Abstract
This report outlines a proof of the theorem described in COSMOPEN: *Dynamic reverse engineering on a budget, How cheap observation techniques can be used to reconstruct complex multi-level behaviour* [1] on the construction of a minimal call-trees from stack traces. Please refer to original publication for details regarding the context of the work, the notation used, and definitions. We repeat below the pseudo-code of the call-tree construction algorithm, and present some key definitions in more detail. We then outline a proof of the characterisation of the call-tree constructed by the algorithm.

Computing Department
Infolab21, South Drive
Lancaster University
LANCASTER LA1 4WA
United Kingdom
T1.: +44 (0) 1524 51 03 38
mail: francois.taiani@comp.lancs.ac.uk
1 Introduction

This short document outlines a proof of the theorem described in [1] on the construction of a minimal call-trees from stack traces. Please refer to [1] for details regarding the context of the work, the notation used, and definitions. We repeat below the pseudo-code of the call-tree construction algorithm, and present some key definitions in more detail. We then outline a proof of the characterisation of the call-tree constructed by the algorithm.

2 Call-tree construction algorithm

Algorithm 1 Transforming a stack trace sequence into a call-tree

1: \texttt{ActiveTreePath} \leftarrow () ; V \leftarrow \emptyset \\
2: \texttt{traceSequence} = (\texttt{trace}_1, \texttt{trace}_2, \ldots, \texttt{trace}_N) \\
3: \textbf{for} i = 1 \textbf{to} |\texttt{traceSequence}| \textbf{do} \\
4: \hspace{0.5em} \textbf{AddTraceToGraph} (\texttt{traceSequence}[i], V, E, \texttt{ActiveTreePath}) \\
5: \textbf{end for} \\
6: \textbf{procedure} \texttt{AddTraceToGraph} (\texttt{trace}, V, E, \texttt{ActiveTreePath}) \\
7: \hspace{0.5em} \textbf{for} j = 1 \textbf{to} \min(|\texttt{ActiveTreePath}|, |\texttt{trace}| - 1) \textbf{do} \\
8: \hspace{1em} \textbf{if} symbol(\texttt{ActiveTreePath}[j]) \neq \texttt{trace}[j] \textbf{then} \textbf{break} \\
9: \hspace{0.5em} \textbf{end for} \\
10: \hspace{0.5em} \texttt{truncate}(\texttt{ActiveTreePath}, j) \\
11: \texttt{previousNode} \leftarrow \texttt{last}(\texttt{ActiveTreePath}) \\
12: \textbf{for} k = j \textbf{to} |\texttt{trace}| \textbf{do} \\
13: \hspace{0.5em} \texttt{newNode} \leftarrow \texttt{new Node}(\texttt{trace}[k]) \\
14: \hspace{0.5em} \texttt{ActiveTreePath} \leftarrow \texttt{ActiveTreePath} + (\texttt{newNode}) \\
15: \hspace{0.5em} V \leftarrow V \cup \{\texttt{newNode}\} \\
16: \hspace{0.5em} \textbf{if} \texttt{previousNode} \neq \texttt{NIL} \textbf{then} E \leftarrow E \cup \{(\texttt{previousNode}, \texttt{newNode})\} \\
17: \hspace{0.5em} \texttt{previousNode} \leftarrow \texttt{newNode} \\
18: \textbf{end for} \\
19: \textbf{end procedure} \\

In algorithm 1, time stamps (also known as sequence numbers) are allocated by the constructor \texttt{new Node}(\texttt{symbol}) (line 13) in the order in which nodes are created. The same timestamp is never allocated more than once.

3 Definitions and theorem

Call tree A call-tree is a directed tree \((V, E)\) that is associated with two labelling functions: \texttt{symbol} and \texttt{timeStamp}.

\texttt{symbol} associates each node \(x \in V\) of the tree with the name of a function (its \texttt{symbol}). Multiple nodes might be associated with the same symbol.

\texttt{timeStamp} associates each node \(x \in V\) with an integer (its \texttt{time stamp}). Time stamps are unique (no two nodes have the same time stamp) and the time stamp of a child node is strictly greater than that of its parent:

\[
\forall x, y \in V : \quad \texttt{timeStamp}(x) = \texttt{timeStamp}(y) \Rightarrow x = y \quad (1)
\]

\[
\forall x, y \in V : \quad (x, y) \in E \Rightarrow \texttt{timeStamp}(x) < \texttt{timeStamp}(y) \quad (2)
\]

Trace compatibility We say that a call-tree \((V, E)\) is trace-compatible (compatible for short) with a sequence of observed stack traces \texttt{traceSequence} = (\texttt{trace}_1, \texttt{trace}_2, \ldots, \texttt{trace}_N) if and only if there exists a surjective
mapping $\Phi$ from the frames $\langle \text{trace}_i, j \rangle$ of the observed traces onto the nodes of $V$, such that $\Phi$ fulfils the following properties P0, P1, P2 and P3.

**Property P0: Symbol Consistency** Stack frames are mapped onto nodes with the same symbol:

$$\forall \text{trace}_i \in \text{traceSequence}, 1 \leq k \leq |\text{trace}_i| : \text{symbol}(\Phi(\langle \text{trace}_i, k \rangle)) = \text{trace}_i[k]$$

(3)

**Property P1: Trace Inclusion** Each observed trace is mapped by $\Phi$ onto a path of the tree that starts from the tree’s root:

$$\forall t_i \in \text{traceSequence} :$$

$$\Phi(\langle t_i, 1 \rangle) \text{ is the root of } (V, E) \land$$

$$\forall 1 \leq j < |t_i| : \left( \Phi(\langle t_i, j \rangle), \Phi(\langle t_i, j + 1 \rangle) \right) \in E$$

(4)

**Property P2: Breakpoint Discrimination** The last frame of every stack trace (e.g. the frame that corresponds to a breakpoint activation) is mapped to a node that is distinct from those of the stack traces that precede it:

$$\forall t_i, t_j \in \text{traceSequence} : i < j \Rightarrow \forall 1 \leq k \leq |t_i| : \Phi(\langle t_j, |t_j| \rangle) \neq \Phi(\langle t_i, k \rangle)$$

(5)

**Property P3: Order Conservation** The sequence numbers of the nodes of the tree reflect the order in which traces are observed. More precisely: if a trace $\text{trace}_j$ is observed after another trace $\text{trace}_i$, then the tail of $\text{trace}_j$ that does not overlap with $\text{trace}_i$ is mapped onto nodes with higher sequence numbers than the nodes of $\text{trace}_i$.

$$\forall \text{trace}_i, \text{trace}_j \in \text{traceSequence}, \forall k \in \left[1, \min(|\text{trace}_i|, |\text{trace}_j|)\right] :$$

$$(j > i) \land \Phi(\langle \text{trace}_i, k \rangle) \neq \Phi(\langle \text{trace}_j, k \rangle) \Rightarrow$$

$$\forall l \in [k, |\text{trace}_j|], \forall m \in [1, |\text{trace}_i|] :$$

$$\text{timestamp}(\Phi(\langle \text{trace}_j, l \rangle)) > \text{timestamp}(\Phi(\langle \text{trace}_i, m \rangle))$$

(6)

**Homomorphism between call-trees** A call-tree homomorphism $h$ from a call-tree $(V, E)$ onto another calltree $(W, F)$ is a graph homomorphism from $(V, E)$ to $(W, F)$ that fulfils the following additional properties:

$$\forall x \in V : \text{symbol} \left( h(x) \right) = \text{symbol}(x)$$

(7)

$$\forall r, s \in h[V] : \left( \forall x \in h^{-1}\{r\}, \forall y \in h^{-1}\{s\} : \text{timestamp}(x) \leq \text{timestamp}(y) \right)$$

$$\Rightarrow \text{timestamp}(r) \leq \text{timestamp}(s)$$

(8)

where $f[A]$ (resp. $f^{-1}[A]$) denotes the image (resp. the pre-image) of the set $A$ by the function $f$.

A call-tree isomorphism is a bijective call-tree homomorphism. Note that for a call-tree isomorphism (8) is equivalent to

$$\forall x, y \in V : \text{timestamp}(x) \leq \text{timestamp}(y) \Rightarrow \text{timestamp} \left( h(x) \right) \leq \text{timestamp} \left( h(y) \right)$$

(9)

**Theorem 3.1.** The set of call-trees that are trace-compatible with a sequence of observed stack traces is non-empty. The call-tree of this set with the smallest number of nodes is unique (modulo a call-tree isomorphism) and is the call-tree constructed by algorithm 1.

**4 Proof**

Let’s consider a trace sequence $\text{traceSequence} = (\text{trace}_1, \text{trace}_2, ..., \text{trace}_N)$, and let’s call $G_0 = (V_0, E_0)$ the call-tree constructed by algorithm 1 from this trace sequence.

**Lemma 4.1.** $G_0$ is trace compatible with $\text{traceSequence}$. 
Case 1

The proof needs to consider two cases depending whether all \( \tau \) traces between

The proof of property P3 is slightly more technical. In this proof we will use the following sublemma:

\[ \forall \text{trace}_i \in \text{traceSequence}, \forall j \in \left[1, |\text{trace}_i|\right] : \Phi_0\left(\langle \text{trace}_i, j \rangle\right) \equiv \text{ActiveTreePath}_i[j] \tag{10} \]

By construction \( \Phi_0 \) is a function (i.e. it associates each frame \( \langle \text{trace}_i, j \rangle \) with exactly one node of \( G_0 \)), and by construction of algorithm 1, \( \Phi_0 \) is surjective.

Also because of lines 8, 13, and 14 and the constructor property of the \text{new Node}(\text{trace}[k]), we have

\[ \text{symbol}\left(\Phi_0(\langle \text{trace}_i, j \rangle)\right) = \text{symbol}\left(\text{ActiveTreePath}_i[j]\right) = \text{trace}_i[j] \]

and consequently \( \Phi_0 \) fulfills property P0.

Property P1 follows from the creation of new edges at line 16 and the fact that all traces start with the same symbol (the program’s entry point): \( \forall \text{t} \in \text{traceSequence} : \text{t}[1] = \text{trace}_1[1] \).

Property P2 follows from the bound \(|\text{trace}| - 1\) put on the loop that compares \text{ActiveTreePath} with the current trace at line 7. ‘−1’ forces the last frame of the trace (and hence the symbol corresponding to a breakpoint activation) to be included as a new node in the call-tree.

The proof of property P3 is slightly more technical. In this proof we will use the following sublemma:

\textbf{Sublemma 4.1.1.}

\[ \forall \text{trace}_i \in \text{traceSequence}, \forall k, l \in \left[1, |\text{trace}_i|\right] : k < l \Rightarrow \text{timeStamp}\left(\Phi_0(\langle \text{trace}_i, k \rangle)\right) < \text{timeStamp}\left(\Phi_0(\langle \text{trace}_i, l \rangle)\right) \tag{11} \]

**Proof.** This sublemma directly follows from

\[ \forall i \in \left[1, |\text{traceSequence}|\right], \forall k, l \in \left[1, |\text{traceSequence}[i]|\right] : k < l \Rightarrow \text{timeStamp}\left(\text{ActiveTreePath}_i[k]\right) < \text{timeStamp}\left(\text{ActiveTreePath}_i[l]\right) \tag{12} \]

which is itself an invariant of the loop at line 3. This invariant can be proved by recursion using the properties of time stamps and node creation. (The details of this short proof are left out for place reason.) \( \square \)

To prove P3, let’s now assume \( \Phi_0(\langle \text{trace}_i, k \rangle) \neq \Phi_0(\langle \text{trace}_j, k \rangle) \) for some \( i > j \) and \( k \in \left[1, \min(|\text{trace}_i|, |\text{trace}_j|)\right] \).

The proof needs to consider two cases depending whether all traces between \( i \) and \( j \) are at least of length \( k \) or not.

**Case 1** \( \forall r \in [i, j] : |\text{trace}_r| \geq k \)

In this case we can consider the largest \( l \in [i, j - 1] \) such that \( \Phi_0(\langle \text{trace}_l, k \rangle) \neq \Phi_0(\langle \text{trace}_{l+1}, k \rangle) \). Let’s note this value \( l_0 \):

\[ l_0 \equiv \max \left\{ l \in [i, j - 1] \mid \Phi_0(\langle \text{trace}_l, k \rangle) \neq \Phi_0(\langle \text{trace}_{l+1}, k \rangle) \right\} \tag{13} \]

By construction of \( l_0 \) we have

\[ \Phi_0(\langle \text{trace}_{l_0}, k \rangle) \neq \Phi_0(\langle \text{trace}_{l_0+1}, k \rangle) = \Phi_0(\langle \text{trace}_{l_0+2}, k \rangle) = \ldots = \Phi_0(\langle \text{trace}_j, k \rangle) \tag{14} \]

This implies (by definition of \( \Phi_0 \) in equation (10))

\[ \text{ActiveTreePath}_{l_0}[k] \neq \text{ActiveTreePath}_{l_0+1}[k] \tag{15} \]
This again implies that a new node was constructed for \( \text{trace}_{t_{i_0}+1}[k] \) at line 13. Because timestamps are allocated in the order in which nodes are created, this node’s timestamp is higher than those of any previously created nodes, and in particular than the timestamp of \( \text{ActiveTreePath}_i[m] = \Phi_0(\langle \text{trace}_i, m \rangle) \) for all \( m \in [1, \text{trace}_i] \).

\[
\forall m \in [1, \text{trace}_i] : timestamp(\Phi_0(\langle \text{trace}_i, m \rangle)) < timestamp(\Phi_0(\langle \text{trace}_i+1, k \rangle))
\] (16)

From (14) and (16) we thus get (since \( \Phi_0(\langle \text{trace}_i+1, k \rangle) = \Phi_0(\langle \text{trace}_j, k \rangle) \)):

\[
\forall m \in [1, \text{trace}_i] : timestamp(\Phi_0(\langle \text{trace}_i, m \rangle)) < timestamp(\Phi_0(\langle \text{trace}_j, k \rangle))
\] (17)

Combined with sublemma 4.1.1, equation (17) gives

\[
\forall m \in [1, \text{trace}_i], l \in [k, \text{trace}_j] : 
\text{timestamp}(\Phi_0(\langle \text{trace}_i, m \rangle)) < \text{timestamp}(\Phi_0(\langle \text{trace}_j, l \rangle))
\] (18)

which concludes the proof of P3 in this case.

**Case 2** \( \exists r \in [i, j] : |\text{trace}_r| < k \)

In that case we can consider

\[
r_0 \equiv \max \left\{ r \in [i + 1, j - 1] \mid |\text{trace}_r| < k \right\}
\] (19)

By construction, note that we have:

\[
\forall r \in [r_0 + 1, j] : |\text{trace}_r| \geq k
\] (20)

Two subcases follow.

**Case 2.1**

If \( \exists l \in [r_0 + 1, j - 1] : \Phi_0(\langle \text{trace}_l, k \rangle) \neq \Phi_0(\langle \text{trace}_{l+1}, k \rangle) \) then we can define

\[
l_0 \equiv \max \left\{ l \in [r_0 + 1, j - 1] \mid \Phi_0(\langle \text{trace}_l, k \rangle) \neq \Phi_0(\langle \text{trace}_{l+1}, k \rangle) \right\}
\] (21)

and we are back to case 1.

**Case 2.2**

If \( \forall l \in [r_0 + 1, j - 1] : \Phi_0(\langle \text{trace}_l, k \rangle) = \Phi_0(\langle \text{trace}_{l+1}, k \rangle) \) then we note that because \( |\text{trace}_{r_0}| < k \) and \( |\text{trace}_{r_0+1}| \geq k \) then a new node was constructed for \( \text{trace}_{r_0+1}[k] \) at line 13. Because \( \Phi_0(\langle \text{trace}_{r_0+1}, k \rangle) = \Phi_0(\langle \text{trace}_j, k \rangle) \), we are again back to case 1.

This concludes the proof of P3, and more generally the proof of lemma 4.1.

**Lemma 4.2.** If a call-tree \( H = (W, F) \) is trace compatible with \( \text{traceSequence} \) through a mapping \( \Phi_H \), then

\[
\forall w \in W : \Phi_0(\Phi_H^{-1}[\{w\}]) \text{ is a singleton}
\]

where \( f[A] \) (resp. \( f^{-1}[A] \)) denotes the image (resp. the pre-image) of the set \( A \) by the function \( f \).
**Proof.** Let’s first note that because $\Phi_H$ is by definition surjective, $\Phi_H^{-1}([x])$ is non-empty, and $\Phi_0[\Phi_H^{-1}([x])]$ therefore contains at least one element.

Let’s now consider

$$ x, y \in \Phi_0 \left[ \Phi_H^{-1}([w]) \right] $$

(22) means that there exists $\text{trace}_i, \text{trace}_j \in \text{traceSequence}, k \in [1, |\text{trace}_i|]$ and $l \in [1, |\text{trace}_j|]$ such that

$$ \langle \text{trace}_i, k \rangle, \langle \text{trace}_j, l \rangle \in \Phi_H^{-1}([w]) $$

(23)

$$ \Phi_0(\langle \text{trace}_i, k \rangle) = x $$

(24)

$$ \Phi_0(\langle \text{trace}_j, l \rangle) = y $$

(25)

(23) means in turn that:

$$ \Phi_H(\langle \text{trace}_i, k \rangle) = \Phi_H(\langle \text{trace}_j, l \rangle) = w $$

(26)

Because $H$ is a tree, Property P1 of $H$ and (26) imply that $l = k$ and that

$$ \forall m \in [1, k] : \Phi_H(\langle \text{trace}_i, m \rangle) = \Phi_H(\langle \text{trace}_j, m \rangle) $$

(27)

If $i = j$, (24) and (25), and $l = k$ imply trivially that $x = y$. Without loss of generality, we will therefore assume in the remaining that $i < j$.

We will now prove by contradiction that

$$ \forall r \in [i + 1, j] : |\text{trace}_r| > k $$

(28)

$$ \forall r \in [i, j], \forall m \in [1, k] : \Phi_H(\langle \text{trace}_r, m \rangle) = \Phi_H(\langle \text{trace}_i, m \rangle) $$

(29)

Let’s first consider (28) and assume there exists $r \in [i + 1, j]$ such that $|\text{trace}_r| \leq k$. Let’s note $m_r \equiv |\text{trace}_r|$. By applying property P2 of $H$ on $\text{trace}_i$ and $\text{trace}_r$, we get

$$ \Phi_H(\langle \text{trace}_r, m_r \rangle) \neq \Phi_H(\langle \text{trace}_i, m_r \rangle) $$

(30)

Because of (27) and $m_r \equiv |\text{trace}_r| \leq k$ we also have

$$ \Phi_H(\langle \text{trace}_r, m_r \rangle) \neq \Phi_H(\langle \text{trace}_j, m_r \rangle) $$

(31)

From this we also conclude that $r < j$. Since $i < r < j$, we can apply the property P3 of $H$ twice, first on $\text{trace}_i$ and $\text{trace}_r$, and then on $\text{trace}_r$ and $\text{trace}_j$, and we eventually get:

$$ \text{timeStamp} \left( \Phi_H(\langle \text{trace}_i, m_r \rangle) \right) < \text{timeStamp} \left( \Phi_H(\langle \text{trace}_r, m_r \rangle) \right) < \text{timeStamp} \left( \Phi_H(\langle \text{trace}_j, m_r \rangle) \right) $$

(32)

(32) implies that $\Phi_H(\langle \text{trace}_i, m_r \rangle) \neq \Phi_H(\langle \text{trace}_j, m_r \rangle)$, which contradicts (27), thus proving (28).

Turning to (29), let’s assume there exists $r \in [i, j]$ and $n \in [1, k]$ such that

$$ \Phi_H(\langle \text{trace}_r, n \rangle) \neq \Phi_H(\langle \text{trace}_i, n \rangle) $$

(33)

We are back to equation (30) with $m_r$ replaced by $n$, and we derive the same contradiction, thus proving (29).

Using the property P0 of $H$, we derive from (29)

$$ \forall r \in [i, j], \forall m \in [1, k] : \text{trace}_r[m] = \text{trace}_i[m] $$

(34)

By construction of the loop at lines 3-5 of algorithm 1, we derive from (34) and (28) that

$$ \forall r \in [i, j], \forall m \in [1, k] : \text{ActiveTreePath}_r[m] = \text{ActiveTreePath}_i[m] $$

(35)
and by definition of $\Phi_0$ (equation (10)) that
\[
\Phi_0(\langle \text{trace}_i, k \rangle) = \Phi_0(\langle \text{trace}_j, k \rangle)
\] (36)
which with (24) and (25) and $l = k$ leads us to $x = y$ and concludes the proof of lemma 4.2.

**Lemma 4.3.** If a call-tree $H = (W, F)$ is trace compatible with traceSequence, then there exists a surjective call-tree homomorphism from $H$ onto $G_0 = (V_0, E_0)$.

**Proof.** Consider the following mapping $h$ from $H$ onto $G_0$:
\[
\forall w \in W : h(w) \equiv \text{the unique element of } \Phi_0\left[\Phi^{-1}_H\{w\}\right]
\] (37)
Because of lemma 4.2, $h$ is well-defined. Furthermore, because of the properties of set images and pre-images by a function, and because $\Phi_0$ is surjective, we have
\[
\forall x \in V_0 : \exists w \in \Phi_H\left[\Phi^{-1}_0[x]\right] : h(w) = x
\] (38)
h is therefore surjective.
In the following we first prove that $h$ is a graph homomorphism, and then that $h$ fulfils (7) and (8), and is thus a call tree homomorphism.
Let’s consider $v, w \in W$ such that $(v, w) \in F$. We need to prove that $\langle h(v), h(w) \rangle \in E_0$.
Because $\Phi_H$ is surjective, there exists a stack frame $\langle t, k \rangle$ such that
\[
\Phi_H(\langle t, k \rangle) = w
\] (39)
Because $(v, w) \in F$, $w$ is not the root of $H$, and by applying property P1 if $H$ on trace $t$, we conclude that $k > 1$. Again from property P1 we derive
\[
\left(\Phi_H(\langle t, k - 1 \rangle), \Phi_H(\langle t, k \rangle)\right) \in F
\] (40)
Because $H$ is a tree, $w$ has at most one parent. $(v, w) \in F$ and (40) thus imply:
\[
\Phi_H(\langle t, k - 1 \rangle) = v
\] (41)
From (39) and (41) we derive
\[
\langle t, k \rangle \in \Phi^{-1}_H[w]
\] (42)
\[
\langle t, k - 1 \rangle \in \Phi^{-1}_H[v]
\] (43)
And thus after applying the mapping $\Phi_0$ and using the definition of $h$ (37):
\[
\Phi_0(\langle t, k \rangle) \in \Phi_0\left[\Phi^{-1}_H[w]\right] = \{h(w)\}
\] (44)
\[
\Phi_0(\langle t, k - 1 \rangle) \in \Phi_0\left[\Phi^{-1}_H[v]\right] = \{h(v)\}
\] (45)
We finally get:
\[
\Phi_0(\langle t, k \rangle) = h(w)
\] (46)
\[
\Phi_0(\langle t, k - 1 \rangle) = h(v)
\] (47)
By applying property P1 of $G_0$ on trace $t$ we conclude:
\[(h(v), h(w)) = \left( \Phi_0(\langle t, k - 1 \rangle), \Phi_0(\langle t, k \rangle) \right) \in E_0\]
(48)
which shows that $h$ is a graph homomorphism from $H$ onto $G_0$.
We will now turn to equations (7) and (8) to prove that $h$ is a call-tree homomorphism and complete the lemma.
Let’s consider $w \in W$. Using property P0 of $H$ we have
\[\forall \langle t, k \rangle \in \Phi_H^{-1}[w] : symbol(w) = symbol(\Phi_H(\langle t, k \rangle)) = t[k]\]
(49)
Using now the definition of $h$ (37) and the property P0 of $G_0$ we have
\[\forall \langle t, k \rangle \in \Phi_H^{-1}[w] : symbol(h(w)) = symbol(\Phi_0(\langle t, k \rangle)) = t[k]\]
(50)
Because $\Phi_H$ is surjective, and therefore $\Phi_H^{-1}[w]$ is non-empty, we derive from (49) and (50) that:
\[symbol(w) = symbol(h(w))\]
(51)
which proves (7) for $h$.
Turning now to the proof of property (8), let’s consider $r, s \in h(W) = V_0$, and let’s assume
\[\forall u \in h^{-1}[r], \forall v \in h^{-1}[s] : \text{timeStamp}(u) \leq \text{timeStamp}(v)\]
(52)
Let’s note $R$ and $S$ the sets of stack frames such that
\[R \equiv \Phi_0^{-1}[r] \land S \equiv \Phi_0^{-1}[s]\]
(53)
By definition of $h$ (37) and lemma 4.2 we have
\[h^{-1}[r] = \Phi_H[R]\]
(54)
\[h^{-1}[s] = \Phi_H[S]\]
(55)
Because by definition of $R$ we have $\forall \langle t_i, k \rangle \in R : \Phi_0(\langle t_i, k \rangle) = r$, property P1 of $G_0$ and the fact that $G_0$ is a tree means all frames of $R$ are at the same stack depth $k_r$:
\[\forall \langle t_i, k \rangle \in R : k = k_r\]
(56)
The same is true of $S$. There exists a stack depth $m_s$ such that:
\[\forall \langle t_j, m \rangle \in S : m = m_s\]
(57)
Let’s consider two stack frames from $R$ and $S$:
\[\langle t_i, k_r \rangle \in R \land \langle t_j, m_s \rangle \in S\]
(58)
Note that by definition of $R$ and $S$ (53), we have
\[\Phi_0(\langle t_i, k_r \rangle) = r\]
\[\Phi_0(\langle t_j, m_s \rangle) = s\]
(59)
We first prove by contradiction that the following holds:
\[(j \geq i) \lor \left( \forall k \leq k_r : \Phi_H(\langle t_j, k \rangle) = \Phi_H(\langle t_i, k \rangle) \right)\]
(60)
Let’s assume the negation of (60):

\[(j < i) \land (\exists k \leq k_r : \Phi_H((t_j, k)) \neq \Phi_H((t_i, k)))\]  

(61)

We can apply property P3 of H on \(t_j\) and \(t_i\), and we get

\[
timeStamp(\Phi_H((t_j, m_s))) < timeStamp(\Phi_H((t_i, k_r)))
\]

(62)

Because \(\Phi_H((t_j, m_s)) \in \Phi_H[S] = h^{-1}[s]\) and \(\Phi_H((t_i, k_r)) \in \Phi_H[R] = h^{-1}[r]\), (62) contradicts (52). We therefore conclude that (60) must be true.

Because

\[(i = j) \Rightarrow (\Phi_H((t_j, k_r)) = \Phi_H((t_i, k_r)))\]

and

\[(\Phi_H((t_j, k_r)) = \Phi_H((t_i, k_r))) \Rightarrow (\forall k \leq k_r : \Phi_H((t_j, k)) = \Phi_H((t_i, k)))\]

(63)

(63) is furthermore equivalent to

\[(j > i) \lor (\forall k \leq k_r : \Phi_H((t_j, k)) = \Phi_H((t_i, k)))\]

(64)

The remaining of the proof considers two cases, depending whether \(m_s < k_r\).

**Case 1** \(m_s < k_r\)

If we assume \(\Phi_H((t_j, m_s)) = \Phi_H((t_i, m_s))\), then \(m_s < k_r\), property P1 of H, and (2) give us

\[
timeStamp(\Phi_H((t_j, m_s))) = timeStamp(\Phi_H((t_i, m_s)))
\]

(64)

\[
timeStamp(\Phi_H((t_j, m_s))) < timeStamp(\Phi_H((t_i, k_r)))
\]

(65)

The same equation as (62), which yields the same contradiction with (52). From this we conclude that:

\[\Phi_H((t_j, m_s)) \neq \Phi_H((t_i, m_s))\]  

(66)

With (63) and \(m_s < k_r\) (case assumption), this implies that

\[j > i\]  

(67)

(66) and (67) mean we can apply property P3 of \(\Phi_0\) to \(t_i\) and \(t_j\). We get

\[
timeStamp(\Phi_0((t_j, m_s))) > timeStamp(\Phi_0((t_i, k_r)))
\]

(68)

Using (59), this yields

\[timeStamp(s) > timeStamp(r)\]

(69)

and proves (8) in this case.

**Case 2** \(m_s \geq k_r\)

We consider in turn the two alternatives of (63).

If \(j > i\), then by construction of algorithm 1 and the property of the node constructor \texttt{new Node(trace[k])}\) at line 13, we have

\[
timeStamp(\Phi_0((t_i, k_r))) \leq timeStamp(\Phi_0((t_j, k_r)))
\]

(70)
Because of sublemma 4.1.1 on $\Phi_0$ and $m_s \geq k_r$ we also have

\[
timeStamp(\Phi_0((t_j, k_r))) \leq timeStamp(\Phi_0((t_j, m_s)))
\]  
(71)

From (70) and (71) we derive

\[
timeStamp(\Phi_0((t_i, k_r))) \leq timeStamp(\Phi_0((t_j, m_s)))
\]  
(72)

which using (59), gives us

\[
timeStamp(r) \leq timeStamp(s)
\]  
(73)

and proves (8) when $j > i$.

If we turn to the second term of (63) and assume

\[
\Phi_H((t_j, k_r)) = \Phi_H((t_i, k_r))
\]  
(74)

Because of (58) and (54) we have

\[
\Phi_H((t_i, k_r)) \in \Phi_H[R] = h^{-1}[r]
\]  
(75)

With (74), we thus have

\[
\Phi_H((t_j, k_r)) \in h^{-1}[r]
\]  
(76)

From which we derive

\[
(t_j, k_r) \in \Phi^{-1}_H[h^{-1}[r]]
\]  
\[
\Phi_0((t_j, k_r)) \in \Phi_0[\Phi^{-1}_H[h^{-1}[r]]]
\]  
(77)

By definition of $h$ (37) and lemma 4.2 this yields

\[
\Phi_0((t_j, k_r)) = r
\]  
(79)

Because $m_s \geq k_r$, sublemma 4.1.1 on $t_j$ gives us

\[
timeStamp(\Phi_0((t_j, m_s))) \geq timeStamp(\Phi_0((t_j, k_r)))
\]  
(80)

This, with (59) and (79) leads to

\[
timeStamp(s) \geq timeStamp(r)
\]  
(81)

and concludes the proof (8) for $h$.

$h$ is thus a surjective graph homomorphism that fulfils (7) and (8), which proves lemma 4.3.

**Theorem 3.1.** The set of call-trees that are trace-compatible with a sequence of observed stack traces is non-empty. The call-tree of this set with the smallest number of nodes is unique (modulo a call-tree isomorphism) and is the call-tree constructed by algorithm 1.

**Proof.** As before, let traceSequence be a sequence of observed stack traces, and $G_0 = (E_0, V_0)$ the call-tree constructed by algorithm 1 for traceSequence.

Because of lemma 4.1, the set of call-trees that are trace-compatible with traceSequence contains $G_0$ and is therefore non-empty.

Let’s consider $H = (W, F)$ a call tree that is trace-compatible with traceSequence. Because of lemma 4.3, there exists a surjective call-tree homomorphism $h$ from $H$ onto $G_0$. Because $h$ is surjective we have

\[
|E_0| \leq |W|
\]  
(82)
i.e., $G_0$ has the smallest number of nodes of all call-trees compatible with traceSequence.

Let’s now consider $H_0 = (W_0, F_0)$, a call tree that is trace-compatible with traceSequence and has the same number of nodes as $G_0$.

$$|E_0| = |W_0|$$

Because of lemma 4.3, there exists a surjective call-tree homomorphism $h_0$ from $H_0$ onto $G_0$.

Because $h_0$ is surjective, and its domain and co-domain are finite and contain the same number of elements, then $h_0$ is a bijection. $H_0$ onto $G_0$ are therefore identical modulo a call-tree isomorphism.

$G_0$ is therefore unique (modulo a call-tree isomorphism), which concludes the proof of theorem 3.1.

\[ \square \]

References