CFL Reduce to walk

Monday, July 11, 2022 9:58 PM

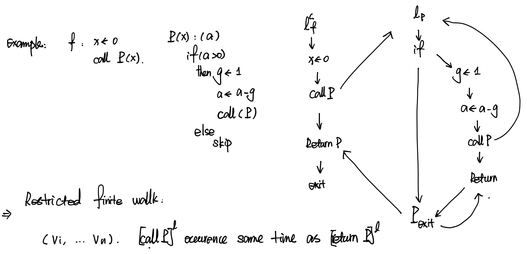
CFL-Reachability : labels of edges composing into a matched word

label matching can be reduced to restrict the occurence times equality

⇒ higher efficient, & symbolic occurence time.

for example:

① interprocedure call can be reduced to valid walk by



⇒ Restricted finite walk:

$(v_i, \dots v_n)$. [call P] occurence same time as [return P]

can generalize to.

② Program slicing Problem:

Example Application Ω:

program slicing Problem:

Bookword / Forward slicing:

⇒ by define as a single-target L(slice)-problem:

where L defined as:

unbalanced-right := unbalanced-right matched

| unbalanced-right); $1 \le i \le$ callsite. ⇒ ✓

| ε

slice := unbalanced-right realizable

⇒ by restricting inequality between occurence times of vertices.

③. Data flow

standard Data reachbility definition analysis is a speeial case of valid walk.

by set the vertice kill x. weight 0

vertice gen x weight 1.

⇒ valid walk = feasible path.

④. IFDS: (Interprocedural, Finite, Distributive, Subset problems)

⇒ : 1. supergraph G^*

2. $\mathbb{D}$: domain of dataflow facts ⇝ $\Lambda \in \mathbb{D}$. the initial data-fact.

Each program point ⇝ exploded to a memember ⇝ $\in 2^{\mathbb{D}}$.

3. Assign of distributive dataflow functions : $2^{\mathbb{D}} \to 2^{\mathbb{D}}$, to edges of G^*.

⇒ Solve it as realizable-path reachability problem.

⇒ exploded supergraph : $G^\#$. for each node in G^*:

⇒ $G^\#(n) = \{ V_{G^\#(n)}, E_{G^\#(n)} \}$.

$d \in V_{G^\#(n)}$: $d \in \mathbb{D}$: a dataflow fact.

$e = (d_1, d_2)$. $d_1 \in V_{G^\#(n_1)}$

$d_2 \in V_{G^\#(n_2)}$.

dataflow facts d_2 in node n_2 if d_1 is true in n_1.

↓ ↓ : the fact transformation function from n_1 to n_2

... d_2 in node n_2 then $f(d_1) = d_2$ is true in n_2

⇒ Formally $G^\#$:

$V = \forall$ node $n \in G^*$, $\exists$ a node $\langle n, \Lambda \rangle$ in $G^\#$.

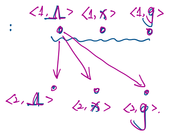
$\cup$ $\forall n \in G^*$. $d \in \mathbb{D}$. $\exists$ a node $\langle n, d \rangle$ in $G^\#$.

E: $\forall$ = edge $e = (\langle m, \Lambda \rangle, \langle n, d \rangle)$ $\forall d \in f(\phi)$.

$\cup$ edge $e = (\langle m, d_1 \rangle, \langle n, d_2 \rangle)$ $\forall d_1, d_2 \in \mathbb{D}$. $d_2 \in f(\{d_1\}) \wedge d_2 \notin f(\phi)$

$\cup$ edge $e = (\langle m, \Lambda \rangle, \langle n, \Lambda \rangle)$.

Example :



data flow facts : $\Lambda \cup \{x, g\}$: • x : x is the possible uninitialized variable.

$\langle 1, x \rangle$: the node for start point in program where data-flow-fact x is true.

fact transformation function f : $\lambda S. \{x, g\}$

i.e. $f(\phi) = \{x, g\}$ ⇒ edges : $(\langle \Lambda, x \rangle, \langle \Lambda, y \rangle, \langle \Lambda, \Lambda$

⇒ composition of edges. ⇒ pasting nodes.