

Data-flow based Model Analysis

Christian Saad and Bernhard Bauer

(Meta) Modeling allows to formally capture the syntactical structure of application domains

but

Validation of (static) semantics is still a challenge

(Meta) Modeling allows to formally capture the syntactical structure of application domains

but

Validation of (static) semantics is still a challenge

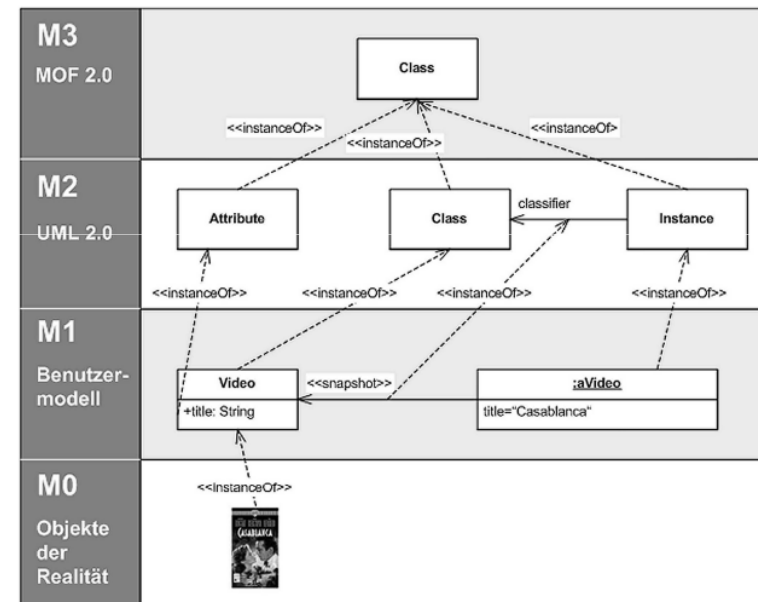
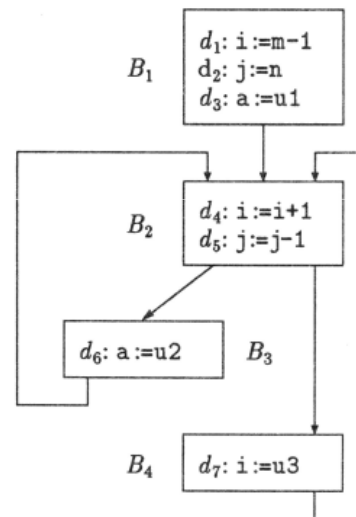
⇒ Use Data-flow Analysis to enable context-sensitive model analysis

■ Data-flow Analysis

- Used in Compiler Construction
- Calculates information flow in directed graphs
- Approximate min/max flow for cyclic paths

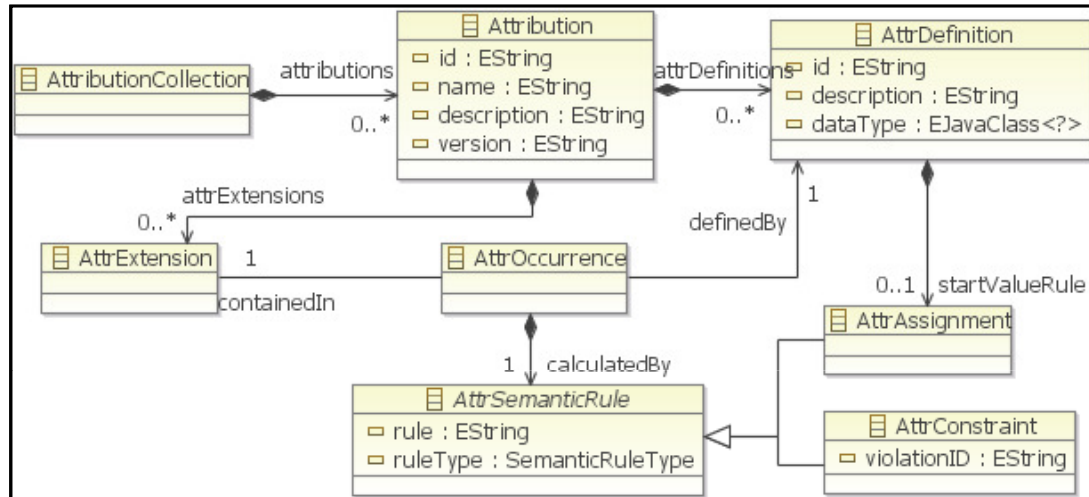
```

/* d1 */ i := m-1;
/* d2 */ j := n;
/* d3 */ a := u1;
do
  /* d4 */ i := i+1;
  /* d5 */ j := j-1;
  if e1 then
    /* d6 */ a := u2
  else begin
    /* d7 */ i := u3;
  end
while e2
    
```



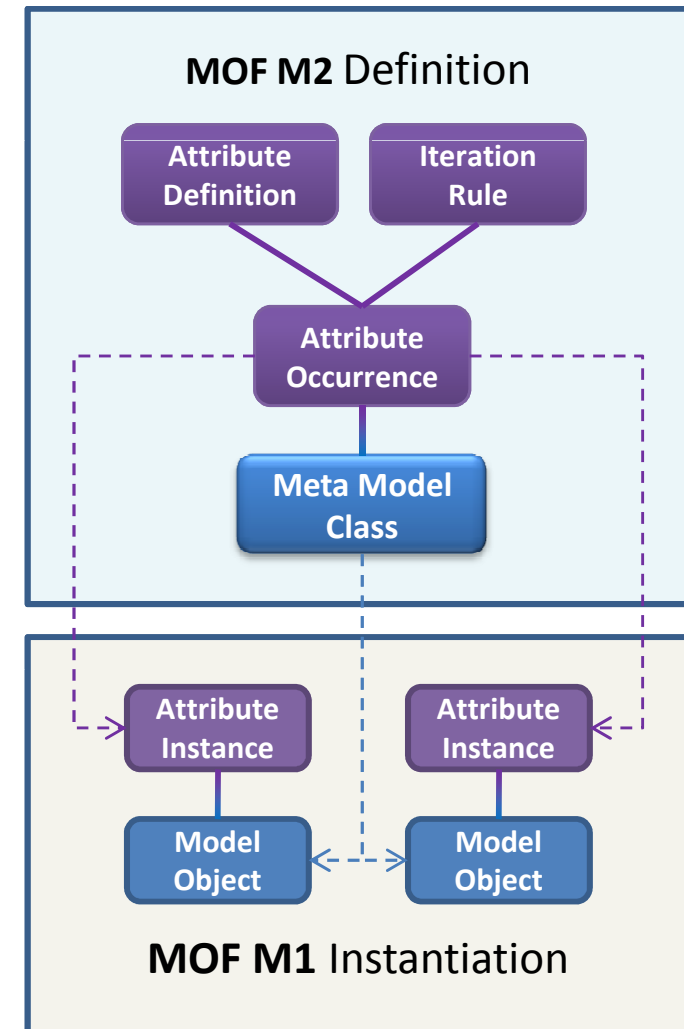
Stay in the **Modeling Domain!**

- › DFA definition **based on Meta Model**
- › Respect concepts like **Generalization** etc.
- › Inspired by **Attribute Grammars**

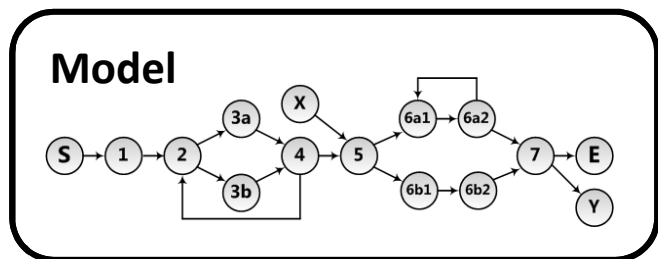
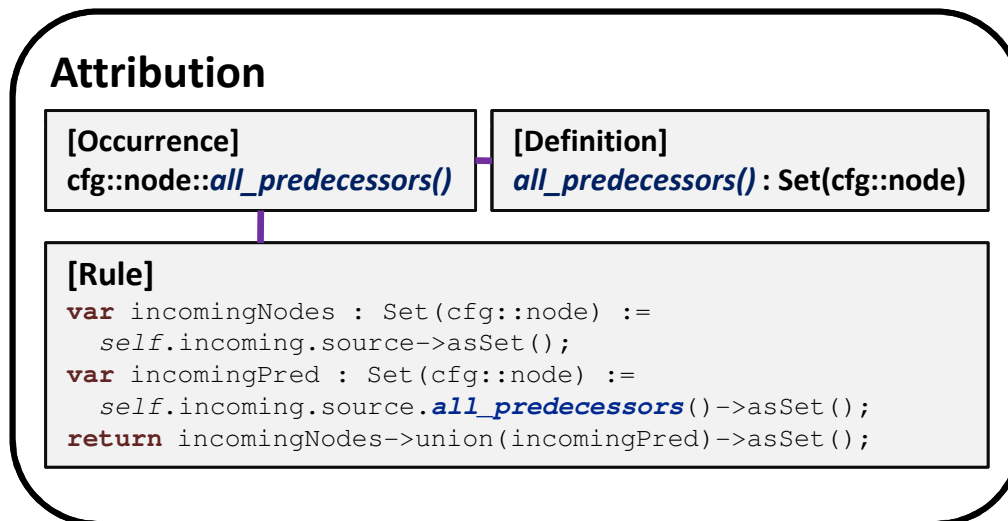
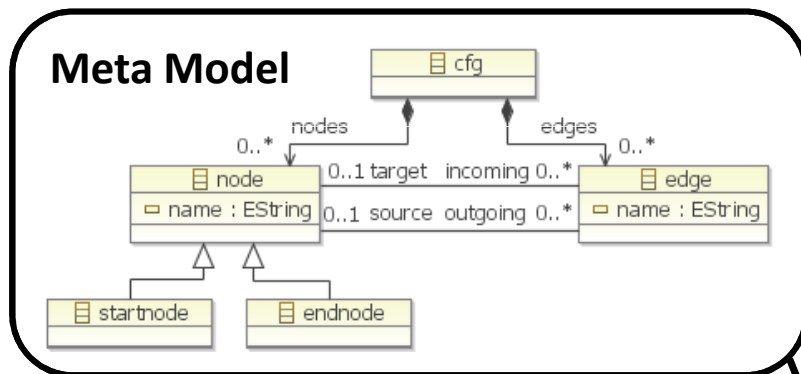


DFA equations/rules

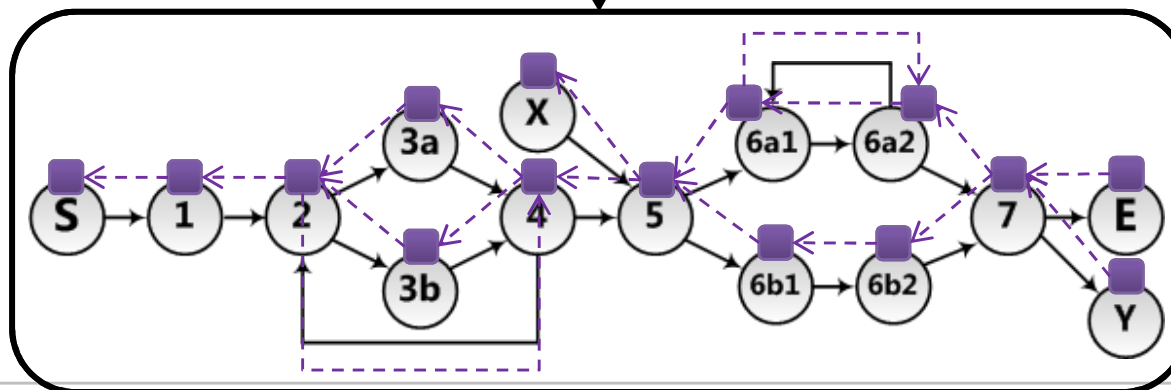
- › **Java-based** or
- › **Extended (Imperative) OCL**



Control-Flow Graph - Calculate *transitive predecessors*



Instantiation



Evaluation

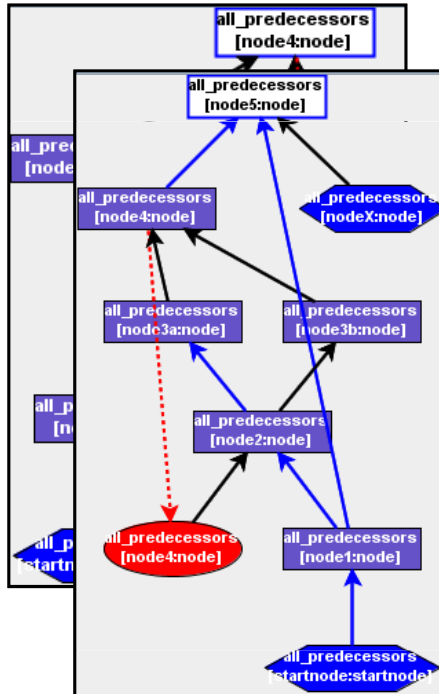
■ Traditional DFA Evaluation

- › Worklist algorithm (execute rule, add depending rules to worklist)
- › Problem: Requests dynamic, output dependencies not known!

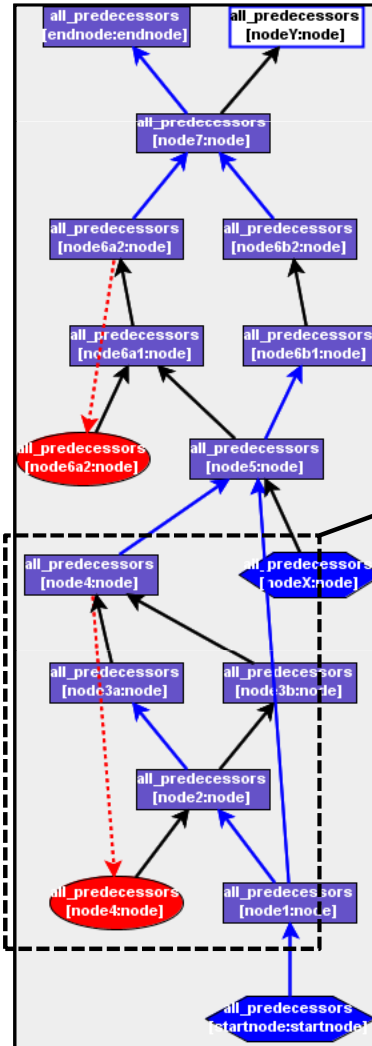
■ Dependency Chain Algorithm

- › Uses Tree-like Directed-Acyclic Graph structure
- 1. Build Chain
 - » **Record dependencies** and **execute** requested rule recursively
 - » Replace **cyclic dependencies** with virtual nodes
 - » **Merge chains** if necessary
- 2. Evaluate Chain
 - » **Identify unstable** virtual nodes
 - » Perform **bottom-up reevaluation** until stable
 - » Adjust dependency chain if necessary

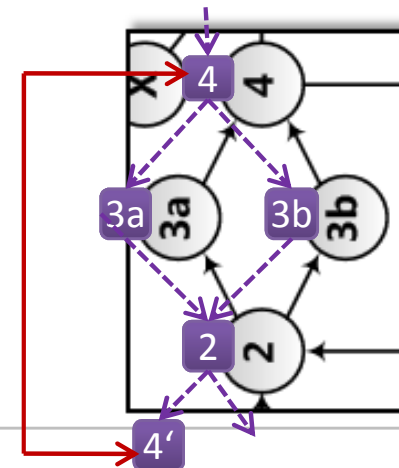
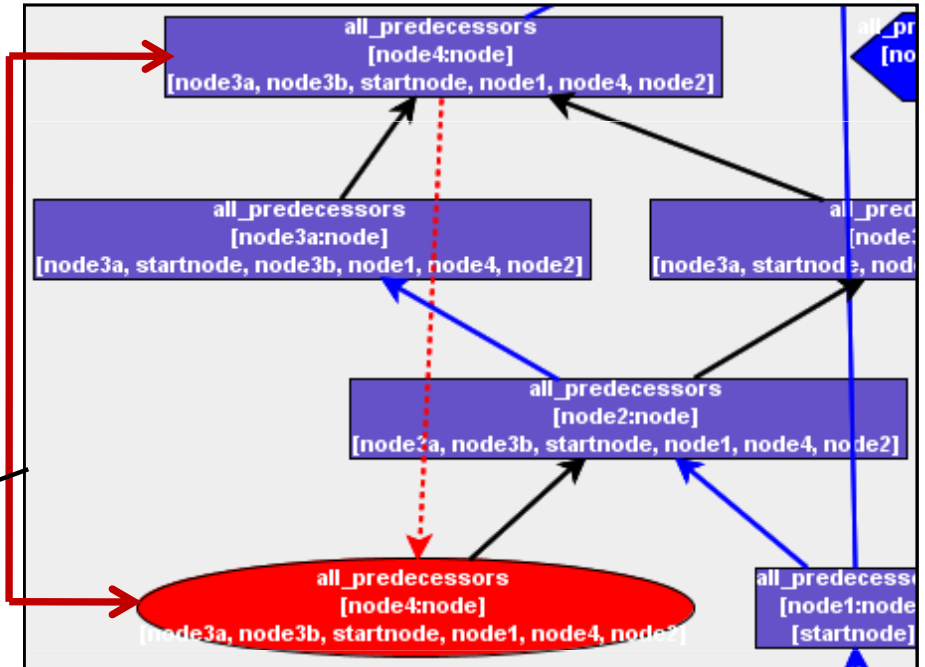
Build and Merge

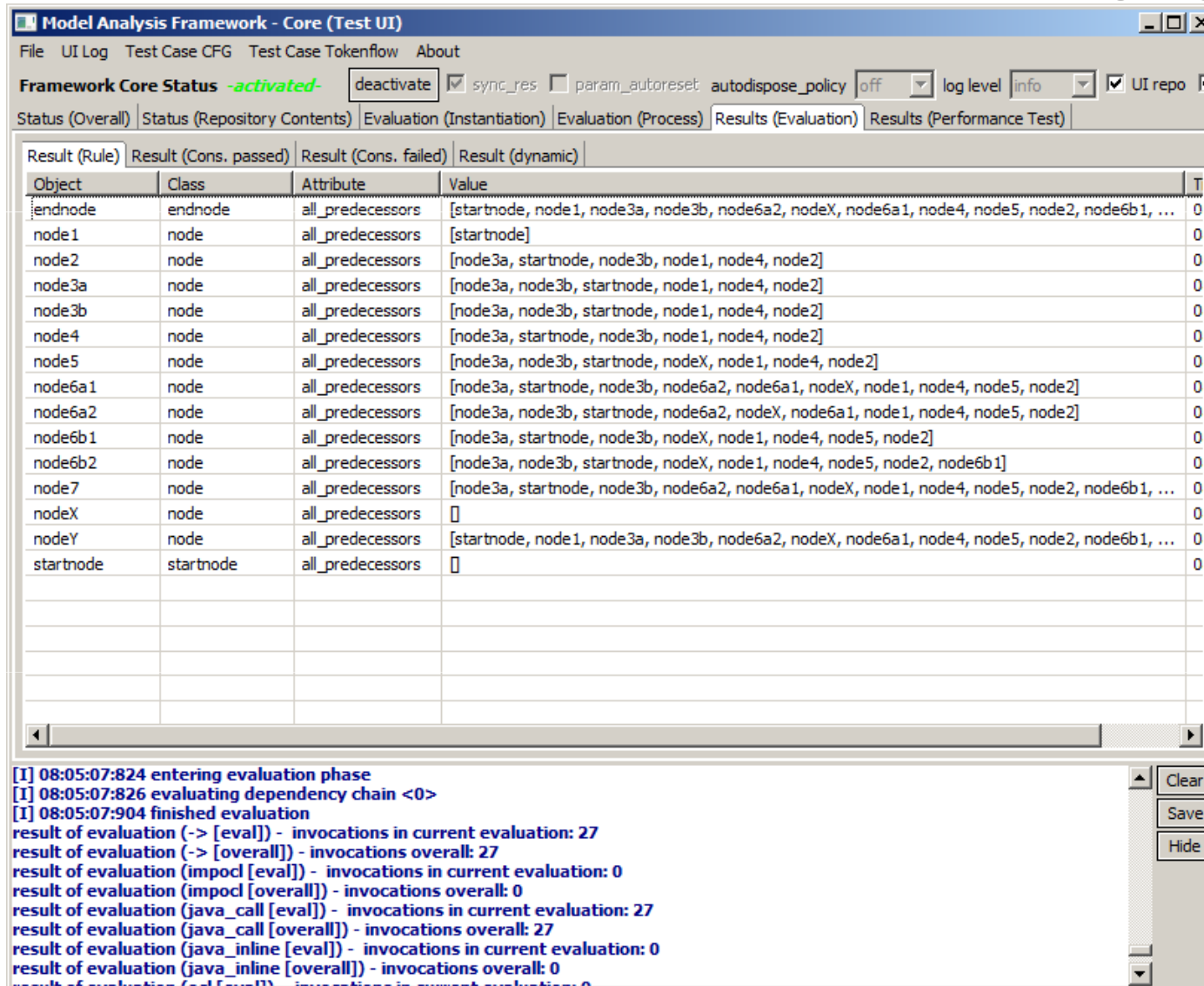


Built



Reevaluate until values are stable





Model Analysis Framework - Core (Test UI)

File UI Log Test Case CFG Test Case Tokenflow About

Framework Core Status **-activated-** deactivate sync_res param_autoreset autodispose_policy off log level info UI repo

Status (Overall) Status (Repository Contents) Evaluation (Instantiation) Evaluation (Process) Results (Evaluation) Results (Performance Test)

Result (Rule)	Result (Cons. passed)	Result (Cons. failed)	Result (dynamic)	
Object	Class	Attribute	Value	T
endnode	endnode	all_predecessors	[startnode, node 1, node3a, node3b, node6a2, nodeX, node6a1, node4, node5, node2, node6b1, ...]	0
node 1	node	all_predecessors	[startnode]	0
node2	node	all_predecessors	[node3a, startnode, node3b, node1, node4, node2]	0
node3a	node	all_predecessors	[node3a, node3b, startnode, node1, node4, node2]	0
node3b	node	all_predecessors	[node3a, node3b, startnode, node1, node4, node2]	0
node4	node	all_predecessors	[node3a, startnode, node3b, node1, node4, node2]	0
node5	node	all_predecessors	[node3a, node3b, startnode, nodeX, node1, node4, node2]	0
node6a1	node	all_predecessors	[node3a, startnode, node3b, node6a2, node6a1, nodeX, node1, node4, node5, node2]	0
node6a2	node	all_predecessors	[node3a, node3b, startnode, node6a2, nodeX, node6a1, node1, node4, node5, node2]	0
node6b1	node	all_predecessors	[node3a, startnode, node3b, nodeX, node1, node4, node5, node2]	0
node6b2	node	all_predecessors	[node3a, node3b, startnode, nodeX, node1, node4, node5, node2, node6b1]	0
node7	node	all_predecessors	[node3a, startnode, node3b, node6a2, node6a1, nodeX, node1, node4, node5, node2, node6b1, ...]	0
nodeX	node	all_predecessors	[]	0
nodeY	node	all_predecessors	[startnode, node 1, node3a, node3b, node6a2, nodeX, node6a1, node4, node5, node2, node6b1, ...]	0
startnode	startnode	all_predecessors	[]	0

[I] 08:05:07:824 entering evaluation phase
[I] 08:05:07:826 evaluating dependency chain <0>
[I] 08:05:07:904 finished evaluation
result of evaluation (-> [eval]) - invocations in current evaluation: 27
result of evaluation (-> [overall]) - invocations overall: 27
result of evaluation (impocl [eval]) - invocations in current evaluation: 0
result of evaluation (impocl [overall]) - invocations overall: 0
result of evaluation (java_call [eval]) - invocations in current evaluation: 27
result of evaluation (java_call [overall]) - invocations overall: 27
result of evaluation (java_inline [eval]) - invocations in current evaluation: 0
result of evaluation (java_inline [overall]) - invocations overall: 0

Clear Save Hide

<http://code.google.com/p/model-analysis-framework/>

Control-Flow Graph Analysis

Strongly Connected Components (SCC)

cfg::node::all_predecessors(): Set(tokenflow::Token)

```
var directPred : Set(node) := self.direct_predecessors();
var incomingPred : Set(node) :=
    self.direct_predecessors().all_predecessors()->asSet();
return directPred->union(incomingPred)
```

cfg::edge::scc_id(): Integer

```
var self_pred : Set(node) := self.all_predecessors()->including(self);
var all_pred : Set(node) := self.direct_predecessors().all_predecessors()->asSet();

if (all_pred==self_pred) then return self_pred->hashCode() else return 0 endif;
```

cfg::edge::scc_inedges(): Set(cfg::edge)

```
if (self.source = null or self.target = null) then return Set{} endif;

var inConnectors : Set(edge) := Set{};
var inSCC : Boolean := self.source.scc_id() != 0 and self.target.scc_id() != 0;

if (inSCC) then self.source.incoming->forEach(incomingEdge) {
    if (incomingEdge.source.scc_id() = self.source.scc_id())
        then inConnectors := inConnectors->union(incomingEdge.scc_inedges()->asSet())
        else inConnectors := inConnectors->union(Set{incomingEdge})->asSet() endif
} endif;

return inConnectors-self.scc_innereedges()
```

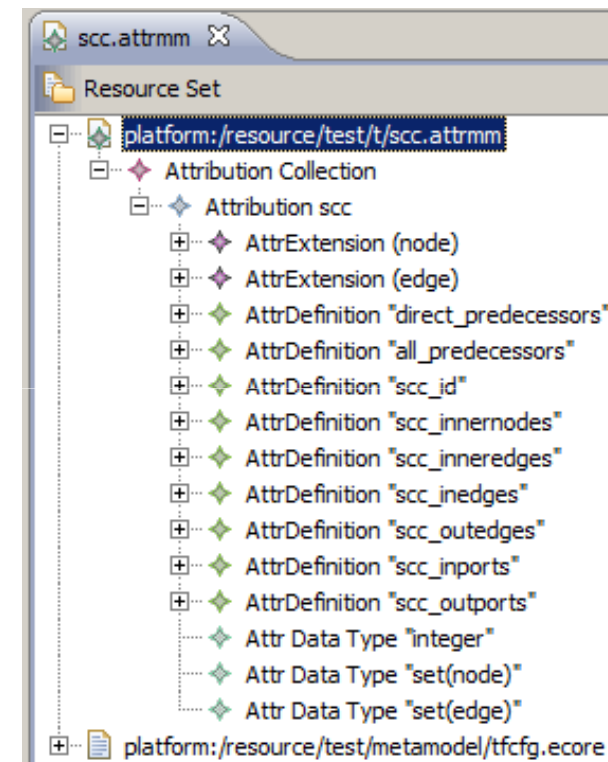
Reachability Analysis (compare traditional DFA)

cfg::node::is_reachable(): Boolean

```
return self.incoming.source.is_reachable()->includes(true)
```

cfg::node::is_live(): Boolean

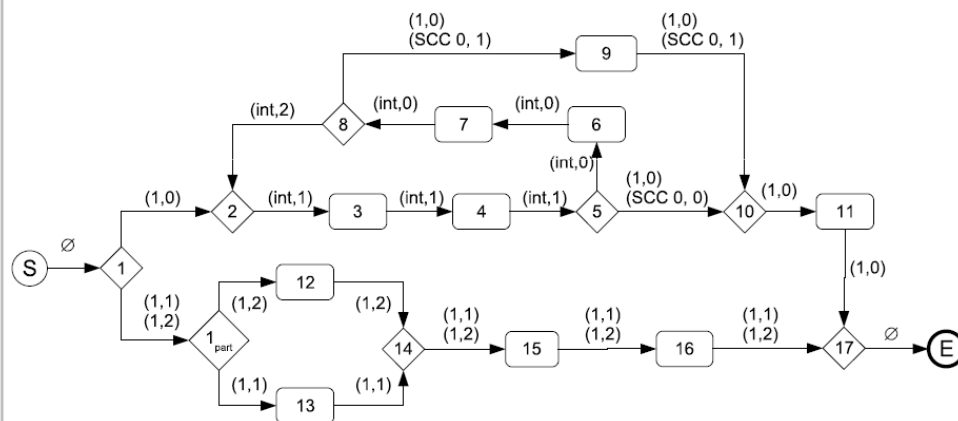
```
return self.outgoing.target.is_live()->includes(true)
```



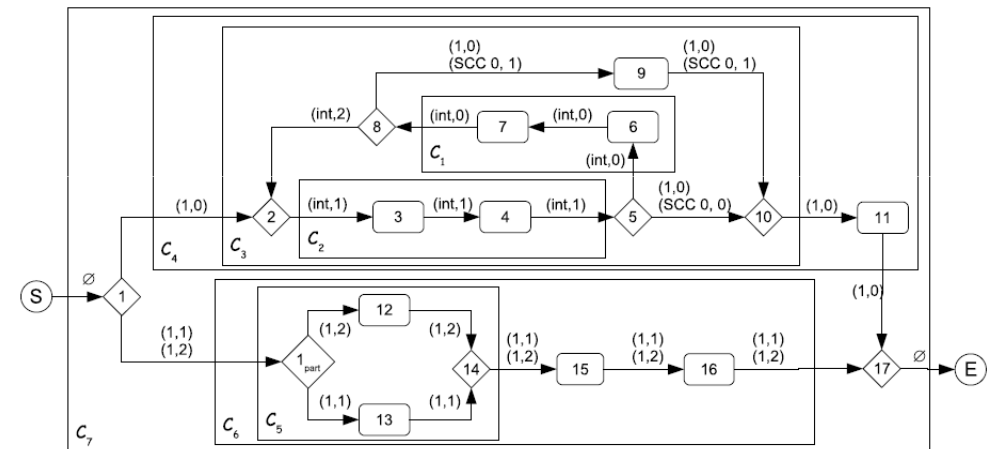
Token-flow Analysis [1]

- Extract **Single-Entry-Single-Exit (SESE)** components
- Required for validation, transformation, ...

1. Create and Merge Tokens



2. Identify components with identical token sets



[1] Mathias Götz, Stephan Roser, Florian Lautenbacher, and Bernhard Bauer. *Token Analysis of Graph-Oriented Process Models*. 13th IEEE International EDOC Conference, September 2009.

```

1. cfg::edge::tokens_initial() : Set(tokenflow::Token)

if (self.source = null) then return Set{} endif;

var tokens : Set(tokenflow::Token) := Set{};

self.source.incoming->forEach(incomingEdge)
{
    tokens := tokens->union(incomingEdge.tokens_initial())
};

var sccID : Integer := self.scc_id();
var sourceOutgoing : Integer = self.source.outgoing->size();
if (sccID=0 and sourceOutgoing>1)
then
{
    va
    ne
    va
    if
    th
    {
        self.tokens_initial()->forEach(token)
        {
            var entry : Set(tokenflow::Token) := originMap->get(token.splitNode);
        }
        if (entry = null) then entry := Set{} endif;
        {
            entry := entry->including(token);
            originMap->put(token.splitNode, entry);
        }
        if (entry->size()==token.outgoingCount)
        then
            tokens := tokens-entry
        endif;
    }
}
endif;
return tokens;
    
```

```

2. cfg::edge::tokens() : Set(tokenflow::Token)

var tokens : Set(tokenflow::Token) := Set{}->union(self.tokens_initial());
var originMap : Dict(String, Set(tokenflow::Token)) := Dict{};

self.tokens_initial()->forEach(token)
{
    var entry : Set(tokenflow::Token) := originMap->get(token.splitNode);
}
if (entry = null) then entry := Set{} endif;
{
    entry := entry->including(token);
    originMap->put(token.splitNode, entry);
}
if (entry->size()==token.outgoingCount)
then
    tokens := tokens-entry
endif;
return tokens;
    
```

```

3. cfg::edge::tokenstack() : JavaObject
Java-based DFA rule
    
```



Result (tokenstack at end node)

```

1<?xml version="1.0" encoding="UTF-8"?>
2<component>
3  <type> ACYCLIC </type>
4  <tokenset> [] </tokenset>
5  <inputedge> S->1 </inputedge>
6  <outputedge> 17->E </outputedge>
7  <inneredges>
8    <edge> 11->17 </edge>
9    <edge> 1->2 </edge>
10   <edge> 16->17 </edge>
11   <edge> 1->1part </edge>
12 </inneredges>
13 <trivialnodes>
14   <node> 17 </node>
15   <node> 1 </node>
16 </trivialnodes>
17 <subcomponents>
18   <component>
19     <type> SEQUENCE </type>
20     <tokenset> [(1,2,2)] </tokenset>
21     <inputedge> 1->2 </inputedge>
22     <outputedge> 11->17 </outputedge>
23     <inneredges>
24       <edge> 10->11 </edge>
25     </inneredges>
26     <trivialnodes>
27       <node> 11 </node>
28     </trivialnodes>
29     <subcomponents>
30       <component>
31         <type> CYCLIC </type>
32         <tokenset> [(1,2,2)] </tokenset>
33         <inputedge> 1->2 </inputedge>
34         <outputedge> 10->11 </outputedge>
35         <inneredges>
36           <edge> 5->10 </edge>
37           <edge> 9->10 </edge>
38           <edge> 8->9 </edge>
39         </inneredges>
40         <trivialnodes>
41           <node> 10 </node>
42           <node> 9 </node>
43         </trivialnodes>
44         <allinneredges>
45           <edge> 5->10 </edge>
46           <edge> 9->10 </edge>
47           <edge> 8->9 </edge>
48         </allinneredges>
49       </component>
50     </subcomponents>
51   </component>
52 </subcomponents>
53 </trivialnodes>
54 </trivialnodes>
55 </inneredges>
56 </outputedge>
57 </inputedge>
58 </tokenset>
59 </type>
60 </component>
    
```

- **Concept**
 - › Improve evaluation algorithm
 - › Practical and theoretical evaluation of performance

- **Implementation (MAF)**
 - › Provide graphical IDE
 - › Run as server

- **Additional Use Cases**
 - › Compute model metrics / check modeling guidelines
 - › „Traditional DFA“, e.g. on model of Java code
 - › Validation of (UML) models

Thank you for your attention!

Questions?