ER Catcher - Appendix

May 2020

1 Static Happens-Before Formulation

In this section, we first formulate happens-before relation and its properties in event-based systems, then prove Static Vector Clock is able to report sound happens-before relations.

1.1 Happens-Before

Happens-Before (HB) is a strict partially ordered set on state nodes N which is the set of m_s (start), m_e (end), and m_i (invoke) for all $m \in M$ where M is the set of all asynchronous tasks. The HB relation is defined as:

$$n_1 \prec n_2 \iff (H(n_2) \Rightarrow H(n_1) \land HBD(n_1, n_2)$$

where $H(n_1)$ means the node n_1 is executed (or happens) and $HBD(n_1, n_2)$ means the node n_1 happens before node n_2 in all possible orders of execution in run time. Additionally, threads can be FIFO which means the tasks (callback methods) are enqueued sequentially. Each task m is associated with a delay m_d (which is 0 by default) and the start state of m happens at least m_d seconds later than the invocation of m. Moreover, we assume tasks in FIFO threads cannot be deleted prior to their executions.

Let say $HB^*(N)$ is the complete and sound happens-before ordered set on nodes N: it has all the correct happens-before relations and no single one of the relations are incorrect. If an app is executed in real world, two nodes with a correct happens-before relation always will be executed in that order.

Theorem 0. *HB^* satisfies the properties below ($M(t)$ is the set of tasks that run in thread t).*

- *Irreflexivity*

$$\forall n \in N : n \nprec n$$

- *Transitivity:*

$$\forall n, n', n'' \in N : n \prec n' \land n' \prec n'' \Rightarrow n \prec n''$$

- *Asymmetry:*
$$\nexists n, n' \in N : n \prec n' \wedge n' \prec n$$

- *Intra-Task Order or* ***principles 1 and 2*** *(m_B is the invocation nodes that executed in task m and $SUCC(n)$ is the set of outgoing neighbors of node n):*
$$\forall m \in M(t) : m_i \prec m_s \prec m_n \wedge$$
$$\forall n \in m_B : m_s \prec n \prec m_e \wedge (\forall n' \in SUCC(n) : n \prec n')$$

- *Task Atomicty:*
$$\nexists m, m' \in M(t) : (m'_s \prec m_e \wedge m'_s \not\prec m_s) \vee (m'_e \prec m_e \wedge m'_e \not\prec m_s)$$

- *Same Task-Queue Order or* ***principle 3*** *(m_d is the delay between invocation and start of task m):*
$$\textit{if } t \textit{ is a FIFO queue} \Rightarrow \forall m, m' \in M(t) : m_i \prec m'_i \wedge m_d \prec m'_d \Rightarrow m_e \prec m'_s$$

The following lemmas prove the correctness of each aforementioned property. Since the first property, Irreflexivity, is trivial (no state happens-before itself), we did not provide a proof for it.

Lemma 1. *HB^* is transitive or*
$$\forall n, n', n'' \in N : n \prec n' \wedge n' \prec n'' \Rightarrow n \prec n''$$

Proof. For any three nodes n_1, n_2, and n_3 we have: n $\prec n' \wedge n' \prec n''$
$\equiv (H(n') \Rightarrow H(n) \wedge HBD(n, n'))$
$\wedge (H(n'') \Rightarrow H(n') \wedge HBD(n', n''))$
$\equiv (H(n'') \Rightarrow H(n) \wedge HBD(n, n')) \wedge HBD(n', n''))$
$\equiv (H(n'') \Rightarrow H(n) \wedge HBD(n, n''))$
$\equiv n \prec n''$
□

Lemma 2. *HB^* is asymmetry or*
$$\nexists n, n' \in N : n \prec n' \wedge n' \prec n$$

Proof. We prove this lemma by contradiction. $\exists n, n' \in N : n \prec n' \wedge n' \prec n$
$\equiv \exists n, n' \in N : n \prec n(Transitivity)$
which leads to $n \prec n$ that contradicts Irreflexivity property. Then, the lemma is proved. □

Lemma 3. *HB^* satisfies Intra-Task Order property or :*
$$\forall m \in M(t) : m_i \prec m_s \wedge m_s \prec m_e$$
and
$$\forall n \in m_B : m_s \prec n \wedge n \prec m_e \wedge (\forall n' \in SUCC(n) : n \prec n')$$

Proof. The first part is trivial since based on our assumption, the tasks are not deleted once they invoked and the end of a task cannot happens before the start. For the second part, since $SUCC(n)$ are the edges of a dominance directed graph of the control flow graph (according to the definition of edges in CSF), every path started from the entry point of the method (m_s) and reaches to n' must pass the node n, we can say $H(n') \Rightarrow H(n) \wedge HBD(n, n')$. $\square$

Lemma 4. *HB^* satisfies Task Atomicty property or :*

$$\nexists m, m' \in M(t) : (m'_s \prec m_e \wedge m'_s \not\prec m_s) \vee (m'_e \prec m_e \wedge m'_e \not\prec m_s)$$

Proof. We prove this lemma by contradiction. If there exists such a pair, it means in the middle of execution of one task (when it is not finished) another task starts on the same thread. Since there exists atomicity property for execution of tasks in a single thread, this scenario cannot be happened. As a result, the lemma is proved. $\square$

Lemma 5. *HB^* satisfies Same Task-Queue Order property or :*

$$\textit{if } t \textit{ is a FIFO queue} \Rightarrow \forall m, m' \in M(t) : m_i \prec m'_i \wedge m_d \prec m'_d \Rightarrow m_e \prec m'_s$$

Proof. We prove this lemma by contradiction. Let say there is a real-world scenario that $m_i \prec m'_i \wedge m_d \leq m'_d$ but $m_e \not\prec m'_s$. In other words, if we order the nodes based on the time of their executions m'_s will be places before m_e. Assume *time* is the indicator of the time of execution of each node. Because of $m_i \prec m'_i$ we know that $time(m_i) < time(m'_i)$. The time that a task is enqueued in the target thread is $time(m_i) + m_d$. As a result, we have

$$time(m_i) < time(m'_i) \wedge m_d \leq m'_d \Rightarrow time(m_i) + m_d < time(m'_i) + m'_d$$

Also, we know $time(m'_s) < time(m_e)$. Because of the Task Atomicty property, $time(m'_e) < time(m_s)$. As a result, we have $time(m'_s) < time(m_s)$ which contradicts the FIFO property of the task queue (because $time(m_i) + m_d < time(m'_i) + m'_d$). Therefore, the lemma is proved. $\square$

As a result of these lemmas, Theory 0 is proved.

Corollary 1. *A relation $HB(N)$ is sound (it does not have any false happens-before relation) if $HB(n) \subseteq HB^*(N)$.*

1.2 Static Vector Clock

Now, we show why the properties of SVC resemble a sound happens-before relation set. Formally, in an event-based system with $|T|$ threads, $SVC(n) =< S_1, \cdots, S_{|T|} >$, where S_i is the minimized set of states running in thread T_i happening before state n. S_i is minimized when there does not exist any happens-before relation between any of its members. $SVC(n)_i$ is minimized if for any node $n_p \in SVC(n)_i$ there is no other node $n_q \in SVC(n)_i$ that n_q happens before n_j ($\nexists n_p, n_q \in SVC(n)_i : n_q \prec n_p$).

Theorem 1. *State a_1 happens before a_n if there exists a sequence of nodes $(a_1, \cdots, a_n)$ such that $\forall 1 \leq i < n a_i \in SVC(a_{i+1})$.*

Proof. If there exists a sequence of nodes $(a_1, \cdots, a_n)$ such that $\forall 1 \leq i < n a_i \in SVC(a_{i+1})$ we can apply the definition of SVC on the sequence $(a_i \in SVC(a_{i+1}) \Rightarrow a_i \prec a_{i+1})$ which results $a_1 \prec a_n$.

Note that the reverse of this lemma (if $a_1 \prec a_n$ then there exists a sequence of nodes $(a_1, \cdots, a_n)$ such that $\forall 1 \leq i < n a_i \in SVC(a_{i+1})$) is only true when SVC resembles HB^*. □

Theorem 2. *Two states a and b may happen without any order if there exists two threads, t_1 and t_2, such that $SVC(a)_{t_1} \prec SVC(b)_{t_1}$ and $SVC(b)_{t_2} \prec SVC(a)_{t_2}$*

Proof. We prove this lemma by contradiction. Let say there is a happens-before relation between a and b (let say $a \prec b$ since they are selected arbitrary). Since happens-before is transitive (Lemma 1), all nodes that happens-before a (in any thread) must happens-before b. In other words $\forall t \forall n \in SVC(a)_t \Rightarrow n \in SVC(b)_t$. Let $n_b \in SVC(b)_{t_2}$ and $n_a \in SVC(a)_{t_2}$ that $n_b \prec n_a$. According to our assumption we know $a \prec b$ and because of transitivity we have $n_b \prec n_a \prec a \prec b$. Since $SVC(b)_{t_2}$ is minimized (according to the definition of SVC), n_b cannot be a member of $SVC(b)_{t_2}$. As a result of this contradiction, a and b cannot have a happens-before relation. □

And finally, we prove why the proposed algorithm for calculation SVC (Algorithm1) produces a sound SVC.

Theorem 3. *The static vector clock calculated by Algorithm1, SVC, is sound. In other words, the happens-before relation extracted from SVC is a subset of $HB*(N)$.*

Algorithm1 consists of two part, first it calculates SVC^0 then apply a topological order traverse until it reaches a fix point. First we prove SVC^0 is sound, then we show the second part of the algorithm only adds correct happens-before relations which keeps SVC sound.

Lemma 6. *The initialized SVC also known as SVC^0 is sound.*

Proof. According to Algorithm1, SVC^0 captures the Intra-Task Order property (Lemma 3). Therefore, no false happens-before related exists in SVC^0 and it is sound. □

Lemma 7. *Each iteration over reversed topological order of SVC in Algorithm1 does not add any false happens-before relation.*

Proof. In an iteration in Algorithm1, there is only one modification on SVC where $SVC(m_s)_t \leftarrow SVC(m_s)_t \cup m'.e$. In this case, $m'.i \prec m.i$ and their thread is a FIFO queue. Therefore, the introduced happens-before relation between m_s and m'_e can be inferred from Same Task-Queue Order property of happens-before relations (Lemma 5). Since there is no other addition to SVC (minimize procedure does not add any new node to SVC), all happens-before relations added because of changes in SVC are correct. $\square$

According to these lemmas, Theorem 3 is proved.