RTML formulas. Talk outline

- VOS (*Virtual Online Shop*) example
- Atoms (Language of events)
- Instance monitor formulas
- Class monitor formulas
- Translation to Java code
VOS Example: VOS Abstract

1. **RECEIVE** itemRequest (qty, item_id)
2. **SWITCH** Available?
   - **NO**
     - **INVOKE** notAvail
   - **YES**
     - **INVOKE** offer (cost)
     - **ON MESSAGE** offerNack
     - **PICK** offerChange
3. **ON MESSAGE** getOrdererData (ucc_data)
4. **INVOKE** getOrdererDataNack
5. **ON MESSAGE** getOrdererDataAck (transfer_id)
6. **SWITCH** Payment
   - **NO**
     - **FAIL**
   - **YES**
     - **SUCC**
VOS Example: Bank

- [RECEIVE] startPayment (cost, scc_data)
  - [SWITCH] SCC correct?
    - [INVOKE] startPaymentAck
    - [INVOKE] startPaymentNack
  - NO
    - [INVOKE] getOrdererData (ucc_data)
      - [SWITCH] UCC correct?
        - YES
          - [INVOKE] getOrdererDataAck (transfer_id)
            - YES
              - [ON MESSAGE] confirmPayment
            - NO
              - [INVOKE] getOrdererDataNack
        - NO
          - [INVOKE] getOrdererDataNack
  - [PICK] FAIL

- [ON MESSAGE] cancelPayment
  - FAIL

- [INVOKE] confirmPaymentNack
  - NO
    - [INVOKE] confirmPaymentAck
      - YES
        - [SWITCH] confirmPayment
      - NO
        - [INVOKE] confirmPaymentNack
  - FAIL

- [RECEIVE] startPayment
  - FAIL

- SUCC

Fabio Barbon
Monitoring Instances and Classes of Web Services
Virtual Online Shop example

1. **RetriesOnSuccCount**: count the number of items offered to the User before the User accepts to buy.

2. **PaymentTime**: compute the time requested to finalize the payment with the bank from the payment request.

3. **GlobalStoreCcNotRefuse**: the credentials of the Store have never been refused by the Bank in any execution of the VOS.

4. **CountStoreCCRefused**: count the total number of times the Bank has refused the credentials of the Store on all the executions of the VOS.

5. **AverageUserRetriesCount**: average number of times the user gets and refuses an offer from the VOS.

6. **AveragePaymentTime**: average duration of the interactions with the bank for the payment procedure.
Language atoms (events)

\[ e ::= \text{link.start} \mid \text{link.end} \mid \text{msg(link.input/output = msg[opt-constraints]])} \mid \text{cause(link.var = val}) \mid \text{cause(link.state = label}) \]

Examples:
- \[\text{msg(Bank.input } = \text{ startPayment)}\]
- \[\text{msg(Shop.input } = \text{ request [item = 'Book']})\]
- \[\text{cause(Bank.state } = \text{ FAIL})\]
- \[\text{cause(Bank.startPayment\_cost = e100})\]

Fabio Barbon

Monitoring Instances and Classes of Web Services
Instance monitor formulas

\[ b ::= e \mid Y b \mid O b \mid H b \mid b \land b \mid
\]

\[ n = n \mid n > n \mid \neg b \mid b \land b \mid true \]

\[ n ::= \text{count} (b) \mid \text{time} (b) \mid b?n:n \mid
\]

\[ n + n \mid n - n \mid n \ast n \mid n/n \mid 0 \mid 1 \mid ...
\]

- \( Y b \) means “\( b \) was true in the previous step”;
- \( O b \) means “\( b \) was true (at least) once in the past”;
- \( H b \) means “\( b \) was true always in the past”;
- \( b_1 S b_2 \) means “\( b_1 \) has been true since \( b_2 \)”.

![Instance Monitor Diagram]
- **RetriesOnSuccCount:**
  \[ O(\text{cause}(\text{VOS.state} = \text{SUCC}))? \]
  \[ \text{count}(\text{msg}(\text{VOS.output} = \text{offer}))) : 0 \]

- **PaymentTime:**
  \[ \text{time}((\neg(\text{cause}(\text{Bank.state} = \text{SUCC}, \text{FAIL}))\text{S}) \]
  \[ \text{msg}(\text{Bank.input} = \text{startPayment}))) \]
Class monitor formulas

\[ B ::= \text{And} (b) \mid Y \ B \mid O \ B \mid H \ B \mid B \ S \ B \mid \]
\[ N = N \mid N > N \mid \neg B \mid B \land B \mid \text{true} \]
\[ N ::= \text{Count} (b) \mid \text{Sum} (n) \mid \]
\[ N + N \mid N - N \mid N \times N \mid N / N \mid 0 \mid 1 \mid \ldots \]

(“Average (n)” abbreviates “Sum (n)/Count (O start)”)
GlobalStoreCcNotRefuse:
And (¬O msg(Store.input = startPaymentNack))

CountStoreCCRefused:
Count (O msg(Store.input = startPaymentNack))

AverageUserRetriesCount:
Average (count (msg(VOS.output = offer)))

AveragePaymentTime:
Average (time ((¬(cause(Bank.state = SUCC, FAIL)) S msg(Bank.input = startPayment))))
Keep in a map $Val$ the (numerical or boolean) value for each sub-formula. On events, update $Val$ bottom-up on the structure of the formula.

- $Val(e) := \text{“if event } e \text{ is occurring”}$
- $Val(b_1 \land b_2) := (Val(b_1) \land Val(b_2))$
- $Val(s_1 = s_2) := (Val(s_1) = Val(s_2))$
- $Val(Y b) := Val_{old}(b)$
- $Val(O b) := Val_{old}(O b) \lor Val(b)$
- $Val(count(b)) := \text{if } Val(b) \text{ then } (Val_{old}(count(b)) + 1) \text{ else } Val_{old}(count(b))$
- $Val(time(b)) := \text{if } Val(b) \land Val_{old}(b) \text{ then } (Val_{old}(time(b)) + \text{elapsed}) \text{ else } Val_{old}(time(b))$
- $...$
An instance monitor for every sub-formula “under” And, Count and Sum

An aggregating monitor built with same idea of map Val for instance monitor. For terminal sub-formula:

Val(And(b)) := \( \bigwedge_{i \in I(b)} i.\text{Val}(b) \)

Val(Count(b)) := \( \sum_{i \in I(b)} (\text{if } i.\text{Val}(b) \text{ then } 1 \text{ else } 0) \)

Val(Sum(n)) := \( \sum_{i \in I(n)} i.\text{Val}(n) \)
package monitor.instance;
import org.astroproject.monitor.core.*;
import org.astroproject.monitor.utility.Utility;

public class Class_VOS_GlobStoreRefuseCc implements IProcessClassMonitor, IStaticPropertyChangedMonitor {
    // Formula: "COUNT 0 KNOW Store.state = pc234"
    public int status() { return IMonitor.STATUS_RUNNING; }
    public void init() { }
    public void update() {
        state[0]=Utility.computeCount("monitor.instance.VOS__GlobStoreRefuseCc_0");
    }
    float state[] = new float[1];
    public float value() { return state[0]; }
    public String getErrorNode() { return "no error"; }
    public String getProcessName() { return "VOS"; }
    public String getProperty() { return "GlobStoreRefuseCc"; }
    public String getDescription() { return "Store credentials are never refused by bank"; }
}
package monitor.instance;
import org.astroproject.monitor.core.*;
import java.util.ArrayList;
import java.lang.Integer;
public class VOS__GlobStoreRefuseCc_0 implements IProcessInstanceMonitor, IServantMonitor, IBooleanPropertyMonitor {
    private float state[] = new float[3];
    private boolean is_valid = true;
    public void evolve(BpelMsg msg) {
        if (msg.getSenderType().indexOf("Store")==0 ||
            msg.getReceiverType().indexOf("Store")==0) {
            float next[] = new float[3];
            next[1]=(msg.getOperation.equals("startPaymentNack"))?1:0;
            is_valid=(next[2]==1);
            state=next;
        }
        return;
    }
    void init() {
        state[1]=(Store.p1())?1:0;
        state[2]=state[1];
        is_valid=(state[2]==1);
        return;
    }
    public boolean isValid() { return is_valid; }
}