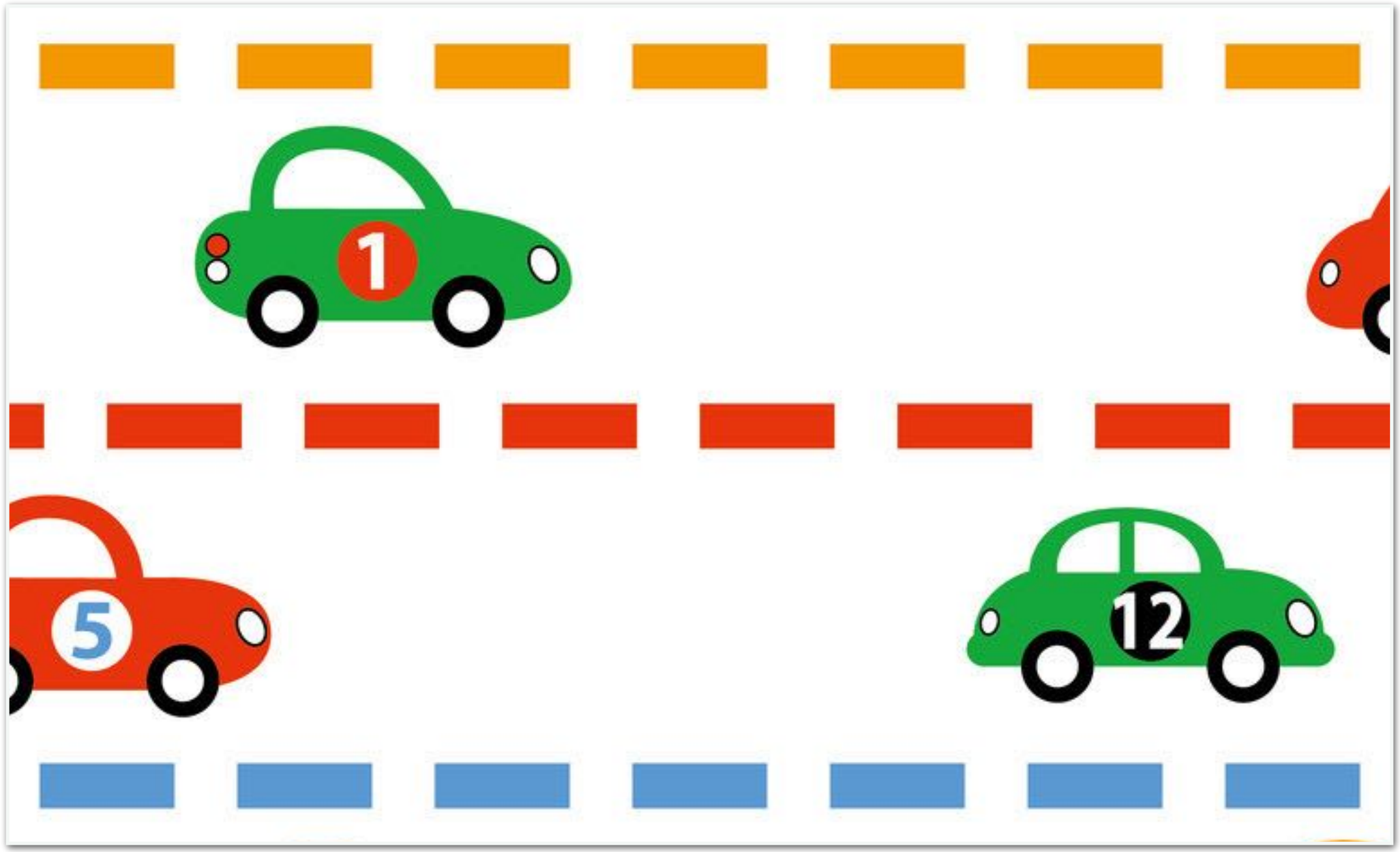
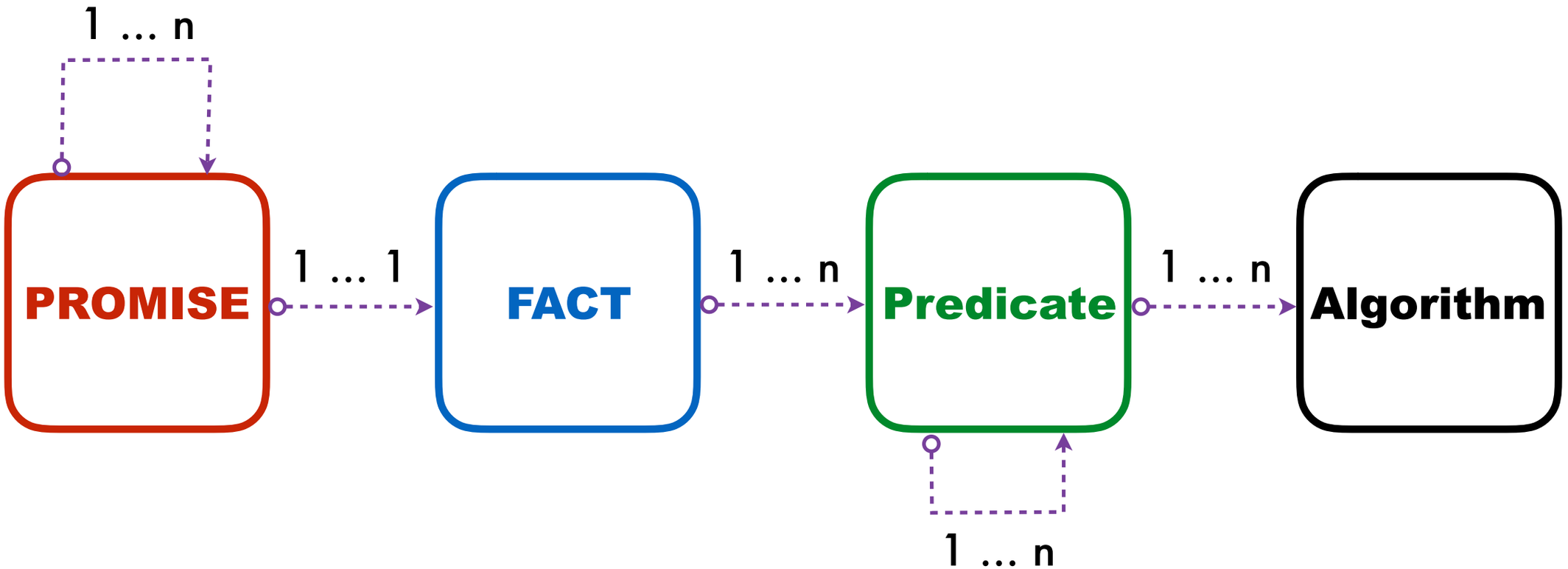
Thoughtworks



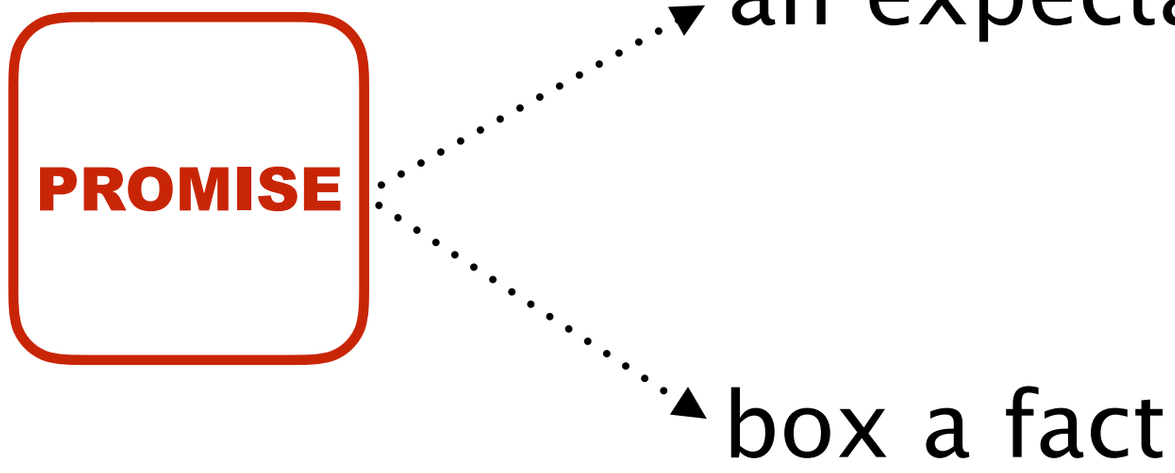

# Promise based assertion framework

王博

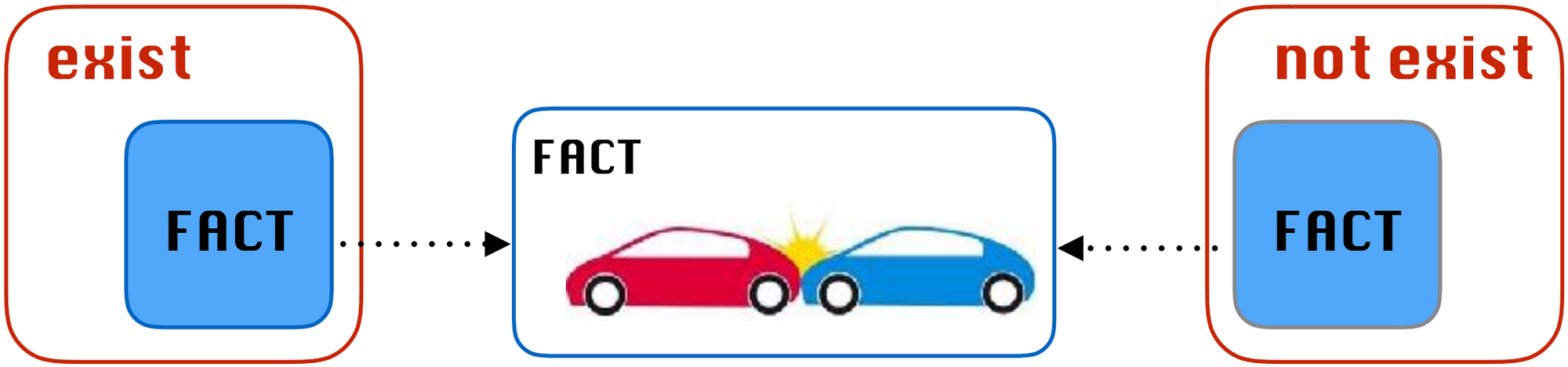
# Conception



# Promise



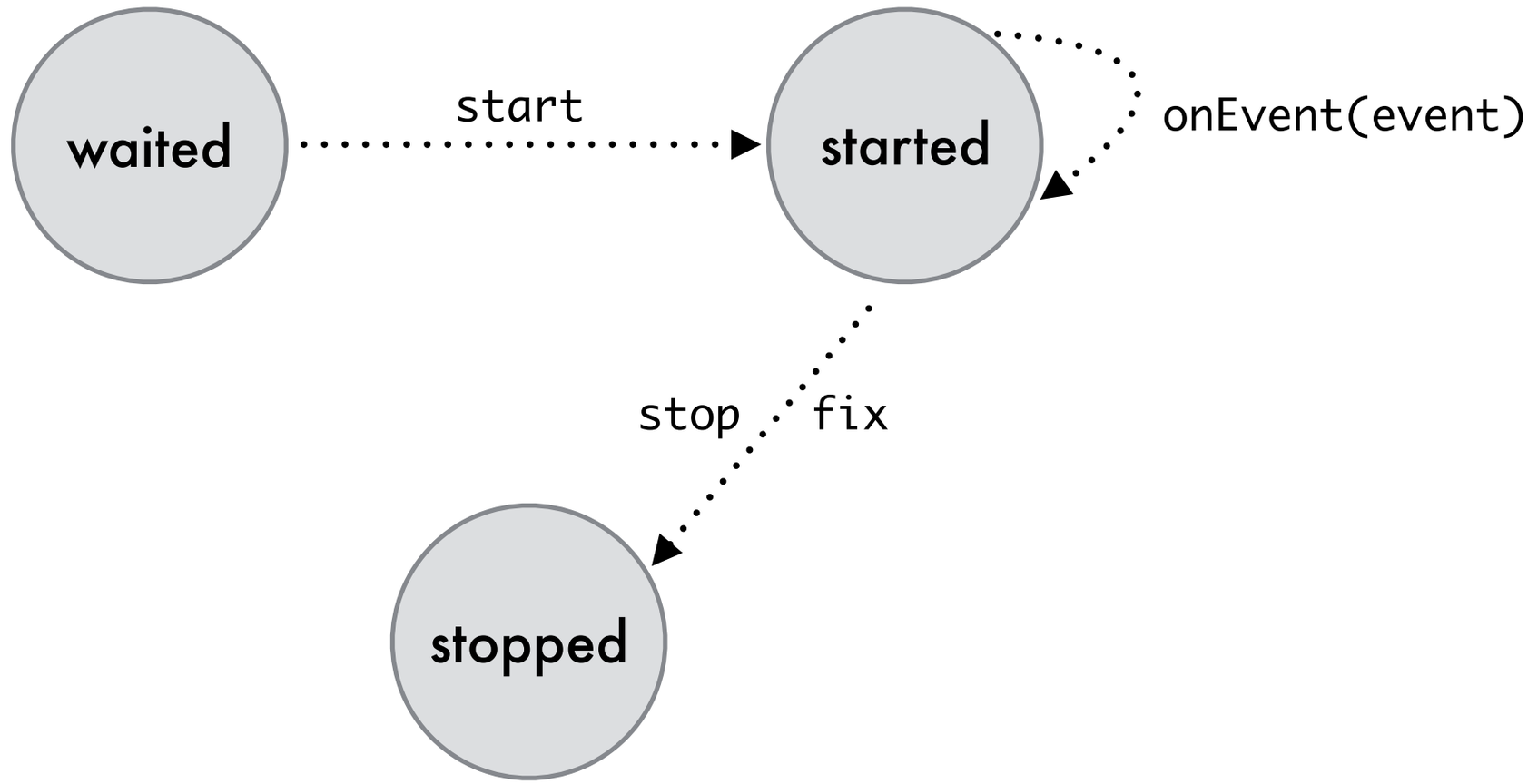
# Promise : two basic type



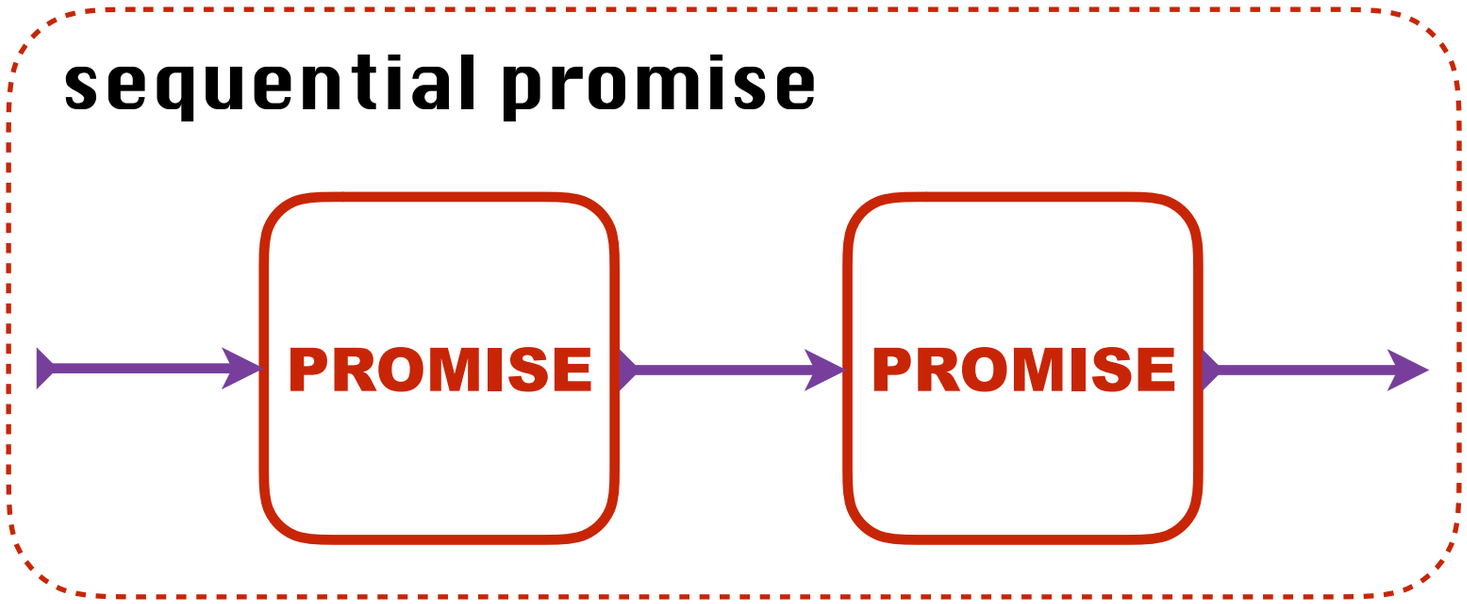

promise: exist collision

promise: not exist collision

# Basic Promise : state

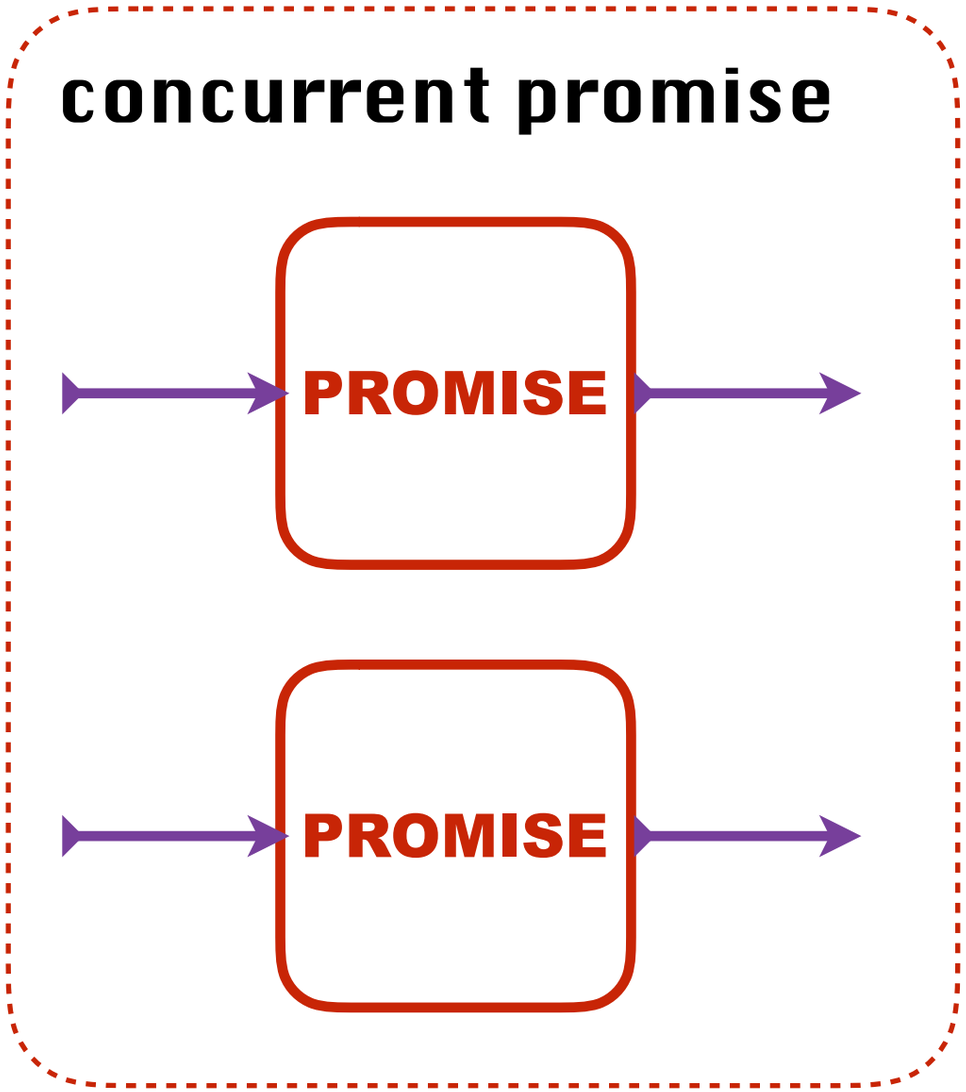


# Promise : relationship



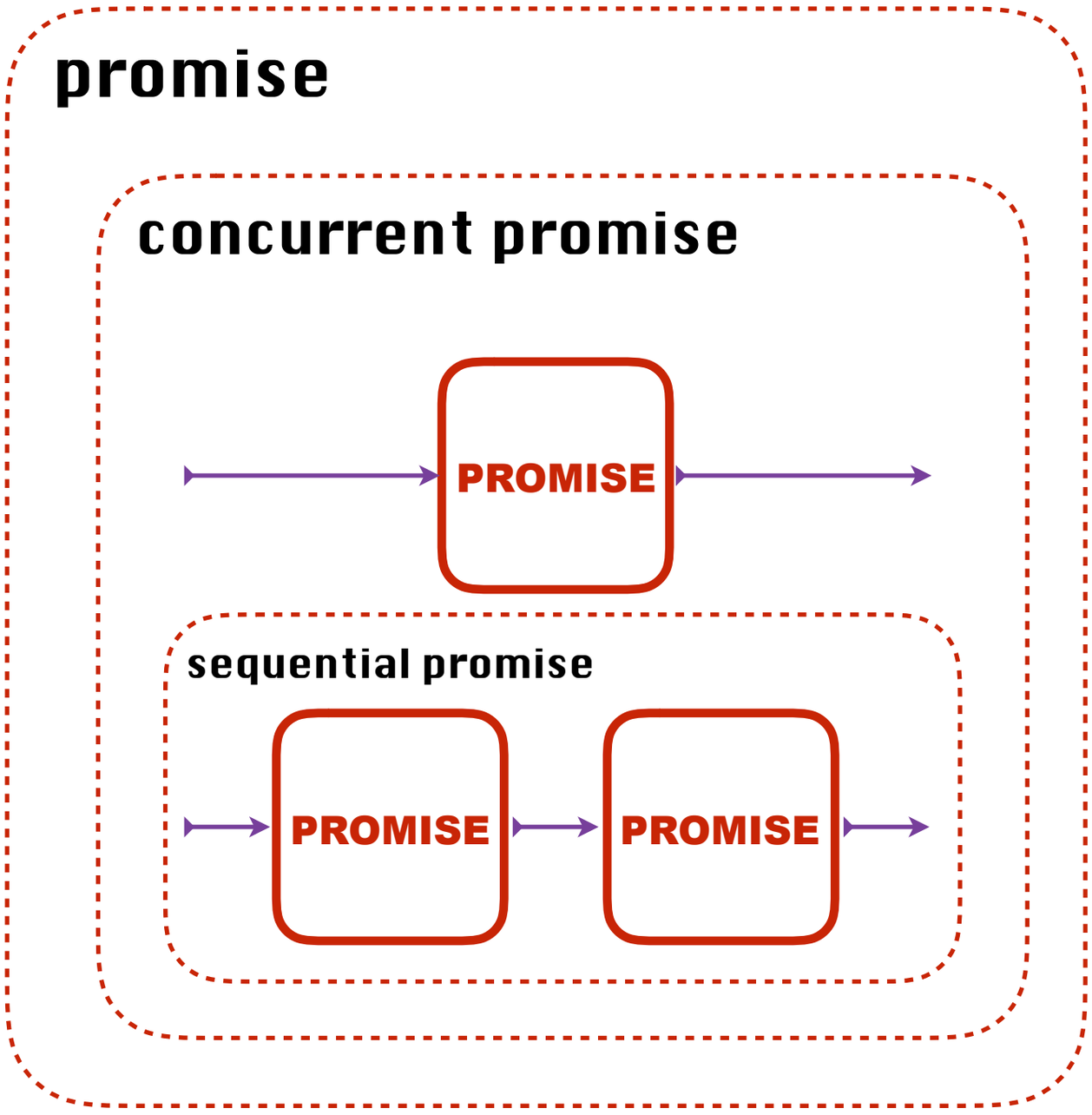

1. vehicle 0 is stop
2. distance of vehicle 0 and vehicle 1 is between 5m and 15m

# Promise : relationship



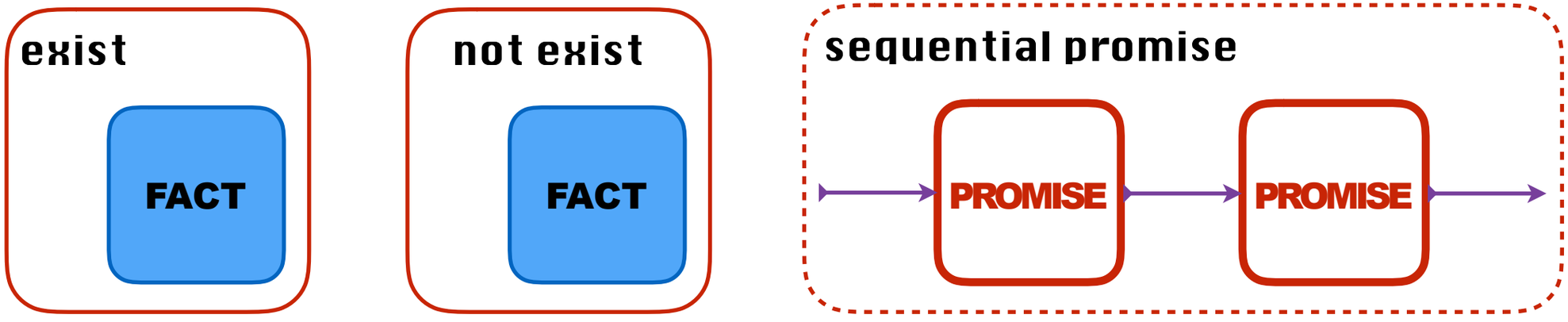

1. vehicle 0 is not collision
2. vehicle 0 is not stop

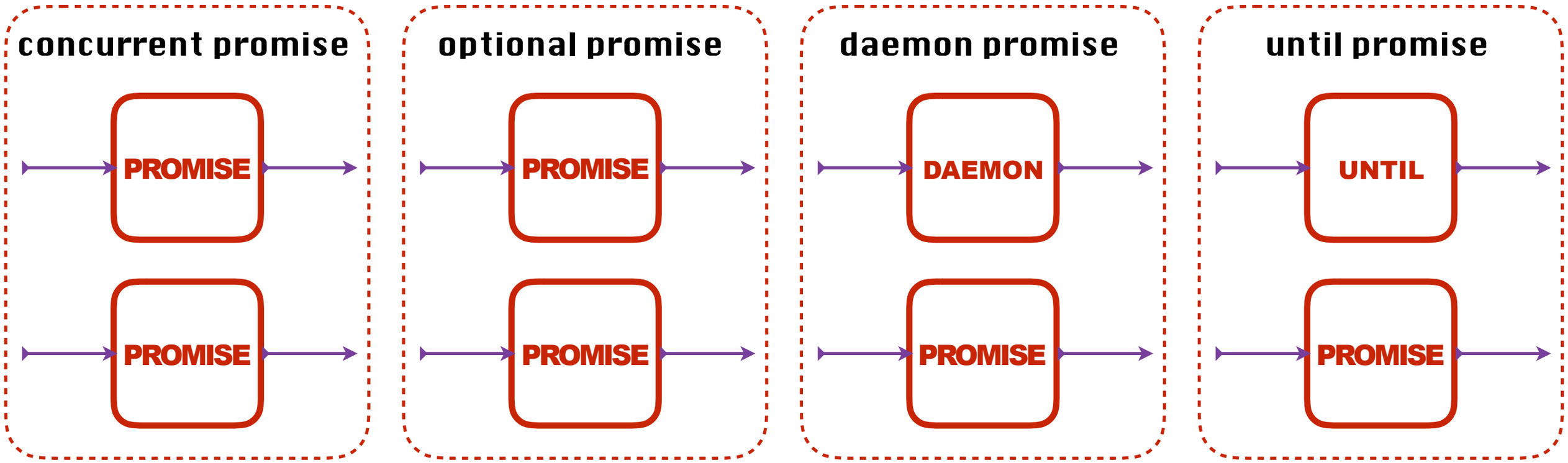
# Promise : composite




1. vehicle 0 is not collision
2. vehicle 0 is stop
3. distance of vehicle 0 and vehicle 1 ...

# Promise : all


exist
FACT
not exist
FACT
sequential promise
PROMISE
PROMISE


concurrent promise
PROMISE
PROMISE
optional promise
PROMISE
PROMISE
daemon promise
DAEMON
PROMISE
until promise
UNTIL
PROMISE

# Promise : result

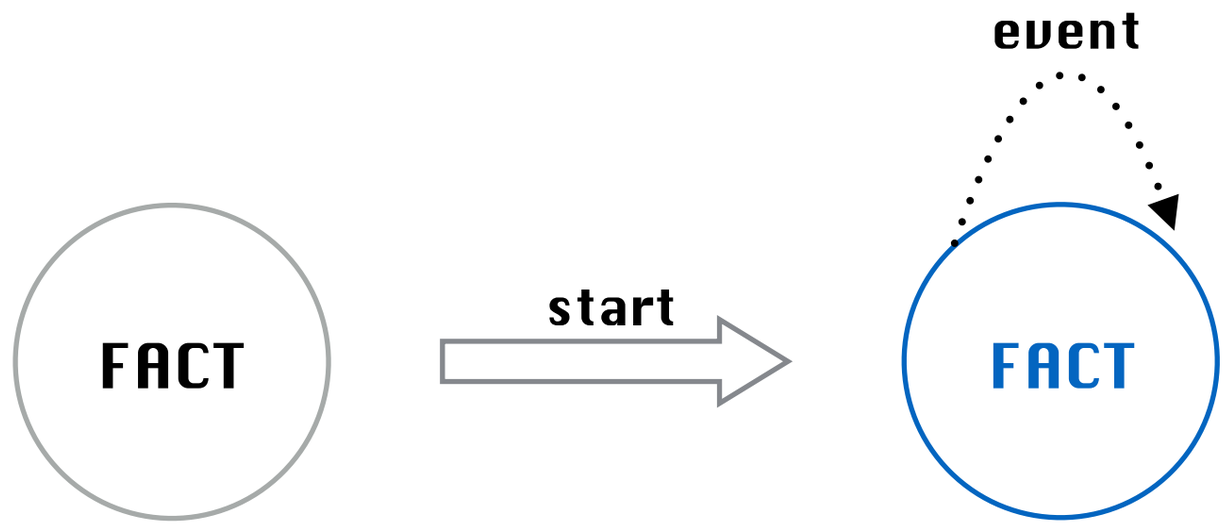
unknown

success

failed

fixed

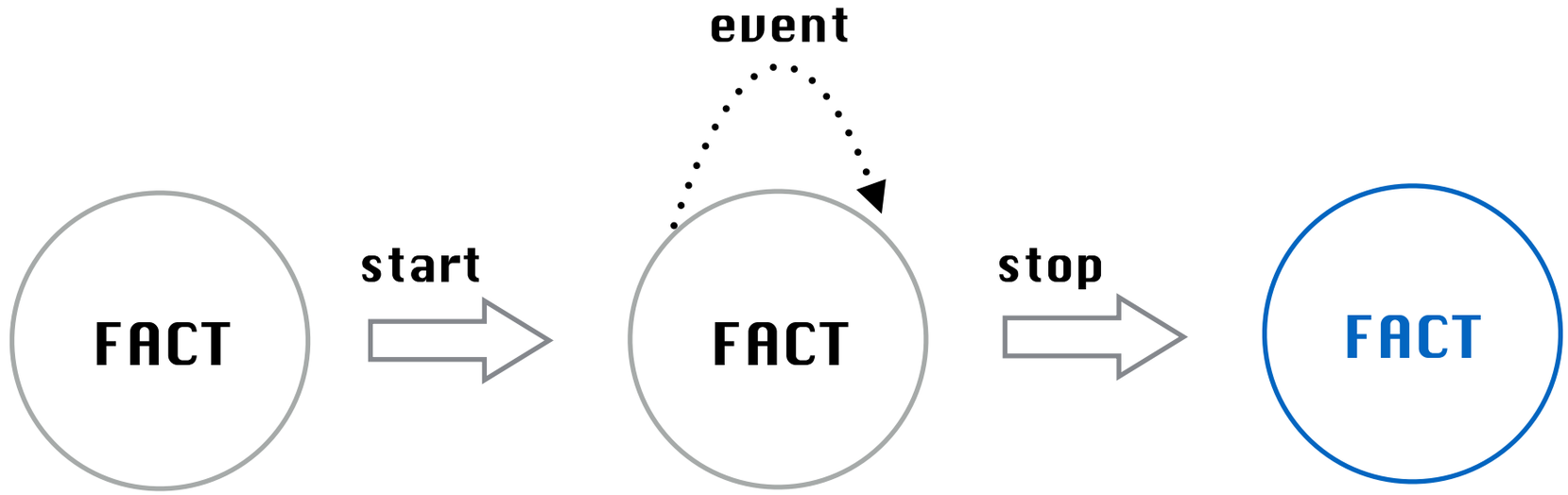
# Fact




not occurred

occurred ?

# Closure Fact



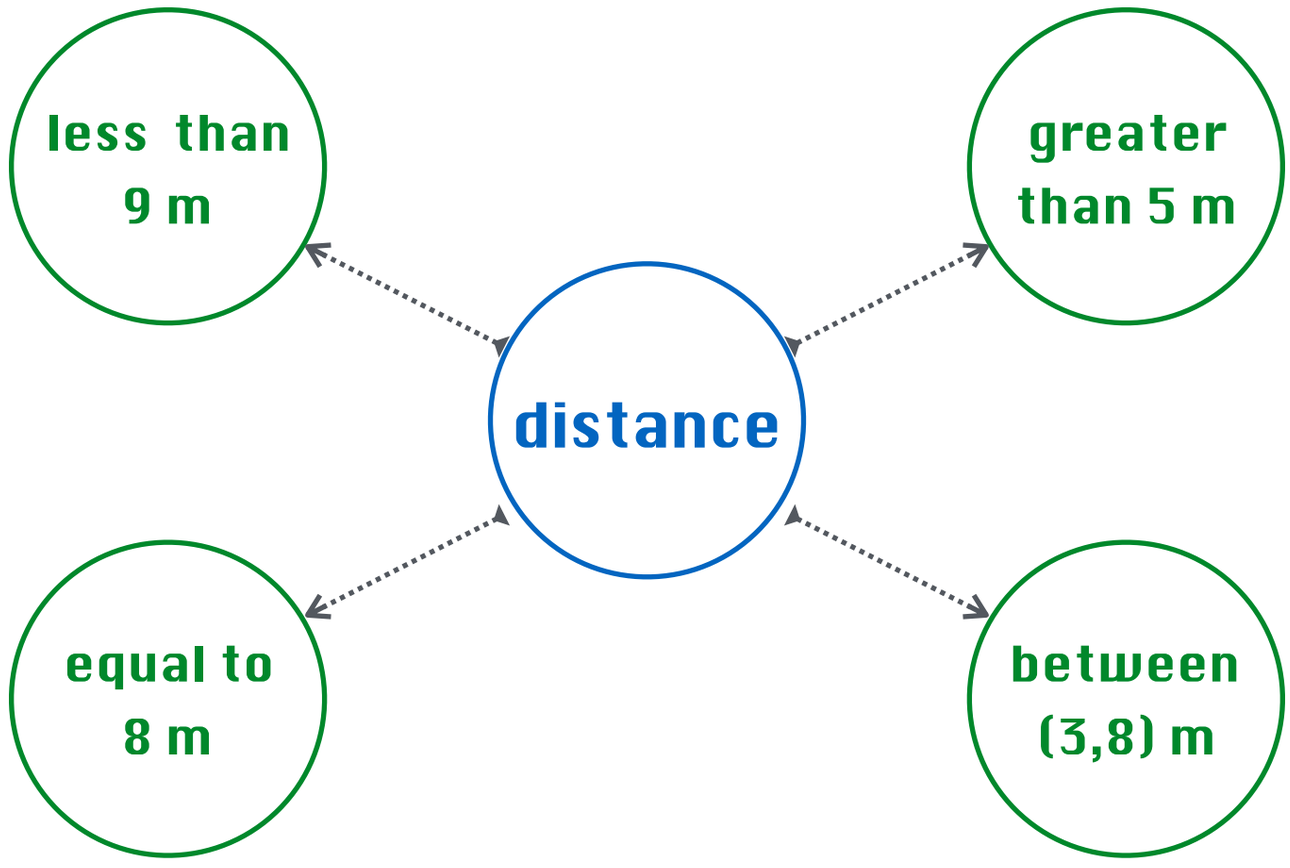
# Fact without predicate


They simply present a state.

# Fact with predicate


They often obtain value from event and environment.
After compose a predicate, then makes a fact.

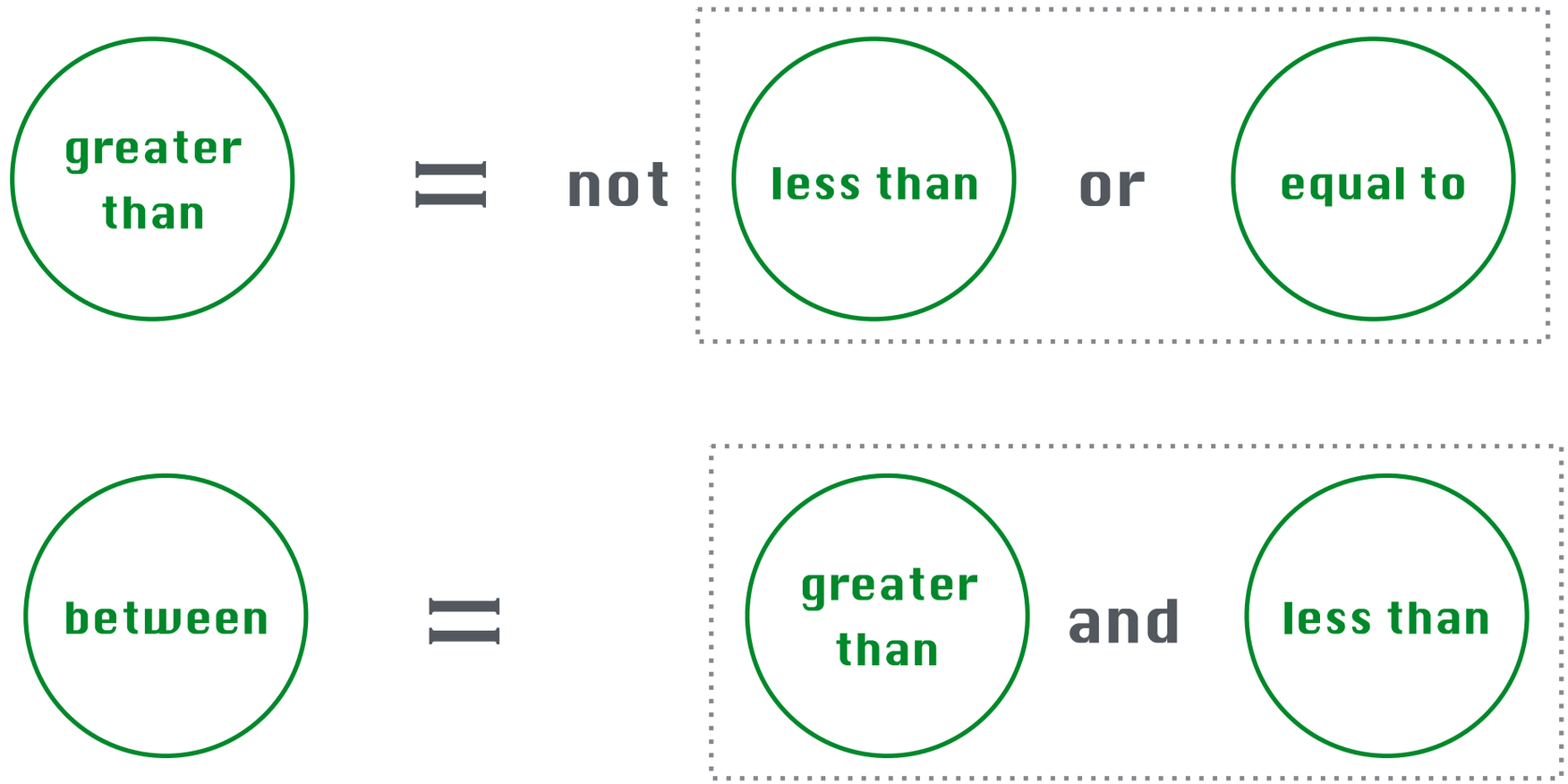
# Fact with predicate



# Predicate

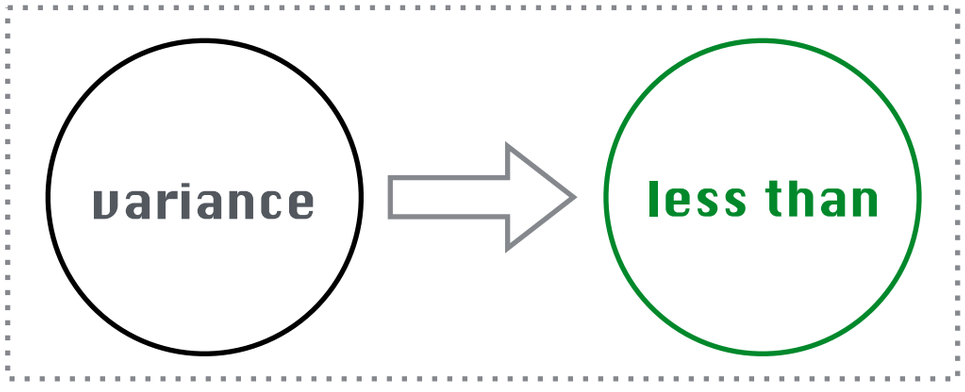
```
BOOL PRED(CONST T& VALUE);
```

# Predicate : conjunct



# Predicate : compose with algorithm

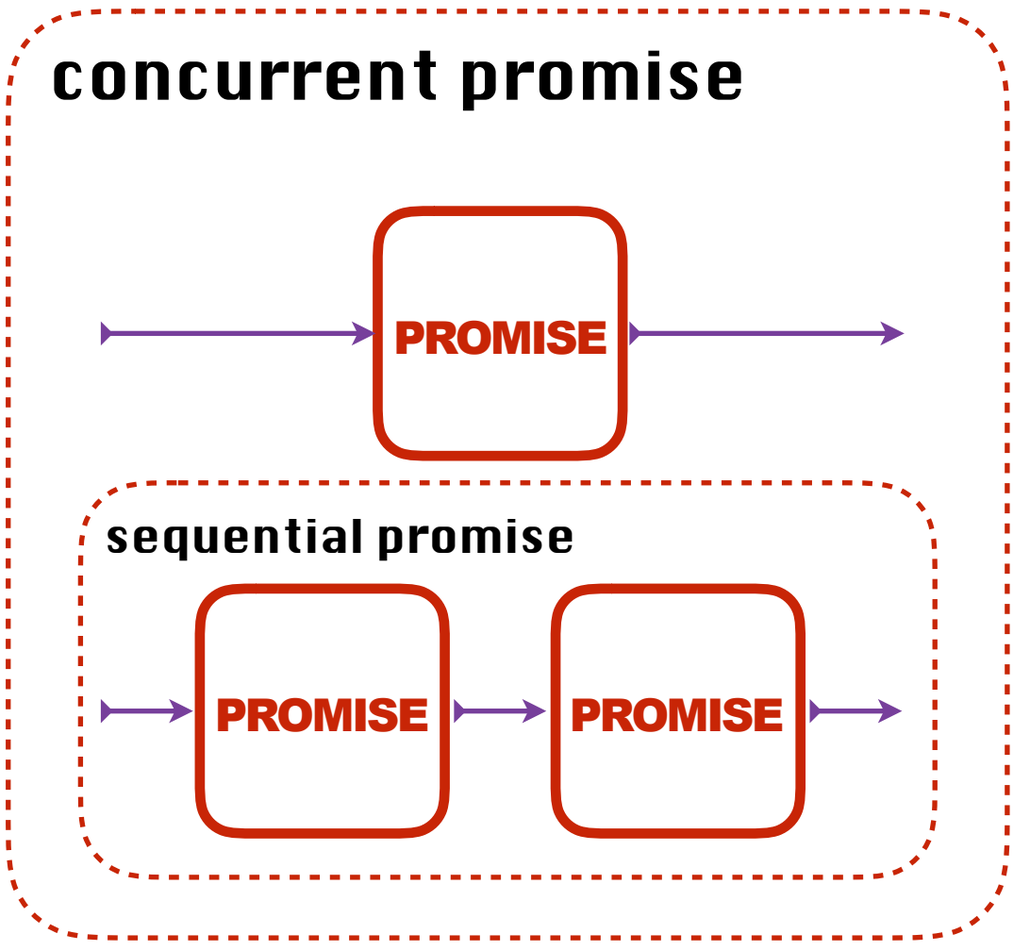


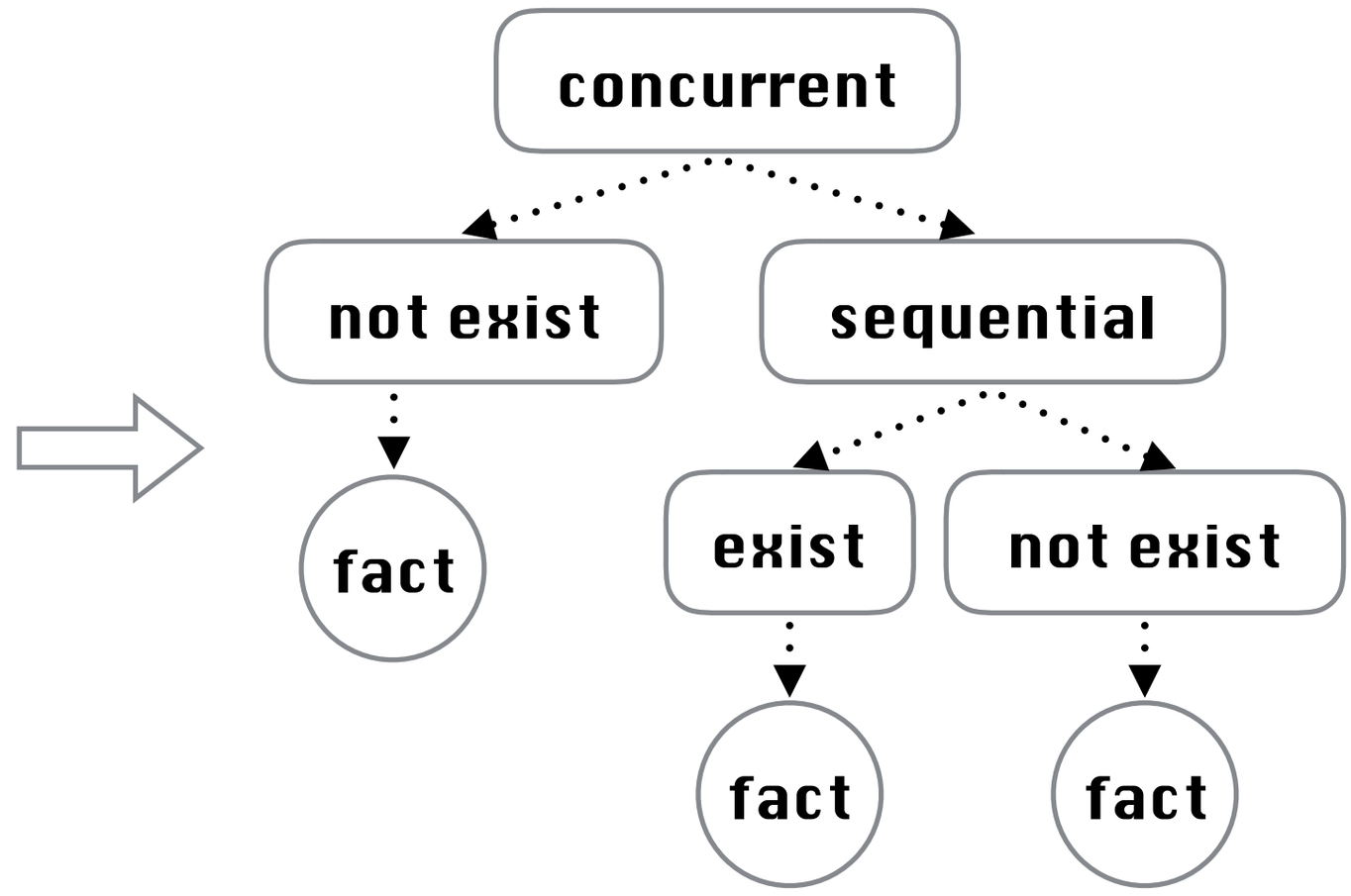



# Algorithm


Execute a special calculation, could be composed and reused.

# How to describe? DSL



# Inner DSL : C++

```
auto promise = __con( __not_exist(__fact(Collision, 0))
                    , __seq( __exist(__fact(LaneChange, 0).predOf(EqualTo(1))
                           , __not_exist(__fact(LaneGap, 0).predOf(GreaterThan(1)))
                           )
                    );

simulation.evaluate(promise);
```

# Outer DSL