Initial Exploration

Yifei Sun

December 5, 2023

Objectives

1. model distributed semantics
2. verify a given semantic satisfies its specifications
3. check pair-wise compatibility
4. composition of two or more systems

Definitions: Operation

Operation are tuples: (proc, type, obj, ival, oval, stime, rtime) [1, pp.3].

```
class Operation(NamedTuple):
    proc: int # process id
    type: str # operation type
    obj: int # object id
    ival: Any # input value
    stime: int # start time
    rtime: int = None # return time
    oval: Any = None # output value
    symbol: str = None # readable representation
```

Definitions: History

A set of operations, contains all operations invoked in a given execution, describes the **observable outcomes of executions** [1, pp.4].

It's possible that a set of symbols and their relations:

- does not form a history (invalid)
- forms multiple histories (ambiguous)

Types:

- $H|_{\text{rd}} = \{\text{op} \in H : \text{op.type} = \text{rd}\}$
- $H|_{\text{wr}} = \{\text{op} \in H : \text{op.type} = \text{wr}\}$

Definitions: History

Relations:

- returns-before: $\mathrm{rb} \triangleq \{(a, b) : a, b \in H \land \mathrm{a.rtime} < \mathrm{b.stime}\}$
- same-session: $\mathrm{ss} \triangleq \{(a, b) : a, b \in H \land \mathrm{a.proc} = \mathrm{b.proc}\}$
- session-order: $\mathrm{so} \triangleq \mathrm{rb} \cap \mathrm{ss}$

```
class History:
    def __init__(self: Self,
      ops: set[Operation],
      **kwargs: set[Relation]
    ) -> None: ...
```

Definitions: Anstract Execution

An abstract execution is

- built on top of a history
- **captures the non-determinism, and constraints**
- an event graph $A = (H, \mathrm{vis}, \mathrm{ar}, \mathrm{hb})$ [2, pp.25-27,34-35]
- can be visualized as a directed graph

Definitions: Anstract Execution

Relations:

- vis (visibility): $a \xrightarrow{\text{vis}} b$
- ...

```
class AbstractExecution:
    def __init__(self: Self,
      hist: History,
      **kwargs: set[Relation]
    ) -> None: ...
```

Example: Monotonic Reads

```
1: ___ op_a: set("somekey", "someval")
         op_b: get("somekey") -> "someval"
2:       ___      ___
                  op_c: get("somekey") -> "someval"
```

- $a \xrightarrow{\text{vis}} b$
- $b \xrightarrow{\text{so}} c$
  - $b \xrightarrow{\text{rb}} c$
  - $b \xrightarrow{\text{ss}} c$

References

[1] P. Viotti and M. Vukolić, “Consistency in Non-Transactional Distributed Storage Systems”. 2016.

[2] S. Burckhardt, *Principles of Eventual Consistency*, Principles of Eventual Consistency., vol. 1. in Foundations and Trends® in Programming Languages, vol. 1. Now Publishers, 2014, pp. 1–150. [Online]. Available: https://www.microsoft.com/en-us/research/publication/principles-of-eventual-consistency/