Baumkonstruktion

- Knoten: Input
  - Input-Datei
  - node trace: Ausführungspfad, der vom Knoten selbst abgedeckt wird (Sammlung aller Branches)
  - tree trace: Sammlung der Ausführungspfade, Vereinigung node traces der Kinderknoten mit Teilbäumen.
  - N: Anzahl wie oft dieses Input (oder Kinder) als Seed ausgewählt wurde.
- Kanten: Eltern-Kind-Beziehung
- root node: Elternteil der Anfangsinputs. Leere Input-Datei und leere node trace.

Baumkonstruktion

- Test des Programms mit allen Anfangsinputs, um zugehörige node trace zu erhalten.
- Eintrag der Vereinigung der node traces der Anfangsinputs als tree trace bei root node und notieren, wie oft jeder Zweig abgedeckt ist.
- Nach Erzeugung eines neuen Inputs der neuen Pfad findet, Aktualisierung des Baumes:
  - Input wird als Kind zum zugehörigen Elternknoten hinzugefügt.
  - Node trace des neuen Knotens wird zum tree trace der Elternknoten hinzugefügt und Häufigkeit der Abdeckung der Zweige aktualisiert.

Seed-Auswahlphase

- Beginn der Suche des nächsten Inputs bei root node.
- **tmp node** zeigt auf den aktuell ausgewählten Knoten.
- Berechnungsmethode nach UCT-Algorithmus:

$$SeedScore = \frac{SV_i}{n_i} + c\sqrt{\frac{\ln(N_i)}{n_i}}$$

- Berechunung von SV_i:
  - ☐ Identifikation der am wenigsten abgedeckten Zweige in tree trace von tmp node.
  - ☐ Anzahl der Zweige zählen, die in tree trace von Kind i enthalten sind.
- Bei gleichem höchstem SeedScore, Wahl des ersten der beiden.
- Auswahlphase endet, wenn tmp node ein Blatt ist.

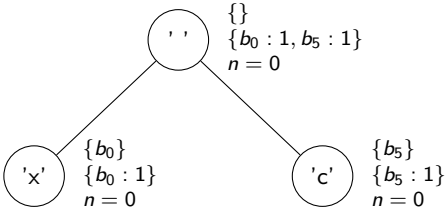
Seed-Mutationsphase

- Ausführung einer Serie von zufälligen Mutationsstrategien der Seed-Datei, zur Erzeugung neuer Testfälle.
- Test des Zielprogramms mit neu erzeugtem Input.
- Input interessant, wenn ein neuer Pfad abgedeckt wird.
- Interessante Inputs werden als Kinderknoten zur Baumstruktur hinzugefügt.

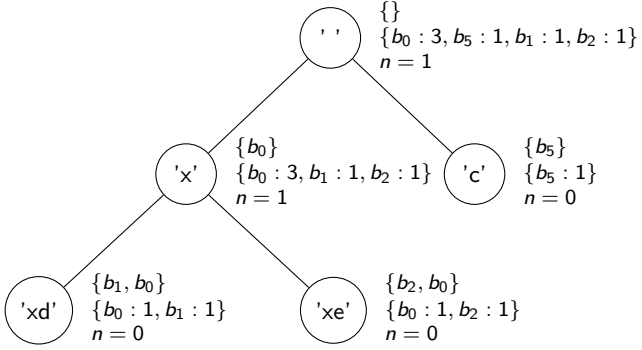
Backpropagation-Phase

- Aktualisierung der tree trace-Information der Elternknoten bis zur root node.
- Bei jedem Elternteil, Erhöhung von N um eins
  N: Anzahl, wie oft ein Input als Seed ausgewählt wurde

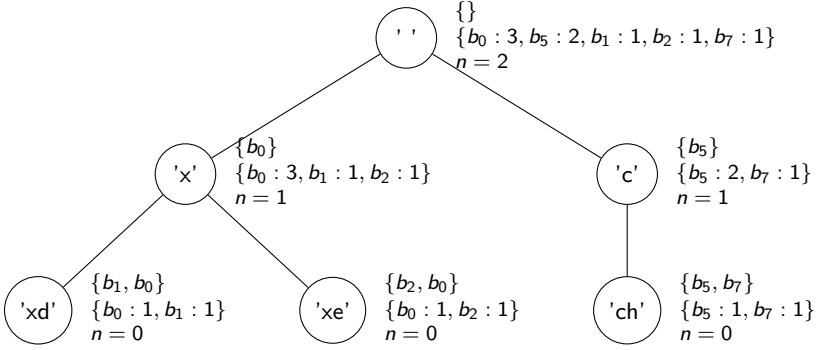
Beispiel



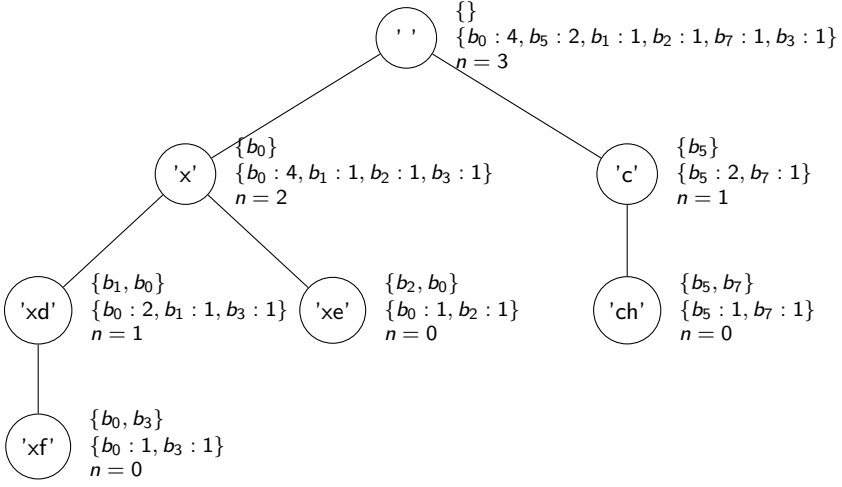
Beispiel


' '
{}
$\{b_0 : 3, b_5 : 1, b_1 : 1, b_2 : 1\}$
$n = 1$
'x'
$\{b_0\}$
$\{b_0 : 3, b_1 : 1, b_2 : 1\}$
$n = 1$
'c'
$\{b_5\}$
$\{b_5 : 1\}$
$n = 0$
'xd'
$\{b_1, b_0\}$
$\{b_0 : 1, b_1 : 1\}$
$n = 0$
'xe'
$\{b_2, b_0\}$
$\{b_0 : 1, b_2 : 1\}$
$n = 0$

Beispiel


' '
{}
$\{b_0 : 3, b_5 : 2, b_1 : 1, b_2 : 1, b_7 : 1\}$
$n = 2$
'x'
$\{b_0\}$
$\{b_0 : 3, b_1 : 1, b_2 : 1\}$
$n = 1$
'c'
$\{b_5\}$
$\{b_5 : 2, b_7 : 1\}$
$n = 1$
'xd'
$\{b_1, b_0\}$
$\{b_0 : 1, b_1 : 1\}$
$n = 0$
'xe'
$\{b_2, b_0\}$
$\{b_0 : 1, b_2 : 1\}$
$n = 0$
'ch'
$\{b_5, b_7\}$
$\{b_5 : 1, b_7 : 1\}$
$n = 0$

Beispiel


' '
{}
$\{b_0 : 4, b_5 : 2, b_1 : 1, b_2 : 1, b_7 : 1, b_3 : 1\}$
$n = 3$
'x'
$\{b_0\}$
$\{b_0 : 4, b_1 : 1, b_2 : 1, b_3 : 1\}$
$n = 2$
'c'
$\{b_5\}$
$\{b_5 : 2, b_7 : 1\}$
$n = 1$
'xd'
$\{b_1, b_0\}$
$\{b_0 : 2, b_1 : 1, b_3 : 1\}$
$n = 1$
'xe'
$\{b_2, b_0\}$
$\{b_0 : 1, b_2 : 1\}$
$n = 0$
'ch'
$\{b_5, b_7\}$
$\{b_5 : 1, b_7 : 1\}$
$n = 0$
'xf'
$\{b_0, b_3\}$
$\{b_0 : 1, b_3 : 1\}$
$n = 0$

Pseudocode

1: **for** `seed` $\in$ `seedfiles` **do**
2: `root.children.add(seed)`
3: `root.tree_trace` $= \biguplus_{\texttt{child} \in \texttt{root.children}}$ `child.node_trace`

Pseudocode

1: **for** `seed` $\in$ `seedfiles` **do**
2: `root.children.add(seed)`
3: `root.tree_trace` $= \biguplus_{\texttt{child} \in \texttt{root.children}}$ `child.node_trace`
4: **while** *time budget* **do**
5: `best` $\leftarrow$ `node` $\leftarrow$ `root` ▷ tree policy $\rightarrow$ find input to mutate
6: **while** `node` *is inner node* **do**
7: `node` $= \arg\max_{\texttt{nd} \in \textit{node.children}} \mathrm{SCORE}(\texttt{nd})$
8: `best` $= \arg\max_{\texttt{nd} \in \{\texttt{node},\texttt{best}\}} \mathrm{SCORE}(\texttt{nd})$

Pseudocode

```
1: for seed ∈ seedfiles do
2:     root.children.add(seed)
3: root.tree_trace = ⨄_{child∈root.children} child.node_trace
4: while time budget do
5:     best ← node ← root                          ▷ tree policy → find input to mutate
6:     while node is inner node do
7:         node = arg max_{nd∈node.children} SCORE(nd)
8:         best = arg max_{nd∈{node,best}} SCORE(nd)
9:     new_traces = {}                             ▷ simulate
10:    for mutant ∈ MUTATE(best) do
11:        best.children.add(mutant)
12:        new_traces ⨄= mutant.node_trace
```

Pseudocode

```
1: for seed ∈ seedfiles do
2:     root.children.add(seed)
3: root.tree_trace = ⨄_{child∈root.children} child.node_trace
4: while time budget do
5:     best ← node ← root                          ▷ tree policy → find input to mutate
6:     while node is inner node do
7:         node = arg max_{nd∈node.children} SCORE(nd)
8:         best = arg max_{nd∈{node,best}} SCORE(nd)
9:     new_traces = {}                             ▷ simulate
10:    for mutant ∈ MUTATE(best) do
11:        best.children.add(mutant)
12:        new_traces ⊎= mutant.node_trace
13:    for node ∈ "best and its ancestors" do      ▷ update
14:        node.n+ = 1
15:        node.tree_trace ⊎= new_traces
```

Pseudocode

procedure SCORE(node)
  if `node` $=$ `root` **then**
    return -∞
  $lcbs =$ `rare_branches(parent.tree_trace)` ▷ $lcbs(A, B, B, C) = \{A, C\}$
  return $\frac{|node.node\_trace \cap lcbs|}{n_i} + \sqrt{c \frac{\ln(\texttt{parent.n})}{n_i}}$ ▷ UCT equation

Pseudocode

procedure SCORE(node)
  if `node` $=$ `root` **then**
    return -∞
  $lcbs =$ `rare_branches(parent.tree_trace)` ▷ $lcbs(A, B, B, C) = \{A, C\}$
  return $\frac{|node.node\_trace \cap lcbs|}{n_i} + \sqrt{c \frac{\ln(\texttt{parent.n})}{n_i}}$ ▷ UCT equation

procedure MUTATE(node)
  repeat
    create mutant
    `mutant.n` $= 0$
    `mutant.node_trace` $=$ EXECUTE_ON_PUT(`mutant`)
    `mutant.tree_trace` $=$ `mutant.node_trace`
    if $mutant.node\_trace \cap set(root.tree\_trace) \neq \emptyset$ **then**: ▷ discovered new branch
      yield `mutant`
  until *sufficient mutants*