Semantics of Metamodeling

Formal Semantics of Metamodeling Frameworks
Ethan Jackson
ISIS, Vanderbilt University

Semantic Anchoring Infrastructure
Kai Chen
ISIS, Vanderbilt University

Chess Review
May 11, 2005
Berkeley, CA

Metamodels As Formal Objects

Pumping Lemma
If it is accepted by the DFA

? If Graph structure, type structure, containment, aspects, etc... are valid
Denotational Structural Semantics

1. Define an algebraic structure that is rich enough to express all of the modeling concepts. (e.g. a graph with typed vertices and edges). All realizations form a set.

2. Define constraints on this set so that only valid models are included.

3. A metamodel carves out a subset of this valid space.

4. A meta-metamodel exists if it is a metamodel that carves out the set of all metamodels.

Tailoring the Semantics

Metamodeling constraints

Topological Layer

Object Instantiation Layer

Extension Layer

Metamodel Constraints

$G = (V(G), E(G), E(G)_t; \rightarrow, \delta), \delta^{+}(I), \delta^{-}(I)$

$T_S = (M, G, I, AttrSpace, InAttr: AttrNames, AttrValues, InstAttrValue, InstType)$

$EX_{G,E} = \{M_{G,E}, E_{G,E}, ExtensionClass, ExtensionOf, ExtensionBase, IModelVertex, IModelEdge, IModelGraph\}$
The Layered Approach

By choosing a layer, one chooses the set of concepts that naturally suit the semantic domain.

Layer Preservation

Layers progressively adds information. This is stated formally as layer preservation.

Downcasting

\[(2.94)\quad \text{Downcast}_{k+1} : M_{k+1} \rightarrow M_k, \quad j > k > 1\]

\[(2.95)\quad \text{Downcast}_{k+1}(m) = m \cdot M_k\]

\[(2.96)\quad \text{Downcast}_{j+1}(m) = m \cdot M_j\]

Upcasting

\[(2.96)\quad \text{Upcast}_{k+1} : M_{k+1} \rightarrow M_k, \quad j > k > 1\]

\[(2.97)\quad \text{Upcast}_{k+1}(m) \in M_{k+1} \implies \text{Upcast}_{k+1}(m), M_{k+1} = m\]

\[(2.98)\quad \forall m \in M, \quad M_k = \text{Downcast}_{k+1}(M_{k+1})\]

Layer Preserving

Definition 2.45. Let \(k \geq 1\) and \(M = (M_1, M_2, \ldots, M_j)\) be a set where the \(n^{th}\) element is a set \(M_n\) such that \(M_0 \subseteq M_{n+1}\). The set \(M\) is layer preserving if:

\[(2.98)\quad \forall m \in M, \quad M_1 = \text{Downcast}_{k+1}(M_{k+1})\]
Applications

Giotto Metamodel is a **formal** object that simultaneously describes a modeling environment and a structural semantics. MoC is descriptive, yet "baggage-free" because it builds off of the formal definition of the metamodel.

Semantic Anchoring Infrastructure

- **Semantic Unit**
  - A well-defined DSML that captures the semantics of a particular model of computation.

- **Semantic Anchoring**
  - Define the semantics of a DSML through the transformational specification to a semantic unit.
A Semantic Unit for Timed Automata Based Modeling Language

- Common semantic domain for varied timed automata based modeling languages.
  - Guard
  - Priority
  - Synchronization

FSM Metamodel
FSM Model

Metamodel for AsmL Abstract Data Model
**AsmL Abstract Data Model**

```plaintext
interface Event
  structure ModelEvent implements Event
  structure LocalEvent implements Event

class FSM
  id as String
  var outputEvents as Seq of ModelEvent
  var localEvents as Set of LocalEvent
  var initialState as State
  children as Set of State

class State
  id as String
  var active as Boolean = false
  var initial as Boolean
  var initialState as State?
  var master as State?
  var owns as Set of State
  var onTransitions as Set of Transition

class Transition
  id as String
  var guard as Boolean
  var preemtive as Boolean
  var triggerEvent as Event?
  var outputEvent as Event?
  src as State
```

**AsmL Behavioral Semantic Specifications**

```plaintext
invokeSlaves (fsm as FSM, s as State, c as Event) =
  require isHierarchicalState (s) and s.initialState <> null
  step
  let ids as State = getActiveSlave (fsm, s, c)
  step
  let pt as Transition? = getPreemptiveTransition (fsm, ids, c)
  step
  if pt <> null then
doTransition (fsm, ids, pt)
  else
    step
    if isHierarchicalState (ids) then
      invokeSlaves (fsm, ids, c)
    step
    let nxt as Transition? = getNonpreemptiveTransition (fsm, ids, c)
    step
    if nxt <> null then
doTransition (fsm, ids, nxt)
  doTransition (fsm as FSM, s as State, t as Transition) =
  require t.active
  step
  endState (s)
  step
  if t.outputEvent <> null then
  emitEvent (fsm, t.outputEvent)
```
Transformational Specifications

AsmL Data Model in XML Format

Chess Review, May 11, 2005 15

Chess Review, May 11, 2005 16
AsmL Data Model

Distributed Real-time System (1)

• Abstract Syntax (1)

• Component Interactions
Distributed Real-time System (2)

- Abstract Syntax (2)

- Component Behaviors

Distributed Real-time System (3)

• AsmL Abstract Data Structure

```plaintext
class State
  id      as String
  option  as STATEOPTION

class Transition
  id      as String
  option  as TRANSITIONOPTION

abstract class TimeAutomata
  var currentState                 as State? = null
  abstract property states         as Set of State
  abstract property transitions    as Set of Transition
  abstract property localClocks    as Set of Clock
  abstract property outTransitions as Map of <State, Set of Transition>
  abstract property srcState      as Map of <Transition, State>
  abstract property dstState      as Map of <Transition, State>
  abstract property syns          as Map of <Transition, (SignalChannel, SYNMODE)>

abstract class DRTSystem

abstract property timeAutomatas  as Set of TimeAutomata
abstract property signalChannels as Set of SignalChannel
abstract property signalRouters as Set of SignalRouter
var activeAutomatas             as Set of TimeAutomata = {}
```
• **AsmL Behavior Semantics Specification**

// In the current clock time, whether the time guard of an transition is true
IsTimeGuardTrueNow (t as Transition) as Boolean
require t in me.transitions
step return TimeGuard (t)

// In the next clock time, whether the time guard of an transition is true
IsTimeGuardTrueNext (t as Transition) as Boolean
require t in me.transitions
step
  forall c in globalClocks
    c.Go ()
  forall c in me.localClocks
    c.Go ()
step
  let next = IsTimeGuardTrueNow (t)
step
  forall c in globalClocks
    c.Back ()
  forall c in localClocks
    c.Back ()
step return next

• **Asynchronous Component Interaction**
• **Simulation**
• **Verification**