Proof of `advance` in the UCK State Machine

Terminology.

- *A* ***well-defined state*** *is an instance of the type* State *or* Right[KeyInputError, State]
- *A* ***state*** *is an instance of the type* Either[KeyInputError, State]
- *A* ***node*** *is an instance of the type* Node *or* (Node, NodeId).

When multiple interpretations are possible, the context will always make clear which one we are referring to. In particular, depending on the situation, beginRollback s *and* endRollback s *will return either a* State *or a* Right[KeyInputError, State].

Definition 1. *Let s be a state,* **defined s** *is true if s is a well-defined state. When this is the case,* **get s** *yields the* State *instance out of the option.*

Definition 2. *Let $t := (v_1, v_2, v_3)$ a triple. The projections of the triple are* $\mathbf{proj_1}\ \mathbf{t} := v_1$, $\mathbf{proj_2}\ \mathbf{t} := v_2$ *and* $\mathbf{proj_3}\ \mathbf{t} := v_3$.

Trees

Definition 3. *A tree of nodes is a data structure defined inductively as either:*

- Endpoint
- ContentNode $sub\ n$, *where sub is a tree and n a node.*
- ArticulationNode $l\ r$, *where l and r are both trees.*

Although this is not the most intuitive definition, it has the advantage of being able to represent forests without running into measure decreaseness problems. In fact one can see content nodes as being trees and articulation nodes as forests.

Definition 4. *Let tr be a tree of nodes,* **size tr** *is defined inductively as:*

- size Endpoint $:= 0$
- size (ContentNode $sub\ n$) $:=$ (size sub) $+ 1$
- size (ArticulationNode $l\ r$) $:=$ (size l) $+$ (size r)

Definition 5. *Let tr be a tree of nodes and $init$ a value,* **traverse tr init** $\mathbf{f_1}\ \mathbf{f_2}$ *is defined inductively as:*

- traverse Endpoint $init\ f_1\ f_2 := init$
- traverse (ContentNode $sub\ n$) $init\ f_1\ f_2 := f_2($traverse $sub\ f_1(init, n)\ f_1\ f_2, n)$
- traverse (ArticulationNode $l\ r$) $init\ f_1\ f_2 :=$ traverse r (traverse $l\ init\ f_1\ f_2$) $f_1\ f_2$

Definition 6. *Let tr be a tree of nodes and $init$ a value,* **scan tr init** $\mathbf{f_1}\ \mathbf{f_2}$ *is defined inductively as:*

- scan Endpoint $init\ f_1\ f_2 := [\ ]$

- scan (ContentNode sub n) $init$ f_1 f_2 :=

$$[(init, n, \downarrow)] \mathbin{++} \text{scan } sub\ f_1(init, n)\ f_1\ f_2 \mathbin{++} [(\text{traverse } sub\ f_1(init, n)\ f_1\ f_2, n, \uparrow)]$$

- scan (ArticulationNode l r) $init$ f_1 f_2 :=

$$\text{scan } l\ init\ f_1\ f_2 \mathbin{++} \text{scan } r\ (\text{traverse } l\ init\ f_1\ f_2)\ f_1\ f_2$$

Claim 7. *Let tr be a tree of nodes,*

$$\text{size } tr \geq 0$$

Proof. Straight induction on tr. □

Claim 8. *For any tree of nodes tr, value $init$ and functions f_1, f_2*

$$\text{size } (\text{scan } tr\ init\ f_1\ f_2) = 2 \cdot \text{size } tr$$

Proof. Straight induction on tr. □

Lemma 9. *For any tree of nodes tr, value $init$ and functions f_1, f_2*

If $tr =$ ContentNode sub n:

$$(\text{scan } tr\ init\ f_1\ f_2)[i] = \begin{cases} (init, n, \downarrow) & \textit{if } i = 0 \\ (\text{traverse } sub\ f_1(init, n)\ f_1\ f_2, n, \uparrow) & \textit{if } i = 2 \cdot \text{size } tr - 1 \\ (\text{scan } sub\ f_1(init, n)\ f_1\ f_2)[i-1] & \textit{otherwise} \end{cases}$$

If $tr =$ ArticulationNode l r:

$$(\text{scan } tr\ init\ f_1\ f_2)[i] = \begin{cases} (\text{scan } l\ init\ f_1\ f_2)[i] & \textit{if } i < 2 \cdot \text{size } l \\ (\text{scan } r\ (\text{traverse } l\ init\ f_1\ f_2)\ f_1\ f_2)[i - 2 \cdot \text{size } l] & \textit{otherwise} \end{cases}$$

Proof. Application of union indexing property in lists □

Lemma 10. *Let tr be a tree of nodes tr, $init$ a value, f_1, f_2 two functions and $i < 2 \cdot \text{size } tr$ a non-negative integer. Let $l :=$ scan tr $init$ f_1 f_2:*

$$\text{proj}_1\ (l[i]) = \begin{cases} init & \textit{if } i = 0 \\ f_1(\text{proj}_1\ (l[i-1]), \text{proj}_2\ (l[i-1])) & \textit{if } \text{proj}_3\ (l[i-1]) = \downarrow \\ f_2(\text{proj}_1\ (l[i-1]), \text{proj}_2\ (l[i-1])) & \textit{if } \text{proj}_3\ (l[i-1]) = \uparrow \end{cases} \tag{1}$$

and

$$\text{traverse } tr\ init\ f_1\ f_2 = \begin{cases} init & \textit{if } \text{size } tr = 0 \\ f_2(\text{proj}_1\ (l[2 \cdot \text{size } tr - 1]), \text{proj}_2\ (l[2 \cdot \text{size } tr - 1])) & \textit{otherwise} \end{cases} \tag{2}$$

Proof. Induction on tr:

- If $tr =$ Endpoint, then size $tr = 0$.
- If $tr =$ ContentNode sub n:
  - If $i = 0$ then by Lem. 9 $l[i] = (init, n, \downarrow)$.

– If $i = 2 \cdot \text{size } tr - 1$, then by Lem. 9 $l[i] = (\text{traverse } sub\ f_1(init, n)\ f_1\ f_2, n, \uparrow)$.

Let $l_{sub} := \text{scan } sub\ f_1(init, n)\ f_1\ f_2$.

If size $sub = 0$, then $i = 1$:

$$\begin{aligned}
\text{traverse } sub\ f_1(init,n)\ f_1\ f_2 =& f_1(init,n) && \text{by IH}\\
=& f_1(\text{proj}_1\ (l[0]), \text{proj}_2\ (l[0])) && \text{by Lem. 9}\\
=& f_1(\text{proj}_1\ (l[i-1]), \text{proj}_2\ (l[i-1]))
\end{aligned}$$

Moreover we know that $\text{proj}_3\ (l[0]) = \downarrow$

Otherwise:

$$\begin{aligned}
&\text{traverse } sub\ f_1(init,n)\ f_1\ f_2\\
=& f_2(\text{proj}_1\ (l_{sub}[2 \cdot \text{size } sub - 1]), \text{proj}_2\ (l_{sub}[2 \cdot \text{size } sub - 1]))\\
=& f_2(\text{proj}_1\ (l_{sub}[2 \cdot \text{size } tr - 3]), \text{proj}_2\ (l_{sub}[2 \cdot \text{size } tr - 3])) && \text{unfolding Def. 4}\\
=& f_2(\text{proj}_1\ (l[2 \cdot \text{size } tr - 2]), \text{proj}_2\ (l[2 \cdot \text{size } tr - 2])) && \text{by Lem. 9}\\
=& f_2(\text{proj}_1\ (l[i-1]), \text{proj}_2\ (l[i-1]))
\end{aligned}$$

– Else apply induction hypothesis with sub, $f_1(init, n)$ and $i - 1$. Use Lem. 9 when $i > 1$.

– To prove (2) we know by Def. 4 that size $tr > 0$. The result is then immediate from Lem. 9.

- If $tr = \text{ArticulationNode } le\ ri$:

– If $i < 2 \cdot \text{size } le$, apply induction hypothesis with le, $init$ and i, and Lem. 9.

– If $i = 2 \cdot \text{size } le$:

Let $l_{le} := (\text{scan } le\ init\ f_1\ f_2)$ and $l_{ri} := (\text{scan } ri\ (\text{traverse } le\ init\ f_1\ f_2)\ f_1\ f_2)$

$$\begin{aligned}
\text{proj}_1\ (l_{ri}[0]) &= \text{traverse } le\ init\ f_1\ f_2 && \text{by IH on } ri\\
\text{proj}_1\ (l_{ri}[i - 2 \cdot \text{size } le]) &= \text{traverse } le\ init\ f_1\ f_2\\
\text{proj}_1\ (l[i]) &= \text{traverse } le\ init\ f_1\ f_2 && \text{by Lem. 9}\\
&= f_2(\text{proj}_1\ (l_{le}[2 \cdot \text{size } le - 1]), \text{proj}_2\ (l_{le}[2 \cdot \text{size } le - 1])) && \text{by IH on } le\\
&= f_2(\text{proj}_1\ (l_{le}[i-1]), \text{proj}_2\ (l_{le}[i-1]))
\end{aligned}$$

– Else apply induction hypothesis with ri, traverse $le\ init\ f_1\ f_2$ and $i - 2 \cdot \text{size } le$ and Lem. 9.

– If size $tr = 0$ then by Claim 7 and unfolding Def. 4, size $le = 0$ and size $ri = 0$.

$$\begin{aligned}
\text{traverse } tr\ init\ f_1\ f_2 &= \text{traverse } ri\ (\text{traverse } le\ init\ f_1\ f_2)\ f_1\ f_2 && \text{unfolding Def. 5}\\
&= \text{traverse } le\ init\ f_1\ f_2 && \text{by IH on } ri\\
&= init && \text{by IH on } l
\end{aligned}$$

Else if size $ri = 0$:

$$\begin{aligned}
\text{traverse } tr\ init\ f_1\ f_2 &= \text{traverse } ri\ (\text{traverse } le\ init\ f_1\ f_2)\ f_1\ f_2 && \text{unfolding Def. 5}\\
&= \text{traverse } le\ init\ f_1\ f_2 && \text{by IH on } ri\\
&= f_2(\text{proj}_1\ (l_{le}[2 \cdot \text{size } le - 1]), \text{proj}_2\ (l_{le}[2 \cdot \text{size } le - 1])) && \text{by IH on } le\\
&= f_2(\text{proj}_1\ (l[2 \cdot \text{size } le - 1]), \text{proj}_2\ (l[2 \cdot \text{size } le - 1])) && \text{by Lem. 9}\\
&= f_2(\text{proj}_1\ (l[2 \cdot \text{size } tr - 1]), \text{proj}_2\ (l[2 \cdot \text{size } tr - 1])) && \text{by Def. 4}
\end{aligned}$$

When size $ri > 0$:

$$\begin{aligned}
&\text{traverse } tr\ init\ f_1\ f_2 \\
&= \text{traverse } ri\ (\text{traverse } le\ init\ f_1\ f_2)\ f_1\ f_2 && \text{unfolding Def. 5} \\
&= f_2(\text{proj}_1\ (l_{ri}[2 \cdot \text{size } ri - 1]), \text{proj}_2\ (l_{ri}[2 \cdot \text{size } ri - 1])) && \text{by IH on } ri \\
&= f_2(\text{proj}_1\ (l[2 \cdot \text{size } ri + 2 \cdot \text{size } le - 1]), \text{proj}_2\ (l[2 \cdot \text{size } ri + 2 \cdot \text{size } le - 1])) && \text{by Lem. 9} \\
&= f_2(\text{proj}_1\ (l[2 \cdot \text{size } tr - 1]), \text{proj}_2\ (l[2 \cdot \text{size } tr - 1])) && \text{by Def. 4}
\end{aligned}$$

□

Claim 11. *For any tree of nodes tr, values $init_1$, $init_2$, functions $f_1^1, f_2^1, f_1^2, f_2^2$ and non-negative integer $i < 2 \cdot \text{size } tr$.*

$$\text{proj}_2\ ((\text{scan } tr\ init_1\ f_1^1\ f_2^1)[i]) = \text{proj}_2\ ((\text{scan } tr\ init_2\ f_1^2\ f_2^2)[i])$$

and

$$\text{proj}_3\ ((\text{scan } tr\ init_1\ f_1^1\ f_2^1)[i]) = \text{proj}_3\ ((\text{scan } tr\ init_2\ f_1^2\ f_2^2)[i])$$

Proof. Induction on tr using Lem. 9. □

We now define the two functions we will be applying during a tree traversal.

Definition 12 (`traverseInFun`, `traverseOutFun`)**.** *Let s be a state and n be a node,*

$$f_{down}(s, n) := \begin{cases} \text{handleNode (get } s)\ n & \textit{if } n \textit{ is an Action node and } \text{defined } s \\ \text{beginRollback (get } s) & \textit{if } n \textit{ is a Rollback node and } \text{defined } s \\ s & \textit{otherwise} \end{cases}$$

$$f_{up}(s, n) := \begin{cases} \text{endRollback (get } s) & \textit{if } n \textit{ is a Rollback node and } \text{defined } s \\ s & \textit{otherwise} \end{cases}$$

In the rest of the document, unless stated otherwise, init will be a state and traverse $tr\ init$ and scan $tr\ init$ will be referring to as respectively traverse $tr\ init\ f_{down}\ f_{up}$ and scan $tr\ init\ f_{down}\ f_{up}$

Claim 13 (`traverseTransactionProp`)**.** *For any tree of nodes tr, state $init$, if* defined (traverse $tr\ init$) *then*

$$\text{defined } init \wedge (\text{get (traverse } tr\ init)).rollbackStack = (\text{get } init).rollbackStack \wedge$$

$$(\text{get (traverse } tr\ init)).globalKeys = (\text{get } init).globalKeys$$

Proof. Straight induction on tr. □

Claim 14 (`scanTransactionProp`)**.** *Let tr be a tree of nodes, $init$ a state, $0 \leq i \leq j < 2 \cdot \text{size } tr$ two integers and let $l :=$ scan $tr\ init$.*

$$\text{defined } (\text{proj}_1\ (l[j])) \implies \text{defined } (\text{proj}_1\ (l[i]))$$

Proof. By induction on j: the base case is immediate and Lem. 10 combined with the induction hypothesis concludes the proof. □

Corollary 15 (`scanTransactionProp`)**.** *Let tr be a tree of nodes, $init$ a state, $0 \leq i < 2 \cdot \text{size } tr$ an integer and let $l :=$ scan $tr\ init$.*

$$\text{defined (traverse } tr\ init) \implies \text{defined } (\text{proj}_1\ (l[i]))$$

Proof. By Lem. 10 and Claim 14 setting $j = 2 \cdot \text{size } tr - 1$. □

Lemma 16. *Let tr be a tree of nodes, $init$ a state and $l :=$ scan tr $init$ then*

$$\text{defined (traverse } tr\ init) \iff \begin{cases} \text{defined } init \\ \forall\, 0 \leq i < 2 \cdot \text{size } tr - 1, \text{ defined } (\text{proj}_1\ (l[i])) \implies \text{defined } (\text{proj}_1\ (l[i+1])) \end{cases}$$

Proof. Let's first note that by Lem. 10, $\text{proj}_1\ (l[0]) = init$.

If defined (traverse tr $init$) then by Cor. 15, defined $\text{proj}_1\ (l[i])$ for all $0 \leq i < 2 \cdot \text{size } tr$.

If the right statement is true, then defined $(\text{proj}_1\ (l[2 \cdot \text{size } tr - 1]))$ and therefore by Lem. 10 we have defined (traverse tr $init$). □

Claim 17 (`findBeginRollback`)**.** *Let tr be a tree of nodes, $init_1, init_2$ be states and let $l_1 :=$ scan tr $init_1$, $l_2 :=$ scan tr $init_2$. If there exists a non-negative integer $i < 2 \cdot (\text{size } tr)$, well-defined states s_i^1, s_i^2 and a Rollback node n such that $l_1[i] = (s_i^1, n, \uparrow)$ and $l_2[i] = (s_i^2, n, \uparrow)$, then there is an integer j, well-defined states s_j^1, s_j^2 and a tree sub such that*

$$0 \leq j < i \qquad l_1[j] = (s_j^1, n, \downarrow) \qquad l_2[j] = (s_j^2, n, \downarrow) \qquad \text{size } sub < \text{size } tr$$

$$s_i^1 = \text{traverse } sub \text{ (beginRollback (get } s_j^1)) \qquad s_i^2 = \text{traverse } sub \text{ (beginRollback (get } s_j^2))$$

Proof. Induction on tr:

- If $tr =$ Endpoint, then size $tr = 0$ which means the precondition is never met.
- If $tr =$ ContentNode str c:
  - If $c \neq n$ then by Lem. 9, $0 < i < 2 \cdot (\text{size } tr) - 1$, $l_1[i] = (\text{scan } str\ f_{down}(init_1, n))[i-1]$ and $l_2[i] = (\text{scan } str\ f_{down}(init_2, n))[i-1]$. By induction hypothesis there is an integer j, well-defined state s_j^1, s_j^2 and a tree sub such that:

    $$(\text{scan } str\ f_{down}(init_1, n))[j] = (s_j^1, n, \downarrow) \qquad (\text{scan } str\ f_{down}(init_2, n))[j] = (s_j^2, n, \downarrow)$$

    $$s_i^1 = \text{traverse } sub \text{ (beginRollback (get } s_j^1)) \qquad s_i^2 = \text{traverse } sub \text{ (beginRollback (get } s_j^2))$$

    $$0 \leq j < i - 1 \qquad \text{size } sub < \text{size } tr$$

    Since $l_1[j+1] = (\text{scan } str\ f_{down}(init_1, n))[j]$ and $l_2[j+1] = (\text{scan } str\ f_{down}(init_2, n))[j]$, $j+1$, sub, s_j^1 and s_j^2 satisfy the above conditions.
  - If $c = n$, then $i = 2 \cdot (\text{size } tr) - 1$ is valid and therefore $j = 0$, $s_j^1 = init_1$, $s_j^2 = init_2$ and $sub = str$.
- If $tr =$ ArticulationNode $left$ $right$:
  - If $i < 2 \cdot \text{size } left$, then by Lem. 9, $l_1[i] = (\text{scan } left\ init_1)[i]$ and $l_2[i] = (\text{scan } left\ init_2)[i]$. By induction hypothesis, there are j, well-defined s_j^1, s_j^2 and sub such that.

    $$(\text{scan } left\ init_1)[j] = (s_j^1, n, \downarrow) \qquad (\text{scan } left\ init_2)[j] = (s_j^2, n, \downarrow)$$

    $$s_i^1 = \text{traverse } sub \text{ (beginRollback (get } s_j^1)) \qquad s_i^2 = \text{traverse } sub \text{ (beginRollback (get } s_j^2))$$

    $$0 \leq j < i \qquad \text{size } sub < \text{size } left$$

    Since $l_1[j] = (\text{scan } left\ init_1)[j]$ and $l_2[j] = (\text{scan } left\ init_2)[j]$, j, s_j^1, s_j^2 and sub satisfy the claim.
  - If $i \geq 2 \cdot \text{size } left$, then by Lem. 9, $l_1[i] = (\text{scan } right \text{ (traverse } left\ init_1))[i - 2 \cdot \text{size } left]$ and $l_2[i] = (\text{scan } right \text{ (traverse } left\ init_2))[i - 2 \cdot \text{size } left]$. By induction hypothesis, there are j, well-defined s_j^1, s_j^2 and sub such that.

    $$(\text{scan } right \text{ (traverse } left\ init_1))[j] = (s_j^1, n, \downarrow) \qquad (\text{scan } right \text{ (traverse } left\ init_2))[j] = (s_j^2, n, \downarrow)$$

$$s_i^1 = \text{traverse } sub \text{ (beginRollback (get } s_j^1)) \qquad s_i^2 = \text{traverse } sub \text{ (beginRollback (get } s_j^2))$$
$$0 \leq j < i - 2 \cdot \text{size } left \qquad \text{size } sub < \text{size } right$$

Since $l_1[j + 2 \cdot \text{size } left] = (\text{scan } right \text{ (traverse } left\ init_1))[j]$ and $l_2[j + 2 \cdot \text{size } left] = (\text{scan } right \text{ (traverse } left\ init_2))[j]$, $j + 2 \cdot \text{size } left$, s_j^1, s_j^2 and sub satisfy the claim.

□

Corollary 18 (`findBeginRollback`)**.** *Let tr be a tree of nodes, init a states and let* $l := \text{scan } tr\ init$*. If there exists a non-negative integer* $i < 2 \cdot (\text{size } tr)$*, a well-defined states* s_i *and a rollback node* n *such that* $l[i] = (s_i, n, \uparrow)$*, then there is an integer* j*, a well-defined states* s_j*, and a tree sub such that*

$$0 \leq j < i \qquad l[j] = (s_j, n, \downarrow) \qquad s_i = \text{traverse } sub \text{ (beginRollback } s_j) \qquad \text{size } sub < \text{size } tr$$

Proof. By Claim 17 with $init_1 = init_2$. □

Active Keys Lemmas

Definition 19. *Let s be a well-defined state and k a key,* $\mathbf{act_k\ s}$ *is the value associated to key in the active keys of the state (i.e. we first look at the local keys, then the global ones filtering the consumed contracts).*

Definition 20. *Let n be a node,* **mapping n** *is defined as:*

- mapping *(*Create *id k) :=* KeyInactive
- mapping *(*Fetch *id k) :=* KeyActive *id*
- mapping *(*Lookup *result k) := result*
- mapping *(*Exercise *id k) :=* KeyActive *id*

Lemma 21 (already proven)**.** *For any well-defined state s and Action node n,*

$$\text{defined (handleNode } s\ n) \iff \text{act}_{n.k}\ s = \text{mapping } n$$

Corollary 22. *For any well-defined states s_1, s_2 and Action node n, if* defined (handleNode s_1 n) *and* defined (handleNode s_2 n)

$$\text{act}_{n.k}\ s_1 = \text{act}_{n.k}\ s_2$$

Proof. Direct consequence of Lem. 21. □

Lemma 23 (already proven)**.** *For any well-defined states s_1, s_2 and Action node n, if* defined (handleNode s_1 n) *and* defined (handleNode s_2 n)

$$\text{act}_{n.k}\ (\text{get (handleNode } s_1\ n)) = \text{act}_{n.k}\ (\text{get (handleNode } s_2\ n))$$

Lemma 24 (already proven)**.** *For any well-defined state s, Action node n, key k_2, if n has no key or $k_2 \neq n.k$ and if* defined (handleNode s n),

$$\text{act}_{k_2}\ (\text{get (handleNode } s\ n)) = \text{act}_{k_2}\ s$$

Lemma 25 (already proven)**.** *For any well-defined state s, node n, key k,*

$$\text{act}_k\ (\text{beginRollback } s) = \text{act}_k\ s$$

Lemma 26 (already proven)**.** *For any well-defined state s, node n, key k, function g :* State → State *and:*

- $g(\text{beginRollback } s).rollbackStack = (\text{beginRollback } s).rollbackStack$
- $g(\text{beginRollback } s).globalKeys = (\text{beginRollback } s).globalKeys$

We have:

$$\text{act}_k\ (\text{endRollback } g(\text{beginRollback } s)) = \text{act}_k\ s$$

The real deal

Definition 27 (`appearsAtIndex`, `doesNotAppearBefore`, `firstAppears`)**.** *Let tr be a tree of nodes, $init$ a value, k a key, f_1, f_2 two functions, $i < 2 \cdot \text{size } tr$ a non-negative integer and let $l := \text{scan } tr \; init \; f_1 \; f_2$.*

We say that k does not appear before i if for all $0 \leq j < i$, $(\text{proj}_2 \; l[j]).k \neq k$ or $\text{proj}_3 \; l[j] = \uparrow$

We say that i is the first appearance of k in l if $(\text{proj}_2 \; l[i]).k = k$, $\text{proj}_3 \; l[i] = \downarrow$ and k does not appear before i.

Claim 28 (`doesNotAppearBeforeSame`, `firstAppearsSame`)**.** *Let tr be a tree of nodes, $init, init_2$ a state, $f_1^1, f_2^1, f_1^2, f_2^2$ functions, k a key, i a non-negative integer smaller than $2 \cdot \text{size } tr$, $l_1 := \text{scan } tr \; init_1 \; f_1^1 \; f_2^1$ and $l_2 := \text{scan } tr \; init_2 \; f_1^2 \; f_2^2$.*

$$k \text{ does not appear before } i \text{ in } l_1 \iff k \text{ does not appear before } i \text{ in } l_2$$

and in particular

$$i \text{ is the first appearance of } k \text{ in } l_1 \iff i \text{ is the first appearance of } k \text{ in } l_2$$

Proof. Consequence of Claim 11. □

Claim 29 (`findFirstAppears`)**.** *Let tr be a tree of nodes, $init$ a state, k a key, $0 \leq i_1 < i_2 < 2 \cdot \text{size } tr$ twos integers and let $l := \text{scan } tr \; init$. If k appears before i_2 in l but does not before i_1, then there exists an integer $i_1 \leq j < i_2$ such that j is the first appearance of k in l.*

Proof. Immediate from Def. 27. □

Claim 30. *Let tr be a tree of nodes, $init$ a state, k a key, $0 \leq j < i < 2 \cdot \text{size } tr$ two integers and let $l := \text{scan } tr \; init$. If k does not appear before i in l and $\text{defined } (\text{proj}_1 \; (l[i]))$ (and $\text{defined } (\text{proj}_1 \; (l[j]))$ by Claim 14) then*

$$\text{act}_k \; (\text{get } (\text{proj}_1 \; (l[i]))) = \text{act}_k \; (\text{get } (\text{proj}_1 \; (l[j])))$$

Proof. By induction on i.

If $i = 0$ then the precondition is never met.

Else let $(s_{i-1}, n_{i-1}, dir_{i-1}) := l[i-1]$ and $s_i := \text{proj}_1 \; (l[i])$. By Claim 14 defined s_{i-1}.

By Lem. 10 we either have:

- $s_i = f_{down}(s_{i-1}, n_{i-1})$ and $dir_{i-1} = \downarrow$.

  If n_{i-1} is an Action node, since k does not appear before i, $k \neq n_{i-1}.k$, then:

$$\begin{aligned} \text{act}_k \; (\text{get } s_i) &= \text{act}_k \; (\text{get } (\text{handleNode } (\text{get } s_{i-1}) \; n_{i-1})) \\ &= \text{act}_k \; (\text{get } s_{i-1}) && \text{from Lem. 24} \\ &= \text{act}_k \; (\text{get } (\text{proj}_1 \; (l[j]))) && \text{for all } j < i-1 \text{ by IH} \end{aligned}$$

  If n_{i-1} is a Rollback node then

$$\begin{aligned} \text{act}_k \; (\text{get } s_i) &= \text{act}_k \; (\text{beginRollback } (\text{get } s_{i-1})) \\ &= \text{act}_k \; (\text{get } s_{i-1}) && \text{by Lem. 25} \\ &= \text{act}_k \; (\text{get } (\text{proj}_1 \; (l[j]))) && \text{for all } j < i-1 \text{ by IH} \end{aligned}$$

- $s_i = f_{up}(s_{i-1}, n_{i-1})$ and $dir_{i-1} = \uparrow$.

If n_{i-1} is an Action node then

$$\begin{aligned}
\text{act}_k\ (\text{get}\ s_i) &= \text{act}_k\ (\text{get}\ s_{i-1}) \\
&= \text{act}_k\ (\text{get}\ (\text{proj}_1\ (l[j]))) && \text{for all } j < i-1 \text{ by IH}
\end{aligned}$$

If n_{i-1} is a Rollback node then

$$\text{act}_k\ (\text{get}\ s_i) = \text{act}_k\ (\text{endRollback}\ (\text{get}\ s_{i-1}))$$

By Cor. 18 there exists an integer $0 \leq j' < i-1$, a well-defined state $s_{j'} = \text{proj}_1\ (l[j'])$ and a tree sub such that $s_{i-1} = \text{traverse}\ sub\ (\text{beginRollback}\ (\text{get}\ s_{j'}))$. Therefore

$$\begin{aligned}
\text{act}_k\ (\text{get}\ s_i) &= \text{act}_k\ (\text{endRollback}\ (\text{get}\ (\text{traverse}\ sub\ (\text{beginRollback}\ (\text{get}\ s'_j))))) \\
&= \text{act}_k\ (\text{get}\ s_{j'}) && \text{by Lem. 26} \\
&= \text{act}_k\ (\text{get}\ (\text{proj}_1\ (l[j'])))
\end{aligned}$$

In addition, by the induction hypothesis

$$\begin{aligned}
\text{act}_k\ (\text{get}\ s_{i-1}) &= \text{act}_k\ (\text{get}\ (\text{proj}_1\ (l[j]))) && \text{for all } j < i-1 \\
&= \text{act}_k\ (\text{get}\ (\text{proj}_1\ (l[j']))) \\
&= \text{act}_k\ (\text{get}\ s_i)
\end{aligned}$$

$\square$

Corollary 31. *Let tr be a tree of nodes, $init$ a state, k a key, $0 \leq j < i < 2 \cdot \text{size}\ tr$ two integers and let $l := \text{scan}\ tr\ init$. If i is the first appearance of k in l and* $\text{defined}\ (\text{proj}_1\ (l[i]))$ *(and* $\text{defined}\ (\text{proj}_1\ (l[j]))$ *by Claim 14), then*

$$\text{act}_k\ (\text{get}\ (\text{proj}_1\ (l[i]))) = \text{act}_k\ (\text{get}\ (\text{proj}_1\ (l[j])))$$

Proof. Direct consequence of Claim 30. $\square$

Corollary 32. *Let tr be a tree of nodes, $init$ a state, $0 \leq i < 2 \cdot \text{size}\ tr$ an integer, let $l := \text{scan}\ tr\ init$ and $(s, n, dir) := l[i]$. If n is an Action node with a well-defined key, i is the first appearance of $n.k$ in l and* defined s*, then*

$$\text{act}_{n.k}\ (\text{get}\ init) = \text{act}_{n.k}\ (\text{get}\ s)$$

and in particular

$$\text{defined}\ (\text{handleNode}\ (\text{get}\ s)\ n) \iff \text{act}_{n.k}\ (\text{get}\ init) = \text{mapping}\ n$$

Proof. The first statement is a direct consequence of Cor. 31. Applying Lem. 21 gives us the second statement. $\square$

Claim 33. *Let tr be a tree of nodes, $init_1$, $init_2$ two state, k a key, $0 \leq i < 2 \cdot \text{size}\ tr$ an integer, let $l_1 := \text{scan}\ tr\ init_1$, $l_2 := \text{scan}\ tr\ init_2$, $s_i^1 := \text{proj}_1\ (l_1[i])$ and $s_i^2 := \text{proj}_1\ (l_2[i])$. If k appears before i in l_1 and l_2,* defined s_i^1 *and* defined s_i^2*, then*

$$\text{act}_k\ (\text{get}\ s_i^1) = \text{act}_k\ (\text{get}\ s_i^2)$$

Proof. By strong induction on i:

If $i = 0$ then the precondition is never met.

Else let $(s_{i-1}^1, n_{i-1}^1, dir_{i-1}^1) := l_1[i-1]$, $(s_{i-1}^2, n_{i-1}^2, dir_{i-1}^2) := l_2[i-1]$. By Claim 11, $n_{i-1} := n_{i-1}^1 = n_{i-1}^2$ and $dir_{i-1} := dir_{i-1}^1 = dir_{i-1}^2$.

By Lem. 10 we either have:

- $s_i^1 = f_{down}(s_{i-1}^1, n_{i-1})$, $s_i^2 = f_{down}(s_{i-1}^2, n_{i-1})$ and $dir_{i-1} = \downarrow$

  If n_{i-1} is an Action node then:

  – If $n_{i-1}.k = k$:

$$\begin{aligned}
\text{act}_{n_{i-1}.k}\ (\text{get } s_i^1) &= \text{act}_{n_{i-1}.k}\ (\text{get (handleNode (get } s_{i-1}^1)\ n_{i-1})) \\
&= \text{act}_{n_{i-1}.k}\ (\text{get (handleNode (get } s_{i-1}^2)\ n_{i-1})) && \text{by Lem. 23} \\
&= \text{act}_{n_{i-1}.k}\ (\text{get } s_i^2)
\end{aligned}$$

  – Otherwise, k appears before $i-1$ in l_1 and l_2:

$$\begin{aligned}
\text{act}_k\ (\text{get } s_i^1) &= \text{act}_k\ (\text{get (handleNode (get } s_{i-1}^1)\ n_{i-1})) \\
&= \text{act}_k\ (\text{get } s_{i-1}^1) && \text{by Lem. 24} \\
&= \text{act}_k\ (\text{get } s_{i-1}^2) && \text{by IH} \\
&= \text{act}_k\ (\text{get (handleNode (get } s_{i-1}^2)\ n_{i-1})) && \text{by Lem. 24} \\
&= \text{act}_k\ (\text{get } s_i^2)
\end{aligned}$$

  If n_{i-1} is a Rollback node then

$$\begin{aligned}
\text{act}_k\ (\text{get } s_i^1) &= \text{act}_k\ (\text{beginRollback (get } s_{i-1}^1)) \\
&= \text{act}_k\ (\text{get } s_{i-1}^1) && \text{by Lem. 25} \\
&= \text{act}_k\ (\text{get } s_{i-1}^2) && \text{by IH} \\
&= \text{act}_k\ (\text{beginRollback (get } s_{i-1}^2)) && \text{by Lem. 25} \\
&= \text{act}_k\ (\text{get } s_i^2)
\end{aligned}$$

- $s_i^1 = f_{up}(s_{i-1}^1, n_{i-1})$, $s_i^2 = f_{up}(s_{i-1}^2, n_{i-1})$ and $dir_{i-1} = \uparrow$

  If n_{i-1} is an Action node then

$$\begin{aligned}
\text{act}_k\ (\text{get } s_i^1) &= \text{act}_k\ (\text{get } s_{i-1}^1) \\
&= \text{act}_k\ (\text{get } s_{i-1}^2) && \text{by IH} \\
&= \text{act}_k\ (\text{get } s_i^2)
\end{aligned}$$

  If n_{i-1} is a Rollback node then

$$\text{act}_k\ (\text{get } s_i^1) = \text{act}_k\ (\text{endRollback (get } s_{i-1}^1))$$

  By Claim 17 there exists an integer $0 \leq j < i-1$, well-defined states s_j^1, s_j^2 and tree *sub* such that $s_{i-1}^1 =$ traverse *sub* (beginRollback (get s_j^1)), $s_{i-1}^2 =$ traverse *sub* (beginRollback (get s_j^2)), $l_1[j] = (s_j^1, n_{i-1}, \downarrow)$, $l_2[j] = (s_j^2, n_{i-1}, \downarrow)$. Therefore

$$\begin{aligned}
\text{act}_k\ (\text{get } s_i^1) &= \text{act}_k\ (\text{endRollback (get (traverse } sub \text{ (beginRollback (get } s_j^1))))) \\
&= \text{act}_k\ (\text{get } s_j^1) && \text{by Lem. 26}
\end{aligned}$$

  If k appears before j in l_1 (and l_2 by Claim 28) then we can use the induction hypothesis and go backward.

$$\begin{aligned}
&= \text{act}_k\ (\text{get } s_j^2) && \text{by IH} \\
&= \text{act}_k\ (\text{endRollback (get (traverse } sub \text{ (beginRollback (get } s_j^2))))) && \text{by Lem. 26} \\
&= \text{act}_k\ (\text{get } s_i^2)
\end{aligned}$$

If k does not appear before j in l_1 and l_2, we can use Claim 29 to obtain the index $j \le j' < i$ such that j' is the first appearance of k in l_1 and l_2.

Since $j' < i$, by Claim 14, defined $(\text{proj}_1\ (l_1[j']))$, defined $(\text{proj}_1\ (l_2[j']))$, defined $(\text{proj}_1\ (l_1[j'+1]))$ and defined $(\text{proj}_1\ (l_2[j'+1]))$. By Lem. 10, $\text{proj}_1\ (l_1[j'+1]) = \text{handleNode}\ (\text{get}\ (\text{proj}_1\ (l_1[j'])))\ (\text{proj}_2\ (l_1[j']))$ and $(\text{proj}_2\ (l_1[j'])).k = k$. Similarly, $\text{proj}_1\ (l_2[j'+1]) = \text{handleNode}\ (\text{get}\ (\text{proj}_1\ (l_2[j'])))\ (\text{proj}_2\ (l_2[j']))$ and $(\text{proj}_2\ (l_2[j'])).k = k$. Therefore:

$$\begin{aligned}
\text{act}_k\ s_j^1 &= \text{act}_k\ (\text{get}\ (\text{proj}_1\ (l_1[j']))) && \text{by Cor. 31}\\
&= \text{act}_k\ (\text{get}\ (\text{proj}_1\ (l_2[j']))) && \text{by Cor. 22}\\
&= \text{act}_k\ (\text{get}\ s_j^2) && \text{by Cor. 31}\\
&= \text{act}_k\ (\text{endRollback}\ (\text{get}\ (\text{traverse}\ \mathit{sub}\ (\text{beginRollback}\ (\text{get}\ s_j^2))))) && \text{by Lem. 25}\\
&= \text{act}_k\ (\text{get}\ s_i^2)
\end{aligned}$$

□

Corollary 34. *Let tr be a tree of nodes, $init_1, init_2$ states, $0 \le i < 2 \cdot \text{size}\ tr - 1$ an integer, let $l_1 := \text{scan}\ tr\ init_1$, $l_2 := \text{scan}\ tr\ init_2$, $(s^1, n^1, dir^1) := l_1[i]$ and $(s^2, n^2, dir^2) := l_2[i]$. By Claim 11, $n := n^1 = n^2$. If n is an Action node, $n.k$ appears before i in l_1 and l_2,* defined s^1 *and* defined s^2, *then*

$$\text{defined}\ (\text{proj}_1\ (l_1[i+1])) \iff \text{defined}\ (\text{proj}_1\ (l_2[i+1]))$$

Proof. Consequence of Claim 33 and Lem. 21. □

Empty state traversal

Definition 35. *Let tr be a tree of nodes. We respectively define the mapping* **collect tr** *as* traverse tr [ ] $f_{collect}$ id *and* **collectTrace tr** *as* scan tr [ ] $f_{collect}$ id*, where*

$$f_{collect}(m, n) = \begin{cases} m + (n.k \to \text{mapping } n) & \textit{if } n \textit{ is an Action node with a well-defined key and } n.k \notin m \\ m & \textit{otherwise} \end{cases}$$

Definition 36. *We define the empty state ε_{tr} as the state whose rollback stack, locally created contracts set, consumed contracts set and local keys map are empty, but whose global key map is* collect tr

Claim 37. *Let tr be a tree of nodes and $0 \leq i < 2 \cdot \text{size } tr$ an integer*

$$\text{proj}_1\ ((\text{collectTrace } tr)[i]) \subseteq \text{collect } tr$$

Proof. By backward induction on i and applying Lem. 10. ☐

Claim 38. *Let tr be a tree of nodes, $0 \leq i < 2 \cdot \text{size } tr$ an integer and k a key.*

$$k \textit{ does not appear before } i \textit{ in } \text{collectTrace } tr \iff k \notin \text{proj}_1\ ((\text{collectTrace } tr)[i])$$

In particular

$$k \textit{ does not appear before } 2 \cdot \text{size } tr - 1 \textit{ in } \text{collectTrace } tr \iff k \notin \text{collect } tr$$

Proof. Induction on i, applying Lem. 10. ☐

Corollary 39. *Let tr be a tree of nodes, $0 \leq i < 2 \cdot \text{size } tr$ an integer, $n := \text{proj}_2\ ((\text{collectTrace } tr)[i])$. If n is an Action node with a well-defined key and i is the first appearance of $n.k$ in* collectTrace tr*, then*

$$(\text{collect } tr)[n.k] = \text{mapping } n$$

Proof. By Claim 38, $n.k \notin \text{proj}_1\ ((\text{collectTrace } tr)[i])$. By applying Lem. 10 we have that $(n.k \to \text{mapping } n) \in \text{proj}_1\ ((\text{collectTrace } tr)[i+1])$ if $i < 2 \cdot \text{size } tr - 1$ and $(n.k \to \text{mapping } n) \in \text{collect } tr$ otherwise. In the first case we make use of Claim 37 to prove the claim. ☐

Corollary 40. *Let tr be a tree of nodes and k a key. If $k \in \text{collect } tr$ then there exists an integer $0 \leq i < 2 \cdot \text{size } tr - 1$ such that i is the first appearance of k in* collectTrace tr.

Proof. By Claim 38, if $k \in \text{collect } tr$ then k does not appear before $2 \cdot \text{size } tr - 1$. Claim 29 concludes the proof. ☐

Corollary 41. *Let tr be a tree of nodes, $init$ a state, $0 \leq i < 2 \cdot \text{size } tr - 1$ an integer, let $l := \text{scan } tr\ init$, and $n := \text{proj}_2\ (l[i])$. If n is an Action node with a well-defined key, i is the first appearance of $n.k$ in l and* defined $(\text{proj}_1\ (l[i]))$ *then*

$$\text{defined } (\text{proj}_1\ (l[i+1])) \iff \text{act}_{n.k}\ (\text{get } init) = (\varepsilon_{tr}.globalKeys)[n.k]$$

Proof. Let $l_\varepsilon = \text{scan } tr\ \varepsilon_{tr}$. We know by Claim 11 that $n = \text{proj}_2\ (l_\varepsilon[i])$. and $\text{proj}_1\ (l_\varepsilon[i+1]) = \text{handleNode } (\text{proj}_1\ (l_\varepsilon[i]))\ n$. Furthemore by Claim 28, $n.k$ does not appear before i in l_ε. Finally by Cor. 15, if defined (traverse $tr\ \varepsilon_{tr}$) then defined $(\text{proj}_1\ (l_\varepsilon[i]))$.

$$
\begin{aligned}
&\text{act}_{n.k}\ (\text{get } init) = (\varepsilon_{tr}.globalKeys)[n.k] && \\
\iff\ &\text{act}_{n.k}\ (\text{get } init) = (\text{collect } tr)[n.k] && \\
\iff\ &\text{act}_{n.k}\ (\text{get } init) = \text{ mapping } n && \text{by Cor. 39} \\
\iff\ &\text{defined (handleNode (get (proj}_1\ (l[i])))\ n) && \text{by Cor. 32} \\
\iff\ &\text{defined (proj}_1\ (l[i+1])) && \text{by Lem. 10}
\end{aligned}
$$

□

Corollary 42. *Let tr be a tree of nodes, $init$ a state, $0 \leq i < 2 \cdot \text{size } tr - 1$ an integer, let $l := \text{scan } tr\ init$, and $n := \text{proj}_2\ (l[i])$. If n is an Action node with a well-defined key,* defined (traverse $tr\ \varepsilon_{tr}$)*, $n.k$ appears before i in l and* defined (proj$_1$ $(l[i])$) *then*

$$\text{defined (proj}_1\ (l[i+1]))$$

Proof. By Cor. 15, if defined (traverse $tr\ \varepsilon_{tr}$) then defined (proj$_1$ ((scan $tr\ \varepsilon_{tr})[i]$)) and defined (proj$_1$ ((scan $tr\ \varepsilon_{tr})[i+1]$)). Applying Cor. 34 concludes the proof. □

Final result. *Let tr be a tree of nodes and $init$ a well-defined state. If* defined (traverse $tr\ \varepsilon_{tr}$)*, then*

$$(\forall (k \to m) \in \varepsilon_{tr}.globalKeys,\ \text{act}_k\ (\text{get } init) = m) \iff \text{defined (traverse } tr\ init)$$

Proof. Let $l := \text{scan } tr\ init$.

($\Rightarrow$) direction: By Lem. 16, we only need to prove that for an arbitrary $0 \leq i < 2 \cdot \text{size } tr - 1$, defined (proj$_1$ $(l[i])$) $\implies$ defined (proj$_1$ $(l[i+1])$). If $n := \text{proj}_2\ (l[i])$ is a Rollback node, if $\text{proj}_3\ (l[i]) = \uparrow$ or if n is an Action node with no key, then by Lem. 10 (and Lem. 24 in the Action node case) this is automatically true.

If n is an Action node then either:

- i is the first appearance of $n.k$ in which case by Cor. 41 validates the claim
- $n.k$ appears before i in which case by Cor. 42, defined (proj$_1$ $(l[i+1])$)

($\Leftarrow$) direction: Assume there exists a key k such that $k \in \varepsilon_{tr}.globalKeys$ and $\text{act}_k\ (\text{get } init) \neq (\varepsilon_{tr}.globalKeys)[k]$. Then $k \in$ collect tr and by Cor. 40, there is a $0 \leq i < 2 \cdot \text{size } tr - 1$ such that i is the first appearance of k in collectTrace tr. In particular this means that $n := \text{proj}_2$ ((collectTrace $tr)[i]$) is an Action node with a well-defined key and $k = n.k$. By Cor. 41 this means that either ¬defined (proj$_1$ $(l[i])$) or ¬defined (proj$_1$ $(l[i+1])$), which both imply by Cor. 15 that ¬defined (traverse $tr\ init$). □