On safety in distributed computing

Srivatsan Ravi

Safety in distributed computing

1. Something "bad" never happens
2. Some invariant holds at every step in the execution
3. If something bad happens in an execution, it happens because of some particular step in the execution

Safety properties

1. A *property* is a set of histories
2. What does it mean for a set of histories exported by a concurrent implementation to be safe?

Defining Safety

1. The Alpern-Schneider topology
2. The Lynch definition

Defining safety: Alpern-Schneider Topology

Alpern-Schneider Topology

A property O is *finitely observable* iff:

$$\forall H \in \mathcal{H}_{inf}\colon\ H \in O \Rightarrow (\exists H' \in \mathcal{H}_{fin}; H' < H \wedge (\forall H'' \in \mathcal{H}_{inf}; H' < H'',\ H'' \in O))$$

1. If $O_1, O_2, \ldots, O_n$ are finitely observable, then $\cap_{i=1}^{n} O_i$ is also finitely observable
2. The potentially infinite union of finitely observable properties is also finitely observable.

Defining safety: Alpern-Schneider Topology

Alpern-Schneider Topology

A property O is *finitely observable* iff:

$$\forall H \in \mathcal{H}_{inf}\colon\ H \in O \Rightarrow (\exists H' \in \mathcal{H}_{fin}; H' < H \wedge (\forall H'' \in \mathcal{H}_{inf};\ H' < H'',\ H'' \in O))$$

1. If $O_1, O_2, \ldots, O_n$ are finitely observable, then $\cap_{i=1}^{n} O_i$ is also finitely observable
2. The potentially infinite union of finitely observable properties is also finitely observable.

The set $\mathcal{O}$ of finitely observable properties is a topology on $\mathcal{H}_{inf}$

Defining safety: Alpern-Schneider Topology

Alpern-Schneider Topology

- *Safety properties* are the *closed sets* in the topology
  - A set if closed if its complement is open
  - A closed set contains all its *limit-points*
- AS-topology defined on the set of infinite histories
- Notion of safety not defined for finite histories

Formal definition of safety

Safety property [Lynch, Distributed Algorithms]

- every prefix H' of a history $H \in \mathcal{P}$ is also in $\mathcal{P}$
  - *prefix-closure*: an incorrect execution cannot turn into a correct one in the future

Formal definition of safety

Safety property [Lynch, Distributed Algorithms]

- every prefix H' of a history $H \in \mathcal{P}$ is also in $\mathcal{P}$
  - *prefix-closure*: an incorrect execution cannot turn into a correct one in the future
- for any infinite sequence of finite histories $H^0, H^1, \ldots$ such that for all i, $H^i \in \mathcal{P}$ and H^i is a prefix of H^{i+1}, the infinite history that is the *limit* of the sequence is also in $\mathcal{P}$.
  - *limit-closure*: the infinite limit of an ever-extending safe execution must be also safe.

Formal definition of safety

Safety property [Lynch, Distributed Algorithms]

- every prefix H' of a history $H \in \mathcal{P}$ is also in $\mathcal{P}$
  - *prefix-closure*: an incorrect execution cannot turn into a correct one in the future
- for any infinite sequence of finite histories $H^0, H^1, \ldots$ such that for all i, $H^i \in \mathcal{P}$ and H^i is a prefix of H^{i+1}, the infinite history that is the *limit* of the sequence is also in $\mathcal{P}$.
  - *limit-closure*: the infinite limit of an ever-extending safe execution must be also safe.

Sufficient to prove all finite histories are safe

Proving a property to be safe

Prefix-closure

Constructively from the extended history

Limit-closure

Application of *König's Path Lemma*:
If G is an infinite connected finitely branching rooted directed graph, then G contains an infinite sequence of non-repeating vertices starting from the root

Limit-closure

1. A property that is not limit-closed
2. Proving limit-closure of safety properties using *König's Path Lemma*

Multi-objects

Transactions

- Sequence of *abortable reads* and *writes* on *objects*
- Transactions can *commit* by invoking *tryC* (*take effect*) or *abort*

Multi-objects

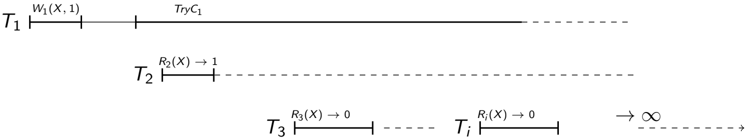
Transactions

- Sequence of *abortable reads* and *writes* on *objects*
- Transactions can *commit* by invoking *tryC* (*take effect*) or *abort*

Opacity

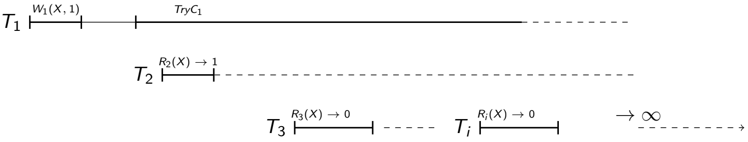
1. History is *opaque* if there exists an equivalent *completion* that is legal and respects the real-time order of transactions.
   - Totally-order transactions such that every t-read returns the value of the latest written t-write.
2. *Completion* by including matching responses to incomplete t-operations and aborting incomplete transactions

Opacity and limit-closure



1. Mutually overlapping transactions
2. Suppose a serialization S of H exists
    - There exists $n \in \mathbb{N}$; $seq(S)[n] = T_1$
    - Consider the transaction T_i at index $n+1$
    - For any $i \geq 3$, T_i must precede T_1 in any serialization

Opacity and limit-closure



1. Consider the set of histories in which every transactional operation is complete in the infinite history?
2. Is the resulting property limit-closed?

Opacity and limit-closure: Prelude to the proof

Live set of T

$Lset_H(T)$: T and every transaction T' such that neither the last event of T' precedes the first event of T in H nor the last event of T precedes the first event of T' in H.

T' succeeds the live set of T ($T \prec_H^{LS} T'$) if for all $T'' \in Lset_H(T)$, T'' is complete and the last event of T'' precedes the first event of T'.

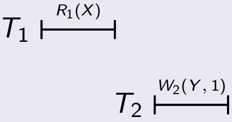
Opacity and limit-closure: Prelude to the proof

Live set of T

$Lset_H(T)$: T and every transaction T' such that neither the last event of T' precedes the first event of T in H nor the last event of T precedes the first event of T' in H.

T' succeeds the live set of T ($T \prec_H^{LS} T'$) if for all $T'' \in Lset_H(T)$, T'' is complete and the last event of T'' precedes the first event of T'.

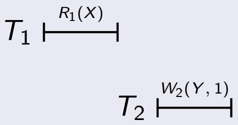
Live set: An example



- T_1 and T_2 overlap
- *Live set of* $T_1 = \{T_1\}$
- T_2 *succeeds the live set of* T_1

Opacity and limit-closure: Prelude to the proof

Live set: An example



We can find a serialization in which T_1 precedes T_2

Given any serialization of a du-opaque history, permute transactions without rendering any t-read illegal.

Lemma

Let H be a finite opaque history and assume $T_k \in txns(H)$ be a complete transaction in H such that every transaction in $Lset_H(T_k)$ is complete in H. Then there exists a serialization S of H such that for all $T_k, T_m \in txns(H)$; $T_k \prec_H^{LS} T_m$, we have $T_k <_S T_m$.

Opacity and limit-closure: The proof

Step 1: Construction of rooted directed graph G_H

Vertices of G_H

- Root vertex: (H^0, S^0) (empty histories)
- Non-root vertex: (H^i, S^i)
- S^i is a serialization of H^i
- S^i respects *live set* relation

Opacity and limit-closure: The proof

Step 1: Construction of rooted directed graph G_H

Vertices of G_H

- Root vertex: (H^0, S^0) (empty histories)
- Non-root vertex: (H^i, S^i)
- S^i is a serialization of H^i
- S^i respects *live set* relation

Edges of G_H

- $cseq_i(S^j)$; $j \geq i$: subsequence of $seq(S^j)$ reduced to transactions that are *complete in* H^i *w.r.t* H
- $(H^i, S^i) \rightarrow (H^{i+1}, S^{i+1})$ if $cseq_i(S^i) = cseq_i(S^{i+1})$

G_H is finitely branching

Out-degree of (H^i, S^i) bounded by the number of possible permutations of the set $txns(S^{i+1})$.

Opacity and limit-closure: The proof

Step 2: Application of *König's Path Lemma*

If G is an infinite connected finitely branching rooted directed graph, then G contains an infinite sequence of non-repeating vertices starting from the root.

G_H is finitely branching

Out-degree of (H^i, S^i) bounded by the number of possible permutations of the set $txns(S^{i+1})$.

G_H is connected

- Given (H^{i+1}, S^{i+1}), $\exists$ (H^i, S^i): $seq(S^i)$ is subsequence of $seq(S^{i+1})$
- $seq(S^{i+1})$ contains every complete transaction that takes its last step in H in H^i
- $cseq_i(S^i) = cseq_i(S^{i+1})$
- Iteratively construct a path from (H^0, S^0) to each (H^i, S^i)

Opacity and limit-closure: The proof

Step 2: Application of *König's Path Lemma*

G_H is an infinite finitely branching connected rooted directed graph

- G_H is infinite (by construction)
- Apply *König's Path Lemma* to G_H
  - Derive infinite sequence $\mathcal{L}$ of non-repeating vertices of G_H starting from root

Opacity and limit-closure: The proof

Step 2: Application of *König's Path Lemma*

G_H is an infinite finitely branching connected rooted directed graph

- G_H is infinite (by construction)
- Apply *König's Path Lemma* to G_H
  - Derive infinite sequence $\mathcal{L}$ of non-repeating vertices of G_H starting from root

$$\mathcal{L} = (H^0, S^0), (H^1, S^1), \ldots, (H^i, S^i), \ldots$$

$$\downarrow$$

$$\text{In } \mathcal{L},\ \forall j > i : cseq_i(S^i) = cseq_i(S^j)$$

Opacity and limit-closure: The proof

Step 3: Define a bijective mapping from $txns(H)$ to $\mathbb{N}$

$$f : \mathbb{N} \rightarrow txns(H) :$$

$$f(1) = T_0$$

$$\forall k \in \mathbb{N} \setminus \{1\} : f(k) = cseq_i(S^i)[k]; i = min\{\ell \in \mathbb{N} | \forall j > \ell : cseq_\ell(S^\ell)[k] = cseq_j(S^j)[k]\}$$

Step 3: Define a bijective mapping from $txns(H)$ to $\mathbb{N}$

$$f : \mathbb{N} \rightarrow txns(H) :$$

$$f(1) = T_0$$

$$\forall k \in \mathbb{N} \setminus \{1\} : f(k) = cseq_i(S^i)[k]; i = min\{\ell \in \mathbb{N} | \forall j > \ell : cseq_\ell(S^\ell)[k] = cseq_j(S^j)[k]\}$$

$$\Downarrow$$

Index of a transaction that is complete w.r.t H is *fixed*

Opacity and limit-closure: The proof

Step 3: Define a bijective mapping from $txns(H)$ to $\mathbb{N}$

f is *bijective*

- for every $T \in txns(H), \exists k$: $f(k) = T$
- for every k, m: $f(k) = f(m) \Rightarrow k = m$

Why?

- Suppose $cseq_i(S^i) = [1, 2, \ldots, k, \ldots]$
- If last step of T_k in H is in H^i, for all $j > i$:
  - $cseq_j(S^j) = [1, 2, \ldots, k, \ldots]$
  - T_k remains in the same position in any extension!

Opacity and limit-closure: The proof

Step 4: Construct a serialization S of H from f

f is *bijective*

- for every $T \in txns(H)$, $\exists k$: $f(k) = T$
- for every k, m: $f(k) = f(m) \Rightarrow k = m$

$$\Downarrow$$

$\mathcal{F} = f(1), f(2), \ldots, f(i), \ldots$ is an infinite sequence of transactions.

Step 4: Construct a serialization S of H from f

$\mathcal{F} = f(1), f(2), \ldots, f(i), \ldots$ is an infinite sequence of transactions.

Opacity and limit-closure: The proof

Step 4: Construct a serialization S of H from f

$\mathcal{F} = f(1), f(2), \ldots, f(i), \ldots$ is an infinite sequence of transactions.

And finally,

Constructing S

- $seq(S) = \mathcal{F}$
- for each t-complete transaction T_k in H, $S|k = H|k$
- each complete T_k, but not t-complete in H, $S|k = H|k \cdot tryA_k \cdot A_k$

Opacity and limit-closure: The proof

Step 5: Prove S is a serialization of H

Constructing S

- $seq(S) = \mathcal{F}$
- for each t-complete transaction T_k in H, $S|k = H|k$
- each complete T_k, but not t-complete in H, $S|k = H|k \cdot tryA_k \cdot A_k$

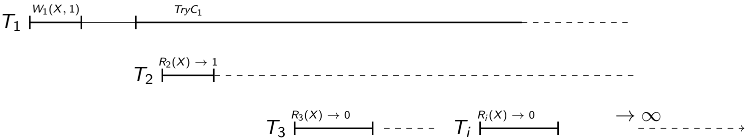
S is a serialization of H

- S is equivalent to some t-completion of H
- Every t-complete prefix of S is a serialization of some complete subsequence of a prefix of H
  - S is *legal*
  - S respects the *real-time order* of H
  - every t-read is legal in corresponding *local serialization*

Opacity and safety

1. Under restriction that every transaction issues only finitely many t-operations and is eventually complete, opacity is a safety property
2. Take a TM implementation M in which every transactional is complete in the infinite history. Then, sufficient to prove every finite history of M is opaque

Defining safety for infinite histories



1. Define an infinite history H to be opaque *iff* every finite prefix of H (including H itself if finite) is final-state opaque
2. Prefix-closed and limit-closed by definition
3. But no serialization defined for the infinite history. Does this matter?

Linearizability

Data type

1. Specified as *Mealy machine*
   - In response to an input, the object makes a transition from one state to another and responds with an output
   - Object transitions from one state to another after an operation specified by the *sequential specification*

Linearizability

Data type

1. Specified as *Mealy machine*
   - In response to an input, the object makes a transition from one state to another and responds with an output
   - Object transitions from one state to another after an operation specified by the *sequential specification*

1. A history H is linearizable w.r.t data type τ if there exists a sequential history equivalent to *some completion of H* that is consistent with the *sequential specification of τ* and respects the *real-time order* of operations in H
2. *Completion* by removing invocations or adding matching responses

Linearizability is a safety property

Step 1: Construction of rooted directed graph G_H

Vertices of G_H

- Root vertex: (H^0, L^0) (empty histories)
- Non-root vertex: (H^i, L^i)
- L^i is a linearization of H^i

Edges of G_H

- $(H^i, L^i) \rightarrow (H^{i+1}, L^{i+1})$ if $cseq_i(L^i)$ is a subsequence of $cseq_i(L^{i+1})$

Linearizability is a safety property

Step 2: Application of *König's Path Lemma*

G_H is finitely branching

Out-degree of (H^i, L^i) is finite for *finite types*

G_H is connected

- Iteratively construct a path from (H^0, L^0) to each (H^i, L^i)

Linearizability is a safety property

1. Linearizability is prefix-closed
   - Given linearization L of H, construct a linearization of the prefix of H by completing incomplete operations as in L
2. For *finite*, *deterministic* and *total* types, linearizability is a safety property

Concluding remarks

1. Liveness is defined on infinite histories, so must safety

Concluding remarks

1. Liveness is defined on infinite histories, so must safety
2. To prove that an implementation I satisfies a safety property P, sufficient to prove every finite history H exported by I is contained in P
    - To need to worry about the correctness of the infinite history

THANK YOU!