String Prefix Domain

1. Concrete Domain Definition

We define the concrete lattice and the concrete operators + (string concatenation) and ≤ (lexicographic comparison).

1.1 Preliminary Definitions

Let $str \in String$, $str_1 \preceq str_2$ mean that str_1 is a prefix of str_2; $str_1 \prec str_2$ mean that str_1 is a strict prefix of str_2; and $str_1 \oplus str_2$ mean the greatest common prefix of str_1 and str_2. For single strings, ≤ is standard lexicographic comparison and < is strict lexicographic comparison.

1.2 Concrete Lattice

The concrete lattice is the powerset lattice of strings where the partial order is subset inclusion, join is union, and meet is intersection: $\mathcal{L} = (\mathscr{P}(String), \subseteq, \cup, \cap)$.

1.3 Concrete Operators

- Concatenization. For $X, Y \in \mathcal{L}$, $X + Y = \{\ x \cdot y \mid x \in X,\ y \in Y\ \}$.
- Lexicographic comparison. For $X, Y \in \mathcal{L}$, $X \leq Y = A \cup B$ where:
  - $A = \begin{cases} \{\textsf{true}\} & \text{if } \exists x \in X, y \in Y \text{ s.t. } x \leq y \\ \{\} & \text{otherwise} \end{cases}$
  - $B = \begin{cases} \{\textsf{false}\} & \text{if } \exists x \in X, y \in Y \text{ s.t. } y < x \\ \{\} & \text{otherwise} \end{cases}$

2. Abstract Domain Definition

We define the abstract lattice and the abstract operators + (string concatenation) and ≤ (lexicographic comparison). This is a simpler and more streamlined definition than the one in the paper draft.

2.1 Abstract Lattice

The abstract lattice is $\mathcal{L}^\sharp = (Pre, \sqsubseteq, \sqcup, \sqcap)$ where:

- $(str, b) \in Pre = String \times Boolean\ \cup\ \{\ \bot\ \}$

  where $b = \textsf{true}$ means str is an exact string and $b = \textsf{false}$ means str is a prefix of an unknown string.
  - $\top = (\epsilon, \textsf{false})$
  - $\bot$ means an uninitialized string value.

  Think of an element $(str, \textsf{true})$ as representing the set containing a single string str, an element $(str, \textsf{false})$ as representing the set containing all strings beginning with str, and $\bot$ as representing the empty set. Then the definition of $\sqsubseteq$ below is just subset; the definition of $\sqcup$ is almost set union (except the resulting set can contain infinite spurious strings), and the definition of $\sqcap$ is set intersection.

- Dealing with $\bot$. Let X be any lattice element, then:
  - $\bot \sqsubseteq X$
  - $\bot \sqcup X = X \sqcup \bot = X$
  - $\bot \sqcap X = X \sqcap \bot = \bot$

- $(str_1, b_1) \sqsubseteq (str_2, b_2)$ iff
  - $b_2 = \textsf{false}$ and $str_2 \preceq str_1$
  - $b_1 = \textsf{true}, b_2 = \textsf{true}$ and $str_1 = str_2$

- $(str_1, b_1) \sqcup (str_2, b_2) =$
  - (str_1, b_1) if $str_1 = str_2, b_1 = b_2 = \textsf{true}$
  - $(str_1 \oplus str_2, \textsf{false})$ otherwise

- $(str_1, b_1) \sqcap (str_2, b_2) =$
  - (str_1, b_1) if $b_2 = \textsf{false},\ str_2 \preceq str_1$
  - (str_2, b_2) if $b_1 = \textsf{false},\ str_1 \preceq str_2$
  - $\bot$ otherwise

2.2 Abstract Operators

- Concatenization. Let X be any element, then:
  - $\bot + X = X + \bot = \bot$
  - $(str_1, \textsf{true}) + (str_2, b_2) = (str_1 \cdot str_2, b_2)$
  - $(str_1, \textsf{false}) + (str_2, b_2) = (str_1, \textsf{false})$

- Lexicographic comparison. Let X be any element, then:
  - $\bot \leq X = X \leq \bot = \{\}$
  - $(str_1, \textsf{true}) \leq (str_2, \textsf{true}) = \begin{cases} \{\textsf{true}\} & \text{if } str_1 \leq str_2 \\ \{\textsf{false}\} & \text{otherwise} \end{cases}$
  - $(str_1, \textsf{true}) \leq (str_2, \textsf{false}) = \begin{cases} \{\textsf{true}\} & \text{if } str_1 \leq str_2 \\ \{\textsf{true}, \textsf{false}\} & \text{if } str2 \prec str_1 \\ \{\textsf{false}\} & \text{otherwise} \end{cases}$
  - $(str_1, \textsf{false}) \leq (str_2, \textsf{true}) = \begin{cases} \{\textsf{true}\} & \text{if } str_1 < str_2 \\ \{\textsf{true}, \textsf{false}\} & \text{if } str_1 = str_2 \\ \{\textsf{false}\} & \text{otherwise} \end{cases}$
  - $(str_1, \textsf{false}) \leq (str_2, \textsf{false}) = \begin{cases} \{\textsf{true}\} & \text{if } str_1 < str_2 \\ \{\textsf{true}, \textsf{false}\} & \text{if } str2 \prec str_1 \\ \{\textsf{false}\} & \text{otherwise} \end{cases}$

3. Soundness

We define the concretization function γ between $\mathcal{L}$ and $\mathcal{L}^\sharp$ and then prove that the abstract operators are sound.

3.1 Concretization

$$\begin{aligned} \gamma &\in \mathcal{L}^\sharp \to \mathcal{L} \\ \gamma(\bot) &= \{\} \\ \gamma((str, \textsf{true})) &= \{str\} \\ \gamma((str, \textsf{false})) &= \{\ str_i \mid str \preceq str_i\ \} \end{aligned}$$

3.2 Proofs

Theorem 1 (SOUNDNESS OF +). *For $p_1, p_2 \in Pre$, $\gamma(p_1) + \gamma(p_2) \subseteq \gamma(p_1 + p_2)$.*

Proof. Tables 1 and 2 represent a proof by cases. Each cell in Table 2 is a subset of the corresponding cell in Table 1. □

$p_1\downarrow \quad p_2\rightarrow$	$\bot$	$(str_2, \textsf{true})$	$(str_2, \textsf{false})$
$\bot$	$\{\}$	$\{\}$	$\{\}$
$(str_1, \textsf{true})$	$\{\}$	$\{str_1 \cdot str_2\}$	$\{str_1 \cdot str_2 \cdot \Sigma^*\}$
$(str_1, \textsf{false})$	$\{\}$	$\{str_1 \cdot \Sigma^*\}$	$\{str_1 \cdot \Sigma^*\}$

Table 1. Abstract operator cases; each cell is $\gamma(p_1 + p_2)$. We use $str \cdot \Sigma^*$ to represent the set of strings whose prefix is str.

$\gamma(p_1)\downarrow \quad \gamma(p_2)\rightarrow$	$\{\}$	str_2	$str_2 \cdot \Sigma^*$
$\{\}$	$\{\}$	$\{\}$	$\{\}$
str_1	$\{\}$	$\{str_1 \cdot str_2\}$	$\{str_1 \cdot str_2 \cdot \Sigma^*\}$
$str_1 \cdot \Sigma^*$	$\{\}$	$\{str_1 \cdot \Sigma^*\}$	$\{str_1 \cdot \Sigma^*\}$

Table 2. Concrete operator cases; each cell is $\gamma(p_1) + \gamma(p_2)$. We use $str \cdot \Sigma^*$ to represent the set of strings whose prefix is str.

Theorem 2 (SOUNDNESS OF $\leq$)**.** *For $p_1, p_2 \in Pre$, $\gamma(p_1) \leq \gamma(p_2) \subseteq p_1 \leq p_2$.*

Proof. Tables 3 and 4 represent a proof by cases. Each cell in Table 4 is a subset of the corresponding cell in Table 3. □

$p_1\downarrow \quad p_2\rightarrow$	$\bot$	$(str_2, \textsf{true})$	$(str_2, \textsf{false})$
$\bot$	$\{\}$	$\{\}$	$\{\}$
$(str_1, \textsf{true})$	$\{\}$	$\begin{cases} \{\textsf{true}\} & \text{if } str_1 \leq str_2 \\ \{\textsf{false}\} & \text{otherwise} \end{cases}$	$\begin{cases} \{\textsf{true}\} & \text{if } str_1 \leq str_2 \\ \{\textsf{true}, \textsf{false}\} & \text{if } str2 \prec str_1 \\ \{\textsf{false}\} & \text{otherwise} \end{cases}$
$(str_1, \textsf{false})$	$\{\}$	$\begin{cases} \{\textsf{true}\} & \text{if } str_1 < str_2 \\ \{\textsf{true}, \textsf{false}\} & \text{if } str_1 = str_2 \\ \{\textsf{false}\} & \text{otherwise} \end{cases}$	$\begin{cases} \{\textsf{true}\} & \text{if } str_1 < str_2 \\ \{\textsf{true}, \textsf{false}\} & \text{if } str2 \prec str_1 \\ \{\textsf{false}\} & \text{otherwise} \end{cases}$

Table 3. Abstract operator cases; each cell is $p_1 \leq p_2$.

$\gamma(p_1)\downarrow \quad \gamma(p_2)\rightarrow$	$\{\}$	str_2	$str_2 \cdot \Sigma^*$
$\{\}$	$\{\}$	$\{\}$	$\{\}$
str_1	$\{\}$	$\begin{cases} \{\textsf{true}\} & \text{if } str_1 \leq str_2 \\ \{\textsf{false}\} & \text{otherwise} \end{cases}$	$\begin{cases} \{\textsf{true}\} & \text{if } str_1 \leq str_2 \\ \{\textsf{true}, \textsf{false}\} & \text{if } str2 \prec str_1 \\ \{\textsf{false}\} & \text{otherwise} \end{cases}$
$str_1 \cdot \Sigma^*$	$\{\}$	$\begin{cases} \{\textsf{true}\} & \text{if } str_1 < str_2 \\ \{\textsf{true}, \textsf{false}\} & \text{if } str_1 = str_2 \\ \{\textsf{false}\} & \text{otherwise} \end{cases}$	$\begin{cases} \{\textsf{true}\} & \text{if } str_1 < str_2 \\ \{\textsf{true}, \textsf{false}\} & \text{if } str2 \prec str_1 \\ \{\textsf{false}\} & \text{otherwise} \end{cases}$

Table 4. Concrete operator cases; each cell is $\gamma(p_1) \leq \gamma(p_2)$. We use $str \cdot \Sigma^*$ to represent the set of strings whose prefix is str.