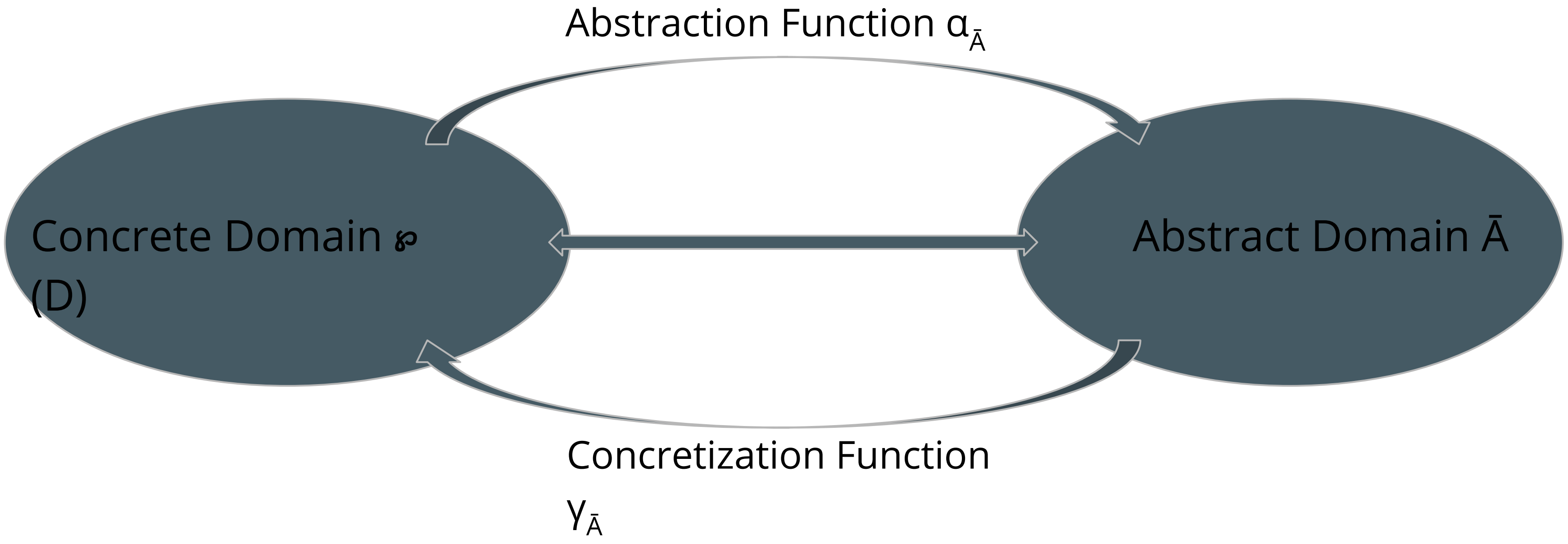
Costantini's "Static Analysis of String Values" - A Summary

Koby Picker
Christian M. Maldonado

Overview

- What is Abstract Interpretation?
- The Concrete Domain
- The Abstract Domains
    a. Character Inclusion
    b. Prefix and Suffix
    c. Bricks
    d. String Graphs

Abstract Interpretation


Abstraction Function $\alpha_{\bar{A}}$
Concrete Domain ℘ (D)
Abstract Domain Ā
Concretization Function $\gamma_{\bar{A}}$

Concrete Domain

- Given an alphabet K, a finite set of characters ...
- Strings = Sequence of characters (potentially infinite)

$S = K^*$ *, where* A^* *is an ordered sequence of elements in A*

$$A^* = \{a_1 \dots a_n : \forall i \in [1 \dots n] : a_i \in A\}$$

Concrete Semantics

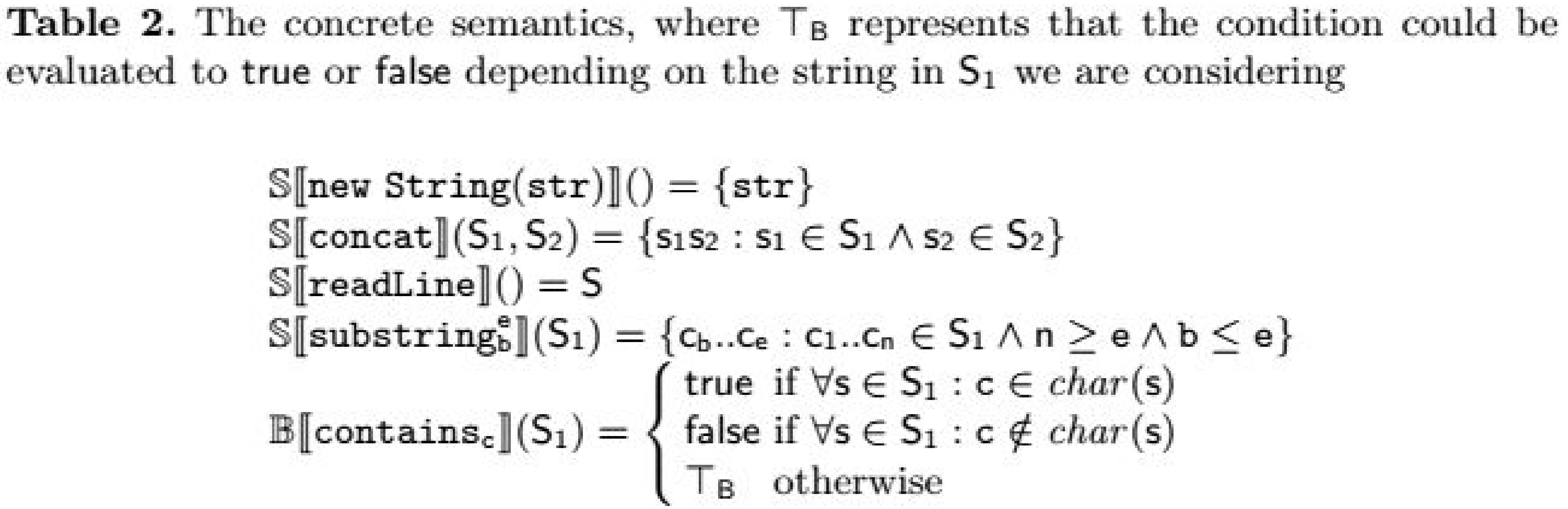


Table 2. The concrete semantics, where $\top_{\mathsf{B}}$ represents that the condition could be evaluated to true or false depending on the string in S_1 we are considering

$$\mathbb{S}[\![\texttt{new String(str)}]\!]() = \{\texttt{str}\}$$

$$\mathbb{S}[\![\texttt{concat}]\!](\mathsf{S}_1, \mathsf{S}_2) = \{\mathsf{s}_1\mathsf{s}_2 : \mathsf{s}_1 \in \mathsf{S}_1 \wedge \mathsf{s}_2 \in \mathsf{S}_2\}$$

$$\mathbb{S}[\![\texttt{readLine}]\!]() = \mathsf{S}$$

$$\mathbb{S}[\![\texttt{substring}_{\mathsf{b}}^{\mathsf{e}}]\!](\mathsf{S}_1) = \{\mathsf{c}_{\mathsf{b}}..\mathsf{c}_{\mathsf{e}} : \mathsf{c}_1..\mathsf{c}_{\mathsf{n}} \in \mathsf{S}_1 \wedge \mathsf{n} \geq \mathsf{e} \wedge \mathsf{b} \leq \mathsf{e}\}$$

$$\mathbb{B}[\![\texttt{contains}_{\mathsf{c}}]\!](\mathsf{S}_1) = \begin{cases} \mathsf{true} \text{ if } \forall \mathsf{s} \in \mathsf{S}_1 : \mathsf{c} \in \mathit{char}(\mathsf{s}) \\ \mathsf{false} \text{ if } \forall \mathsf{s} \in \mathsf{S}_1 : \mathsf{c} \notin \mathit{char}(\mathsf{s}) \\ \top_{\mathsf{B}} \quad \text{otherwise} \end{cases}$$

Overview

- Abstract Interpretation
- The Concrete Domain
- The Abstract Domains
    a. Character Inclusion
    b. Prefix and Suffix
    c. Bricks
    d. String Graphs

Character Inclusion - CI

CI consists of *certainly* contained characters and *maybe contained* characters

$$\mathcal{CI} = \{(C, MC) : C, MC \in \wp(K) \wedge C \subseteq MC\} \cup \bot_{\mathcal{CI}}$$

CI is partially ordered

We can define the *least upper bound* and *greatest lower bound*

Semantics of $\mathcal{CI}$

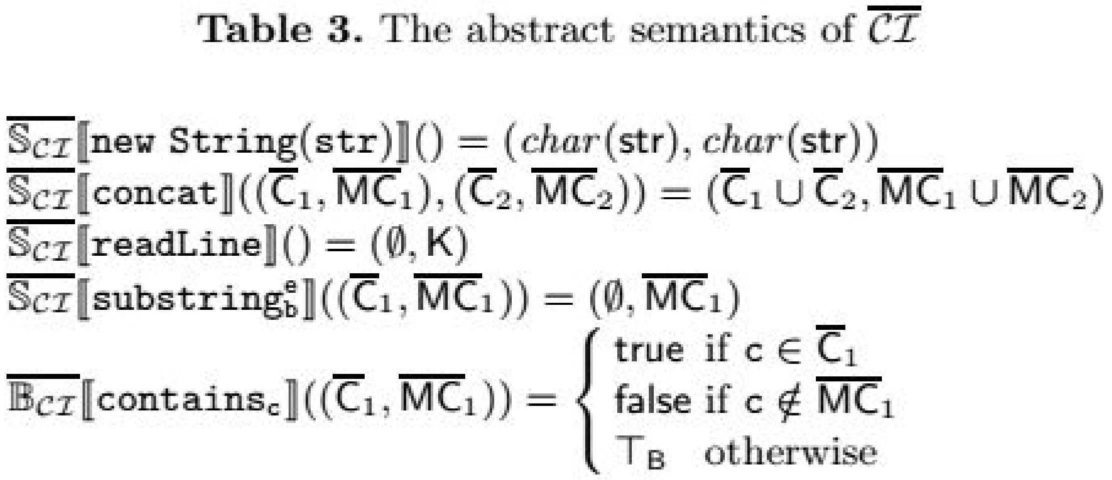


Table 3. The abstract semantics of $\overline{\mathcal{CI}}$

$$\overline{\mathbb{S}_{\mathcal{CI}}}[\![\texttt{new String(str)}]\!]() = (\mathit{char}(\mathsf{str}), \mathit{char}(\mathsf{str}))$$
$$\overline{\mathbb{S}_{\mathcal{CI}}}[\![\texttt{concat}]\!]((\overline{\mathsf{C}}_1, \overline{\mathsf{MC}}_1), (\overline{\mathsf{C}}_2, \overline{\mathsf{MC}}_2)) = (\overline{\mathsf{C}}_1 \cup \overline{\mathsf{C}}_2, \overline{\mathsf{MC}}_1 \cup \overline{\mathsf{MC}}_2)$$
$$\overline{\mathbb{S}_{\mathcal{CI}}}[\![\texttt{readLine}]\!]() = (\emptyset, \mathsf{K})$$
$$\overline{\mathbb{S}_{\mathcal{CI}}}[\![\texttt{substring}_{\mathsf{b}}^{\mathsf{e}}]\!]((\overline{\mathsf{C}}_1, \overline{\mathsf{MC}}_1)) = (\emptyset, \overline{\mathsf{MC}}_1)$$
$$\overline{\mathbb{B}_{\mathcal{CI}}}[\![\texttt{contains}_{\mathsf{c}}]\!]((\overline{\mathsf{C}}_1, \overline{\mathsf{MC}}_1)) = \begin{cases} \mathsf{true} \ \text{if } \mathsf{c} \in \overline{\mathsf{C}}_1 \\ \mathsf{false} \ \text{if } \mathsf{c} \notin \overline{\mathsf{MC}}_1 \\ \top_{\mathsf{B}} \quad \text{otherwise} \end{cases}$$

Prefix - $\mathcal{PR}$

- String = Sequence of characters which *begins* with a certain sequence of characters and ends with any string (ϵ included).

$$\overline{\mathcal{PR}} = \mathsf{K}^* \cup \bot_{\overline{\mathcal{PR}}}$$

- Partial order:

$$\overline{\mathsf{S}} \leq_{\overline{\mathcal{PR}}} \overline{\mathsf{T}} \Leftrightarrow \overline{\mathsf{S}} = \bot_{\overline{\mathcal{PR}}} \vee (\forall \mathrm{i} \in [0, len(\overline{\mathsf{T}}) - 1] : len(\overline{\mathsf{T}}) \leq len(\overline{\mathsf{S}}) \wedge \overline{\mathsf{T}}[\mathrm{i}] = \overline{\mathsf{S}}[\mathrm{i}])$$

An abstract string S is smaller than T if T is a prefix of S or if S is the bottom of the domain

Prefix (Cont.)

Least Upper Bound:

$\sqcup_{PR}$ (S_1, S_2)= Longest common prefix between two strings.

Greatest Lower Bound:

$$\sqcap_{\overline{\mathcal{PR}}}(\overline{\mathsf{S}}_1, \overline{\mathsf{S}}_2) = \begin{cases} \overline{\mathsf{S}}_1 & \text{if } \overline{\mathsf{S}}_1 \leq_{\overline{\mathcal{PR}}} \overline{\mathsf{S}}_2 \\ \overline{\mathsf{S}}_2 & \text{if } \overline{\mathsf{S}}_2 \leq_{\overline{\mathcal{PR}}} \overline{\mathsf{S}}_1 \\ \bot_{\overline{\mathcal{PR}}} & \text{otherwise} \end{cases}$$

Semantics of $\mathcal{PR}$

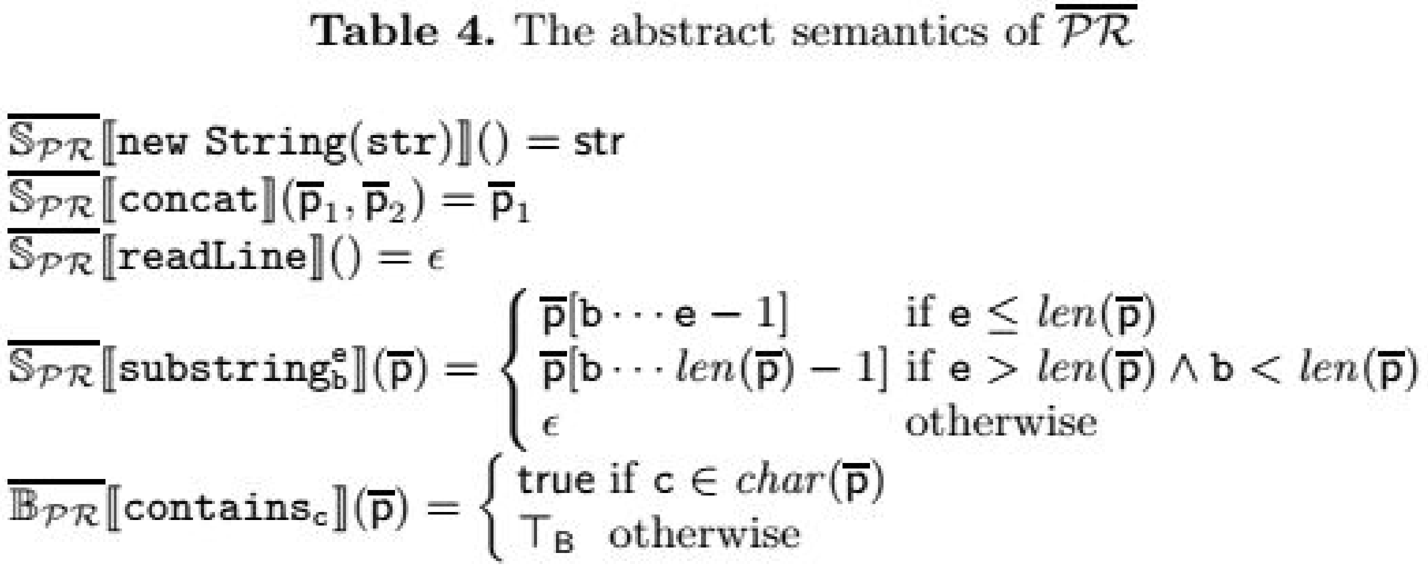


Table 4. The abstract semantics of $\overline{\mathcal{PR}}$

$$\overline{\mathbb{S}_{\mathcal{PR}}}[\![\texttt{new String(str)}]\!]() = \mathsf{str}$$

$$\overline{\mathbb{S}_{\mathcal{PR}}}[\![\texttt{concat}]\!](\overline{\mathsf{p}}_1, \overline{\mathsf{p}}_2) = \overline{\mathsf{p}}_1$$

$$\overline{\mathbb{S}_{\mathcal{PR}}}[\![\texttt{readLine}]\!]() = \epsilon$$

$$\overline{\mathbb{S}_{\mathcal{PR}}}[\![\texttt{substring}_{\mathsf{b}}^{\mathsf{e}}]\!](\overline{\mathsf{p}}) = \begin{cases} \overline{\mathsf{p}}[\mathsf{b} \cdots \mathsf{e} - 1] & \text{if } \mathsf{e} \leq len(\overline{\mathsf{p}}) \\ \overline{\mathsf{p}}[\mathsf{b} \cdots len(\overline{\mathsf{p}}) - 1] & \text{if } \mathsf{e} > len(\overline{\mathsf{p}}) \wedge \mathsf{b} < len(\overline{\mathsf{p}}) \\ \epsilon & \text{otherwise} \end{cases}$$

$$\overline{\mathbb{B}_{\mathcal{PR}}}[\![\texttt{contains}_{\mathsf{c}}]\!](\overline{\mathsf{p}}) = \begin{cases} \mathsf{true} \text{ if } \mathsf{c} \in char(\overline{\mathsf{p}}) \\ \top_{\mathsf{B}} \quad \text{otherwise} \end{cases}$$

Suffix - $\mathcal{SF}$

- String = Sequence of characters which *ends* with a certain sequence of characters.

$$\overline{\mathcal{SU}} = \mathsf{K}^* \cup \bot_{\overline{\mathcal{SU}}}$$

- The Suffix abstract domain is nearly analogous to the Prefix abstraction
- Partial Order:

$$\overline{\mathsf{S}} \leq_{\overline{\mathcal{SU}}} \overline{\mathsf{T}} \Leftrightarrow \overline{\mathsf{S}} = \bot_{\overline{\mathcal{SU}}} \vee (\forall \mathrm{i} \in [0, len(\overline{\mathsf{T}}) - 1] : len(\overline{\mathsf{T}}) \leq len(\overline{\mathsf{S}}) \wedge \overline{\mathsf{T}}[\mathrm{i}] = \overline{\mathsf{S}}[\mathrm{i} + len(\overline{\mathsf{S}}) - len(\overline{\mathsf{T}})])$$

Suffix (Cont.)

Least Upper Bound:

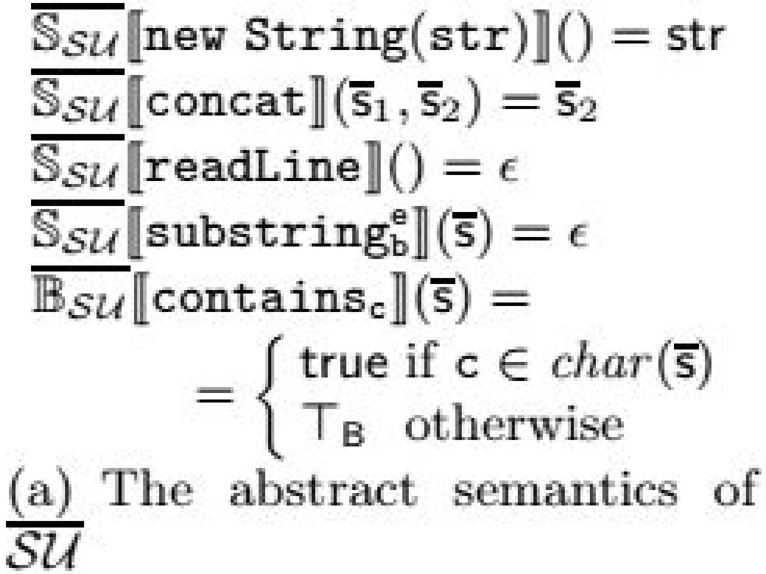
$\sqcup_{SU}$ (S_1, S_2)= Longest common suffix between two strings.

Greatest Lower Bound:

$\sqcap_{SU}$ (S_1, S_2) = Smallest suffix if they are comparable

$\perp_{SU}$ if they are not comparable

Semantics of $\mathcal{SU}$



$$\overline{\mathbb{S}_{\mathcal{SU}}}[\![\texttt{new String(str)}]\!]() = \mathsf{str}$$
$$\overline{\mathbb{S}_{\mathcal{SU}}}[\![\texttt{concat}]\!](\bar{\mathsf{s}}_1, \bar{\mathsf{s}}_2) = \bar{\mathsf{s}}_2$$
$$\overline{\mathbb{S}_{\mathcal{SU}}}[\![\texttt{readLine}]\!]() = \epsilon$$
$$\overline{\mathbb{S}_{\mathcal{SU}}}[\![\texttt{substring}^{\mathsf{e}}_{\mathsf{b}}]\!](\bar{\mathsf{s}}) = \epsilon$$
$$\overline{\mathbb{B}_{\mathcal{SU}}}[\![\texttt{contains}_{\mathsf{c}}]\!](\bar{\mathsf{s}}) = \\ = \begin{cases} \mathsf{true\ if\ c} \in \mathit{char}(\bar{\mathsf{s}}) \\ \top_{\mathsf{B}} \quad \text{otherwise} \end{cases}$$

(a) The abstract semantics of $\overline{\mathcal{SU}}$

Bricks - ***BR***

Significantly, Bricks capture both *inclusion* and *order*

An example brick:

Representing strings with bricks:

Bricks - ***BR*** - Definition and partial order

***Definition*: BR =** *B^* , that is, the set of all finite sequences composed of bricks*

Partial order between single bricks:

Bricks - ***BR*** - Least upper bound

***<u>Definition</u>*: BR =** *B^* , that is, the set of all finite sequences composed of bricks*

LUB between single bricks:

The LUB is the union of each brick's set of strings, and the union of their repetition intervals.

Bricks - ***BR*** - Least upper bound example

$L_1 = [\mathit{star}, \mathit{grape}]^{0,1}[\mathit{fruit}]^{0,1}$ $\quad$ $L_2 = [\mathit{grape}]^{1,1}\ [\mathit{tomato}]^{0,1}$

$[\mathit{star}, \mathit{grape}]^{0,1}, [\mathit{grape}]^{1,1} \qquad [\mathit{fruit}]^{0,1}, \ [\mathit{tomato}]^{0,1})$

$= \quad [\mathit{star}, \mathit{grape}]^{0,1}[\mathit{fruit}, \mathit{tomato}]^{0,1}$

Derives ε, *“starfruit”, “grapefruit”, “grapetomato”, “startomato”, and each singleton string*

Bricks - ***BR*** - Widening operator

The widening operator: Given *n* = max(*len*(L_1), *len*(L_2)) , define constants k_L, k_S, k_I

$$\nabla_{\overline{\mathcal{BR}}}(\overline{\mathsf{L}}_1, \overline{\mathsf{L}}_2) = \begin{cases} \top_{\overline{\mathcal{BR}}} & \text{if } (\overline{\mathsf{L}}_1 \not\leq_{\overline{\mathcal{BR}}} \overline{\mathsf{L}}_2 \wedge \overline{\mathsf{L}}_2 \not\leq_{\overline{\mathcal{BR}}} \overline{\mathsf{L}}_1) \vee \\ & \qquad (\exists \mathsf{i} \in [1,2] : len(\overline{\mathsf{L}}_i) > \mathsf{k}_L) \\ w(\overline{\mathsf{L}}_1, \overline{\mathsf{L}}_2) & \text{otherwise} \end{cases}$$

$$\text{where } \ w(\overline{\mathsf{L}}_1, \overline{\mathsf{L}}_2) \ = \ [\overline{\mathcal{B}}_1^{\mathsf{new}}(\overline{\mathsf{L}}_1[1], \overline{\mathsf{L}}_2[1]); \overline{\mathcal{B}}_2^{\mathsf{new}}(\overline{\mathsf{L}}_1[2], \overline{\mathsf{L}}_2[2]); \ldots; \overline{\mathcal{B}}_n^{\mathsf{new}}(\overline{\mathsf{L}}_1[\mathsf{n}], \overline{\mathsf{L}}_2[\mathsf{n}])],$$

$$\overline{\mathcal{B}}_i^{\mathsf{new}}([\overline{\mathsf{S}}_{1i}]^{\mathsf{m}_{1i},\mathsf{M}_{1i}}, [\overline{\mathsf{S}}_{2i}]^{\mathsf{m}_{2i},\mathsf{M}_{2i}}) = \begin{cases} \top_{\overline{\mathcal{B}}} & \text{if } |\overline{\mathsf{S}}_{1i} \cup \overline{\mathsf{S}}_{2i}| > \mathsf{k}_S \\ & \qquad \vee \overline{\mathsf{L}}_1[\mathsf{i}] = \top_{\overline{\mathcal{B}}} \vee \overline{\mathsf{L}}_2[\mathsf{i}] = \top_{\overline{\mathcal{B}}} \\ [\overline{\mathsf{S}}_{1i} \cup \overline{\mathsf{S}}_{2i}]^{(0,\infty)} & \text{if } (\mathsf{M} - \mathsf{m}) > \mathsf{k}_I \\ [\overline{\mathsf{S}}_{1i} \cup \overline{\mathsf{S}}_{2i}]^{(\mathsf{m},\mathsf{M})} & \text{otherwise} \end{cases}$$

Bricks - *BR* - Semantics

Table 5. The abstract semantics of $\overline{\mathcal{BR}}$

$$\overline{\mathbb{S}_{\mathcal{BR}}}[\![\texttt{new String(str)}]\!]() = [\{\texttt{str}\}]^{1,1}$$

$$\overline{\mathbb{S}_{\mathcal{BR}}}[\![\texttt{concat}]\!](\overline{\mathsf{b}}_1, \overline{\mathsf{b}}_2) = \overline{normBricks}(\overline{concatList}(\overline{\mathsf{b}}_1, \overline{\mathsf{b}}_2))$$

$$\overline{\mathbb{S}_{\mathcal{BR}}}[\![\texttt{readLine}]\!]() = \top_{\overline{\mathcal{BR}}}$$

$$\overline{\mathbb{S}_{\mathcal{BR}}}[\![\texttt{substring}^{\mathsf{e}}_{\mathsf{b}}]\!](\overline{\mathsf{b}}) = \begin{cases} [\overline{\mathsf{T}}']^{1,1} & \text{if } \overline{\mathsf{b}}[0] = [\overline{\mathsf{T}}]^{1,1} \land \forall \overline{\mathsf{t}} \in \overline{\mathsf{T}} : len(\overline{\mathsf{t}}) \geq \mathsf{e} \\ \top_{\overline{\mathcal{BR}}} & \text{otherwise} \end{cases}$$

$$\boxed{\overline{\mathsf{T}}' = \{\overline{\mathsf{t}}.\texttt{substring}(\mathsf{b}, \mathsf{e}) \forall \overline{\mathsf{t}} \in \overline{\mathsf{T}}\}}$$

$$\overline{\mathbb{B}_{\mathcal{BR}}}[\![\texttt{contains}_{\mathsf{c}}]\!](\overline{\mathsf{b}}) = \begin{cases} \mathsf{true} \text{ if } \exists \overline{\mathsf{B}} \in \overline{\mathsf{b}} : \overline{\mathsf{B}} = [\overline{\mathsf{T}}]^{m,M} \land 1 \leq m \leq M \land (\forall \overline{\mathsf{t}} \in \overline{\mathsf{T}} : \mathsf{c} \in char(\overline{\mathsf{t}})) \\ \mathsf{false} \text{ if } \forall [\overline{\mathsf{T}}]^{\mathsf{m},\mathsf{M}} \in \overline{\mathsf{b}}, \forall \overline{\mathsf{t}} \in \overline{\mathsf{T}} : \mathsf{c} \notin char(\overline{\mathsf{t}}) \\ \top_{\mathsf{B}} \quad \text{otherwise} \end{cases}$$

Type Graphs

- A type graph T is triplet (N, A_F, A_B) where (N, A_F) is a rooted tree whose arcs in A_F are called forward arcs, and A_B is a restricted class of arcs, backward arcs, superimposed on (N, A_F).
- Suitable for representing a set of terms
- A node n ∈ N can can be in one of three classes:
  a. Simple
  b. Functor
  c. OR
- n/i denotes the i-th son of node n, and the set of sons of a node n is then denoted as {n/1,..., n/k}

String Graphs - $\mathcal{SG}$

- Adaptation of a Type Graph to strings
- Differences:
  a. Simple nodes have labels from the set {max, ⊥, ϵ} ∪ K
  b. The only functor is *concat*
- $\mathcal{SG}$ = $\mathcal{NSG}$, where $\mathcal{NSG}$ is the set of all Normal String Graphs.
- $\perp_{\mathcal{SG}}$ = A string graph made by one bottom node
- $T_{\mathcal{SG}}$ = A string graph made by only one node, a **max**-node
- Partial order:

$$\overline{\mathsf{T}}_1 \leq_{\overline{\mathcal{SG}}} \overline{\mathsf{T}}_2 \Leftrightarrow \overline{\mathsf{T}}_1 = \perp_{\overline{\mathcal{SG}}} \vee (\leq (\overline{\mathsf{n}}_0, \overline{\mathsf{m}}_0, \emptyset) : \overline{\mathsf{n}}_0 = \overline{root}(\overline{\mathsf{T}}_1) \wedge \overline{\mathsf{m}}_0 = \overline{root}(\overline{\mathsf{T}}_2))$$

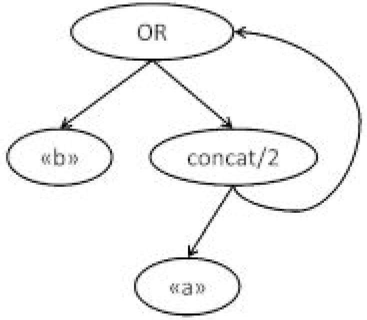


Fig. 5. An example of string graph

String Graphs - $\mathcal{SG}$ (Cont.)

Least Upper Bound:

$$\sqcup_{SG}(T_1, T_2) = \mathit{normStringGraph}(\ OR(T_1, T_2)\ )$$

Semantics of $\mathcal{SG}$

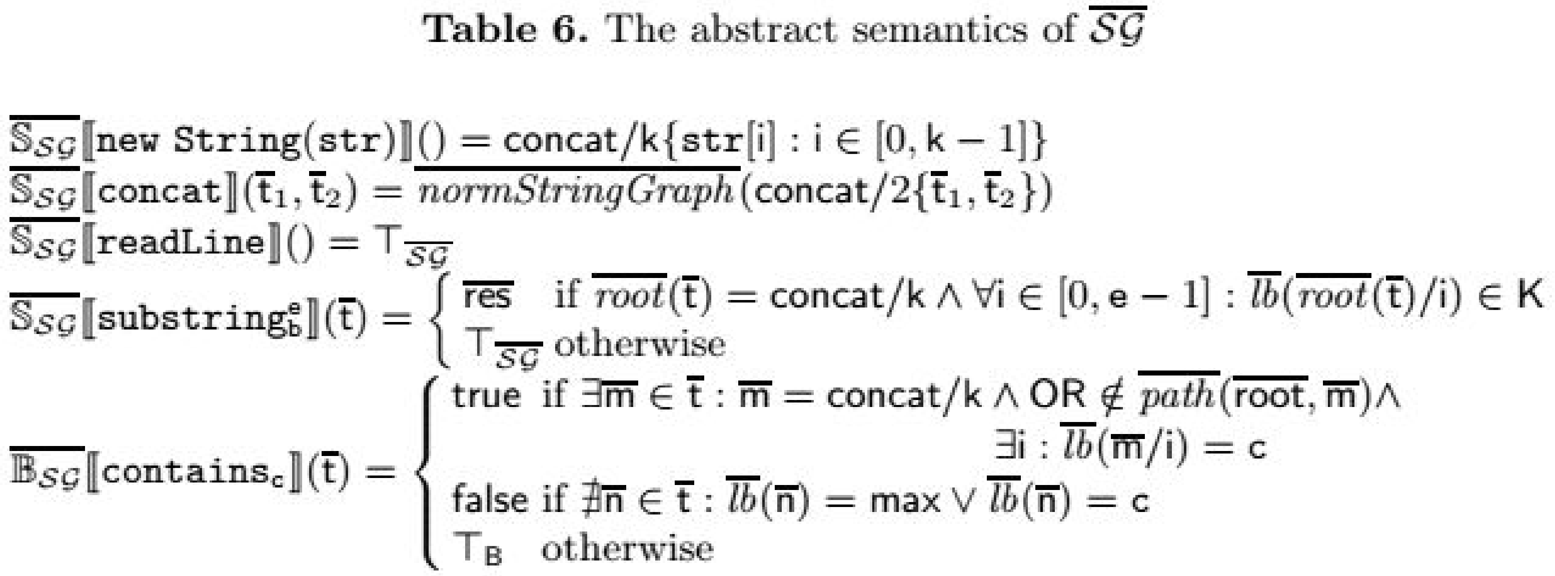


Table 6. The abstract semantics of $\overline{\mathcal{SG}}$

$$\overline{\mathbb{S}_{\mathcal{SG}}}[\![\texttt{new String(str)}]\!]() = \mathsf{concat/k}\{\texttt{str}[\mathsf{i}] : \mathsf{i} \in [0, \mathsf{k} - 1]\}$$

$$\overline{\mathbb{S}_{\mathcal{SG}}}[\![\texttt{concat}]\!](\bar{\mathsf{t}}_1, \bar{\mathsf{t}}_2) = \overline{normStringGraph}(\mathsf{concat}/2\{\bar{\mathsf{t}}_1, \bar{\mathsf{t}}_2\})$$

$$\overline{\mathbb{S}_{\mathcal{SG}}}[\![\texttt{readLine}]\!]() = \top_{\overline{\mathcal{SG}}}$$

$$\overline{\mathbb{S}_{\mathcal{SG}}}[\![\texttt{substring}^{\mathsf{e}}_{\mathsf{b}}]\!](\bar{\mathsf{t}}) = \begin{cases} \overline{\mathsf{res}} & \text{if } \overline{root}(\bar{\mathsf{t}}) = \mathsf{concat/k} \wedge \forall \mathsf{i} \in [0, \mathsf{e} - 1] : \overline{lb}(\overline{root}(\bar{\mathsf{t}})/\mathsf{i}) \in \mathsf{K} \\ \top_{\overline{\mathcal{SG}}} & \text{otherwise} \end{cases}$$

$$\overline{\mathbb{B}_{\mathcal{SG}}}[\![\texttt{contains}_{\mathsf{c}}]\!](\bar{\mathsf{t}}) = \begin{cases} \mathsf{true} \;\; \text{if } \exists \bar{\mathsf{m}} \in \bar{\mathsf{t}} : \bar{\mathsf{m}} = \mathsf{concat/k} \wedge \mathsf{OR} \notin \overline{path}(\overline{\mathsf{root}}, \bar{\mathsf{m}}) \wedge \\ \qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad \exists \mathsf{i} : \overline{lb}(\bar{\mathsf{m}}/\mathsf{i}) = \mathsf{c} \\ \mathsf{false} \; \text{if } \nexists \bar{\mathsf{n}} \in \bar{\mathsf{t}} : \overline{lb}(\bar{\mathsf{n}}) = \mathsf{max} \vee \overline{lb}(\bar{\mathsf{n}}) = \mathsf{c} \\ \top_{\mathsf{B}} \quad \text{otherwise} \end{cases}$$

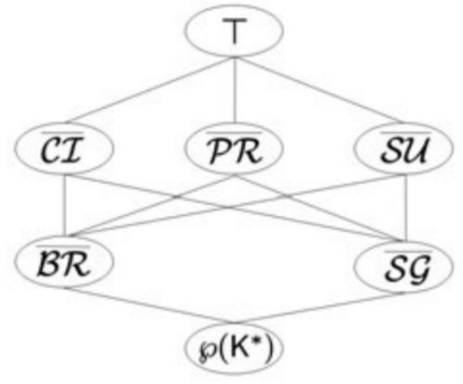
Conclusion

- Two axes of precision in string value analyzers:
  - Character containment in a string
  - Position in the string
- Character inclusion ($\mathcal{CI}$)
  - Considers character containment
  - Discards the order
- Prefix ($\mathcal{PR}$) and Suffix ($\mathcal{SU}$)
  - Collect only partial information about character containment
  - Consider order only in the initial/final segment of the string

Conclusion (Cont.)

- Bricks ($\mathcal{BR}$)
  - Considers character containment
  - Considers order inside the string
- String Graph ($\mathcal{SG}$)
  - Considers character containment
  - Considers order inside the string

So $\mathcal{BR}$ and $\mathcal{SG}$ seem to be the most precise.



T
$\overline{\mathcal{CI}}$
$\overline{\mathcal{PR}}$
$\overline{\mathcal{SU}}$
$\overline{\mathcal{BR}}$
$\overline{\mathcal{SG}}$
$\wp(K^*)$

Fig. 7. The hierarchy of abstract domains

Reference