CS:4980 Topics in Computer Science II

Introduction to Automated Reasoning

Theory Solvers III

Cesare Tinelli

Spring 2024

Credits

These slides are based on slides originally developed by **Cesare Tinelli** at the University of Iowa, and by **Clark Barrett**, **Caroline Trippel**, and **Andrew (Haoze) Wu** at Stanford University. Adapted by permission.

Roadmap for Today

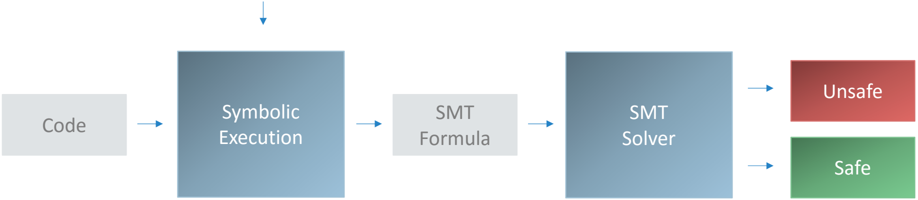
Theory Solvers

- Strings

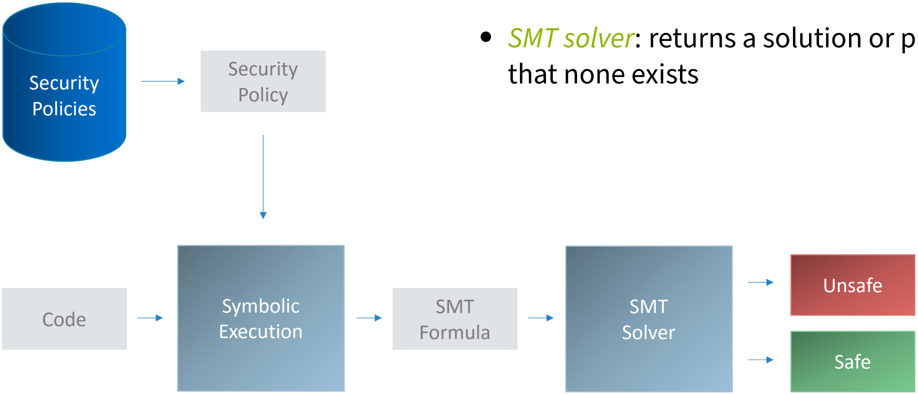
Motivation: Symbolic Execution

Symbolic Execution

- *Enumerate program paths* that end in a bad state
  - (e.g., invalid memory access)
- Represent program *inputs* as SMT *variables*
- Translate *statements* in the path into *constraints* on the variables
- Constraints represent *all possible executions* along the path
- Solving the constraints determines whether the path is *feasible*



Example: Symbolic Execution for Security



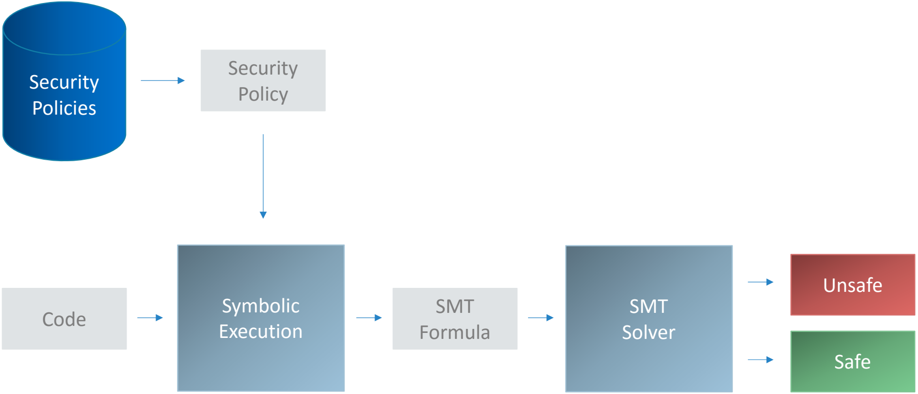
Security Vulnerabilities

- Input: *code* and *security policy*
- *Symbolic execution*: generates formula satisfiable iff code can violate security policy
- *SMT solver*: returns a solution or proves that none exists

String Analysis

Strings in Symbolic Execution

- Input code may manipulate *strings*



Basic Theory of Strings

Alphabet

$\mathcal{A}$ fixed *finite* set of characters

Constants

Empty string	$\epsilon : \mathsf{String}$	(i.e., $\mathrm{rank}(\epsilon) = \langle \mathsf{String} \rangle$)
Character string	$c : \mathsf{String}$ for all $c \in \mathcal{A}$	
Integer numeral	$n : \mathsf{Int}$ for all $n \geqslant 0$	

Operators

Concatenation	$\_ \cdot \_ : \mathsf{String} \times \mathsf{String} \to \mathsf{String}$	(i.e., $\mathrm{rank}(\cdot) = \langle \mathsf{String}, \mathsf{String}, \mathsf{String} \rangle$)
Length	$\lvert \_ \rvert : \mathsf{String} \to \mathsf{Int}$	
Membership	$\_ \in \_ : \mathsf{String} \times \mathsf{RegEx} \to \mathsf{Bool}$	
Addition	$\_ + \_ : \mathsf{Int} \times \mathsf{Int} \to \mathsf{Int}$	
Comparison	$\_ > \_ : \mathsf{Int} \times \mathsf{Int} \to \mathsf{Int}$	

Basic Theory of Strings

Alphabet

$\mathcal{A}$ fixed *finite* set of characters

Constants

Empty string	ϵ : String	(i.e., $\mathrm{rank}(\epsilon) = \langle \text{String} \rangle$)
Character string	c : String for all $c \in \mathcal{A}$	
Integer numeral	n : Int for all $n \geqslant 0$	

Operators

Concatenation	$\_ \cdot \_$: String × String → String	(i.e., $\mathrm{rank}(\cdot) = \langle \text{String}, \text{String}, \text{String} \rangle$)
Length	$\lvert \_ \rvert$: String → Int	
Membership	$\_ \in \_$: String × RegEx → Bool	
Addition	$\_ + \_$: Int × Int → Int	
Comparison	$\_ > \_$: Int × Int → Int	

Basic Theory of Strings

Alphabet

$\mathcal{A}$	fixed *finite* set of characters

Constants

Empty string	ϵ : String	(i.e., $\mathrm{rank}(\epsilon) = \langle$String$\rangle$)
Character string	c : String for all $c \in \mathcal{A}$	
Integer numeral	n : Int for all $n \geqslant 0$	

Operators

Concatenation	$\_ \cdot \_$: String × String → String	(i.e., $\mathrm{rank}(\cdot) = \langle$String, String, String$\rangle$)
Length	$\lvert \_ \rvert$: String → Int	
Membership	$\_ \in \_$: String × RegEx → Bool	
Addition	$\_ + \_$: Int × Int → Int	
Comparison	$\_ > \_$: Int × Int → Int	

Basic Theory of Strings

Alphabet

$\mathcal{A}$ fixed *finite* set of

Constants

Empty string	ϵ : St
Character string	c : St
Integer numeral	n : In

Challenge: complexity

concatenation + equality: *word equations problem*

- Decidable in PSPACE

+ length

- Decidability open

+ replace (all instances of some substring)

- Undecidable

Operators

Concatenation	$\_ \cdot \_ : \text{String} \times \text{String} \rightarrow \text{String}$ (i.e., $\mathrm{rank}(\cdot) = \langle \text{String}, \text{String}, \text{String} \rangle$)
Length	$\lvert \_ \rvert : \text{String} \rightarrow \text{Int}$
Membership	$\_ \in \_ : \text{String} \times \text{RegEx} \rightarrow \text{Bool}$
Addition	$\_ + \_ : \text{Int} \times \text{Int} \rightarrow \text{Int}$
Comparison	$\_ > \_ : \text{Int} \times \text{Int} \rightarrow \text{Int}$

Basic Theory of Strings

Alphabet

$\mathcal{A}$ fixed *finite* set of cha

Constants

Empty string	ϵ : Strin
Character string	c : Strin
Integer numeral	n : Int f

Pragmatic approach

- Rule-based proof system
- Use existing arithmetic theory solver
- Embrace incompleteness

Operators

Concatenation	$\_ \cdot \_ : \text{String} \times \text{String} \to \text{String}$ (i.e., $\mathrm{rank}(\cdot) = \langle \text{String}, \text{String}, \text{String} \rangle$)
Length	$\lvert \_ \rvert : \text{String} \to \text{Int}$
Membership	$\_ \in \_ : \text{String} \times \text{RegEx} \to \text{Bool}$
Addition	$\_ + \_ : \text{Int} \times \text{Int} \to \text{Int}$
Comparison	$\_ > \_ : \text{Int} \times \text{Int} \to \text{Int}$

Satisfiability Proof System for Strings

Proof States

A *proof state* is either:

- One of the distinguished states SAT, UNSAT
- A pair $\langle S; A \rangle$, where S contains *string* constraints and A contains *arithmetic* constraints

Assumptions

- All literals are flat
- For every string variable x in S, there exists a variable ℓ_x, such that $\ell_x = |x| \in S$
- Ignore regular expression membership for now

Notation

Definitions

- $\mathcal{T}(S)$ denotes all terms in S
- $S \models \alpha$ means that α follows from S using the rules of QF_UF
- $A \models_{LIA} \alpha$ means that α follows from A in the theory of linear integer arithmetic

Normalization function for length

- $|\epsilon|\downarrow = 0$
- $|c|\downarrow = 1$ for all $c \in \mathcal{A}$
- $|s_1 \cdot \cdots \cdot s_n|\downarrow = |s_1|\downarrow + \cdots + |s_n|\downarrow$

Core Rules

$$\textbf{A-CONF}\ \frac{A \models_{LIA} \bot}{\text{UNSAT}} \qquad \textbf{A-PROP}\ \frac{A \models_{LIA} s \doteq t \quad s, t \in \mathcal{T}(S)}{S := S, s \doteq t}$$

$$\textbf{S-CONF}\ \frac{S \models \bot}{\text{UNSAT}} \qquad \textbf{S-PROP}\ \frac{S \models s \doteq t \quad s, t \in \mathcal{T}(S) \quad s, t \text{ are } \Sigma_{LIA}\text{-terms}}{A := A, s \doteq t}$$

$$\textbf{S-A}\ \frac{x, y \in \mathcal{T}(S) \cap \mathcal{T}(A) \quad x, y : \text{Int}}{A := A, x \doteq y \qquad A := A, x \doteq y}$$

$$\textbf{L-INTRO}\ \frac{s \in \mathcal{T}(S) \quad s : \text{String}}{S := S, |s| \doteq |s|{\downarrow}} \qquad \textbf{L-VALID}\ \frac{x \in \mathcal{T}(S) \quad x : \text{String}}{S := S, x \doteq \epsilon \qquad A :\doteq A, \ell_x > 0}$$

$$\textbf{CONST-CONF}\ \frac{S \models c \doteq d \quad c \in \mathcal{A} \quad d \in \mathcal{A} \backslash \{c\}}{\text{UNSAT}}$$

$$\textbf{SAT}\ \frac{\text{no other rule applies}}{\text{SAT}}$$

Example Derivation

Let $S_0 = \{\, x \doteq y \cdot x \cdot z,\ y \doteq \text{“}a\text{”},\ \ell_x \doteq |x|,\ \ell_y \doteq |y|,\ \ell_z \doteq |z| \,\}$

$A_0 = \emptyset$

$$\begin{array}{c}
\langle S_0;\ A_0 \rangle \\ \hline
\langle |y \cdot x \cdot z| \doteq |y| + |x| + |z|;\ \emptyset \rangle \\ \hline
\langle |\text{“}a\text{”}| \doteq 1;\ \emptyset \rangle \\ \hline
\langle \emptyset;\ \ell_x \doteq \ell_y + \ell_x + \ell_z \rangle \\ \hline
\langle \emptyset;\ \ell_y \doteq 1 \rangle \\ \hline
\begin{array}{cc}
\begin{array}{c}
\langle z \doteq \epsilon;\ \emptyset \rangle \\ \hline
\langle |\epsilon| \doteq 0;\ \emptyset \rangle \\ \hline
\langle \emptyset;\ \ell_z \doteq 0 \rangle \\ \hline
\text{UNSAT}
\end{array}
&
\begin{array}{c}
\langle \emptyset;\ \ell_z > 0 \rangle \\ \hline
\text{UNSAT}
\end{array}
\end{array}
\end{array}$$

For each derivation step, we show only the difference between the derived state and the previous one

Example Derivation

Let $S_0 = \{\, x \doteq y \cdot x \cdot z,\ y \doteq \text{``}a\text{''},\ \ell_x \doteq |x|,\ \ell_y \doteq |y|,\ \ell_z \doteq |z| \,\}$

$A_0 = \emptyset$

$$\begin{array}{c}
\langle S_0;\ A_0 \rangle \\ \hline
\langle |y \cdot x \cdot z| \doteq |y| + |x| + |z|;\ \emptyset \rangle \\ \hline
\langle |\,\text{``}a\text{''}\,| \doteq 1;\ \emptyset \rangle \\ \hline
\langle \emptyset;\ \ell_x \doteq \ell_y + \ell_x + \ell_z \rangle \\ \hline
\langle \emptyset;\ \ell_y \doteq 1 \rangle \\ \hline
\begin{array}[t]{cc}
\begin{array}[t]{c}
\langle z \doteq \epsilon;\ \emptyset \rangle \\ \hline
\langle |\epsilon| \doteq 0;\ \emptyset \rangle \\ \hline
\langle \emptyset;\ \ell_z \doteq 0 \rangle \\ \hline
\text{UNSAT}
\end{array}
&
\begin{array}[t]{c}
\langle \emptyset;\ \ell_z > 0 \rangle \\ \hline
\text{UNSAT}
\end{array}
\end{array}
\end{array}$$

For each derivation step, we show only the difference between the derived state and the previous one

Concatenation Rules

If x is a variable of S, we can recursively expand x by substituting using equalities from S whose right-hand sides are concatenation terms until this is no longer possible

If t is the result, we write $S \models^{*}_{\cdot} x = t$

We write $\mathbf{z}$ as a short-hand for a concatenation of zero or more variables ($\mathbf{z} = z_1 \cdot z_2 \cdot \cdots \cdot z_n$, with $\mathbf{z} = \epsilon$ when $n = 0$)

$$\textbf{C-Eq}\quad \frac{S \models^{*}_{\cdot} x \doteq \mathbf{z} \qquad S \models^{*}_{\cdot} y \doteq \mathbf{z}}{S := S, x \doteq y}$$

$$\textbf{C-Split}\quad \frac{S \models^{*}_{\cdot} x \doteq \mathbf{w} \cdot u \cdot \mathbf{z} \qquad S \models^{*}_{\cdot} x \doteq \mathbf{w} \cdot v \cdot \mathbf{z}'}{\begin{array}{c} A := A, \ell_u > \ell_v;\ S := S, u \doteq v \cdot k \\ A := A, \ell_u < \ell_v;\ S := S, v \doteq u \cdot k \\ A := A, \ell_u \doteq \ell_v;\ S := S, u \doteq v \end{array}}$$

Note: k is a fresh variable

Concatenation Rules

If x is a variable of S, we can recursively expand x by substituting using equalities from S whose right-hand sides are concatenation terms until this is no longer possible

If t is the result, we write $S \models^{*}_{\cdot} x = t$

We write $\mathbf{z}$ as a short-hand for a concatenation of zero or more variables ($\mathbf{z} = z_1 \cdot z_2 \cdot \cdots \cdot z_n$, with $\mathbf{z} = \epsilon$ when $n = 0$)

$$\textbf{C-Eq} \quad \frac{S \models^{*}_{\cdot} x \doteq \mathbf{z} \qquad S \models^{*}_{\cdot} y \doteq \mathbf{z}}{S := S, x \doteq y}$$

$$\textbf{C-Split} \quad \frac{S \models^{*}_{\cdot} x \doteq \mathbf{w} \cdot u \cdot \mathbf{z} \qquad S \models^{*}_{\cdot} x \doteq \mathbf{w} \cdot v \cdot \mathbf{z}'}{\begin{array}{c} A := A, \ell_u > \ell_v;\ S := S, u \doteq v \cdot k \\ A := A, \ell_u < \ell_v;\ S := S, v \doteq u \cdot k \\ A := A, \ell_u \doteq \ell_v;\ S := S, u \doteq v \end{array}}$$

Note: k is a fresh variable

Concatenation Rules

If x is a variable of S, we can recursively expand x by substituting using equalities from S whose right-hand sides are concatenation terms until this is no longer possible

If t is the result, we write $S \models^{*}_{\cdot} x = t$

We write $\boldsymbol{z}$ as a short-hand for a concatenation of zero or more variables ($\boldsymbol{z} = z_1 \cdot z_2 \cdot \cdots \cdot z_n$, with $\boldsymbol{z} = \epsilon$ when $n = 0$)

$$\textbf{C-EQ}\ \frac{S \models^{*}_{\cdot} x \doteq \boldsymbol{z} \quad S \models^{*}_{\cdot} y \doteq \boldsymbol{z}}{S := S, x \doteq y}$$

$$\textbf{C-SPLIT}\ \frac{S \models^{*}_{\cdot} x \doteq \boldsymbol{w} \cdot u \cdot \boldsymbol{z} \quad S \models^{*}_{\cdot} x \doteq \boldsymbol{w} \cdot v \cdot \boldsymbol{z}'}{\begin{array}{c} A := A, \ell_u > \ell_v;\ S := S, u \doteq v \cdot k \\ A := A, \ell_u < \ell_v;\ S := S, v \doteq u \cdot k \\ A := A, \ell_u \doteq \ell_v;\ S := S, u \doteq v \end{array}}$$

Note: k is a fresh variable

Concatenation Rules

If x is a variable of S, we can recursively expand x by substituting using equalities from S whose right-hand sides are concatenation terms until this is no longer possible

If t is the result, we write $S \models^*_{\cdot} x = t$

We write $\boldsymbol{z}$ as a short-hand for a concatenation of zero or more variables ($\boldsymbol{z} = z_1 \cdot z_2 \cdot \; \cdots \; \cdot z_n$, with $\boldsymbol{z} = \epsilon$ when $n = 0$)

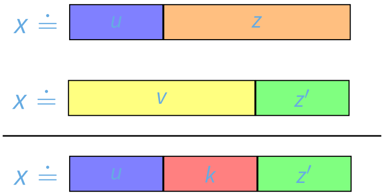
$$\textbf{C-EQ}\ \frac{S \models^*_{\cdot} x \doteq \boldsymbol{z} \quad S \models^*_{\cdot} y \doteq \boldsymbol{z}}{S := S, x \doteq y}$$

$$\textbf{C-SPLIT}\ \frac{S \models^*_{\cdot} x \doteq \boldsymbol{w} \cdot u \cdot \boldsymbol{z} \quad S \models^*_{\cdot} x \doteq \boldsymbol{w} \cdot v \cdot \boldsymbol{z}'}{\begin{array}{c} A := A, \ell_u > \ell_v;\ S := S, u \doteq v \cdot k \\ A := A, \ell_u < \ell_v;\ S := S, v \doteq u \cdot k \\ A := A, \ell_u \doteq \ell_v;\ S := S, u \doteq v \end{array}}$$

Note: k is a fresh variable

Example of C-Split

$$\textbf{C-SPLIT}\quad \frac{S \models^{*}_{.} x \doteq \boldsymbol{w}\cdot u\cdot \boldsymbol{z} \qquad S \models^{*}_{.} x \doteq \boldsymbol{w}\cdot v\cdot \boldsymbol{z}'}{\begin{array}{c} A := A, \ell_u > \ell_v;\ S := S, u \doteq v\cdot k \\ A := A, \ell_u < \ell_v;\ S := S, v \doteq u\cdot k \\ A := A, \ell_u \doteq \ell_v;\ S := S, u \doteq v \end{array}}$$



Core and Concat Rules

$$\textbf{A-C}\textsc{onf}\quad \frac{A \models_{LIA} \bot}{\text{UNSAT}} \qquad\qquad \textbf{A-P}\textsc{rop}\quad \frac{A \models_{LIA} s \doteq t \quad s, t \in \mathcal{T}(S)}{S := S, s \doteq t}$$

$$\textbf{S-C}\textsc{onf}\quad \frac{S \models \bot}{\text{UNSAT}} \qquad\qquad \textbf{S-P}\textsc{rop}\quad \frac{S \models s \doteq t \quad s, t \in \mathcal{T}(S) \quad s, t \text{ are } \Sigma_{LIA}\text{-terms}}{A := A, s \doteq t}$$

$$\textbf{C}\textsc{onst}\textbf{-C}\textsc{onf}\quad \frac{S \models c \doteq d \quad c \in \mathcal{A} \quad d \in \mathcal{A} \backslash \{c\}}{\text{UNSAT}} \qquad\qquad \textbf{S-A}\quad \frac{x, y \in \mathcal{T}(S) \cap \mathcal{T}(A) \quad x, y : \text{Int}}{A := A, x \doteq y \qquad A := A, x \doteq y}$$

$$\textbf{L-I}\textsc{ntro}\quad \frac{s \in \mathcal{T}(S) \quad s : \text{String}}{S := S, |s| \doteq |s|{\downarrow}} \qquad\qquad \textbf{L-V}\textsc{alid}\quad \frac{x \in \mathcal{T}(S) \quad x : \text{String}}{S := S, x \doteq \epsilon \qquad A :\doteq A, \ell_x > 0}$$

$$\textbf{S}\textsc{at}\quad \frac{\text{no other rule applies}}{\text{SAT}} \qquad\qquad \textbf{C-E}\textsc{q}\quad \frac{S \models^{*}_{\cdot} x \doteq \boldsymbol{z} \quad S \models^{*}_{\cdot} y \doteq \boldsymbol{z}}{S := S, x \doteq y}$$

$$\textbf{C-S}\textsc{plit}\quad \frac{S \models^{*}_{\cdot} x \doteq \boldsymbol{w} \cdot u \cdot \boldsymbol{z} \quad S \models^{*}_{\cdot} x \doteq \boldsymbol{w} \cdot v \cdot \boldsymbol{z}'}{\begin{array}{c} A := A, \ell_u > \ell_v;\ S := S, u \doteq v \cdot k \\ A := A, \ell_u < \ell_v;\ S := S, v \doteq u \cdot k \\ A := A, \ell_u \doteq \ell_v;\ S := S, u \doteq v \end{array}}$$

Properties of the proof system

Is the proof system sound? terminating?

Properties of the proof system

The proof system is

- refutation sound
    - easily checkable by examining each proof rule
- solution sound
    - proving this is highly non-trivial
- not terminating
    - for pathological unsat cases, **C-SPLIT** can be applied infinitely often
- incomplete
    - a consequence of non-termination

Properties of the proof system

The proof system is

- refutation sound
  - easily checkable by examining each proof rule
- solution sound
  - proving this is highly non-trivial
- not terminating
  - for pathological unsat cases, **C-SPLIT** can be applied infinitely often
- incomplete
  - a consequence of non-termination

Properties of the proof system

The proof system is

- refutation sound
    - easily checkable by examining each proof rule
- solution sound
    - proving this is highly non-trivial
- not terminating
    - for pathological unsat cases, **C-Split** can be applied infinitely often
- incomplete
    - a consequence of non-termination

Properties of the proof system

The proof system is

- refutation sound
  - easily checkable by examining each proof rule
- solution sound
  - proving this is highly non-trivial
- not terminating
  - for pathological unsat cases, **C-Split** can be applied infinitely often
- incomplete
  - a consequence of non-termination

Properties of the proof system

The proof system is

- refutation sound
  - easily checkable by examining each proof rule
- solution sound
  - proving this is highly non-trivial
- not terminating
  - for pathological unsat cases, **C-SPLIT** can be applied infinitely often
- incomplete
  - a consequence of non-termination

Iterating to Improve the Solver: More String Operators

1. Extend the theory by adding *new operators*
    - substr(x, n, m) : String, the maximal substring of x, starting at position n, with length $\leqslant m$
    - contains(x, y) : Bool, true iff x contains y as a substring
    - index_of(x, y, n) : Int, position of the first occurrence of y in x, starting from position n
    - replace(x, y, z) : String, the result of replacing the first occurrence of x in y by z

2. Implement them by reduction to the core theory

Iterating to Improve the Solver: More String Operators

1. Extend the theory by adding *new operators*
    - $\mathsf{substr}(x, n, m) : \mathsf{String}$, the maximal substring of x, starting at position n, with length $\leqslant m$
    - $\mathsf{contains}(x, y) : \mathsf{Bool}$, true iff x contains y as a substring
    - $\mathsf{index\_of}(x, y, n) : \mathsf{Int}$, position of the first occurrence of y in x, starting from position n
    - $\mathsf{replace}(x, y, z) : \mathsf{String}$, the result of replacing the first occurrence of x in y by z
2. Implement them by reduction to the core theory

Iterating to Improve the Solver: More String Operators

1. Extend the theory by adding *new operators*
    - $\mathsf{substr}(x, n, m) : \mathsf{String}$, the maximal substring of x, starting at position n, with length $\leqslant m$
    - $\mathsf{contains}(x, y) : \mathsf{Bool}$, true iff x contains y as a substring
    - $\mathsf{index\_of}(x, y, n) : \mathsf{Int}$, position of the first occurrence of y in x, starting from position n
    - $\mathsf{replace}(x, y, z) : \mathsf{String}$, the result of replacing the first occurrence of x in y by z
2. Implement them by reduction to the core theory

New Operators as Macros

$$x \mathbin{\dot{-}} y \quad \equiv \quad \max(x - y, 0)$$

$$\begin{aligned} x \doteq \mathrm{substr}(y, n, m) \quad \equiv \quad & \mathtt{ite}(\ 0 \leqslant n < |y| \wedge 0 < m, \\ & \quad y \doteq z_1 \cdot x \cdot z_2 \wedge |z_1| \doteq n \wedge |z_2| \doteq |y| \mathbin{\dot{-}} (m + n), \\ & \quad x \doteq \epsilon\ ) \end{aligned}$$

$$\mathrm{contains}(y, z) \quad \equiv \quad \exists\, k.\ 0 \leqslant k \leqslant |y| - |z| \wedge \mathrm{substr}(y, k, |z|) \doteq z$$

$$\begin{aligned} x \doteq \mathrm{index\_of}(y, z, n) \quad \equiv \quad & \mathtt{ite}(\ 0 \leqslant n \leqslant |y| \wedge \mathrm{contains}(y', z), \\ & \quad \mathrm{substr}(y', x', |z|) \doteq z \wedge \neg\mathrm{contains}(\mathrm{substr}(y', 0, x' + |z| - 1), z), \\ & \quad x \doteq -1\ ) \\ & \text{with } y' \doteq \mathrm{substr}(y, n, |y| - n) \text{ and } x' \doteq x - n \end{aligned}$$

$$\begin{aligned} x \doteq \mathrm{replace}(y, z, w) \quad \equiv \quad & \mathtt{ite}(\ \mathrm{contains}(y, z) \wedge z \doteq \epsilon, \\ & \quad x \doteq z_1 \cdot w \cdot z_2 \wedge y \doteq z_1 \cdot z \cdot z_2 \wedge \mathrm{index\_of}(y, z, 0) \doteq |z_1|, \\ & \quad x \doteq y\ ) \end{aligned}$$

New Operators as Macros

$$x \mathbin{\dot{-}} y \quad \equiv \quad \max(x - y, 0)$$

$$\begin{aligned} x \doteq \mathrm{substr}(y, n, m) \quad \equiv \quad & \mathtt{ite}(\, 0 \leqslant n < |y| \wedge 0 < m, \\ & \quad y \doteq z_1 \cdot x \cdot z_2 \wedge |z_1| \doteq n \wedge |z_2| \doteq |y| \mathbin{\dot{-}} (m + n), \\ & \quad x \doteq \epsilon \,) \end{aligned}$$

$$\mathrm{contains}(y, z) \quad \equiv \quad \exists\, k.\, 0 \leqslant k \leqslant |y| - |z| \wedge \mathrm{substr}(y, k, |z|) \doteq z$$

$$\begin{aligned} x \doteq \mathrm{index\_of}(y, z, n) \quad \equiv \quad & \mathtt{ite}(\, 0 \leqslant n \leqslant |y| \wedge \mathrm{contains}(y', z), \\ & \quad \mathrm{substr}(y', x', |z|) \doteq z \wedge \neg\mathrm{contains}(\mathrm{substr}(y', 0, x' + |z| - 1), z), \\ & \quad x \doteq -1 \,) \\ & \text{with } y' \doteq \mathrm{substr}(y, n, |y| - n) \text{ and } x' \doteq x - n \end{aligned}$$

$$\begin{aligned} x \doteq \mathrm{replace}(y, z, w) \quad \equiv \quad & \mathtt{ite}(\, \mathrm{contains}(y, z) \wedge z \doteq \epsilon, \\ & \quad x \doteq z_1 \cdot w \cdot z_2 \wedge y \doteq z_1 \cdot z \cdot z_2 \wedge \mathrm{index\_of}(y, z, 0) \doteq |z_1|, \\ & \quad x \doteq y \,) \end{aligned}$$

New Operators as Macros

$$
\begin{aligned}
x \dot{-} y \quad &\equiv \quad \max(x - y, 0) \\[2ex]
x \doteq \mathrm{substr}(y, n, m) \quad &\equiv \quad \texttt{ite}(\, 0 \leqslant n < |y| \wedge 0 < m, \\
&\qquad\qquad y \doteq z_1 \cdot x \cdot z_2 \wedge |z_1| \doteq n \wedge |z_2| \doteq |y| \dot{-} (m + n), \\
&\qquad\qquad x \doteq \epsilon \,) \\[2ex]
\mathrm{contains}(y, z) \quad &\equiv \quad \exists\, k.\, 0 \leqslant k \leqslant |y| - |z| \wedge \mathrm{substr}(y, k, |z|) \doteq z \\[2ex]
x \doteq \mathrm{index\_of}(y, z, n) \quad &\equiv \quad \texttt{ite}(\, 0 \leqslant n \leqslant |y| \wedge \mathrm{contains}(y', z), \\
&\qquad\qquad \mathrm{substr}(y', x', |z|) \doteq z \wedge \neg\mathrm{contains}(\mathrm{substr}(y', 0, x' + |z| - 1), z), \\
&\qquad\qquad x \doteq -1 \,) \\
&\quad\ \text{with } y' \doteq \mathrm{substr}(y, n, |y| - n) \text{ and } x' \doteq x - n \\[2ex]
x \doteq \mathrm{replace}(y, z, w) \quad &\equiv \quad \texttt{ite}(\, \mathrm{contains}(y, z) \wedge z \doteq \epsilon, \\
&\qquad\qquad x \doteq z_1 \cdot w \cdot z_2 \wedge y \doteq z_1 \cdot z \cdot z_2 \wedge \mathrm{index\_of}(y, z, 0) \doteq |z_1|, \\
&\qquad\qquad x \doteq y \,)
\end{aligned}
$$

New Operators as Macros

$$x \dot{-} y \quad \equiv \quad \max(x - y, 0)$$

$$\begin{aligned} x \doteq \mathrm{substr}(y, n, m) \quad \equiv \quad \mathtt{ite}(\ & 0 \leqslant n < |y| \wedge 0 < m, \\ & y \doteq z_1 \cdot x \cdot z_2 \wedge |z_1| \doteq n \wedge |z_2| \doteq |y| \dot{-} (m + n), \\ & x \doteq \epsilon\ ) \end{aligned}$$

$$\mathrm{contains}(y, z) \quad \equiv \quad \exists\, k.\, 0 \leqslant k \leqslant |y| - |z| \wedge \mathrm{substr}(y, k, |z|) \doteq z$$

$$\begin{aligned} x \doteq \mathrm{index\_of}(y, z, n) \quad \equiv \quad \mathtt{ite}(\ & 0 \leqslant n \leqslant |y| \wedge \mathrm{contains}(y', z), \\ & \mathrm{substr}(y', x', |z|) \doteq z \wedge \neg \mathrm{contains}(\mathrm{substr}(y', 0, x' + |z| - 1), z), \\ & x \doteq -1\ ) \end{aligned}$$

with $y' \doteq \mathrm{substr}(y, n, |y| - n)$ and $x' \doteq x - n$

$$\begin{aligned} x \doteq \mathrm{replace}(y, z, w) \quad \equiv \quad \mathtt{ite}(\ & \mathrm{contains}(y, z) \wedge z \doteq \epsilon, \\ & x \doteq z_1 \cdot w \cdot z_2 \wedge y \doteq z_1 \cdot z \cdot z_2 \wedge \mathrm{index\_of}(y, z, 0) \doteq |z_1|, \\ & x \doteq y\ ) \end{aligned}$$

New Operators as Macros

$$x \mathbin{\dot{-}} y \quad\equiv\quad \max(x - y, 0)$$

$$\begin{aligned}
x \doteq \mathrm{substr}(y, n, m) \quad\equiv\quad & \mathtt{ite}(\, 0 \leqslant n < |y| \wedge 0 < m, \\
& \qquad y \doteq z_1 \cdot x \cdot z_2 \wedge |z_1| \doteq n \wedge |z_2| \doteq |y| \mathbin{\dot{-}} (m + n), \\
& \qquad x \doteq \epsilon \,)
\end{aligned}$$

$$\mathrm{contains}(y, z) \quad\equiv\quad \exists\, k.\, 0 \leqslant k \leqslant |y| - |z| \wedge \mathrm{substr}(y, k, |z|) \doteq z$$

$$\begin{aligned}
x \doteq \mathrm{index\_of}(y, z, n) \quad\equiv\quad & \mathtt{ite}(\, 0 \leqslant n \leqslant |y| \wedge \mathrm{contains}(y', z), \\
& \qquad \mathrm{substr}(y', x', |z|) \doteq z \wedge \neg\mathrm{contains}(\mathrm{substr}(y', 0, x' + |z| - 1), z), \\
& \qquad x \doteq -1 \,)
\end{aligned}$$

with $y' \doteq \mathrm{substr}(y, n, |y| - n)$ and $x' \doteq x - n$

$$\begin{aligned}
x \doteq \mathrm{replace}(y, z, w) \quad\equiv\quad & \mathtt{ite}(\, \mathrm{contains}(y, z) \wedge z \doteq \epsilon, \\
& \qquad x \doteq z_1 \cdot w \cdot z_2 \wedge y \doteq z_1 \cdot z \cdot z_2 \wedge \mathrm{index\_of}(y, z, 0) \doteq |z_1|, \\
& \qquad x \doteq y \,)
\end{aligned}$$

Reasoning about New Operators: Performance

Iterate and Improve

- Extend the implementation to reason directly on the new operators
- How?
  - Keep formulas with original new operators
  - Periodically try to simplify them based on new knowledge

Reasoning about New Operators: Performance

Iterate and Improve

- Extend the implementation to reason directly on the new operators
- How?
  - Keep formulas with original new operators
  - Periodically try to simplify them based on new knowledge

Reasoning about New Operators: Performance

Iterate and Improve

- Extend the implementation to reason directly on the new operators
- How?
  - Keep formulas with original new operators
  - Periodically try to simplify them based on new knowledge

Reasoning about New Operators: Performance

Iterate and Improve

- Extend the implementation to reason directly on the new operators
- How?
  - Keep formulas with original new operators
  - Periodically try to simplify them based on new knowledge

Simplification rules for New Operators

Example: contains (l_1, l_2 denote string constants)

$$
\begin{array}{rcll}
\text{contains}(l_1, l_2) & \longrightarrow & \top & \text{if } l_1 \text{ contains } l_2 \\
\text{contains}(l_1, l_2) & \longrightarrow & \bot & \text{if } l_1 \text{ does not contain } l_2 \\
\text{contains}(l_1, l_2 \cdot \boldsymbol{t}) & \longrightarrow & \bot & \text{if } l_1 \text{ does not contain } l_2 \\
\text{contains}(l_1, l_2 \cdot \boldsymbol{t}) & \longrightarrow & \bot & \text{if contains}(l_1 \backslash l_2, \boldsymbol{t}) \longrightarrow^{*} \bot \\
\text{contains}(l_1, x \cdot \boldsymbol{t}) & \longrightarrow & \bot & \text{if contains}(l_1, \boldsymbol{t}) \longrightarrow^{*} \bot \\
\text{contains}(l_1 \cdot \boldsymbol{t}, l_2) & \longrightarrow & \top & \text{if } l_1 \text{ contains } l_2 \\
\text{contains}(x \cdot \boldsymbol{t}, s) & \longrightarrow & \top & \text{if contains}(\boldsymbol{t}, s) \longrightarrow^{*} \top \\
\text{contains}(t \cdot \boldsymbol{s}, t \cdot \boldsymbol{u}) & \longrightarrow & \top & \text{if contains}(\boldsymbol{s}, \boldsymbol{u}) \longrightarrow^{*} \top \\
\text{contains}(l_1 \cdot \boldsymbol{t}, l_2) & \longrightarrow & \text{contains}(\boldsymbol{t}, l_2) & \text{if } l_1 \sqcup_l l_2 = \epsilon \\
\text{contains}(\boldsymbol{t} \cdot l_1, l_2) & \longrightarrow & \text{contains}(\boldsymbol{t}, l_2) & \text{if } l_1 \sqcup_r l_2 = \epsilon
\end{array}
$$

$$\text{contains}(\epsilon, t) = \top \quad \longrightarrow \quad \epsilon = t$$

$$\ldots$$

Simplification rules for New Operators

Example: contains (l_1, l_2 denote string constants)

$$
\begin{aligned}
\text{contains}(l_1, l_2) &\longrightarrow \top && \text{if } l_1 \text{ contains } l_2 \\
\text{contains}(l_1, l_2) &\longrightarrow \bot && \text{if } l_1 \text{ does not contain } l_2 \\
\text{contains}(l_1, l_2 \cdot \boldsymbol{t}) &\longrightarrow \bot && \text{if } l_1 \text{ does not contain } l_2 \\
\text{contains}(l_1, l_2 \cdot \boldsymbol{t}) &\longrightarrow \bot && \text{if contains}(l_1 \backslash l_2, \boldsymbol{t}) \longrightarrow^* \bot \\
\text{contains}(l_1, x \cdot \boldsymbol{t}) &\longrightarrow \bot && \text{if contains}(l_1, \boldsymbol{t}) \longrightarrow^* \bot \\
\text{contains}(l_1 \cdot \boldsymbol{t}, l_2) &\longrightarrow \top && \text{if } l_1 \text{ contains } l_2 \\
\text{contains}(x \cdot \boldsymbol{t}, s) &\longrightarrow \top && \text{if contains}(\boldsymbol{t}, s) \longrightarrow^* \top \\
\text{contains}(t \cdot \boldsymbol{s}, t \cdot \boldsymbol{u}) &\longrightarrow \top && \text{if contains}(\boldsymbol{s}, \boldsymbol{u}) \longrightarrow^* \top \\
\text{contains}(l_1 \cdot \boldsymbol{t}, l_2) &\longrightarrow \text{contains}(\boldsymbol{t}, l_2) && \text{if } l_1 \sqcup_l l_2 = \epsilon \\
\text{contains}(\boldsymbol{t} \cdot l_1, l_2) &\longrightarrow \text{contains}(\boldsymbol{t}, l_2) && \text{if } l_1 \sqcup_r l_2 = \epsilon
\end{aligned}
$$

$$\text{contains}(\epsilon, t) = \top \longrightarrow \epsilon = t$$

$$\cdots$$

Simplification rules for New Operators

Example: contains (l_1, l_2 denote string constants)

$$
\begin{array}{rcll}
\text{contains}(l_1, l_2) & \longrightarrow & \top & \text{if } l_1 \text{ contains } l_2 \\
\text{contains}(l_1, l_2) & \longrightarrow & \bot & \text{if } l_1 \text{ does not contain } l_2 \\
\text{contains}(l_1, l_2 \cdot \boldsymbol{t}) & \longrightarrow & \bot & \text{if } l_1 \text{ does not contain } l_2 \\
\text{contains}(l_1, l_2 \cdot \boldsymbol{t}) & \longrightarrow & \bot & \text{if contains}(l_1 \setminus l_2, \boldsymbol{t}) \longrightarrow^* \bot \\
\text{contains}(l_1, x \cdot \boldsymbol{t}) & \longrightarrow & \bot & \text{if contains}(l_1, \boldsymbol{t}) \longrightarrow^* \bot \\
\text{contains}(l_1 \cdot \boldsymbol{t}, l_2) & \longrightarrow & \top & \text{if } l_1 \text{ contains } l_2 \\
\text{contains}(x \cdot \boldsymbol{t}, s) & \longrightarrow & \top & \text{if contains}(\boldsymbol{t}, s) \longrightarrow^* \top \\
\text{contains}(t \cdot \boldsymbol{s}, t \cdot \boldsymbol{u}) & \longrightarrow & \top & \text{if contains}(\boldsymbol{s}, \boldsymbol{u}) \longrightarrow^* \top \\
\text{contains}(l_1 \cdot \boldsymbol{t}, l_2) & \longrightarrow & \text{contains}(\boldsymbol{t}, l_2) & \text{if } l_1 \sqcup_l l_2 = \epsilon \\
\text{contains}(\boldsymbol{t} \cdot l_1, l_2) & \longrightarrow & \text{contains}(\boldsymbol{t}, l_2) & \text{if } l_1 \sqcup_r l_2 = \epsilon
\end{array}
$$

$$\text{contains}(\epsilon, t) = \top \quad \longrightarrow \quad \epsilon = t$$

$$\cdots$$

Simplification rules for New Operators

Example: contains (l_1, l_2 denote string constants)

$$
\begin{array}{rcll}
\text{contains}(l_1, l_2) & \longrightarrow & \top & \text{if } l_1 \text{ contains } l_2 \\
\text{contains}(l_1, l_2) & \longrightarrow & \bot & \text{if } l_1 \text{ does not contain } l_2 \\
\text{contains}(l_1, l_2 \cdot \boldsymbol{t}) & \longrightarrow & \bot & \text{if } l_1 \text{ does not contain } l_2 \\
\text{contains}(l_1, l_2 \cdot \boldsymbol{t}) & \longrightarrow & \bot & \text{if } \text{contains}(l_1 \setminus l_2, \boldsymbol{t}) \longrightarrow^* \bot \\
\text{contains}(l_1, x \cdot \boldsymbol{t}) & \longrightarrow & \bot & \text{if } \text{contains}(l_1, \boldsymbol{t}) \longrightarrow^* \bot \\
\text{contains}(l_1 \cdot \boldsymbol{t}, l_2) & \longrightarrow & \top & \text{if } l_1 \text{ contains } l_2 \\
\text{contains}(x \cdot \boldsymbol{t}, s) & \longrightarrow & \top & \text{if } \text{contains}(\boldsymbol{t}, s) \longrightarrow^* \top \\
\text{contains}(t \cdot \boldsymbol{s}, t \cdot \boldsymbol{u}) & \longrightarrow & \top & \text{if } \text{contains}(\boldsymbol{s}, \boldsymbol{u}) \longrightarrow^* \top \\
\text{contains}(l_1 \cdot \boldsymbol{t}, l_2) & \longrightarrow & \text{contains}(\boldsymbol{t}, l_2) & \text{if } l_1 \sqcup_l l_2 = \epsilon \\
\text{contains}(\boldsymbol{t} \cdot l_1, l_2) & \longrightarrow & \text{contains}(\boldsymbol{t}, l_2) & \text{if } l_1 \sqcup_r l_2 = \epsilon
\end{array}
$$

$$\text{contains}(\epsilon, t) = \top \longrightarrow \epsilon = t$$

$$\cdots$$

Simplification rules for New Operators

Example: contains (l_1, l_2 denote string constants)

$$
\begin{array}{rcll}
\text{contains}(l_1, l_2) & \longrightarrow & \top & \text{if } l_1 \text{ contains } l_2 \\
\text{contains}(l_1, l_2) & \longrightarrow & \bot & \text{if } l_1 \text{ does not contain } l_2 \\
\text{contains}(l_1, l_2 \cdot \boldsymbol{t}) & \longrightarrow & \bot & \text{if } l_1 \text{ does not contain } l_2 \\
\text{contains}(l_1, l_2 \cdot \boldsymbol{t}) & \longrightarrow & \bot & \text{if contains}(l_1 \backslash l_2, \boldsymbol{t}) \longrightarrow^* \bot \\
\text{contains}(l_1, x \cdot \boldsymbol{t}) & \longrightarrow & \bot & \text{if contains}(l_1, \boldsymbol{t}) \longrightarrow^* \bot \\
\text{contains}(l_1 \cdot \boldsymbol{t}, l_2) & \longrightarrow & \top & \text{if } l_1 \text{ contains } l_2 \\
\text{contains}(x \cdot \boldsymbol{t}, s) & \longrightarrow & \top & \text{if contains}(\boldsymbol{t}, s) \longrightarrow^* \top \\
\text{contains}(t \cdot \boldsymbol{s}, t \cdot \boldsymbol{u}) & \longrightarrow & \top & \text{if contains}(\boldsymbol{s}, \boldsymbol{u}) \longrightarrow^* \top \\
\text{contains}(l_1 \cdot \boldsymbol{t}, l_2) & \longrightarrow & \text{contains}(\boldsymbol{t}, l_2) & \text{if } l_1 \sqcup_l l_2 = \epsilon \\
\text{contains}(\boldsymbol{t} \cdot l_1, l_2) & \longrightarrow & \text{contains}(\boldsymbol{t}, l_2) & \text{if } l_1 \sqcup_r l_2 = \epsilon
\end{array}
$$

$$\text{contains}(\epsilon, t) = \top \quad \longrightarrow \quad \epsilon = t$$

$$\cdots$$

Simplification rules for New Operators

Example: contains (l_1, l_2 denote string constants)

$$
\begin{array}{rcll}
\text{contains}(l_1, l_2) & \longrightarrow & \top & \text{if } l_1 \text{ contains } l_2 \\
\text{contains}(l_1, l_2) & \longrightarrow & \bot & \text{if } l_1 \text{ does not contain } l_2 \\
\text{contains}(l_1, l_2 \cdot \boldsymbol{t}) & \longrightarrow & \bot & \text{if } l_1 \text{ does not contain } l_2 \\
\text{contains}(l_1, l_2 \cdot \boldsymbol{t}) & \longrightarrow & \bot & \text{if contains}(l_1 \backslash l_2, \boldsymbol{t}) \longrightarrow^* \bot \\
\text{contains}(l_1, x \cdot \boldsymbol{t}) & \longrightarrow & \bot & \text{if contains}(l_1, \boldsymbol{t}) \longrightarrow^* \bot \\
\text{contains}(l_1 \cdot \boldsymbol{t}, l_2) & \longrightarrow & \top & \text{if } l_1 \text{ contains } l_2 \\
\text{contains}(x \cdot \boldsymbol{t}, s) & \longrightarrow & \top & \text{if contains}(\boldsymbol{t}, s) \longrightarrow^* \top \\
\text{contains}(t \cdot \boldsymbol{s}, t \cdot \boldsymbol{u}) & \longrightarrow & \top & \text{if contains}(\boldsymbol{s}, \boldsymbol{u}) \longrightarrow^* \top \\
\text{contains}(l_1 \cdot \boldsymbol{t}, l_2) & \longrightarrow & \text{contains}(\boldsymbol{t}, l_2) & \text{if } l_1 \sqcup_l l_2 = \epsilon \\
\text{contains}(\boldsymbol{t} \cdot l_1, l_2) & \longrightarrow & \text{contains}(\boldsymbol{t}, l_2) & \text{if } l_1 \sqcup_r l_2 = \epsilon
\end{array}
$$

$$
\text{contains}(\epsilon, t) = \top \quad \longrightarrow \quad \epsilon = t
$$

$$
\cdots
$$

Simplification rules for New Operators

Example: contains (${l_1}$, l_2 denote string constants)

$$\begin{aligned}
\text{contains}(l_1, l_2) &\longrightarrow \top && \text{if } l_1 \text{ contains } l_2 \\
\text{contains}(l_1, l_2) &\longrightarrow \bot && \text{if } l_1 \text{ does not contain } l_2 \\
\text{contains}(l_1, l_2 \cdot \boldsymbol{t}) &\longrightarrow \bot && \text{if } l_1 \text{ does not contain } l_2 \\
\text{contains}(l_1, l_2 \cdot \boldsymbol{t}) &\longrightarrow \bot && \text{if contains}(l_1 \backslash l_2, \boldsymbol{t}) \longrightarrow^* \bot \\
\text{contains}(l_1, x \cdot \boldsymbol{t}) &\longrightarrow \bot && \text{if contains}(l_1, \boldsymbol{t}) \longrightarrow^* \bot \\
\text{contains}(l_1 \cdot \boldsymbol{t}, l_2) &\longrightarrow \top && \text{if } l_1 \text{ contains } l_2 \\
\text{contains}(x \cdot \boldsymbol{t}, s) &\longrightarrow \top && \text{if contains}(\boldsymbol{t}, s) \longrightarrow^* \top \\
\text{contains}(t \cdot \boldsymbol{s}, t \cdot \boldsymbol{u}) &\longrightarrow \top && \text{if contains}(\boldsymbol{s}, \boldsymbol{u}) \longrightarrow^* \top \\
\text{contains}(l_1 \cdot \boldsymbol{t}, l_2) &\longrightarrow \text{contains}(\boldsymbol{t}, l_2) && \text{if } l_1 \sqcup_l l_2 = \epsilon \\
\text{contains}(\boldsymbol{t} \cdot l_1, l_2) &\longrightarrow \text{contains}(\boldsymbol{t}, l_2) && \text{if } l_1 \sqcup_r l_2 = \epsilon
\end{aligned}$$

$$\text{contains}(\epsilon, t) = \top \longrightarrow \epsilon = t$$

$$\cdots$$

Simplification rules for New Operators

Example: contains $\qquad$ (l_1, l_2 denote string constants)

$$
\begin{array}{rcll}
\mathrm{contains}(l_1, l_2) & \longrightarrow & \top & \text{if } l_1 \text{ contains } l_2 \\
\mathrm{contains}(l_1, l_2) & \longrightarrow & \bot & \text{if } l_1 \text{ does not contain } l_2 \\
\mathrm{contains}(l_1, l_2 \cdot \boldsymbol{t}) & \longrightarrow & \bot & \text{if } l_1 \text{ does not contain } l_2 \\
\mathrm{contains}(l_1, l_2 \cdot \boldsymbol{t}) & \longrightarrow & \bot & \text{if } \mathrm{contains}(l_1 \backslash l_2, \boldsymbol{t}) \longrightarrow^* \bot \\
\mathrm{contains}(l_1, x \cdot \boldsymbol{t}) & \longrightarrow & \bot & \text{if } \mathrm{contains}(l_1, \boldsymbol{t}) \longrightarrow^* \bot \\
\mathrm{contains}(l_1 \cdot \boldsymbol{t}, l_2) & \longrightarrow & \top & \text{if } l_1 \text{ contains } l_2 \\
\mathrm{contains}(x \cdot \boldsymbol{t}, s) & \longrightarrow & \top & \text{if } \mathrm{contains}(\boldsymbol{t}, s) \longrightarrow^* \top \\
\mathrm{contains}(t \cdot \boldsymbol{s}, t \cdot \boldsymbol{u}) & \longrightarrow & \top & \text{if } \mathrm{contains}(\boldsymbol{s}, \boldsymbol{u}) \longrightarrow^* \top \\
\mathrm{contains}(l_1 \cdot \boldsymbol{t}, l_2) & \longrightarrow & \mathrm{contains}(\boldsymbol{t}, l_2) & \text{if } l_1 \sqcup_l l_2 = \epsilon \\
\mathrm{contains}(\boldsymbol{t} \cdot l_1, l_2) & \longrightarrow & \mathrm{contains}(\boldsymbol{t}, l_2) & \text{if } l_1 \sqcup_r l_2 = \epsilon
\end{array}
$$

$$\mathrm{contains}(\epsilon, t) = \top \quad \longrightarrow \quad \epsilon = t$$

…

Simplification rules for New Operators

Example: contains (l_1, l_2 denote string constants)

$$
\begin{array}{rcll}
\text{contains}(l_1, l_2) & \longrightarrow & \top & \text{if } l_1 \text{ contains } l_2 \\
\text{contains}(l_1, l_2) & \longrightarrow & \bot & \text{if } l_1 \text{ does not contain } l_2 \\
\text{contains}(l_1, l_2 \cdot \boldsymbol{t}) & \longrightarrow & \bot & \text{if } l_1 \text{ does not contain } l_2 \\
\text{contains}(l_1, l_2 \cdot \boldsymbol{t}) & \longrightarrow & \bot & \text{if contains}(l_1 \backslash l_2, \boldsymbol{t}) \longrightarrow^* \bot \\
\text{contains}(l_1, x \cdot \boldsymbol{t}) & \longrightarrow & \bot & \text{if contains}(l_1, \boldsymbol{t}) \longrightarrow^* \bot \\
\text{contains}(l_1 \cdot \boldsymbol{t}, l_2) & \longrightarrow & \top & \text{if } l_1 \text{ contains } l_2 \\
\text{contains}(x \cdot \boldsymbol{t}, s) & \longrightarrow & \top & \text{if contains}(\boldsymbol{t}, s) \longrightarrow^* \top \\
\text{contains}(t \cdot \boldsymbol{s}, t \cdot \boldsymbol{u}) & \longrightarrow & \top & \text{if contains}(\boldsymbol{s}, \boldsymbol{u}) \longrightarrow^* \top \\
\text{contains}(l_1 \cdot \boldsymbol{t}, l_2) & \longrightarrow & \text{contains}(\boldsymbol{t}, l_2) & \text{if } l_1 \sqcup_l l_2 = \epsilon \\
\text{contains}(\boldsymbol{t} \cdot l_1, l_2) & \longrightarrow & \text{contains}(\boldsymbol{t}, l_2) & \text{if } l_1 \sqcup_r l_2 = \epsilon
\end{array}
$$

$$\text{contains}(\epsilon, t) = \top \longrightarrow \epsilon = t$$

$$\cdots$$

Simplification rules for New Operators

Example: contains (l_1, l_2 denote string constants)

$$\begin{array}{rcll}
\text{contains}(l_1, l_2) & \longrightarrow & \top & \text{if } l_1 \text{ contains } l_2 \\
\text{contains}(l_1, l_2) & \longrightarrow & \bot & \text{if } l_1 \text{ does not contain } l_2 \\
\text{contains}(l_1, l_2 \cdot \boldsymbol{t}) & \longrightarrow & \bot & \text{if } l_1 \text{ does not contain } l_2 \\
\text{contains}(l_1, l_2 \cdot \boldsymbol{t}) & \longrightarrow & \bot & \text{if contains}(l_1 \backslash l_2, \boldsymbol{t}) \longrightarrow^{*} \bot \\
\text{contains}(l_1, x \cdot \boldsymbol{t}) & \longrightarrow & \bot & \text{if contains}(l_1, \boldsymbol{t}) \longrightarrow^{*} \bot \\
\text{contains}(l_1 \cdot \boldsymbol{t}, l_2) & \longrightarrow & \top & \text{if } l_1 \text{ contains } l_2 \\
\text{contains}(x \cdot \boldsymbol{t}, s) & \longrightarrow & \top & \text{if contains}(\boldsymbol{t}, s) \longrightarrow^{*} \top \\
\text{contains}(t \cdot \boldsymbol{s}, t \cdot \boldsymbol{u}) & \longrightarrow & \top & \text{if contains}(\boldsymbol{s}, \boldsymbol{u}) \longrightarrow^{*} \top \\
\text{contains}(l_1 \cdot \boldsymbol{t}, l_2) & \longrightarrow & \text{contains}(\boldsymbol{t}, l_2) & \text{if } l_1 \sqcup_l l_2 = \epsilon \\
\text{contains}(\boldsymbol{t} \cdot l_1, l_2) & \longrightarrow & \text{contains}(\boldsymbol{t}, l_2) & \text{if } l_1 \sqcup_r l_2 = \epsilon
\end{array}$$

$$\text{contains}(\epsilon, t) = \top \quad \longrightarrow \quad \epsilon = t$$

$$\ldots$$

Simplification rules for New Operators

Example: contains (l_1, l_2 denote string constants)

$$
\begin{array}{rcll}
\text{contains}(l_1, l_2) & \longrightarrow & \top & \text{if } l_1 \text{ contains } l_2 \\
\text{contains}(l_1, l_2) & \longrightarrow & \bot & \text{if } l_1 \text{ does not contain } l_2 \\
\text{contains}(l_1, l_2 \cdot \boldsymbol{t}) & \longrightarrow & \bot & \text{if } l_1 \text{ does not contain } l_2 \\
\text{contains}(l_1, l_2 \cdot \boldsymbol{t}) & \longrightarrow & \bot & \text{if } \text{contains}(l_1 \backslash l_2, \boldsymbol{t}) \longrightarrow^* \bot \\
\text{contains}(l_1, x \cdot \boldsymbol{t}) & \longrightarrow & \bot & \text{if } \text{contains}(l_1, \boldsymbol{t}) \longrightarrow^* \bot \\
\text{contains}(l_1 \cdot \boldsymbol{t}, l_2) & \longrightarrow & \top & \text{if } l_1 \text{ contains } l_2 \\
\text{contains}(x \cdot \boldsymbol{t}, s) & \longrightarrow & \top & \text{if } \text{contains}(\boldsymbol{t}, s) \longrightarrow^* \top \\
\text{contains}(t \cdot \boldsymbol{s}, t \cdot \boldsymbol{u}) & \longrightarrow & \top & \text{if } \text{contains}(\boldsymbol{s}, \boldsymbol{u}) \longrightarrow^* \top \\
\text{contains}(l_1 \cdot \boldsymbol{t}, l_2) & \longrightarrow & \text{contains}(\boldsymbol{t}, l_2) & \text{if } l_1 \sqcup_l l_2 = \epsilon \\
\text{contains}(\boldsymbol{t} \cdot l_1, l_2) & \longrightarrow & \text{contains}(\boldsymbol{t}, l_2) & \text{if } l_1 \sqcup_r l_2 = \epsilon
\end{array}
$$

$$\text{contains}(\epsilon, t) = \top \quad \longrightarrow \quad \epsilon = t$$

$$\cdots$$

Simplification rules for New Operators

Example: contains (\l_1, l_2 denote string constants)

$$
\begin{array}{rcll}
\text{contains}(l_1, l_2) & \longrightarrow & \top & \text{if } l_1 \text{ contains } l_2 \\
\text{contains}(l_1, l_2) & \longrightarrow & \bot & \text{if } l_1 \text{ does not contain } l_2 \\
\text{contains}(l_1, l_2 \cdot \boldsymbol{t}) & \longrightarrow & \bot & \text{if } l_1 \text{ does not contain } l_2 \\
\text{contains}(l_1, l_2 \cdot \boldsymbol{t}) & \longrightarrow & \bot & \text{if contains}(l_1 \backslash l_2, \boldsymbol{t}) \longrightarrow^* \bot \\
\text{contains}(l_1, x \cdot \boldsymbol{t}) & \longrightarrow & \bot & \text{if contains}(l_1, \boldsymbol{t}) \longrightarrow^* \bot \\
\text{contains}(l_1 \cdot \boldsymbol{t}, l_2) & \longrightarrow & \top & \text{if } l_1 \text{ contains } l_2 \\
\text{contains}(x \cdot \boldsymbol{t}, s) & \longrightarrow & \top & \text{if contains}(\boldsymbol{t}, s) \longrightarrow^* \top \\
\text{contains}(t \cdot \boldsymbol{s}, t \cdot \boldsymbol{u}) & \longrightarrow & \top & \text{if contains}(\boldsymbol{s}, \boldsymbol{u}) \longrightarrow^* \top \\
\text{contains}(l_1 \cdot \boldsymbol{t}, l_2) & \longrightarrow & \text{contains}(\boldsymbol{t}, l_2) & \text{if } l_1 \sqcup_l l_2 = \epsilon \\
\text{contains}(\boldsymbol{t} \cdot l_1, l_2) & \longrightarrow & \text{contains}(\boldsymbol{t}, l_2) & \text{if } l_1 \sqcup_r l_2 = \epsilon
\end{array}
$$

$$
\text{contains}(\epsilon, t) = \top \longrightarrow \epsilon = t
$$

$$\cdots$$

Simplification rules for New Operators

Example: contains (l_1, l_2 denote string constants)

$$
\begin{array}{rcll}
\text{contains}(l_1, l_2) & \longrightarrow & \top & \text{if } l_1 \text{ contains } l_2 \\
\text{contains}(l_1, l_2) & \longrightarrow & \bot & \text{if } l_1 \text{ does not contain } l_2 \\
\text{contains}(l_1, l_2 \cdot \boldsymbol{t}) & \longrightarrow & \bot & \text{if } l_1 \text{ does not contain } l_2 \\
\text{contains}(l_1, l_2 \cdot \boldsymbol{t}) & \longrightarrow & \bot & \text{if contains}(l_1 \backslash l_2, \boldsymbol{t}) \longrightarrow^* \bot \\
\text{contains}(l_1, x \cdot \boldsymbol{t}) & \longrightarrow & \bot & \text{if contains}(l_1, \boldsymbol{t}) \longrightarrow^* \bot \\
\text{contains}(l_1 \cdot \boldsymbol{t}, l_2) & \longrightarrow & \top & \text{if } l_1 \text{ contains } l_2 \\
\text{contains}(x \cdot \boldsymbol{t}, s) & \longrightarrow & \top & \text{if contains}(\boldsymbol{t}, s) \longrightarrow^* \top \\
\text{contains}(t \cdot \boldsymbol{s}, t \cdot \boldsymbol{u}) & \longrightarrow & \top & \text{if contains}(\boldsymbol{s}, \boldsymbol{u}) \longrightarrow^* \top \\
\text{contains}(l_1 \cdot \boldsymbol{t}, l_2) & \longrightarrow & \text{contains}(\boldsymbol{t}, l_2) & \text{if } l_1 \sqcup_l l_2 = \epsilon \\
\text{contains}(\boldsymbol{t} \cdot l_1, l_2) & \longrightarrow & \text{contains}(\boldsymbol{t}, l_2) & \text{if } l_1 \sqcup_r l_2 = \epsilon
\end{array}
$$

$$
\text{contains}(\epsilon, t) = \top \quad \longrightarrow \quad \epsilon = t
$$

$$
\cdots
$$

Reasoning about New Operators: More Performance

Iterate and Improve

- Supercharge the simplifier
- Many simplifications are conditional
- Build a mini-inference engine inside the simplifier to verify *simplification conditions*

Conditional Simplifications based on String Length

Notation: $\vdash C$ states that simplifier can prove simplification condition C

$$
\begin{array}{rcll}
t \doteq s & \longrightarrow & \bot & \text{if} \quad \vdash |t| > |s| \\
t \doteq s \cdot r \cdot q & \longrightarrow & t \doteq s \cdot q \wedge r \doteq \epsilon & \text{if} \quad \vdash |s| + |q| \geqslant |t| \\
\mathrm{contains}(t, s) & \longrightarrow & t \doteq s & \text{if} \quad \vdash |s| \geqslant |t| \\
\mathrm{substr}(t, v, w) & \longrightarrow & \epsilon & \text{if} \quad \vdash 0 > v \vee v \geqslant |t| \vee 0 \geqslant w \\
\mathrm{substr}(t \cdot s, v, w) & \longrightarrow & \mathrm{substr}(s, v - |t|, w) & \text{if} \quad \vdash v \geqslant |t| \\
\mathrm{substr}(s \cdot t, v, w) & \longrightarrow & \mathrm{substr}(s, v, w) & \text{if} \quad \vdash |s| \geqslant v + w \\
\mathrm{substr}(t \cdot s, 0, w) & \longrightarrow & t \cdot \mathrm{substr}(s, 0, w - |t|) & \text{if} \quad \vdash w \geqslant |t| \\
\mathrm{index\_of}(t, s, v) & \longrightarrow & \mathtt{ite}(\mathrm{substr}(t, v) \doteq s, v, -1) & \text{if} \quad \vdash v + |s| \geqslant |t| \\
\cdots & & &
\end{array}
$$

Conditional Simplifications based on String Length

Notation: $\vdash C$ states that simplifier can prove simplification condition C

$$\begin{array}{rcll}
t \doteq s & \longrightarrow & \bot & \text{if} \quad \vdash |t| > |s| \\
t \doteq s \cdot r \cdot q & \longrightarrow & t \doteq s \cdot q \wedge r \doteq \epsilon & \text{if} \quad \vdash |s| + |q| \geqslant |t| \\
\mathrm{contains}(t, s) & \longrightarrow & t \doteq s & \text{if} \quad \vdash |s| \geqslant |t| \\
\mathrm{substr}(t, v, w) & \longrightarrow & \epsilon & \text{if} \quad \vdash 0 > v \vee v \geqslant |t| \vee 0 \geqslant w \\
\mathrm{substr}(t \cdot s, v, w) & \longrightarrow & \mathrm{substr}(s, v - |t|, w) & \text{if} \quad \vdash v \geqslant |t| \\
\mathrm{substr}(s \cdot t, v, w) & \longrightarrow & \mathrm{substr}(s, v, w) & \text{if} \quad \vdash |s| \geqslant v + w \\
\mathrm{substr}(t \cdot s, 0, w) & \longrightarrow & t \cdot \mathrm{substr}(s, 0, w - |t|) & \text{if} \quad \vdash w \geqslant |t| \\
\mathrm{index\_of}(t, s, v) & \longrightarrow & \mathtt{ite}(\mathrm{substr}(t, v) \doteq s, v, -1) & \text{if} \quad \vdash v + |s| \geqslant |t| \\
\dots & & &
\end{array}$$

Conditional Simplifications based on String Length

Notation: $\vdash C$ states that simplifier can prove simplification condition C

$$
\begin{array}{rcll}
t \doteq s & \longrightarrow & \bot & \text{if} \quad \vdash |t| > |s| \\
t \doteq s \cdot r \cdot q & \longrightarrow & t \doteq s \cdot q \wedge r \doteq \epsilon & \text{if} \quad \vdash |s| + |q| \geqslant |t| \\
\text{contains}(t, s) & \longrightarrow & t \doteq s & \text{if} \quad \vdash |s| \geqslant |t| \\
\text{substr}(t, v, w) & \longrightarrow & \epsilon & \text{if} \quad \vdash 0 > v \vee v \geqslant |t| \vee 0 \geqslant w \\
\text{substr}(t \cdot s, v, w) & \longrightarrow & \text{substr}(s, v - |t|, w) & \text{if} \quad \vdash v \geqslant |t| \\
\text{substr}(s \cdot t, v, w) & \longrightarrow & \text{substr}(s, v, w) & \text{if} \quad \vdash |s| \geqslant v + w \\
\text{substr}(t \cdot s, 0, w) & \longrightarrow & t \cdot \text{substr}(s, 0, w - |t|) & \text{if} \quad \vdash w \geqslant |t| \\
\text{index\_of}(t, s, v) & \longrightarrow & \mathtt{ite}(\text{substr}(t, v) \doteq s, v, -1) & \text{if} \quad \vdash v + |s| \geqslant |t| \\
\ldots & & &
\end{array}
$$

Conditional Simplifications based on String Length

Notation: $\vdash C$ states that simplifier can prove simplification condition C

$$
\begin{array}{rcll}
t \doteq s & \longrightarrow & \bot & \text{if} \quad \vdash |t| > |s| \\
t \doteq s \cdot r \cdot q & \longrightarrow & t \doteq s \cdot q \wedge r \doteq \epsilon & \text{if} \quad \vdash |s| + |q| \geqslant |t| \\
\mathsf{contains}(t, s) & \longrightarrow & t \doteq s & \text{if} \quad \vdash |s| \geqslant |t| \\
\mathsf{substr}(t, v, w) & \longrightarrow & \epsilon & \text{if} \quad \vdash 0 > v \vee v \geqslant |t| \vee 0 \geqslant w \\
\mathsf{substr}(t \cdot s, v, w) & \longrightarrow & \mathsf{substr}(s, v - |t|, w) & \text{if} \quad \vdash v \geqslant |t| \\
\mathsf{substr}(s \cdot t, v, w) & \longrightarrow & \mathsf{substr}(s, v, w) & \text{if} \quad \vdash |s| \geqslant v + w \\
\mathsf{substr}(t \cdot s, 0, w) & \longrightarrow & t \cdot \mathsf{substr}(s, 0, w - |t|) & \text{if} \quad \vdash w \geqslant |t| \\
\mathsf{index\_of}(t, s, v) & \longrightarrow & \mathtt{ite}(\mathsf{substr}(t, v) \doteq s, v, -1) & \text{if} \quad \vdash v + |s| \geqslant |t| \\
\ldots & & &
\end{array}
$$

Conditional Simplifications based on String Length

Notation: $\vdash C$ states that simplifier can prove simplification condition C

$$\begin{aligned}
t \doteq s &\longrightarrow \bot && \text{if} \quad \vdash |t| > |s| \\
t \doteq s \cdot r \cdot q &\longrightarrow t \doteq s \cdot q \wedge r \doteq \epsilon && \text{if} \quad \vdash |s| + |q| \geqslant |t| \\
\text{contains}(t, s) &\longrightarrow t \doteq s && \text{if} \quad \vdash |s| \geqslant |t| \\
\text{substr}(t, v, w) &\longrightarrow \epsilon && \text{if} \quad \vdash 0 > v \vee v \geqslant |t| \vee 0 \geqslant w \\
\text{substr}(t \cdot s, v, w) &\longrightarrow \text{substr}(s, v - |t|, w) && \text{if} \quad \vdash v \geqslant |t| \\
\text{substr}(s \cdot t, v, w) &\longrightarrow \text{substr}(s, v, w) && \text{if} \quad \vdash |s| \geqslant v + w \\
\text{substr}(t \cdot s, 0, w) &\longrightarrow t \cdot \text{substr}(s, 0, w - |t|) && \text{if} \quad \vdash w \geqslant |t| \\
\text{index\_of}(t, s, v) &\longrightarrow \texttt{ite}(\text{substr}(t, v) \doteq s, v, -1) && \text{if} \quad \vdash v + |s| \geqslant |t| \\
&\dots
\end{aligned}$$

Conditional Simplifications based on String Length

Notation: $\vdash C$ states that simplifier can prove simplification condition C

$$
\begin{array}{rcll}
t \doteq s & \longrightarrow & \bot & \text{if} \quad \vdash |t| > |s| \\
t \doteq s \cdot r \cdot q & \longrightarrow & t \doteq s \cdot q \wedge r \doteq \epsilon & \text{if} \quad \vdash |s| + |q| \geqslant |t| \\
\text{contains}(t, s) & \longrightarrow & t \doteq s & \text{if} \quad \vdash |s| \geqslant |t| \\
\text{substr}(t, v, w) & \longrightarrow & \epsilon & \text{if} \quad \vdash 0 > v \vee v \geqslant |t| \vee 0 \geqslant w \\
\text{substr}(t \cdot s, v, w) & \longrightarrow & \text{substr}(s, v - |t|, w) & \text{if} \quad \vdash v \geqslant |t| \\
\text{substr}(s \cdot t, v, w) & \longrightarrow & \text{substr}(s, v, w) & \text{if} \quad \vdash |s| \geqslant v + w \\
\text{substr}(t \cdot s, 0, w) & \longrightarrow & t \cdot \text{substr}(s, 0, w - |t|) & \text{if} \quad \vdash w \geqslant |t| \\
\text{index\_of}(t, s, v) & \longrightarrow & \mathtt{ite}(\text{substr}(t, v) \doteq s, v, -1) & \text{if} \quad \vdash v + |s| \geqslant |t| \\
\cdots & & &
\end{array}
$$

Conditional Simplifications based on String Length

Notation: $\vdash C$ states that simplifier can prove simplification condition C

$$
\begin{array}{rcll}
t \doteq s & \longrightarrow & \bot & \text{if} \quad \vdash |t| > |s| \\
t \doteq s \cdot r \cdot q & \longrightarrow & t \doteq s \cdot q \wedge r \doteq \epsilon & \text{if} \quad \vdash |s| + |q| \geqslant |t| \\
\mathrm{contains}(t, s) & \longrightarrow & t \doteq s & \text{if} \quad \vdash |s| \geqslant |t| \\
\mathrm{substr}(t, v, w) & \longrightarrow & \epsilon & \text{if} \quad \vdash 0 > v \vee v \geqslant |t| \vee 0 \geqslant w \\
\mathrm{substr}(t \cdot s, v, w) & \longrightarrow & \mathrm{substr}(s, v - |t|, w) & \text{if} \quad \vdash v \geqslant |t| \\
\mathrm{substr}(s \cdot t, v, w) & \longrightarrow & \mathrm{substr}(s, v, w) & \text{if} \quad \vdash |s| \geqslant v + w \\
\mathrm{substr}(t \cdot s, 0, w) & \longrightarrow & t \cdot \mathrm{substr}(s, 0, w - |t|) & \text{if} \quad \vdash w \geqslant |t| \\
\mathrm{index\_of}(t, s, v) & \longrightarrow & \mathtt{ite}(\mathrm{substr}(t, v) \doteq s, v, -1) & \text{if} \quad \vdash v + |s| \geqslant |t| \\
\ldots & & &
\end{array}
$$

Conditional Simplifications based on String Length

Notation: $\vdash C$ states that simplifier can prove simplification condition C

$$\begin{array}{rcll}
t \doteq s & \longrightarrow & \bot & \text{if} \quad \vdash |t| > |s| \\
t \doteq s \cdot r \cdot q & \longrightarrow & t \doteq s \cdot q \wedge r \doteq \epsilon & \text{if} \quad \vdash |s| + |q| \geqslant |t| \\
\mathrm{contains}(t, s) & \longrightarrow & t \doteq s & \text{if} \quad \vdash |s| \geqslant |t| \\
\mathrm{substr}(t, v, w) & \longrightarrow & \epsilon & \text{if} \quad \vdash 0 > v \vee v \geqslant |t| \vee 0 \geqslant w \\
\mathrm{substr}(t \cdot s, v, w) & \longrightarrow & \mathrm{substr}(s, v - |t|, w) & \text{if} \quad \vdash v \geqslant |t| \\
\mathrm{substr}(s \cdot t, v, w) & \longrightarrow & \mathrm{substr}(s, v, w) & \text{if} \quad \vdash |s| \geqslant v + w \\
\mathrm{substr}(t \cdot s, 0, w) & \longrightarrow & t \cdot \mathrm{substr}(s, 0, w - |t|) & \text{if} \quad \vdash w \geqslant |t| \\
\mathrm{index\_of}(t, s, v) & \longrightarrow & \texttt{ite}(\mathrm{substr}(t, v) \doteq s, v, -1) & \text{if} \quad \vdash v + |s| \geqslant |t| \\
\cdots & & &
\end{array}$$

Conditional Simplifications based on String Length

Notation: $\vdash C$ states that simplifier can prove simplification condition C

$$\begin{array}{rcll}
t \doteq s & \longrightarrow & \bot & \text{if} \quad \vdash |t| > |s| \\
t \doteq s \cdot r \cdot q & \longrightarrow & t \doteq s \cdot q \wedge r \doteq \epsilon & \text{if} \quad \vdash |s| + |q| \geqslant |t| \\
\mathrm{contains}(t, s) & \longrightarrow & t \doteq s & \text{if} \quad \vdash |s| \geqslant |t| \\
\mathrm{substr}(t, v, w) & \longrightarrow & \epsilon & \text{if} \quad \vdash 0 > v \vee v \geqslant |t| \vee 0 \geqslant w \\
\mathrm{substr}(t \cdot s, v, w) & \longrightarrow & \mathrm{substr}(s, v - |t|, w) & \text{if} \quad \vdash v \geqslant |t| \\
\mathrm{substr}(s \cdot t, v, w) & \longrightarrow & \mathrm{substr}(s, v, w) & \text{if} \quad \vdash |s| \geqslant v + w \\
\mathrm{substr}(t \cdot s, 0, w) & \longrightarrow & t \cdot \mathrm{substr}(s, 0, w - |t|) & \text{if} \quad \vdash w \geqslant |t| \\
\mathrm{index\_of}(t, s, v) & \longrightarrow & \texttt{ite}(\mathrm{substr}(t, v) \doteq s, v, -1) & \text{if} \quad \vdash v + |s| \geqslant |t| \\
\ldots & & &
\end{array}$$

Reasoning about Regular Expressions

Regular Expression Membership Example

$$\begin{aligned} &x \in [0..9]^* \cdot \texttt{"a"} \cdot \mathcal{A}^* \cdot \texttt{"b"} \cdot \mathcal{A}^* \quad \wedge \\ &x \notin [0..9]^* \cdot \texttt{"a"} \cdot \mathcal{A}^* \end{aligned}$$

Automata-based approach

$$\frac{x \in R_1 \quad x \notin R_2}{x \in R_1 \cap \mathrm{comp}(R_2)}$$

Problem:

Reasoning about Regular Expressions

Regular Expression Membership Example

$$x \in [0..9]^* \cdot \texttt{"a"} \cdot \mathcal{A}^* \cdot \texttt{"b"} \cdot \mathcal{A}^* \quad \wedge$$
$$x \notin [0..9]^* \cdot \texttt{"a"} \cdot \mathcal{A}^*$$

Automata-based approach

$$\frac{x \in R_1 \quad x \notin R_2}{x \in R_1 \cap \mathrm{comp}(R_2)}$$

Problem: Complement and intersection are expensive

Reasoning about Regular Expressions

Regular Expression Membership Example

$$\begin{aligned} &x \in [0..9]^* \cdot \texttt{"a"} \cdot \mathcal{A}^* \cdot \texttt{"b"} \cdot \mathcal{A}^* \quad \wedge \\ &x \notin [0..9]^* \cdot \texttt{"a"} \cdot \mathcal{A}^* \end{aligned}$$

Automata-based approach

$$\frac{x \in R_1 \quad x \notin R_2}{x \in R_1 \cap \mathsf{comp}(R_2)}$$

Problem: Membership constraints may lead to non-terminating unfolding:

Reasoning about Regular Expressions

Regular Expression Membership Example

$$x \in [0..9]^* \cdot \texttt{"a"} \cdot \mathcal{A}^* \cdot \texttt{"b"} \cdot \mathcal{A}^* \quad \wedge$$
$$x \notin [0..9]^* \cdot \texttt{"a"} \cdot \mathcal{A}^*$$

Automata-based approach

$$\frac{x \in R_1 \qquad x \notin R_2}{x \in R_1 \cap \mathrm{comp}(R_2)}$$

Problem: Membership constraints may lead to non-terminating unfolding:

Example: $x_1 \in [0..9]^*$ is equivalent to

$$x \doteq \epsilon \ \vee \ x \in [0..9] \ \vee \ (\exists u, v, w.\, x \doteq u \cdot v \cdot w \ \wedge \ u \in [0..9] \ \wedge \ v \in [0..9]^* \wedge w \in [0..9])$$

Reasoning about Regular Expressions

Regular Expression Membership Example

$$x \in [0..9]^* \cdot \texttt{"a"} \cdot \mathcal{A}^* \cdot \texttt{"b"} \cdot \mathcal{A}^* \quad \wedge$$
$$x \notin [0..9]^* \cdot \texttt{"a"} \cdot \mathcal{A}^*$$

Word-based approach with incomplete procedures

$$\frac{x \in R_1 \quad x \notin R_2 \quad \mathcal{L}(R_1) \subseteq \mathcal{L}(R_2)}{\text{UNSAT}}$$

Use fast, incomplete procedure to verify $\mathcal{L}(R_1) \subseteq \mathcal{L}(R_2)$

Notation: $\mathcal{L}(R)$ denotes the language generated by regex R

Proving $\mathcal{L}(R_1) \subseteq \mathcal{L}(R_2)$

(1) $$\frac{}{\mathcal{L}(R) \subseteq \mathcal{L}(R)}$$

(2) $$\frac{}{\mathcal{L}(\epsilon) \subseteq \mathcal{L}(R^*)}$$

(3) $$\frac{\text{for all } x \in \mathcal{L}(R),\ |x| = 1}{\mathcal{L}(R) \subseteq \mathcal{L}(\mathcal{A})}$$

(4) $$\frac{}{\mathcal{L}(R) \subseteq \mathcal{L}(\mathcal{A}^*)}$$

(5) $$\frac{}{\mathcal{L}(R) \subseteq \mathcal{L}(R^*)}$$

(6) $$\frac{\mathcal{L}(R_1) \subseteq \mathcal{L}(R_2)}{\mathcal{L}(R_1^*) \subseteq \mathcal{L}(R_2^*)}$$

(7) $$\frac{\mathcal{L}(R_1) \subseteq \mathcal{L}(R_2) \quad \mathcal{L}(R_2) \subseteq \mathcal{L}(R_3)}{\mathcal{L}(R_1) \subseteq \mathcal{L}(R_3)}$$

(8) $$\frac{\mathcal{L}(R_1) \subseteq \mathcal{L}(S_1) \quad \mathcal{L}(R_2) \subseteq \mathcal{L}(S_2)}{\mathcal{L}(R_1 \cdot R_2) \subseteq \mathcal{L}(S_1 \cdot S_2)}$$

(9) $$\frac{c_3 \preceq c_1 \quad c_2 \preceq c_4}{\mathcal{L}([c_1..c_2]) \subseteq \mathcal{L}([c_3..c_4])}$$

$c \preceq d$ iff c equals d or precedes d lexicographically ($c, d \in \mathcal{A}$)

Proving $\mathcal{L}(R_1) \subseteq \mathcal{L}(R_2)$

(1) $$\frac{}{\mathcal{L}(R) \subseteq \mathcal{L}(R)}$$

(2) $$\frac{}{\mathcal{L}(\epsilon) \subseteq \mathcal{L}(R^*)}$$

(3) $$\frac{\text{for all } x \in \mathcal{L}(R),\ |x| = 1}{\mathcal{L}(R) \subseteq \mathcal{L}(\mathcal{A})}$$

(4) $$\frac{}{\mathcal{L}(R) \subseteq \mathcal{L}(\mathcal{A}^*)}$$

(5) $$\frac{}{\mathcal{L}(R) \subseteq \mathcal{L}(R^*)}$$

(6) $$\frac{\mathcal{L}(R_1) \subseteq \mathcal{L}(R_2)}{\mathcal{L}(R_1^*) \subseteq \mathcal{L}(R_2^*)}$$

(7) $$\frac{\mathcal{L}(R_1) \subseteq \mathcal{L}(R_2) \quad \mathcal{L}(R_2) \subseteq \mathcal{L}(R_3)}{\mathcal{L}(R_1) \subseteq \mathcal{L}(R_3)}$$

(8) $$\frac{\mathcal{L}(R_1) \subseteq \mathcal{L}(S_1) \quad \mathcal{L}(R_2) \subseteq \mathcal{L}(S_2)}{\mathcal{L}(R_1 \cdot R_2) \subseteq \mathcal{L}(S_1 \cdot S_2)}$$

(9) $$\frac{c_3 \preceq c_1 \quad c_2 \preceq c_4}{\mathcal{L}([c_1..c_2]) \subseteq \mathcal{L}([c_3..c_4])}$$

$c \preceq d$ iff c equals d or precedes d lexicographically ($c, d \in \mathcal{A}$)

Proving $\mathcal{L}(R_1) \subseteq \mathcal{L}(R_2)$

(1) $$\frac{}{\mathcal{L}(R) \subseteq \mathcal{L}(R)}$$

(2) $$\frac{}{\mathcal{L}(\epsilon) \subseteq \mathcal{L}(R^*)}$$

(3) $$\frac{\text{for all } x \in \mathcal{L}(R),\ |x| = 1}{\mathcal{L}(R) \subseteq \mathcal{L}(\mathcal{A})}$$

(4) $$\frac{}{\mathcal{L}(R) \subseteq \mathcal{L}(\mathcal{A}^*)}$$

(5) $$\frac{}{\mathcal{L}(R) \subseteq \mathcal{L}(R^*)}$$

(6) $$\frac{\mathcal{L}(R_1) \subseteq \mathcal{L}(R_2)}{\mathcal{L}(R_1^*) \subseteq \mathcal{L}(R_2^*)}$$

(7) $$\frac{\mathcal{L}(R_1) \subseteq \mathcal{L}(R_2) \quad \mathcal{L}(R_2) \subseteq \mathcal{L}(R_3)}{\mathcal{L}(R_1) \subseteq \mathcal{L}(R_3)}$$

(8) $$\frac{\mathcal{L}(R_1) \subseteq \mathcal{L}(S_1) \quad \mathcal{L}(R_2) \subseteq \mathcal{L}(S_2)}{\mathcal{L}(R_1 \cdot R_2) \subseteq \mathcal{L}(S_1 \cdot S_2)}$$

(9) $$\frac{c_3 \preceq c_1 \quad c_2 \preceq c_4}{\mathcal{L}([c_1..c_2]) \subseteq \mathcal{L}([c_3..c_4])}$$

$c \preceq d$ iff c equals d or precedes d lexicographically ($c, d \in \mathcal{A}$)

Proving $\mathcal{L}(R_1) \subseteq \mathcal{L}(R_2)$

(1) $$\frac{}{\mathcal{L}(R) \subseteq \mathcal{L}(R)}$$

(2) $$\frac{}{\mathcal{L}(\epsilon) \subseteq \mathcal{L}(R^*)}$$

(3) $$\frac{\text{for all } x \in \mathcal{L}(R),\ |x| = 1}{\mathcal{L}(R) \subseteq \mathcal{L}(\mathcal{A})}$$

(4) $$\frac{}{\mathcal{L}(R) \subseteq \mathcal{L}(\mathcal{A}^*)}$$

(5) $$\frac{}{\mathcal{L}(R) \subseteq \mathcal{L}(R^*)}$$

(6) $$\frac{\mathcal{L}(R_1) \subseteq \mathcal{L}(R_2)}{\mathcal{L}(R_1^*) \subseteq \mathcal{L}(R_2^*)}$$

(7) $$\frac{\mathcal{L}(R_1) \subseteq \mathcal{L}(R_2) \quad \mathcal{L}(R_2) \subseteq \mathcal{L}(R_3)}{\mathcal{L}(R_1) \subseteq \mathcal{L}(R_3)}$$

(8) $$\frac{\mathcal{L}(R_1) \subseteq \mathcal{L}(S_1) \quad \mathcal{L}(R_2) \subseteq \mathcal{L}(S_2)}{\mathcal{L}(R_1 \cdot R_2) \subseteq \mathcal{L}(S_1 \cdot S_2)}$$

(9) $$\frac{c_3 \preceq c_1 \quad c_2 \preceq c_4}{\mathcal{L}([c_1..c_2]) \subseteq \mathcal{L}([c_3..c_4])}$$

$c \preceq d$ iff c equals d or precedes d lexicographically ($c, d \in \mathcal{A}$)

Exercise

Using the proof rules above, prove that

$$\mathcal{L}([0..1]^* \cdot \mathcal{A}^* \cdot \texttt{"b"} \cdot \mathcal{A}^*) \subseteq \mathcal{L}([0..9]^* \cdot \mathcal{A}^*)$$

$$\frac{\dfrac{\dfrac{0 \le 0 \quad 1 \le 9}{\mathcal{L}([0..1]) \subseteq \mathcal{L}([0..9])}\,(9)}{\mathcal{L}([0..1]^*) \subseteq \mathcal{L}([0..9]^*)}\,(6) \qquad \dfrac{}{\mathcal{L}(\mathcal{A}^* \cdot \texttt{"b"} \cdot \mathcal{A}^*) \subseteq \mathcal{L}(\mathcal{A}^*)}\,(4)}{\mathcal{L}([0..1]^* \cdot \mathcal{A}^* \cdot \texttt{"b"} \cdot \mathcal{A}^*) \subseteq \mathcal{L}([0..9]^* \cdot \mathcal{A}^*)}\,(8)$$

Exercise

Using the proof rules above, prove that

$$\mathcal{L}([0..1]^* \cdot \mathcal{A}^* \cdot \texttt{"b"} \cdot \mathcal{A}^*) \subseteq \mathcal{L}([0..9]^* \cdot \mathcal{A}^*)$$

$$\cfrac{\cfrac{\cfrac{0 \leq 0 \quad 1 \leq 9}{\mathcal{L}([0..1]) \subseteq \mathcal{L}([0..9])}\;(9)}{\mathcal{L}([0..1]^*) \subseteq \mathcal{L}([0..9]^*)}\;(6) \qquad \cfrac{}{\mathcal{L}(\mathcal{A}^* \cdot \texttt{"b"} \cdot \mathcal{A}^*) \subseteq \mathcal{L}(\mathcal{A}^*)}\;(4)}{\mathcal{L}([0..1]^* \cdot \mathcal{A}^* \cdot \texttt{"b"} \cdot \mathcal{A}^*) \subseteq \mathcal{L}([0..9]^* \cdot \mathcal{A}^*)}\;(8)$$

More Information

Strings Papers

- “A DPLL(T) Theory Solver for a Theory of Strings and Regular Expressions” by Tianyi Liang, Andrew Reynolds, Cesare Tinelli, Clark Barrett, and Morgan Deters. In Proceedings of the 26th International Conference on Computer Aided Verification (CAV ’14), (Armin Biere and Roderick Bloem, eds.), July 2014, pp. 646-662. Vienna, Austria.
- “An Efficient SMT Solver for String Constraints” by Tianyi Liang, Andrew Reynolds, Nestan Tsiskaridze, Cesare Tinelli, Clark Barrett, and Morgan Deters. Formal Methods in System Design, vol. 48, no. 3, June 2016, pp. 206-234, Springer US.
- “Scaling up DPLL(T) String Solvers Using Context-Dependent Simplification” by Andrew Reynolds, Maverick Woo, Clark Barrett, David Brumley, Tianyi Liang, and Cesare Tinelli. In Proceedings of the 29th International Conference on Computer Aided Verification (CAV ’17), (Rupak Majumdar and Viktor Kuncak, eds.), July 2017, pp. 453-474. Heidelberg, Germany.
- “High-Level Abstractions for Simplifying Extended String Constraints in SMT” by Andrew Reynolds, Andres Nötzli, Clark Barrett, and Cesare Tinelli. In Proceedings of the 31st International Conference on Computer Aided Verification (CAV ’19), (Isil Dillig and Serdar Tasiran, eds.), July 2019, pp. 23-42. New York, New York.
- “Even Faster Conflicts and Lazier Reductions for String Solvers” by Andres Nötzli, Andrew Reynolds, Haniel Barbosa, Clark Barrett, and Cesare Tinelli. In Proceedings of the 34th International Conference on Computer Aided Verification (CAV ’22), (Sharon Shoham and Yakir Vizel, eds.), Aug. 2022, pp. 205-226. Haifa, Israel.

Amazon’s Zelkova Tool

- J. Backes et al., “Semantic-based Automated Reasoning for AWS Access Policies using SMT,” 2018 Formal Methods in Computer Aided Design (FMCAD), Austin, TX, 2018.