SCCharts in Motion
Interactive Model-Based Compilation for a Railway System

Christian Motika, Steven Smyth, and Reinhard von Hanxleden

Real-Time Systems and Embedded Systems Group
Department of Computer Science
Kiel University, Germany

SYNCHRON 2014
Aussois, 1 Dec. 2014
Reactive Embedded Systems

- Embedded systems often \textit{safety-critical}
- \textit{React} to inputs with computed outputs, \textit{state based} computations
- Computations often exploit \textit{concurrency}
  - Threads $\leadsto$ Non-Determinism
    $\rightarrow$ \textbf{Synchronous languages}: Lustre, Esterel, SCADE, SyncCharts
  - Sequentiality hard to model
    $\rightarrow$ \textbf{Sequentially Constructive Charts (SCCharts)}

\textbf{SCCharts well-suited for safety-critical systems}
Recall: Sequentially Constructive Charts – SCCharts

- André’s SyncCharts Syntax
- + Sequentially Constructive Semantics

1. Core features
2. Extended feat.

Model transformations: Extended \( \rightarrow \ldots \rightarrow \) Core
SCCharts for Safety-Critical Systems

- Language/semantics well-suited
- ... but that is not enough
  - **Compiler** must be reliable
    - (well structured, understandable, extensible, maintainable, ...)
  - **Modeling**: Toolchain must facilitate building reliable models
    - (abstraction mechanisms, support to understand language&models, simulations, optimizations, fine-tuning, ...)
  - **Practicability**: Challenge real-life examples!

→ That’s what this talk is about!
Part I
Compiler

Original Model → Intermediate Models → Fully transformed Model

Part II
Modeling

Textual Modeling → Select Transformation → Compiler

Part III
Practicability

Real-Life Evaluation

Single-Pass Language-Driven Incremental Compilation (SLIC)
Part I
Compiler

Part II
Modeling

Part III
Practicability

Single-Pass Language-Driven Incremental Compilation (SLIC)
AO – A Simple SCChart

- Initially set $O$ to $false$
- Wait for input $A$ to become $true$
- Once $A$ is $true$:
  - Take transition $WA \rightarrow DA$
  - Set $O$ to $true$

Extended feature: *Initialization*
AO – Applying Transformations \((\rightarrow \text{SYNCHRON '13: ABRO})\)
AO – Applying Initialization Transformation
AO – Applying Entry Transformation
Initialization Transformation Implementation

```java
def void transformInitialization(State state) {
    val initializedValuedObjects = state.valuedObjects.filter[initialValue != null]

    // Walk thru all initialized valuedObjects
    for (valuedObject : initializedValuedObjects) {

        // For every initialization: Create entry action
        val entryAction = state.createEntryAction

        // Copy the initial value to entry action assignment
        entryAction.addAssignment(valuedObject.assign(valuedObject.initialValue.copy))

        // Clear initialization (=> no initialization any more)
        valuedObject.setInitialValue(null)
    }
}
```
SCCharts Extended Feature Compilation

- Sequence *derived* from dependencies: produces & not-handled-by
- *Single-Pass Language-Driven Incremental Compilation (SLIC)*
Single-Pass Language-Driven Incremental Compilation [ISoLA’14]

- Single-pass sequence derived from dependencies produces & not-handled-by
- Requirement: No cycles
- Trade-off: More & simple ↔ less & complex
- SLIC Characteristic: Intermediate results = valid models

- Idea: Writing simple compiler, surprisingly also very practical
- Discussion: Usable also for other languages/compilers?
SCCharts Compilation - Advantages

- Validation
  - Each compilation step is *simple* → *Understandable*
  - Each transformation can be *inspected/tested* separately
  - Intermediate results are *valid models* → *Well structured*
  - New extended features can be *easily added* → *Extendable*
SCCharts Compiler Demo

SCCharts in Motion
Part I
Compiler

Part II
Modeling

Part III
Practicability

Single-Pass Language-Driven Incremental Compilation (SLIC)
Traditional Modeling & SW Synthesis User Story

1. User edits/draws model
2. Compiler parses model and synthesis code
3. User may inspect final artifacts

- Appropriate for advanced users
- But little guidance for beginners
- Compiler is black box
- Difficult for compiler writer
- Hardly allows to fine-tune and optimize the intermediate and/or resulting artifacts
- Hard to extend
SCCharts Modeling & Advantages

- View original and transformed model
  - Understanding language and models ✓
    - Appropriate for advanced users and beginners
    - Facilitates validation for compiler writer

- View effects of intermediate transformations
  - Optimization & fine-tuning ✓
SCCharts Interactive Modeling Details

Textual Modeling

Select Transformation

Modeled Diagram

(Intermediate) Transformed Diagram / Code

Compiler
SCCharts Modeling Demo
Part I
Compiler

Original Model → Intermediate Models → Fully transformed Model

Part II
Modeling

Textual Modeling → Select Transformation → Compiler

Intermediate Models → Fully transformed Model → Modeled Diagram

(Intermediate) Transformed Diagram / Code

Part III
Practicability

Real-Life Evaluation

1: !S/S
2: C

Christian Motika
SCCharts in Motion
Model Railway Project

http://rtsys.informatik.uni-kiel.de/confluence/display/SS14Railway

- SCCharts student project
  (7 participants)

- Project size
  - States: 1,628 (modeled)
    *States: 135,000* (expanded)
  - Transition: 2,219 (modeled)
    Transitions: 152,000 (expanded)
  - Concurrent Regions: 17,000 (expanded)
  - Generated C-Code: 650,000 lines
  - Compile time: 2-3 min, response time: <2ms

→ Medium-Size Example ✓
Project Results

- Improvements in efficiency, stability
  → Maintainability ✓
    - Compile Time (eAllContents)

- New extended features
  → Extendability ✓
    - Reference state expansion
    - Arrays
    - Hostcode function calls

- Results + Evaluation Survey → Technical Report
Survey – Tooling Evaluation

Maintainability

- Excellent
- Good
- Bad
- Worst

- Esterel
- SyncCharts
- SCCharts
- Ptolemy
- C
- Java
- Haskell

Debugging

- Excellent
- Good
- Bad
- Worst

- Esterel
- SyncCharts
- SCCharts
- Ptolemy
- C
- Java
- Haskell

SCCharts Quality of Modeling

- Professional
- Advanced
- Ok
- Hardly usable

- Code Generation
- Understanding Semantics
- User Interface
- Documentation
- Support

SCCharts Tooling Quality

- Professional
- Advanced
- Ok
- Hardly usable

- Project Start
- Project End
Railway Project Contributors

- Karsten Rathlev
- Carsten Sprung
- Caroline Butschek
- Alexander Schulz-Rosengarten
- Niclas Flieger
- Nis Börge Wechselberg
- Stanislaw Nasin
Conclusions

- Model based compilation \((SLIC) \rightarrow Reliable Compiler\)
- Interactive modeling \(\rightarrow Reliable Models\)
- Practicability \(\rightarrow Real-Life Models\)

www.SCCharts.com
To Go Further

http://www.sccharts.com


That’s all folks! — Any questions or suggestions?

SCCharts in Motion
Interactive Model-Based Compilation for a Railway System

Christian Motika, Steven Smyth, and Reinhard von Hanxleden

Real-Time Systems and Embedded Systems Group
Department of Computer Science
Kiel University, Germany

SYNCHRON 2014
Aussois, 1 Dec. 2014
ABRO – The “Hello World” of the Synchronous World

- Initially set $O$ to $false$
- Concurrently wait for inputs $A$ and $B$ to become $true$
- Once both are $true$, take termination immediately and set $O$ to $true$
- Reset behavior with $R$
- Strong preempt emission of $O$ when $R$ is $true$

Extended features: (a) Strong Abort transition, (b) Entry action
ABRO – Applying Transformations

1. initialization

2. entry

AO
Sequentially Constructive MoC

- Natural sequencing prescribes deterministic scheduling
  - `stmt1; stmt2`
  - `trigger/effect`

- Only concurrent data dependencies matter
  - Sequential data dependencies do not lead to rejection

- Deterministic concurrent scheduling:
  - Distinguish between relative and absolute writes
    - Absolute writes: `x = false`
    - Relative writes: `x = x | true`
    - Reads: `y = x`
    - (1) Absolute writes, (2) relative writes, (3) reads

- Sequentially Constructiveness fully subsumes
  *Berry Constructiveness*
Synchronous Program Classes

- Sequentially Constructive (S)
- Logically Correct (L)
- Pnueli-Shalev Constructive (P)
- Berry Constructive (B)
- Acyclic SC (A)

Static cycles
Dynamic scheduling
Sequences of values
Cycles of concurrent dependencies or concurrent writes
Out-of-order scheduling
Ineffective writes
Speculate on absence or presence
Speculate on absence

Christian Motika
SCCharts in Motion
SyncCharts

- **Statechart** dialect for specifying *deterministic* & robust *concurrency*

- SyncCharts:
  - Hierarchy, Concurrency, Broadcast
  - Synchrony Hypothesis
    1. Discrete ticks
    2. Computations: Zero time

[Charles André, Semantics of SyncCharts, 2003]

[Gerald Lüttgen, 2001]
Causality in SyncCharts

- 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)**
Causality in SyncCharts (cont’d)
Concurrency with Threads

Typical *observer pattern* implemented with Java Threads

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

    public interface Listener {
        public void valueChanged(int newValue);
    }

    public void addListener(Listener listener) {
        listeners.add(listener);
    }

    public void setValue(int newValue) {
        value = newValue;
        Iterator i = listeners.iterator();
        while (i.hasNext()) {
            ((Listener)i.next()).valueChanged(newValue);
        }
    }
}
```

E. A. Lee, The Problem with Threads, 2006

Not thread safe! E.g., multiple threads call `setValue()`.
SCCharts Compilation Overview

- Extended feature compilation (1): *SLIC approach*
- Also further compilation:
  - Normalization (2), mapping to SCG (3), sequentialization, ...
Harel Wristwatch – Citizen Quartz Multi-Alarm III
Railway Project – Dynamic Controller
Railway Installation
Track Layout
Project Overview - Controller Size

(taken from the final presentation of the railway project)
Tooling Evaluation - Compiler Performance

(taken from the final presentation of the railway project)