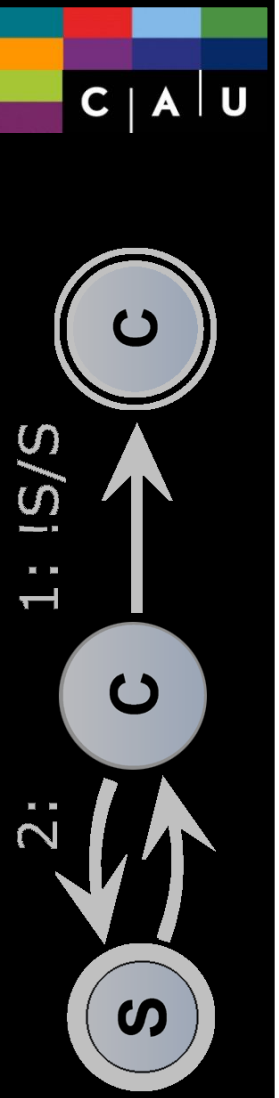
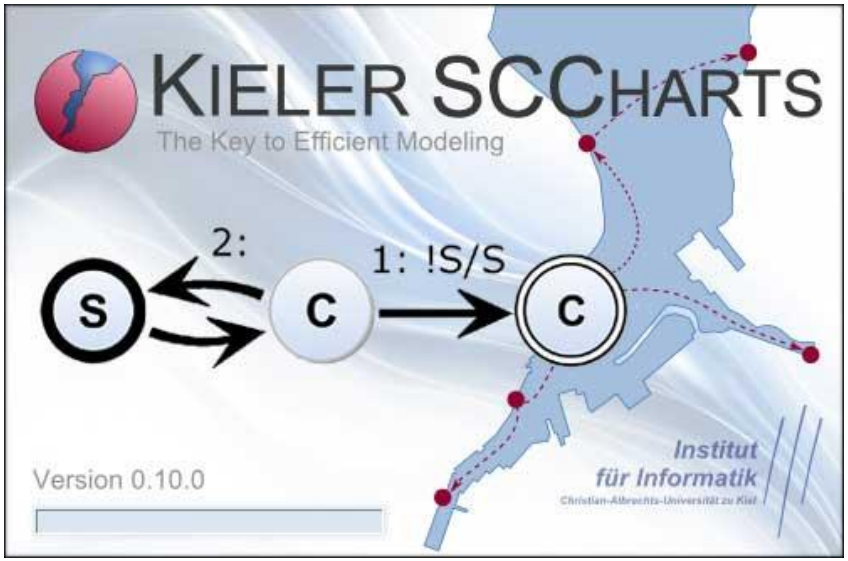
SCCharts
KIELER Compiler
Usage & Tests
Applications



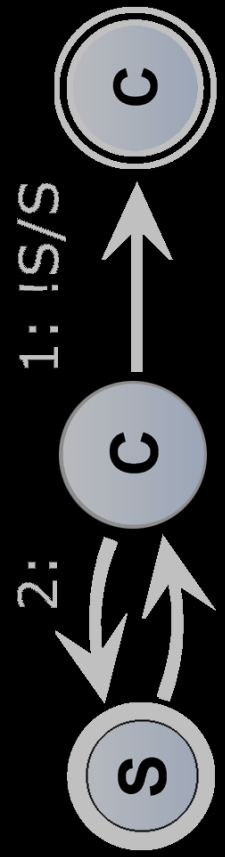
Updates on SCCharts

Christian Motika • Steven Smyth

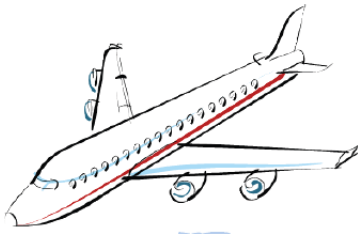


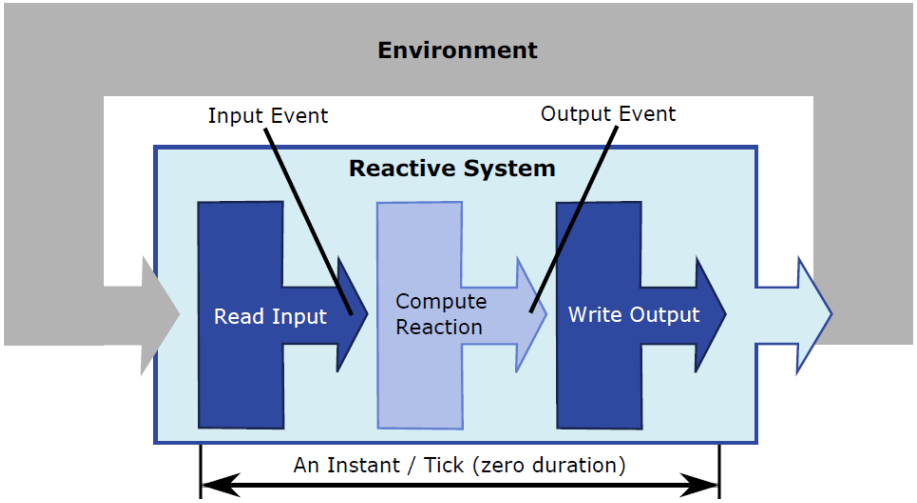
SYNCHRON 2015

04. DEC 2015, Kiel



Reactive System

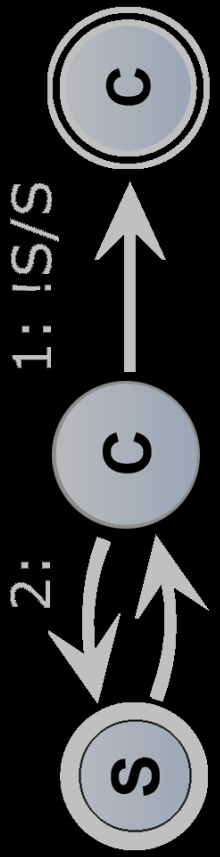




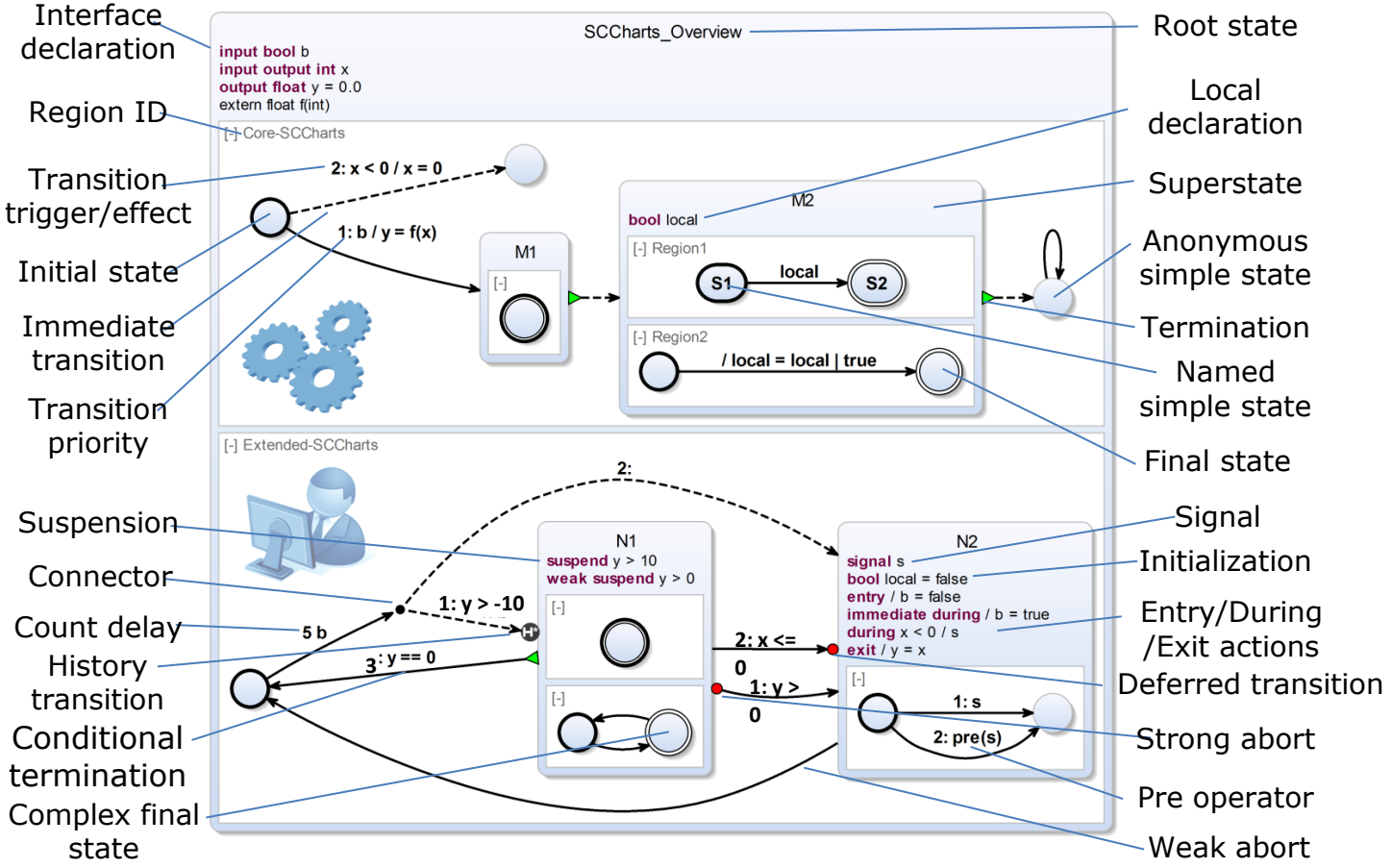
- Safety-critical systems
- State based reactions
- Concurrency

⇒ Synchronous Language

SCCharts = SyncCharts Syntax + Sequential Constructive Semantics

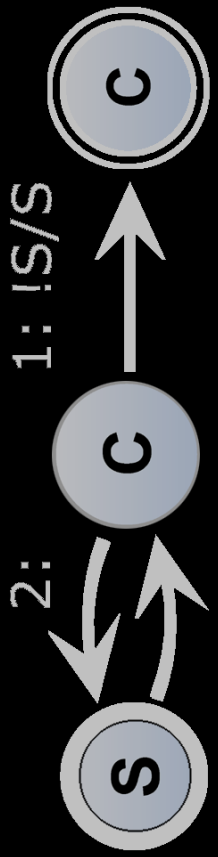


Recall SCCharts

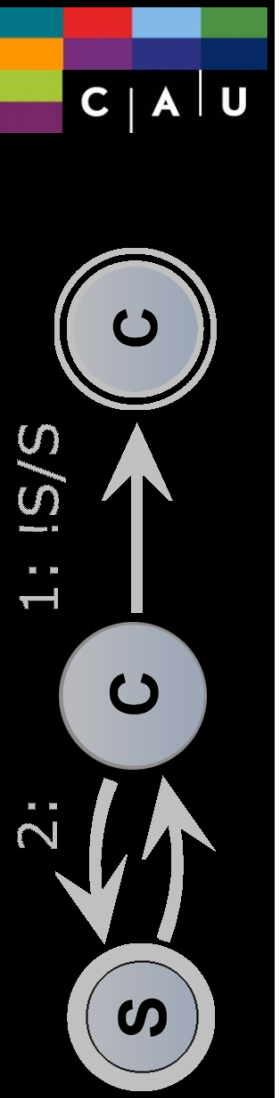


Core-SCCharts
Small set of simple features ease down stream compilation

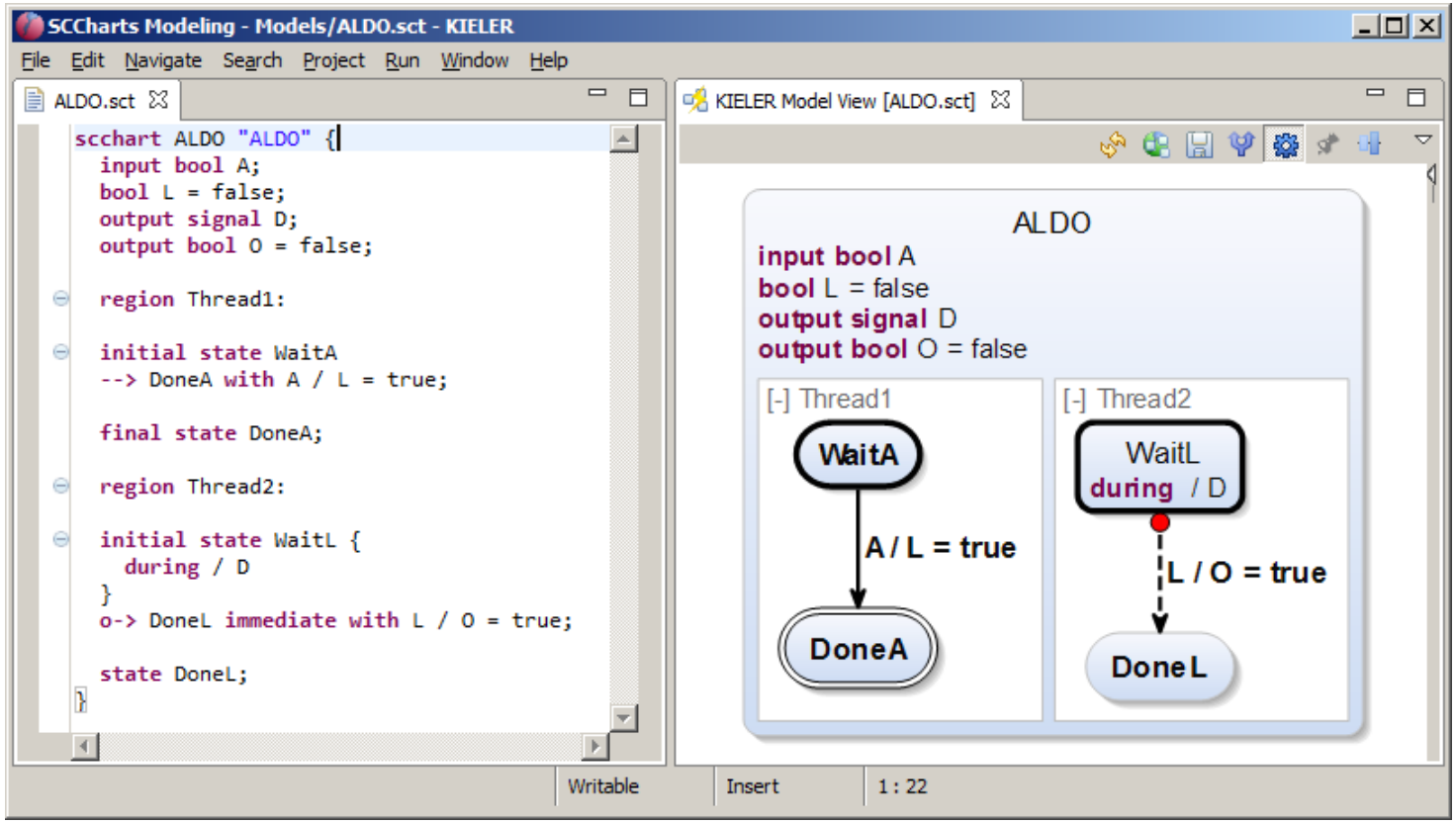
Extended-SCCharts
Rich set of advanced features ease modeling


C
1: IS/S
C
2:
S


ALDO
in Arcadia
"Look, it's flying!"
"Yes, I did it with a magic spell."

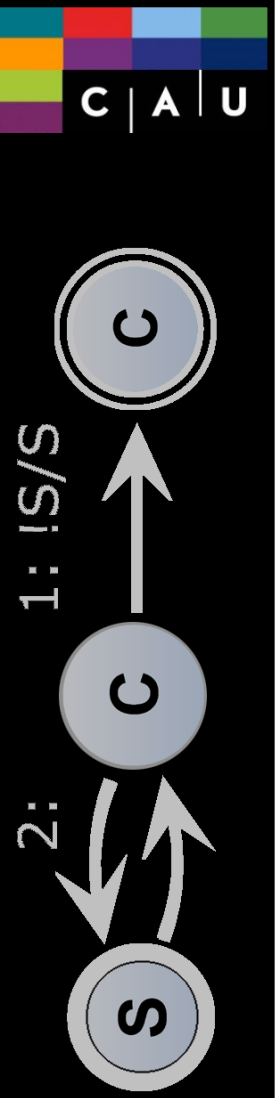


ALDO Example

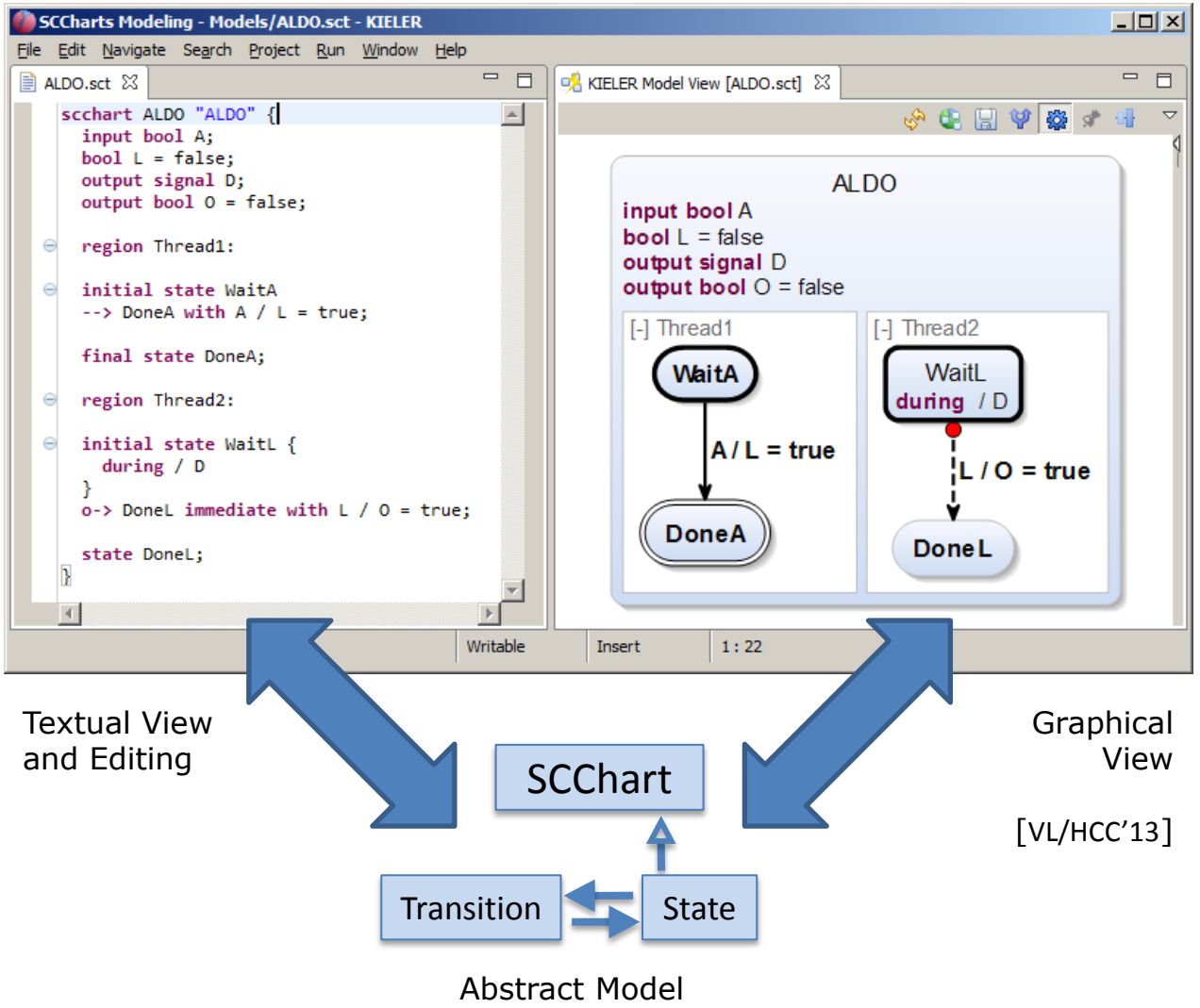


- Interface
- Local Variables
- Signals

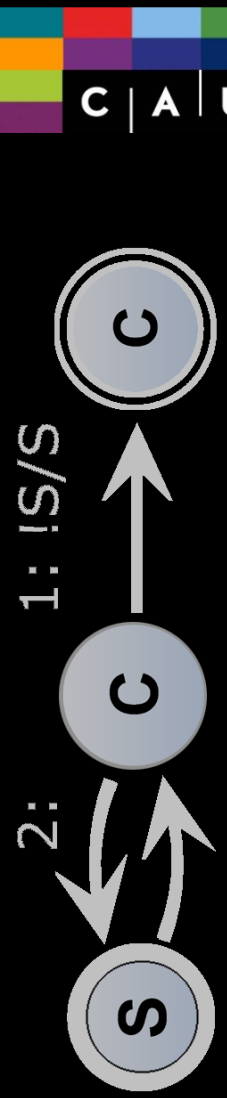
- Concurrency
- Instantaneous Communication
- Preemption



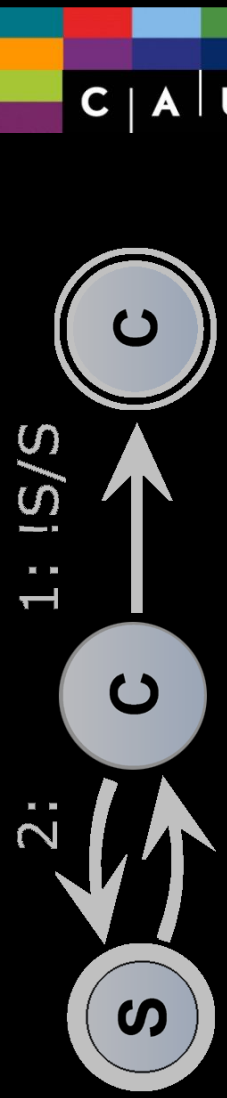
Modeling ALDO



[VL/HCC'13]

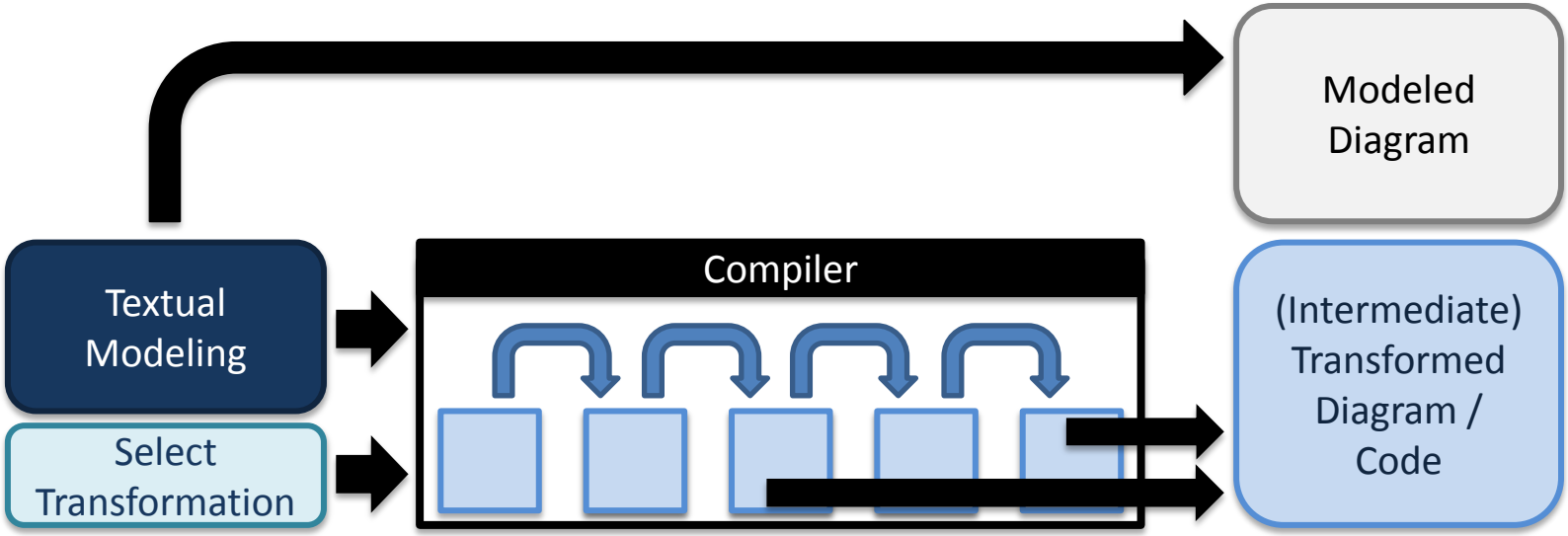


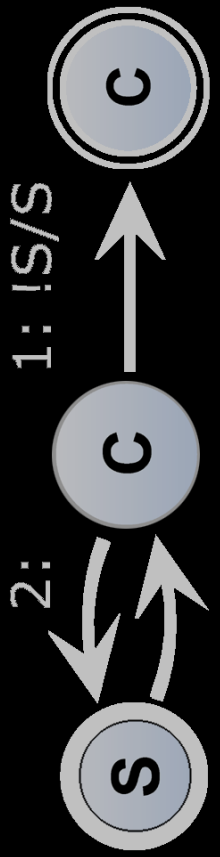
Modeling ALDO Demo



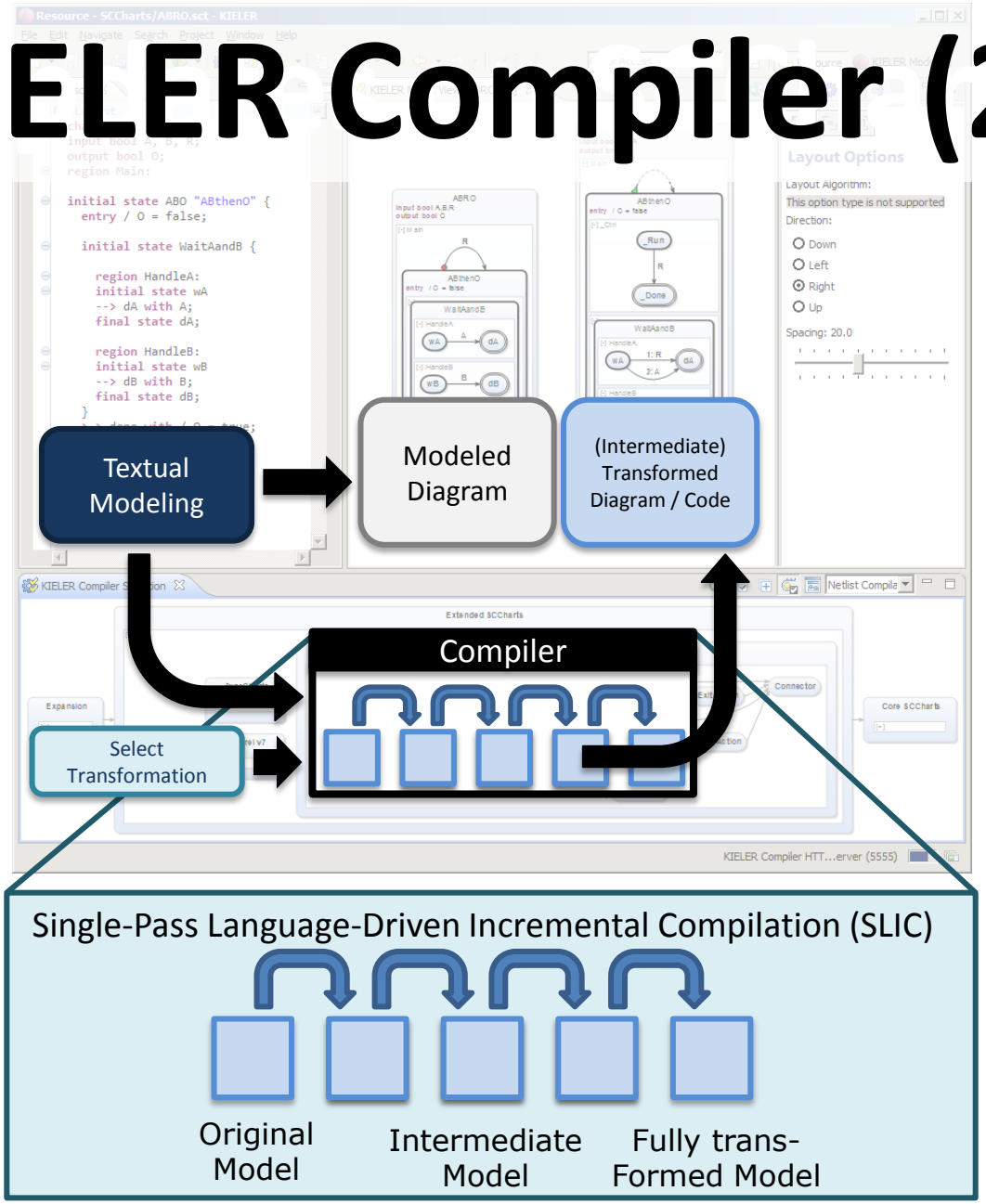
KIELER Compiler

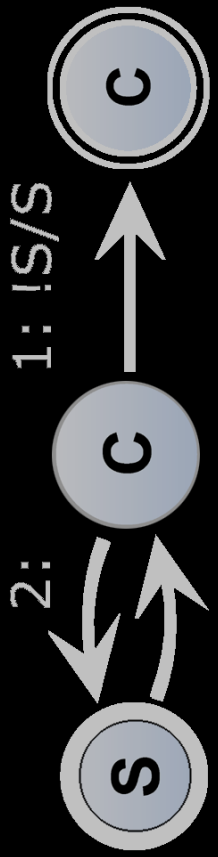
- Reliable Compiler + Reliable Models + Praticability [SYNCHRON'14]
- **S**ingle-Pass **L**anguage-Driven **I**ncremental **C**ompilation (SLIC) [ISOLA 14]
  - Interactive Model-Transformation-Based Compiler
  - Intemediate Results: White-Box Compiler





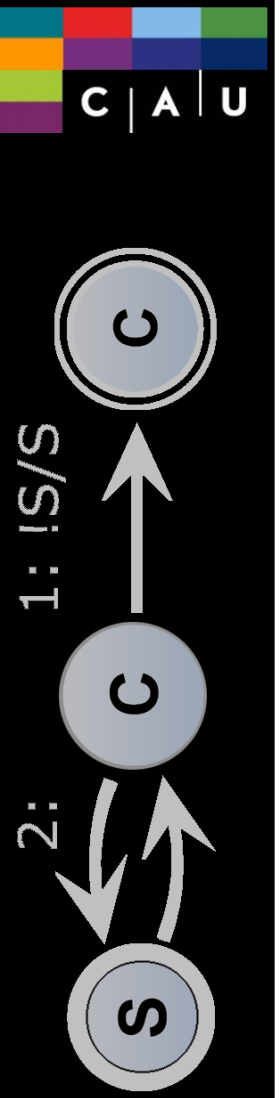
KIELER Compiler (2)



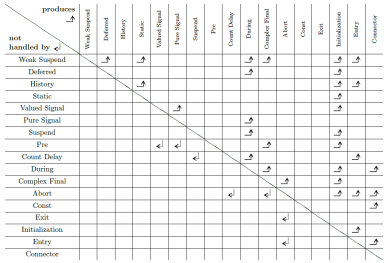


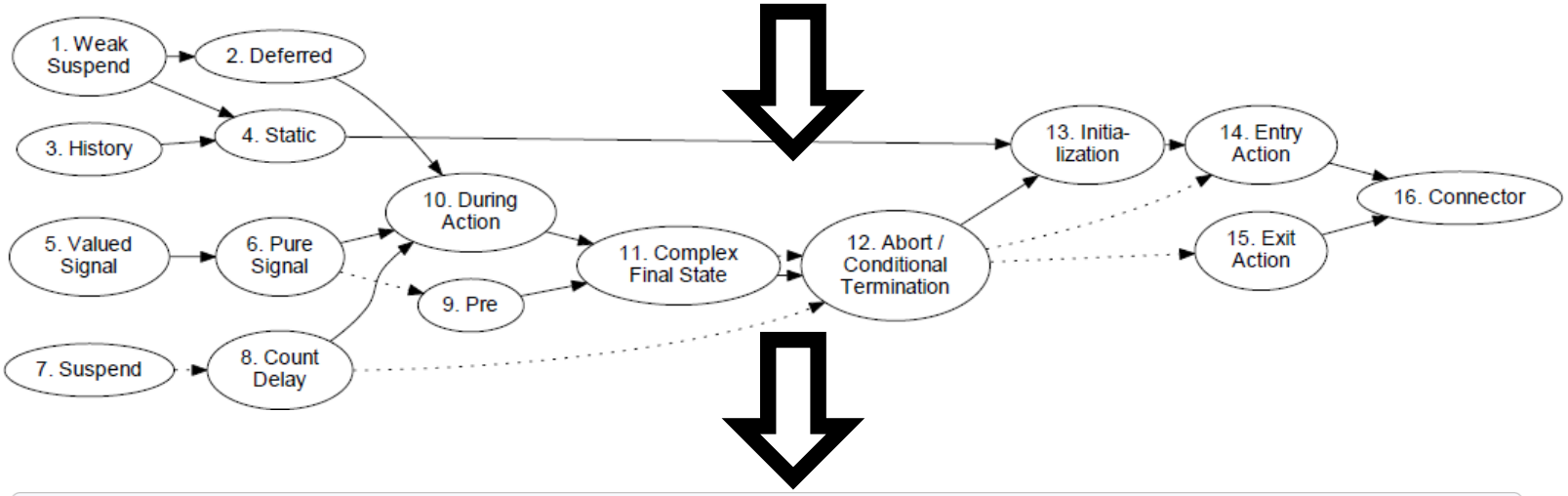
SLIC Order

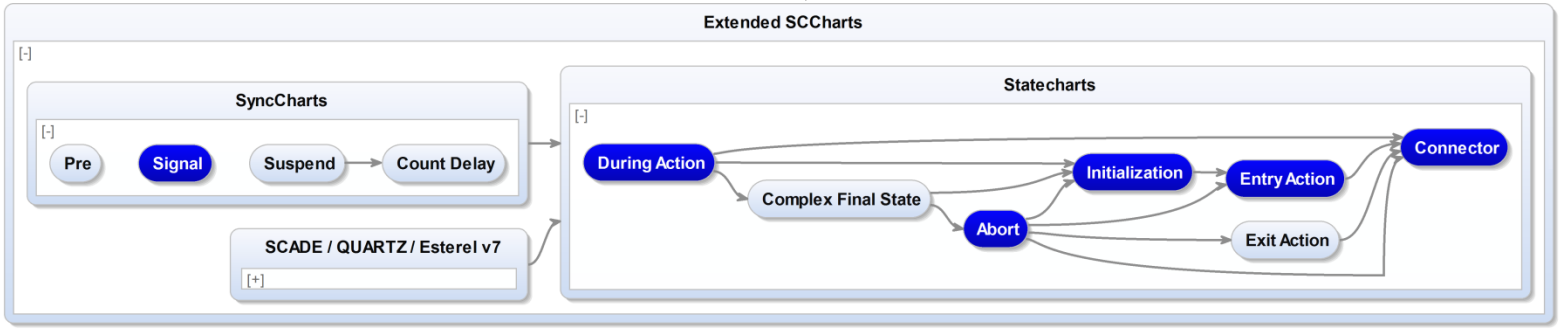
not handled by ↓ / produces →	Weak Suspend	Deferred	History	Static	Valued Signal	Pure Signal	Suspend	Pre	Count Delay	During	Complex Final	Abort	Const	Exit	Initialization	Entry	Connector
Weak Suspend		↳		↳						↳	↳				↳	↳	
Deferred										↳					↳		
History				↳											↳	↳	
Static															↳		
Valued Signal						↳									↳		
Pure Signal										↳							
Suspend										↳					↳		
Pre					←	←					↳				↳		
Count Delay							←			↳						↳	
During											↳				↳		↳
Complex Final												↳			↳		
Abort									←		←				↳	↳	↳
Const																	↳
Exit												←					
Initialization																↳	
Entry												←					↳
Connector																	


C | A | U
C
1: IS/S
C
2:
S

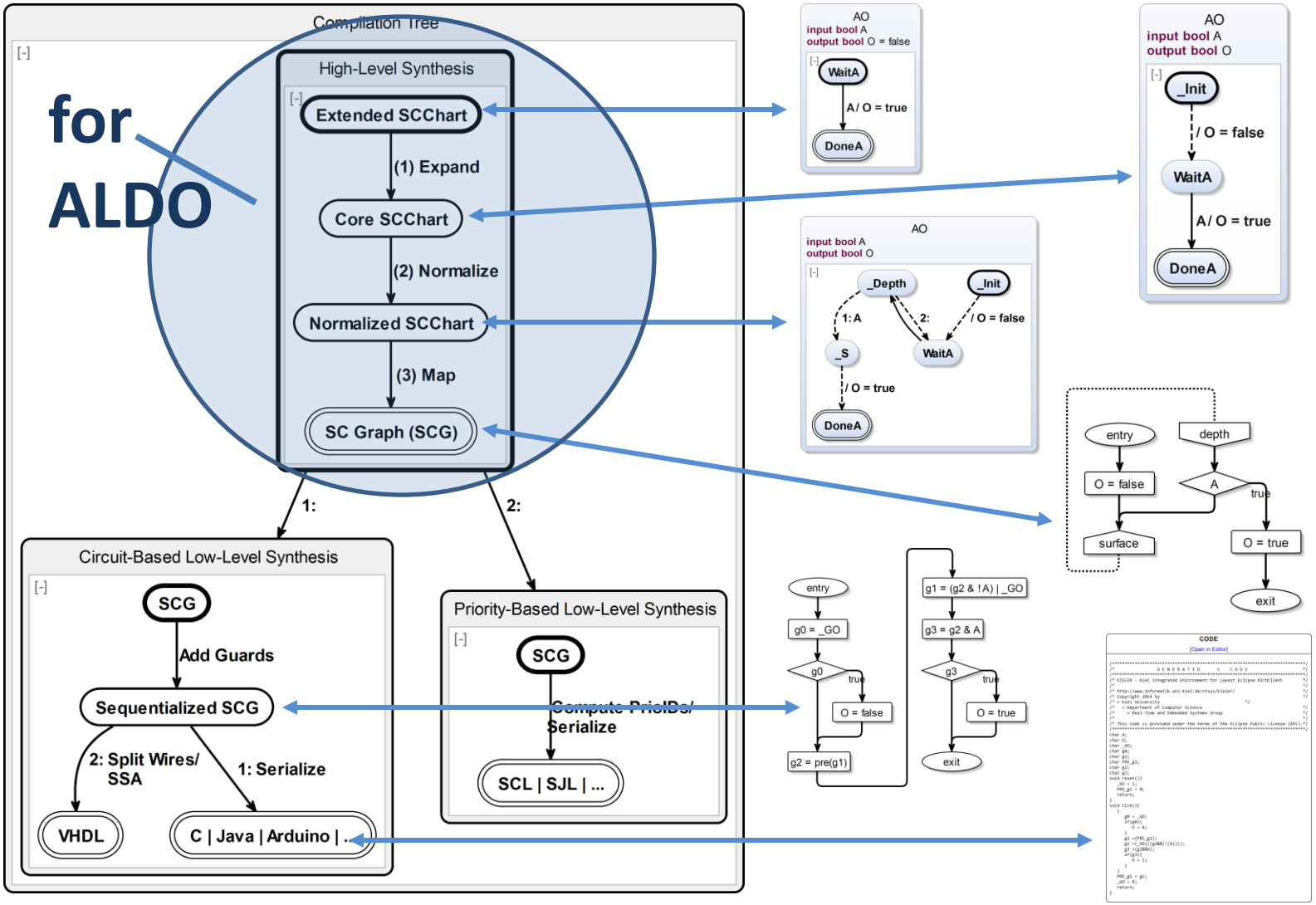
SLIC Order

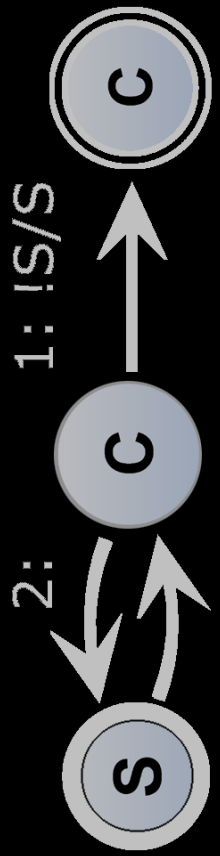

produces
not handled by
Weak Suspend
Deferred
History
Static
Valued Signal
Pure Signal
Suspend
Pre
Count Delay
During
Complex Final
Abort
Const
Exit
Initialization
Entry
Connector

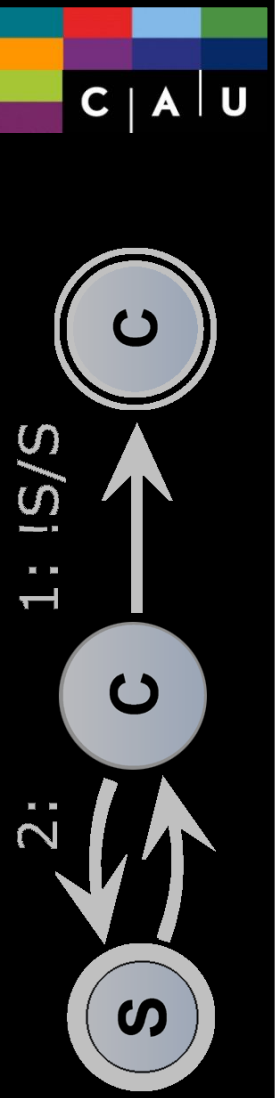

1. Weak Suspend
2. Deferred
3. History
4. Static
5. Valued Signal
6. Pure Signal
7. Suspend
8. Count Delay
9. Pre
10. During Action
11. Complex Final State
12. Abort / Conditional Termination
13. Initia-lization
14. Entry Action
15. Exit Action
16. Connector


Extended SCCharts
[-]
SyncCharts
[-]
Pre
Signal
Suspend
Count Delay
SCADE / QUARTZ / Esterel v7
[+]
Statecharts
[-]
During Action
Complex Final State
Abort
Initialization
Entry Action
Exit Action
Connector

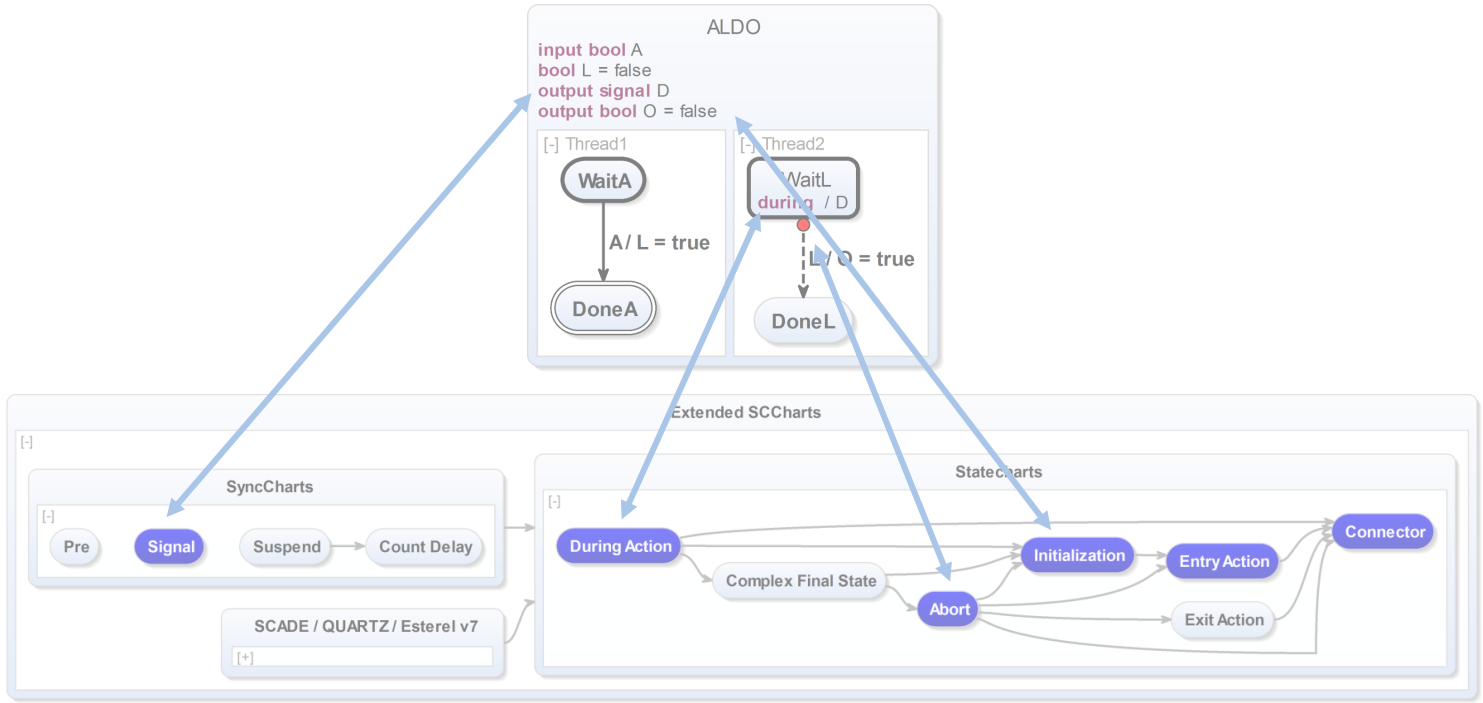
SCCharts Compilation

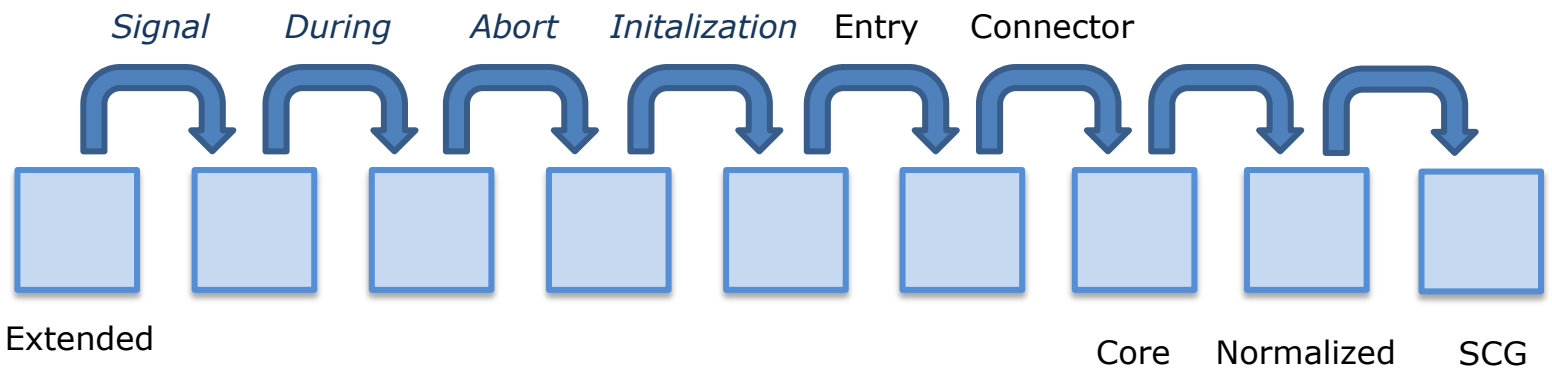


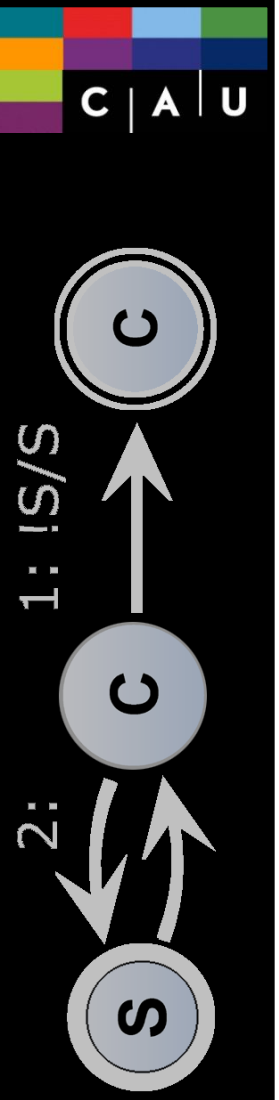




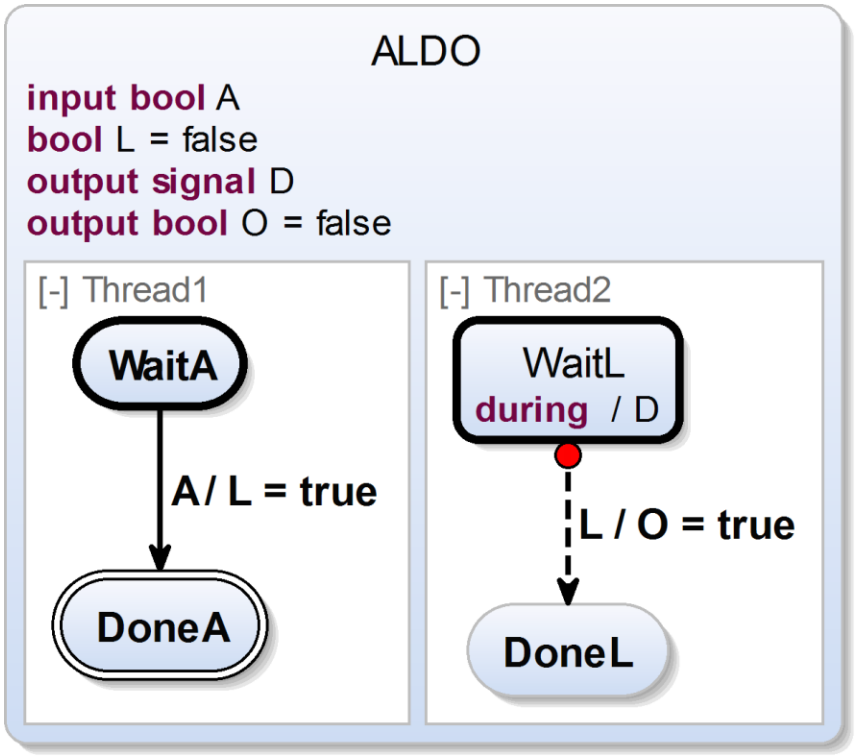
Compiling ALDO (1)

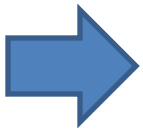




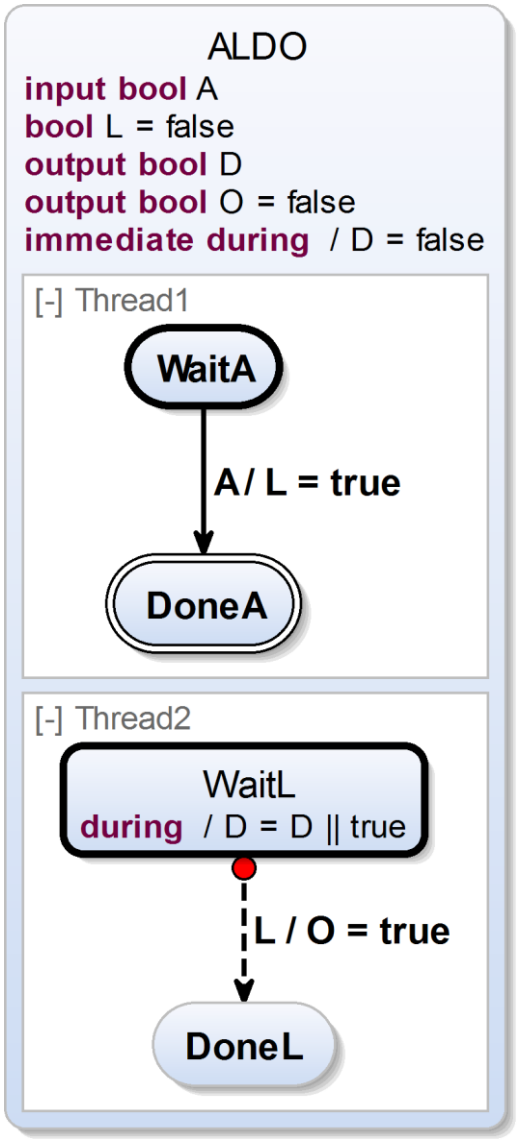


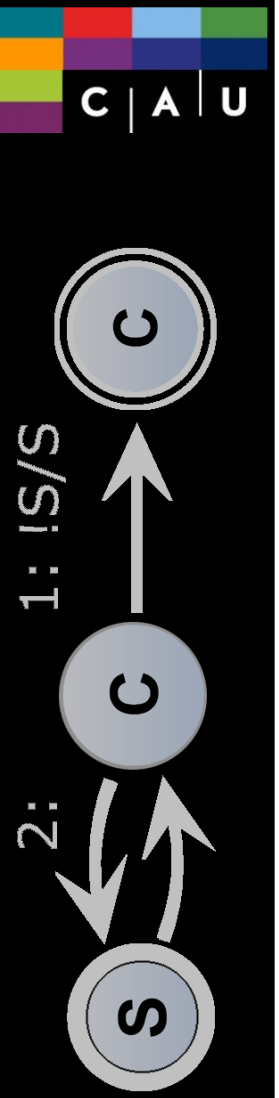
Compiling ALDO (2)



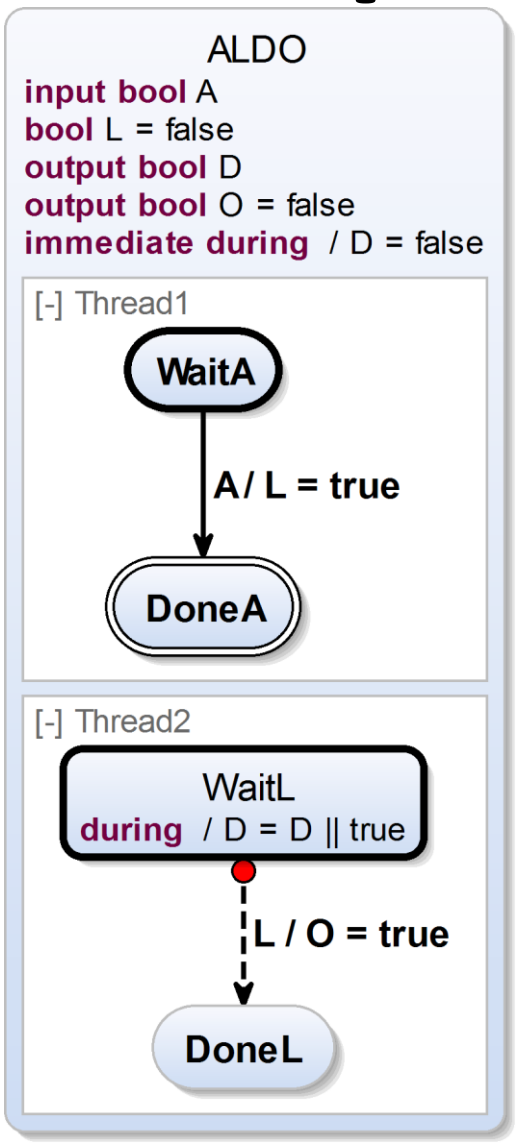


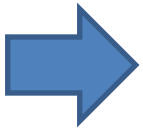
expand signal



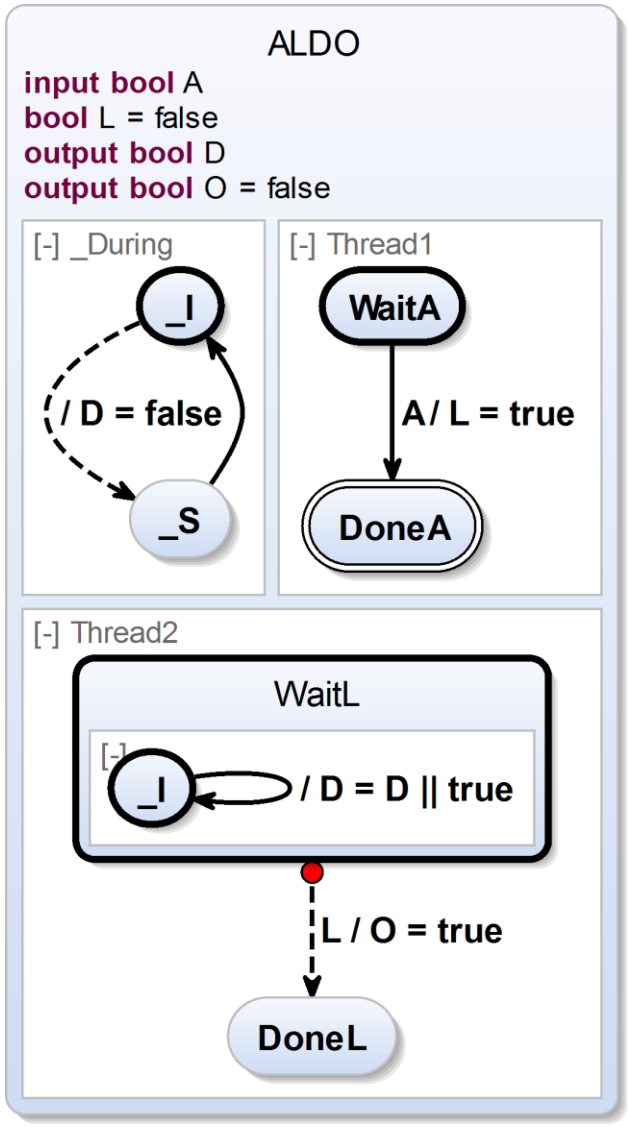


Compiling ALDO (3)

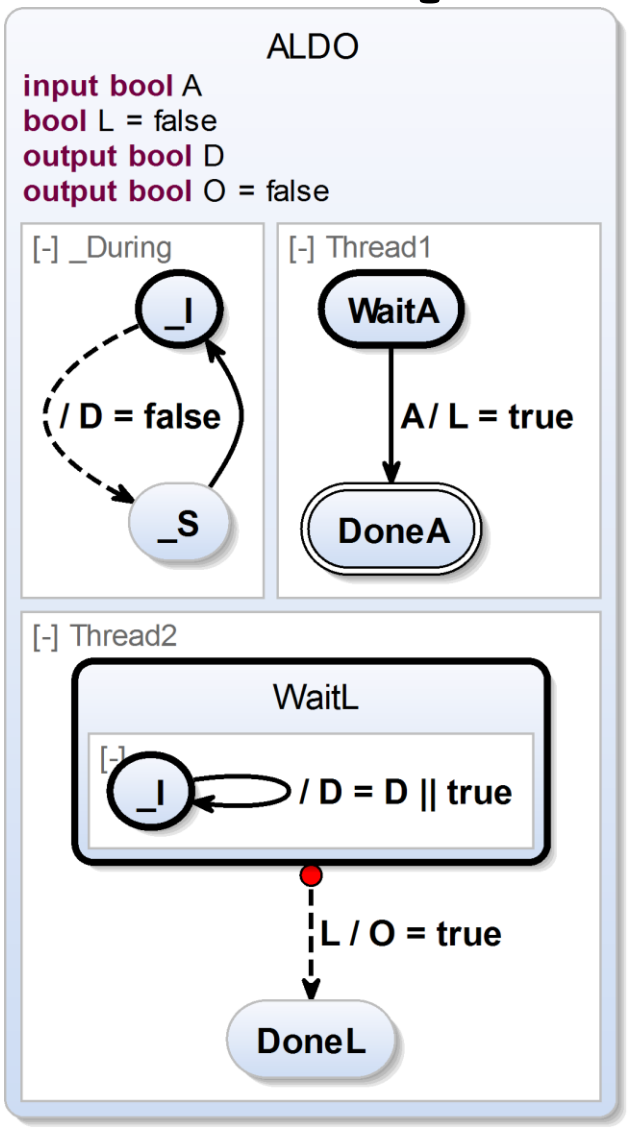




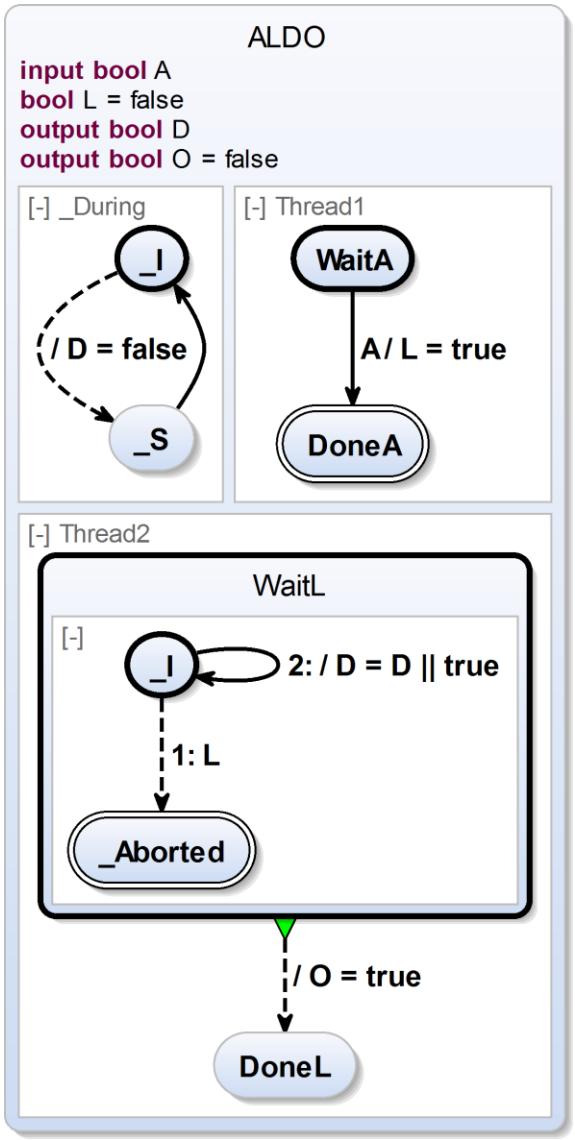
expand
during action

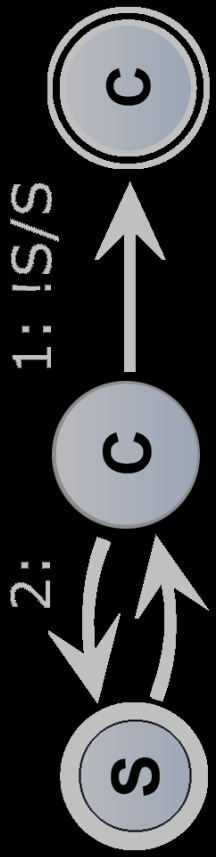


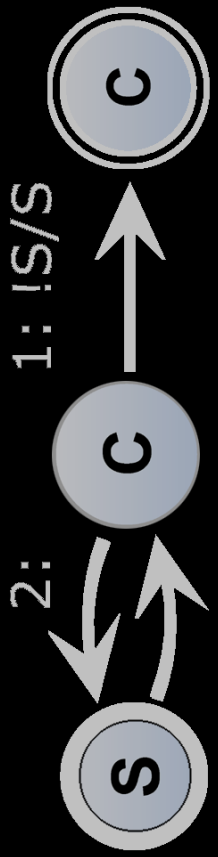
Compiling ALDO (4)



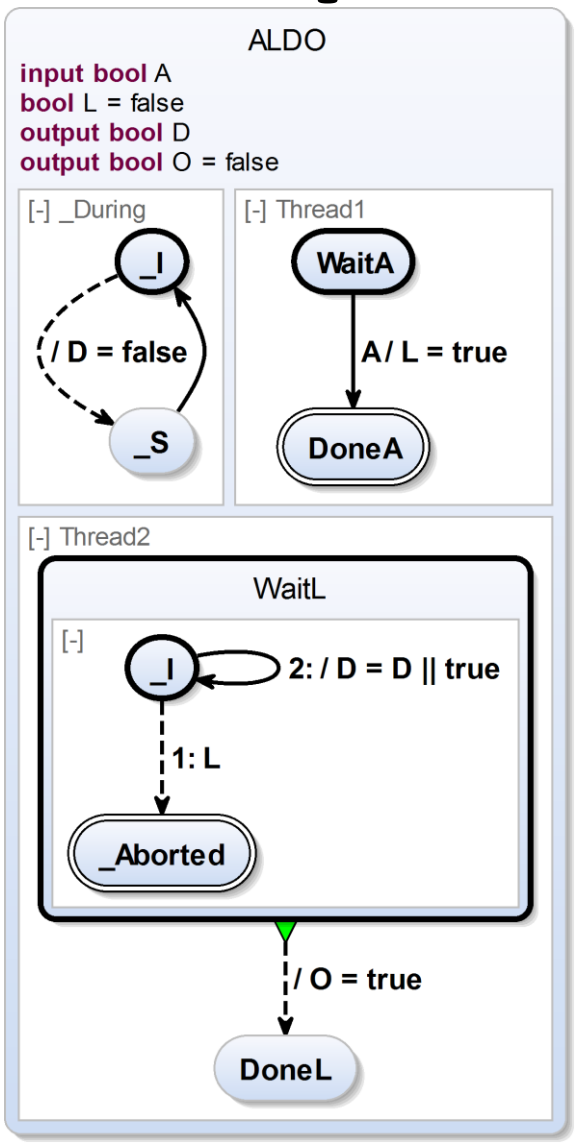
expand
abort



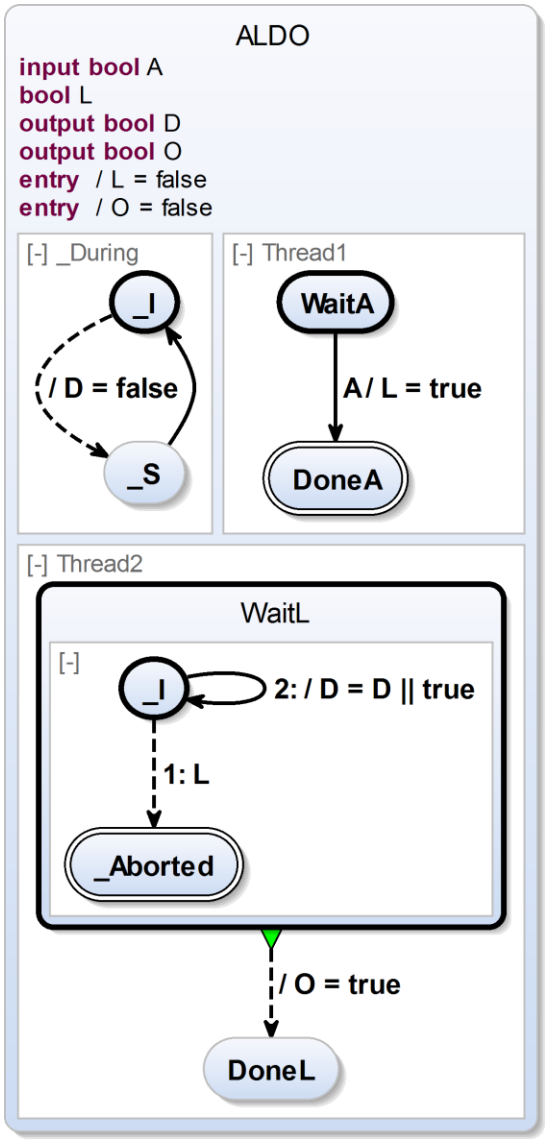


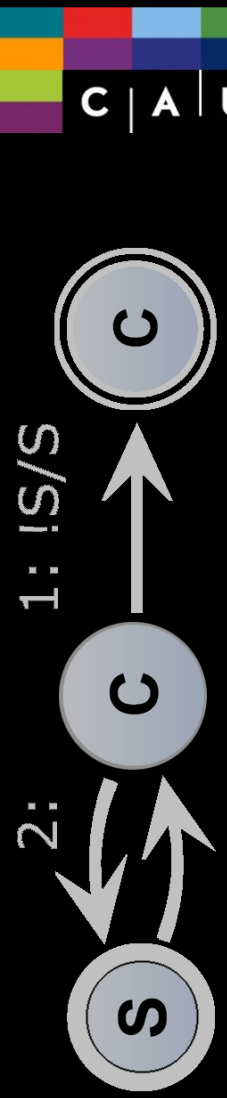


Compiling ALDO (5)

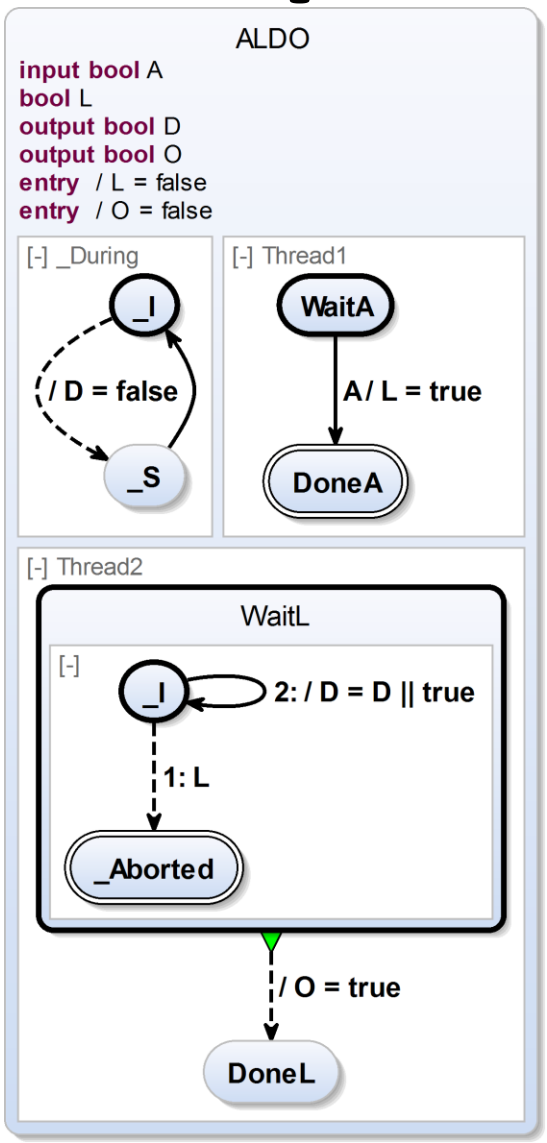


expand initialization

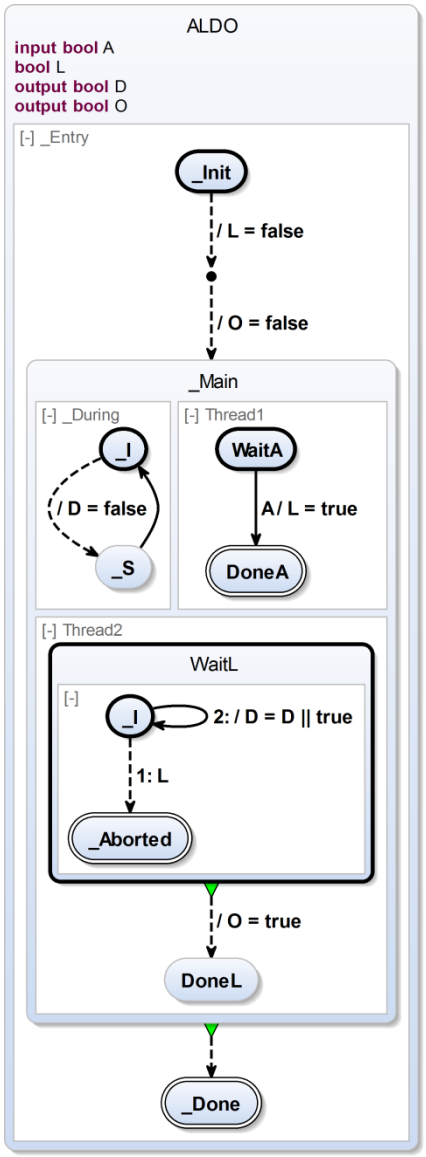


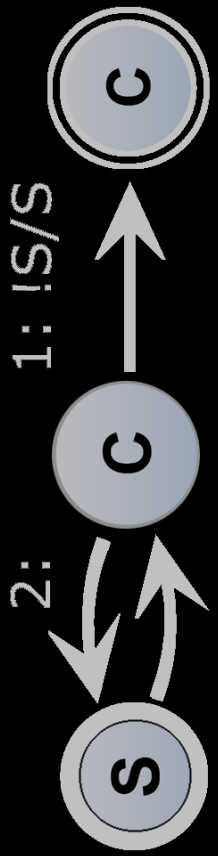


Compiling ALDO (6)

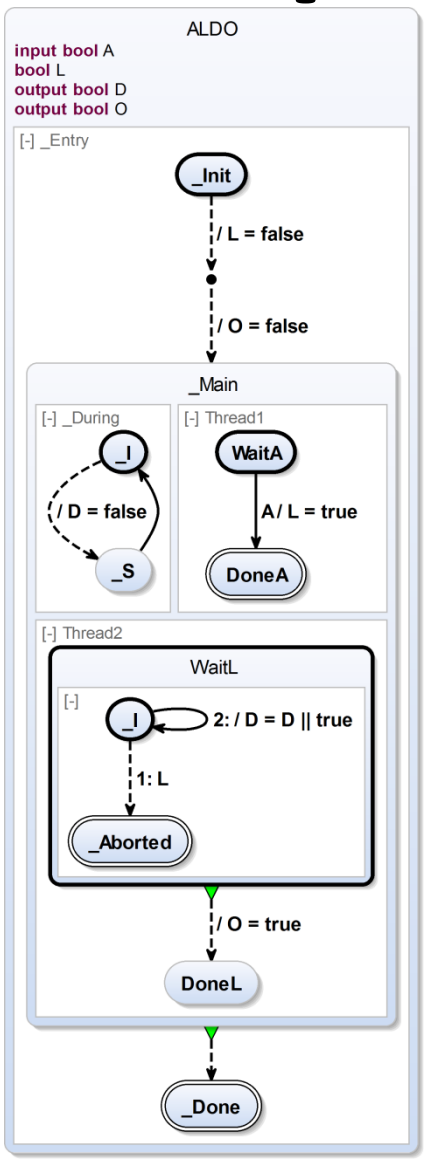


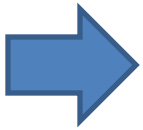
expand entry



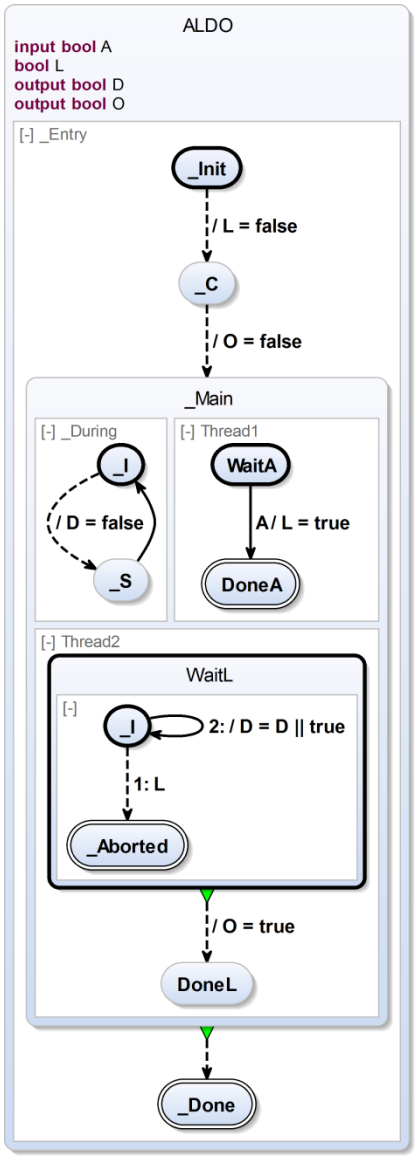


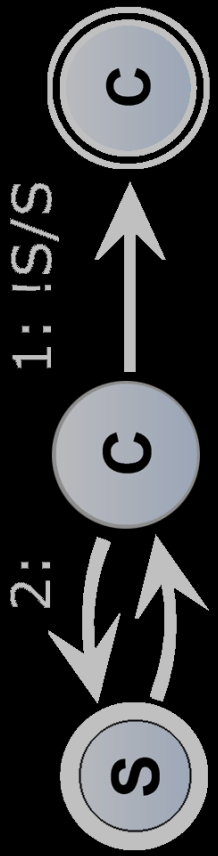
Compiling ALDO (7)



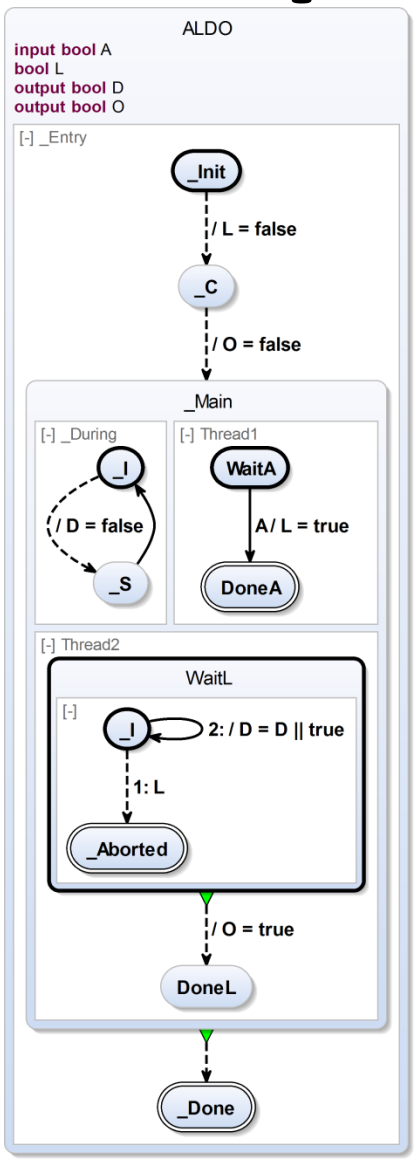


expand
connector

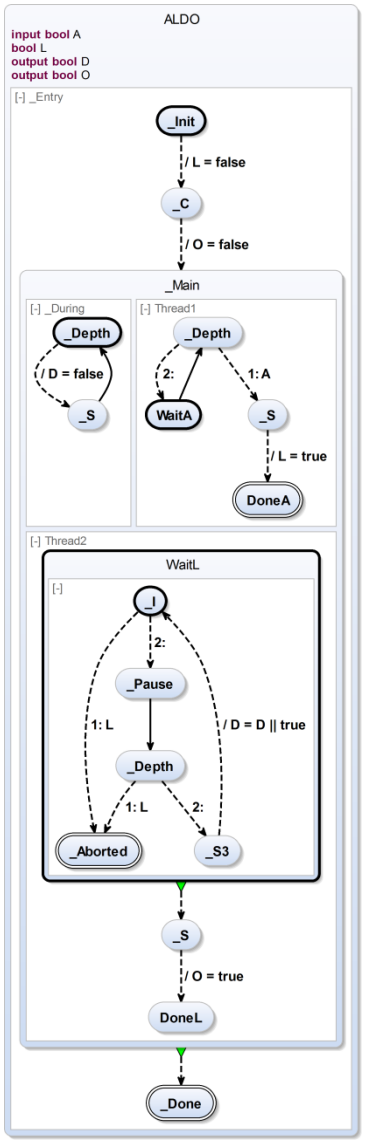


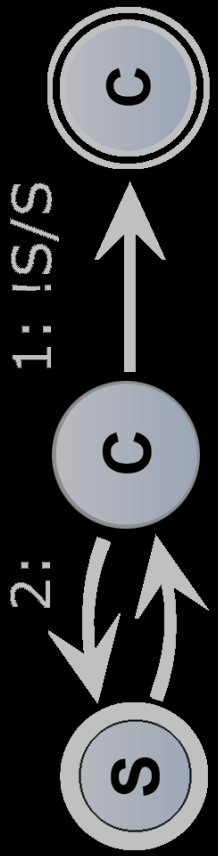


Compiling ALDO (8)

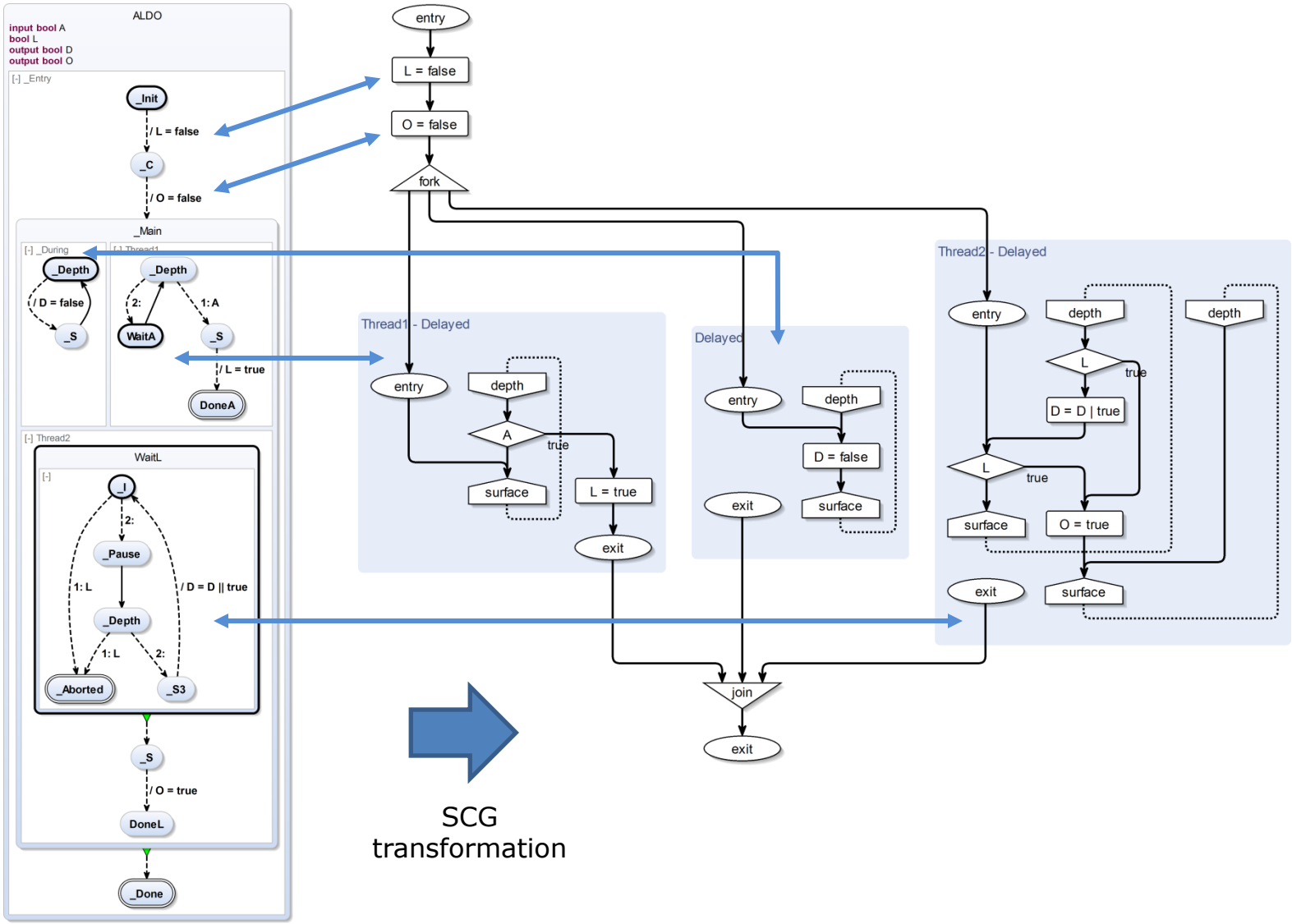


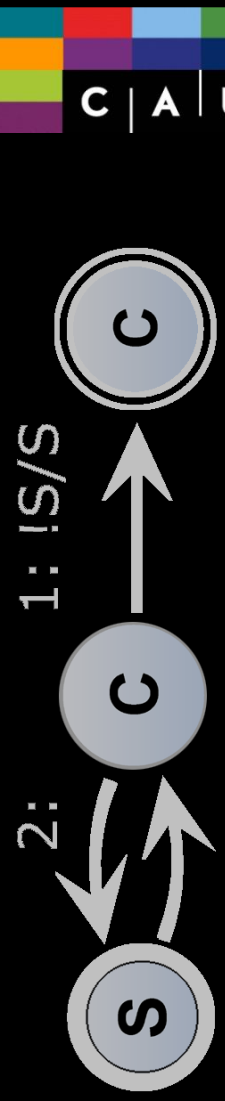
normalization



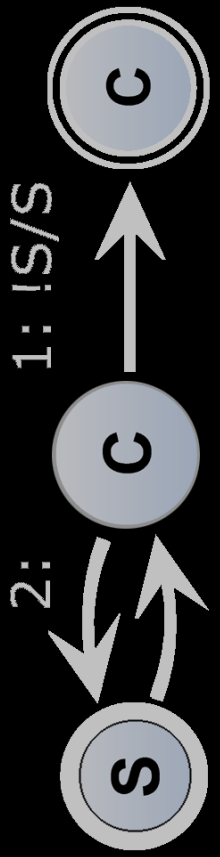


Compiling ALDO (9)

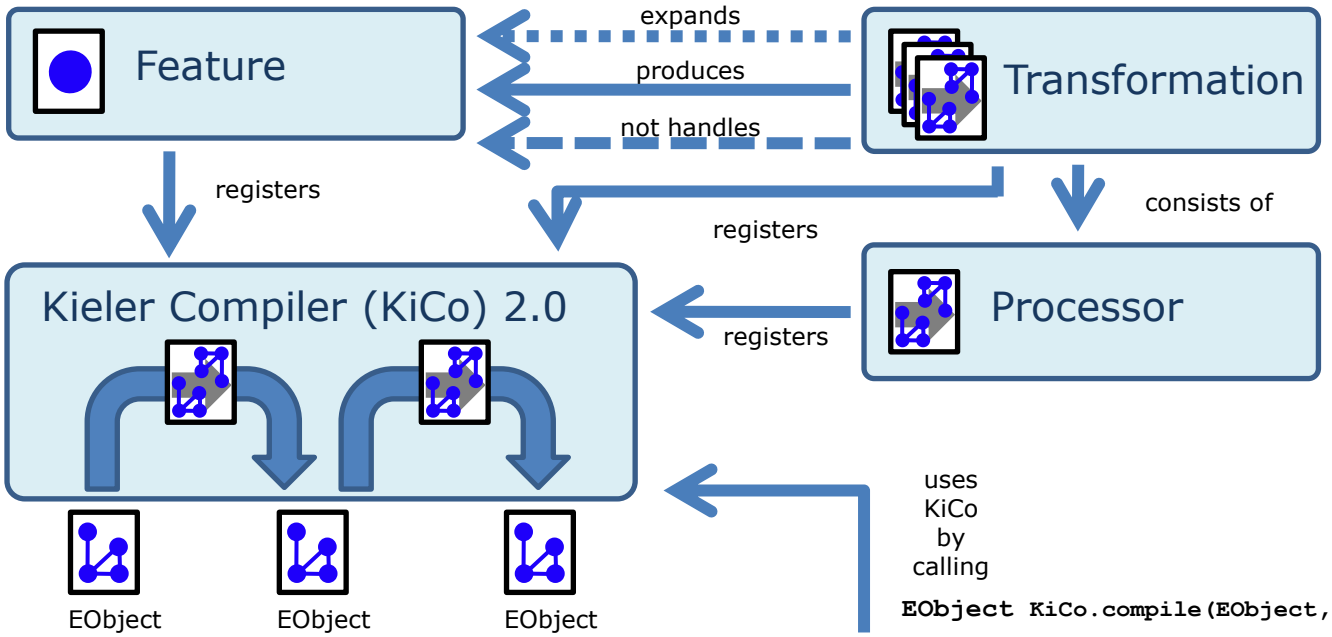


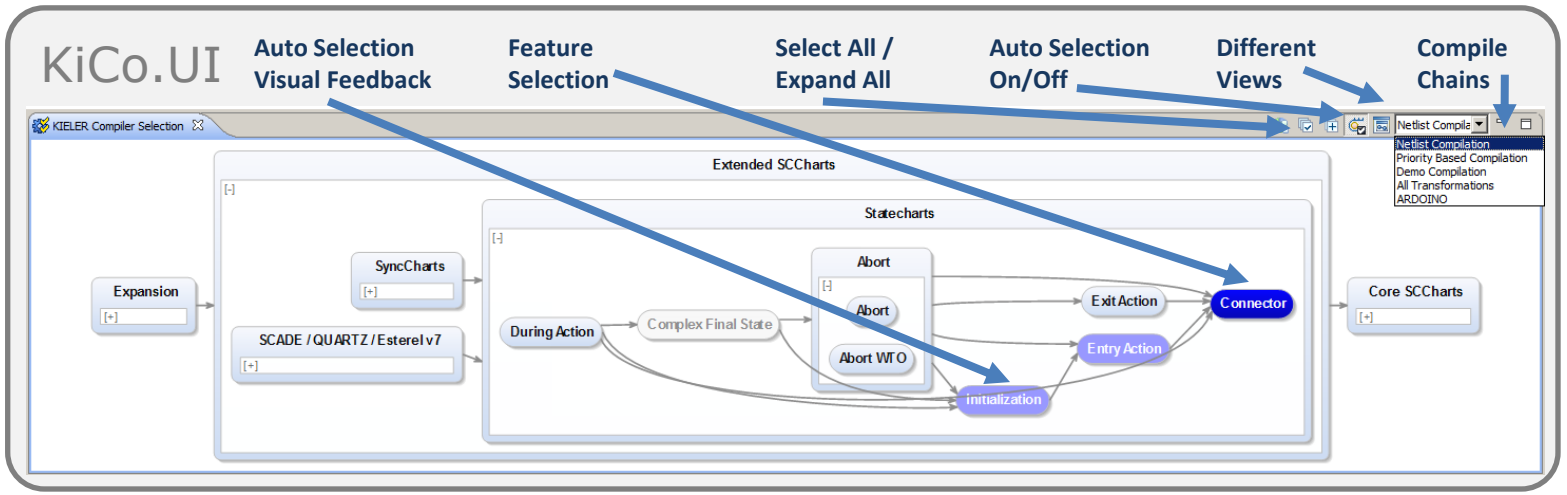


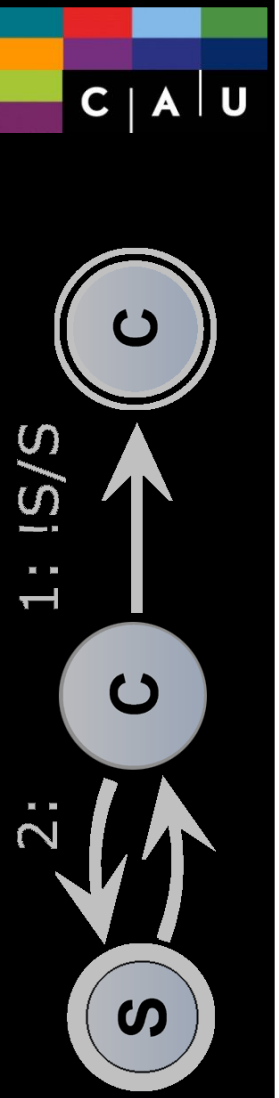
Compiling ALDO Demo


C
1: IS/S
C
2:
S

Usage


Feature
expands
produces
not handles
Transformation
registers
registers
consists of
Kieler Compiler (KiCo) 2.0
registers
Processor
EObject
EObject
EObject
uses
KiCo
by
calling
EObject KiCo.compile(EObject, ID1, ID2)


KiCo.UI
Auto Selection
Visual Feedback
Feature
Selection
Select All /
Expand All
Auto Selection
On/Off
Different
Views
Compile
Chains
KIELER Compiler Selection
Netlist Compile
Netlist Compilation
Priority Based Compilation
Demo Compilation
All Transformations
ARDOINO
Extended SCCharts
Statecharts
Expansion
SyncCharts
SCADE / QUARTZ / Esterel v7
During Action
Complex Final State
Abort
Abort
Abort WTO
Exit Action
Entry Action
Connector
initialization
Core SCCharts



Usage (2)

Feature

expands

produces

not handles

Transformation

registers

registers

consists of

Kieler Compiler (KiCo) 2.0

registers

Processor

EObject

EObject

EObject

uses KiCo by calling

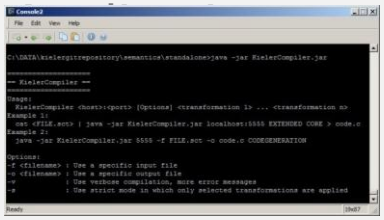
```
EObject KiCo.compile(EObject, ID1, ID2)
```

KiCo.Server
HTTP Protocol

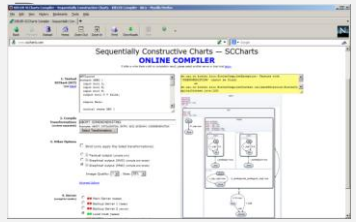
1. Read request
2. Parse Model
3. Call `KiCo.compile()`
4. Serialize result
5. Send back result

HTTP request / respond

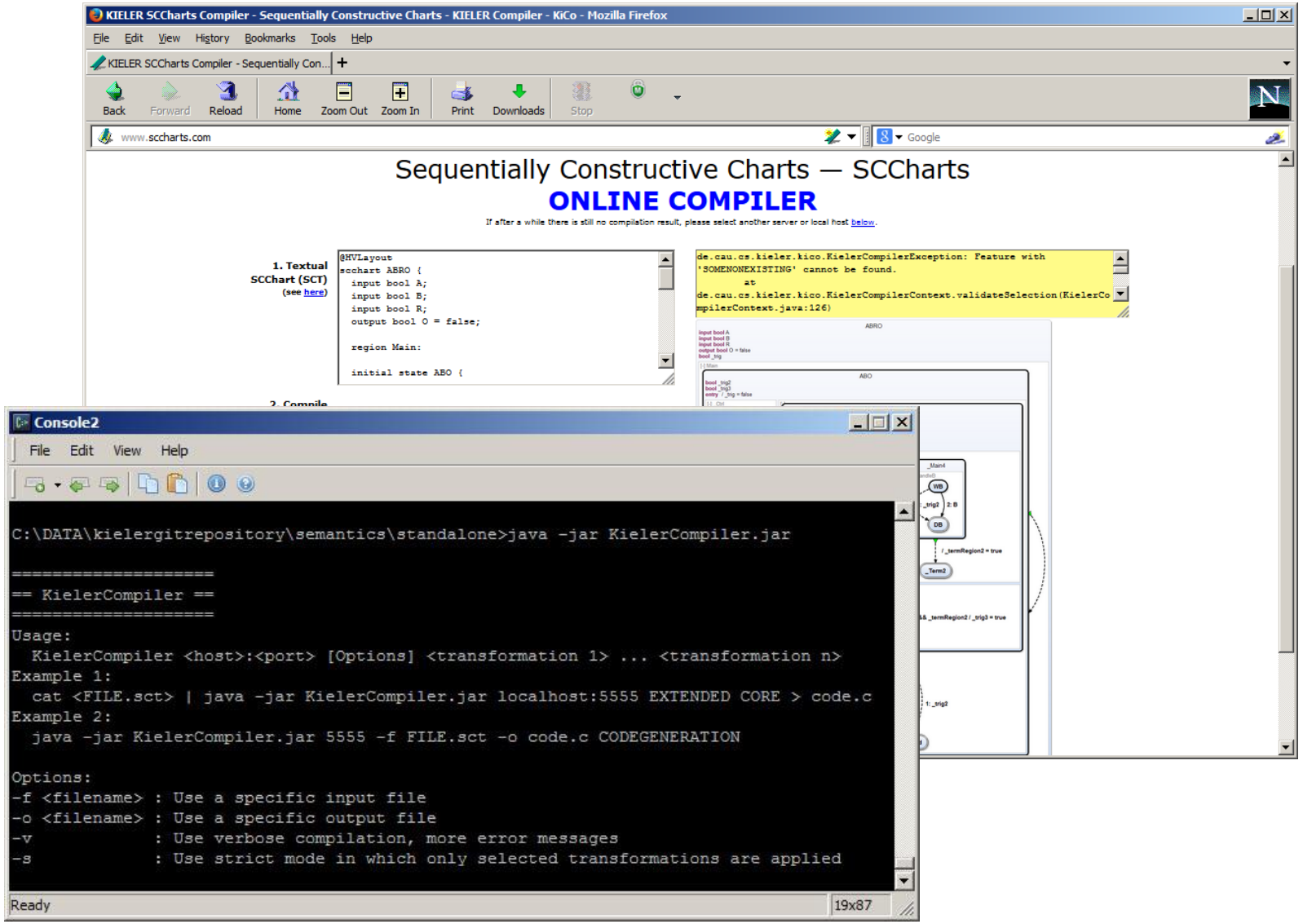
External

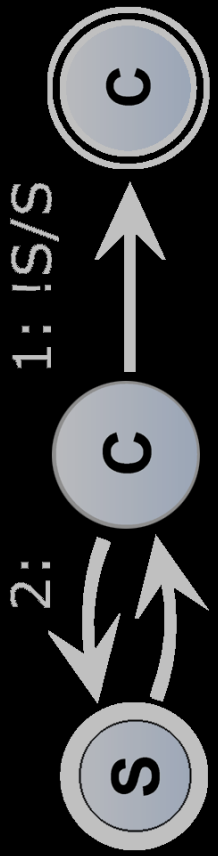


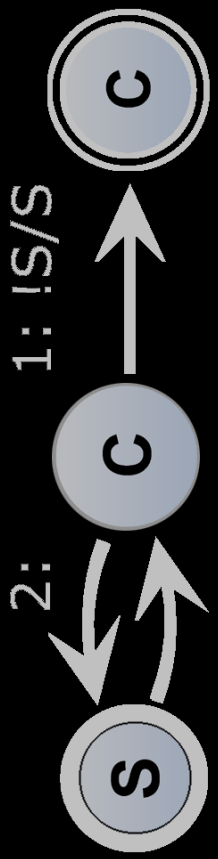
Web



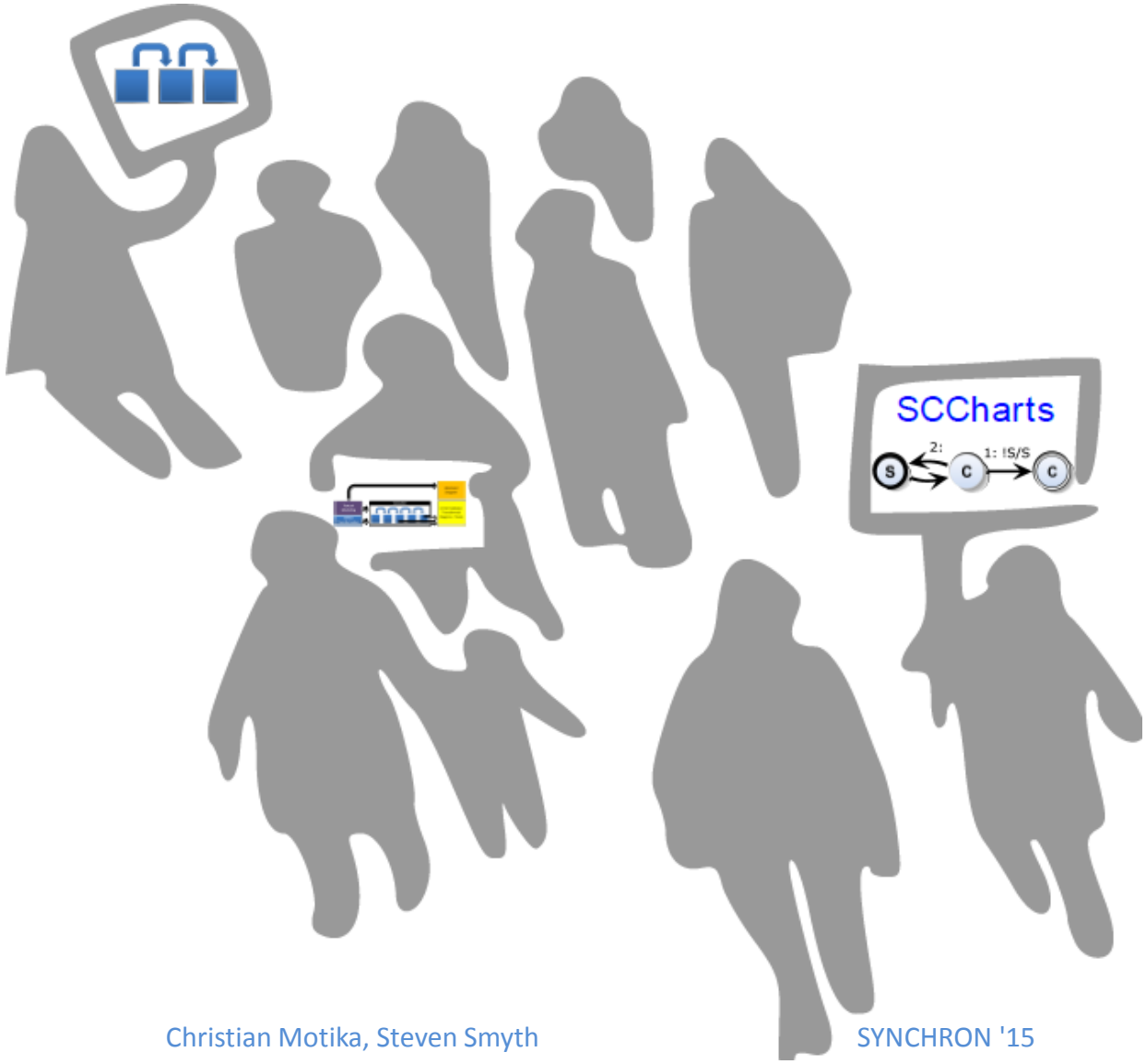
Usage (3)

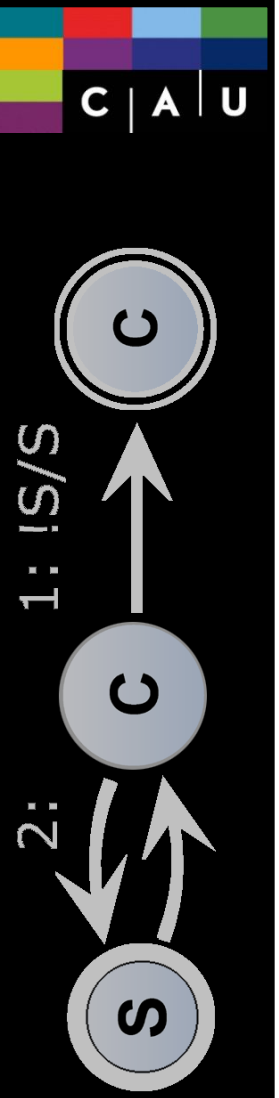

KIELER SCCharts Compiler - Sequentially Constructive Charts - KIELER Compiler - KiCo - Mozilla Firefox
Sequentially Constructive Charts — SCCharts
ONLINE COMPILER
1. Textual SCChart (SCT)
@HVLayout
scchart ABRO {
input bool A;
input bool B;
input bool R;
output bool O = false;
region Main:
initial state ABO {
de.cau.cs.kieler.kico.KielerCompilerException: Feature with 'SOMENONEXISTING' cannot be found.
at de.cau.cs.kieler.kico.KielerCompilerContext.validateSelection(KielerCompilerContext.java:126)
Console2
C:\DATA\kielergitrepository\semantics\standalone>java -jar KielerCompiler.jar
== KielerCompiler ==
Usage:
KielerCompiler <host>:<port> [Options] <transformation 1> ... <transformation n>
Example 1:
cat <FILE.sct> | java -jar KielerCompiler.jar localhost:5555 EXTENDED CORE > code.c
Example 2:
java -jar KielerCompiler.jar 5555 -f FILE.sct -o code.c CODEGENERATION
Options:
-f <filename> : Use a specific input file
-o <filename> : Use a specific output file
-v : Use verbose compilation, more error messages
-s : Use strict mode in which only selected transformations are applied
Ready
19x87


C
1: IS/S
C
2:
S

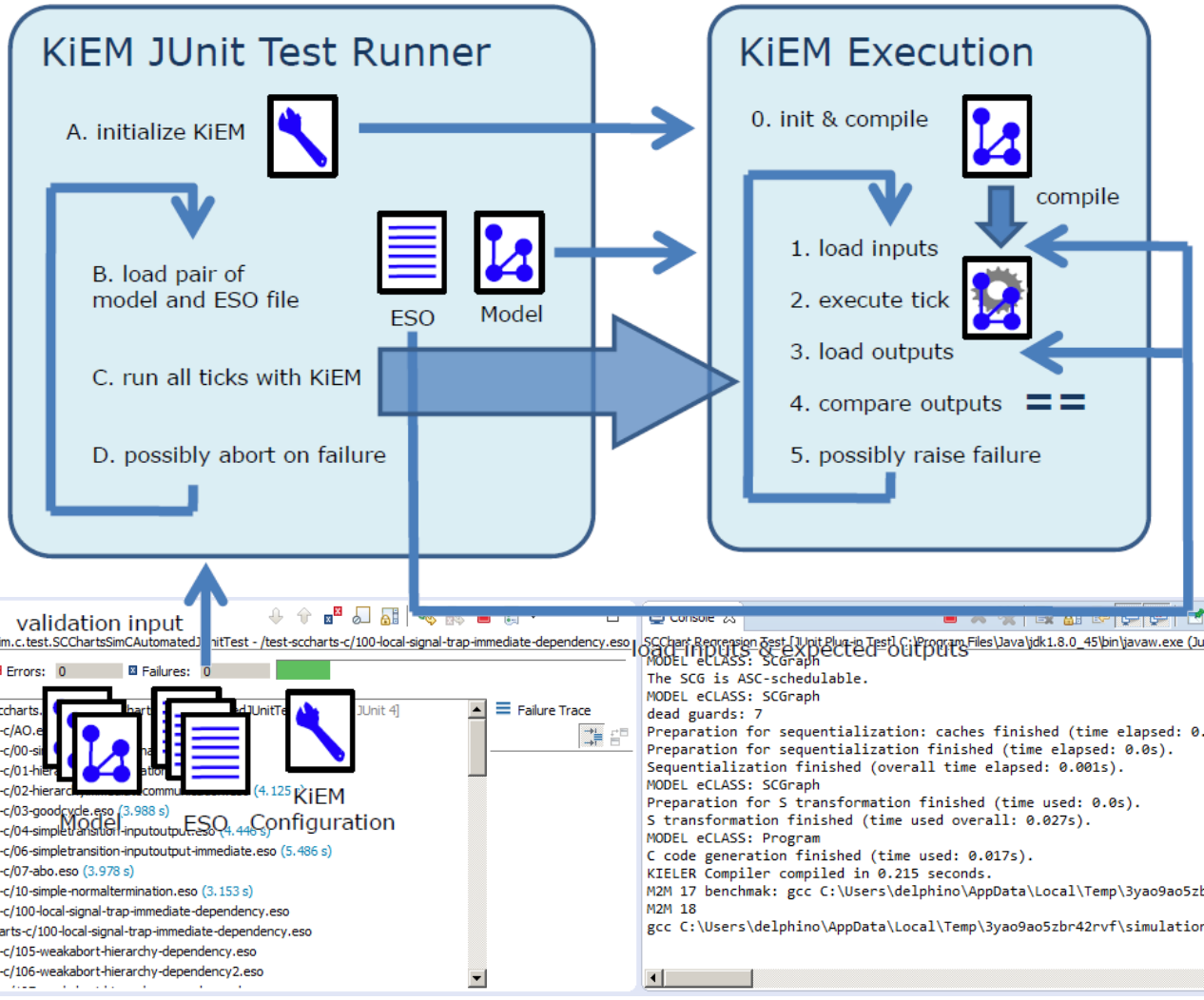


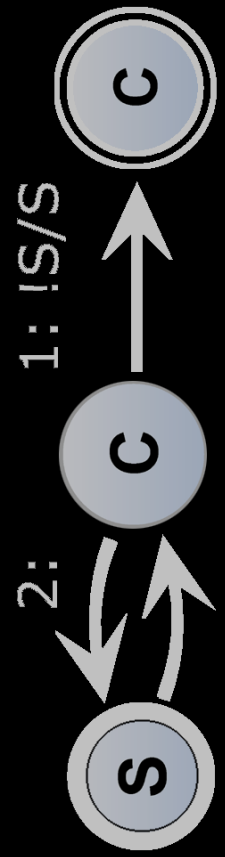
Usage Demo



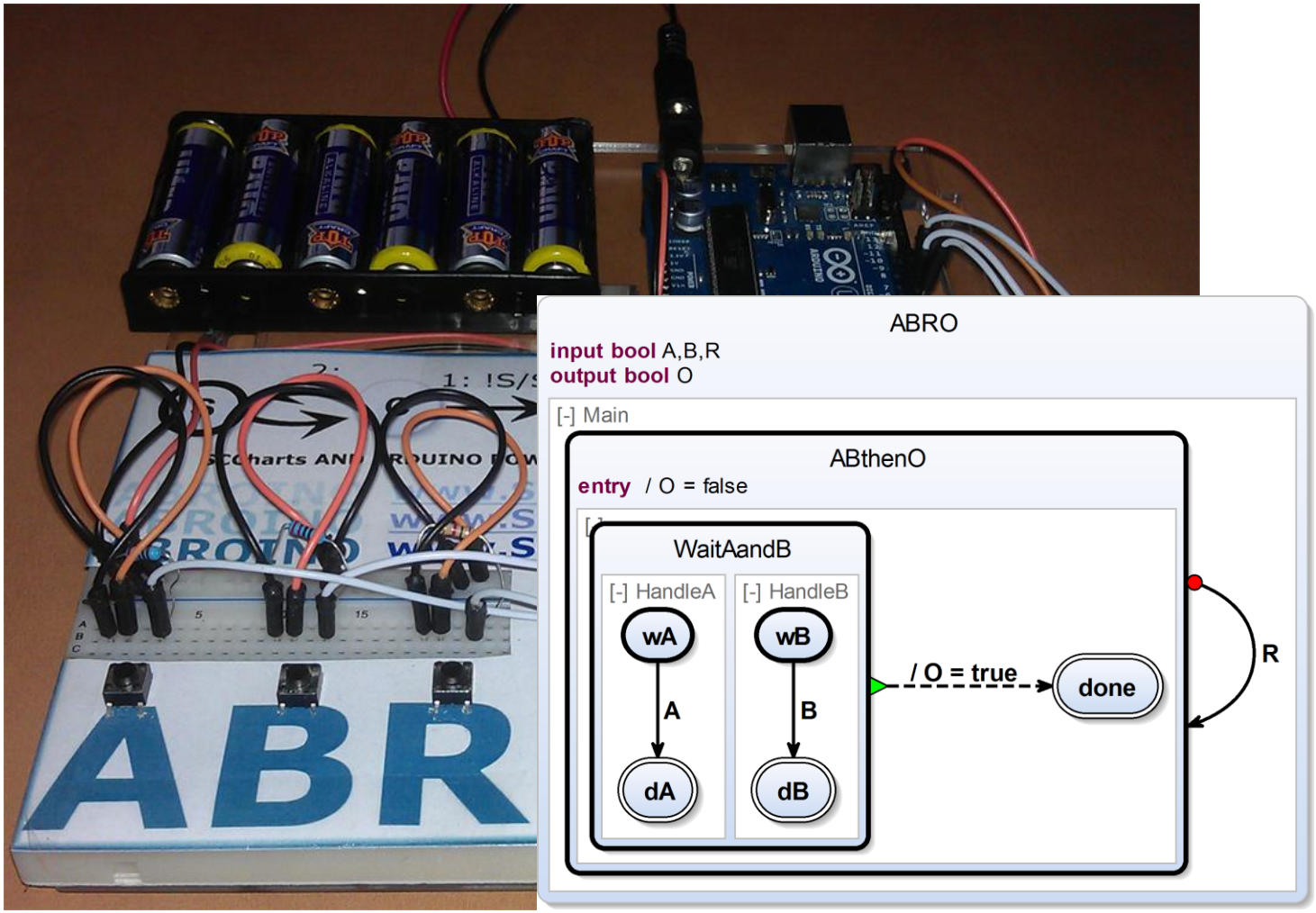


Regression Tests

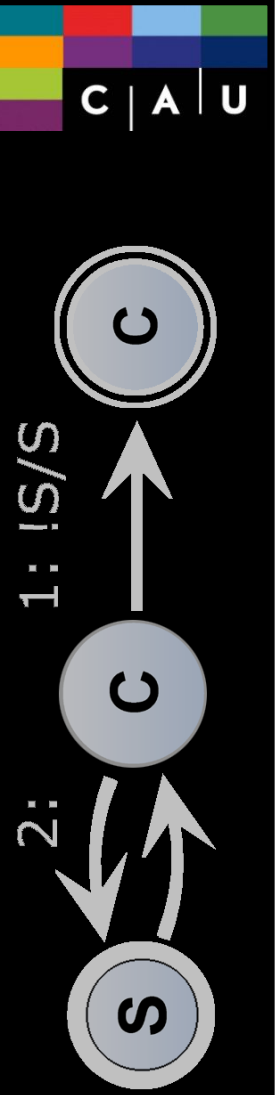




Applications

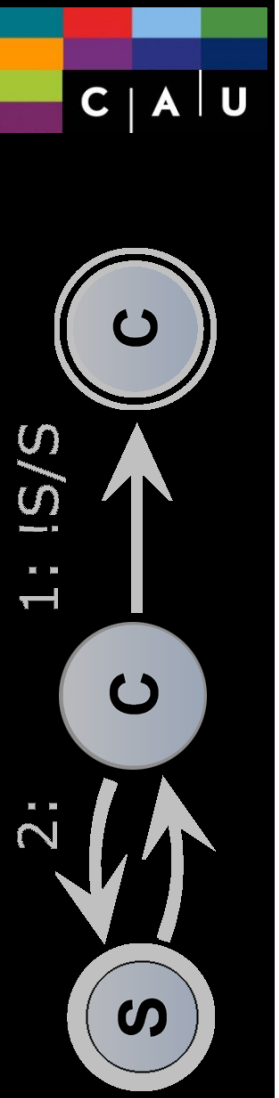


ABROINO: ABRO SCChart running on Arduino, Dec 2014



More Applications…

Some more
student projects… ☺



Proxy States

Application models often contained proxy states that

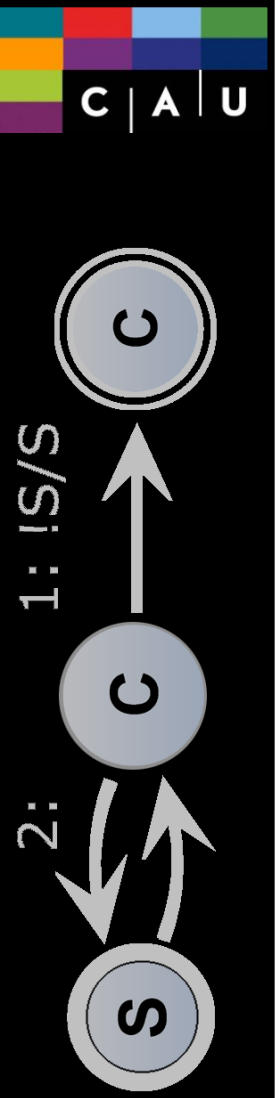
- accumulate equations
- and were left immediately

```
Dep_IC_ST
entry debug && depTrack == 1 / println("[trainNum][IC-IC] Leaving IC_ST_1")
entry debug && depTrack == 2 / println("[trainNum][IC-IC] Leaving IC_ST_2")
entry debug && depTrack == 3 / println("[trainNum][IC-IC] Leaving IC_ST_3")
entry  / railPoint(29,STRAIGHT)
entry depTrack == 1 / railPoint(24,STRAIGHT)
entry depTrack > 1 / railPoint(24,BRANCH)
entry depTrack == 2 / railPoint(23,BRANCH)
entry depTrack == 3 / railPoint(23,STRAIGHT)
entry depTrack == 1 / railSignal(IC_ST_1, FWD, GREEN)
entry depTrack == 2 / railSignal(IC_ST_2, FWD, GREEN)
entry depTrack == 3 / railSignal(IC_ST_3, FWD, GREEN)
entry depTrack == 1 / railTrack(IC_ST_1,FWD,trainNum,NORMAL)
entry depTrack == 2 / railTrack(IC_ST_2,FWD,trainNum,NORMAL)
entry depTrack == 3 / railTrack(IC_ST_3,FWD,trainNum,NORMAL)
entry  / railSignal(IC_LN_0, FWD, RED)
entry  / railTrack(IC_LN_0,FWD,trainNum,NORMAL)
entry  / railTrack(IC_ST_4,FWD,trainNum,NORMAL)
```

Proxy state within the Railway project

```
calculateNewThickness
entry  / barThickness = barTickCounter
entry  / barTolerance = barThickness * TOLERANCE_BARS_PERCENTAGE / 100
entry  / barThicknessMax = barThickness + barTolerance
entry  / reader.out(THICKNESS: , barThickness)
entry  / reader.out(THICKNESS: , barThicknessMax)
entry  / reader.beep()
```

Proxy state within the Mindstorms project



Proxy States

Application models often contained proxy states that

- accumulate equations
- and were left immediately

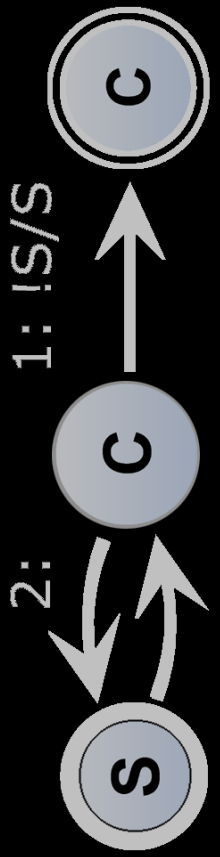
Would be really cool if we could express this in a dataflow way.

However, we don't want to change the semantics of Core SCCharts!

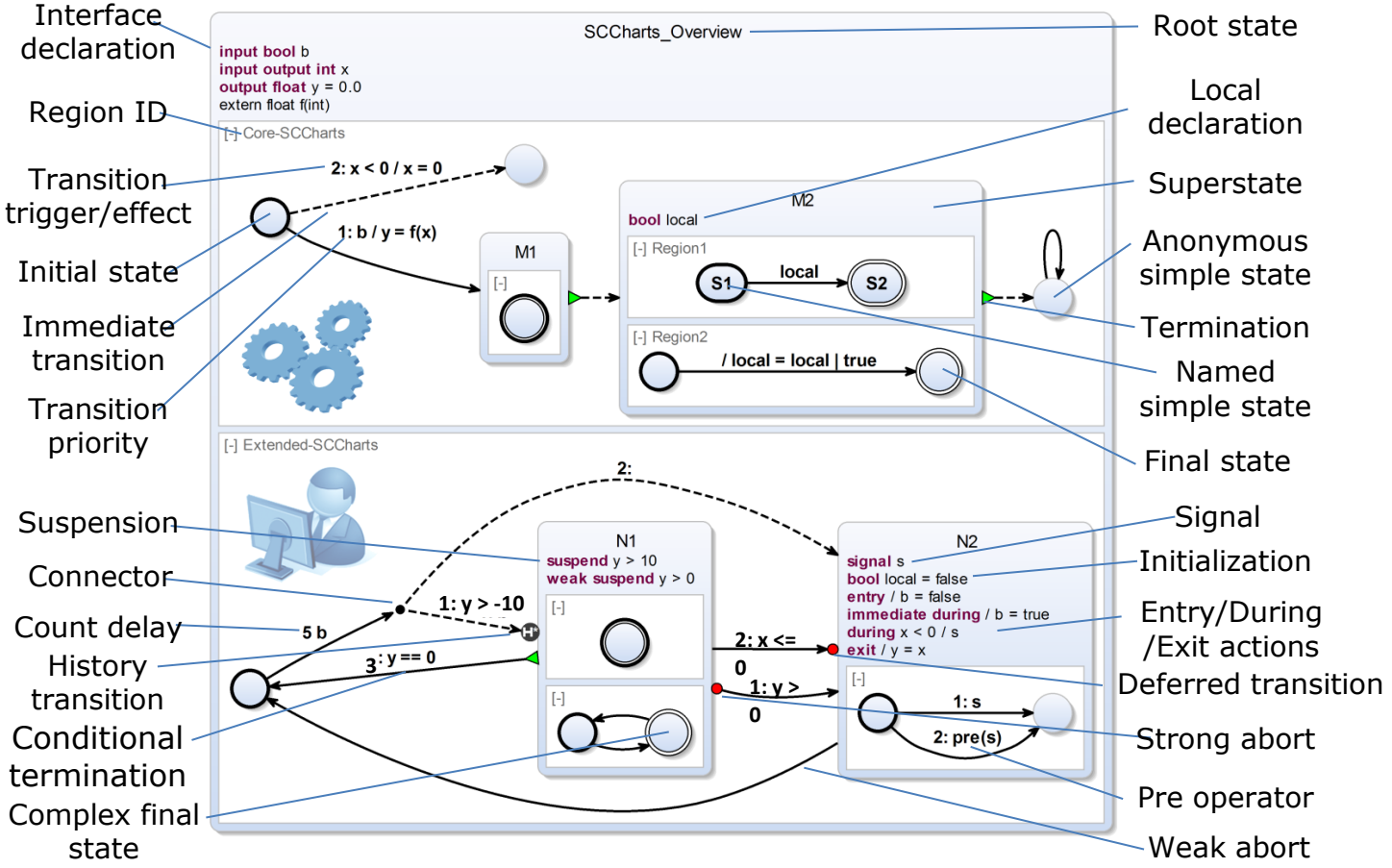
→ Add a new Extended SCCharts feature!

```
calculateNewThickness
entry  / barThickness = barTickCounter
entry  / barTolerance = barThickness * TOLERANCE_BARS_PERCENTAGE / 100
entry  / barThicknessMax = barThickness + barTolerance
entry  / reader.out(THICKNESS: , barThickness)
entry  / reader.out(THICKNESS: , barThicknessMax)
entry  / reader.beep()
```

Proxy state within the Mindstorms project

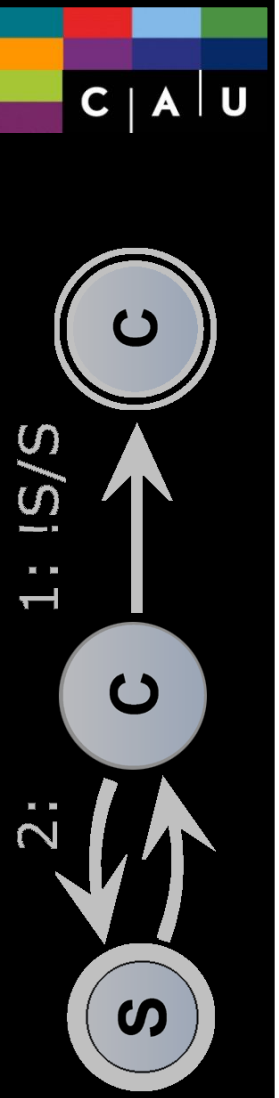


Recall SCCharts



Core-SCCharts
Small set of simple features ease down stream compilation

Extended-SCCharts
Rich set of advanced features ease modeling



Dataflow Regions

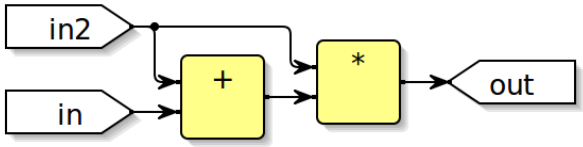
Add dataflow regions as Extended SCCharts feature

```
dataflow:
  input bool in;
  output bool out;

  out = in;
```

Simple input/output example

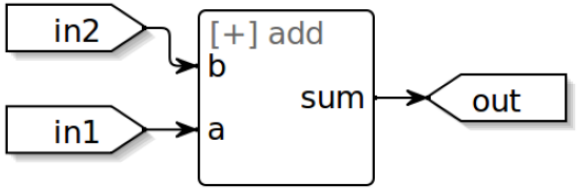

```
dataflow:
  input int in, in2;
  output int out;

  out = (in + in2) * in2;
```

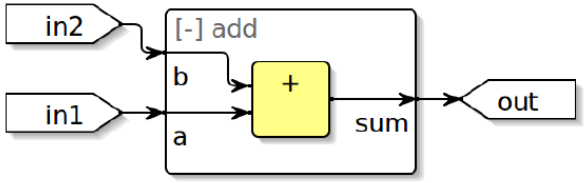
Simple equation example

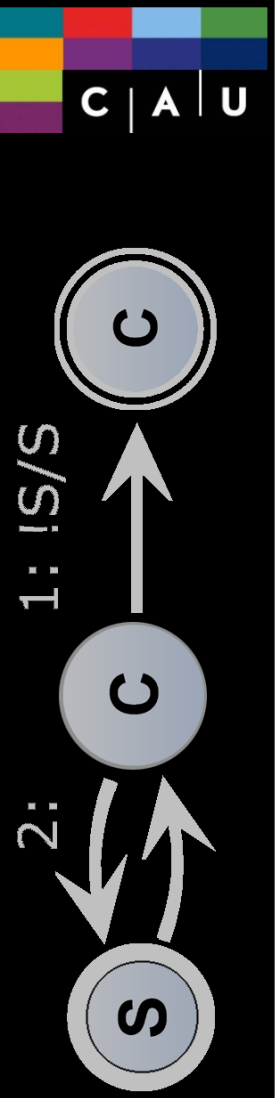



```
dataflow:
  input int in1, in2;
  output int out;

  node add(int a, b)
          returns (int sum) {
    sum = a + b;
  }

  call = add(in1, in2);
  out = call.sum;
```

Node example



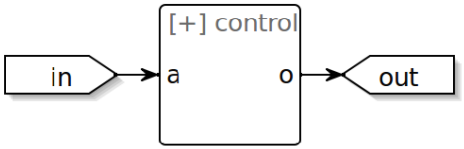




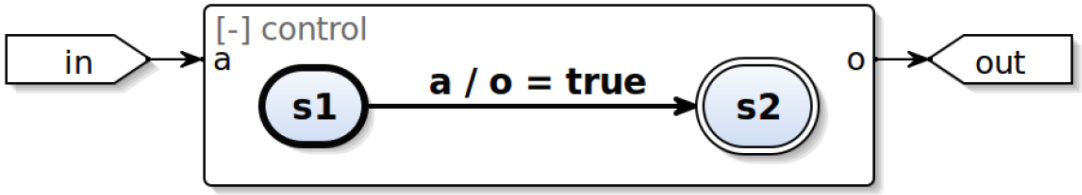
Dataflow Regions

Dataflow regions and control-flow regions co-exist

```
dataflow:
  input bool in;
  output bool out;

  node control(bool a) returns (bool o) {
    initial state s1
      ⟶ s2 with a / o = true;
    final state s2;
  }
  call = control(in);
  out = call.o;
```



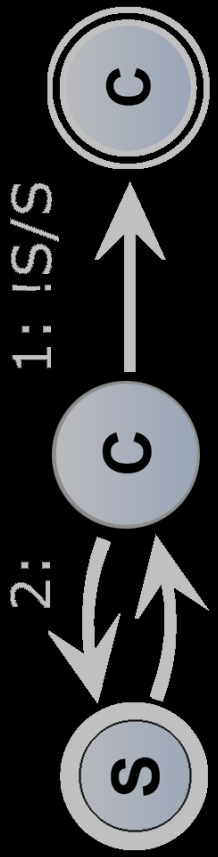
Defining new node
Containing an
control-flow region



Expanded *control* shows
embedded sub-chart

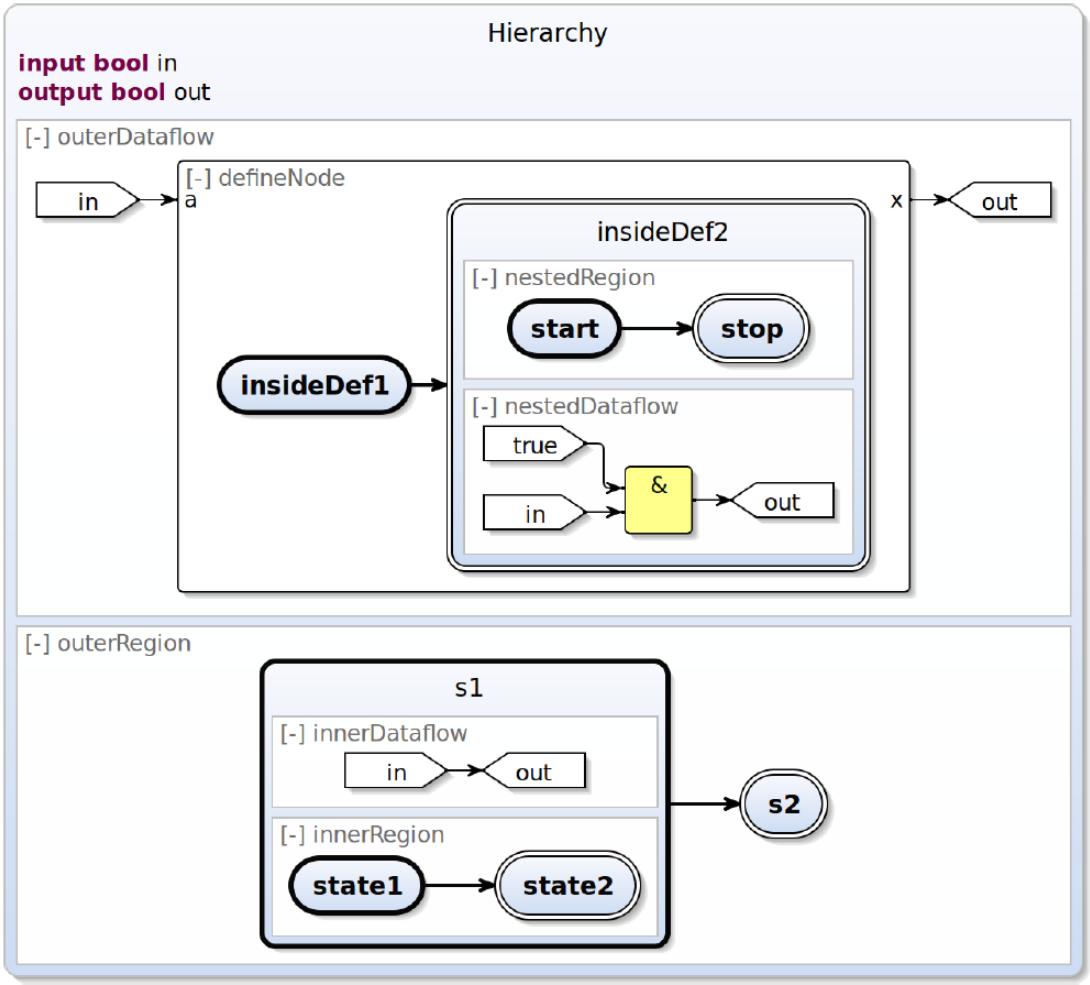
```
dataflow:
  input bool in1, in2;
  output bool out;

  call = ref flipflop(in1, in2);
  out = call.O;
```

Or simply reference
another SCChart

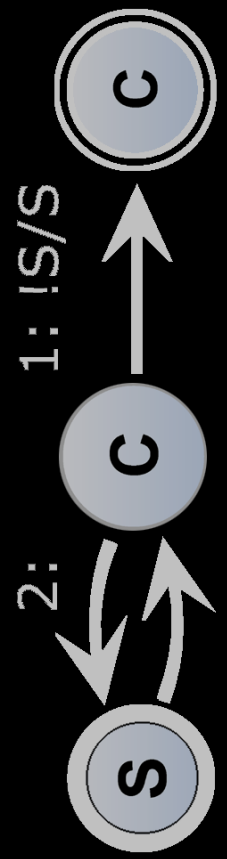


Dataflow Regions

Dataflow regions and control-flow regions co-exist

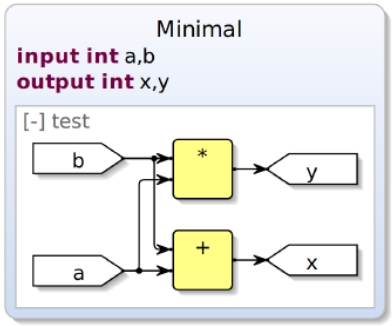


Hybrid SCCharts example



Dataflow Transformation

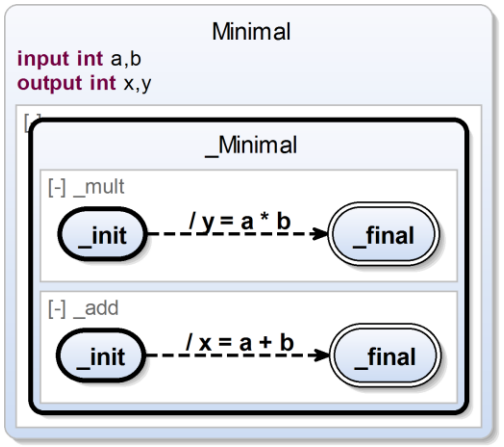
Several approaches a possible to transform the extended feature



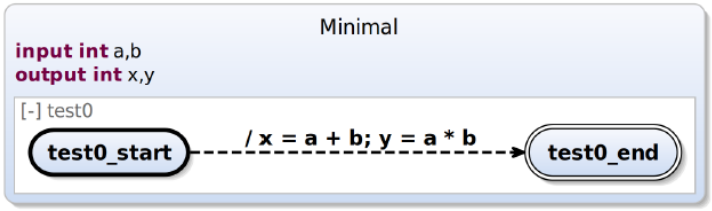
Minimal example

```
1 scchart Minimal {
2   input int a, b;
3   output int x, y;
4
5   dataflow test:
6     x = a + b;
7     y = a * b;
8 }
```

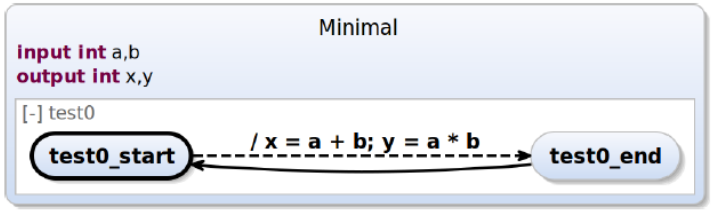
Transform single pass evaluation with concurrent regions



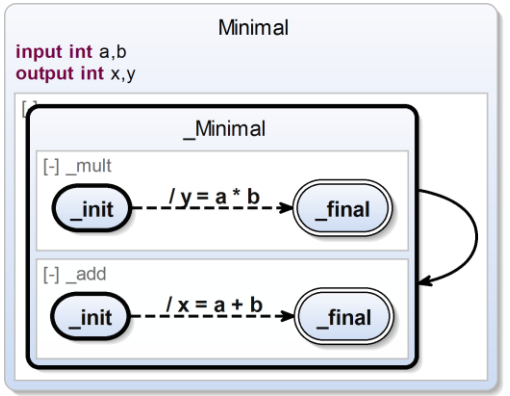
Transform single pass evaluation with initial and final state



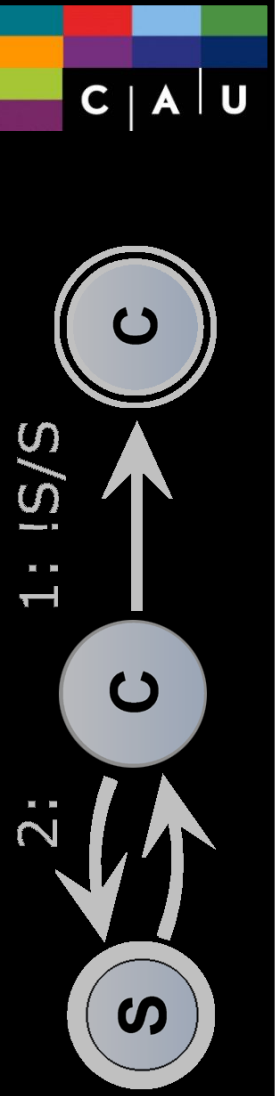
Transform frequent evaluation



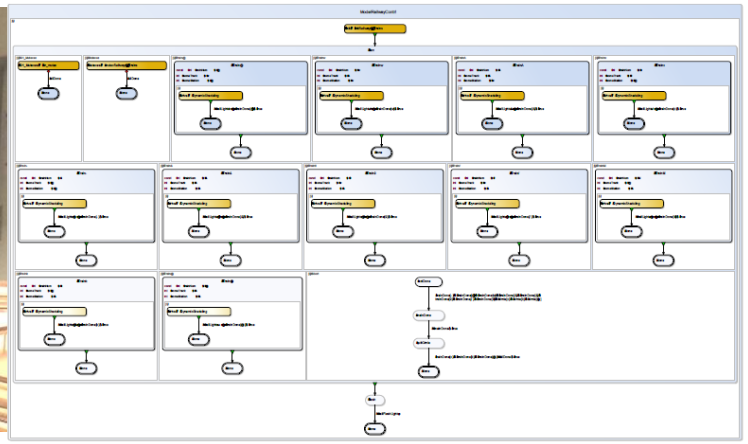
Transform frequent evaluation



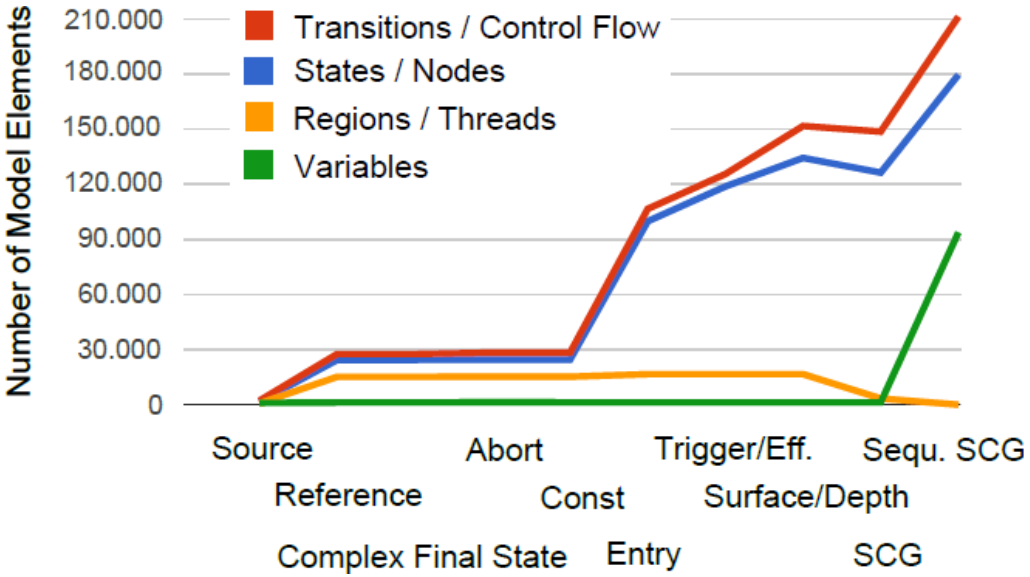
Initialize-Update-Read protocol



Applications

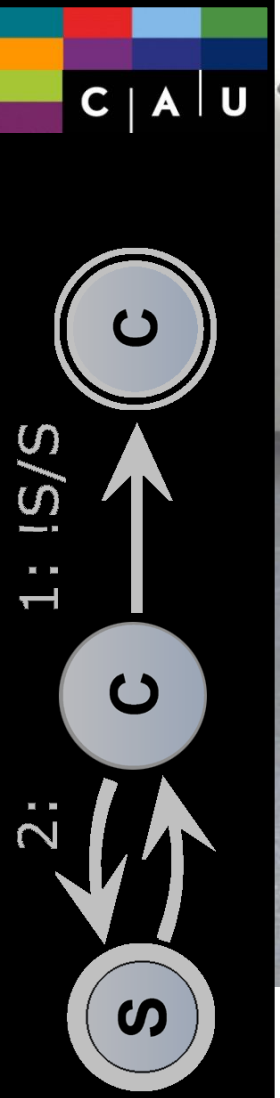


SCCharts Model Railway Controller Project 2014



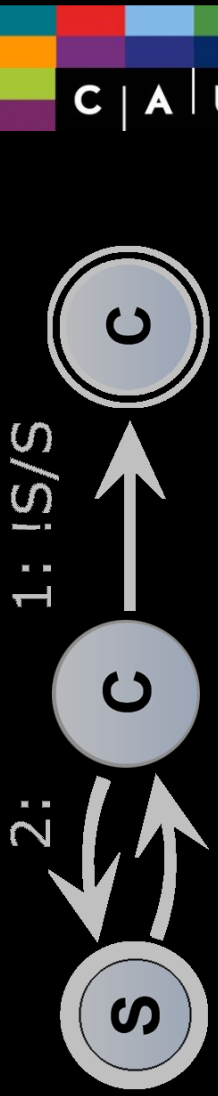
STATES:

- 1,628 modeled
- 135,000 expanded

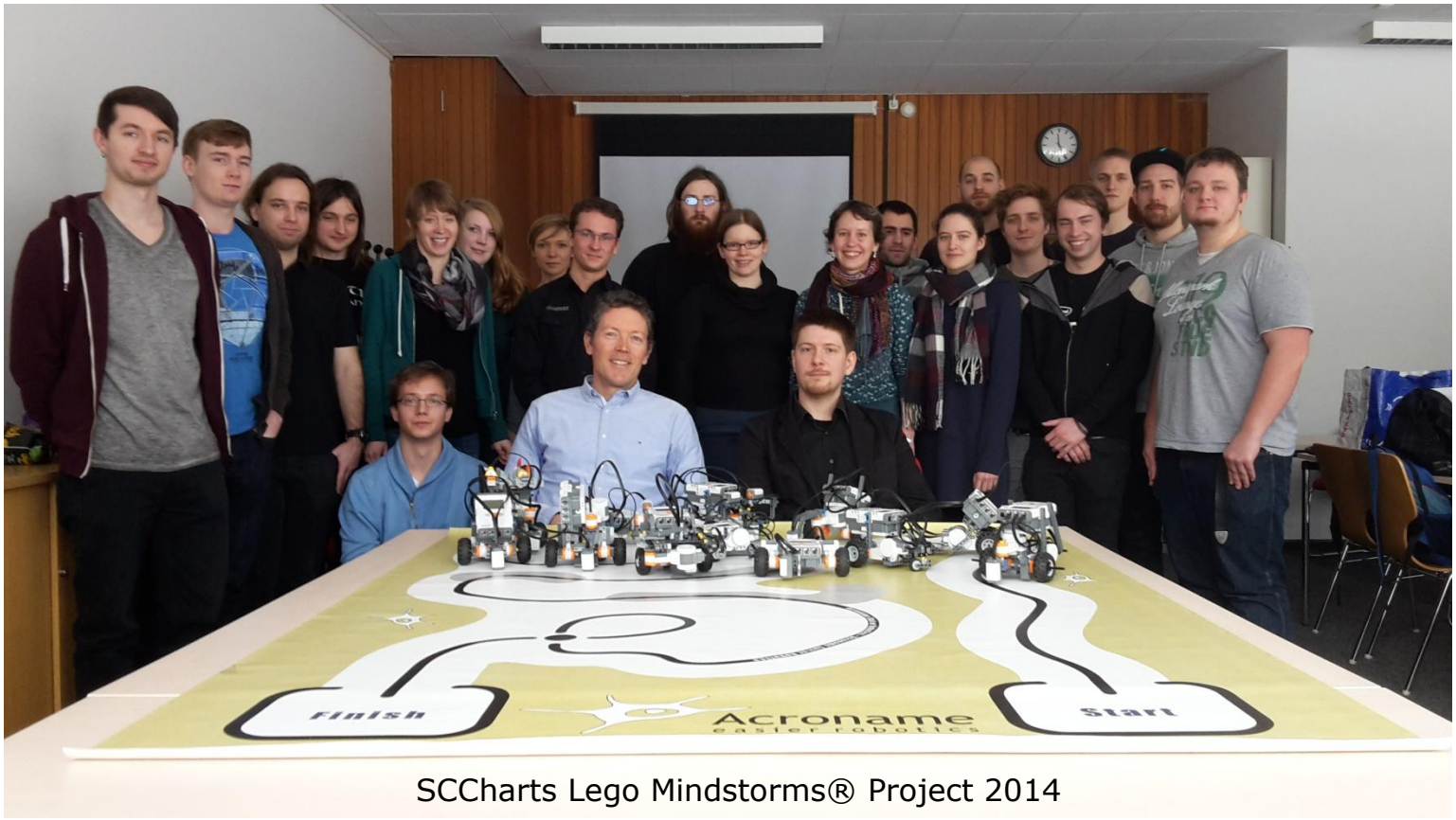


Applications (3)

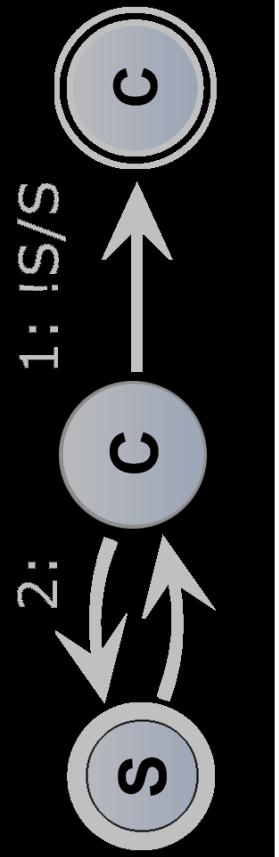
SCCharts Quadrocopter Project 2015



Applications (4)



SCCharts Lego Mindstorms® Project 2014



To Go Further

CHARLES ANDRÉ.
Semantics of SyncCharts, 2003.

GÉRARD BERRY.
The Esterel v5 Language Primer, 2000.

MOTIKA, C., SMYTH, S., AND VON HANXLEDEN, R.
Compiling SCCharts – A Case-Study on Interactive Model-Based Compilation.
6th International Symposium On Leveraging Applications of Formal Methods, Verification (ISoLA'14), Corfu, Oct 2014.

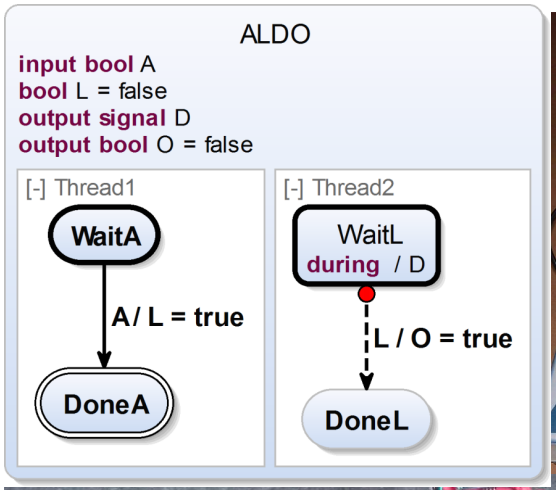
SCHNEIDER, C., SPÖNEMANN, M., AND VON HANXLEDEN, R.
Just model! – Putting automatic synthesis of node-link-diagrams into practice.
In Proceedings of the IEEE Symposium on Visual Languages and Human-Centric Computing (VL/HCC'13) (San Jose, CA, USA, 15–19 Sept. 2013).

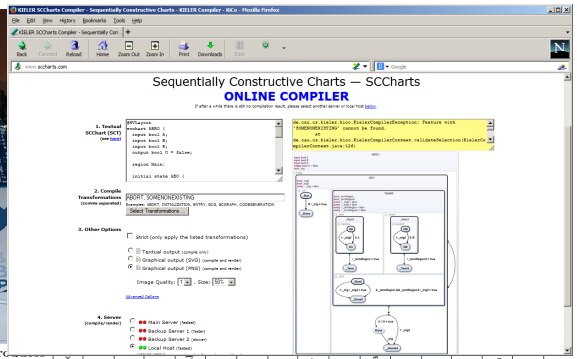
UNI KIEL, REAL-TIME AND EMBEDDED SYSTEMS GROUP.
KIELER & SCCharts webpage.
http://www.informatik.uni-kiel.de/en/rtsys/kieler/. & http://www.sccharts.com.

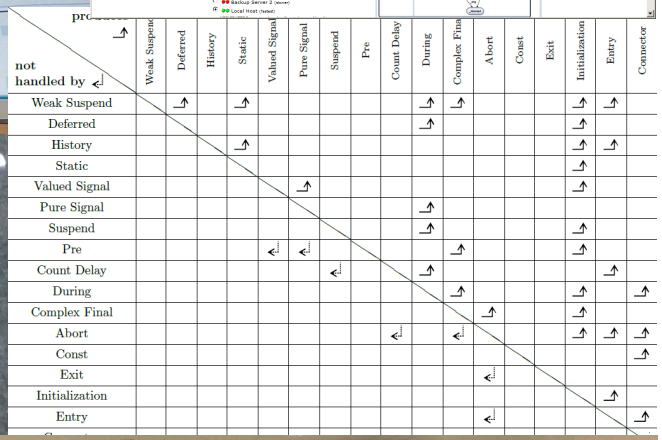
VON HANXLEDEN, R., LEE, E. A., MOTIKA, C., AND FUHRMANN, H.
Multi-view modeling and pragmatics in 2020 — position paper on designing complex cyber-physical systems.
In Proceedings of the 17th International Monterey Workshop on Development, Operation and Management of Large-Scale Complex IT Systems, LNCS (Oxford, UK, Dec. 2012), vol. 7539.

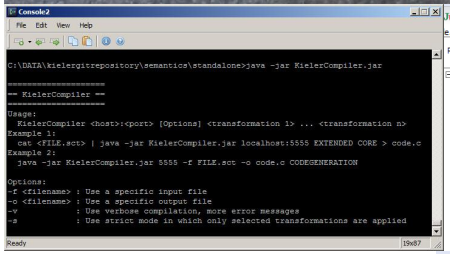
VON HANXLEDEN, R., DUDERSTADT, B., MOTIKA, C., SMYTH, S., MENDLER, M., AGUADO, J., MERCER, S., AND O'BRIEN, O.
Sequentially Constructive Concurrency—A conservative extension of the synchronous model of computation.
Proc. ACM SIGPLAN conference on Programming Language Design and Implementation (PLDI'14), Edinburgh, Jun 2014.

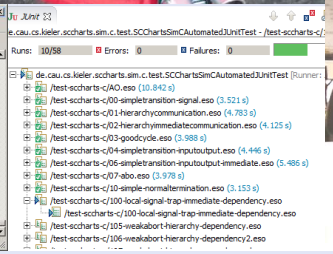
That's all Folks - Thank You!

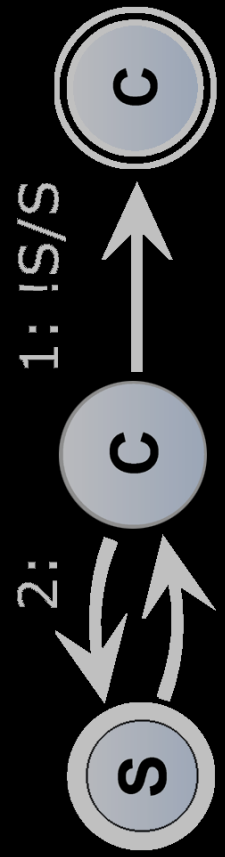






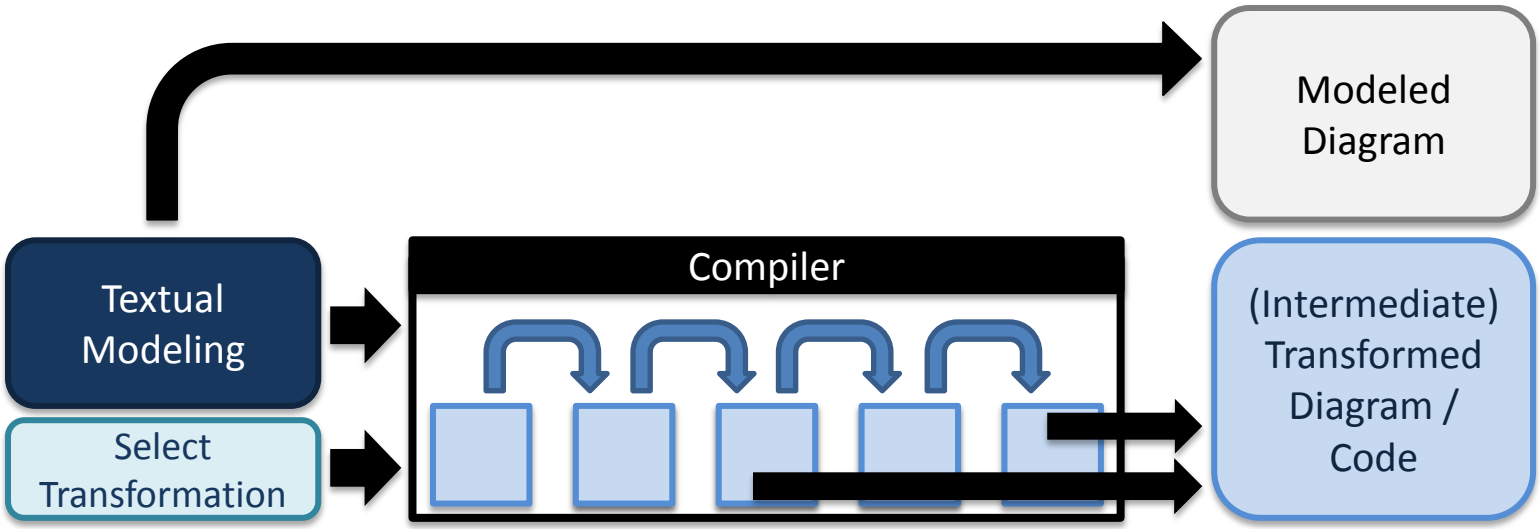


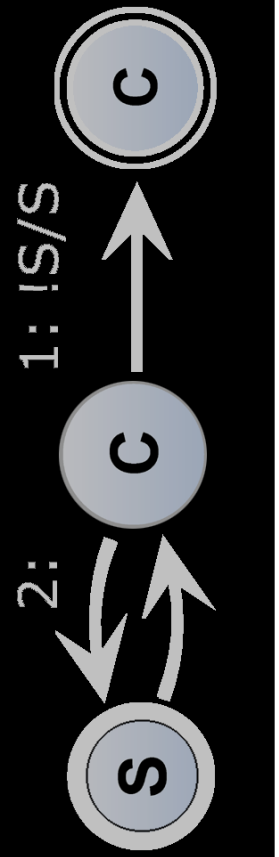




Traditional vs. Interactive SLIC

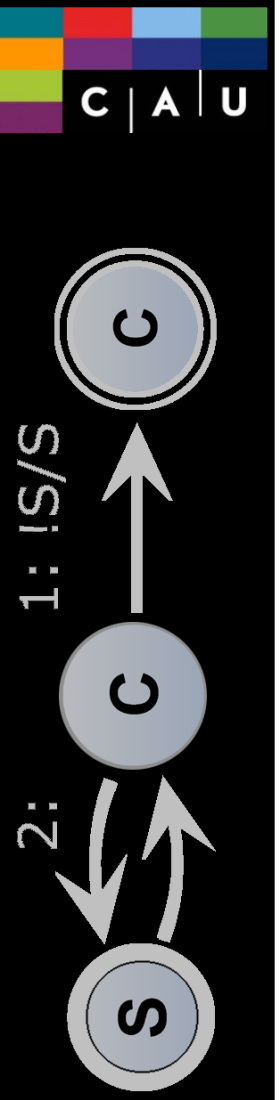
VS.





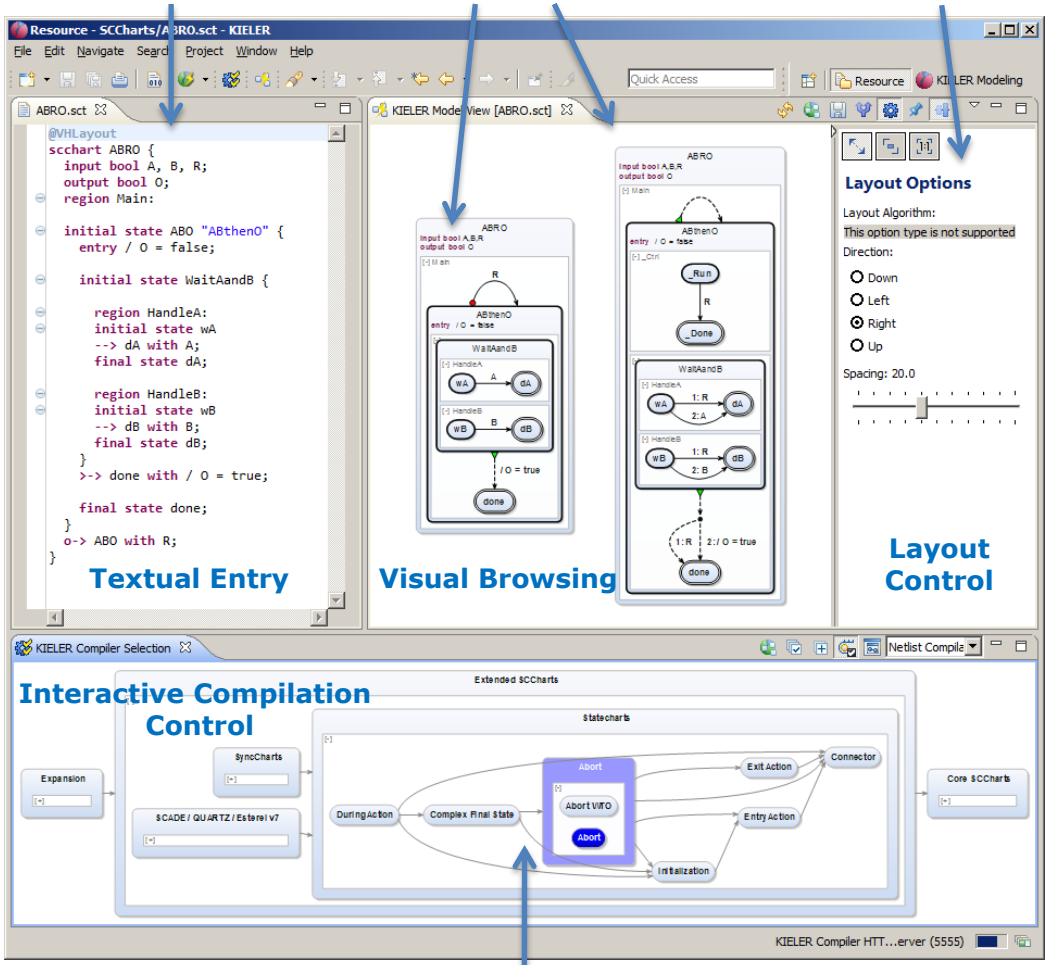
Traditional vs. Interactive SLIC

	Traditional	Interactive SLIC
Understand language feature	-	+
Understand language	-	+
Compare language features	-	+
Compare compilation options	-	+
Fine tuning	-	+
Choose best suited features	-	+
Choose best suited transformations	-	+
Study static feature semantics	-	+
Study dynamic feature semantics	-	+
Understanding the models	-	+
Maintainability	-	+
Selective validation	-	+
Isolated error fixes	-	+
Extending the language/compiler	-	+
Performance	+	+/-

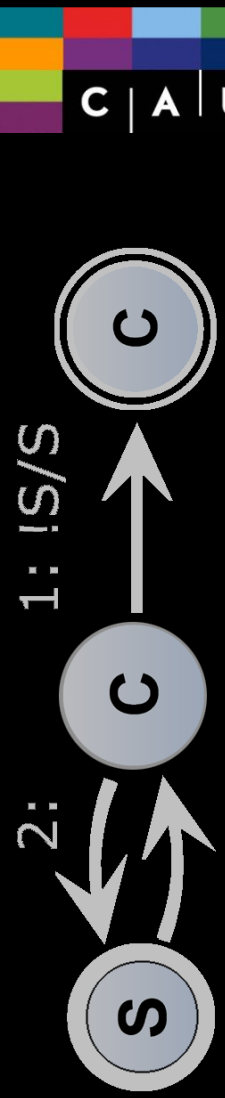


SCCharts Modeling User Story

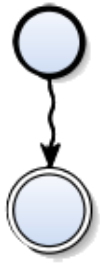
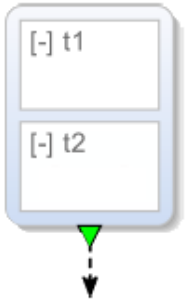
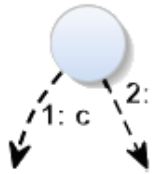
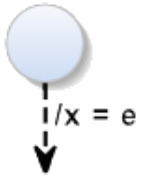
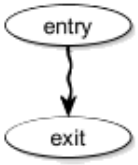
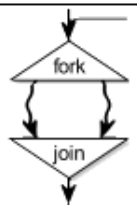
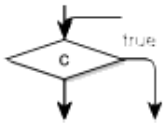
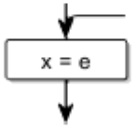
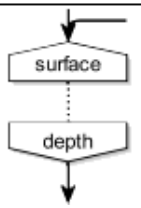
1. Edit SCT code3. Inspect original + transformed SCChart 4. Adjust layout

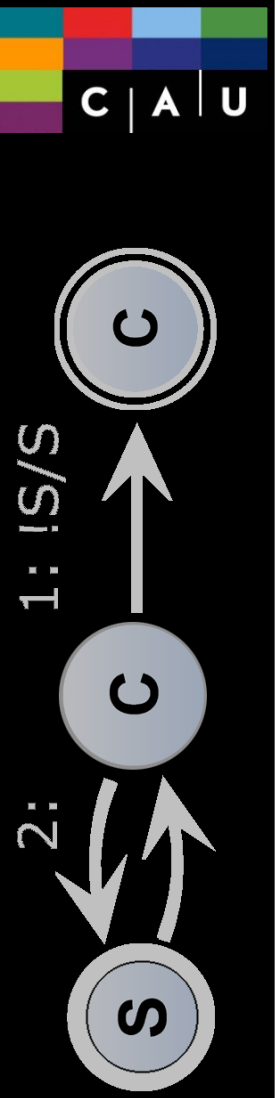


2. Select transformations

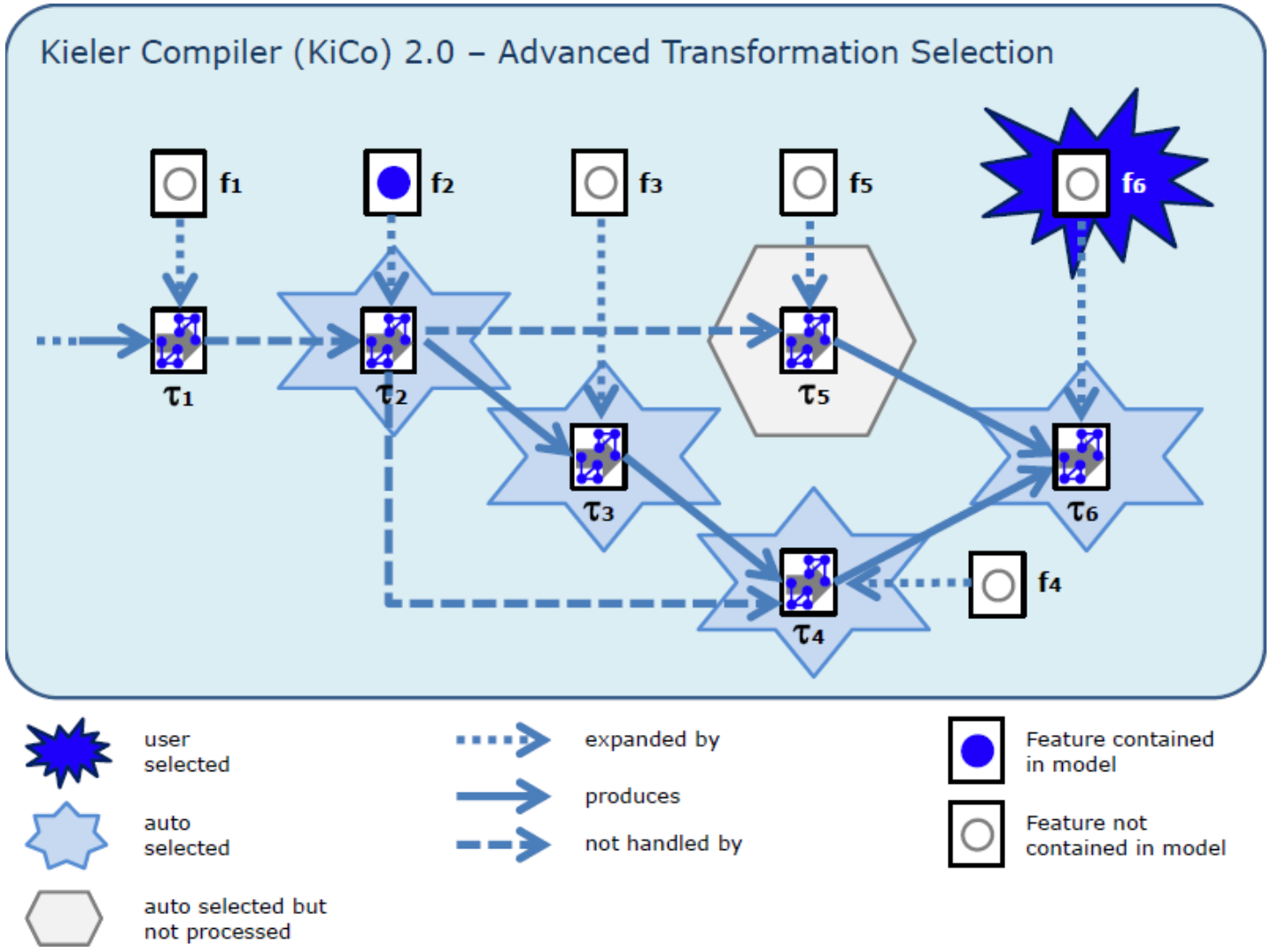


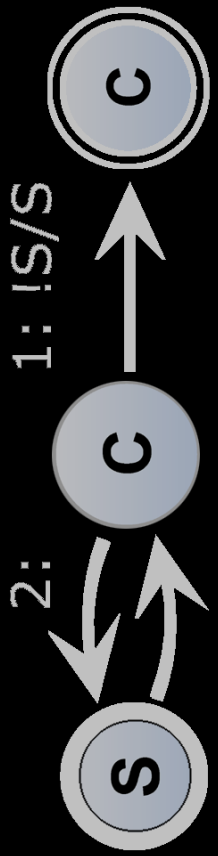
Mapping SCCharts to SCG

	Region (Thread)	Superstate (Parallel)	Trigger (Conditional)	Action (Assignment)	State (Delay)
Normalized SCCharts					
SCG					

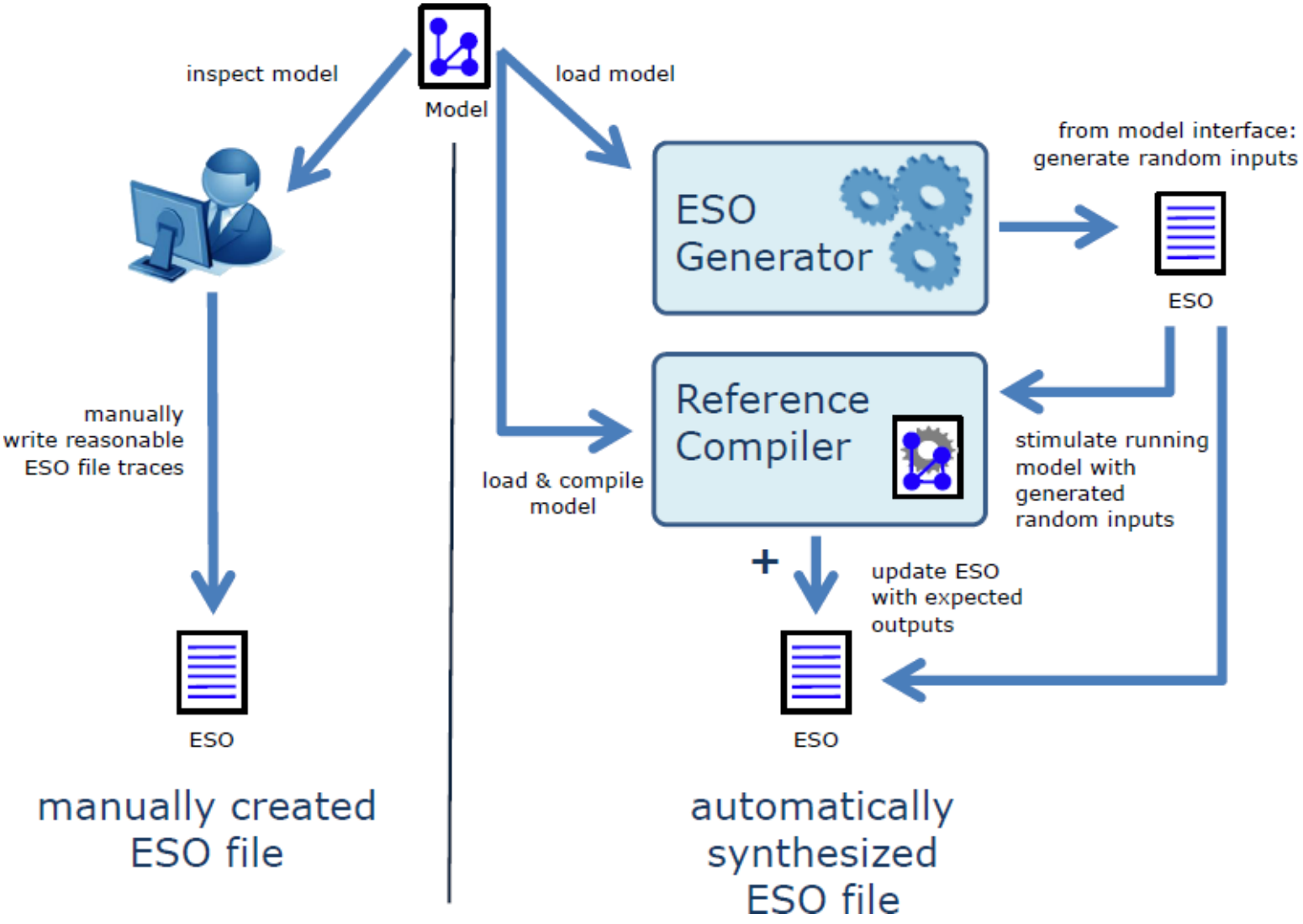


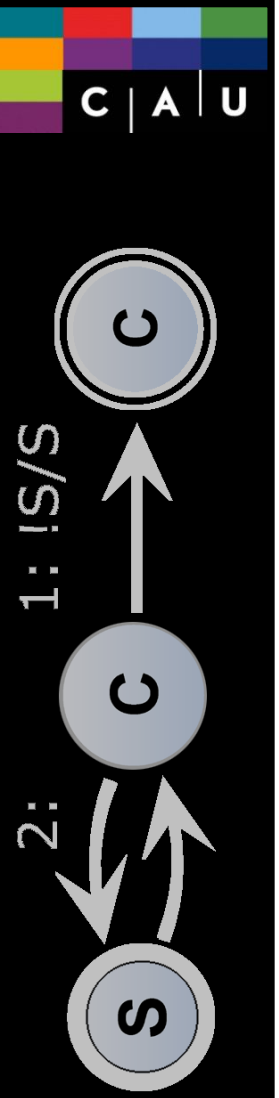
KiCo Selection Algorithm



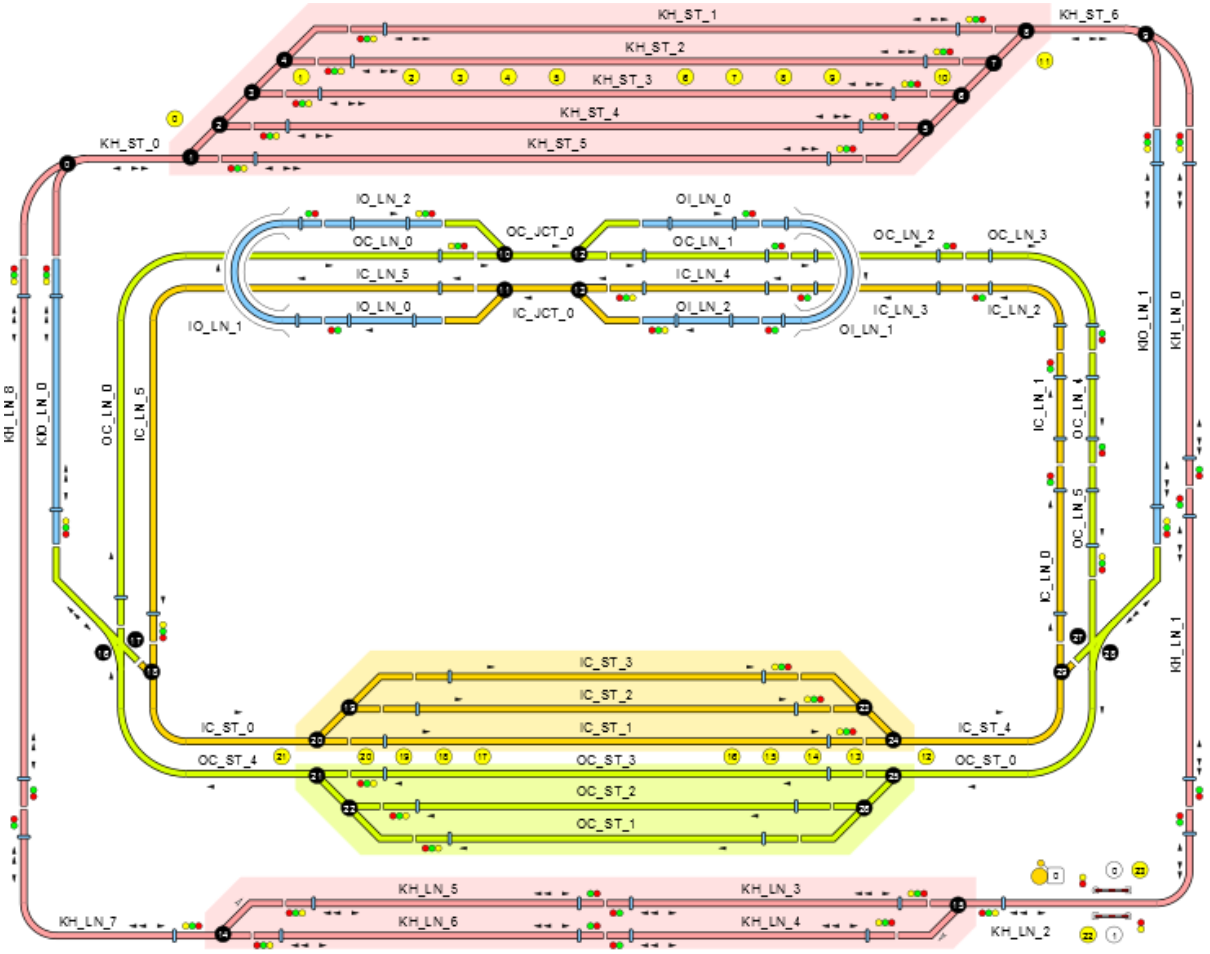


Test File Creation (ESO)





Model Railway Track Scheme



Track segments & Directions

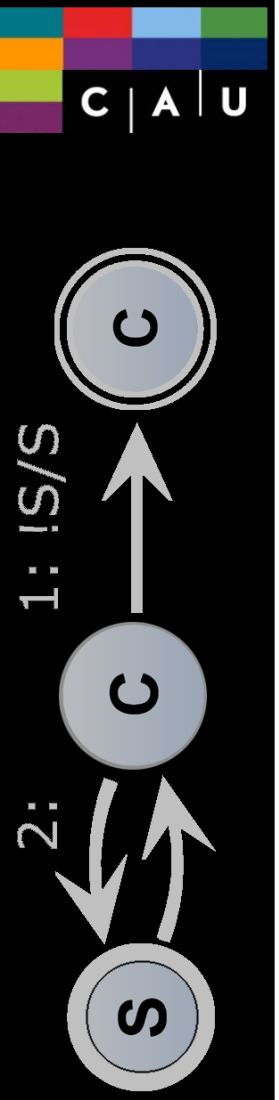
- Inner Circle
- Outer Circle
- Kicking Horse Pass
- Interconnections
- Unidirectional block
- Bidirectional block with forward direction
- Preferred direction

Electronics

- Block isolation
- Reed contact
- Block signal
- Lighting
- Point operating unit

Track specialties

- Bridge
- Point or crossing
- Railroad crossing



SCCharts Meta Model