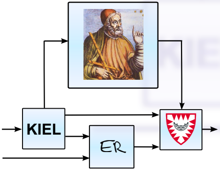
Executing SyncCharts with Ptolemy

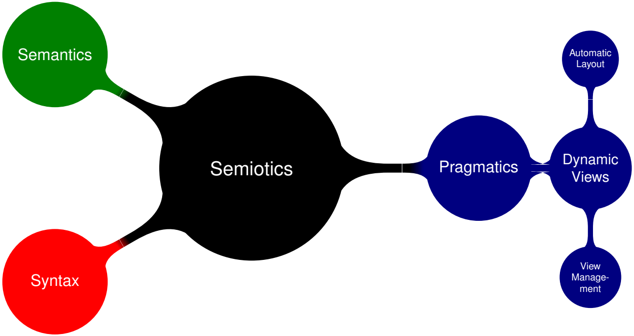
Christian Motika

Real-Time Systems and Embedded Systems Group
Department of Computer Science
Christian-Albrechts-Universität zu Kiel, Germany

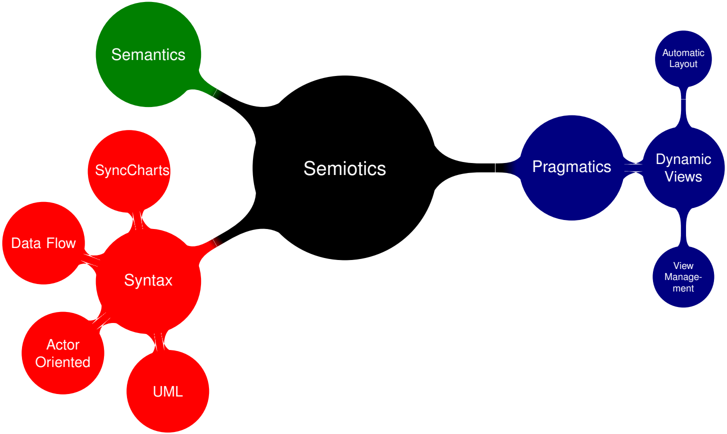


SYNCHRON Workshop 2010
Frejús, 30.11.2010

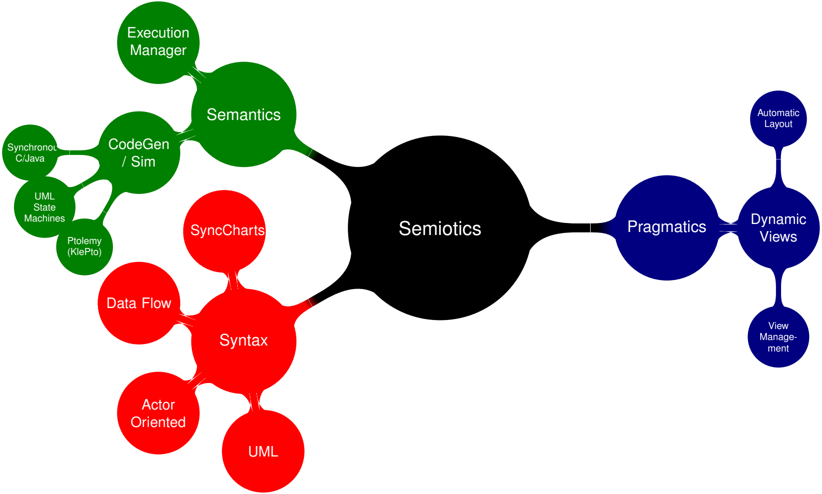
KIELER Semiotics



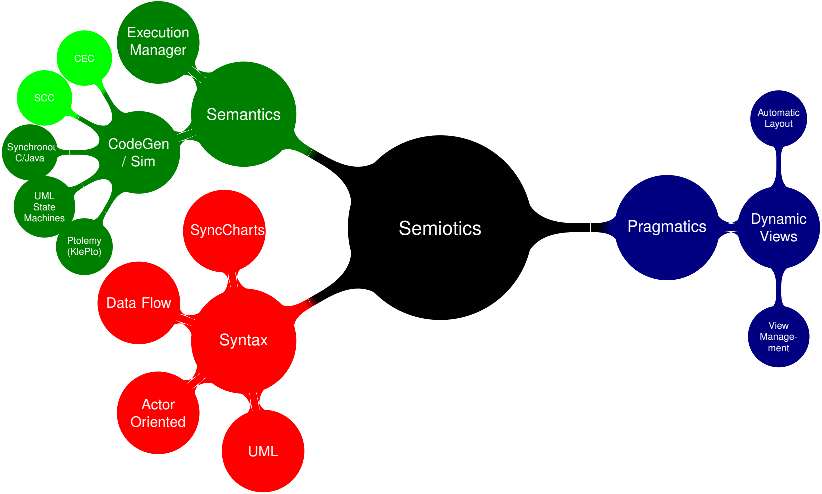
KIELER Semiotics



KIELER Semiotics

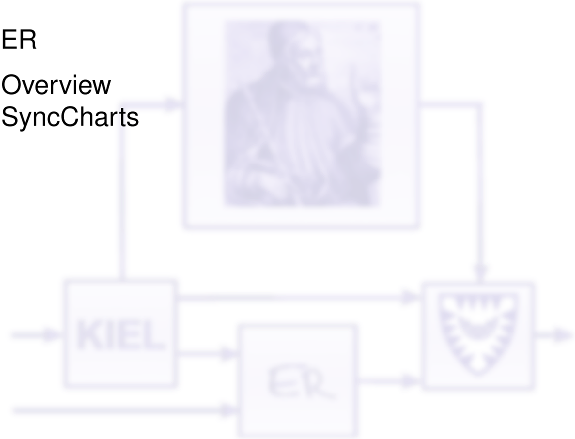


KIELER Semiotics



Overview

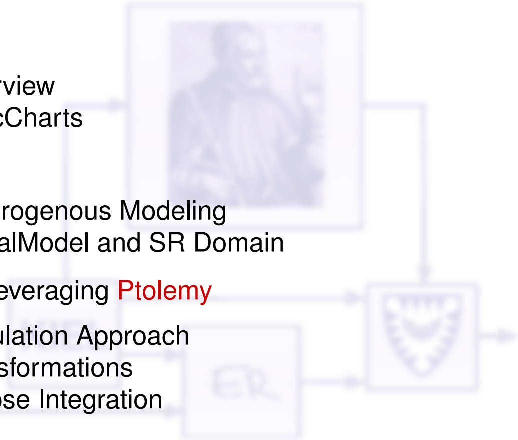
- KIELER
  - Overview
  - SyncCharts



Overview

- KIELER
  - Overview
  - SyncCharts
- Ptolemy
  - Heterogenous Modeling
  - ModalModel and SR Domain

Overview



- KIELER
  - Overview
  - SyncCharts
- Ptolemy
  - Heterogenous Modeling
  - ModalModel and SR Domain
- KIELER leveraging Ptolemy
  - Simulation Approach
  - Transformations
  - Eclipse Integration

Overview

- KIELER
  - Overview
  - SyncCharts
- Ptolemy
  - Heterogenous Modeling
  - ModalModel and SR Domain
- KIELER leveraging Ptolemy
  - Simulation Approach
  - Transformations
  - Eclipse Integration
- Summary

What is KIELER?

- Kiel Integrated Environment for Layout Eclipse Rich Client

What is KIELER?

- Kiel Integrated Environment for Layout Eclipse Rich Client
- Modeling platform and test bed

What is KIELER?

- Kiel Integrated Environment for Layout Eclipse Rich Client
- Modeling platform and test bed
  - Improve pragmatics

What is KIELER?

- Kiel Integrated Environment for Layout Eclipse Rich Client
- Modeling platform and test bed
  - Improve pragmatics
- Open source and Eclipse based (plug-ins)

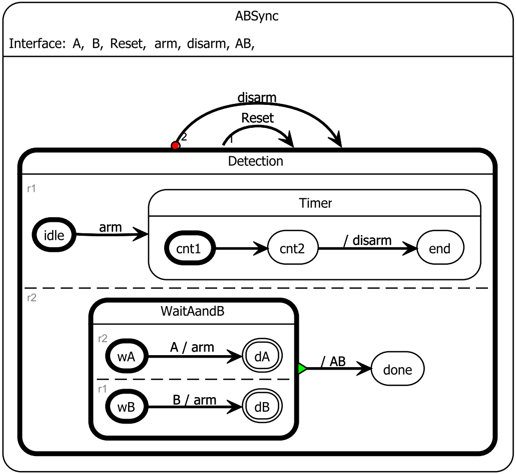
What is KIELER?

- Kiel Integrated Environment for Layout Eclipse Rich Client
- Modeling platform and test bed
  - Improve pragmatics
- Open source and Eclipse based (plug-ins)
- General concepts:

What is KIELER?

- Kiel Integrated Environment for Layout Eclipse Rich Client
- Modeling platform and test bed
  - Improve pragmatics
- Open source and Eclipse based (plug-ins)
- General concepts:
  - Generic approaches
  - Symbiosis w/ Eclipse technologies (e.g., EMF, GMF, TMF, Xpand, Xtend)
  - Interfaces to other tools (Ptolemy, Papyrus)

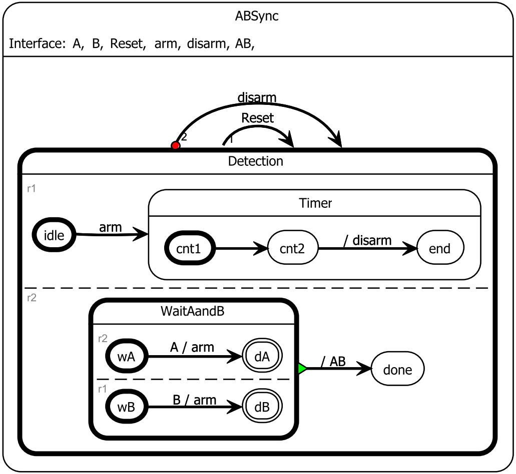
SyncCharts



Charles André, Computing SyncCharts Reactions, 2003

- Statechart dialect
- Mealy machine

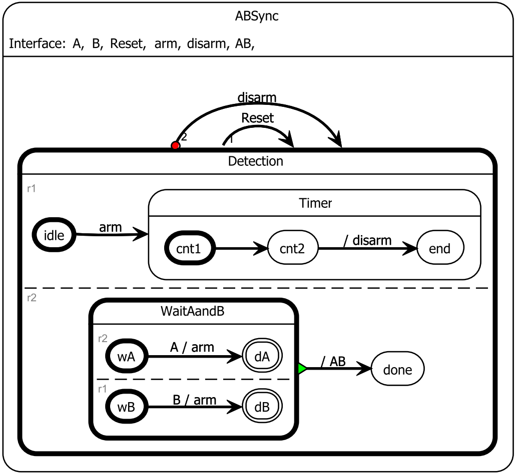
SyncCharts



Charles André, Computing SyncCharts Reactions, 2003

- Statechart dialect
- Mealy machine with
  - Parallelism, hierarchy, compound events, broadcast

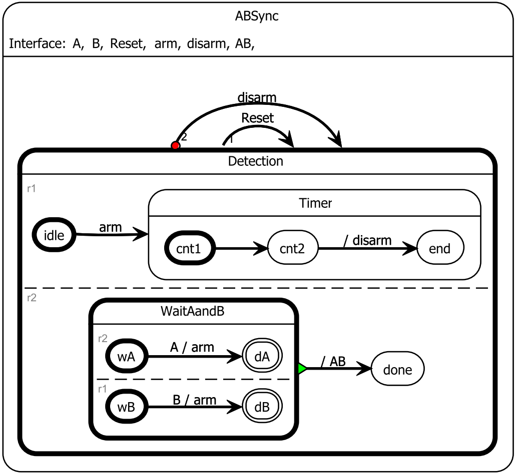
SyncCharts



Charles André, Computing SyncCharts Reactions, 2003

- Statechart dialect
- Mealy machine with
  - Parallelism, hierarchy, compound events, broadcast
- Graphical notation for the Esterel synchronous language

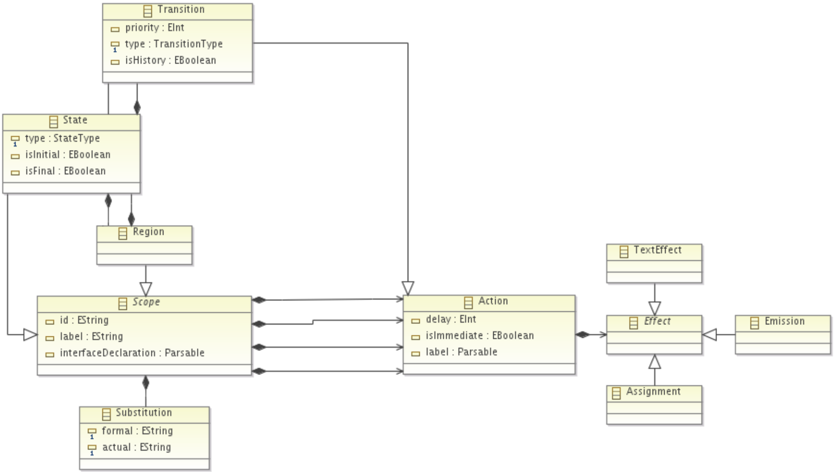
SyncCharts

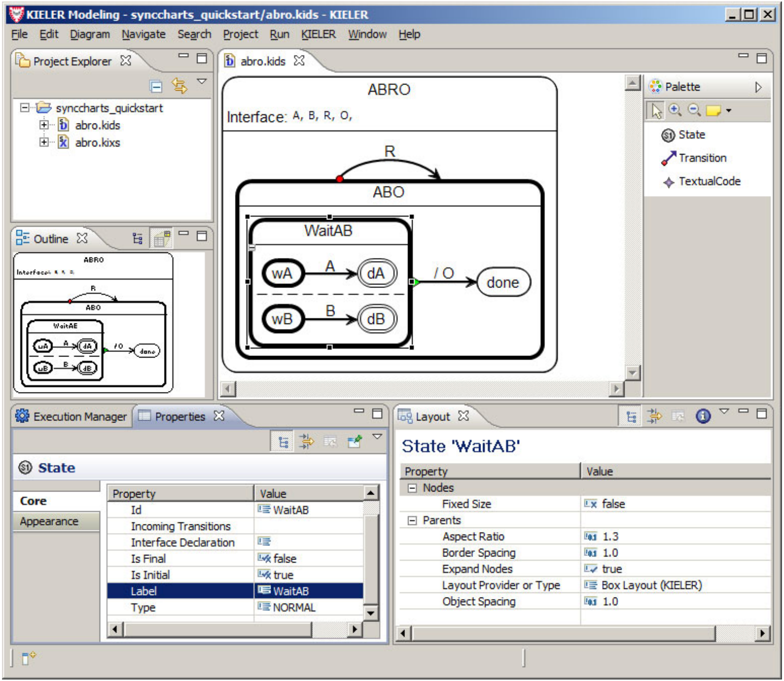


Charles André, Computing SyncCharts Reactions, 2003

- Statechart dialect
- Mealy machine with
  - Parallelism, hierarchy, compound events, broadcast
- Graphical notation for the Esterel synchronous language
- Synchrony hypothesis
  - Discrete ticks
  - Computations take no time

Abstract Syntax (EMF)




KIELER Modeling - syncchарts_quickstart/abro.kids - KIELER
File Edit Diagram Navigate Search Project Run KIELER Window Help
Project Explorer
syncchарts_quickstart
abro.kids
abro.kixs
abro.kids
ABRO
Interface: A, B, R, O,
R
ABO
WaitAB
wA
A
dA
wB
B
dB
/ O
done
Palette
State
Transition
TextualCode
Outline
Execution Manager
Properties
State
Core
Appearance
Property
Value
Id
WaitAB
Incoming Transitions
Interface Declaration
Is Final
false
Is Initial
true
Label
WaitAB
Type
NORMAL
Layout
State 'WaitAB'
Property
Value
Nodes
Fixed Size
false
Parents
Aspect Ratio
1.3
Border Spacing
1.0
Expand Nodes
true
Layout Provider or Type
Box Layout (KIELER)
Object Spacing
1.0

Overview

- KIELER
  - Overview
  - SyncCharts
- Ptolemy
  - Heterogenous Modeling
  - ModalModel and SR Domain
- KIELER leveraging Ptolemy
  - Simulation Approach
  - Transformations
  - Eclipse Integration
- Summary

Ptolemy

- „The Ptolemy project studies heterogeneous modeling, simulation, and design of concurrent systems.“

Introduction to Ptolemy II, UC Berkeley

Ptolemy

- „The Ptolemy project studies heterogeneous modeling, simulation, and design of concurrent systems.“

  Introduction to Ptolemy II, UC Berkeley

- Executable Models to describe behavior of reactive systems

Ptolemy

- „The Ptolemy project studies heterogeneous modeling, simulation, and design of concurrent systems.“

  Introduction to Ptolemy II, UC Berkeley

- Executable Models to describe behavior of reactive systems
- Ptolemy models are a set of interacting components → Actor-Oriented Design

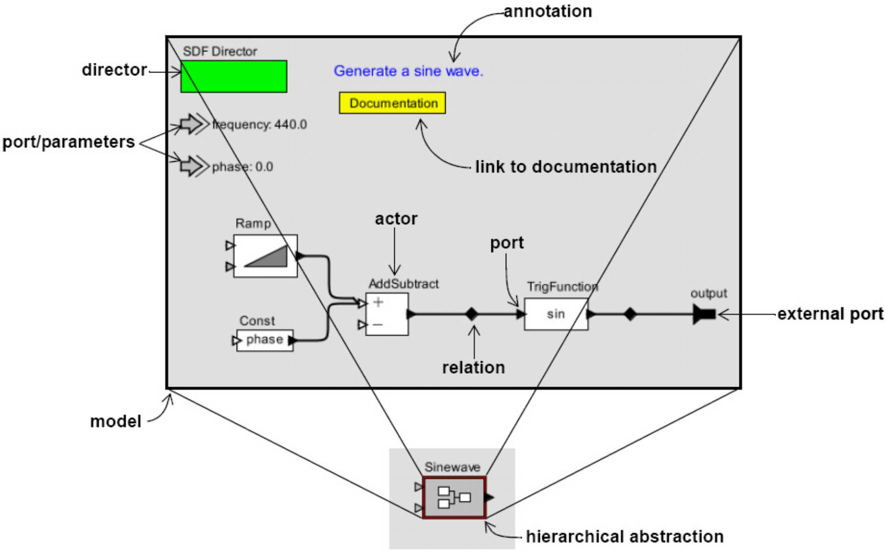
Ptolemy

- „The Ptolemy project studies heterogeneous modeling, simulation, and design of concurrent systems.“

  Introduction to Ptolemy II, UC Berkeley

- Executable Models to describe behavior of reactive systems
- Ptolemy models are a set of interacting components $\rightarrow$ Actor-Oriented Design
- Constructed under a model of computation (MoC)

Ptolemy Actor Example



Introduction to Ptolemy II, UC Berkeley, 2008

Model of Computation

- Defines interaction of system components

Model of Computation

- Defines interaction of system components
  - Semantics of a model

Model of Computation

- Defines interaction of system components
  - Semantics of a model
- Ptolemy Model can have more than one MoC

Model of Computation

- Defines interaction of system components
  - Semantics of a model
- Ptolemy Model can have more than one MoC
- MoC domains/directors:
  - Process Networks (PN)

Model of Computation

- Defines interaction of system components
  - Semantics of a model
- Ptolemy Model can have more than one MoC
- MoC domains/directors:
  - Process Networks (PN)
  - Continuous Time (CT)

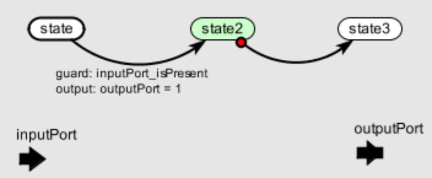
Model of Computation

- Defines interaction of system components
  - Semantics of a model
- Ptolemy Model can have more than one MoC
- MoC domains/directors:
  - Process Networks (PN)
  - Continuous Time (CT)
  - Finite State Machines (FSM)
  - Synchronous Reactive (SR)

Model of Computation

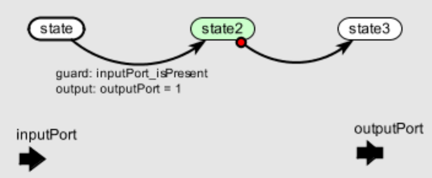
- Defines interaction of system components
  - Semantics of a model
- Ptolemy Model can have more than one MoC
- MoC domains/directors:
  - Process Networks (PN)
  - Continuous Time (CT)
  - Finite State Machines (FSM)
  - Synchronous Reactive (SR)
  - ...

ModalModel Domain



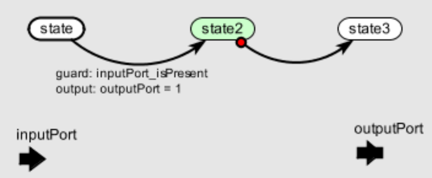
- Entities not actors but states

ModalModel Domain



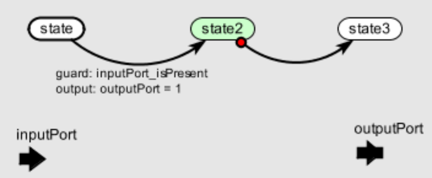
- Entities not actors but states
- Execution: Strictly ordered sequence of state transitions

ModalModel Domain



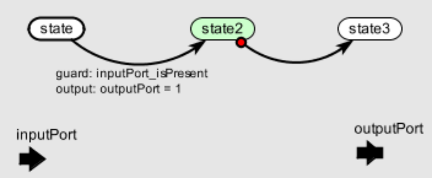
- Entities not actors but states
- Execution: Strictly ordered sequence of state transitions
- Build-in expression language to evaluate guards

ModalModel Domain



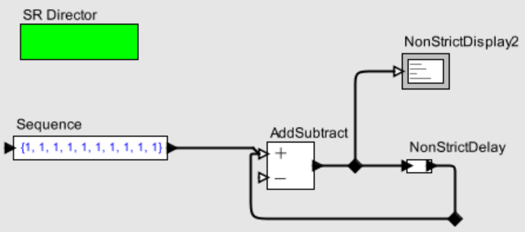
- Entities not actors but states
- Execution: Strictly ordered sequence of state transitions
- Build-in expression language to evaluate guards
- Refinements (multiple)

ModalModel Domain



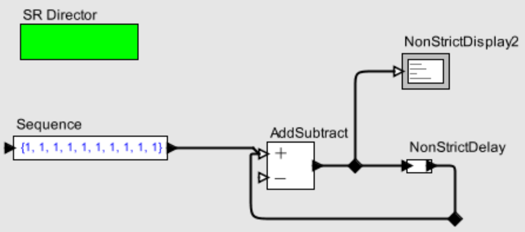
- Entities not actors but states
- Execution: Strictly ordered sequence of state transitions
- Build-in expression language to evaluate guards
- Refinements (multiple)
- Reset and preemptive transitions

Synchronous Reactive Domain



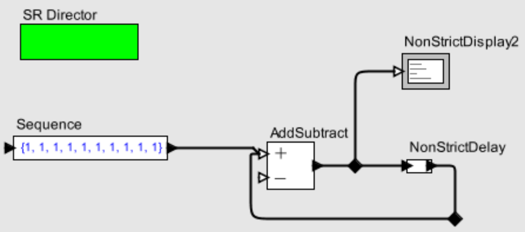
- Zero-Delay blocks

Synchronous Reactive Domain



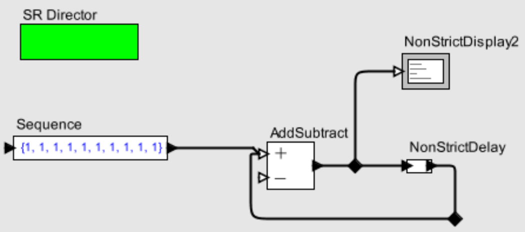
- Zero-Delay blocks
- Instantaneous communication

Synchronous Reactive Domain



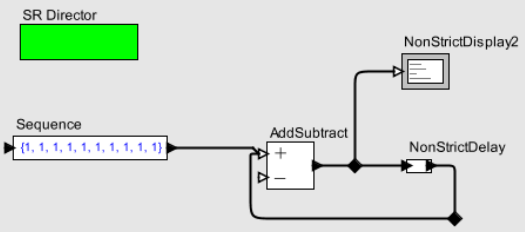
- Zero-Delay blocks
- Instantaneous communication
- Feedback

Synchronous Reactive Domain



- Zero-Delay blocks
- Instantaneous communication
- Feedback
- Fixed point ⇔ Stable state

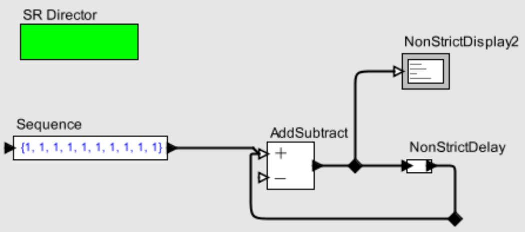
Synchronous Reactive Domain



- Zero-Delay blocks
- Instantaneous communication
- Feedback

- Fixed point ⇔ Stable state
- Values from flat lattice

Synchronous Reactive Domain



- Zero-Delay blocks
- Instantaneous communication
- Feedback
- Fixed point ⇔ Stable state
- Values from flat lattice
- Determinism ⇔ Unique solution

Overview

- KIELER
  - Overview
  - SyncCharts
- Ptolemy
  - Heterogenous Modeling
  - ModalModel and SR Domain
- KIELER leveraging Ptolemy
  - Simulation Approach
  - Transformations
  - Eclipse Integration
- Summary

Ptolemy Simulation Engine

- Mapping SyncCharts to Ptolemy:
  Mealy machine

Ptolemy Simulation Engine

- Mapping SyncCharts to Ptolemy:
  Mealy machine $\leftrightarrow$ ModalModel

Ptolemy Simulation Engine

- Mapping SyncCharts to Ptolemy:

  Mealy machine $\leftrightarrow$ ModalModel

  Orthogonality

Ptolemy Simulation Engine

- Mapping SyncCharts to Ptolemy:

Mealy machine	$\leftrightarrow$	ModalModel
Orthogonality	$\leftrightarrow$	Concurrent Actors (inherent)

Ptolemy Simulation Engine

- Mapping SyncCharts to Ptolemy:

Mealy machine	$\leftrightarrow$	ModalModel
Orthogonality	$\leftrightarrow$	Concurrent Actors (inherent)
Hierarchy		

Ptolemy Simulation Engine

- Mapping SyncCharts to Ptolemy:

Mealy machine	$\leftrightarrow$	ModalModel
Orthogonality	$\leftrightarrow$	Concurrent Actors (inherent)
Hierarchy	$\leftrightarrow$	Compound Actors, state refinements

Ptolemy Simulation Engine

- Mapping SyncCharts to Ptolemy:

Mealy machine	$\leftrightarrow$	ModalModel
Orthogonality	$\leftrightarrow$	Concurrent Actors (inherent)
Hierarchy	$\leftrightarrow$	Compound Actors, state refinements
Compound events		

Ptolemy Simulation Engine

- Mapping SyncCharts to Ptolemy:

Mealy machine	$\leftrightarrow$	ModalModel
Orthogonality	$\leftrightarrow$	Concurrent Actors (inherent)
Hierarchy	$\leftrightarrow$	Compound Actors, state refinements
Compound events	$\leftrightarrow$	Expression language

- Interesting:

Ptolemy Simulation Engine

- Mapping SyncCharts to Ptolemy:

Mealy machine	$\leftrightarrow$	ModalModel
Orthogonality	$\leftrightarrow$	Concurrent Actors (inherent)
Hierarchy	$\leftrightarrow$	Compound Actors, state refinements
Compound events	$\leftrightarrow$	Expression language

- Interesting:
  - Implicit broadcast vs. explicit signal representation

Ptolemy Simulation Engine

- Mapping SyncCharts to Ptolemy:

Mealy machine	$\leftrightarrow$	ModalModel
Orthogonality	$\leftrightarrow$	Concurrent Actors (inherent)
Hierarchy	$\leftrightarrow$	Compound Actors, state refinements
Compound events	$\leftrightarrow$	Expression language

- Interesting:
  - Implicit broadcast vs. explicit signal representation
  - Signal coherence (must/cannot analysis)

Ptolemy Simulation Engine

- Mapping SyncCharts to Ptolemy:

Mealy machine	$\leftrightarrow$	ModalModel
Orthogonality	$\leftrightarrow$	Concurrent Actors (inherent)
Hierarchy	$\leftrightarrow$	Compound Actors, state refinements
Compound events	$\leftrightarrow$	Expression language

- Interesting:
  - Implicit broadcast vs. explicit signal representation
  - Signal coherence (must/cannot analysis)
  - Transition priorities

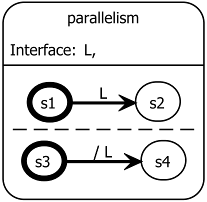
Ptolemy Simulation Engine

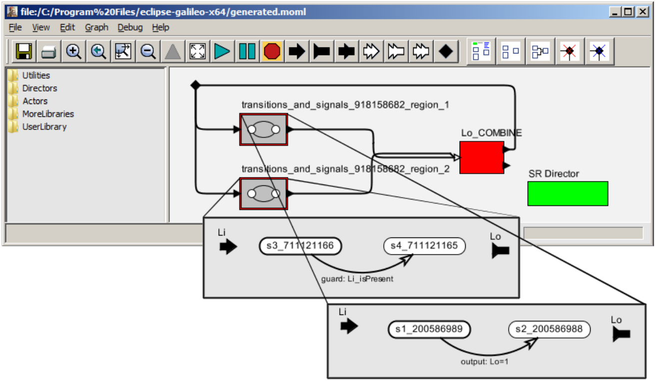
- Mapping SyncCharts to Ptolemy:

Mealy machine	$\leftrightarrow$	ModalModel
Orthogonality	$\leftrightarrow$	Concurrent Actors (inherent)
Hierarchy	$\leftrightarrow$	Compound Actors, state refinements
Compound events	$\leftrightarrow$	Expression language

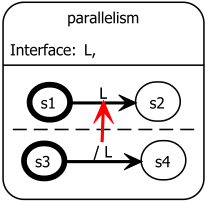
- Interesting:
  - Implicit broadcast vs. explicit signal representation
  - Signal coherence (must/cannot analysis)
  - Transition priorities
  - Normal termination

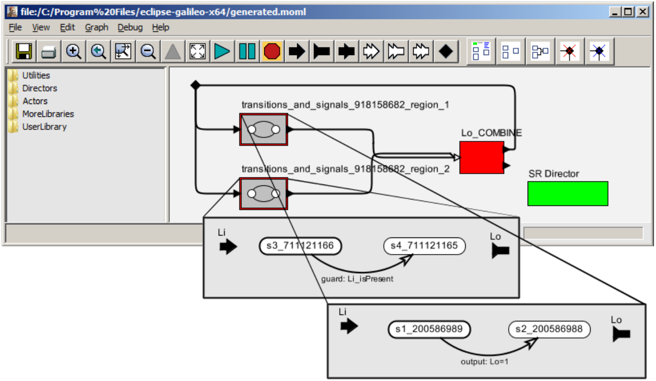
Transformation Example: Parallelism and Signals



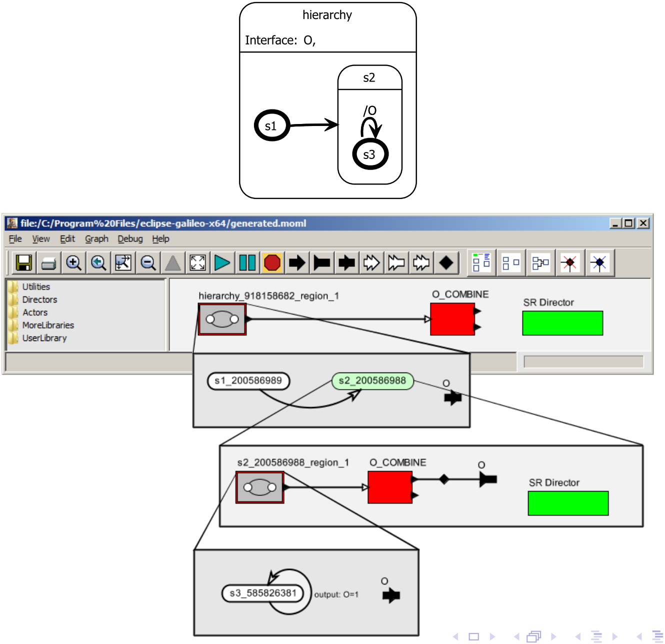


Transformation Example: Parallelism and Signals

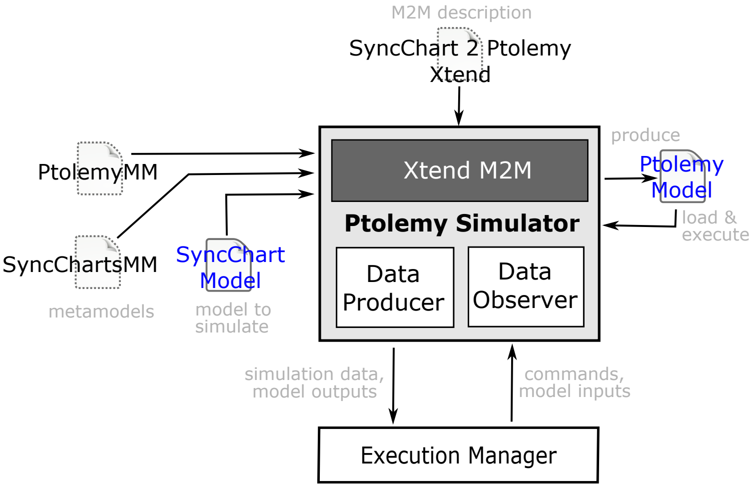




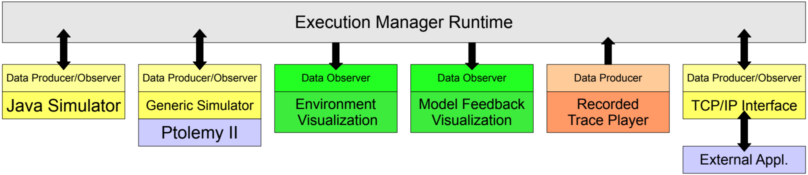
Transformation Example: Hierarchy



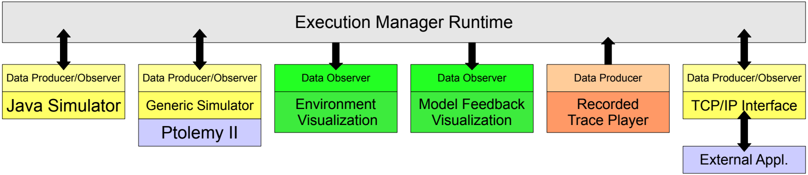
Schematic Overview

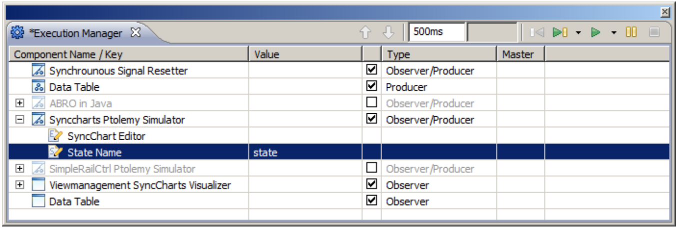


Architecture and User Interface



Architecture and User Interface





KIELER KlePto Simulation Demo

LIVE DEMO

Summary

- KIELER

Summary

- KIELER
- Ptolemy

Summary

- KIELER
- Ptolemy
- KIELER leveraging Ptolemy
  - KlePto concept

Summary

- KIELER
- Ptolemy
- KIELER leveraging Ptolemy
  - KlePto concept
  - Construct runnable Ptolemy models for EMF based models (Xtend)

Summary

- KIELER
- Ptolemy
- KIELER leveraging Ptolemy
  - KlePto concept
  - Construct runnable Ptolemy models for EMF based models (Xtend)
  - Ptolemy integration in Eclipse

Summary

- KIELER
- Ptolemy
- KIELER leveraging Ptolemy
  - KlePto concept
  - Construct runnable Ptolemy models for EMF based models (Xtend)
  - Ptolemy integration in Eclipse
  - Infrastructure for interactive model execution

Summary

- KIELER
- Ptolemy
- KIELER leveraging Ptolemy
  - KlePto concept
  - Construct runnable Ptolemy models for EMF based models (Xtend)
  - Ptolemy integration in Eclipse
  - Infrastructure for interactive model execution
  - Also: Visualization, stepwise transformation, model checking, online debugging, regression tests, validation, ...

To Go Further

ANDRÉ, C.
Computing SyncCharts reactions.
In *SLAP 2003: Synchronous Languages, Applications and Programming, A Satellite Workshop of ECRST 2003* (2004), vol. 88, pp. 3 – 19.

MOTIKA, C., FUHRMANN, H., AND VON HANXLEDEN, R.
Semantics and execution of domain specific models.
In *2nd Workshop Methodische Entwicklung von Modellierungswerkzeugen (MEMWe 2010) at conference INFORMATIK 2010* (Leipzig, Germany, Sept. 2010), GI-Edition – Lecture Notes in Informatics (LNI), Bonner Köllen Verlag.

UC BERKELEY, EECS DEPT.
Ptolemy webpage.
http://ptolemy.eecs.berkeley.edu/.

UNI KIEL, REAL-TIME AND EMBEDDED SYSTEMS GROUP.
KIELER webpage.
http://www.informatik.uni-kiel.de/en/rtsys/kieler/.

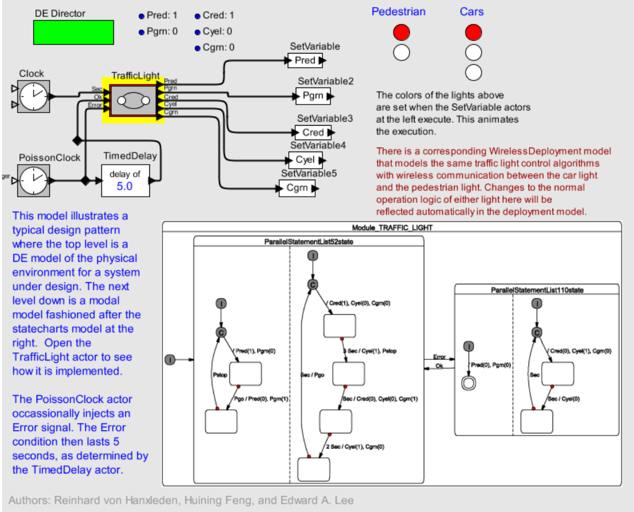
Thank you for your attention and participation!

Any questions or suggestions?

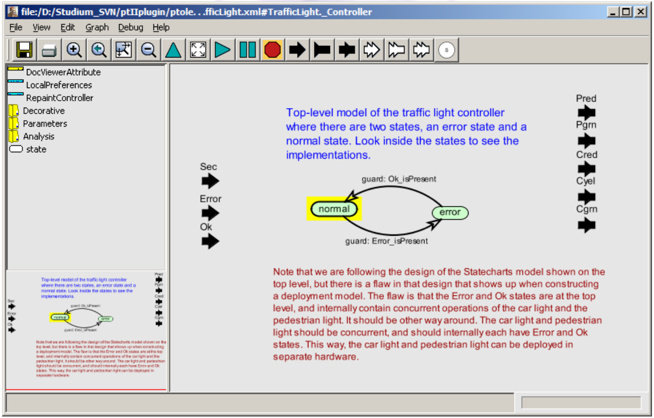
Sample DataComponent

```java
public class DataComponent extends JSONObjectDataComponent
                           implements IJSONObjectDataComponent {
   boolean doneI;

   public void initialize() {
      doneI = false;
   }

   public boolean isObserver() {return true;}
   public boolean isProducer() {return true;}

   public JSONObject step(JSONObject jSONObject)
                  throws KiemExecutionException {
      JSONObject returnObj = new JSONObject();
      if (!doneI && jSONObject.has("I")
         && (JSONSignalValues.isPresent(jSONObject.get("I")))) {
         //change state to doneI when signal I is present
         doneI = true;
         //output signal O
         returnObj.accumulate("O", JSONSignalValues.newValue(true));
      }
      return returnObj;
   }
}
```

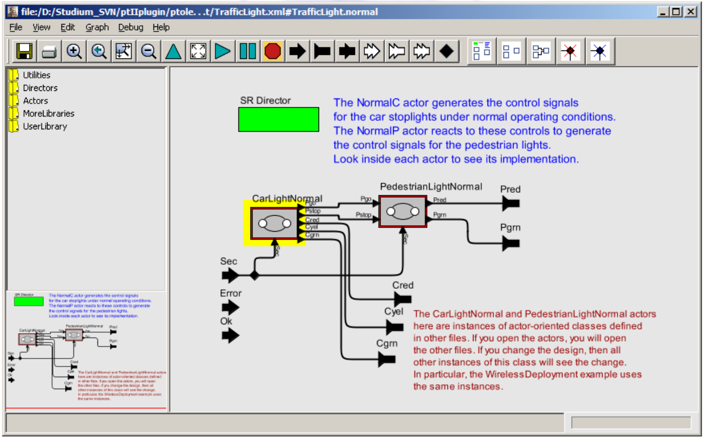
Synchronous/Reactive Modeling Example: TrafficLight



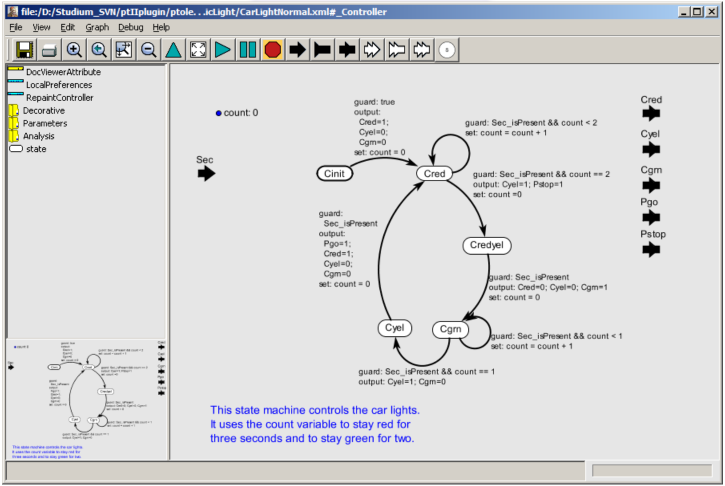
Synchronous/Reactive Modeling Example: TrafficLight



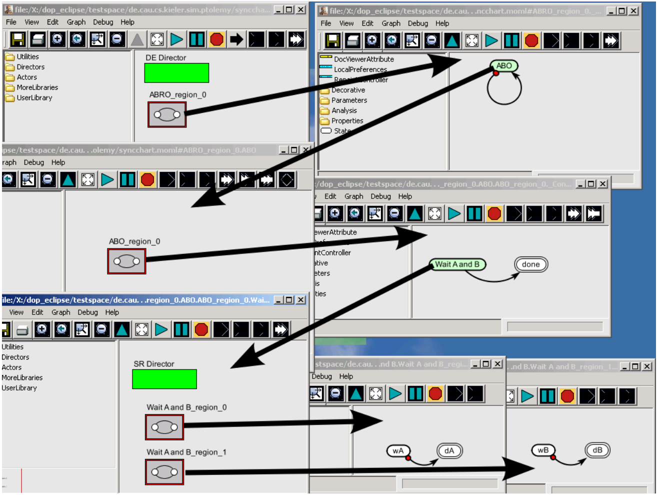
Synchronous/Reactive Modeling Example: TrafficLight



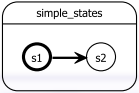
Synchronous/Reactive Modeling Example: TrafficLight

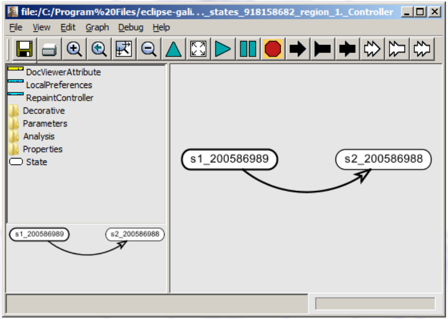


M2M Transformation Results

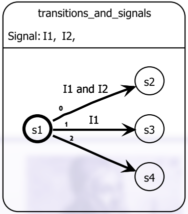


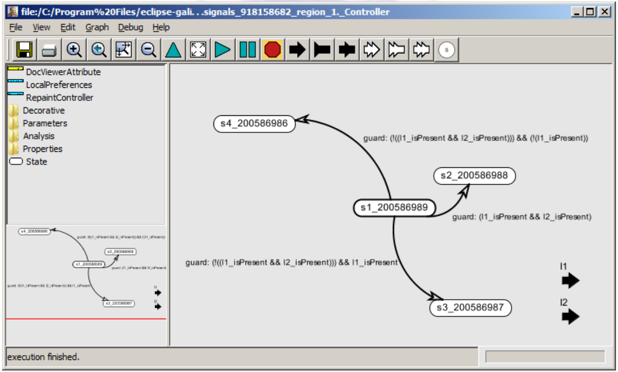
Transformation 1: Simple States



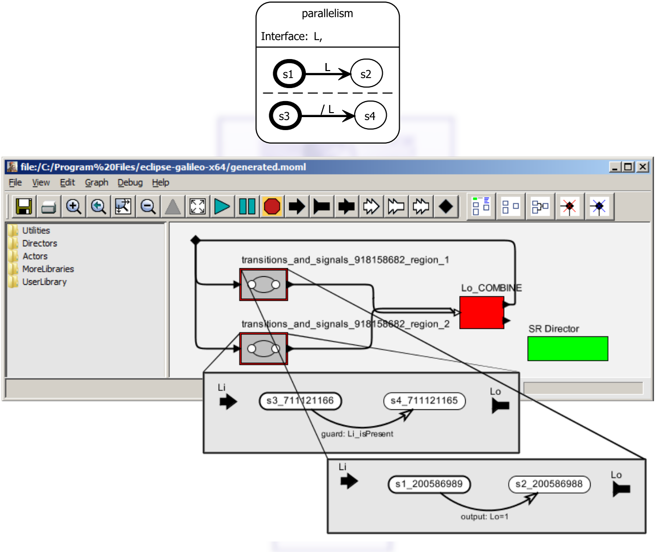


Transformation 2: Transitions (Priorities)

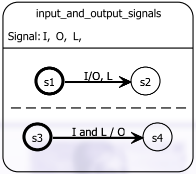


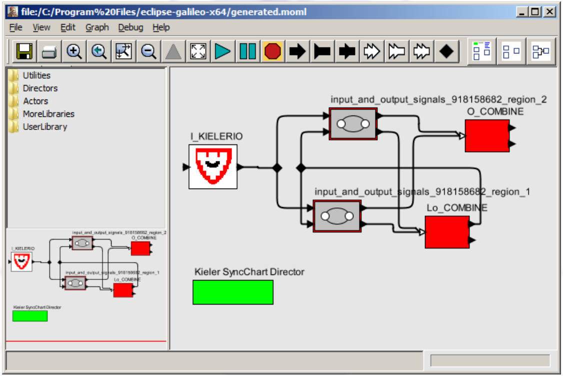


Transformation 3: Parallelism and Signals

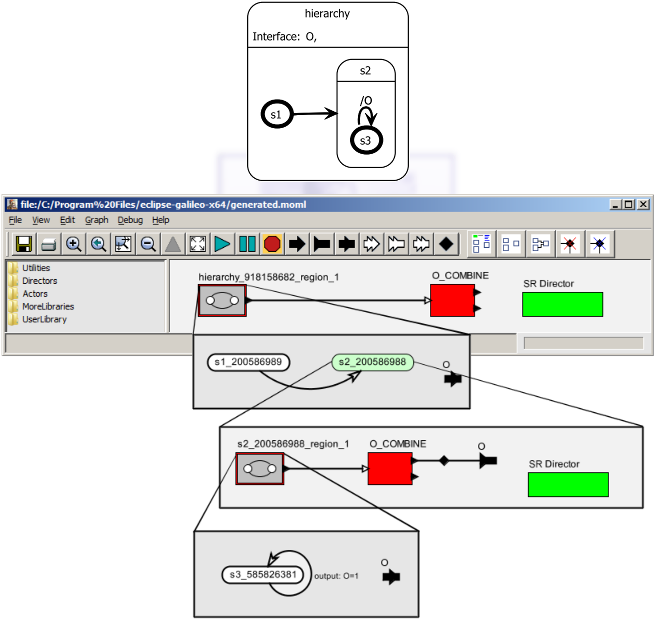


Transformation 4: Inputs and Outputs





Transformation 5: Hierarchy



Java: Load And Execute Ptolemy Model

```java
public class TestLoadPtolemyModel {
    public static void main(String[] args) {
        URI momlFile = URI.createFileURI(new File("kielerio.moml").getAbsolutePath());
        MoMLParser parser = new MoMLParser();
        NamedObj ptolemyModel = null;
        //load & parse model
        ptolemyModel = parser.parse(null, new URL(momlFile.toString()));
        //execute model
        CompositeActor actor = ((CompositeActor) ptolemyModel);
        //create a manager
        Manager manager = actor.getManager();
        if (manager == null) {
             manager = new Manager(actor.workspace(), "kieler manager");
             actor.setManager(manager);
        }
        // run the model
        if (manager != null) {
              List<Actor> children = actor.getChildren();
              manager.initialize();
              for (int i = 0; i < 100; i++) {
                    manager.iterate();
                    // LISTEN FOR OUPUT TO KIELER HERE //
              }
              manager.wrapup();
}    }   }
```

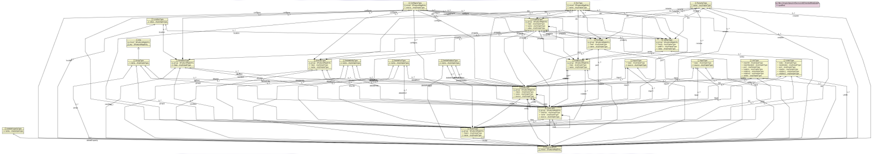
Java: Insert User Data w/ KielerIO

```
public class TestLoadPtolemyModel {
     public static void main(String[] args) {
    //load & parse model
    //execute model
    //create a manager

    //insert user data
    Iterator<Object> childrenIterator = actor.containedObjectsIterator();
    while( childrenIterator.hasNext() ){
          Object child = childrenIterator.next();
          //search for KielerIO ports
          if(child instanceof KielerIO){
                      KielerIO kielerIO = (KielerIO)child;
                      System.out.println(kielerIO.getSignalName());
                      kielerIO.setValue(2);
                      kielerIO.setPresent(true);
                      kielerIO.setPermanent(true);
          }
    }

    // run the model
     }
}
```

Java: KielerIO Ptolemy Actor

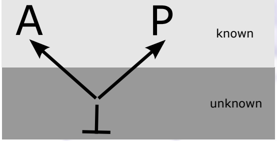
```
public class KielerIO extends TypedAtomicActor {
    public Parameter value;
    public TypedIOPort signal;

    public void setValue(int value) {
        this.value.setExpression(value+"");
    }

    public void fire() throws IllegalActionException {
        if (trigger.getWidth() > 0) {
         if (trigger.hasToken(0)) {
            trigger.get(0); }}

        if (present.getValueAsString().equals("true")) {
         int tokenValue = Integer.valueOf(value.getValueAsString());
         signal.send(0, new IntToken(tokenValue));
            if (permanent.getValueAsString().equals("false")) {
               this.setPresent(false);
            }
        }
        super.fire();
    }
}
```

Moml.ecore - Meta Model (EMF)



Determine Signal Assignment

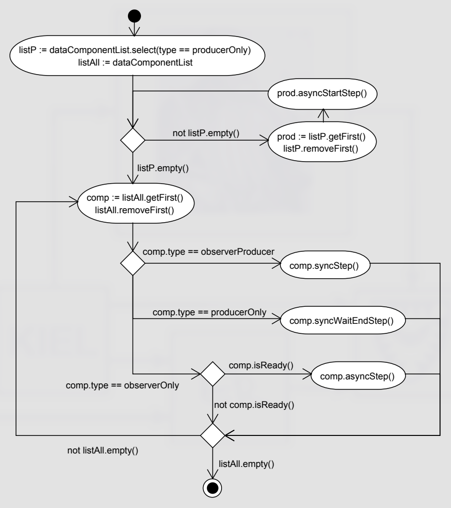
- Signal Coherence Law
  - A Signal can be either present or absent within a tick but not both at the same time.
- Ensure by lattice and ternary logic



- Use extended fixed point iteration:
  1. Start with unknown local and output signals
  2. Determine which signals must be emitted (set to present)
     $\rightarrow$ SR director fixed point iteration (send token)
  3. Determine which signals cannot be emitted (set to absent)
     $\rightarrow$ FSM director (send clear)

Linear Scheduler

Scheduling



listP := dataComponentList.select(type == producerOnly)
listAll := dataComponentList
prod.asyncStartStep()
not listP.empty()
prod := listP.getFirst()
listP.removeFirst()
listP.empty()
comp := listAll.getFirst()
listAll.removeFirst()
comp.type == observerProducer
comp.syncStep()
comp.type == producerOnly
comp.syncWaitEndStep()
comp.isReady()
comp.asyncStep()
comp.type == observerOnly
not comp.isReady()
not listAll.empty()
listAll.empty()

Ptolemy Meta Model (EMF)