Synchronous Languages—Lecture 20

Prof. Dr. Reinhard von Hanxleden

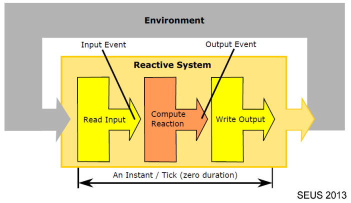
Christian-Albrechts Universität Kiel
Department of Computer Science
Real-Time Systems and Embedded Systems Group

13 January 2013

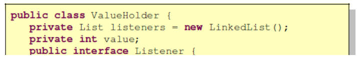
Last compiled: January 31, 2014, 8:56 hrs

SCCharts - Sequentially Constructive Statecharts for Safety-Critical Applications

Reactive Embedded Systems



SEUS 2013

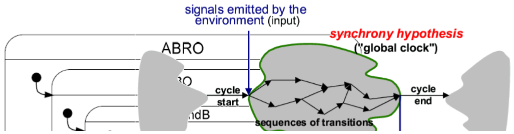


```
public class ValueHolder {
   private List listeners = new LinkedList();
   private int value;
   public interface Listener {
```

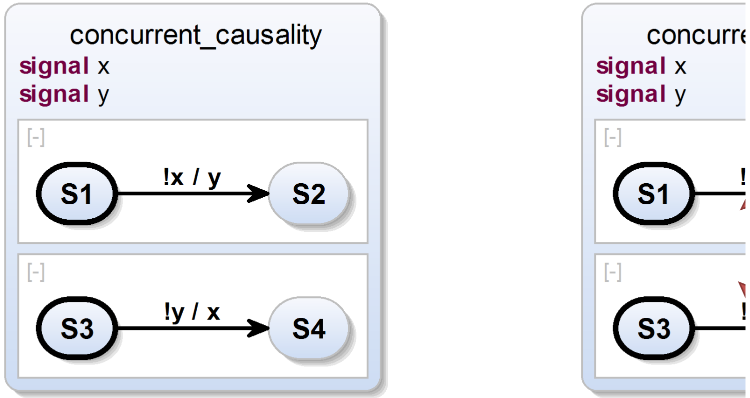
- Embedded systems react to inputs with computed outputs
- Typically state based computations
- Computations often exploit concurrency → Threads
- Threads are problematic ...

SyncCharts

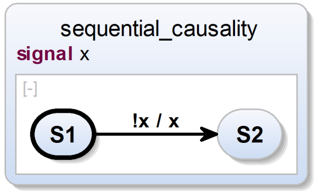
- Statechart dialect for specifying deterministic & robust concurrency
- SyncCharts:
  - Hierarchy, Concurrency, Broadcast
  - Synchrony Hypothesis
    1. Discrete ticks
    2. Computations: Zero time



Causality in SyncCharts



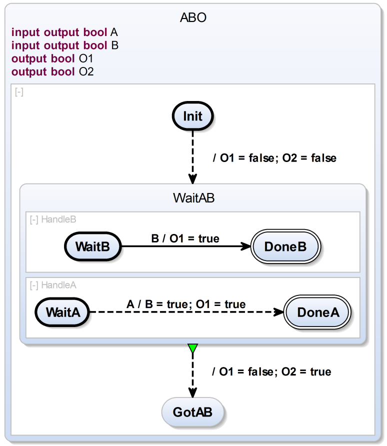
Causality in SyncCharts (cont'd)



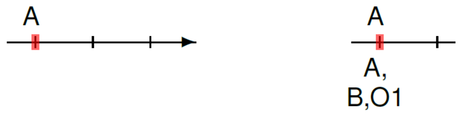
```
if (!done) {
   ...
  done = true;
}
```

- Rejected by SyncCharts compiler
- *Signal Coherence Rule*
- May seem awkward from SyncCharts perspective, but common paradigm
- Deterministic sequential execution possible using *Sequentially Constructive MoC* → **Sequentially Constructive Charts (SCCharts)**

SCCharts Overview



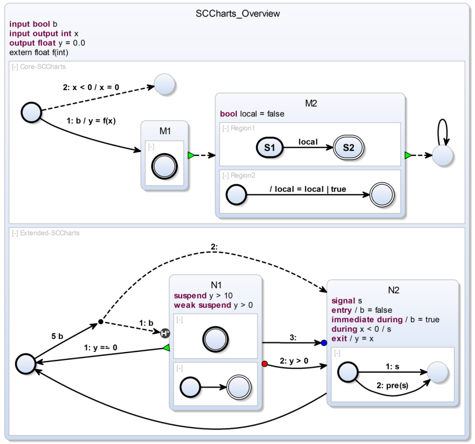
- SCCharts ≙ SyncCharts syntax + Seqentially Constructive semantics
- *Hello World* of Sequential Constructiveness: **ABO**
  - Variables instead of signals
  - Behavior (briefly)
    1. Initialize
    2. Concurrently wait for inputs *A* or *B* to become *true*
    3. Once *A* and *B* are true after the initial tick, take *Termination*
    4. Sequentially set *O1* and *O2*

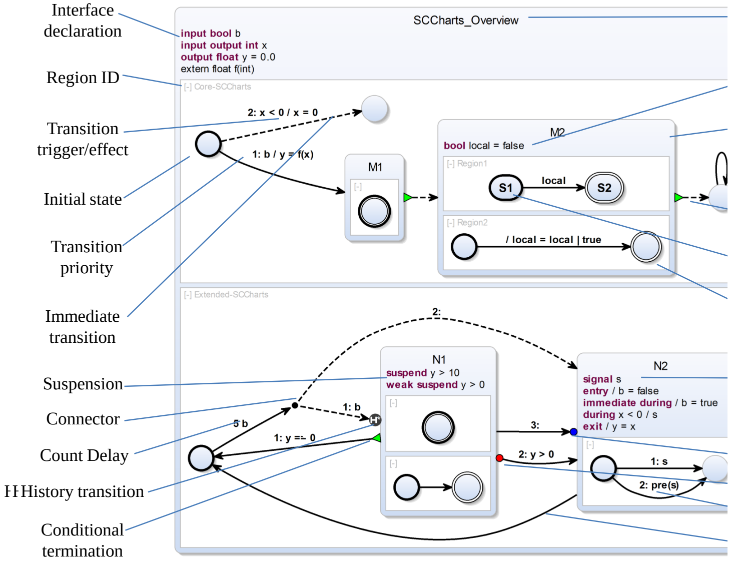


Overview

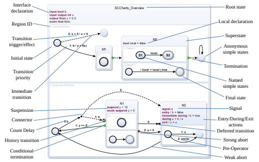
- SCCharts Overview
- Extended SCCharts → Core SCCharts
- Normalizing Core SCCharts
- Implementation in KIELER

SCCharts - Features



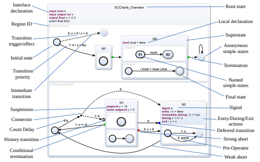


Motivation (Cont'd)



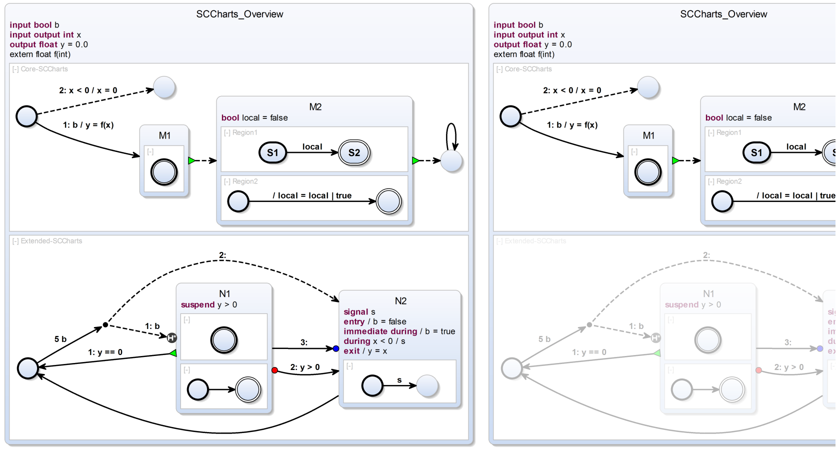
- **Advantages**:
  - Minimal base language (Core SCCharts)
    + advanced features (Extended SCCharts)
    - Similar to Esterel Kernel Statements & Statement Expansion
  - Advanced features are *syntactic sugar*
  - Extensible
  - Compilation (ongoing research)
    - Modular & extensible
    - Less complex
    - Possibly less efficient

Motivation



- **Observation I**: Numerous features
  - ☺ Compactness / readability of models
  - ☹ Steeper learning curve
  - ☹ Direct compilation & verification more complex
- **Observation II**: Various features can be expressed by other ones
- **Consequence**: ⇒ Define extended features by means of base features

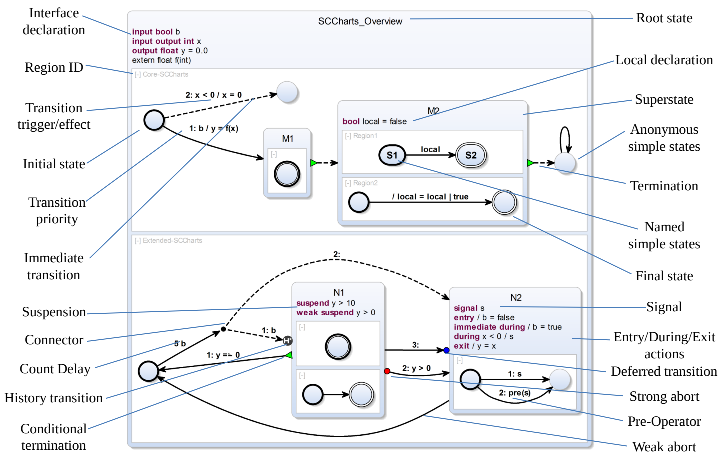
SCCharts - Core & Extended Features



Overview

- ▶ SCCharts Overview
- ▶ Extended SCCharts → Core SCCharts
- ▶ Normalizing Core SCCharts
- ▶ Implementation in KIELER

SCCharts - Core Transformations Examples

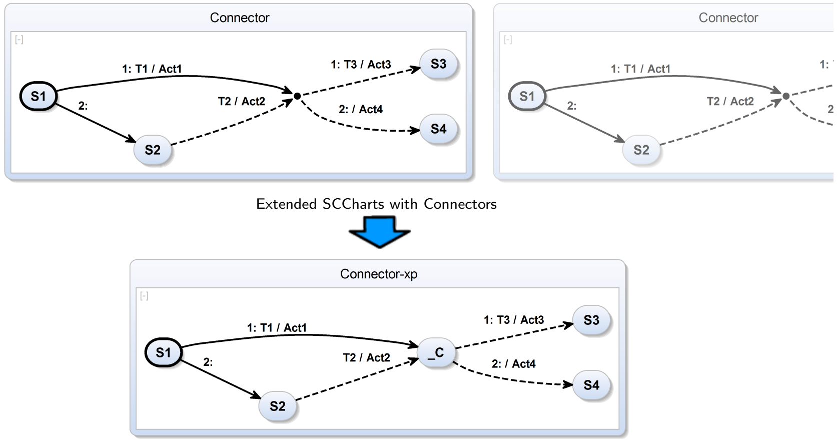


Interface declaration
Region ID
Transition trigger/effect
Initial state
Transition priority
Immediate transition
Suspension
Connector
Count Delay
History transition
Conditional termination

Root state
Local declaration
Superstate
Anonymous simple states
Termination
Named simple states
Final state
Signal
Entry/During/Exit actions
Deferred transition
Strong abort
Pre-Operator
Weak abort

Interface declaration
Region ID
Transition trigger/effect
Initial state
Transition priority
Immediate transition
Suspension
Connector
Count Delay
History transition
Conditional termination

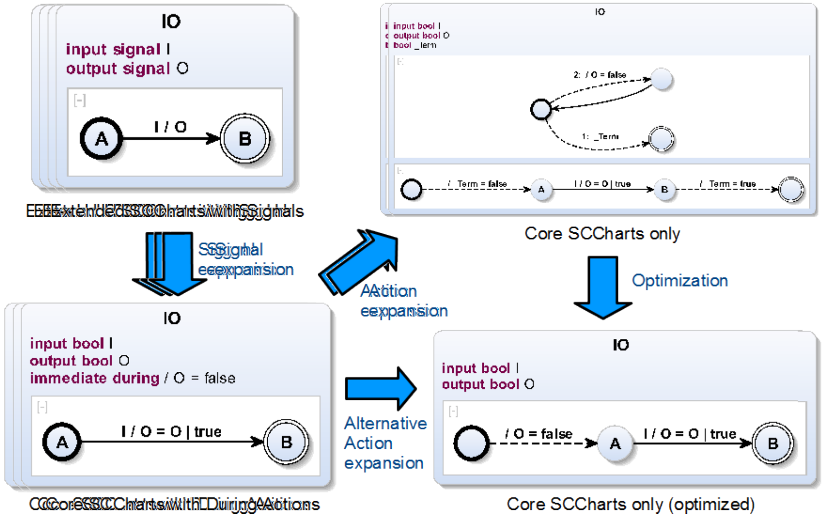
Transforming Connectors



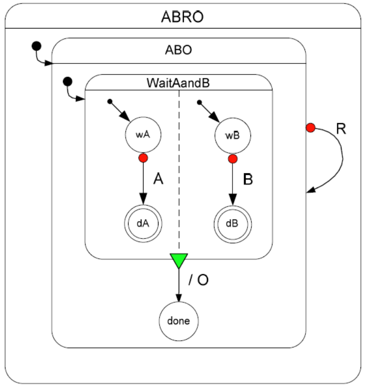
Extended SCCharts with Connectors

Core SCCharts without Connectors

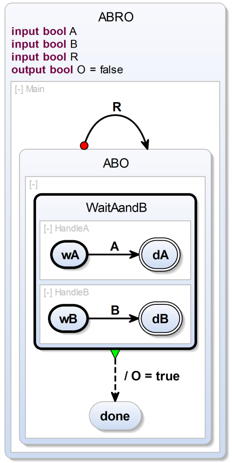
Transforming Signals



SyncChart and SCChart ABRO

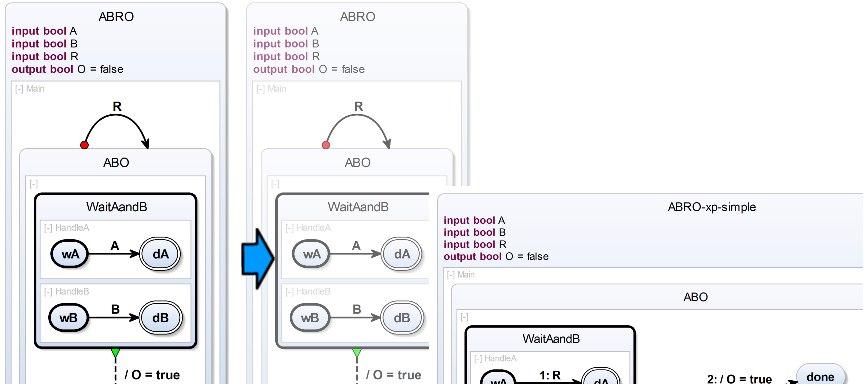


[Charles André, Semantics of SyncCharts, 2003]

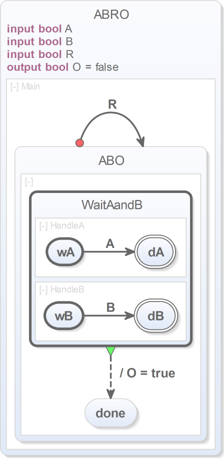


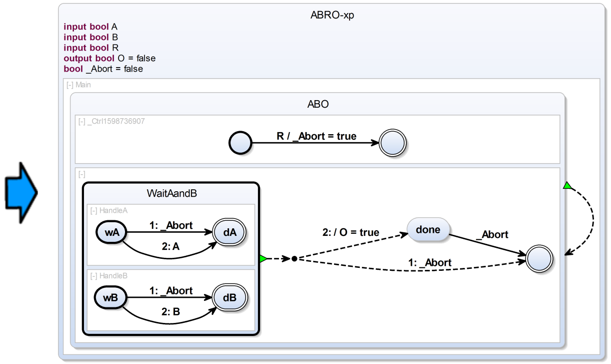
. ABRO SCChart

ABRO - Transforming Strong Aborts



ABRO - Transforming Strong Aborts (cont'd)

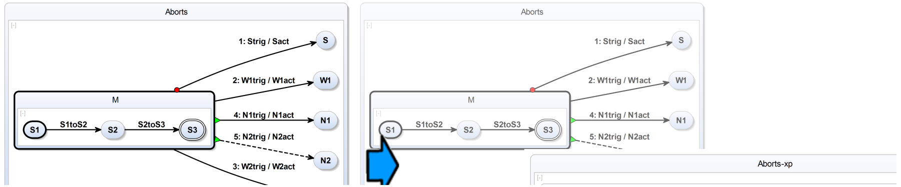




. Core SCChart without Strong Abort and WTO

ABRO SCChart with Strong Abort

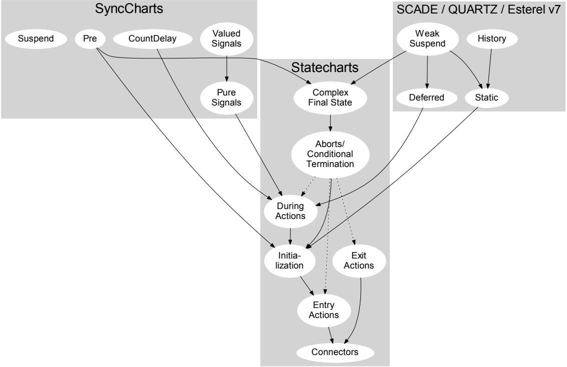
Transforming General Aborts



Overview

- SCCharts Overview
- Extended SCCharts → Core SCCharts
- Normalizing Core SCCharts
- Implementation in KIELER

Compilation

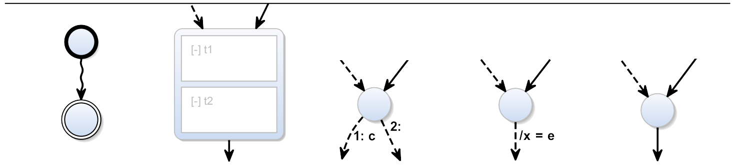


- Some core transformations will produce (use) other extended features
- → Order in which core transformations are applied is important
- → Dependencies (do not have any cycle, which would be forbidden)

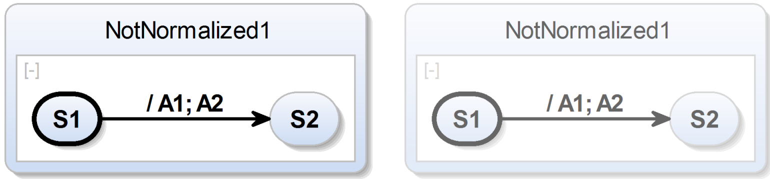
Normalization

- Further simplify compilation process for Core SCCharts
- Allowed patterns:

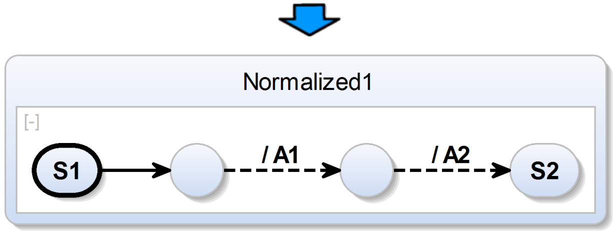
Region	Superstate	Trigger	Action	State
(connected states)	(parallel regions)	(conditionals)	(assignments)	(tick boundary)



Actions Normalization

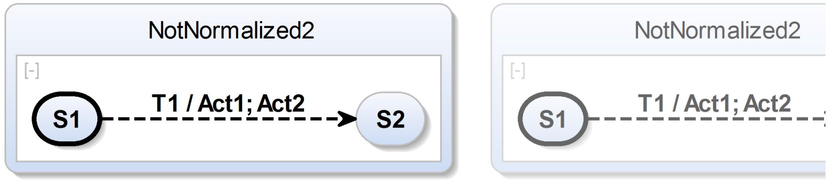


Core SCChart before normalization

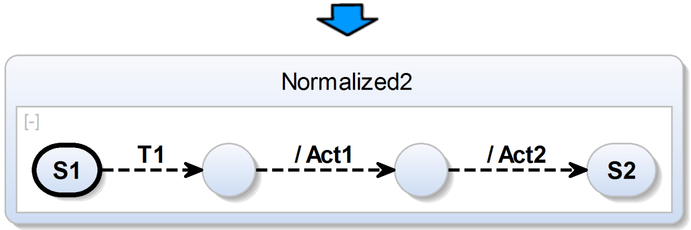


Core SCChart after normalization

Actions Normalization (cont'd)



Core SCChart before normalization

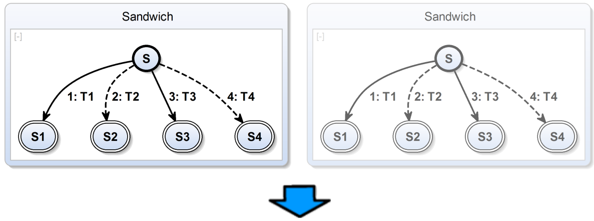


Core SCChart after normalization

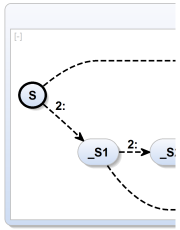
Actions Normalization Implementation Example

```
def void transformTriggerActions(Transition transition) {
  if (((transition.trigger != null || !transition.immediate)
      && !transition.actions.nullOrEmpty) || transition.actions.size > 1) {

    val targetState = transition.targetState
    val parentRegion = targetState.parentRegion
    val transitionOriginalTarget = transition.targetState

    var Transition lastTransition = transition

    for (action : transition.actions.immutableCopy) {

      val actionState = parentRegion.createState(targetState.id + action.id)
      actionState.setTypeConnector

      val actionTransition = createImmediateTransition.addAction(action)
      actionTransition.setSourceState(actionState)

      lastTransition.setTargetState(actionState)
      lastTransition = actionTransition
     }

     lastTransition.setTargetState(transitionOriginalTarget)
  }
}
```

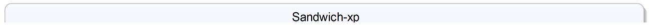
Trigger Normalization



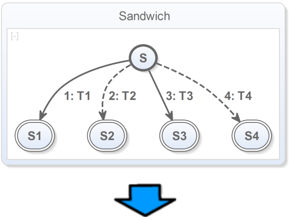
Core SCChart before normalization



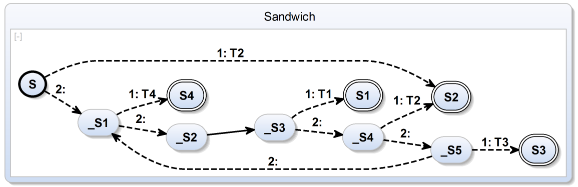
Core SCChart after normalization



Trigger Normalization (Cont'd)

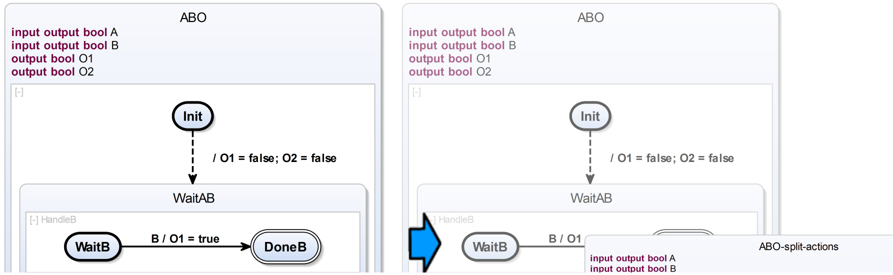


Core SCChart before normalization

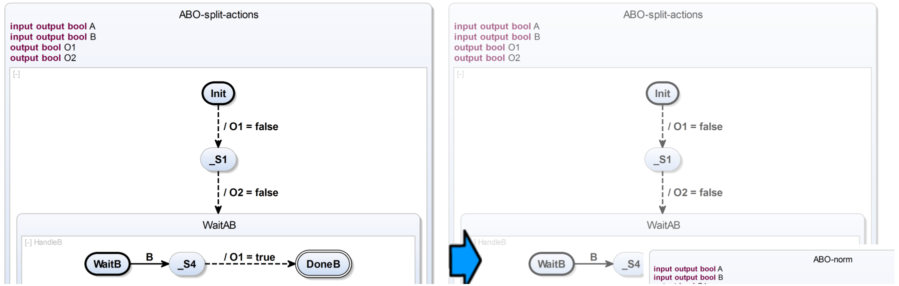


Core SCChart after optimized normalization

ABO - Normalization Example (Actions)



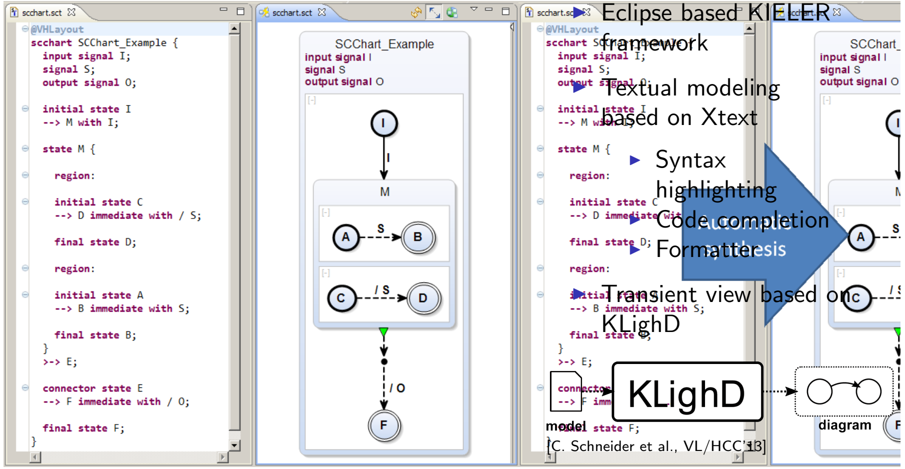
ABO - Normalization Example (Actions & Trigger)



Overview

- SCCharts Overview
- Extended SCCharts → Core SCCharts
- Normalizing Core SCCharts
- Implementation in KIELER

Textual Modeling with KLighD

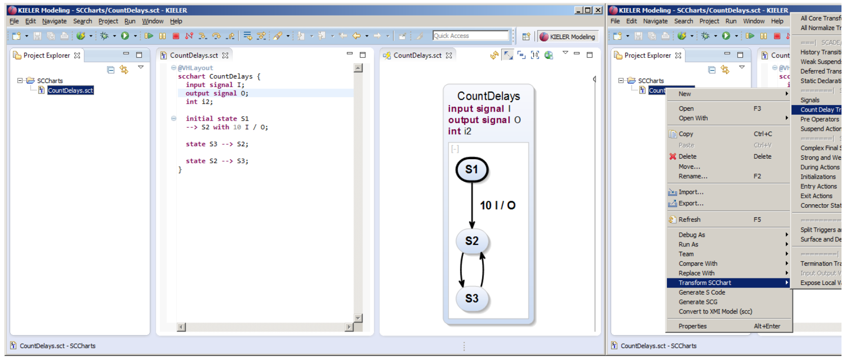


```
@VHLayout
scchart SCChart_Example {
  input signal I;
  signal S;
  output signal O;

  initial state I
  --> M with I;

  state M {

    region:

    initial state C
    --> D immediate with / S;

    final state D;

    region:

    initial state A
    --> B immediate with S;

    final state B;
  }
  >-> E;

  connector state E
  --> F immediate with / O;

  final state F;
}
```

- Eclipse based KIELER framework
- Textual modeling based on Xtext
  - Syntax highlighting
  - Code completion
  - Formatter
- Transient view based on KLighD

[C. Schneider et al., VL/HCC'13]

Transforming SCCharts with KIELER



CountDelays.sct

```
@VHLayout
scchart CountDelays {
  input signal I;
  output signal O;
  int i2;

  initial state S1
  --> S2 with 10 I / O;

  state S3 --> S2;

  state S2 --> S3;
}
```

- SCCharts context menu *Transform SCChart*
- Transformed model (`*.transformed.sct`) is opened and visualized
- Apply core transformations and normalization in one step (→ order!)

Conclusions

- SyncCharts **are** a great choice for specifying deterministic control-flow behavior. . .
- . . . but does not accept sequentiality
  `If (!done) { ... ; done = true;}`
- **SCCharts** extend SyncCharts w.r.t. semantics
  → Sequentially Constructive MoC
  - All valid SyncCharts interpreted as SCCharts **keep** their meaning
- **Core** SCCharts: Few basic features for simpler & more robust compilation
- **Extended** SCCharts: Syntactic sugar, readability, extensible
- **Normalized** SCCharts: Further ease compilation
  → Details in the next lecture :-)

To Go Further

- KIELER website: `http://rtsys.informatik.uni-kiel.de/kieler`
- C. André. *Semantics of SyncCharts*. Technical Report ISRN I3S/RR-2003-24-FR, I3S Laboratory, Sophia-Antipolis, France, April 2003.
- G. Berry. *The foundations of Esterel*. In G. Plotkin, C. Stirling, and M. Tofte, editors, Proof, Language, and Interaction: Essays in Honour of Robin Milner, pages 425-454, Cambridge, MA, USA, 2000.
- R. von Hanxleden, B. Duderstadt, C. Motika, S. Smyth, M. Mendler, J. Aguado, S. Mercer, and O. O'Brien. *SCCharts: Sequentially Constructive Statecharts for Safety-Critical Applications*. Technical Report 1311, Christian-Albrechts-Universitaet zu Kiel, Department of Computer Science, Dec 2013.
- R. von Hanxleden, M. Mendler, J. Aguado, B. Duderstadt, I. Fuhrmann, C. Motika, S. Mercer, and O. O'Brien. *Sequentially Constructive Concurrency - A conservative extension of the synchronous model of computation*. In Proc. Design, Automation and Test in Europe Conference (DATE'13), Grenoble, France, March 2013.