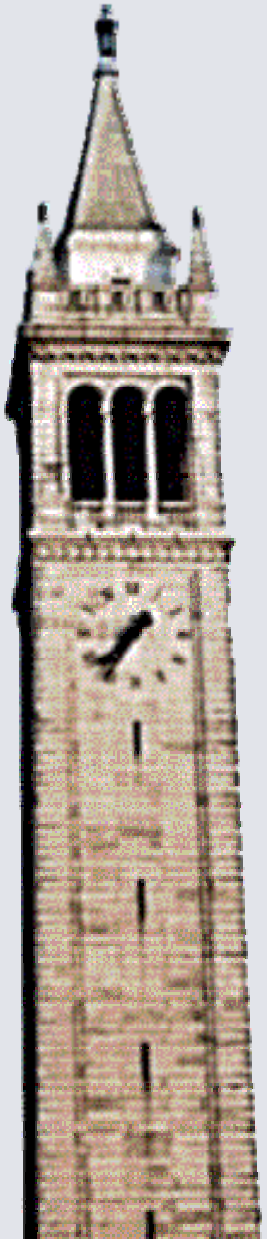
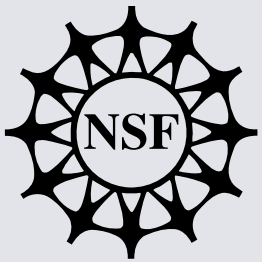
A Semantic Unit for Timed Automata Based Modeling Languages

Edited and presented by
Kai Chen
ISIS, Vanderbilt University

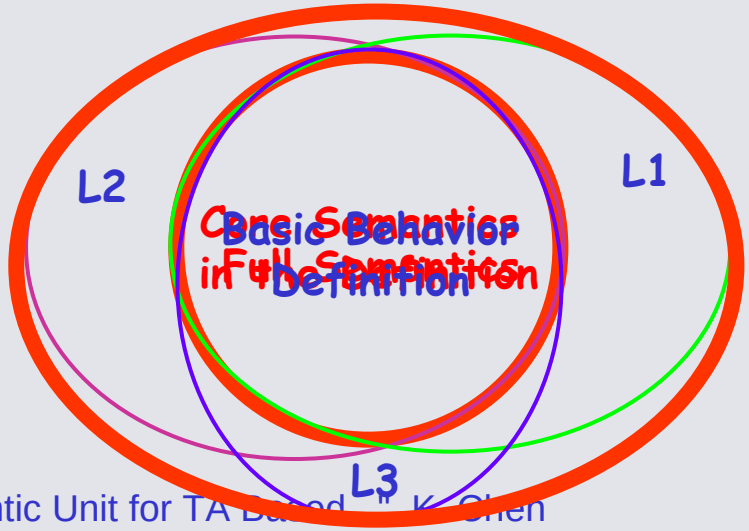


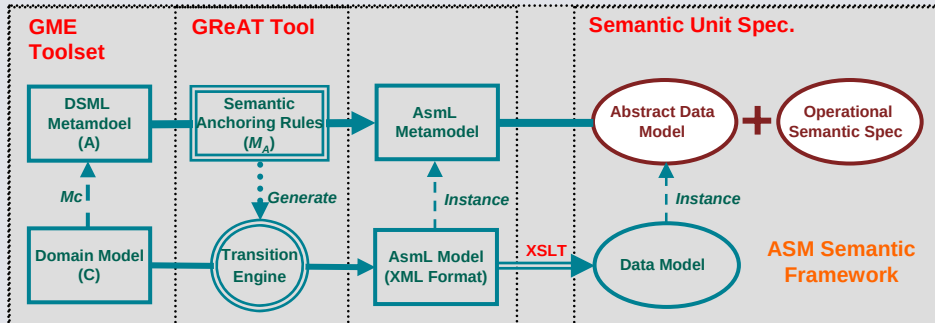


Chess Review
November 21, 2005
Berkeley, CA

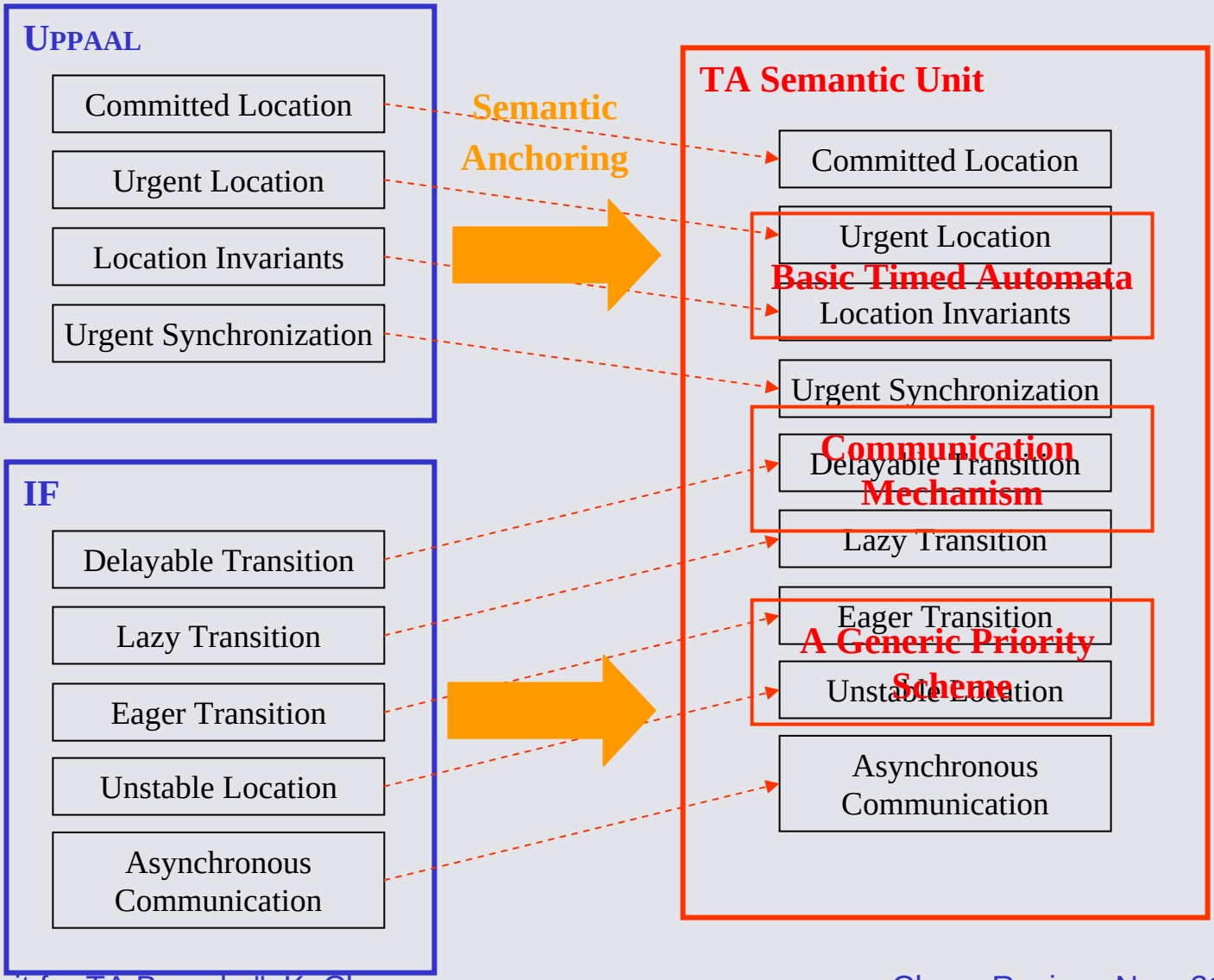
Semantic Units

- Semantic units capture the semantics of basic behavioral categories, such as FSM, DE, TA and SR.
- One basic behavior category may have various tool-defined variants that have similar but different syntax and semantics.
  - Timed Automata variants: the UPPAAL, IF and Kronos languages.
  - They extend the basic Timed Automata to express
    - Interaction among a network of automata
    - Action (transition) priorities.
- What kind of semantics is needed to be captured by a semantic unit?
- Do deep analysis on different language variants and make clear what is the essential semantics for this behavioral category.





Timed Automata Semantic Unit



A Priority-Oriented TA Semantic Unit

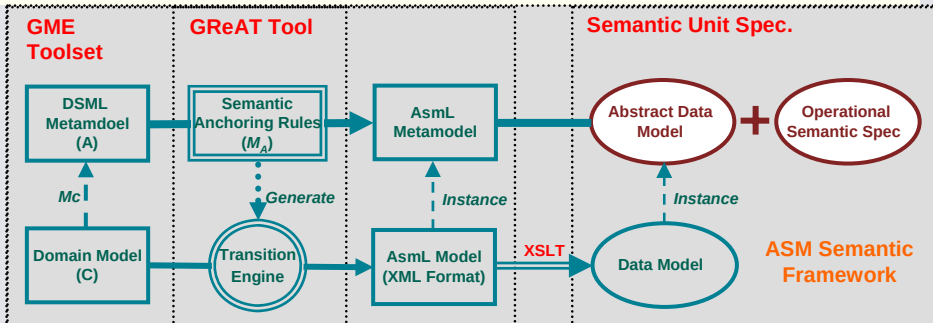
- Semantic Concepts
  - Original Timed Automata Definition
  - Communication among a network of automata
    - Shared variables
    - Synchronous communication
    - Asynchronous communication (via mapping to the synchronous communication through model transformation).
  - A generic priority scheme
    - The bottom priority (the time progress priority)
    - The top priority (enable modeling atomic actions)
    - The urgent priority, which has a series of urgency degrees
- TA Semantic Unit Overview
  - A real-time system contains a set of concurrent components.
  - The behavior of each component is modeled by a timed automaton.
  - Components communicate among each other through shared variables and synchronization.
  - The priority of an action (transition) can be dynamically updated with respect to time progress.
  - Enabled actions with higher priorities will block actions with lower priorities.
  - Non-determinism is supported by allowing multiple enabled actions with the same priority.

TA Semantic Unit Specification

- The Abstract Data Model captures the abstract syntax of TA Semantic Unit.
- A TA Semantic Unit model is an instance of the Abstract Data Model.
- Including Operations and Transformational Rules that interpret the TA Semantic Unit Abstract Data Model.

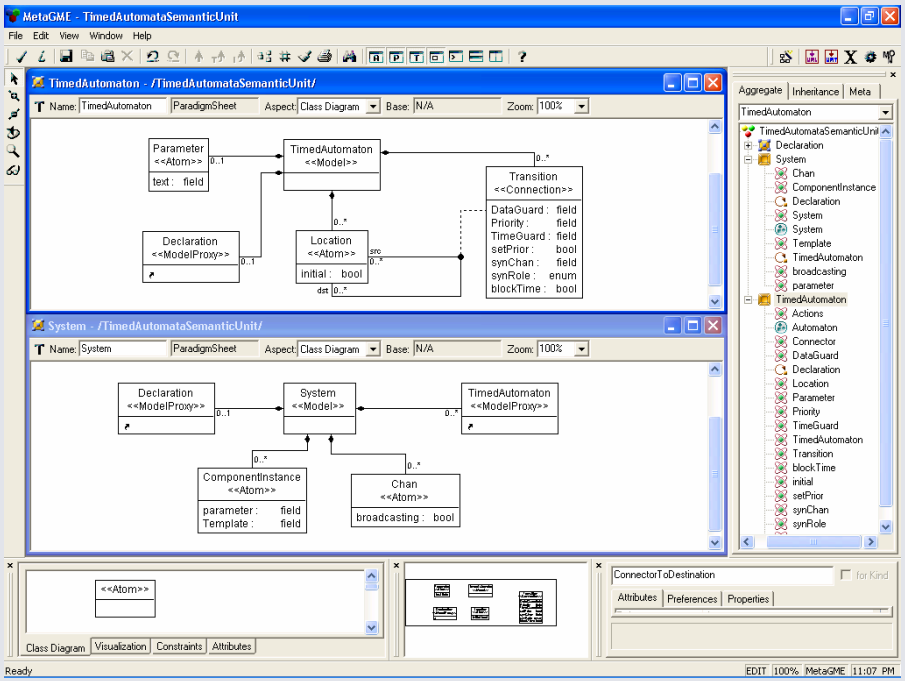
```
class Clock
  var time         as Double = 0
const CLOCKUNIT    as Double
var globalClocks   as Set of Clock = {}
var TimeBlocked    as Boolean =  false
class Location
  const id              as String
  const initial         as Boolean
  const outTransitions  as Set of Transition
class Transition
  const id              as String
  const blockTime       as Boolean
  const setPrior        as Boolean
  const dstLocationID   as String
enum SYNROLE
  SEND
  RECEIVE
class SignalChannel
  const id         as String
  const broadcast  as Boolean
  var senders      as Set of
    (TimedAutomaton, Transition) = {}
  var receivers    as Set of
    (TimedAutomaton, Transition) = {}
```

```
class RTSystem
  Run()
    step until fixpoint
      step RegisterSignalChannels()
      step let T as (TimedAutomaton?,
Transition?) = GetNextTransition()
      step if TimeBlocked then
          if T.Second = null then
            error("The system is blocked.")
          else
            T.First.DoTransition(t.Second)
        else
          if T.Second = null then
            TimeProgress()
          else
```

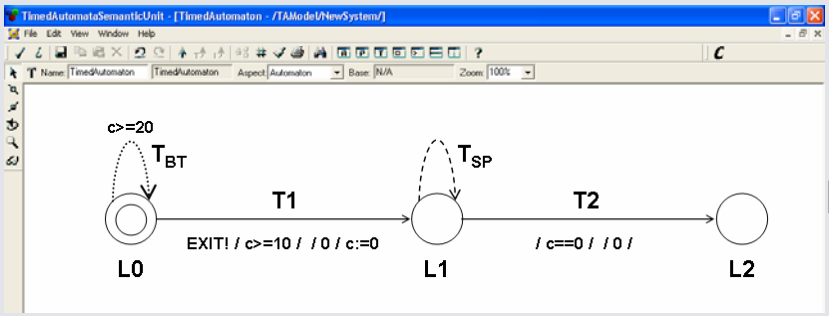


"A Semantic Unit for TA Based...", K. Chen

Modeling Language Specification for TA Semantic Unit



TASU Metamodel



A TASU Automaton (ComponentKindA)

```
class ComponentKindA extends TimedAutomaton
  L0  as Location   = new Location ("L0",true, {T1,TBT})
  L1  as Location   = new Location ("L1",false,{T2,TSP})
  L2  as Location   = new Location ("L2", false, {})
  T1  as Transition = new Transition ("T1",false,false,"L1")
  T2  as Transition = new Transition ("T2",false,false,"L2")
  TBT as Transition = new Transition ("TBT",true,false,"L0")
  TSP as Transition = new Transition ("TSP",false,true,"L1")
  c   as Clock      = new Clock ()
  override property locations as Set of Location
    get return {L0, L1, L2}
  override property transitions as Set of Transition
    get return {T1, T2, TBT, TSP}
  override property localClocks as Set of Clock
    get return {c}
  override property syns as Map of <Transition,
    (SignalChannel, SYNROLE)>
    get return { T1 -> (EXIT, SEND)}
  override TimeGuard (t as Transition) as Boolean
    match t.id
      "T1" : return c.time >= 10
      "T2" : return c.time == 0
      "TBT": return c.time >= 20
      _    : return true
  override DataGuard (t as Transition) as Boolean
    match t.id
      _    : return true
  override DoAction (t as Transition)
    match t.id
      "T1": c.time := 0
      _   : skip
  override Priority (t as Transition) as Integer
    match t.id
      _   : return 0
```

Generated TASU Data Model