Semantics of Version Control Model of Replication

Definition 0.1 (Common Ancestor)**.** A version v is called a common ancestor of versions v_1 and v_2 in G iff $v \to^* v_1$ and $v \to^* v_2$ in G. If v_1 and v_2 are heads of branches b_1 and b_2, i.e., $H(b_1) = v_1$ and $H(b_2) = v_2$, then v is called a commin ancestor of branches b_1 and b_2

Definition 0.2 (Lowest Common Ancestor (LCA))**.** A version v is a lowest common ancestor of versions v_1 and v_2 in G iff:

- v is a common ancestor of v_1 and v_2 in G, and
- There does not exist another common ancestor v' of v_1 and v_2 such that $v \to^* v'$ in G.

LCA of a pair of branches is defined as the LCA of their heads.

Definition 0.3 (Initial Version Graph and State)**.** The graph $G_\odot = (\{v_\odot\}, \emptyset)$ is the initial version graph. The state $\Delta_\odot = (G_\odot, [v_\odot \mapsto b_\odot], [b_\odot \mapsto v_\odot], \emptyset)$ is the initial state.

Theorem 0.4 (**Uniqueness of LCA**)**.** *Assuming that we start with the initial state $\Delta_\odot$, the transition rules in Fig.* **??** *will always lead to a state $\Delta = ((V, E), B, H, L)$ where every distinct pair of branches, $b_1 \in dom(H)$ and $b_2 \in dom(H)$, has a single LCA given by $L(b_1, b_2)$, which is equal to $L(b_2, b_1)$.*

Proof. By induction on Δ. The initial state $\Delta_\odot$ vacuously satisfies the unique LCA condition. The inductive step has three cases:

- Case Commit: LCA function L is not updated in this case. Neither are any new branches created. Hence the proof follows from the induction hypothesis (IH).
- Case Fork: We fork off a new branch b' from b in this case. The LCA of b and b' in this case is clearly $H(b)$ – the version from which b' is forked. Indeed, this is the updated value of $L(b, b')$ in the conclusion.

  For every other branch b'', the LCA of b' and b' (i.e., $L(b', b'')$) is same as the LCA of b and b'' (i.e., $L(b, b'')$), which IH gurantees to be unique. This is because every ancestor of $H(b')$ is either equal of $H(b)$ or is an ancestor of $H(b)$, the hence lowest common ancestor of $H(b'')$ and $H(b)$ should also be the lowest common ancestor of $H(b'')$ and $H(b')$, which is indeed the updated value of $L(b, b'')$ in the conclusion.
- Case Merge: The proof follows from the explanation of Fig. 1
- Case FastFwd: Similar to Merge

□

Lemma 0.5 (Commit sets grow monotonically)**.** *Let $\Delta = ((V, E), N, C, H, L)$ be the state of a version-control system. Forall $v_1, v_2 \in V$, if $v_1 \to^* v_2$ then $C(v_1) \subseteq C(v_2)$.*

Proof. Straightforward induction on Δ. □

Remark The algebraic properties over sets useful for later proofs are stated below:

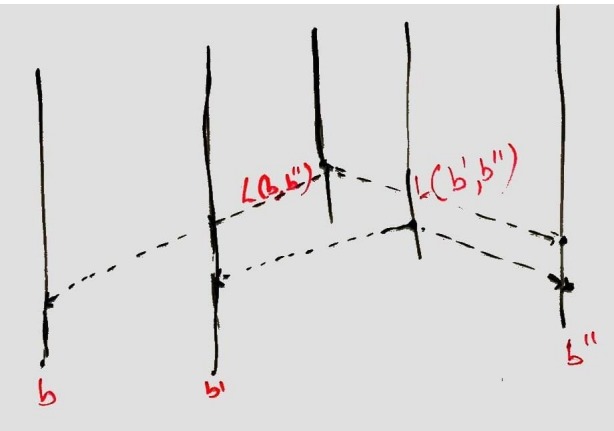

Figure 1: Explanation for the premise $L(b, b'') \rightarrow^* L(b', b'') \vee L(b', b'') \rightarrow^* L(b, b'')$ in FastFwd and Merge rules: Once you merge b' into b, every version $v \leq L(b', b'')$ will be a common ancestor of b and b''. Clearly, $L(b', b'')$ is the lowest among such common ancestors. The current lowest common ancestor of b and b'' is $L(b, b'')$. If $L(b', b'')$ and $L(b, b'')$ are not ancestrally-related, then we end up with two LCAs for b and b'' after the merge. Hence the premise

- **Law1**: Set difference distributes over union. $\forall(A, B, C).\ A - (B \cup C) = (A - B) \cap (A - C)$
- **Law2**: $\forall(A, B, C, D).\ (A - C) \cap (B - C) = \emptyset \wedge C \subseteq D \Rightarrow (A - D) \cap (B - D) = \emptyset$
- **Law3**: Set union distributes over difference. $\forall(A, B, C).\ (A \cup B) - C = (A - C) \cup (B - C)$
- **Law4**: Set union distributes over intersection. $\forall(A, B, C).\ (A \cup B) \cap C = (A \cap C) \cup (B \cap C)$
- **Law5**: $\forall(A, B, C, D).\ (A - C) \cap (B - C) = \emptyset \wedge A \subseteq B \Rightarrow B \subseteq C$

Lemma 0.6 (Commit sets modulo LCA are disjoint). *Let $\Delta = ((V, E), N, C, H, L)$ be the state of a version-control system. Forall distinct $b_1, b_2 \in dom(H)$, and $v_0, v_1, v_2 \in V$ s.t. $v_1 = H(b_1)$ and $v_2 = H(b_2)$ and $v_0 = L(b_1, b_2)$, the following is true: $(C(v_2) - C(v_0)) \cap (C(v_1) - C(v_0)) = \emptyset$.*

Proof. Proof by induction on Δ. We consider the cases below starting with then Merge case.

- Case Merge: In all the cases where $b_1 \neq b$ and $b_2 \neq b$, proof follows from IH because nothing changes for these branches in Δ'. We shall therefore focus on the case when one of b_1 or b_2 is b. Without loss of generality, let's assume $b_1 = b$. Two cases now depending on what b_2 is:
  - Case $b_2 = b'$: From the Merge rule, it's clear that $v_1 = H'(b) = v$ and $v_2 = H'(b') = H(b')$ and $v_0 = L'(b, b') = H(b') = v_2$. Since $v_2 = v_0$, $C(v_2) - C(v_0) = \emptyset$, hence the proof.
  - Case $b_2 \neq b'$. Let $b_2 = b'' \neq b'$. Hence $v_1 = H'(b)$, $v_2 = H'(b'') = H(b'')$, and $v_0 = L'(b, b'')$, which would either be $L(b, b'')$ or $L(b', b'')$ (see the premise of Merge). For notational convenience let's use v' to denote $H'(b') = H(b')$, v'' to denote $H'(b'') = H(b'')$, and v_{old} to denote $H(b)$. The hypotheses are listed below:

$$\begin{array}{ll} C(v_{old}) - C(L(b, b'')) \cap C(v'') - C(L(b, b'')) = \emptyset & IH1 \\ C(v') - C(L(b', b'')) \cap C(v'') - C(L(b', b'')) = \emptyset & IH2 \\ L(b, b'') \rightarrow^* L(b', b'') \vee L(b', b'') \rightarrow^* L(b, b'') & H1 \\ C(H(b')) \supset C(L(b, b')) & H2 \\ C(H(b)) \supset C(L(b, b')) & H3 \end{array}$$

  We will now destruct $H1$ and consider each disjunct separately.

* Case $L(b, b'') \rightarrow^* L(b', b'')$: From Lemma 0.5 we know:

$$C(L(b, b'')) \subseteq C(L(b', b'')) \quad H4$$

The Goal is to prove the following:

$$C'(v) - C'(L'(b, b'')) \ \cap \ C'(v'') - C'(L'(b, b'')) \ = \ \emptyset$$

We shall now rewrite the goal through a series of equalities. First, since $C'(v) = C(v_{old}) \cup C(v')$, and $L'(b, b'') = L(b', b'')$ (since $L(b, b'') \rightarrow^* L(b', b'')$ as per our assumption in this case), and $C'(v'') = C(v'')$:

$$(C(v_{old}) \cup C(v')) - C(L(b', b'')) \ \cap \ C(v'') - C(L(b', b'')) \ = \ \emptyset$$

Rewrite using Algebraic Law 3:

$$(C(v_{old}) - C(L(b', b'')) \ \cup \ C(v') - C(L(b', b''))) \ \cap \ C(v'') - C(L(b', b'')) \ = \ \emptyset$$

Rewrite using Algebraic Law 4:

$$\begin{array}{c} C(v_{old}) - C(L(b', b'')) \ \cap \ C(v'') - C(L(b', b'')) \\ \cup \ C(v') - C(L(b', b'')) \ \cap \ C(v'') - C(L(b', b'')) \ = \ \emptyset \end{array}$$

Rewrite using IH2:

$$C(v_{old}) - C(L(b', b'')) \ \cap \ C(v'') - C(L(b', b'')) \ = \ \emptyset$$

Applying Algebraic Law2 with $C := L(b, b'')$:

$$C(v_{old}) - C(L(b, b'')) \ \cap \ C(v'') - C(L(b, b'')) \ = \ \emptyset \ \ \wedge \ \ L(b, b'') \subseteq L(b', b'')$$

Now both conjuncts of the goal follow from hypotheses $IH1$ and $H4$, respectively.

* Case $L(b', b'') \rightarrow^* L(b, b'')$: From Lemma 0.5 we know:

$$C(L(b', b'')) \subseteq C(L(b, b'')) \quad H5$$

The Goal is to prove the following:

$$C'(v) - C'(L'(b, b'')) \ \cap \ C'(v'') - C'(L'(b, b'')) \ = \ \emptyset$$

We shall now rewrite the goal through a series of equalities. First, since $C'(v) = C(v_{old}) \cup C(v')$, and $L'(b, b'') = L(b, b'')$ (since $L(b', b'') \rightarrow^* L(b, b'')$ as per our assumption in this case), and $C'(v'') = C(v'')$:

$$(C(v_{old}) \cup C(v')) - C(L(b, b'')) \ \cap \ C(v'') - C(L(b, b'')) \ = \ \emptyset$$

Rewrite using Algebraic Law 3:

$$(C(v_{old}) - C(L(b, b'')) \ \cup \ C(v') - C(L(b, b''))) \ \cap \ C(v'') - C(L(b, b'')) \ = \ \emptyset$$

Rewrite using Algebraic Law 4:

$$\begin{array}{c} C(v_{old}) - C(L(b, b'')) \ \cap \ C(v'') - C(L(b, b'')) \\ \cup \ C(v') - C(L(b, b'')) \ \cap \ C(v'') - C(L(b, b'')) \ = \ \emptyset \end{array}$$

Rewrite using IH1:

$$C(v') - C(L(b, b'')) \ \cap \ C(v'') - C(L(b, b'')) \ = \ \emptyset$$

Apply Algebraic Law2 with $C := L(b', b'')$:

$$C(v') - C(L(b', b'')) \ \cap \ C(v'') - C(L(b', b'')) \ = \ \emptyset \ \ \wedge \ \ L(b', b'') \subseteq L(b, b'')$$

Now both conjuncts of the goal follow from hypotheses $IH2$ and $H5$, respectively.

- Case FastFwd: Fast forwarding is a special case of merge, hence the proof is going to be very similar to the Merge case. To reduce the FastFwd case to Merge, we make the following observations about the diferences between both the rules:
  - The subsumption relation $C(H(b)) \supset C(L(b, b'))$ in the premise of Merge is replaced with equality in FastFwd, but this is irrelevant is the subsumption premise is never used in the proof of Merge case.
  - $N'(v) = N(H(b'))$ in FastFwd whereas it is computed using merge in Merge. However, the current theorem says nothing about N, and the rest of Δ is independent of N, hence this is irrelevant.
  - $C'(H'(b)) = C(H(b'))$ in the conclusion of FastFwd whereas it is equal to $C(H(b)) \cup C(H(b'))$ in Merge. However since $C(H(b)) = C(L(b, b')) \subset C(H(b'))$ in the premise of FastFwd, $C'(H'(b))$ in the conclusion is indeed equal to $C(H(b)) \cup C(H(b'))$ like in Merge.

  Consequently, we can reuse the proof of Merge as a proof of FastFwd.
- Case Fork: Branch b' is forked from branch b. In all the cases where $b_1 \neq b'$ and $b_2 \neq b'$, proof follows from IH because nothing changes for these branches in Δ'. We shall therefore focus on the case when one of b_1 or b_2 is b'. Without loss of generality, let's assume $b_2 = b$. We now have two cases depending on what values b_1 can take:
  - Case $b_1 = b$: In this case $L'(b_1, b_2) = L'(b, b') = H(b) = H'(b)$. Hence, $C'(H'(b_1)) - L'(b_1, b_2)$ is equal to $C'(H'(b)) - C'(H'(b)) = \emptyset$. Proof follows.
  - Case $b_1 = b'' \neq b$: In this case, the proof follows straightforwardly from the inductive hypothesis that asserts the lemma for b and b'' in Δ because $L'(b', b'') = L(b, b'')$ and $C(H(b')) = C(H(b))$.
- Case Commit: Proof is trivial because c is a fresh commit id that does not exist in Δ.

□

We would now like to prove that there cannot be two versions/vertices associated with the same set of commit Ids in C.

Theorem 0.7 (Convergence). *Let $\Delta = ((V, E), N, C, H, L)$ be the state of a version-control system. Forall distinct $b_1, b_2 \in dom(H)$, and $v_1, v_2 \in V$ s.t. $v_1 = H(b_1)$ and $v_2 = H(b_2)$, the following is true: $C(v_1) = C(v_2) \Rightarrow N(v_1) = N(v_2)$.*

Proof. Proof by induction on Δ. Base case trivially satisfies as there are no two branches in $\Delta_{\odot}$. Inductive proof has four cases corresponding to the four rules in Fig. **??**:

- Case Commit: Branch b is extended with a fresh commit c, hence there does not exist another branch whose head has same commit set as $H'(b)$.
- Case Fork: In all the cases where $b_1 \neq b'$ and $b_2 \neq b'$, proof follows from IH because nothing changes for these branches in Δ'. We shall therefore focus on the case when one of b_1 or b_2 is b'. Without loss of generality, let's assume $b_2 = b$. Two cases now depending on what b_1 is:
  - Case $b_1 = b$: Proof by definition as Fork defines $C'(H'(b')) = C(H(b)) = C'(H'(b))$ and $N'(H'(b')) = N(H(b)) = N'(H'(b))$.
  - Case $b_1 = b'' \neq b$: Proof follows from IH relating b_1 and b in Δ because $C'(H'(b')) = C(H(b))$ and $N'(H'(b')) = N(H(b))$.
- Case FastFwd: The proof is similar to the Fork case since $C'(H'(b)) = C(H(b')) = C'(H'(b'))$ and $N'(H'(b)) = N(H(b')) = N'(H'(b'))$.

- Case MERGE: In all the cases where $b_1 \neq b$ and $b_2 \neq b$, proof follows from IH because nothing changes for these branches in Δ'. We shall therefore focus on the case when one of b_1 or b_2 is b. Without loss of generality, let's assume $b_1 = b$. Two cases now depending on what b_2 is:
  - Case $b_2 = b'$: The relevant premise of MERGE is reproduced below:

    $$C(H(b)) \supset C(L(b, b')) \quad H0$$

    We shall prove that $C'(H'(b_1))$ cannot be equal to $C'(H'(b_2))$ by first assuming that they are equal and then deriving a contradiction.

    $$C'(H'(b_1)) = C'(H'(b_2)) \quad H1$$

    Since $b_1 = b$, and $b_2 = b'$, and $C'(H'(b')) = C(H(b'))$:

    $$C'(H'(b)) = C(H(b'))$$

    Since $C'(H'(b')) = C(H(b)) \cup C(H(b'))$:

    $$C(H(b)) \cup C(H(b')) = C(H(b')) \quad H2$$

    From Lemma 0.6 on Δ, we know the following:

    $$C(H(b)) - C(L(b, b')) \ \cap \ C(H(b')) - C(L(b, b')) \ = \ \emptyset \quad H3$$

    Rewriting H3 using H2:

    $$C(H(b)) - C(L(b, b')) \ \cap \ (C(H(b)) \cup C(H(b')) - C(L(b, b')) \ = \ \emptyset$$

    Rewriting using Algebraic Law3:

    $$C(H(b)) - C(L(b, b')) \ \cap \ (C(H(b)) - C(L(b, b')) \ \cup \ C(H(b')) - C(L(b, b'))) \ = \ \emptyset$$

    Rewriting using Algebraic Law4:

    $$(C(H(b)) - C(L(b, b')) \ \cap \ C(H(b)) - C(L(b, b'))) \ \cup \ (C(H(b)) - C(L(b, b')) \ \cap \ C(H(b')) - C(L(b, b'))) \ = \ \emptyset$$

    Simplifying, and Rewriting using $H3$:

    $$C(H(b)) - C(L(b, b')) \ = \ \emptyset$$

    Which is possible if and only if:

    $$C(H(b)) \subseteq C(L(b, b')) \quad H4$$

    $H0$ and $H4$ now result in a contradiction.
  - Case $b_2 = b'' \neq b'$: Premises of MERGE:

    $$\begin{array}{ll} C(H(b)) \supset C(L(b, b')) & H1 \\ C(H(b')) \supset C(L(b, b')) & H2 \\ L(b, b'') \rightarrow^* L(b', b'') \ \vee \ L(b', b'') \rightarrow^* L(b, b'') & H3 \end{array}$$

    From the conclusion of MERGE:

    $$\begin{array}{ll} C'(H'(b)) = C(H(b)) \cup C(H(b')) & H4 \\ C'(H'(b'')) = C(H(b'')) & H5 \end{array}$$

    And the premise of the theorem:

    $$C'(H'(b)) = C'(H'(b''))$$

    Due to $H5$, this is equivalent to:

    $$C'(H'(b)) = C(H(b'')) \quad H6$$

From Lemma 0.6 on $H'(b')$ and $H'(b'') = H(b'')$:

$$C'(H'(b)) - C'(L'(b,b'')) \ \cap \ C(H(b'')) - C'(L'(b,b'')) \ = \ \emptyset$$

Rewriting using $H6$:

$$C'(H'(b)) - C'(L'(b,b'')) \ = \ \emptyset$$

Hence:

$$C'(H'(b)) \subseteq C'(L'(b,b''))$$

Rewriting using $H4$:

$$C(H(b)) \cup C(H(b')) \ \subseteq \ C'(L'(b,b'')) \quad H7$$

Let us distruct $H3$ and consider the two disjuncts separately:

* Case $L(b,b'') \rightarrow^* L(b',b'')$: From the conclusion of MERGE:

  $$L'(b,b'') = L(b',b'') \quad H8$$

  Rewriting $H7$ using $H8$:

  $$C(H(b)) \cup C(H(b')) \ \subseteq \ C(L(b',b'')) \quad H7$$

  Which tells us:

  $$C(H(b)) \subseteq C(L(b',b'')) \quad H9$$

  Note that $L(b',b'')$ is LCA of $H(b')$ and $H(b'')$, hence $L(b',b'') \rightarrow^* H(b')$. From Lemma 0.5, we get:

  $$C(L(b',b'')) \subseteq C(H(b')) \quad H10$$

  From $H9$ and $H10$:

  $$C(H(b)) \subseteq C(H(b')) \quad H11$$

  Lemma 0.6 on $H(b)$ and $H(b')$ tells us:

  $$C(H(b)) - C(L(b,b')) \ \cap \ C(H(b')) - C(L(b,b')) \ = \ \emptyset \quad H12$$

  Applying Algebraic Law 5 on $H11$ and $H12$:

  $$C(H(b')) \subseteq C(L(b,b'))$$

  Which contradicts $H2$. Hence $C'(H'(b))$ cannot be equal to $C'(H'(b''))$.

* Case $L(b',b'') \rightarrow^* L(b,b'')$: Very similar to the above case. Using the similar approach as above, we derive $C(H(b')) \subseteq C(H(b))$ (the opposite of $H11$), which leads us to $C(H(b)) \subseteq C(L(b,b'))$, thus contradicting $H1$.

□