Adjusted Algorithm

Jannik Rau

December 2021

Nesting Problem

- The original algorithm operated directly on vertex summaries.
- Neighbours attribute in VS contain VS of neighbours, which potentially contains VS of their neighbours, and so forth.

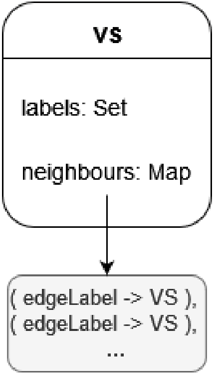

Figure: Vertex Summary

Concept of Schätzle

- Compute Forward k-bisimulation:
  - $u \approx_{fw}^{0} v$ for all $u, v \in V$,
  - $u \approx_{fw}^{i+1} v$ for $i \geq 0$ iff for every successor u' of u there exists a successor v' of v with $l_E((u, u')) = l_E((v, v'))$ and $u' \approx_{fw}^{i} v'$.
- Equality between two vertices is determined by an *ID* value.
- Initially, for every vertex v, $v.ID = 0$.
- In every iteration a vertex receives $(p, w.ID)$ of neighbours w.
- Then a signature $sig(v) = \{(p_1, ID_1), (p_2, ID_2), \ldots\}$ is constructed.
- New *ID* value is computed by hashing the signature.

Concept of Adjusted Algorithm (1/6)

- Compute $CSE^k = (\sim_s, \sim_p, (\sim_s, \sim_p, \dots (\sim_s, \sim_p, \sim_o) \dots))$.
  - $u \approx^1 v$ iff
    - $u \sim_s v$,
    - $\forall (u, u') \in E \; \exists (v, v') \in E$ with $l_E((u, u')) \sim_p l_E((v, v'))$ and $u' \sim_o v'$ and vice versa.
  - $u \approx^{i+1} v$ for $i > 0$ iff
    - $u \sim_s v$,
    - $\forall (u, u') \in E \; \exists (v, v') \in E$ with $l_E((u, u')) \sim_p l_E((v, v'))$ and $u' \approx^i v'$ and vice versa.
- At the end two vertices u, v are considered equal iff $u \approx^k v$.

Concept of Adjusted Algorithm (2/6)

- Equality is determined by an $id_{\sim_s}$ value.
- Additional $id_{\sim_o}$ value is used to compute $id_{\sim_s}$ value.
- In the initialisation step, $id_{\sim_s}$ and $id_{\sim_o}$ is computed for every vertex by constructing respective VS and deriving a numerical value from it.

---

forall $v \in V$ **do in parallel**

  $vs_{\sim_s} \leftarrow \textsc{VertexSchema}(v, E, \sim_s)$;

  $vs_{\sim_o} \leftarrow \textsc{VertexSchema}(v, E, \sim_o)$;

  $v.id_{\sim_s} \leftarrow vs_{\sim_s}.ID$;

  $v.id_{\sim_o} \leftarrow vs_{\sim_o}.ID$;

Concept of Adjusted Algorithm (3/6)

- If k is equal to one, following procedure is executed

```
forall v ∈ V do in parallel
    SignalMessages(t_w, v.id_~o);
    arr_{id_~o} ← MergeMessages((t_w, id_~o));
    v.id_~s ← Hash(arr_{id_~o});
```

Concept of Adjusted Algorithm (4/6)

- If k is greater than one, we first signal initial messages

/* Signal initial messages. Update $v.id_{\sim_s}$ and $v.id_{\sim_o}$ */

forall $v \in V$ **do in parallel**

$\quad \textsc{SignalMessages}(t_w, v.id_{\sim_s}, v.id_{\sim_o})$;

$\quad arr_{id_{\sim_s}} \leftarrow \textsc{MergeMessages}((t_w, id_{\sim_s}))$;

$\quad arr_{id_{\sim_o}} \leftarrow \textsc{MergeMessages}((t_w, id_{\sim_o}))$;

$\quad v.id_{\sim_s} \leftarrow \textsc{Hash}(arr_{id_{\sim_s}})$;

$\quad v.id_{\sim_o} \leftarrow \textsc{Hash}(arr_{id_{\sim_o}})$;

Concept of Adjusted Algorithm (5/6)

- Then we signal $k - 2$ times
- Object value gets only 'forwarded'

/* Signal $k-2$ times. Do not include t_w when updating $v.id_{\sim_o}$. */

for $i = 2$ *to* $k - 1$ **do**

    forall $v \in V$ **do in parallel**

        $\textsc{SignalMessages}(t_w, v.id_{\sim_s}, v.id_{\sim_o})$;

        $arr_{id_{\sim_s}} \leftarrow \textsc{MergeMessages}((t_w, id_{\sim_s}))$;

        $arr_{id_{\sim_o}} \leftarrow \textsc{MergeMessages}(id_{\sim_o})$;

        $v.id_{\sim_s} \leftarrow \textsc{Hash}(arr_{id_{\sim_s}})$;

        $v.id_{\sim_o} \leftarrow \textsc{Hash}(arr_{id_{\sim_o}})$;

Concept of Adjusted Algorithm (6/6)

- Signal final messages containing the 'forwarded' $id_{\sim_o}$ values

```
/* Signal final messages.  Update v.id~s */
forall v ∈ V do in parallel
    SignalMessages(v.id~o);
    arr_id~o ← MergeMessages(id~o);
    v.id~s ← Hash(arr_id~o);
```

Example

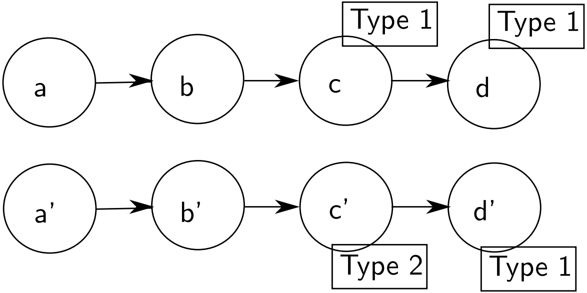

Figure: Example Graph

- $CSE = (T, T, OC_{Type})$
- Initialisation
  - $id_{\sim_s} = 0, \ \forall v \in V$
  - $id_{\sim_o} = 0$ for a, a', b, b' $\quad id_{\sim_o} = 1$ for c, d, d' $\quad id_{\sim_o} = 2$ for c'

Example K = 1

Received messages and resulting *HASH*

- a, a': $(0,0) \rightarrow id_{\sim_s} = HASH([(self, 0), (0,0)])$
- b': $(0,2) \rightarrow id_{\sim_s} = HASH([(self, 0), (0,2)])$
- b, c, c': $(0,1) \rightarrow id_{\sim_s} = HASH([(self, 0), (0,1)])$
- d, d': No messages $\rightarrow id_{\sim_s} = 0$ (=Initial Value)

Equivalence Classes

- $\{a, a'\}$
- $\{b'\}$
- $\{b, c, c'\}$
- $\{d, d'\}$

Example K = 2 Iteration 1

Received messages for $id_{\sim_s}$ and resulting HASH

- $\forall v \in V - \{d, d'\}$: $(0,0) \rightarrow id_{\sim_s} = HASH([(self, 0), (0,0)])$

Received messages for $id_{\sim_o}$ and resulting HASH

- a, a': $(0,0) \rightarrow id_{\sim_o} = HASH([(0,0)])$
- b': $(0,2) \rightarrow id_{\sim_o} = HASH([(0,2)])$
- b, c, c': $(0,1) \rightarrow id_{\sim_o} = HASH([(0,1)])$

Example K = 2 Iteration 2

Received messages and resulting HASH

- a, b, b': $HASH([(0,1)]) \rightarrow id_{\sim_s} = HASH([HASH([(self, 0), (0,0)]), HASH([(0,1)])])$
- a': $HASH([(0,2)]) \rightarrow id_{\sim_s} = HASH([HASH([(self, 0), (0,0)]), HASH([(0,2)])])$
- c, c': $1 \rightarrow id_{\sim_s} =$ **TODO: Here there is some typo**$HASH([HASH([(self, 0), (0,0)]),$
- d, d': No messages $\rightarrow id_{\sim_s} = 0$ (=Initial Value)

Equivalence Classes

- $\{a, b, b'\}$
- $\{a'\}$
- $\{c, c'\}$
- $\{d, d'\}$

Example K = 3 Iteration 1

Received messages for $id_{\sim_s}$ and resulting HASH

- $\forall v \in V - \{d, d'\}$: $(0,0) \rightarrow id_{\sim_s} = HASH([(self, 0), (0,0)])$

Received messages for $id_{\sim_o}$ and resulting HASH

- a, a': $(0,0) \rightarrow id_{\sim_o} = HASH([(0,0)])$
- b': $(0,2) \rightarrow id_{\sim_o} = HASH([(0,2)])$
- b, c, c': $(0,1) \rightarrow id_{\sim_o} = HASH([(0,1)])$

Example K = 3 Iteration 2

Received messages for $id_{\sim_s}$ and resulting HASH

- a, a', b, b': $(0, HASH([(self, 0), (0, 0)])) \rightarrow id_{\sim_s} = HASH([(self, HASH([(self, 0), (0, 0)])), (0, HASH([(self, 0), (0, 0)]))])$
- c, c': $(0, 0) \rightarrow id_{\sim_s} = HASH([(self, HASH([(self, 0), (0, 0)])), (0, 0)])$

Received messages for $id_{\sim_o}$ and resulting HASH

- a, b, b': $HASH([(0, 1)]) \rightarrow id_{\sim_o} = HASH([HASH([(0, 1)])])$
- a': $HASH([(0, 2)]) \rightarrow id_{\sim_o} = HASH([HASH([(0, 2)])])$
- c, c': $1 \rightarrow id_{\sim_o} = HASH([1])$

Example K = 3 Iteration 3

Received messages and resulting HASH

$$\text{LET } X = (self, HASH([(self, HASH([(self, 0), (0, 0)])), (0, HASH([(self, 0), (0, 0)])]))$$

$$\text{LET } Y = (self, HASH([(self, HASH([(self, 0), (0, 0)])), (0, 0)])$$

- a, a': $HASH([HASH([(0, 1)])]) \rightarrow id_{\sim_s} = HASH([X, HASH([HASH([(0, 1)])])])$
- b, b': $HASH([1]) \rightarrow id_{\sim_s} = HASH([X, HASH([1])])$
- c, c': $1 \rightarrow id_{\sim_s} = HASH([Y, 1])$
- d, d': No messages $\rightarrow id_{\sim_s} = 0$ (=Initial Value)

Equivalence Classes

- $\{a, a'\}$
- $\{b, b'\}$
- $\{c, c'\}$
- $\{d, d'\}$

Example Schätzle GSM

- CSE = (T, id, T)
- Initialisation
  - $id_{\sim_s} = 0, \forall v \in V$
  - $id_{\sim_o} = 0, \forall v \in V$

Example Schätze GSM k = 1

Received messages and resulting HASH

-