/output **function** that denotes the **effect** of executing the program
- **Axiomatic Semantics** (*What a program modifies*) [Floyd, Hoare]:
Pairs of **observable properties** that hold before and after program execution

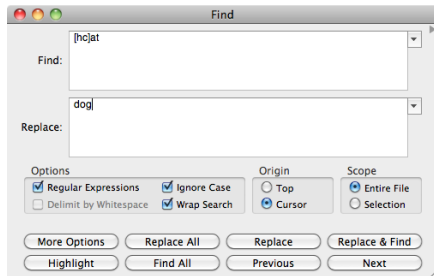
Different purposes, complementary use

A motivating example: regular expressions

Regular expressions

Commonly used for:

- searching and manipulating text based on patterns



Example

Regular expression: [hc]at \Rightarrow (h + c); a; t

Text: the cat eats the bat's hat rather than the rat

Matches: cat, hat

A motivating example: regular expressions

Regular expressions

Commonly used for:

- searching and manipulating text based on patterns
 - representing regular languages in a compact form
 - describing sequences of actions that a system can execute
-
- Regular expressions as a simple programming language
 - Programming constructs: sequence, choice, iteration, stop
 - We define the semantics of regular expressions by applying the three approaches
 - We show that the three semantics are consistent

A motivating example: regular expressions

Regular expressions

Commonly used for:

- searching and manipulating text based on patterns
 - representing regular languages in a compact form
 - describing sequences of actions that a system can execute
-
- Regular expressions as a simple programming language
 - Programming constructs: sequence, choice, iteration, stop
 - We define the semantics of regular expressions by applying the three approaches
 - We show that the three semantics are consistent

Regular expressions: syntax and informal semantics

Abstract syntax

$$E ::= 0 \mid 1 \mid a \mid E + E \mid E;E \mid E^*$$

Operators precedence

* binds more than + and ; ; binds more than +

Informal semantics

- 0 is the empty event
- 1 is the terminal event
- a is an event (or atomic action) where $a \in A$, with A finite alphabet
- $E + F$ can be either E or F (choice operator)
- $E;F$ is the expression E followed by F (sequencing)
- E^* is an n -length sequence of E with $n \geq 0$ (Kleene star)

Regular expressions: syntax and informal semantics

Abstract syntax

$$E ::= 0 \mid 1 \mid a \mid E + E \mid E;E \mid E^*$$

Operators precedence

* binds more than + and ; ; binds more than +

Informal semantics

- 0 is the empty event
- 1 is the terminal event
- a is an event (or atomic action) where $a \in A$, with A finite alphabet
- $E + F$ can be either E or F (choice operator)
- $E;F$ is the expression E followed by F (sequencing)
- E^* is an n -length sequence of E with $n \geq 0$ (Kleene star)

Regular expressions: syntax and informal semantics

Abstract syntax

$$E ::= 0 \mid 1 \mid a \mid E + E \mid E;E \mid E^*$$

Operators precedence

* binds more than + and ; ; binds more than +

Informal semantics

- 0 is the empty event
- 1 is the terminal event
- a is an event (or atomic action) where $a \in A$, with A finite alphabet
- $E + F$ can be either E or F (choice operator)
- $E;F$ is the expression E followed by F (sequencing)
- E^* is an n -length sequence of E with $n \geq 0$ (Kleene star)

Regular expressions: syntax and informal semantics

Abstract syntax

$$E ::= 0 \mid 1 \mid a \mid E + E \mid E;E \mid E^*$$

Operators precedence

* binds more than + and ; ; binds more than +

Informal semantics

- 0 is the empty event
- 1 is the terminal event
- a is an event (or atomic action) where $a \in A$, with A finite alphabet
- $E + F$ can be either E or F (choice operator)
- $E;F$ is the expression E followed by F (sequencing)
- E^* is an n -length sequence of E with $n \geq 0$ (Kleene star)

Regular expressions: syntax and informal semantics

Abstract syntax

$$E ::= 0 \mid 1 \mid a \mid E + E \mid E;E \mid E^*$$

Operators precedence

* binds more than + and ; ; binds more than +

Informal semantics

- 0 is the empty event
- 1 is the terminal event
- a is an event (or atomic action) where $a \in A$, with A finite alphabet
- $E + F$ can be either E or F (choice operator)
- $E;F$ is the expression E followed by F (sequencing)
- E^* is an n -length sequence of E with $n \geq 0$ (Kleene star)

Regular expressions: syntax and informal semantics

Abstract syntax

$$E ::= 0 \mid 1 \mid a \mid E + E \mid E;E \mid E^*$$

Operators precedence

* binds more than + and ; ; binds more than +

Informal semantics

- 0 is the empty event
- 1 is the terminal event
- a is an event (or atomic action) where $a \in A$, with A finite alphabet
- $E + F$ can be either E or F (choice operator)
- $E;F$ is the expression E followed by F (sequencing)
- E^* is an n -length sequence of E with $n \geq 0$ (Kleene star)

Regular expressions: syntax and informal semantics

Abstract syntax

$$E ::= 0 \mid 1 \mid a \mid E + E \mid E;E \mid E^*$$

Operators precedence

* binds more than + and ; ; binds more than +

Informal semantics

- 0 is the empty event
- 1 is the terminal event
- a is an event (or atomic action) where $a \in A$, with A finite alphabet
- $E + F$ can be either E or F (choice operator)
- $E;F$ is the expression E followed by F (sequencing)
- E^* is an n -length sequence of E with $n \geq 0$ (Kleene star)

Regular expressions: informal semantics

With an informal semantics the meaning of composite expressions may be not clear

Example

$$(a + b)^* \qquad (a^* + b^*)^*$$

- They are syntactically different
- What about their meaning?

We shall apply the three approaches used for defining formal semantics to regular expressions

Regular expressions: informal semantics

With an informal semantics the meaning of composite expressions may be not clear

Example

$$(a + b)^* \qquad (a^* + b^*)^*$$

- They are syntactically different
- What about their meaning?

We shall apply the three approaches used for defining formal semantics to regular expressions

Regular expressions: informal semantics

With an informal semantics the meaning of composite expressions may be not clear

Example

$$(a + b)^*$$

$$(a^* + b^*)^*$$

- They are syntactically different
- What about their meaning?

We shall apply the three approaches used for defining formal semantics to regular expressions

Regular expressions: informal semantics

With an informal semantics the meaning of composite expressions may be not clear

Example

$$(a + b)^*$$

$$(a^* + b^*)^*$$

- They are syntactically different
- What about their meaning?

We shall apply the three approaches used for defining formal semantics to regular expressions

Regular expressions: operational semantics

We introduce an **abstract machine** for **executing** regular expressions

Transition relation

- Is a ternary relation $E \xrightarrow{\mu} F$, where $\mu \in A \cup \{\varepsilon\}$ (ε empty action)
- Is defined by an inference system
- Describes, by induction on the structure of the expressions, the behaviour of a machine that takes as input a regular expression and executes it

For a generic operator op we shall have one or more rules like:

$$\frac{E_{i_1} \xrightarrow{\alpha_1} E'_{i_1} \cdots E_{i_m} \xrightarrow{\alpha_m} E'_{i_m}}{op(E_1, \dots, E_n) \xrightarrow{\alpha} op(E'_1, \dots, E'_n)} \quad \text{where } \{i_1, \dots, i_m\} \subseteq \{1, \dots, n\}.$$

Regular expressions: operational semantics

We introduce an **abstract machine** for **executing** regular expressions

Transition relation

- Is a ternary relation $E \xrightarrow{\mu} F$, where $\mu \in A \cup \{\varepsilon\}$ (ε empty action)
- Is defined by an inference system
- Describes, by induction on the structure of the expressions, the behaviour of a machine that takes as input a regular expression and executes it

For a generic operator op we shall have one or more rules like:

$$\frac{E_{i_1} \xrightarrow{\alpha_1} E'_{i_1} \quad \dots \quad E_{i_m} \xrightarrow{\alpha_m} E'_{i_m}}{op(E_1, \dots, E_n) \xrightarrow{\alpha} op(E'_1, \dots, E'_n)} \quad \text{where } \{i_1, \dots, i_m\} \subseteq \{1, \dots, n\}.$$

Regular expressions: operational semantics

We introduce an **abstract machine** for **executing** regular expressions

Transition relation

- Is a ternary relation $E \xrightarrow{\mu} F$, where $\mu \in A \cup \{\varepsilon\}$ (ε empty action)
- Is defined by an **inference system**
- Describes, by induction on the structure of the expressions, the behaviour of a machine that takes as input a regular expression and executes it

For a generic operator op we shall have one or more rules like:

$$\frac{E_{i_1} \xrightarrow{\alpha_1} E'_{i_1} \quad \dots \quad E_{i_m} \xrightarrow{\alpha_m} E'_{i_m}}{op(E_1, \dots, E_n) \xrightarrow{\alpha} op(E'_1, \dots, E'_n)} \quad \text{where } \{i_1, \dots, i_m\} \subseteq \{1, \dots, n\}.$$

Regular expressions: operational semantics

Transition relation rules

$$\text{(Tic)} \quad \frac{}{1 \xrightarrow{\varepsilon} 1}$$

$$\text{(Sum}_1\text{)} \quad \frac{E \xrightarrow{\mu} E'}{E + F \xrightarrow{\mu} E'}$$

$$\text{(Seq}_1\text{)} \quad \frac{E \xrightarrow{a} E'}{E; F \xrightarrow{a} E'; F}$$

$$\text{(Star}_1\text{)} \quad \frac{}{E^* \xrightarrow{\varepsilon} 1}$$

$$\text{(Atom)} \quad \frac{}{a \xrightarrow{a} 1} \quad a \in A$$

$$\text{(Sum}_2\text{)} \quad \frac{F \xrightarrow{\mu} F'}{E + F \xrightarrow{\mu} F'}$$

$$\text{(Seq}_2\text{)} \quad \frac{E \xrightarrow{\varepsilon} 1}{E; F \xrightarrow{\varepsilon} F}$$

$$\text{(Star}_2\text{)} \quad \frac{E \xrightarrow{\mu} E'}{E^* \xrightarrow{\mu} E'; E^*}$$

Structural Operational Semantics (SOS [Plotkin])

Transition relation is the least relation satisfying the above rules

Regular expressions: operational semantics

Transition relation rules

$$\text{(Tic)} \quad \frac{}{1 \xrightarrow{\varepsilon} 1}$$

$$\text{(Sum}_1\text{)} \quad \frac{E \xrightarrow{\mu} E'}{E + F \xrightarrow{\mu} E'}$$

$$\text{(Seq}_1\text{)} \quad \frac{E \xrightarrow{a} E'}{E; F \xrightarrow{a} E'; F}$$

$$\text{(Star}_1\text{)} \quad \frac{}{E^* \xrightarrow{\varepsilon} 1}$$

$$\text{(Atom)} \quad \frac{}{a \xrightarrow{a} 1} \quad a \in A$$

$$\text{(Sum}_2\text{)} \quad \frac{F \xrightarrow{\mu} F'}{E + F \xrightarrow{\mu} F'}$$

$$\text{(Seq}_2\text{)} \quad \frac{E \xrightarrow{\varepsilon} 1}{E; F \xrightarrow{\varepsilon} F}$$

$$\text{(Star}_2\text{)} \quad \frac{E \xrightarrow{\mu} E'}{E^* \xrightarrow{\mu} E'; E^*}$$

1 indicates the terminal state: the machine has completed the execution and loops by executing the empty action

Regular expressions: operational semantics

Transition relation rules

$$\text{(Tic)} \quad \frac{}{1 \xrightarrow{\varepsilon} 1}$$

$$\text{(Sum}_1\text{)} \quad \frac{E \xrightarrow{\mu} E'}{E + F \xrightarrow{\mu} E'}$$

$$\text{(Seq}_1\text{)} \quad \frac{E \xrightarrow{a} E'}{E; F \xrightarrow{a} E'; F}$$

$$\text{(Star}_1\text{)} \quad \frac{}{E^* \xrightarrow{\varepsilon} 1}$$

$$\text{(Atom)} \quad \frac{}{a \xrightarrow{a} 1} \quad a \in A$$

$$\text{(Sum}_2\text{)} \quad \frac{F \xrightarrow{\mu} F'}{E + F \xrightarrow{\mu} F'}$$

$$\text{(Seq}_2\text{)} \quad \frac{E \xrightarrow{\varepsilon} 1}{E; F \xrightarrow{\varepsilon} F}$$

$$\text{(Star}_2\text{)} \quad \frac{E \xrightarrow{\mu} E'}{E^* \xrightarrow{\mu} E'; E^*}$$

Expression a executes action a and stops

Regular expressions: operational semantics

Transition relation rules

$$\text{(Tic)} \quad \frac{}{1 \xrightarrow{\varepsilon} 1}$$

$$\text{(Atom)} \quad \frac{}{a \xrightarrow{a} 1} \quad a \in A$$

$$\text{(Sum}_1\text{)} \quad \frac{E \xrightarrow{\mu} E'}{E + F \xrightarrow{\mu} E'}$$

$$\text{(Sum}_2\text{)} \quad \frac{F \xrightarrow{\mu} F'}{E + F \xrightarrow{\mu} F'}$$

$$\text{(Seq}_1\text{)} \quad \frac{E \xrightarrow{a} E'}{E; F \xrightarrow{a} E'; F}$$

$$\text{(Seq}_2\text{)} \quad \frac{E \xrightarrow{\varepsilon} 1}{E; F \xrightarrow{\varepsilon} F}$$

$$\text{(Star}_1\text{)} \quad \frac{}{E^* \xrightarrow{\varepsilon} 1}$$

$$\text{(Star}_2\text{)} \quad \frac{E \xrightarrow{\mu} E'}{E^* \xrightarrow{\mu} E'; E^*}$$

$E + F$ can behave either as E or as F : if E evolves to E' by performing action μ then $E + F$ can evolve to E' by performing μ ; similarly for F

Regular expressions: operational semantics

Transition relation rules

$$\text{(Tic)} \quad \frac{}{1 \xrightarrow{\varepsilon} 1}$$

$$\text{(Sum}_1\text{)} \quad \frac{E \xrightarrow{\mu} E'}{E + F \xrightarrow{\mu} E'}$$

$$\text{(Seq}_1\text{)} \quad \frac{E \xrightarrow{a} E'}{E; F \xrightarrow{a} E'; F}$$

$$\text{(Star}_1\text{)} \quad \frac{}{E^* \xrightarrow{\varepsilon} 1}$$

$$\text{(Atom)} \quad \frac{}{a \xrightarrow{a} 1} \quad a \in A$$

$$\text{(Sum}_2\text{)} \quad \frac{F \xrightarrow{\mu} F'}{E + F \xrightarrow{\mu} F'}$$

$$\text{(Seq}_2\text{)} \quad \frac{E \xrightarrow{\varepsilon} 1}{E; F \xrightarrow{\varepsilon} F}$$

$$\text{(Star}_2\text{)} \quad \frac{E \xrightarrow{\mu} E'}{E^* \xrightarrow{\mu} E'; E^*}$$

$E; F$ executes the actions of E and, afterwards, the actions of F

Regular expressions: operational semantics

Transition relation rules

$$\text{(Tic)} \quad \frac{}{1 \xrightarrow{\varepsilon} 1}$$

$$\text{(Sum}_1\text{)} \quad \frac{E \xrightarrow{\mu} E'}{E + F \xrightarrow{\mu} E'}$$

$$\text{(Seq}_1\text{)} \quad \frac{E \xrightarrow{a} E'}{E; F \xrightarrow{a} E'; F}$$

$$\text{(Star}_1\text{)} \quad \frac{}{E^* \xrightarrow{\varepsilon} 1}$$

$$\text{(Atom)} \quad \frac{}{a \xrightarrow{a} 1} \quad a \in A$$

$$\text{(Sum}_2\text{)} \quad \frac{F \xrightarrow{\mu} F'}{E + F \xrightarrow{\mu} F'}$$

$$\text{(Seq}_2\text{)} \quad \frac{E \xrightarrow{\varepsilon} 1}{E; F \xrightarrow{\varepsilon} F}$$

$$\text{(Star}_2\text{)} \quad \frac{E \xrightarrow{\mu} E'}{E^* \xrightarrow{\mu} E'; E^*}$$

$E; F$ executes the actions of E and, afterwards, the actions of F

Regular expressions: operational semantics

Transition relation rules

$$\text{(Tic)} \quad \frac{}{1 \xrightarrow{\varepsilon} 1}$$

$$\text{(Sum}_1\text{)} \quad \frac{E \xrightarrow{\mu} E'}{E + F \xrightarrow{\mu} E'}$$

$$\text{(Seq}_1\text{)} \quad \frac{E \xrightarrow{a} E'}{E; F \xrightarrow{a} E'; F}$$

$$\text{(Star}_1\text{)} \quad \frac{}{E^* \xrightarrow{\varepsilon} 1}$$

$$\text{(Atom)} \quad \frac{}{a \xrightarrow{a} 1} \quad a \in A$$

$$\text{(Sum}_2\text{)} \quad \frac{F \xrightarrow{\mu} F'}{E + F \xrightarrow{\mu} F'}$$

$$\text{(Seq}_2\text{)} \quad \frac{E \xrightarrow{\varepsilon} 1}{E; F \xrightarrow{\varepsilon} F}$$

$$\text{(Star}_2\text{)} \quad \frac{E \xrightarrow{\mu} E'}{E^* \xrightarrow{\mu} E'; E^*}$$

$E; F$ executes the actions of E and, afterwards, the actions of F

Regular expressions: operational semantics

Transition relation rules

$$\text{(Tic)} \quad \frac{}{1 \xrightarrow{\varepsilon} 1}$$

$$\text{(Sum}_1\text{)} \quad \frac{E \xrightarrow{\mu} E'}{E + F \xrightarrow{\mu} E'}$$

$$\text{(Seq}_1\text{)} \quad \frac{E \xrightarrow{a} E'}{E; F \xrightarrow{a} E'; F}$$

$$\text{(Star}_1\text{)} \quad \frac{}{E^* \xrightarrow{\varepsilon} 1}$$

$$\text{(Atom)} \quad \frac{}{a \xrightarrow{a} 1} \quad a \in A$$

$$\text{(Sum}_2\text{)} \quad \frac{F \xrightarrow{\mu} F'}{E + F \xrightarrow{\mu} F'}$$

$$\text{(Seq}_2\text{)} \quad \frac{E \xrightarrow{\varepsilon} 1}{E; F \xrightarrow{\varepsilon} F}$$

$$\text{(Star}_2\text{)} \quad \frac{E \xrightarrow{\mu} E'}{E^* \xrightarrow{\mu} E'; E^*}$$

E^* can either directly evolve to 1 or evolve to $E'; E^*$ if E evolves to E'

Regular expressions: operational semantics

Transition relation rules

$$\text{(Tic)} \quad \frac{}{1 \xrightarrow{\varepsilon} 1}$$

$$\text{(Sum}_1\text{)} \quad \frac{E \xrightarrow{\mu} E'}{E + F \xrightarrow{\mu} E'}$$

$$\text{(Seq}_1\text{)} \quad \frac{E \xrightarrow{a} E'}{E; F \xrightarrow{a} E'; F}$$

$$\text{(Star}_1\text{)} \quad \frac{}{E^* \xrightarrow{\varepsilon} 1}$$

$$\text{(Atom)} \quad \frac{}{a \xrightarrow{a} 1} \quad a \in A$$

$$\text{(Sum}_2\text{)} \quad \frac{F \xrightarrow{\mu} F'}{E + F \xrightarrow{\mu} F'}$$

$$\text{(Seq}_2\text{)} \quad \frac{E \xrightarrow{\varepsilon} 1}{E; F \xrightarrow{\varepsilon} F}$$

$$\text{(Star}_2\text{)} \quad \frac{E \xrightarrow{\mu} E'}{E^* \xrightarrow{\mu} E'; E^*}$$

E^* can either directly evolve to 1 or evolve to $E'; E^*$ if E evolves to E'

Regular expressions: operational semantics

Transition relation rules

$$\text{(Tic)} \quad \frac{}{1 \xrightarrow{\varepsilon} 1}$$

$$\text{(Sum}_1\text{)} \quad \frac{E \xrightarrow{\mu} E'}{E + F \xrightarrow{\mu} E'}$$

$$\text{(Seq}_1\text{)} \quad \frac{E \xrightarrow{a} E'}{E; F \xrightarrow{a} E'; F}$$

$$\text{(Star}_1\text{)} \quad \frac{}{E^* \xrightarrow{\varepsilon} 1}$$

$$\text{(Atom)} \quad \frac{}{a \xrightarrow{a} 1} \quad a \in A$$

$$\text{(Sum}_2\text{)} \quad \frac{F \xrightarrow{\mu} F'}{E + F \xrightarrow{\mu} F'}$$

$$\text{(Seq}_2\text{)} \quad \frac{E \xrightarrow{\varepsilon} 1}{E; F \xrightarrow{\varepsilon} F}$$

$$\text{(Star}_2\text{)} \quad \frac{E \xrightarrow{\mu} E'}{E^* \xrightarrow{\mu} E'; E^*}$$

E^* can either directly evolve to 1 or evolve to $E'; E^*$ if E evolves to E'

Regular expressions: operational semantics

Transition relation rules

$$\text{(Tic)} \quad \frac{}{1 \xrightarrow{\varepsilon} 1}$$

$$\text{(Sum}_1\text{)} \quad \frac{E \xrightarrow{\mu} E'}{E + F \xrightarrow{\mu} E'}$$

$$\text{(Seq}_1\text{)} \quad \frac{E \xrightarrow{a} E'}{E; F \xrightarrow{a} E'; F}$$

$$\text{(Star}_1\text{)} \quad \frac{}{E^* \xrightarrow{\varepsilon} 1}$$

$$\text{(Atom)} \quad \frac{}{a \xrightarrow{a} 1} \quad a \in A$$

$$\text{(Sum}_2\text{)} \quad \frac{F \xrightarrow{\mu} F'}{E + F \xrightarrow{\mu} F'}$$

$$\text{(Seq}_2\text{)} \quad \frac{E \xrightarrow{\varepsilon} 1}{E; F \xrightarrow{\varepsilon} F}$$

$$\text{(Star}_2\text{)} \quad \frac{E \xrightarrow{\mu} E'}{E^* \xrightarrow{\mu} E'; E^*}$$

No rule for 0: expression 0 does nothing

0 indicates the **deadlock state**: the machine is stuck

The automaton associated to a regular expression

The SOS inference rules implicitly defines a particular automaton for each regular expression E (essentially a fragment of the whole LTS):

- the initial state is E (we shall often omit to mark it)
- the set of labels is A
- the set of states consists of all regular expressions that can be reached starting from E via a sequence of transitions
- the transition relation is the one induced from the SOS rules
- the only final state is 1 (we shall often omit to mark it)

Semantic correspondence

Given any regular expression E , the automaton generated by the SOS rules has the property of recognizing exactly the language $\mathcal{L}[[E]]$, but it is not the unique automaton satisfying such property.

Other "similar" automata might have less (or more) ε transitions.

A few examples for Regular Expressions

$(a + b)^* \xrightarrow{a} 1; (a + b)^*$

$$\frac{}{a \xrightarrow{a} 1} \text{ (Atom)}$$
$$\frac{}{a + b \xrightarrow{a} 1} \text{ (Sum}_1\text{)}$$
$$\frac{}{(a + b)^* \xrightarrow{a} 1; (a + b)^*} \text{ (Star}_2\text{)}$$

$1; (a + b)^* \xrightarrow{\epsilon} (a + b)^*$

$$\frac{}{1 \xrightarrow{\epsilon} 1} \text{ (Tic)}$$
$$\frac{}{1; (a + b)^* \xrightarrow{\epsilon} (a + b)^*} \text{ (Seq}_2\text{)}$$

A few examples for Regular Expressions

$$(a + b)^* \xrightarrow{a} 1; (a + b)^*$$

$$\frac{}{a \xrightarrow{a} 1} \text{ (Atom)}$$
$$\frac{a \xrightarrow{a} 1}{a + b \xrightarrow{a} 1} \text{ (Sum}_1\text{)}$$
$$\frac{a + b \xrightarrow{a} 1}{(a + b)^* \xrightarrow{a} 1; (a + b)^*} \text{ (Star}_2\text{)}$$

$$1; (a + b)^* \xrightarrow{\varepsilon} (a + b)^*$$

$$\frac{}{1 \xrightarrow{\varepsilon} 1} \text{ (Tic)}$$
$$\frac{1 \xrightarrow{\varepsilon} 1}{1; (a + b)^* \xrightarrow{\varepsilon} (a + b)^*} \text{ (Seq}_2\text{)}$$

Regular expressions: operational semantics

Definition (Traces of Regular expressions)

- Let E be a regular expression and $s \in A^*$ be a string, we write $E \xRightarrow{s} E'$ if there exists $\mu_1, \dots, \mu_n \in A \cup \{\varepsilon\}$ ($n \geq 0$) s.t.:
 - the string $\mu_1 \dots \mu_n$ coincides with s (up to some occurrence of ε)
 - $E \xrightarrow{\mu_1} E_1 \xrightarrow{\mu_2} E_2 \xrightarrow{\mu_3} \dots \xrightarrow{\mu_n} E_n \equiv E'$ (\equiv syntactical equiv.)
- The set of *traces* of E is the set of strings

$$\text{Traces}(E) = \{s \in A^* : E \xRightarrow{s} 1\}$$

Definition (Trace equivalence)

Two regular expressions E and F are *trace equivalent* if

$$\text{Traces}(E) = \text{Traces}(F)$$

Regular expressions: operational semantics

Definition (Traces of Regular expressions)

- Let E be a regular expression and $s \in A^*$ be a string, we write $E \xRightarrow{s} E'$ if there exists $\mu_1, \dots, \mu_n \in A \cup \{\varepsilon\}$ ($n \geq 0$) s.t.:
 - the string $\mu_1 \dots \mu_n$ coincides with s (up to some occurrence of ε)
 - $E \xrightarrow{\mu_1} E_1 \xrightarrow{\mu_2} E_2 \xrightarrow{\mu_3} \dots \xrightarrow{\mu_n} E_n \equiv E'$ (\equiv syntactical equiv.)
- The set of *traces* of E is the set of strings

$$\text{Traces}(E) = \{s \in A^* : E \xRightarrow{s} 1\}$$

Definition (Trace equivalence)

Two regular expressions E and F are *trace equivalent* if

$$\text{Traces}(E) = \text{Traces}(F)$$

Regular expressions: operational semantics

Example

$$(a + b)^* \qquad (a^* + b^*)^*$$

- They are syntactically different
- Are they semantically equivalent?

We have to show that:

- s is a trace of $(a + b)^*$ if and only if s is a trace of $(a^* + b^*)^*$

Regular expressions: operational semantics

Example

$$(a + b)^* \qquad (a^* + b^*)^*$$

- They are syntactically different
- $\text{Traces}((a + b)^*) \stackrel{?}{=} \text{Traces}((a^* + b^*)^*)$

We have to show that:

- s is a trace of $(a + b)^*$ if and only if s is a trace of $(a^* + b^*)^*$

Regular expressions: operational semantics

Example

$$(a + b)^* \qquad (a^* + b^*)^*$$

- They are syntactically different
- $\text{Traces}((a + b)^*) \stackrel{?}{=} \text{Traces}((a^* + b^*)^*)$

We have to show that:

- s is a trace of $(a + b)^*$ if and only if s is a trace of $(a^* + b^*)^*$

Regular expressions: operational semantics

if s is a trace of $(a + b)^*$ then s is a trace of $(a^* + b^*)^*$

Induction on the length of s .

- *Base step:* $|s| = 0$ (i.e., $s = \varepsilon$). Trivial: $(\text{Star}_1), (a^* + b^*)^* \xrightarrow{\varepsilon} 1$
- *Inductive step:* $|s| > 0$, then $s = as'$ or $s = bs'$; w.l.o.g. assume $s = as'$.
The only possible a -transition for $(a + b)^*$ is $(a + b)^* \xrightarrow{a} (a + b)^*$

This is proved via the following derivations:

$$\frac{\frac{\frac{}{a \xrightarrow{a} 1} (\text{Atom})}{a + b \xrightarrow{a} 1} (\text{Sum}_1)}{(a + b)^* \xrightarrow{a} 1; (a + b)^*} (\text{Star}_2) \quad \frac{\frac{}{1 \xrightarrow{\varepsilon} 1} (\text{Tr})}{1; (a + b)^* \xrightarrow{\varepsilon} (a + b)^*} (\text{Seq}_2)$$

Regular expressions: operational semantics

if s is a trace of $(a + b)^*$ then s is a trace of $(a^* + b^*)^*$

Induction on the length of s .

- *Base step:* $|s| = 0$ (i.e., $s = \varepsilon$). Trivial: $(\text{Star}_1), (a^* + b^*)^* \xrightarrow{\varepsilon} 1$
- *Inductive step:* $|s| > 0$, then $s = as'$ or $s = bs'$; w.l.o.g. assume $s = as'$.
The only possible a -transition for $(a + b)^*$ is $(a + b)^* \xrightarrow{a} (a + b)^*$

This is proved via the following derivations:

$$\frac{\frac{\frac{}{a \xrightarrow{a} 1} (\text{Atom})}{a + b \xrightarrow{a} 1} (\text{Sum}_1)}{(a + b)^* \xrightarrow{a} 1; (a + b)^*} (\text{Star}_2) \quad \frac{\frac{}{1 \xrightarrow{\varepsilon} 1} (\text{Tr})}{1; (a + b)^* \xrightarrow{\varepsilon} (a + b)^*} (\text{Seq}_2)$$

Regular expressions: operational semantics

if s is a trace of $(a + b)^*$ then s is a trace of $(a^* + b^*)^*$

Induction on the length of s .

- *Base step:* $|s| = 0$ (i.e., $s = \varepsilon$). Trivial: $(\text{Star}_1), (a^* + b^*)^* \xrightarrow{\varepsilon} 1$
- *Inductive step:* $|s| > 0$, then $s = as'$ or $s = bs'$; w.l.o.g. assume $s = as'$. The only possible a -transition for $(a + b)^*$ is $(a + b)^* \xrightarrow{a} (a + b)^*$

This is proved via the following derivations:

$$\frac{\frac{\frac{}{a \xrightarrow{a} 1} (\text{Atom})}{a + b \xrightarrow{a} 1} (\text{Sum}_1)}{(a + b)^* \xrightarrow{a} 1; (a + b)^*} (\text{Star}_2)$$

$$\frac{\frac{}{1 \xrightarrow{\varepsilon} 1} (\text{Tr})}{1; (a + b)^* \xrightarrow{\varepsilon} (a + b)^*} (\text{Seq}_2)$$

Regular expressions: operational semantics

if s is a trace of $(a + b)^*$ then s is a trace of $(a^* + b^*)^*$

Induction on the length of s .

- *Base step:* $|s| = 0$ (i.e., $s = \varepsilon$). Trivial: $(Star_1)$, $(a^* + b^*)^* \xrightarrow{\varepsilon} 1$
- *Inductive step:* $|s| > 0$, then $s = as'$ or $s = bs'$; w.l.o.g. assume $s = as'$. The only possible a -transition for $(a + b)^*$ is $(a + b)^* \xrightarrow{a} (a + b)^*$

This is proved via the following derivations:

$$\frac{\frac{\frac{}{a \xrightarrow{a} 1} (Atom)}{a + b \xrightarrow{a} 1} (Sum_1)}{(a + b)^* \xrightarrow{a} 1; (a + b)^*} (Star_2)$$

$$\frac{\frac{}{1 \xrightarrow{\varepsilon} 1} (Tic)}{1; (a + b)^* \xrightarrow{\varepsilon} (a + b)^*} (Seq_2)$$

Regular expressions: operational semantics

if s is a trace of $(a + b)^*$ then s is a trace of $(a^* + b^*)^*$

Induction on the length of s .

- *Base step:* $|s| = 0$ (i.e., $s = \varepsilon$). Trivial: (Star_1) , $(a^* + b^*)^* \xrightarrow{\varepsilon} 1$
- *Inductive step:* $|s| > 0$, then $s = as'$ or $s = bs'$; w.l.o.g. assume $s = as'$.

The only possible a -transition for $(a + b)^*$ is $(a + b)^* \xrightarrow{a} (a + b)^*$

By hypothesis, $(a + b)^* \xrightarrow{as'} 1$, thus $(a + b)^* \xrightarrow{s'} 1$.

By induction, we have $(a^* + b^*)^* \xrightarrow{s'} 1$, thus it is sufficient to prove $(a^* + b^*)^* \xrightarrow{a} (a^* + b^*)^*$ to conclude that $(a^* + b^*)^* \xrightarrow{s} 1$.

Regular expressions: operational semantics

if s is a trace of $(a + b)^*$ then s is a trace of $(a^* + b^*)^*$

Induction on the length of s .

- *Base step:* $|s| = 0$ (i.e., $s = \varepsilon$). Trivial: (Star_1) , $(a^* + b^*)^* \xrightarrow{\varepsilon} 1$
- *Inductive step:* $|s| > 0$, then $s = as'$ or $s = bs'$; w.l.o.g. assume $s = as'$.
The only possible a -transition for $(a + b)^*$ is $(a + b)^* \xrightarrow{a} (a + b)^*$
By hypothesis, $(a + b)^* \xrightarrow{as'} 1$, thus $(a + b)^* \xrightarrow{s'} 1$.
By induction, we have $(a^* + b^*)^* \xrightarrow{s'} 1$, thus it is sufficient to prove $(a^* + b^*)^* \xrightarrow{a} (a^* + b^*)^*$ to conclude that $(a^* + b^*)^* \xrightarrow{s} 1$.

Regular expressions: operational semantics

if s is a trace of $(a + b)^*$ then s is a trace of $(a^* + b^*)^*$

Induction on the length of s .

- *Base step:* $|s| = 0$ (i.e., $s = \varepsilon$). Trivial: (Star_1) , $(a^* + b^*)^* \xrightarrow{\varepsilon} 1$
- *Inductive step:* $|s| > 0$, then $s = as'$ or $s = bs'$; w.l.o.g. assume $s = as'$.
The only possible a -transition for $(a + b)^*$ is $(a + b)^* \xrightarrow{a} (a + b)^*$
By hypothesis, $(a + b)^* \xrightarrow{as'} 1$, thus $(a + b)^* \xrightarrow{s'} 1$.
By induction, we have $(a^* + b^*)^* \xrightarrow{s'} 1$, thus it is sufficient to prove $(a^* + b^*)^* \xrightarrow{a} (a^* + b^*)^*$ to conclude that $(a^* + b^*)^* \xrightarrow{s} 1$.

Regular expressions: operational semantics

if s is a trace of $(a + b)^*$ then s is a trace of $(a^* + b^*)^*$

Induction on the length of s .

- *Base step:* $|s| = 0$ (i.e., $s = \varepsilon$). Trivial: (Star_1) , $(a^* + b^*)^* \xrightarrow{\varepsilon} 1$
- *Inductive step:* $|s| > 0$, then $s = as'$ or $s = bs'$; w.l.o.g. assume $s = as'$.
 $(a^* + b^*)^* \xrightarrow{a} (a^* + b^*)^*$:

$$\begin{array}{c}
 \frac{}{a \xrightarrow{a} 1} \text{ (Atom)} \\
 \frac{}{a^* \xrightarrow{a} 1; a^*} \text{ (Star}_2\text{)} \\
 \frac{}{a^* + b^* \xrightarrow{a} 1; a^*} \text{ (Sum}_1\text{)} \\
 \frac{}{(a^* + b^*)^* \xrightarrow{a} 1; a^*; (a^* + b^*)^*} \text{ (Star}_2\text{)}
 \end{array}
 \qquad
 \begin{array}{c}
 \frac{}{1 \xrightarrow{\varepsilon} 1} \text{ (Tic)} \\
 \frac{}{1; a^*; (a^* + b^*)^* \xrightarrow{\varepsilon} a^*; (a^* + b^*)^*} \text{ (Seq}_2\text{)} \\
 \\
 \frac{}{a^* \xrightarrow{\varepsilon} 1} \text{ (Star}_1\text{)} \\
 \frac{}{a^*; (a^* + b^*)^* \xrightarrow{\varepsilon} (a^* + b^*)^*} \text{ (Seq}_2\text{)}
 \end{array}$$

Regular expressions: operational semantics

The abstract machine that describes the execution of a regular expression is a *finite state automaton*

Definition (Regular expressions as finite state automata)

Let E be a reg. expr., the *finite state automaton associated to E* is

$$M_E = (Q_E, A, \rightarrow_E, E, \{1\})$$

- *States*: $Q_E = \{F \mid \exists s \in A^*. E \xrightarrow{s} F\}$ (expressions from E)
- *Actions*: A (alphabet of E)
- *Transition relation*: \rightarrow_E s.t. $F \xrightarrow{\mu}_E F'$ if $F \xrightarrow{\mu} F'$ with $\mu \in A \cup \{\varepsilon\}$
- *Initial state*: expression E
- *Accepting states*: expression 1

Regular expressions: operational semantics

The abstract machine that describes the execution of a regular expression is a *finite state automaton*

Definition (Regular expressions as finite state automata)

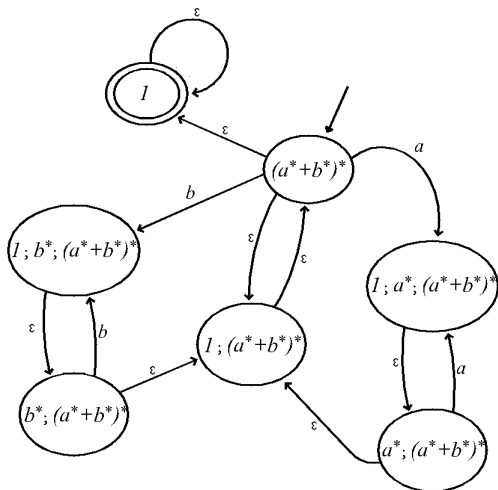
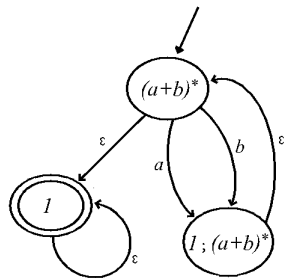
Let E be a reg. expr., the **finite state automaton associated to E** is

$$M_E = (Q_E, A, \rightarrow_E, E, \{1\})$$

- *States*: $Q_E = \{F \mid \exists s \in A^*. E \xrightarrow{s} F\}$ (expressions from E)
- *Actions*: A (alphabet of E)
- *Transition relation*: \rightarrow_E s.t. $F \xrightarrow{\mu}_E F'$ if $F \xrightarrow{\mu} F'$ with $\mu \in A \cup \{\varepsilon\}$
- *Initial state*: expression E
- *Accepting states*: expression 1

Regular expressions: operational semantics

Automata associated to $(a + b)^*$ and $(a^* + b^*)^*$



Regular expressions: operational semantics

Theorem

Let E be a regular expression and M_E the associated automaton, then

$$\text{Traces}(E) = L(M_E)$$

where $L(M_E) = \{s \in A^* : E \xrightarrow{s}_E 1\}$ (language accepted by M_E)

Proof (*sketch*). Two cases:

- \subseteq If $w \in \text{Traces}(E)$, then $E \xrightarrow{w}_E 1$. The proof that $w \in L(M_E)$ proceeds by induction on the length of w .
- \supseteq Given $w \in L(M_E)$, we prove by induction on the length of w that $w \in \text{Traces}(E)$.

Regular expressions: operational semantics

Theorem

Let E be a regular expression and M_E the associated automaton, then

$$\text{Traces}(E) = L(M_E)$$

where $L(M_E) = \{s \in A^* : E \xrightarrow{s}_E 1\}$ (language accepted by M_E)

Proof (*sketch*). Two cases:

- \subseteq If $w \in \text{Traces}(E)$, then $E \xrightarrow{w}_E 1$. The proof that $w \in L(M_E)$ proceeds by induction on the length of w .
- \supseteq Given $w \in L(M_E)$, we prove by induction on the length of w that $w \in \text{Traces}(E)$.

Regular expressions: denotational semantics

Denotational Semantics (*What a program computes*)

- an input/output **relation** that denotes the **effect** of executing the program
- semantic function
- associate to each program a mathematical object, called *denotation*, that represents its meaning

Operators on Languages

To define semantics interpretation function for regular expressions, we need some operators on languages. If L , L_1 and L_2 are sets of strings:

- $L_1 \cdot L_2 = \{xy : x \in L_1 \text{ and } y \in L_2\}$
- $L^* = \bigcup_{n \geq 0} L^n$ where
 - $L^0 = \{\varepsilon\}$
 - $L^{n+1} = L \cdot L^n$

We have: $\emptyset \cdot L = L \cdot \emptyset = \emptyset$ (Why?)

Regular expressions: denotational semantics

Semantic function \mathcal{L} for regular expressions

The denotational semantics is inductively defined by the rules below and associates a subset of A^* to each regular expressions:

$$\mathcal{L}[\] : R.E. \rightarrow 2^{A^*}$$

$$\mathcal{L}[0] = \emptyset$$

$$\mathcal{L}[1] = \{\varepsilon\}$$

$$\mathcal{L}[a] = \{a\} \quad (\text{for } a \in A)$$

$$\mathcal{L}[E + F] = \mathcal{L}[E] \cup \mathcal{L}[F]$$

$$\mathcal{L}[E ; F] = \mathcal{L}[E] \cdot \mathcal{L}[F]$$

$$\mathcal{L}[E^*] = (\mathcal{L}[E])^*$$

Regular expressions: denotational semantics

Example

$$(a + b)^*$$
$$(a^* + b^*)^*$$

- They are syntactically different
- Are they semantically equivalent?

We have to show that:

- $\mathcal{L}[(a + b)^*] \subseteq \mathcal{L}[(a^* + b^*)^*]$
- vice versa

Regular expressions: denotational semantics

Example

$$(a + b)^*$$
$$(a^* + b^*)^*$$

- They are syntactically different

- $\mathcal{L}[(a + b)^*] \stackrel{?}{=} \mathcal{L}[(a^* + b^*)^*]$

We have to show that:

- $\mathcal{L}[(a + b)^*] \subseteq \mathcal{L}[(a^* + b^*)^*]$
- vice versa

Regular expressions: denotational semantics

Example

$$(a + b)^* \qquad (a^* + b^*)^*$$

- They are syntactically different
- $\mathcal{L}[(a + b)^*] \stackrel{?}{=} \mathcal{L}[(a^* + b^*)^*]$

We have to show that:

- $\mathcal{L}[(a + b)^*] \subseteq \mathcal{L}[(a^* + b^*)^*]$
- vice versa

Regular expressions: denotational semantics

$$\mathcal{L}[(a + b)^*] \subseteq \mathcal{L}[(a^* + b^*)^*]$$

We have:

$$\begin{aligned}\mathcal{L}[(a + b)^*] &= (\mathcal{L}[(a + b)])^* \\ &= (\mathcal{L}[a] \cup \mathcal{L}[b])^* \\ &\subseteq (\mathcal{L}[a]^* \cup \mathcal{L}[b]^*)^* \\ &= (\mathcal{L}[a^*] \cup \mathcal{L}[b^*])^* \\ &= \mathcal{L}[a^* + b^*]^* \\ &= \mathcal{L}[(a^* + b^*)^*]\end{aligned}$$

Regular expressions: denotational semantics

$$\mathcal{L}[(a + b)^*] \subseteq \mathcal{L}[(a^* + b^*)^*]$$

We have:

$$\begin{aligned}\mathcal{L}[(a + b)^*] &= (\mathcal{L}[(a + b)])^* \\ &= (\mathcal{L}[a] \cup \mathcal{L}[b])^* \\ &\subseteq (\mathcal{L}[a]^* \cup \mathcal{L}[b]^*)^* \\ &= (\mathcal{L}[a^*] \cup \mathcal{L}[b^*])^* \\ &= \mathcal{L}[a^* + b^*]^* \\ &= \mathcal{L}[(a^* + b^*)^*]\end{aligned}$$

Regular expressions: denotational semantics

$$\mathcal{L}[(a^* + b^*)^*] \subseteq \mathcal{L}[(a + b)^*]$$

We have to prove:

$$(\mathcal{L}[a]^* \cup \mathcal{L}[b]^*)^* \subseteq (\mathcal{L}[a] \cup \mathcal{L}[b])^*$$

We exploit:

$$(\mathcal{L}[a] \cup \mathcal{L}[b])^* = ((\mathcal{L}[a] \cup \mathcal{L}[b])^*)^*$$

Thus, we have just to prove that:

$$(\mathcal{L}[a]^* \cup \mathcal{L}[b]^*)^* \subseteq ((\mathcal{L}[a] \cup \mathcal{L}[b])^*)^*$$

Let $s \in (\mathcal{L}[a]^* \cup \mathcal{L}[b]^*)^*$. Therefore, for some $n \geq 0$, we have $s = s_1 s_2 \cdots s_n$ and either $s_i \in \mathcal{L}[a]^*$ or $s_i \in \mathcal{L}[b]^*$, for all $0 \leq i \leq n$.

Thus, $s_i \in (\mathcal{L}[a] \cup \mathcal{L}[b])^*$, for all $0 \leq i \leq n$, hence $s \in ((\mathcal{L}[a] \cup \mathcal{L}[b])^*)^*$.

Regular expressions: denotational semantics

$$\mathcal{L}[(a^* + b^*)^*] \subseteq \mathcal{L}[(a + b)^*]$$

We have to prove:

$$(\mathcal{L}[a]^* \cup \mathcal{L}[b]^*)^* \subseteq (\mathcal{L}[a] \cup \mathcal{L}[b])^*$$

We exploit:

$$(\mathcal{L}[a] \cup \mathcal{L}[b])^* = ((\mathcal{L}[a] \cup \mathcal{L}[b])^*)^*$$

Thus, we have just to prove that:

$$(\mathcal{L}[a]^* \cup \mathcal{L}[b]^*)^* \subseteq ((\mathcal{L}[a] \cup \mathcal{L}[b])^*)^*$$

Let $s \in (\mathcal{L}[a]^* \cup \mathcal{L}[b]^*)^*$. Therefore, for some $n \geq 0$, we have $s = s_1 s_2 \cdots s_n$ and either $s_i \in \mathcal{L}[a]^*$ or $s_i \in \mathcal{L}[b]^*$, for all $0 \leq i \leq n$.

Thus, $s_i \in (\mathcal{L}[a] \cup \mathcal{L}[b])^*$, for all $0 \leq i \leq n$, hence $s \in ((\mathcal{L}[a] \cup \mathcal{L}[b])^*)^*$.

Regular expressions: denotational semantics

$$\mathcal{L}[(a^* + b^*)^*] \subseteq \mathcal{L}[(a + b)^*]$$

We have to prove:

$$(\mathcal{L}[a]^* \cup \mathcal{L}[b]^*)^* \subseteq (\mathcal{L}[a] \cup \mathcal{L}[b])^*$$

We exploit:

$$(\mathcal{L}[a] \cup \mathcal{L}[b])^* = ((\mathcal{L}[a] \cup \mathcal{L}[b])^*)^*$$

Thus, we have just to prove that:

$$(\mathcal{L}[a]^* \cup \mathcal{L}[b]^*)^* \subseteq ((\mathcal{L}[a] \cup \mathcal{L}[b])^*)^*$$

Let $s \in (\mathcal{L}[a]^* \cup \mathcal{L}[b]^*)^*$. Therefore, for some $n \geq 0$, we have $s = s_1 s_2 \cdots s_n$ and either $s_i \in \mathcal{L}[a]^*$ or $s_i \in \mathcal{L}[b]^*$, for all $0 \leq i \leq n$.

Thus, $s_i \in (\mathcal{L}[a] \cup \mathcal{L}[b])^*$, for all $0 \leq i \leq n$, hence $s \in ((\mathcal{L}[a] \cup \mathcal{L}[b])^*)^*$.

Regular expressions: denotational semantics

$$\mathcal{L}[(a^* + b^*)^*] \subseteq \mathcal{L}[(a + b)^*]$$

We have to prove:

$$(\mathcal{L}[a]^* \cup \mathcal{L}[b]^*)^* \subseteq (\mathcal{L}[a] \cup \mathcal{L}[b])^*$$

We exploit:

$$(\mathcal{L}[a] \cup \mathcal{L}[b])^* = ((\mathcal{L}[a] \cup \mathcal{L}[b])^*)^*$$

Thus, we have just to prove that:

$$(\mathcal{L}[a]^* \cup \mathcal{L}[b]^*)^* \subseteq ((\mathcal{L}[a] \cup \mathcal{L}[b])^*)^*$$

Let $s \in (\mathcal{L}[a]^* \cup \mathcal{L}[b]^*)^*$. Therefore, for some $n \geq 0$, we have $s = s_1 s_2 \cdots s_n$ and either $s_i \in \mathcal{L}[a]^*$ or $s_i \in \mathcal{L}[b]^*$, for all $0 \leq i \leq n$.

Thus, $s_i \in (\mathcal{L}[a] \cup \mathcal{L}[b])^*$, for all $0 \leq i \leq n$, hence $s \in ((\mathcal{L}[a] \cup \mathcal{L}[b])^*)^*$.

Regular expressions: denotational semantics

$$\mathcal{L}[(a^* + b^*)^*] \subseteq \mathcal{L}[(a + b)^*]$$

We have to prove:

$$(\mathcal{L}[a]^* \cup \mathcal{L}[b]^*)^* \subseteq (\mathcal{L}[a] \cup \mathcal{L}[b])^*$$

We exploit:

$$(\mathcal{L}[a] \cup \mathcal{L}[b])^* = ((\mathcal{L}[a] \cup \mathcal{L}[b])^*)^*$$

Thus, we have just to prove that:

$$(\mathcal{L}[a]^* \cup \mathcal{L}[b]^*)^* \subseteq ((\mathcal{L}[a] \cup \mathcal{L}[b])^*)^*$$

Let $s \in (\mathcal{L}[a]^* \cup \mathcal{L}[b]^*)^*$. Therefore, for some $n \geq 0$, we have $s = s_1 s_2 \cdots s_n$ and either $s_i \in \mathcal{L}[a]^*$ or $s_i \in \mathcal{L}[b]^*$, for all $0 \leq i \leq n$.

Thus, $s_i \in (\mathcal{L}[a] \cup \mathcal{L}[b])^*$, for all $0 \leq i \leq n$, hence $s \in ((\mathcal{L}[a] \cup \mathcal{L}[b])^*)^*$.

Equivalence result

Theorem (operational and denotational semantics are equivalent)

Let E be a regular expression, it holds that:

$$w \in \text{Traces}(E) \iff w \in \mathcal{L}[[E]]$$

Proof. Two cases:

\Rightarrow By induction on the structure of E .

\Leftarrow By induction on the structure of E .

Property

Let E and F regular expressions and s a string.

$$E; F \xrightarrow{s} 1 \text{ implies } \exists x, y \text{ s.t. } s = xy \text{ and } E \xrightarrow{x} 1, F \xrightarrow{y} 1$$

Equivalence result

Theorem (operational and denotational semantics are equivalent)

Let E be a regular expression, it holds that:

$$w \in \text{Traces}(E) \iff w \in \mathcal{L}[[E]]$$

Proof. Two cases:

\Rightarrow By induction on the structure of E .

\Leftarrow By induction on the structure of E .

Property

Let E and F regular expressions and s a string.

$$E; F \xRightarrow{s} 1 \text{ implies } \exists x, y \text{ s.t. } s = xy \text{ and } E \xRightarrow{x} 1, F \xRightarrow{y} 1$$

Equivalence result

Theorem (operational and denotational semantics are equivalent)

Let E be a regular expression, it holds that:

$$w \in \text{Traces}(E) \iff w \in \mathcal{L}[[E]]$$

Proof. Two cases:

\Rightarrow By induction on the structure of E .

\Leftarrow By induction on the structure of E .

Property

Let E and F regular expressions and s a string.

$$E; F \xrightarrow{s} 1 \text{ implies } \exists x, y \text{ s.t. } s = xy \text{ and } E \xrightarrow{x} 1, F \xrightarrow{y} 1$$

Regular expressions' semantics: equivalence result

Proof (\Rightarrow). By induction on the structure of E .

$E \equiv 0$ Trivial, because $\text{Traces}(0) = \emptyset = \mathcal{L}[[0]]$.

$E \equiv 1$ Trivial, because $\text{Traces}(1) = \{\varepsilon\} = \mathcal{L}[[1]]$.

$E \equiv a$ Trivial, because $\text{Traces}(a) = \{a\} = \mathcal{L}[[a]]$.

$E \equiv E_1 + E_2$ If $w \in \text{Traces}(E_1 + E_2)$, then $\exists \mu \in A \cup \{\varepsilon\}$ and $w' \in A^*$ with $w = \mu w'$ and

$$E_1 + E_2 \xrightarrow{\mu} F \xrightarrow{w'} 1$$

where

$$E_1 \xrightarrow{\mu} F \xrightarrow{w'} 1 \quad \text{or} \quad E_2 \xrightarrow{\mu} F \xrightarrow{w'} 1$$

By inductive hypothesis

$$w \in \mathcal{L}[[E_1]] \quad \text{or} \quad w \in \mathcal{L}[[E_2]]$$

Thus, $w \in \mathcal{L}[[E_1]] \cup \mathcal{L}[[E_2]] = \mathcal{L}[[E_1 + E_2]]$.

Equivalence result

$E \equiv E_1; E_2$ If $w \in \text{Traces}(E_1; E_2)$, by the previous property, $\exists x, y$ s.t.

$$E_1 \xrightarrow{x} 1 \quad \text{and} \quad E_2 \xrightarrow{y} 1$$

with $w = xy$. By inductive hypothesis, we have

$$x \in \mathcal{L}[[E_1]] \quad \text{and} \quad y \in \mathcal{L}[[E_2]],$$

and, hence, $w \in \mathcal{L}[[E_1]] \cdot \mathcal{L}[[E_2]] = \mathcal{L}[[E_1; E_2]]$.

$E \equiv E_1^*$ Let $S(E_1^*, w)$ be the number of application of $(Star_2)$ in $E_1^* \xrightarrow{w} 1$.

We demonstrate by induction on $n = S(E_1^*, w)$ that

$$w \in \mathcal{L}^n[[E_1]]. \quad (\mathcal{L}^n[[E_1]] \text{ stands for } (\mathcal{L}[[E_1]])^n)$$

...

Equivalence result

$E \equiv E_1^*$...

If $S(E_1^*, w) = 0$, no $(Star_2)$ but $(Star_1)$ used, thus $w = \varepsilon$.
By definition, $\varepsilon \in \mathcal{L}^0[[E_1]] = \{\varepsilon\}$.

If $S(E_1^*, w) = n + 1$, then $\exists x, y$ s.t. $w = xy$ and

$$E_1^* \xrightarrow{x} E_1^* \xrightarrow{y} E_1^* \xrightarrow{\varepsilon} 1$$

with $S(E_1^*, x) = n$.

By (local) induction hypothesis $x \in \mathcal{L}^n[[E_1]]$. Since $S(E_1^*, y) = 1$, $(Star_2)$ is applied only once in $E_1^* \xrightarrow{y} E_1^*$, thus $\exists \mu \in A \cup \{\varepsilon\}$ and $y' \in A^*$ s.t. $y = \mu y'$, $E_1 \xrightarrow{\mu} E'$ and

$$E_1^* \xrightarrow{\mu} E'; E_1^* \xrightarrow{y'} E_1^*.$$

Since $E'; E_1^* \xrightarrow{y'} E_1^*$ does not use $(Star_2)$, we have $E' \xrightarrow{y'} 1$ and, hence, $E_1 \xrightarrow{\mu y'} 1$. By (structural) inductive hypothesis, $y \in \mathcal{L}[[E_1]]$. Using $x \in \mathcal{L}^n[[E_1]]$, we conclude.

Equivalence result

Proof (\Leftarrow). By induction on the structure of E .

For the sake of simplicity, we only consider the case:

$E \equiv E_1^*$ If $w \in \mathcal{L}[[E_1^*]]$, then $\exists n$ s.t. $w \in \mathcal{L}^n[[E_1]]$.

Then, $\exists x_1, \dots, x_n \in \mathcal{L}[[E_1]]$ s.t. $w = x_1 \cdots x_n$.

By inductive hypothesis, $x_i \in \text{Traces}(E_1)$, that is $E_1 \xrightarrow{x_i} 1$.

By repeatedly applying (*Star*₂), we obtain $E_1^* \xrightarrow{x_i} 1; E_1^*$.

Since $1; E_1^* \xrightarrow{\varepsilon} E_1^*$, by (*Seq*₂), and $E_1^* \xrightarrow{\varepsilon} 1$, by (*Star*₁), we have

$$E_1^* \xrightarrow{x_1} 1; E_1^* \xrightarrow{x_2} 1; E_1^* \cdots \xrightarrow{x_n} 1; E_1^* \xrightarrow{\varepsilon} 1$$

and, therefore, $E_1^* \xrightarrow{w} 1$.

Regular expressions: axiomatic semantics

Axiomatic Semantics (*What a program modifies*)

- it relates **observable properties** before and after program execution
 - in stateful languages, e.g., if the initial state of a program fulfils the precondition and the program terminates, then the final state is guaranteed to fulfil the postcondition
- it consists of a set of axioms and inference rules that define a **relation**

Axiomatic semantics of regular expressions

- no state in regular expressions
- the observed property is the capability of equivalent expressions to represent the same regular language
- axioms and rules define an **equivalence relation** $E = F$ that partition the set of all expressions

Regular expressions: axiomatic semantics

Axiomatic Semantics (*What a program modifies*)

- it relates **observable properties** before and after program execution
 - in stateful languages, e.g., if the initial state of a program fulfils the precondition and the program terminates, then the final state is guaranteed to fulfil the postcondition
- it consists of a set of axioms and inference rules that define a **relation**

Axiomatic semantics of regular expressions

- no state in regular expressions
- the observed property is the capability of equivalent expressions to represent the same regular language
- axioms and rules define an **equivalence relation** $E = F$ that partition the set of all expressions

Regular expressions: axiomatic semantics

Axioms for $E = F$

$E + (F + G) = (E + F) + G$	(assoc +)	}	(monoid+)
$E + F = F + E$	(comm +)		
$E + 0 = E$	(unit +)		
$E ; (F ; G) = (E ; F) ; G$	(assoc ;)	}	(monoid ;)
$1 ; E = E$	(unit ;)		
$E ; (F + G) = E ; F + E ; G$	(distribL)	}	(modulo +, ;)
$(E + F) ; G = E ; G + F ; G$	(distribR)		
$0 ; E = 0$	(absorb 0)		
$E + E = E$		}	(idemp +)
$E^* = 1 + E^* ; E$	(unfolding)	}	(rules *)
$E^* = (1 + E)^*$	(absorb *)		
$0^* = 1$	(0 ⁰)		

Regular expressions: axiomatic semantics

Rules for $E = F$

Rule 1 (Substitution):

$$\frac{E = F \quad G = H}{G' = H \quad G' = G} \quad \text{where } G' \text{ is obtained from } G \text{ by replacing an occurrence of } E \text{ by } F$$

Rule 2 (Equation solution):

$$\frac{E = E; F + G}{E = G; F^*} \quad \text{if } F \text{ does not produce } \varepsilon$$

Regular expressions: axiomatic semantics

- The axioms are **sound** w.r.t. the observed property, i.e. = equates expressions representing the same language
 - E.g., given $0; E = 0$, we have:

$$\mathcal{L}[0; E] = \mathcal{L}[0] \cdot \mathcal{L}[E] = \emptyset \cdot \mathcal{L}[E] = \emptyset = \mathcal{L}[0]$$

- Applying the axiomatic approach could be more laborious
 - E.g., proving $E; 0 = 0$ requires the following inference:

$$\begin{array}{c}
 \frac{}{0 = 0; 0} \text{ (absorb 0)} \qquad \frac{}{E; 0 = E; 0} \\
 \hline
 \frac{}{E; 0; 0 = E; 0} \text{ (rule 1)} \qquad \frac{}{E; 0 + 0 = E; 0} \text{ (unit +)} \\
 \hline
 \frac{}{E; 0; 0 + 0 = E; 0} \text{ (rule 2)} \\
 \hline
 \frac{}{E; 0 = 0; 0^*} \text{ (rule 1)} \\
 \hline
 \frac{}{E; 0 = 0} \text{ (rule 1)}
 \end{array}$$

Regular expressions: axiomatic semantics

- The axioms are **sound** w.r.t. the observed property, i.e. = equates expressions representing the same language
 - E.g., given $0; E = 0$, we have:

$$\mathcal{L}[0; E] = \mathcal{L}[0] \cdot \mathcal{L}[E] = \emptyset \cdot \mathcal{L}[E] = \emptyset = \mathcal{L}[0]$$

- Applying the axiomatic approach could be more laborious
 - E.g., proving $E; 0 = 0$ requires the following inference:

$$\begin{array}{c}
 \frac{\frac{\frac{}{0 = 0; 0} \text{ (absorb 0)}}{E; 0; 0 = E; 0} \text{ (rule 1)} \quad \frac{\frac{}{E; 0 + 0 = E; 0} \text{ (unit +)}}{E; 0; 0 + 0 = E; 0} \text{ (rule 1)}}{E; 0; 0 + 0 = E; 0} \text{ (rule 2)}}{E; 0 = 0} \text{ (rule 1)} \\
 \frac{\frac{\frac{}{0; 0^* = 0} \text{ (absorb 0)}}{E; 0 = 0; 0^*} \text{ (rule 1)}}{E; 0 = 0} \text{ (rule 1)}
 \end{array}$$

Regular expressions' semantics: equivalence result

Theorem (axiomatic and denotational semantics are equivalent)

Let E and F be regular expressions, it holds that:

$$E = F \iff \mathcal{L}[[E]] = \mathcal{L}[[F]]$$

Proof (sketch). Two cases:

\Rightarrow (*Soundness*) Easy to prove

\Leftarrow (*Completeness*) Require a bit of work (e.g., expression normalization)

Corollary

The three semantics for regular expressions are equivalent

Regular expressions' semantics: equivalence result

Theorem (axiomatic and denotational semantics are equivalent)

Let E and F be regular expressions, it holds that:

$$E = F \iff \mathcal{L}\llbracket E \rrbracket = \mathcal{L}\llbracket F \rrbracket$$

Proof (sketch). Two cases:

\Rightarrow (*Soundness*) Easy to prove

\Leftarrow (*Completeness*) Require a bit of work (e.g., expression normalization)

Corollary

The three semantics for regular expressions are equivalent

Regular expressions' semantics: equivalence result

Theorem (axiomatic and denotational semantics are equivalent)

Let E and F be regular expressions, it holds that:

$$E = F \iff \mathcal{L}[[E]] = \mathcal{L}[[F]]$$

Proof (sketch). Two cases:

\Rightarrow (*Soundness*) Easy to prove

\Leftarrow (*Completeness*) Require a bit of work (e.g., expression normalization)

Corollary

The three semantics for regular expressions are equivalent