Monitoring of Accumulator

- DSL4SC Example -

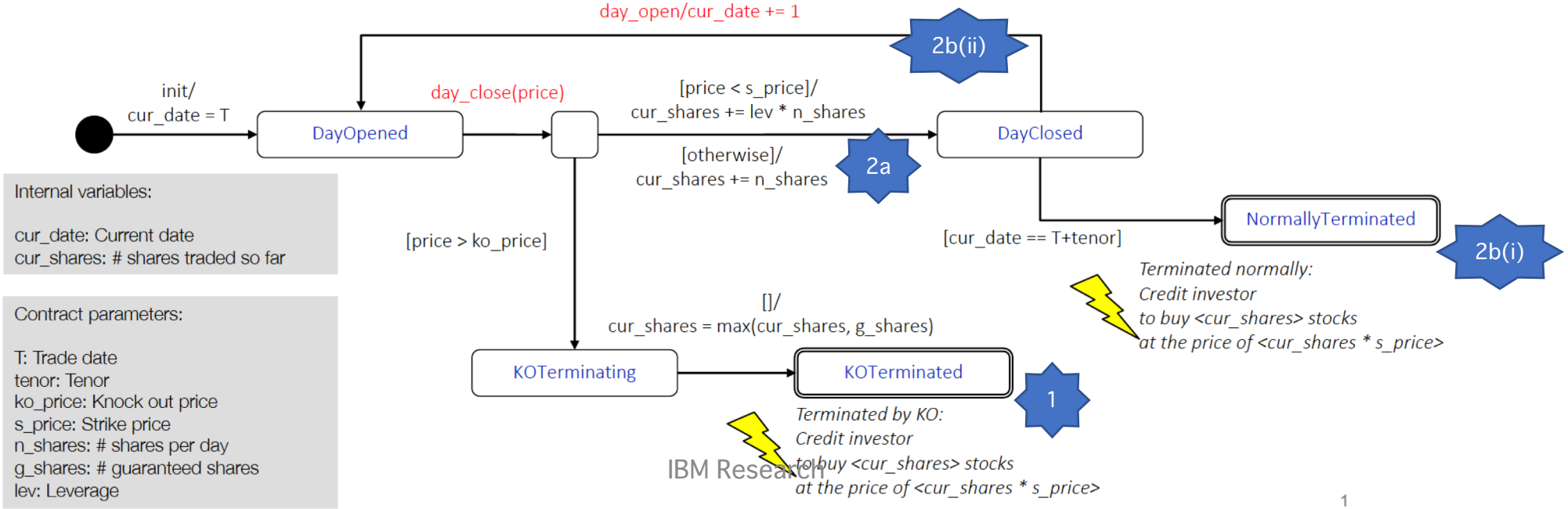
August 2018

Accumulator

Table of Contents

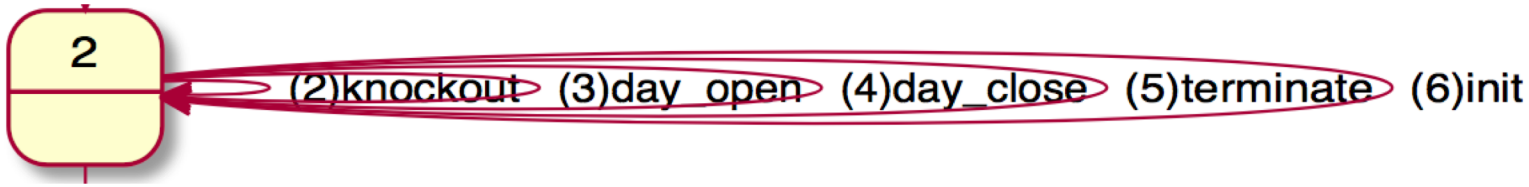
Accumulator Derivative

- Upon arrival of each "day_close" event,
  1. *if* the share price (carried by the event) is high enough,
     perform the knockout (sell-off) operation and terminate ("knockout")
  2. *otherwise*
     a. purchase a certain amount of shares (the amount varies depending on the price)
     b. (i) terminate normally if the tenor/expiration date is reached, or
        (ii) wait for a "day_open" event to repeat the entire process
- Reference: https://en.wikipedia.org/wiki/Accumulator_(structured_product)



Accumulator

- The contract has 5 different transaction events that are *observable*
  - **init, day_open, day_close, knockout, terminate**
  - we assume that all the other operations (like the "share purchase" operation) are *not* observable as transactions.
- For the Accumulator contract to monitor, we assume no other clue. It is as if the contract is defined as the following UML statechart with a single state

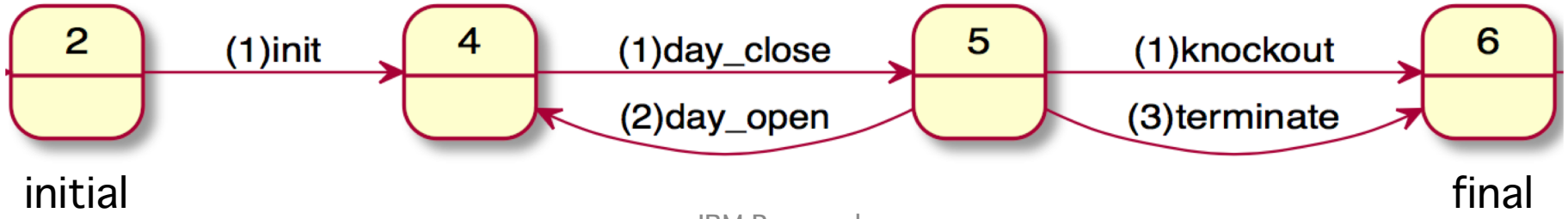


Accumulator Monitor1

- Monitor1 is defined to ensure that the transaction events are emitted in the expected order.
- Monitor1 in DSL4SC

```
protocol
  init;                              // 1st event
  day_close;                         // 2nd event
  (day_open; day_close)*;            // loop (0 or more times)
  (knockout + terminate)             // choice (one of the two events)
;;
```

- Monitor1 (statechart)

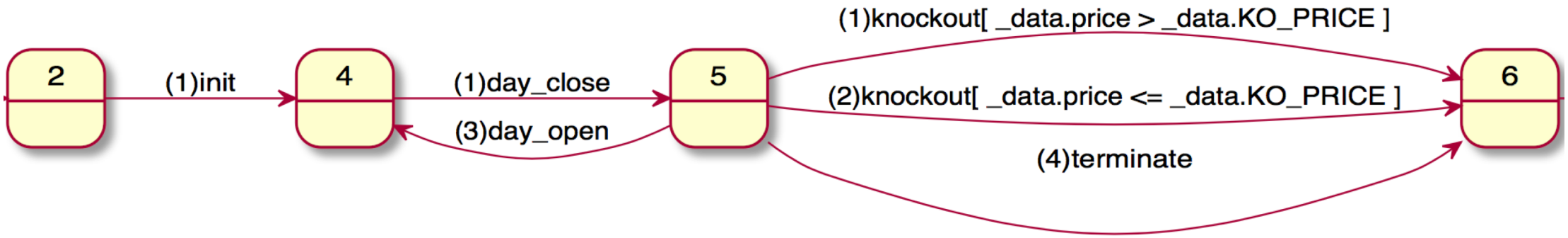


Accumulator Monitor2 (DSL4SC)

- Monitor2 ensures that, when the "knockout" event is observed, the knockout price is really reached.
  - Event parameters are accessible in JavaScript code ("{...}") parts

```
protocol
  init;
  day_close; (day_open; day_close)*;
  (knockout + terminate) ;;

rule
  on day_close do { _data.price = _event.data.price; };
            // memorize the price passed as a parameter of the event
  on knockout when { _data.price > _data.KO_PRICE } do { /*ok*/ };
  on knockout when { _data.price <= _data.KO_PRICE }
            do { raise new Error ("error on knockout"); }
            // ensure that price is greater than KO_PRICE
```

Accumulator Monitor2 (statechart)



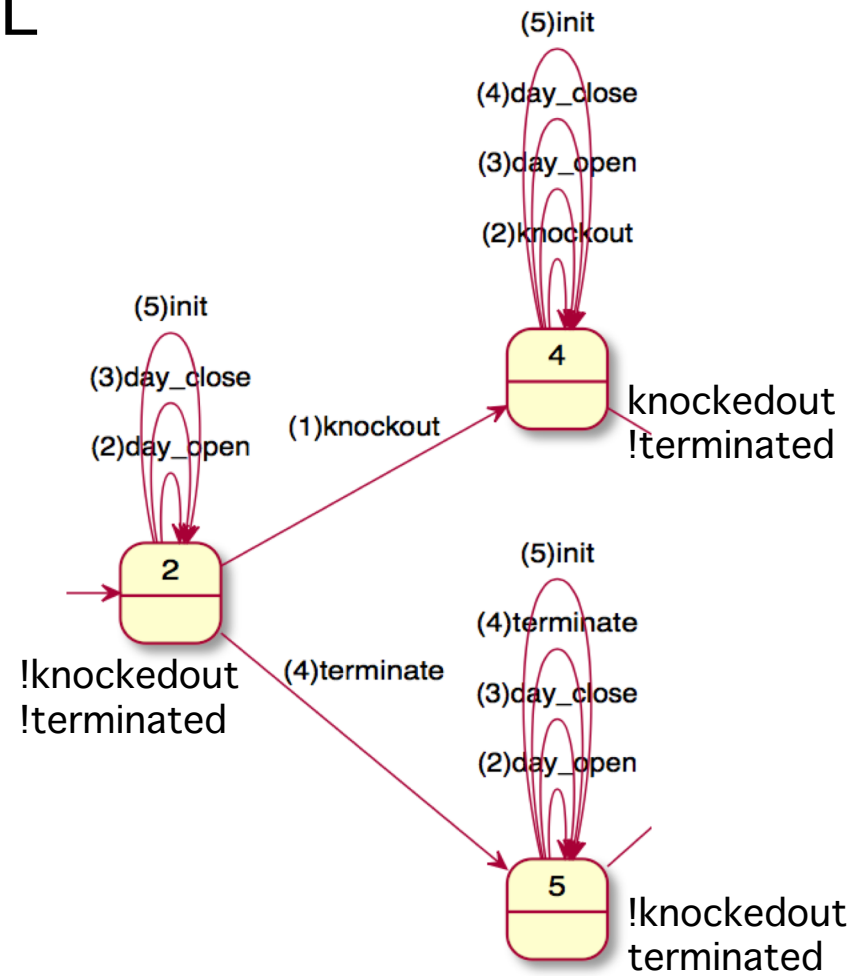
Accumulator Monitor3 (DSL4SC)

- Monitor3 ensures the following safety/liveness properties to hold
  - Safety: it **never** happens that the accumulator becomes both terminated *and* knocked out simultaneously.
  - Liveness: the accumulator will **always** become terminated *or* knocked out **eventually**

```
property
  !knockedout & !terminated;          // initial condition
  [] !(knockedout & terminated);      // safety
  []<> (knockedout | terminated);     // liveness

rule
  on knockout ensure knockedout;      // knockout turns on knockedout
  except on knockout preserve knockedout;

  on terminate ensure terminated;     // terminate turns on terminated
  except on terminate preserve terminated;
```

Accumulator Monitor3 (statechart)

- Monitor3 in SCXML



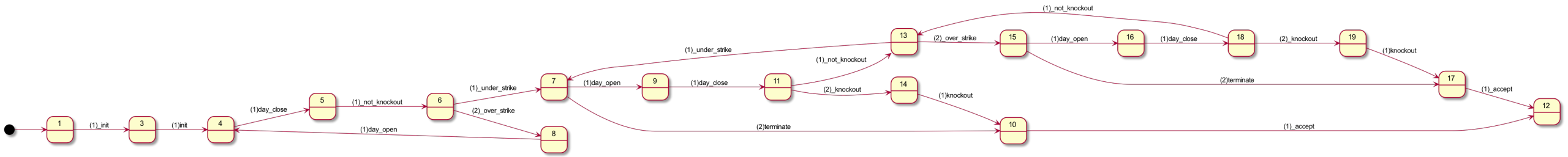
Accumulator Monitor4 (DSL4SC)

- Monitor4 ensures the following "anomaly" cases to be excluded:
  - price never becomes under STRIKE_PRICE
  - price never becomes over STRIKE_PRICE

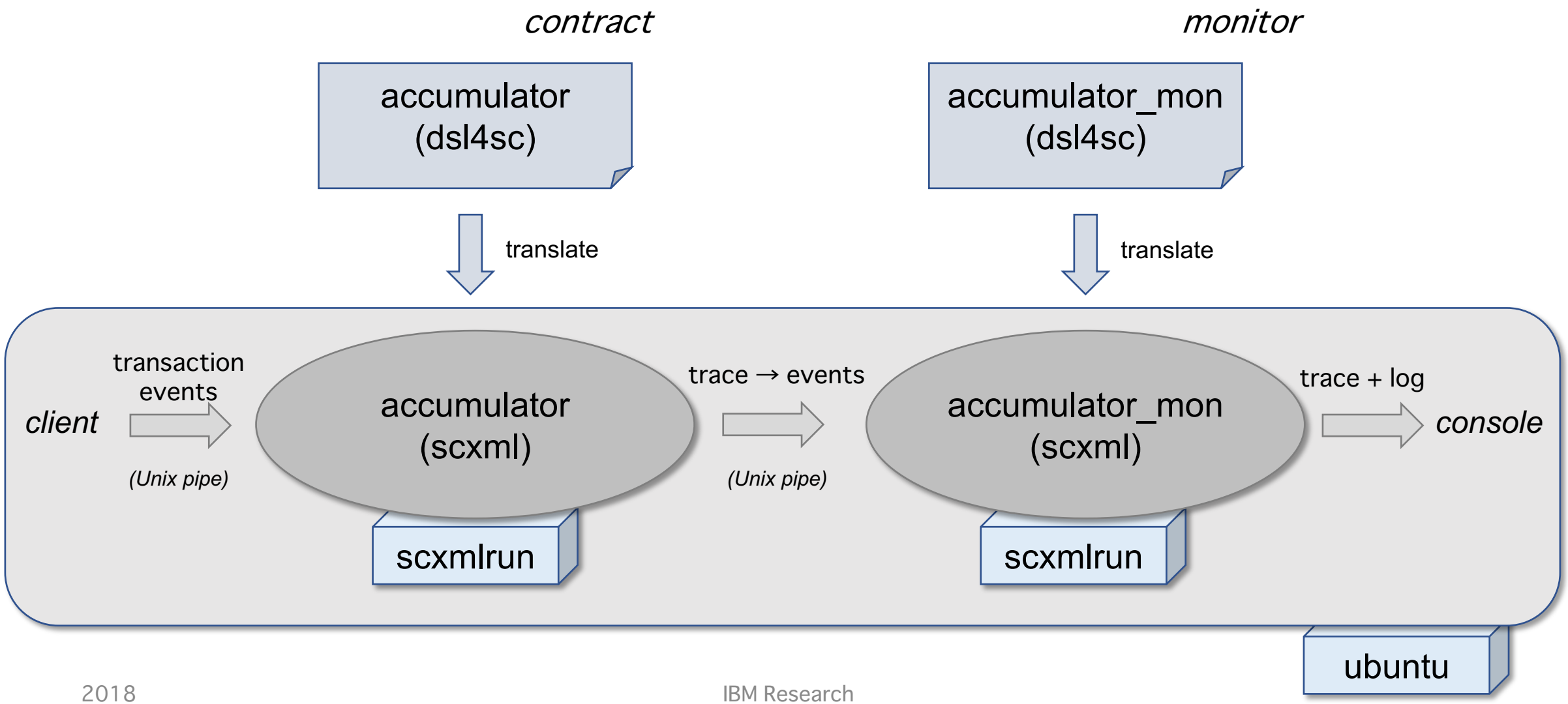
```
protocol
  init;
  day_close;
  (_not_knockout; (_under_strike + _over_strike);
     day_open; day_close)*;
  (_knockout; knockout
     + _not_knockout; (_under_strike + _over_strike); terminate)
;;

property
  is_over_strike;        // initial condition
  !(<{is_over_strike}*>(last & is_over_strike));
  !(<{!is_over_strike}*>(last & !is_over_strike));

rule
  on _over_strike ensure is_over_strike;
  on _under_strike ensure !is_over_strike;
  except on _over_strike, _under_strike preserve is_over_strike;
```

```
rule
  on day_close raise _knockout + _not_knockout
  // raise either "_knockout" or "_not_knockout"
  {
    _data.price = _event.data.price;
    if (_data.price > _data.KO_PRICE)
       __raiseEvent ("_knockout");
    else
       __raiseEvent ("_not_knockout");
  };

  on _not_knockout raise _under_strike + _over_strike
  {
    if (_data.price < _data.STRIKE_PRICE)
       __raiseEvent ("_under_strike");
    else
       __raiseEvent ("_over_strike");
  };
```

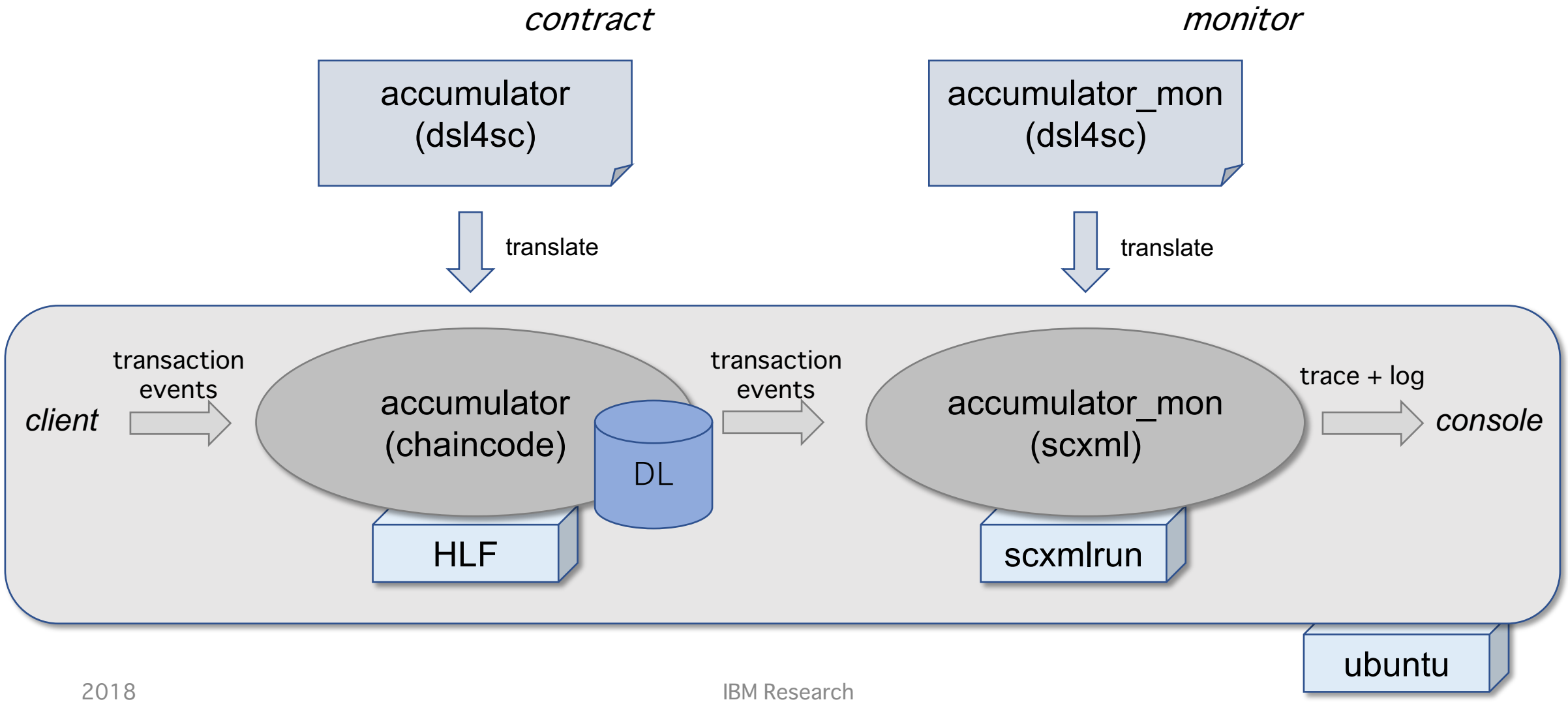
Accumulator Monitor4 (statechart)



Running with scxmlrun, a SCXML interpreter



Running on Hyperledger



Test Cases

Normal cases

- scenario1
  init (KO_PRICE=110, STRIKE_PRICE=90);
  day_close (price=101); day_open;
  day_close (price=89); day_open;
  day_close (price=100);
  terminate

- scenario2
  init (KO_PRICE=110 , STRIKE_PRICE=90);
  day_close (price=111);
  knockout

Error cases

- invalid1
  init (KO_PRICE=110, STRIKE_PRICE=90);
  day_close (price=100);
  day_close (price=101); // missing "day_open"
  terminate

- invalid2
  init (KO_PRICE=110, STRIKE_PRICE=90);
  day_close (price=89);
  knockout   // price <= KO_PRICE

- invalid3
  init (KO_PRICE=110, STRIKE_PRICE=90);
  day_close (price=101); day_open;
  day_close (price=111);
  knockout; terminate   // knockout & terminate

- invalid4
  init (KO_PRICE=110, STRIKE_PRICE=90);
  day_close (price=101); day_open;
  day_close (price=110); day_open;
  day_close (price=111);
  knockout   // anomaly (only for monitor4)

Results

	Monitor1	Monitor2	Monitor3	Monitor4
scenario1	-	-	-	-
scenario2	-	-	-	error
invalid1	error	error	-	error
invalid2	-	error	-	error
invalid3	error	error	error	error
invalid4	-	-	-	error

- "-" / "error" indicate no error / error is detected.
- green indicates correct result.

end of slides