Synchronous Languages—Lecture 11

Prof. Dr. Reinhard von Hanxleden

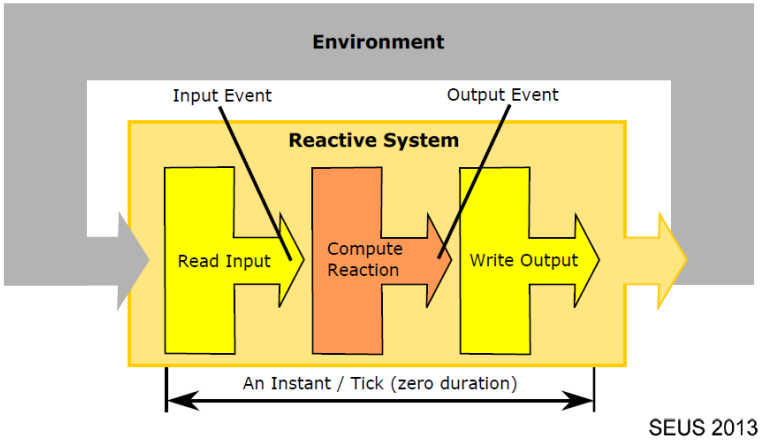
Christian-Albrechts Universität Kiel
Department of Computer Science
Real-Time Systems and Embedded Systems Group

28 May 2020

Last compiled: May 27, 2020, 10:55 hrs

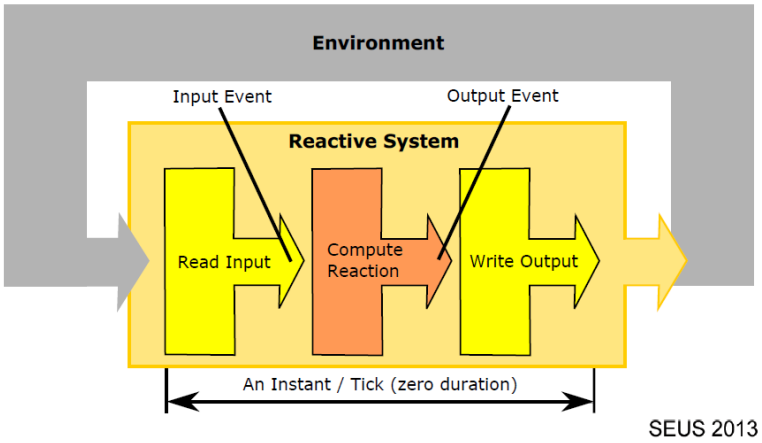
SCCharts — Sequentially Constructive Statecharts for Safety-Critical Applications

Reactive Embedded Systems



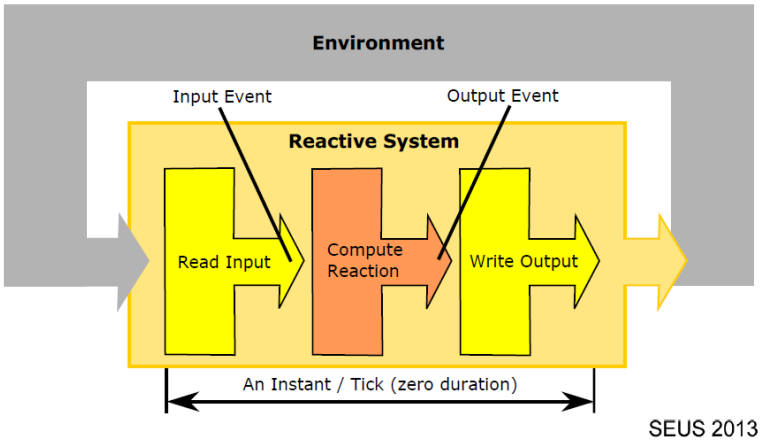
- Embedded systems react to inputs with computed outputs

Reactive Embedded Systems



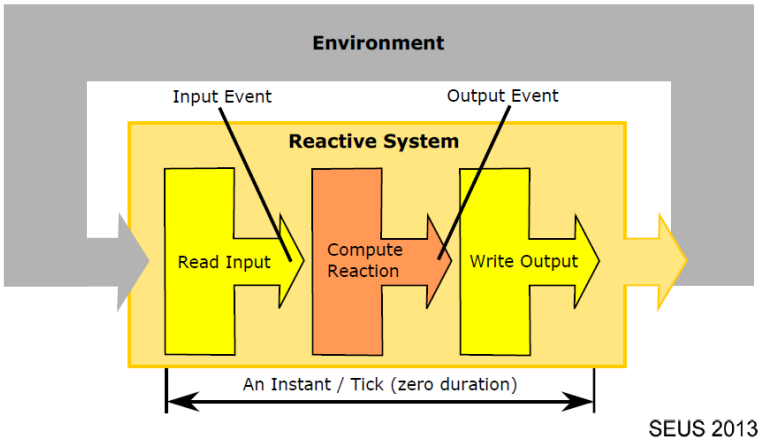
- Embedded systems react to inputs with computed outputs
- Typically state based computations

Reactive Embedded Systems



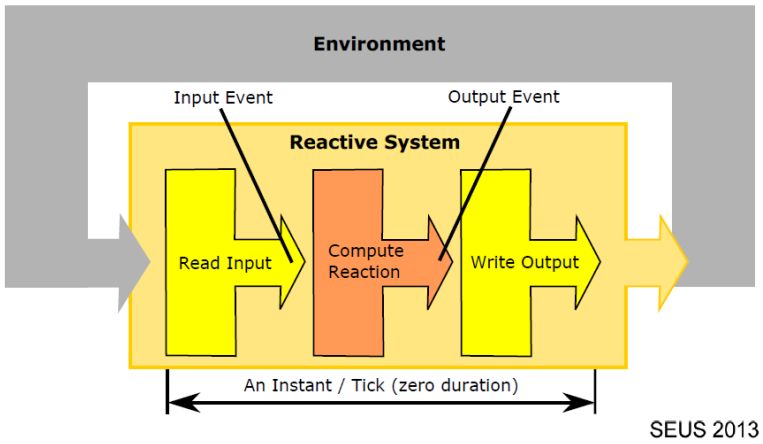
- Embedded systems react to inputs with computed outputs
- Typically state based computations
- Computations often exploit concurrency

Reactive Embedded Systems



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

Reactive Embedded Systems

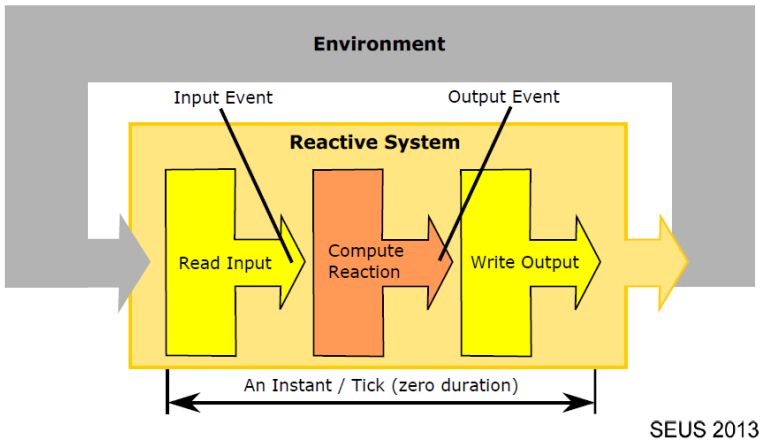


```
public class ValueHolder {
   private List listeners = new LinkedList();
   private int value;
   public interface Listener {
      public void valueChanged(int newValue);
   }
   public void addListener(Listener listener) {
      listeners.add(listener);
   }
   public void setValue(int newValue) {
      value = newValue;
      Iterator i = listeners.iterator();
      while(i.hasNext()) {
         ((Listener)i.next()).valueChanged(newValue);
      }
   }
}
```

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

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

Reactive Embedded Systems



```
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);
      }
   }
}
```

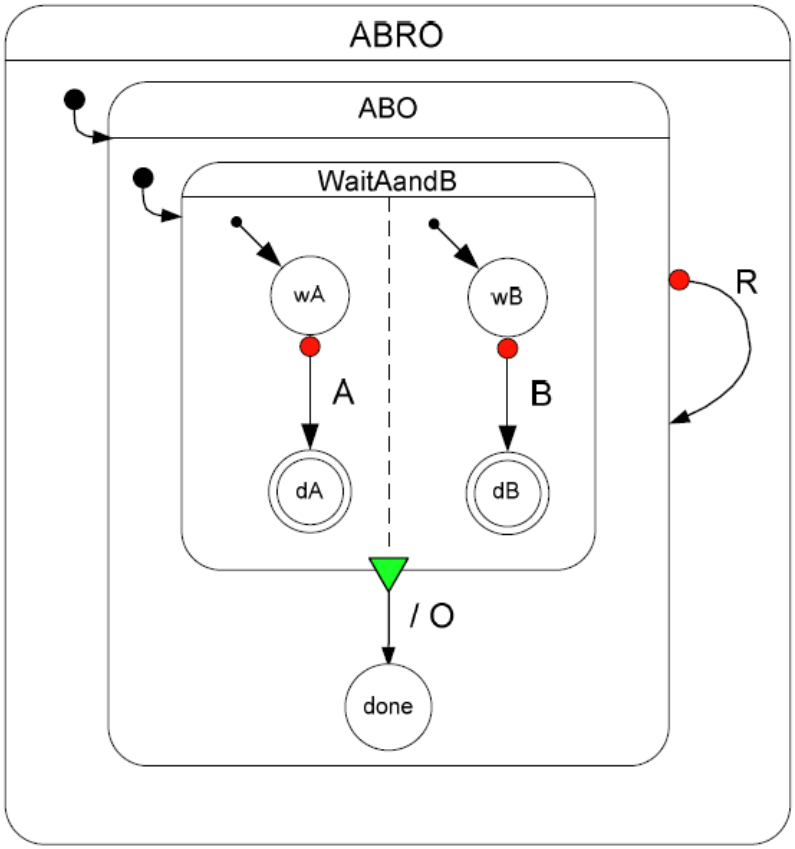
E. A. Lee, The Problem with Threads, 2006

- Embedded systems react to inputs with computed outputs
- Typically state based computations
- Computations often exploit concurrency → Threads
- Threads are problematic → **Synchronous languages**: Lustre, Esterel, SCADE, SyncCharts

SyncCharts

- Statechart dialect for specifying deterministic & robust concurrency

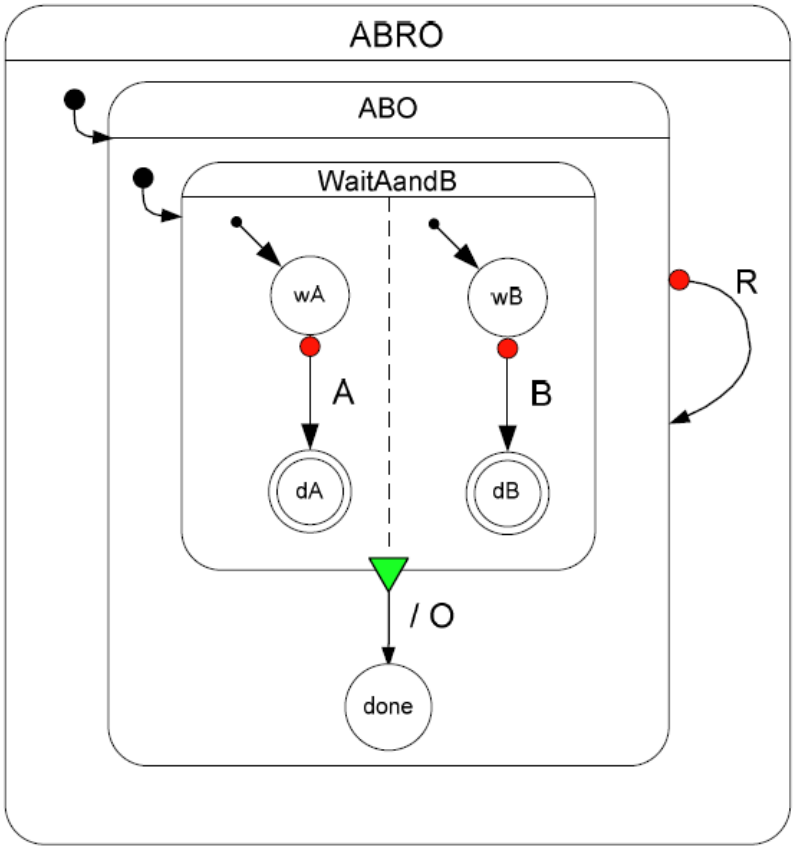
SyncCharts



[Charles André, Semantics of SyncCharts, 2003]

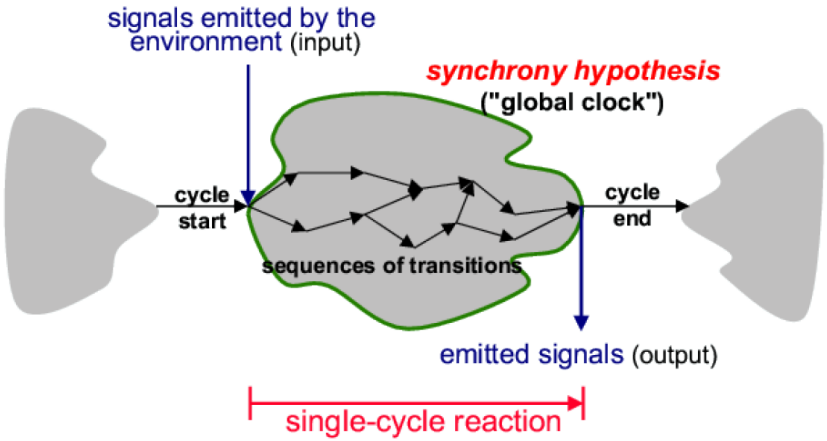
- Statechart dialect for specifying deterministic & robust concurrency
- SyncCharts:
  - Hierarchy, Concurrency, Broadcast

SyncCharts



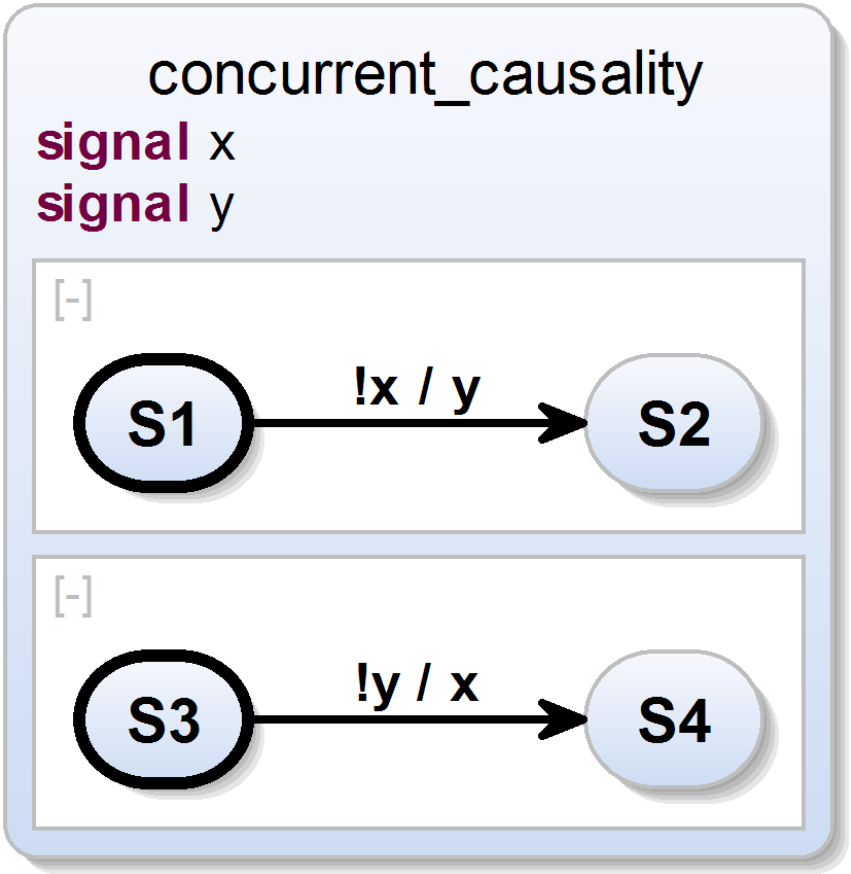
[Charles André, Semantics of SyncCharts, 2003]

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

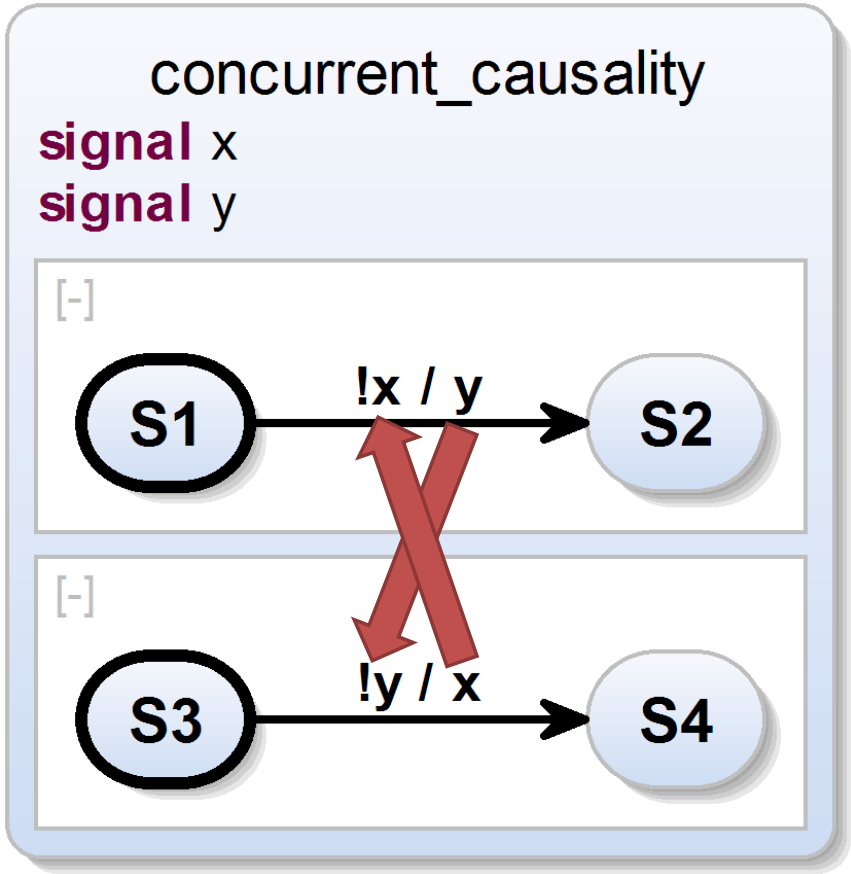


[Gerald Lüttgen, 2001]

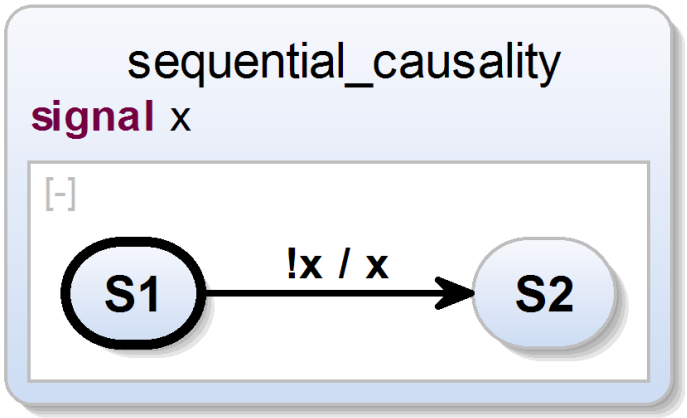
Causality in SyncCharts



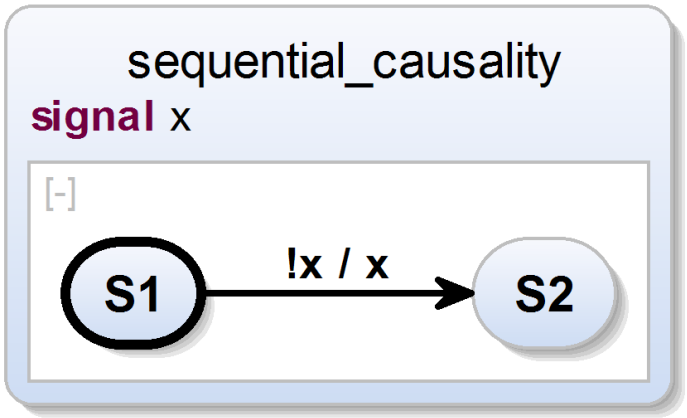
Causality in SyncCharts



Causality in SyncCharts (cont'd)

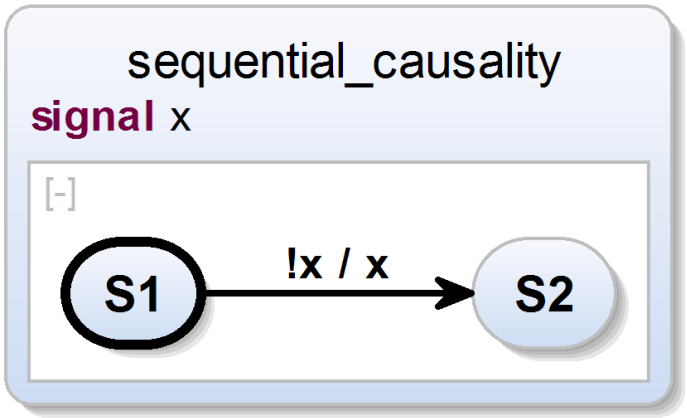


Causality in SyncCharts (cont'd)




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

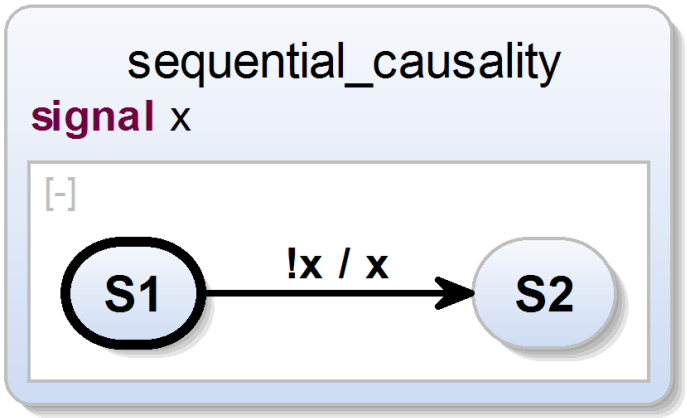
Causality in SyncCharts (cont'd)




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

- Rejected by SyncCharts compiler

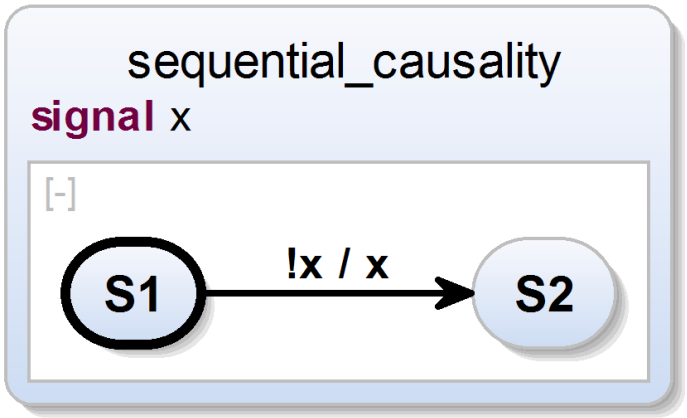
Causality in SyncCharts (cont'd)




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

- Rejected by SyncCharts compiler
- *Signal Coherence Rule*

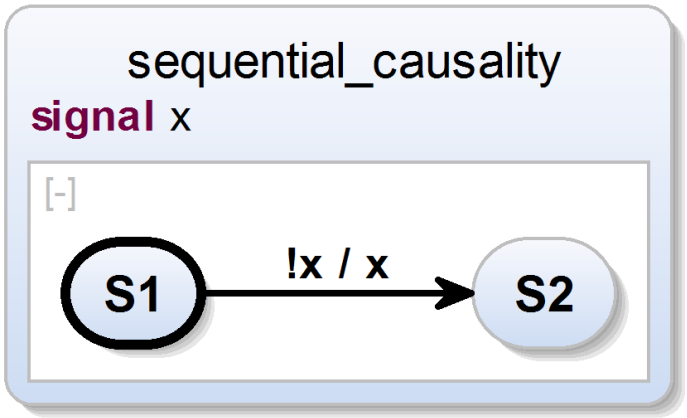
Causality in SyncCharts (cont'd)




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

- Rejected by SyncCharts compiler
- *Signal Coherence Rule*
- May seem awkward from SyncCharts perspective

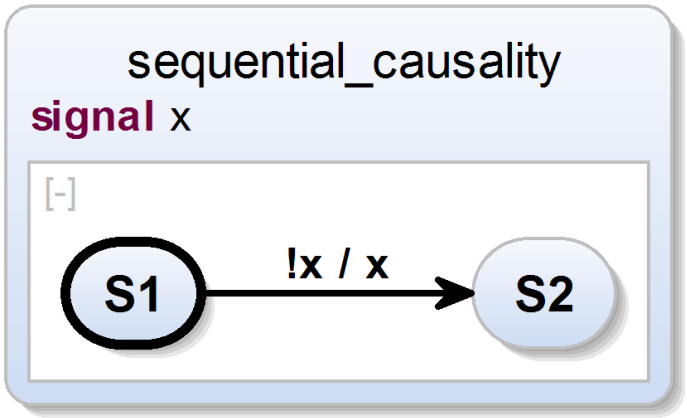
Causality in SyncCharts (cont'd)




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

- Rejected by SyncCharts compiler
- *Signal Coherence Rule*
- May seem awkward from SyncCharts perspective, but common paradigm

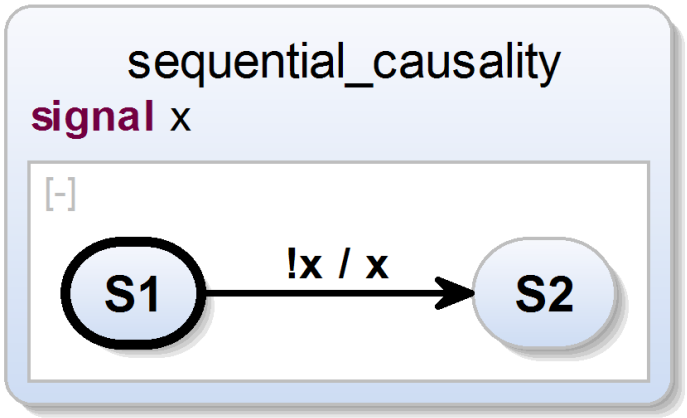
Causality in SyncCharts (cont'd)




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

- Rejected by SyncCharts compiler
- *Signal Coherence Rule*
- May seem awkward from SyncCharts perspective, but common paradigm
- Deterministic sequential execution possible

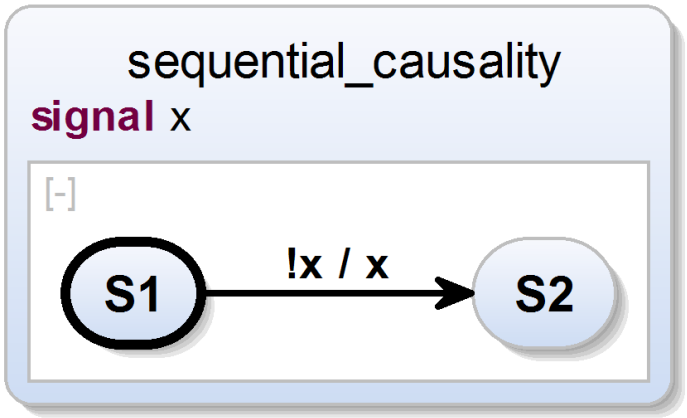
Causality in SyncCharts (cont'd)




```
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*

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)**

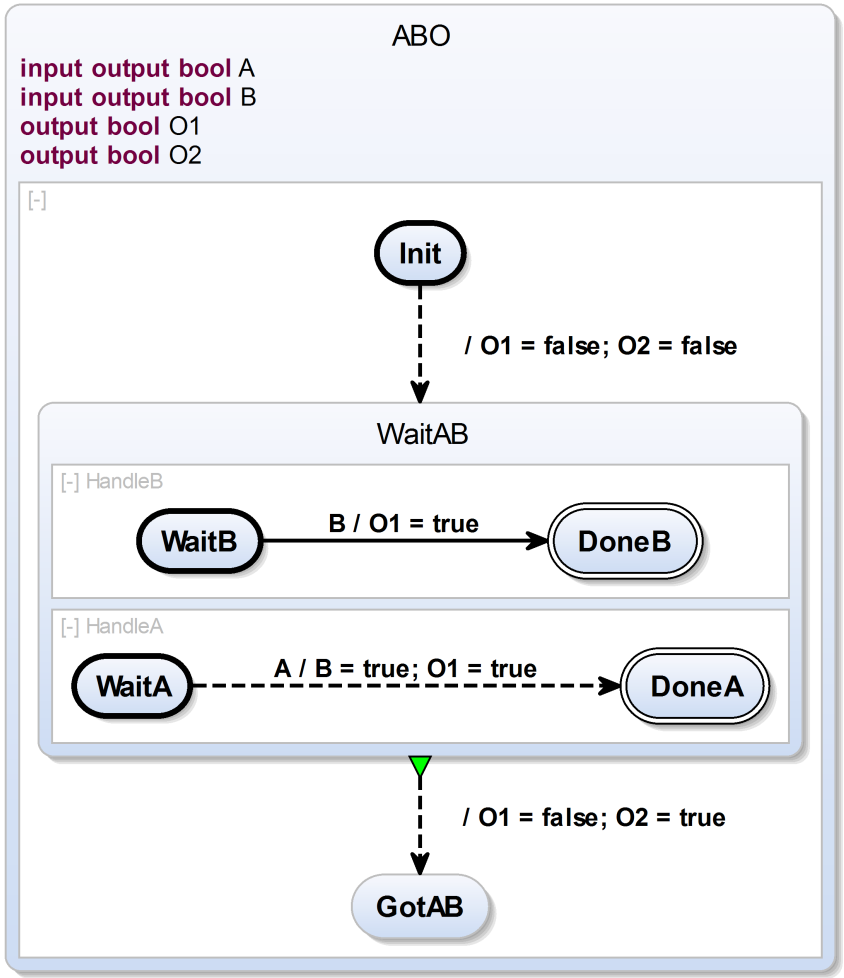
Overview

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

SCCharts Overview

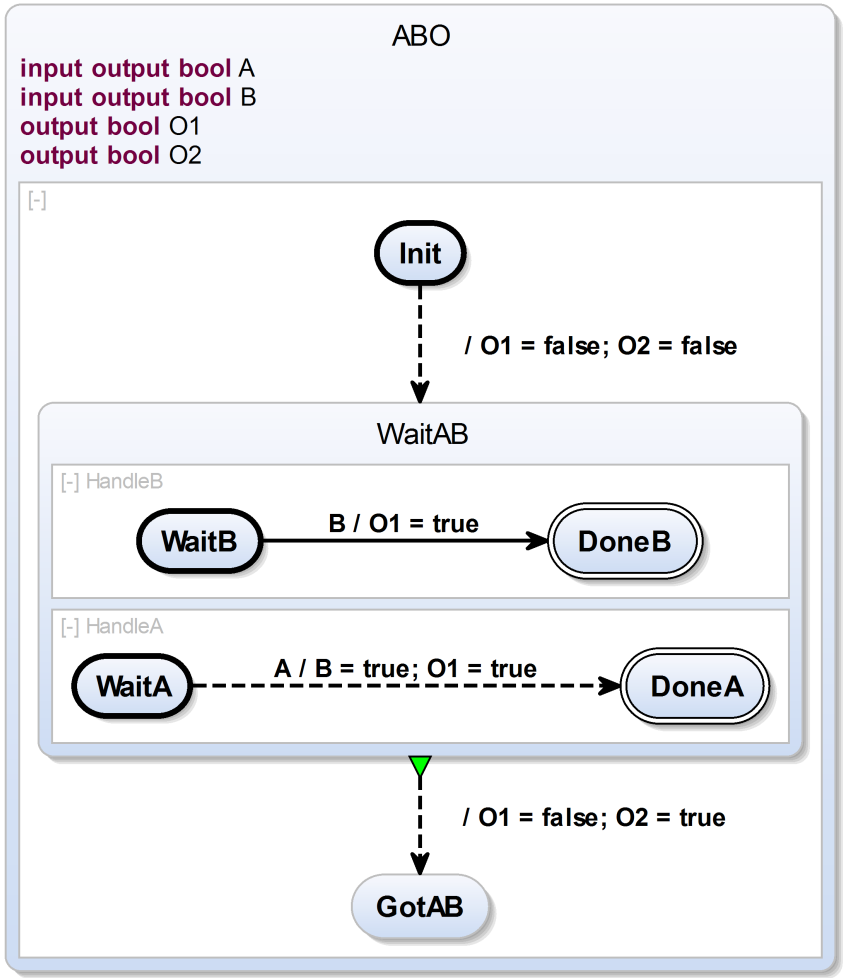
- SCCharts $\widehat{=}$
  SyncCharts syntax +
  Seqentially Constructive semantics

SCCharts Overview



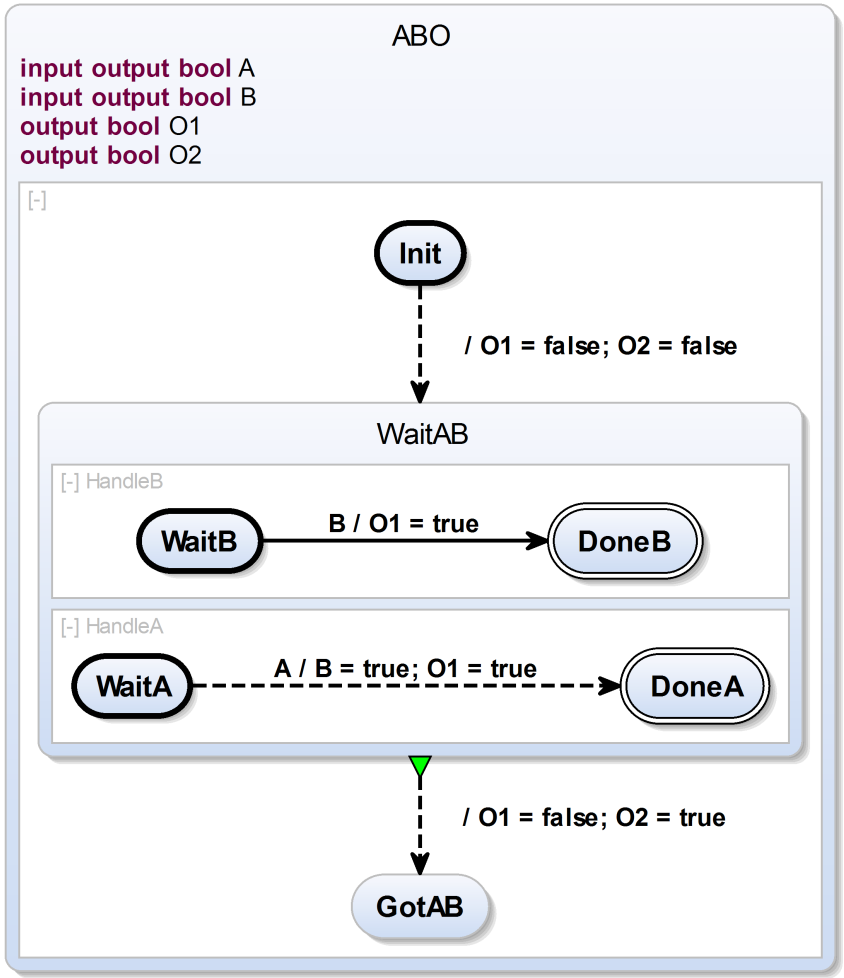
- SCCharts $\widehat{=}$
  SyncCharts syntax +
  Seqentially Constructive semantics
- *Hello World* of Sequential Constructiveness: **ABO**

SCCharts Overview



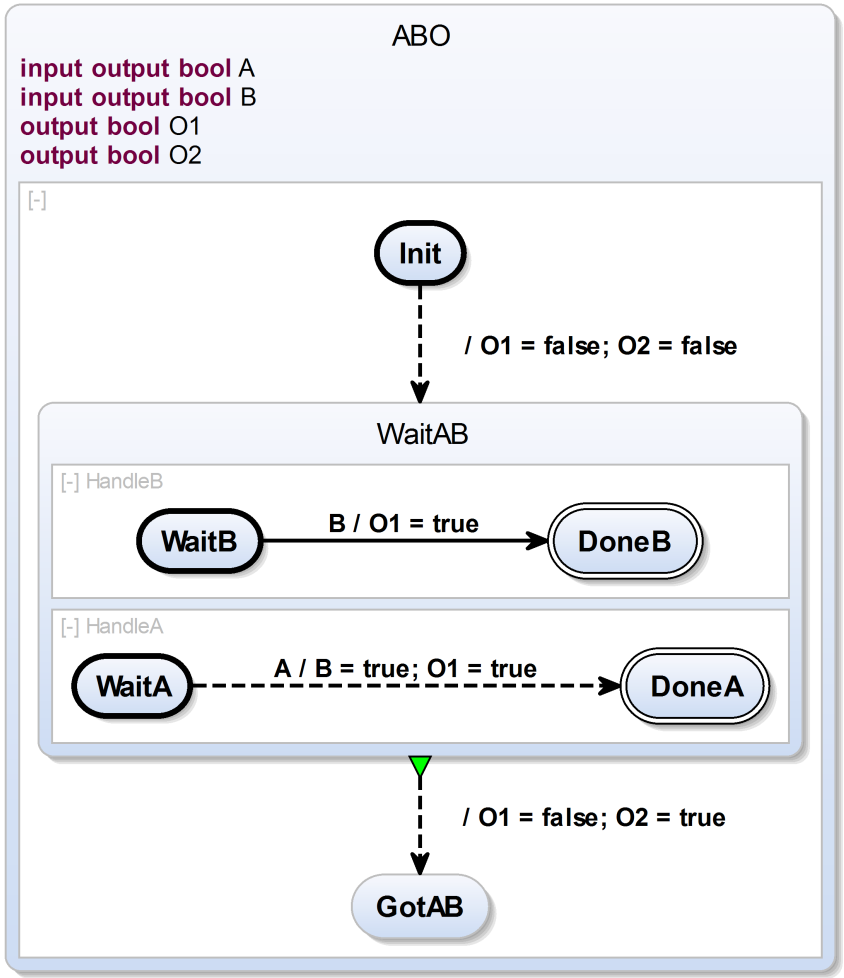
- SCCharts $\widehat{=}$
  SyncCharts syntax +
  Seqentially Constructive semantics
- *Hello World* of Sequential Constructiveness: **ABO**
  - Variables instead of signals

SCCharts Overview



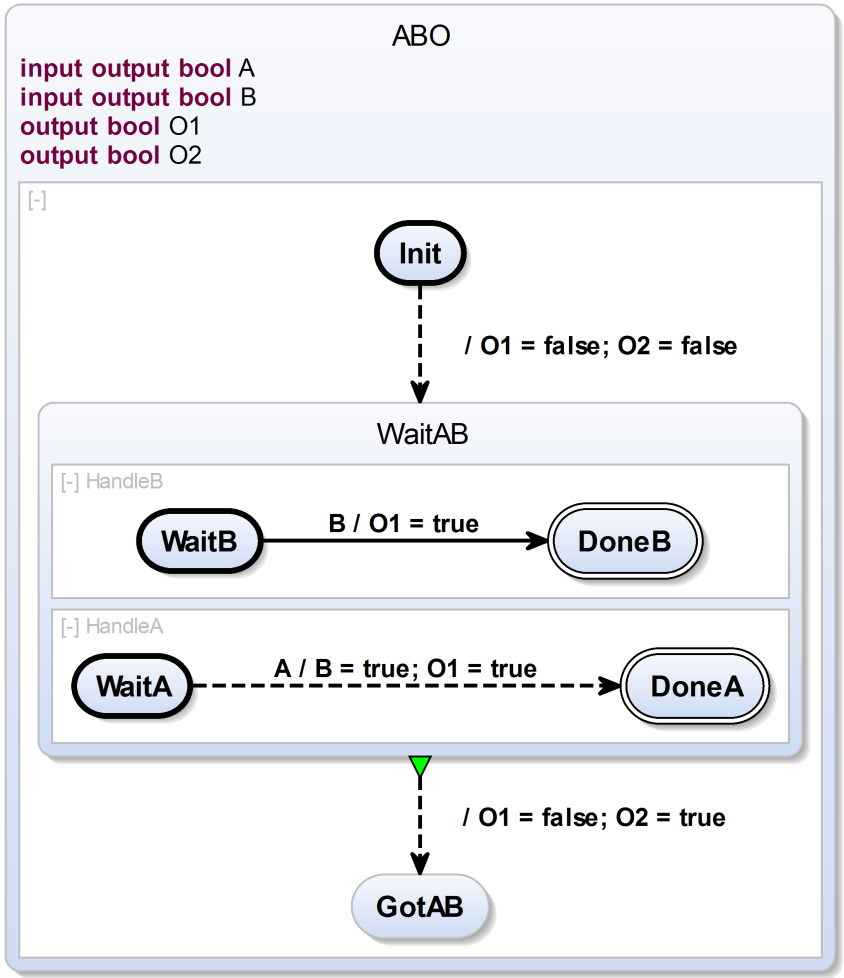
- SCCharts $\widehat{=}$
  SyncCharts syntax +
  Seqentially Constructive semantics
- *Hello World* of Sequential Constructiveness: **ABO**
  - Variables instead of signals
  - Behavior (briefly)

SCCharts Overview



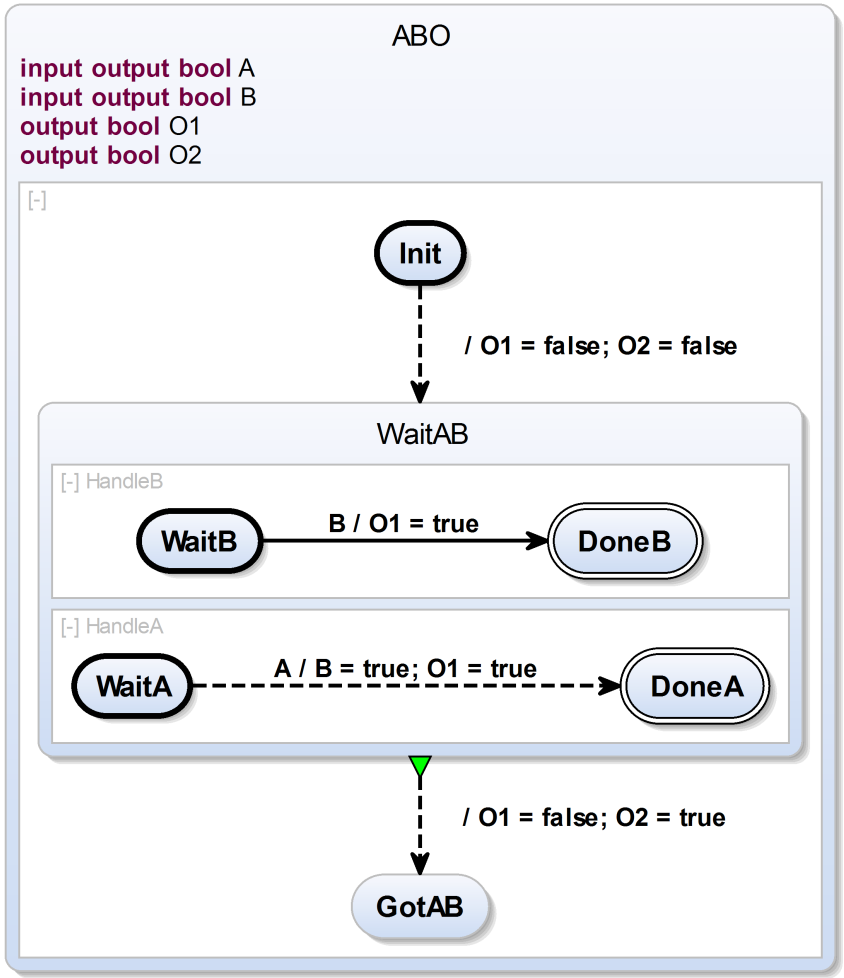
- SCCharts $\widehat{=}$
  SyncCharts syntax +
  Seqentially Constructive semantics
- *Hello World* of Sequential Constructiveness: **ABO**
  - Variables instead of signals
  - Behavior (briefly)
    1. Initialize

SCCharts Overview



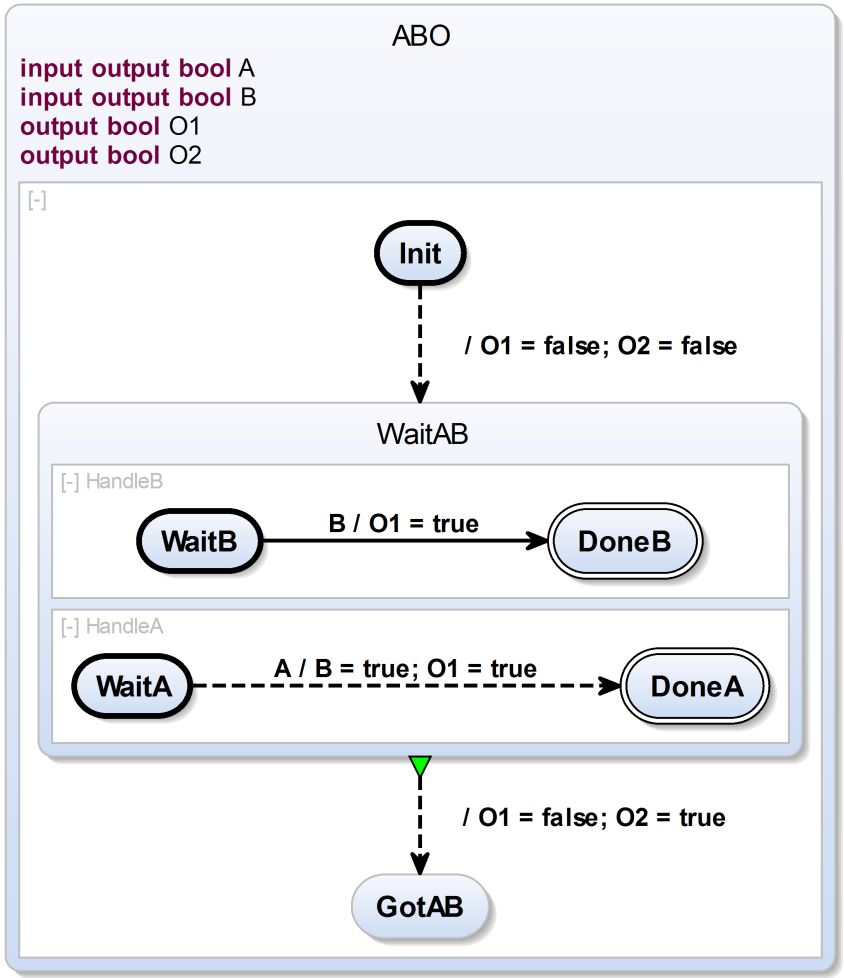
- SCCharts $\widehat{=}$
  SyncCharts syntax +
  Seqentially Constructive semantics
- *Hello World* of Sequential Constructiveness: **ABO**
  - Variables instead of signals
  - Behavior (briefly)
    1. Initialize
    2. Concurrently wait for inputs A or B to become *true*

SCCharts Overview



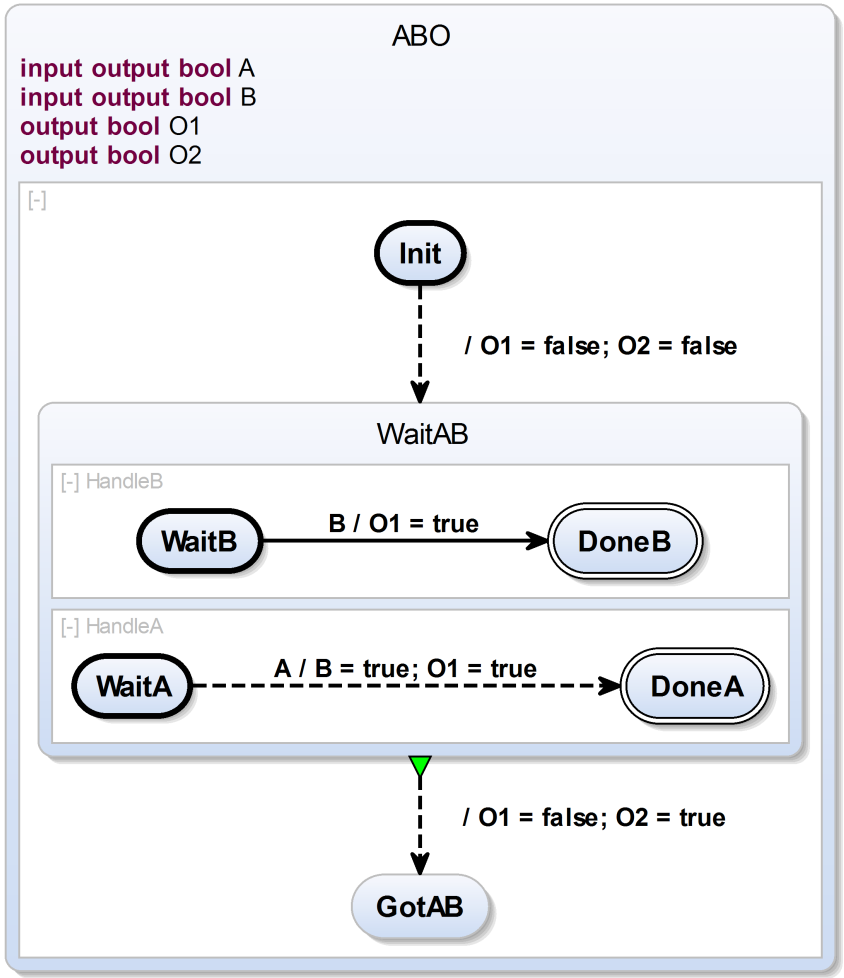
- SCCharts $\widehat{=}$
  SyncCharts syntax +
  Seqentially Constructive semantics
- *Hello World* of Sequential Constructiveness: **ABO**
  - Variables instead of signals
  - Behavior (briefly)
    1. Initialize
    2. Concurrently wait for inputs A or B to become *true*
    3. Once A and B are true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

SCCharts Overview



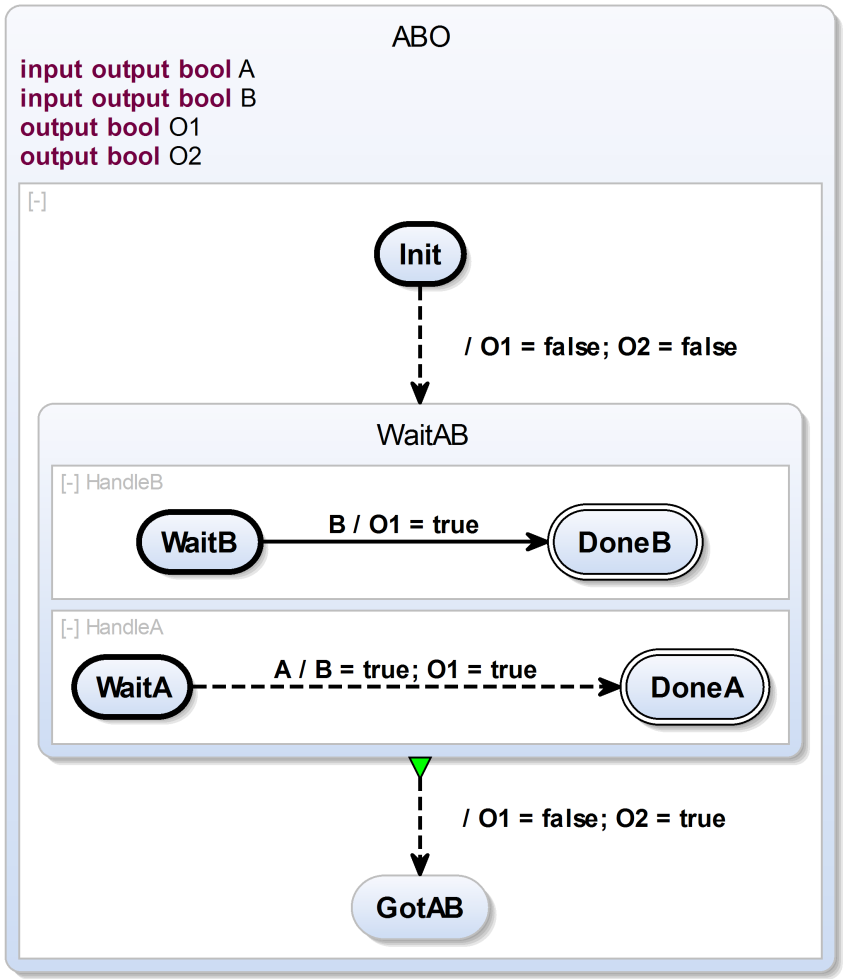
- SCCharts $\widehat{=}$
  SyncCharts syntax +
  Seqentially Constructive semantics
- *Hello World* of Sequential Constructiveness: **ABO**
  - Variables instead of signals
  - Behavior (briefly)
    1. Initialize
    2. Concurrently wait for inputs A or B to become *true*
    3. Once A and B are true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

SCCharts Overview

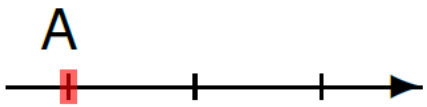


- SCCharts $\widehat{=}$
  SyncCharts syntax +
  Seqentially Constructive semantics
- *Hello World* of Sequential Constructiveness: **ABO**
  - Variables instead of signals
  - Behavior (briefly)
    1. Initialize
    2. Concurrently wait for inputs A or B to become *true*
    3. Once A and B are true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

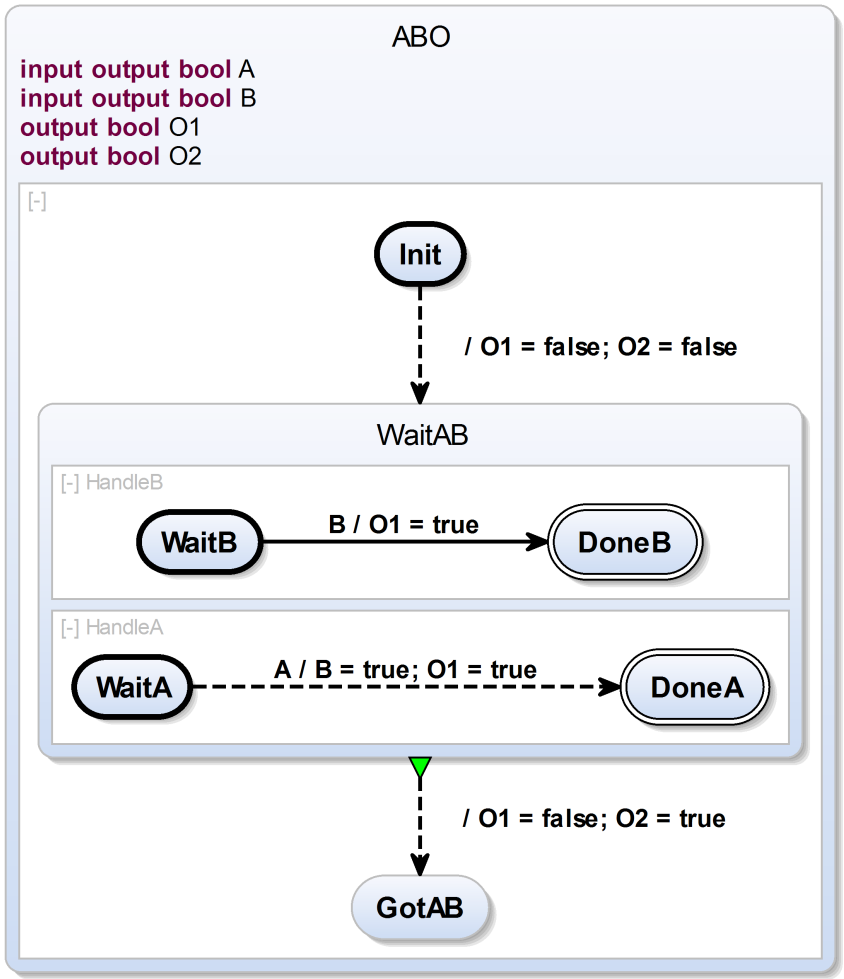
SCCharts Overview



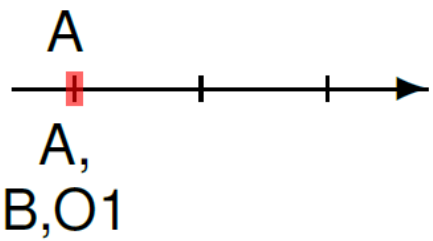
- SCCharts $\widehat{=}$
  SyncCharts syntax +
  Seqentially Constructive semantics
- *Hello World* of Sequential Constructiveness: **ABO**
  - Variables instead of signals
  - Behavior (briefly)
    1. Initialize
    2. Concurrently wait for inputs A or B to become *true*
    3. Once A and B are true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



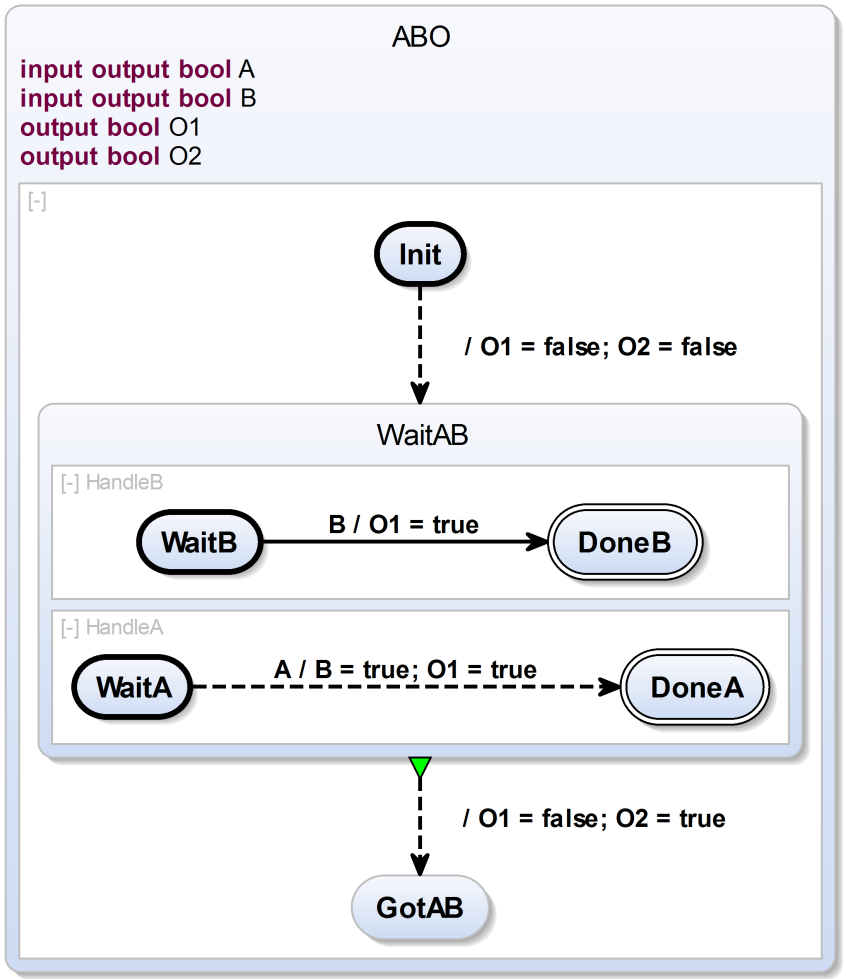
SCCharts Overview



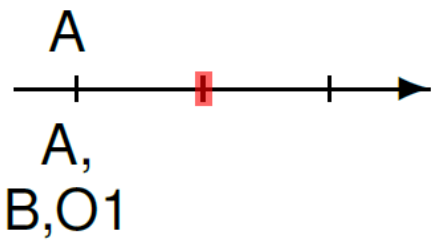
- SCCharts $\widehat{=}$
  SyncCharts syntax +
  Seqentially Constructive semantics
- *Hello World* of Sequential Constructiveness: **ABO**
  - Variables instead of signals
  - Behavior (briefly)
    1. Initialize
    2. Concurrently wait for inputs A or B to become *true*
    3. Once A and B are true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



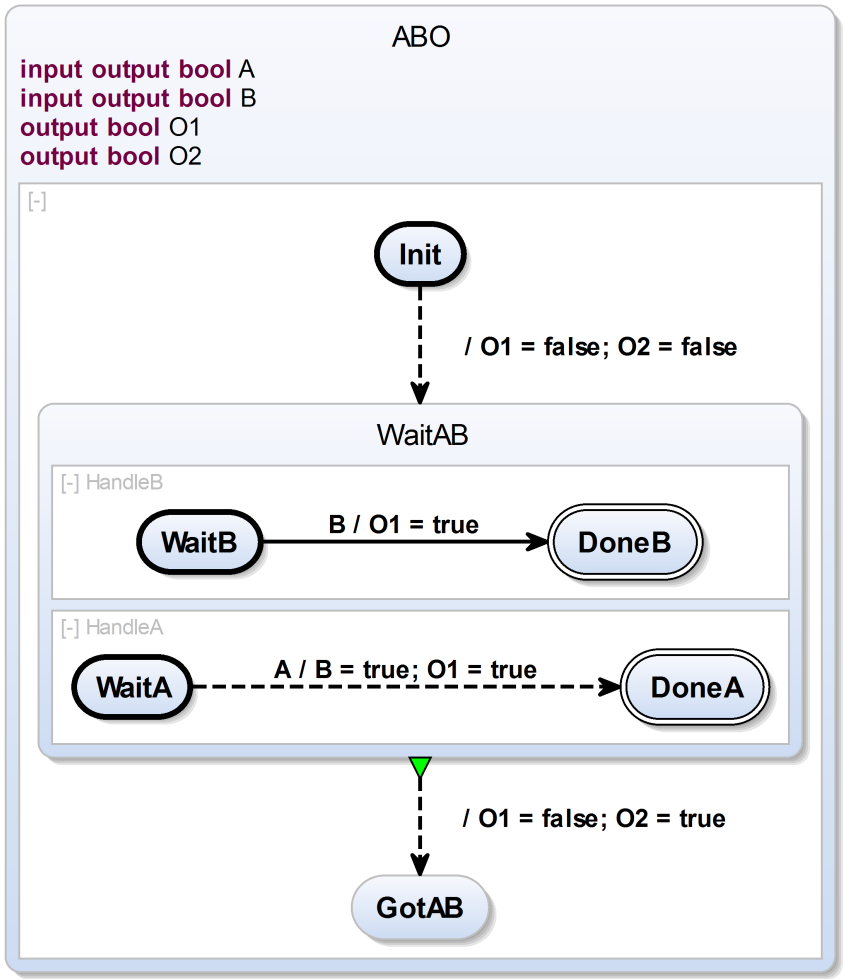
SCCharts Overview



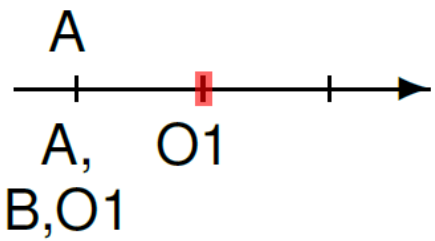
- SCCharts $\widehat{=}$
  SyncCharts syntax +
  Seqentially Constructive semantics
- *Hello World* of Sequential Constructiveness: **ABO**
  - Variables instead of signals
  - Behavior (briefly)
    1. Initialize
    2. Concurrently wait for inputs A or B to become *true*
    3. Once A and B are true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



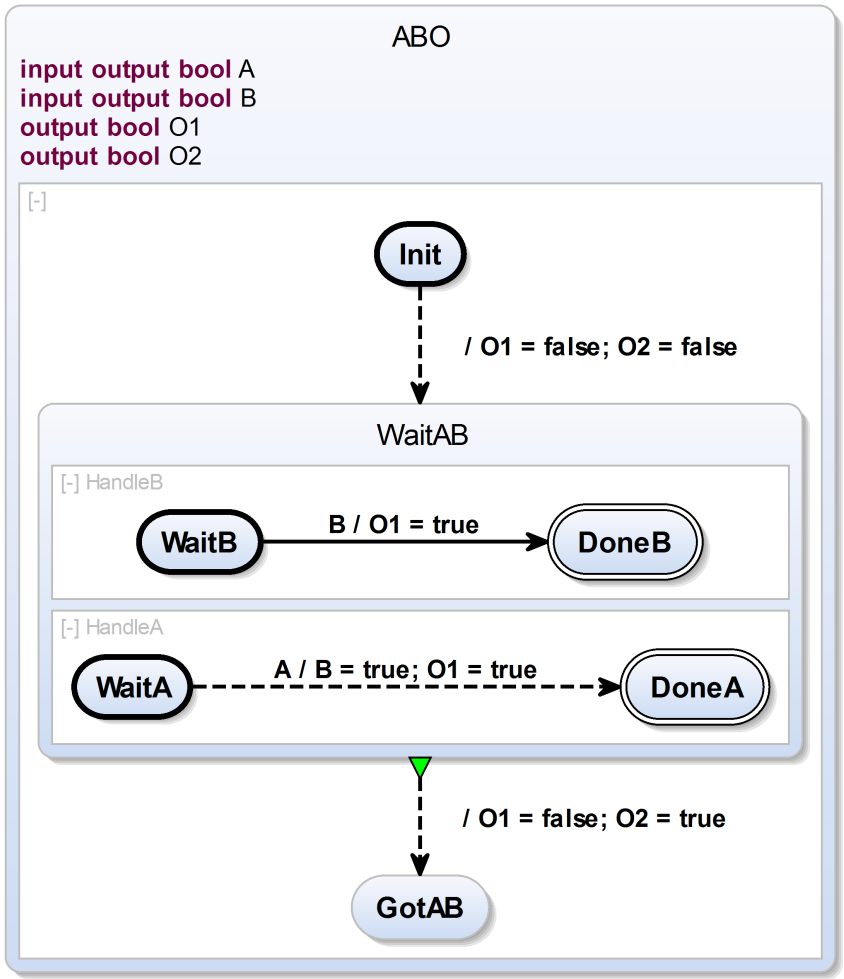
SCCharts Overview



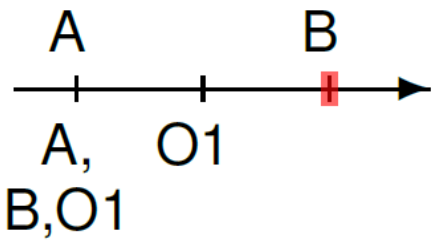
- SCCharts $\widehat{=}$
  SyncCharts syntax +
  Seqentially Constructive semantics
- *Hello World* of Sequential Constructiveness: **ABO**
  - Variables instead of signals
  - Behavior (briefly)
    1. Initialize
    2. Concurrently wait for inputs A or B to become *true*
    3. Once A and B are true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



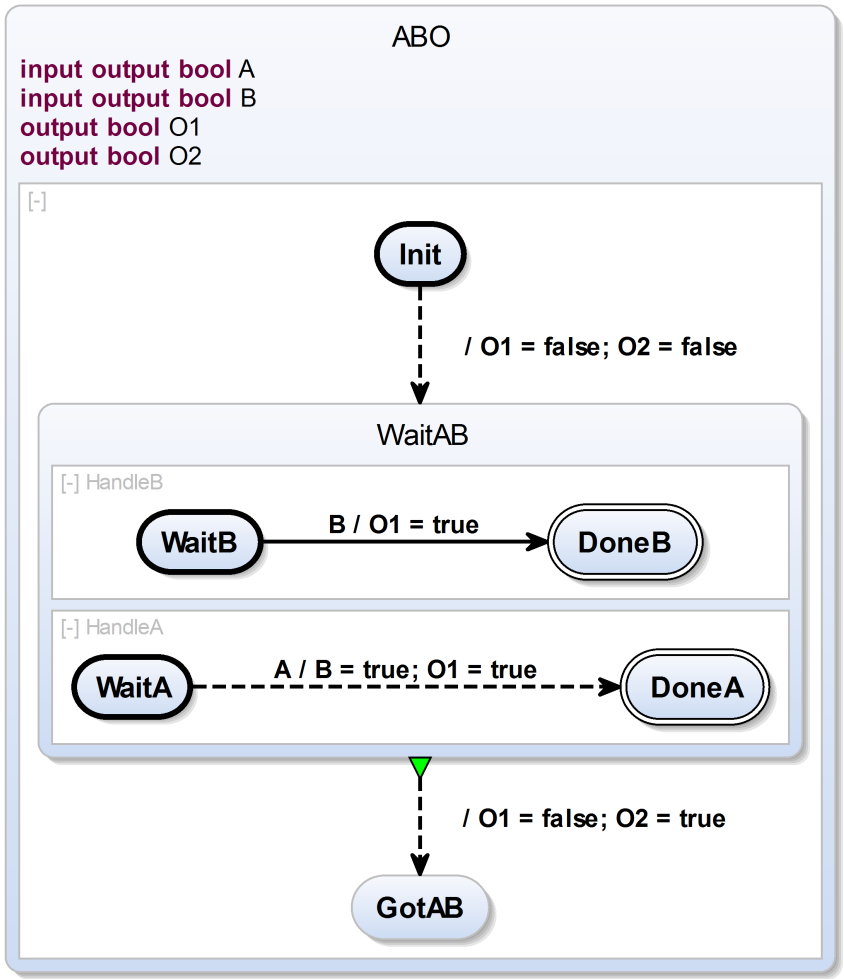
SCCharts Overview



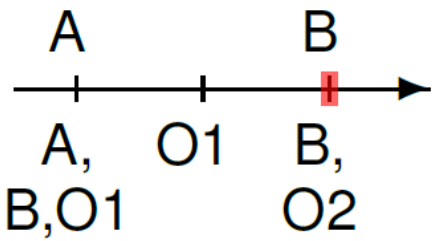
- SCCharts $\widehat{=}$
  SyncCharts syntax +
  Seqentially Constructive semantics
- *Hello World* of Sequential Constructiveness: **ABO**
  - Variables instead of signals
  - Behavior (briefly)
    1. Initialize
    2. Concurrently wait for inputs A or B to become *true*
    3. Once A and B are true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



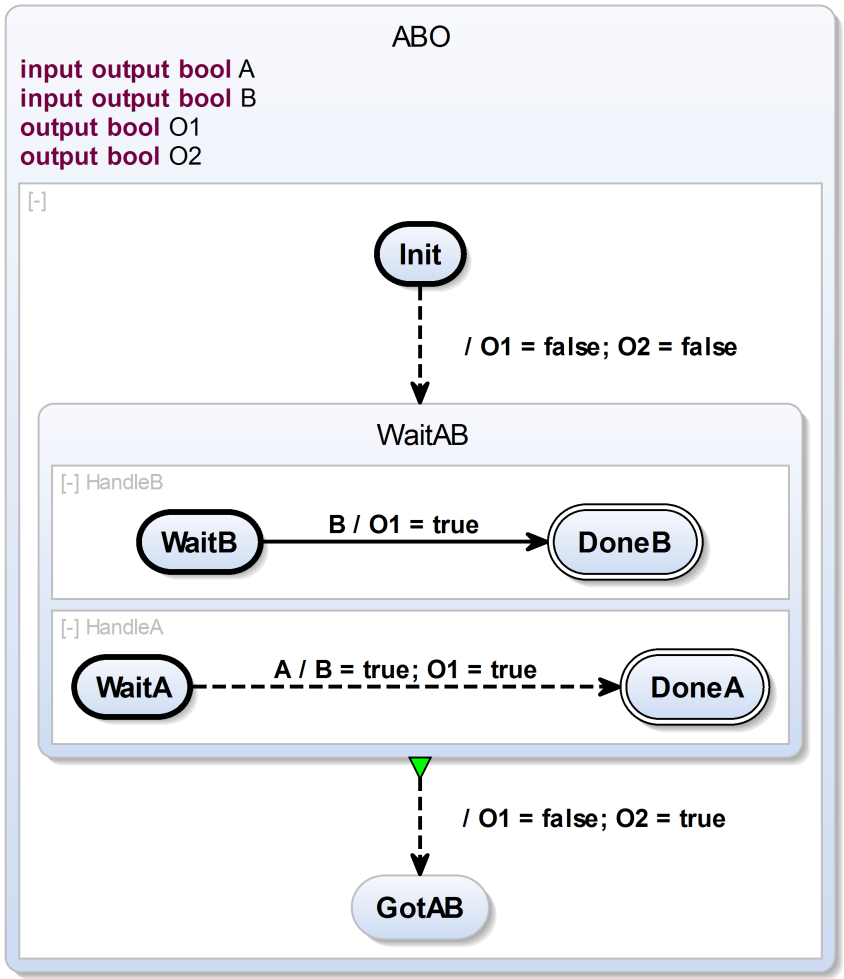
SCCharts Overview



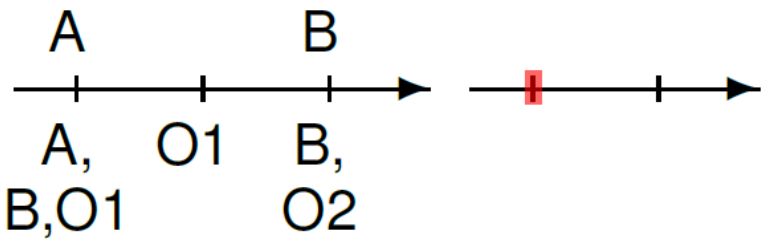
- SCCharts $\widehat{=}$
  SyncCharts syntax +
  Seqentially Constructive semantics
- *Hello World* of Sequential Constructiveness: **ABO**
  - Variables instead of signals
  - Behavior (briefly)
    1. Initialize
    2. Concurrently wait for inputs A or B to become *true*
    3. Once A and B are true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



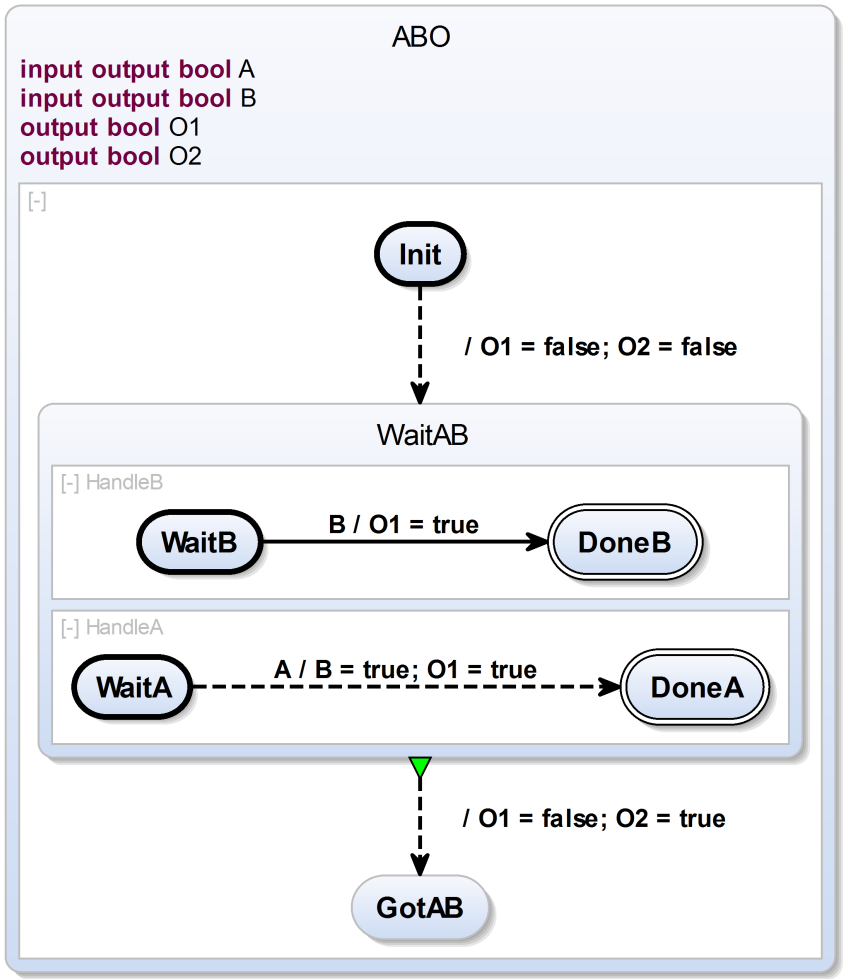
SCCharts Overview



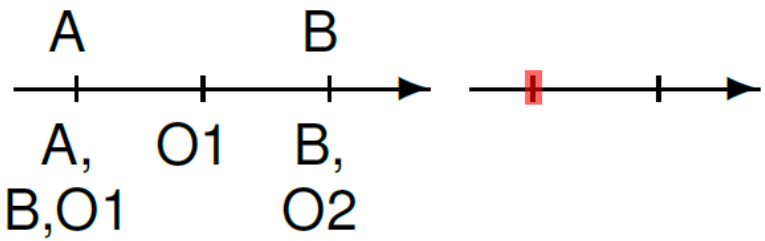
- SCCharts $\widehat{=}$
  SyncCharts syntax +
  Seqentially Constructive semantics
- *Hello World* of Sequential Constructiveness: **ABO**
  - Variables instead of signals
  - Behavior (briefly)
    1. Initialize
    2. Concurrently wait for inputs A or B to become *true*
    3. Once A and B are true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



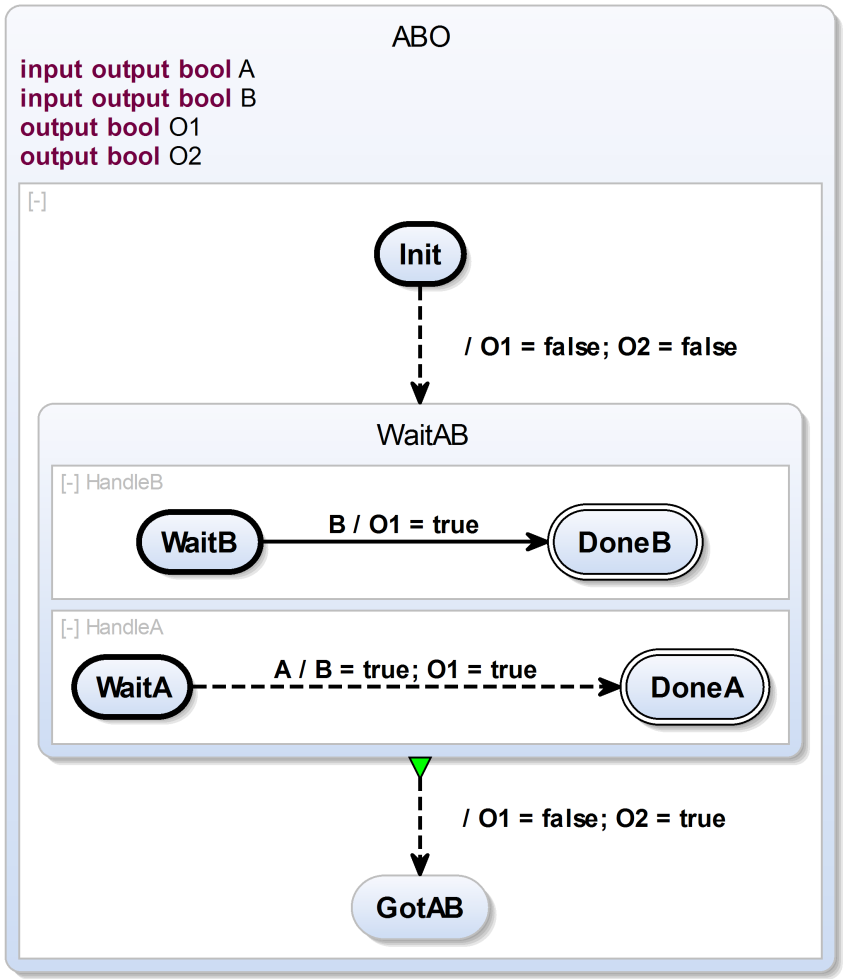
SCCharts Overview



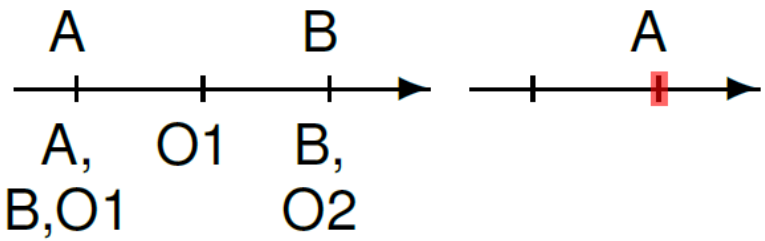
- SCCharts $\widehat{=}$
  SyncCharts syntax +
  Seqentially Constructive semantics
- *Hello World* of Sequential Constructiveness: **ABO**
  - Variables instead of signals
  - Behavior (briefly)
    1. Initialize
    2. Concurrently wait for inputs A or B to become *true*
    3. Once A and B are true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



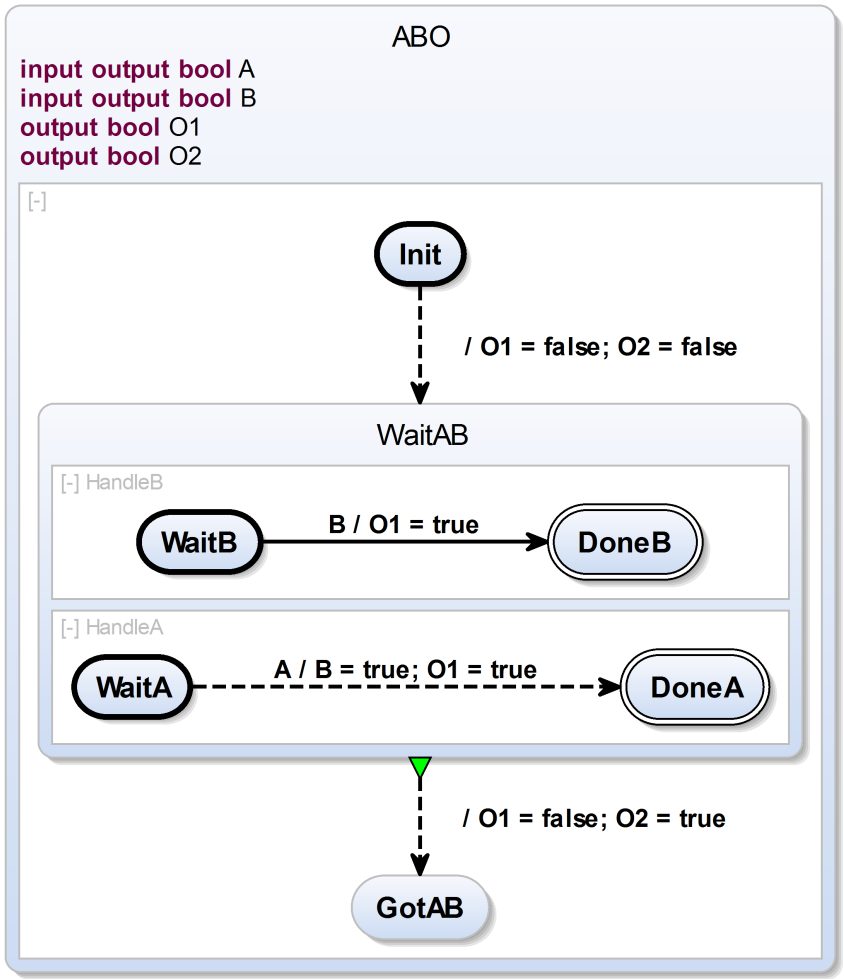
SCCharts Overview



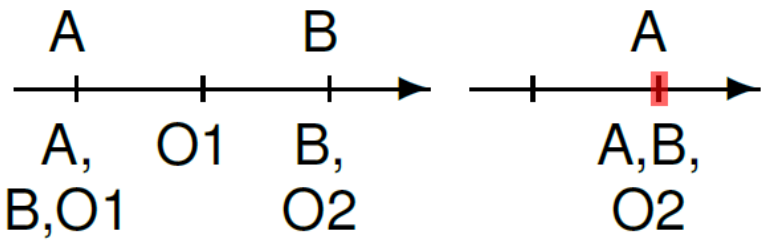
- SCCharts $\widehat{=}$
  SyncCharts syntax +
  Seqentially Constructive semantics
- *Hello World* of Sequential Constructiveness: **ABO**
  - Variables instead of signals
  - Behavior (briefly)
    1. Initialize
    2. Concurrently wait for inputs A or B to become *true*
    3. Once A and B are true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



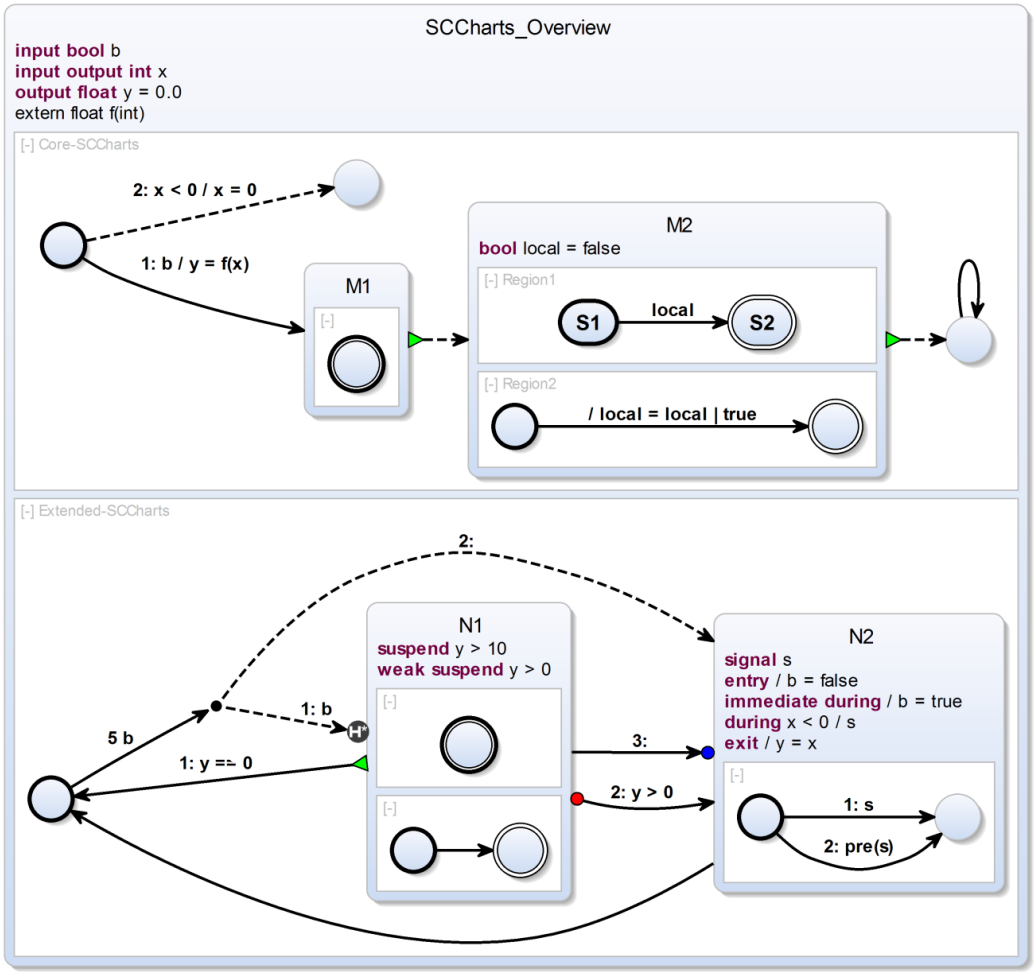
SCCharts Overview

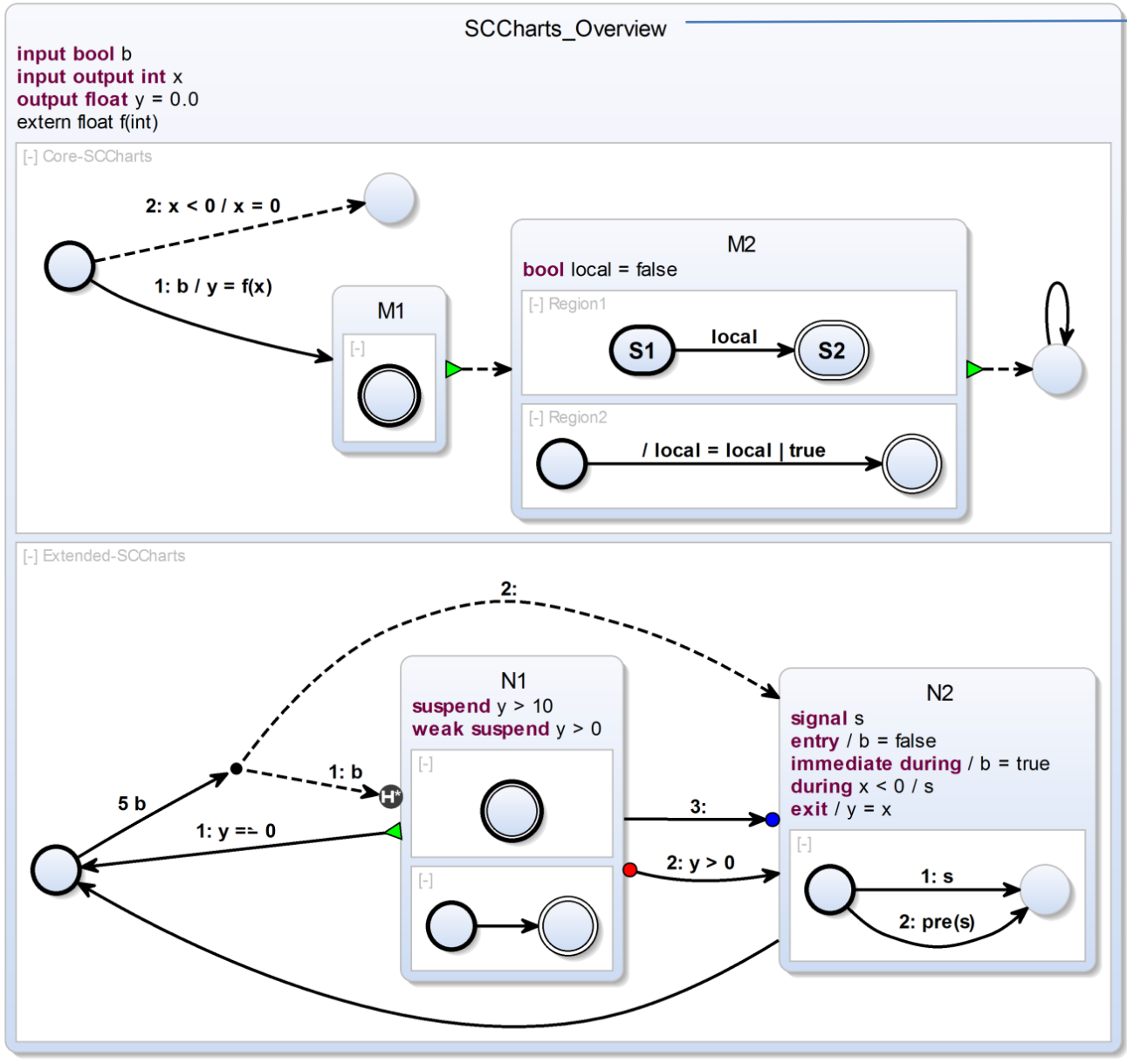


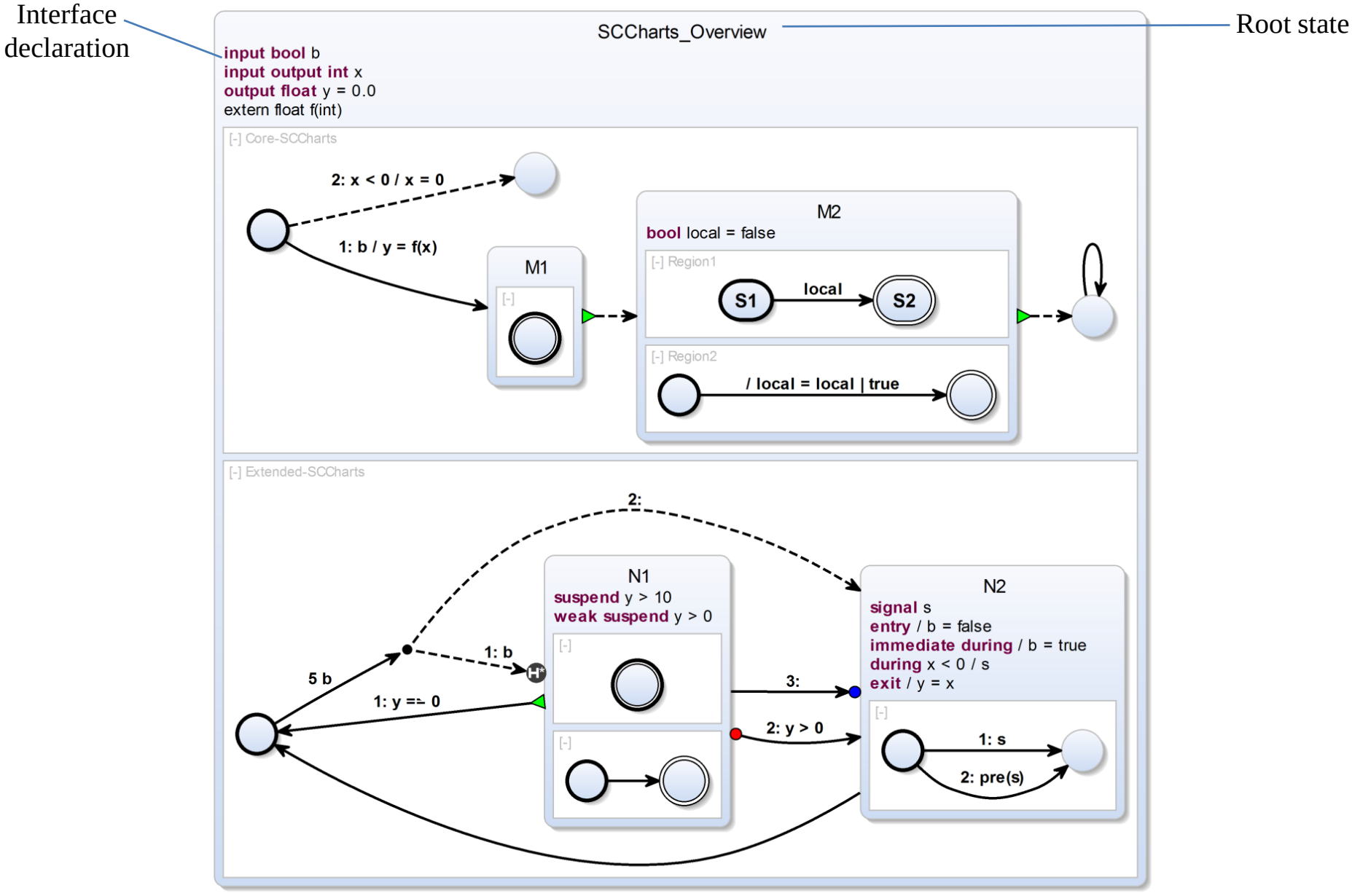
- SCCharts $\widehat{=}$
  SyncCharts syntax +
  Seqentially Constructive semantics
- *Hello World* of Sequential Constructiveness: **ABO**
  - Variables instead of signals
  - Behavior (briefly)
    1. Initialize
    2. Concurrently wait for inputs A or B to become *true*
    3. Once A and B are true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

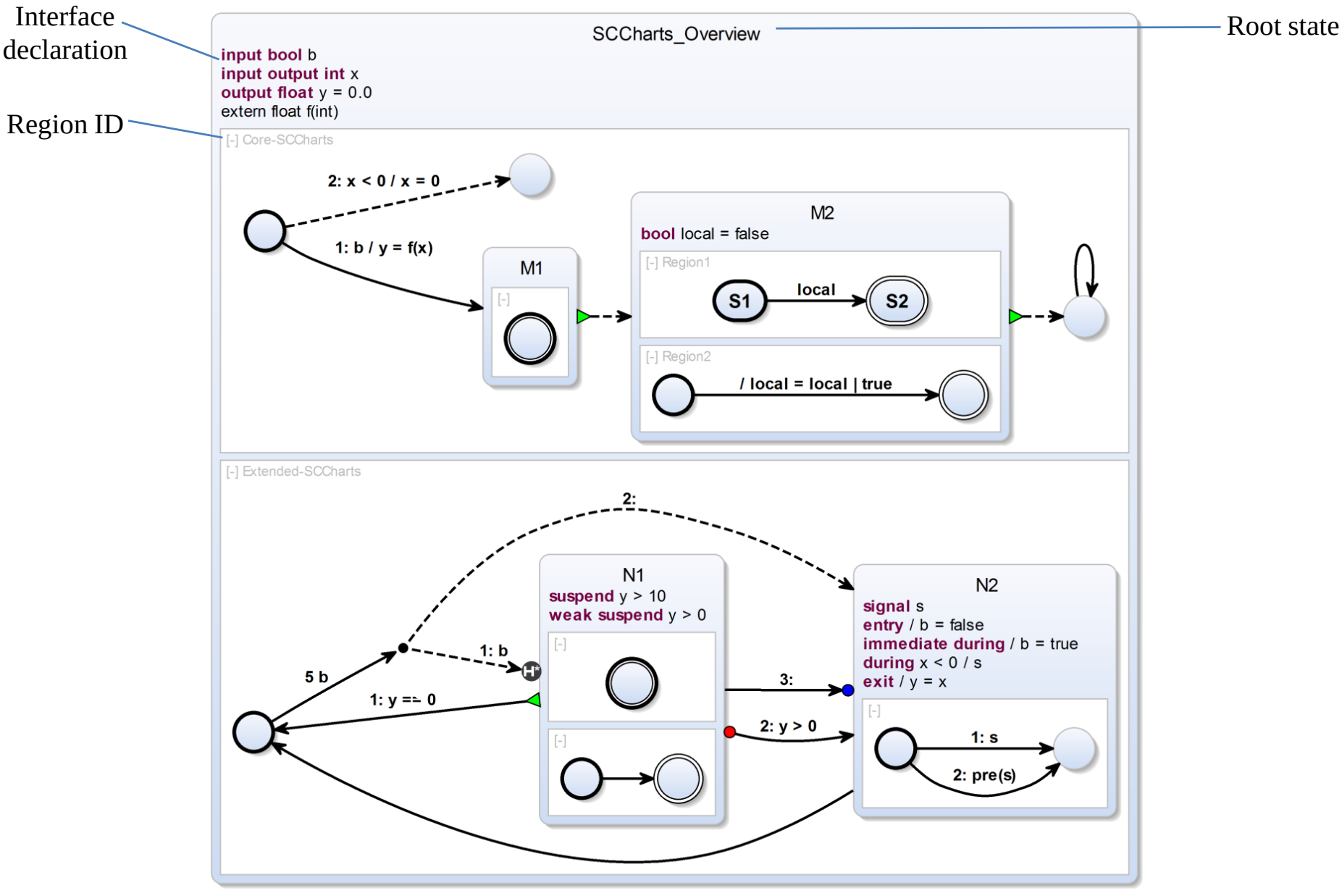


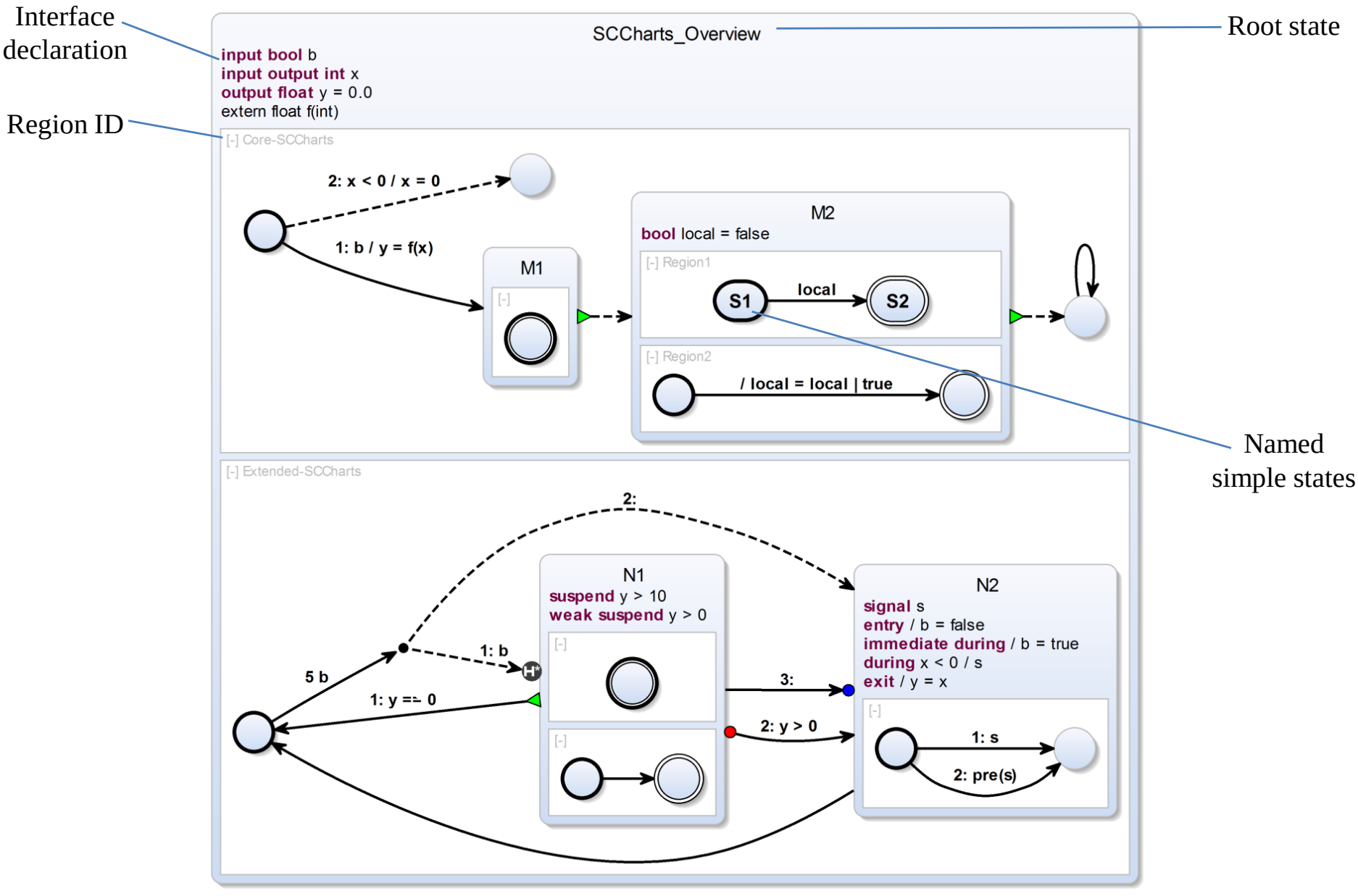
SCCharts — Features

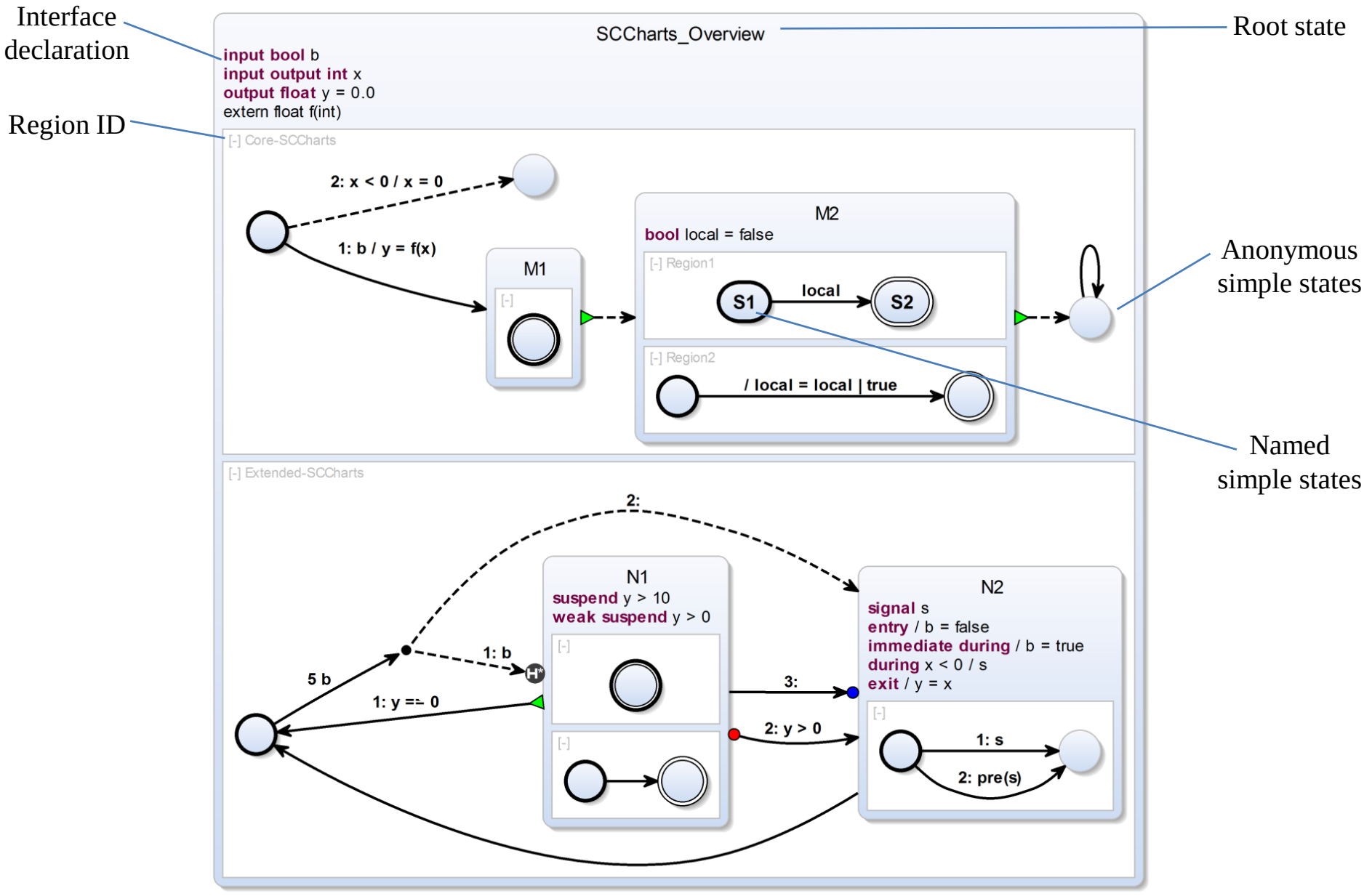


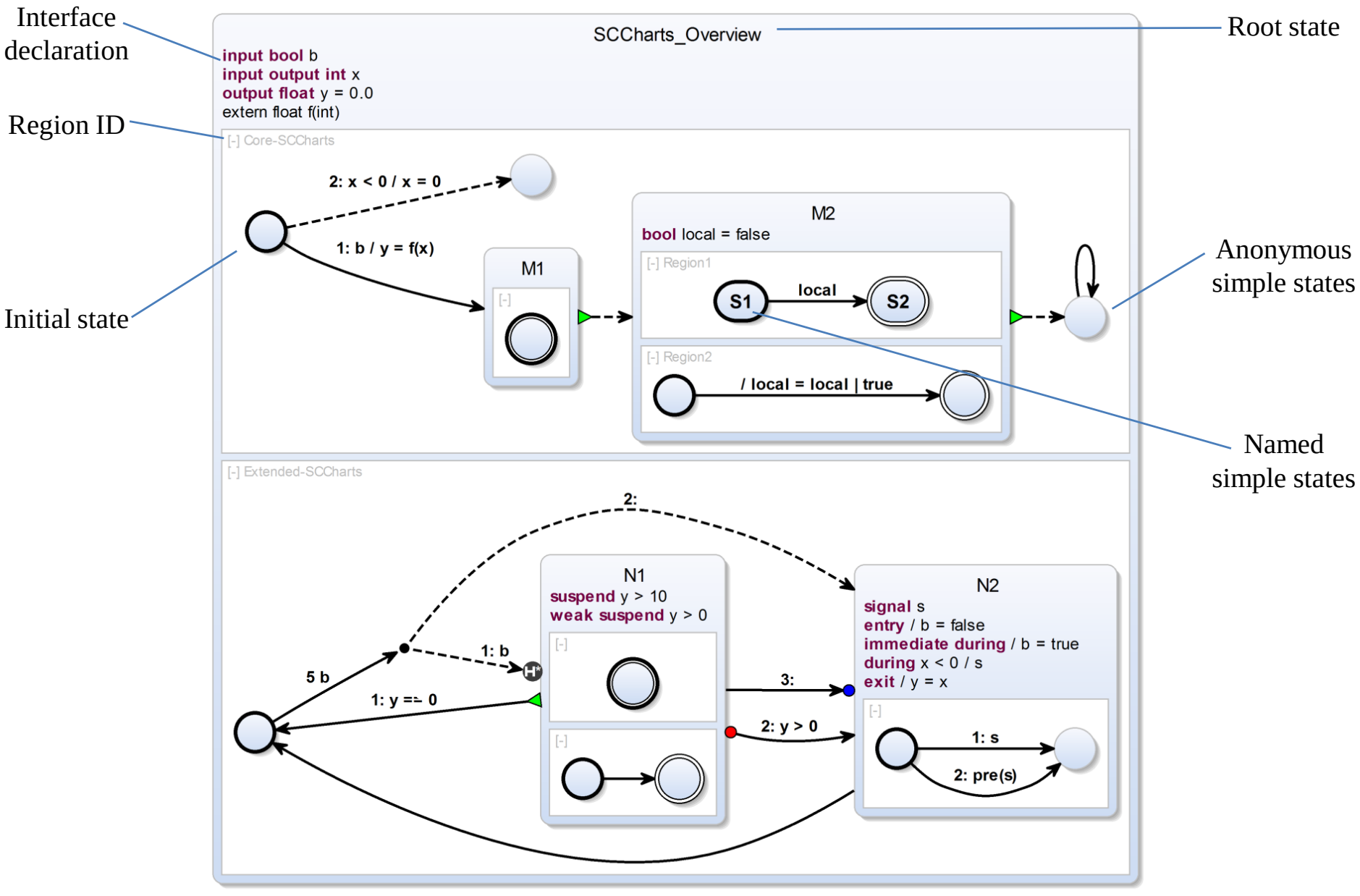

SCCharts_Overview
Root state
input bool b
input output int x
output float y = 0.0
extern float f(int)
[-] Core-SCCharts
2: x < 0 / x = 0
1: b / y = f(x)
M1
[-]
M2
bool local = false
[-] Region1
S1
local
S2
[-] Region2
/ local = local | true
[-] Extended-SCCharts
2:
5 b
1: b
1: y == 0
N1
suspend y > 10
weak suspend y > 0
[-]
[-]
3:
2: y > 0
N2
signal s
entry / b = false
immediate during / b = true
during x < 0 / s
exit / y = x
[-]
1: s
2: pre(s)

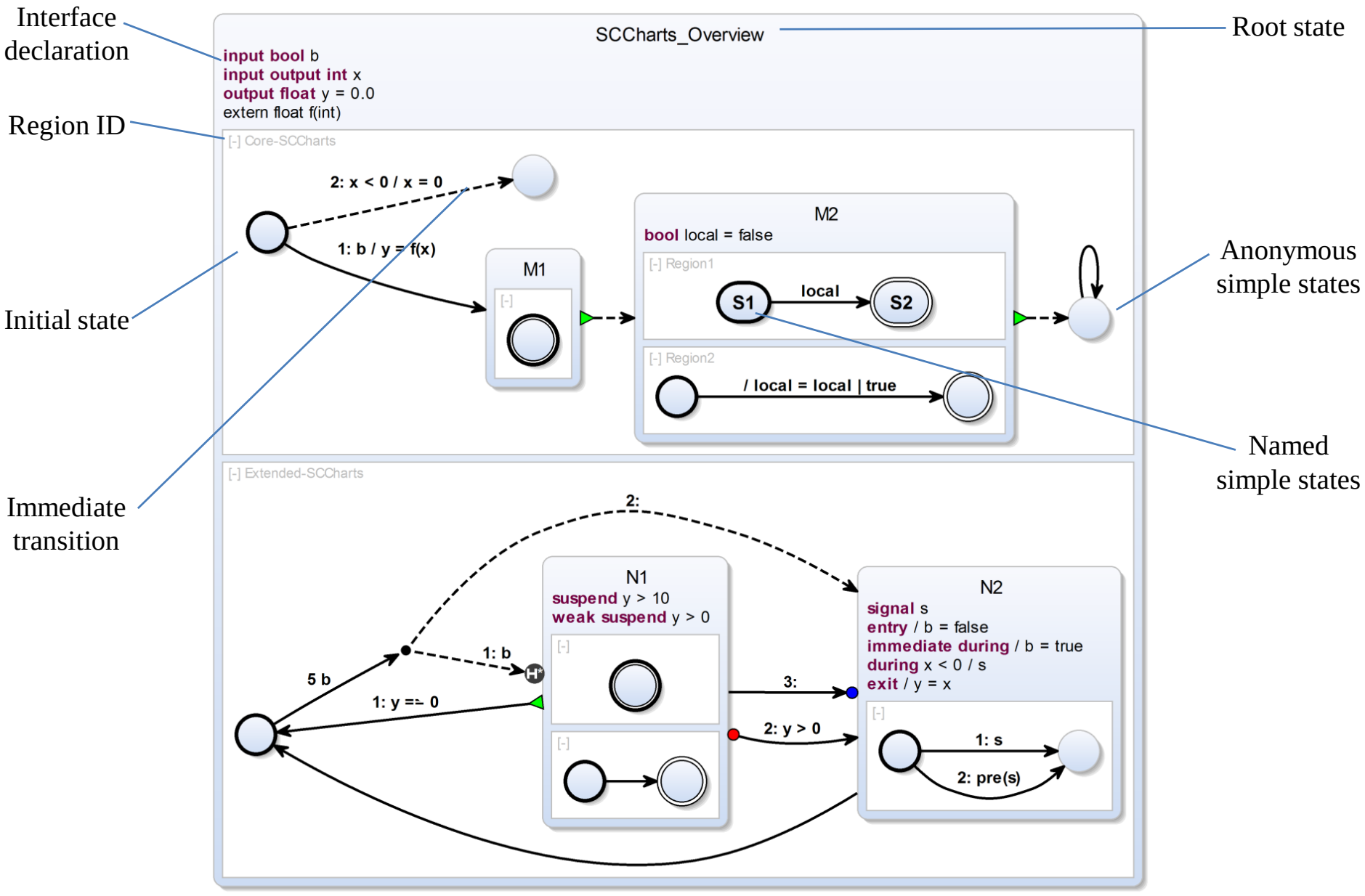

Interface declaration
Root state
SCCharts_Overview
input bool b
input output int x
output float y = 0.0
extern float f(int)
[-] Core-SCCharts
2: x < 0 / x = 0
1: b / y = f(x)
M1
[-]
M2
bool local = false
[-] Region1
S1
local
S2
[-] Region2
/ local = local | true
[-] Extended-SCCharts
2:
5 b
1: b
1: y == 0
N1
suspend y > 10
weak suspend y > 0
[-]
[-]
3:
2: y > 0
N2
signal s
entry / b = false
immediate during / b = true
during x < 0 / s
exit / y = x
[-]
1: s
2: pre(s)

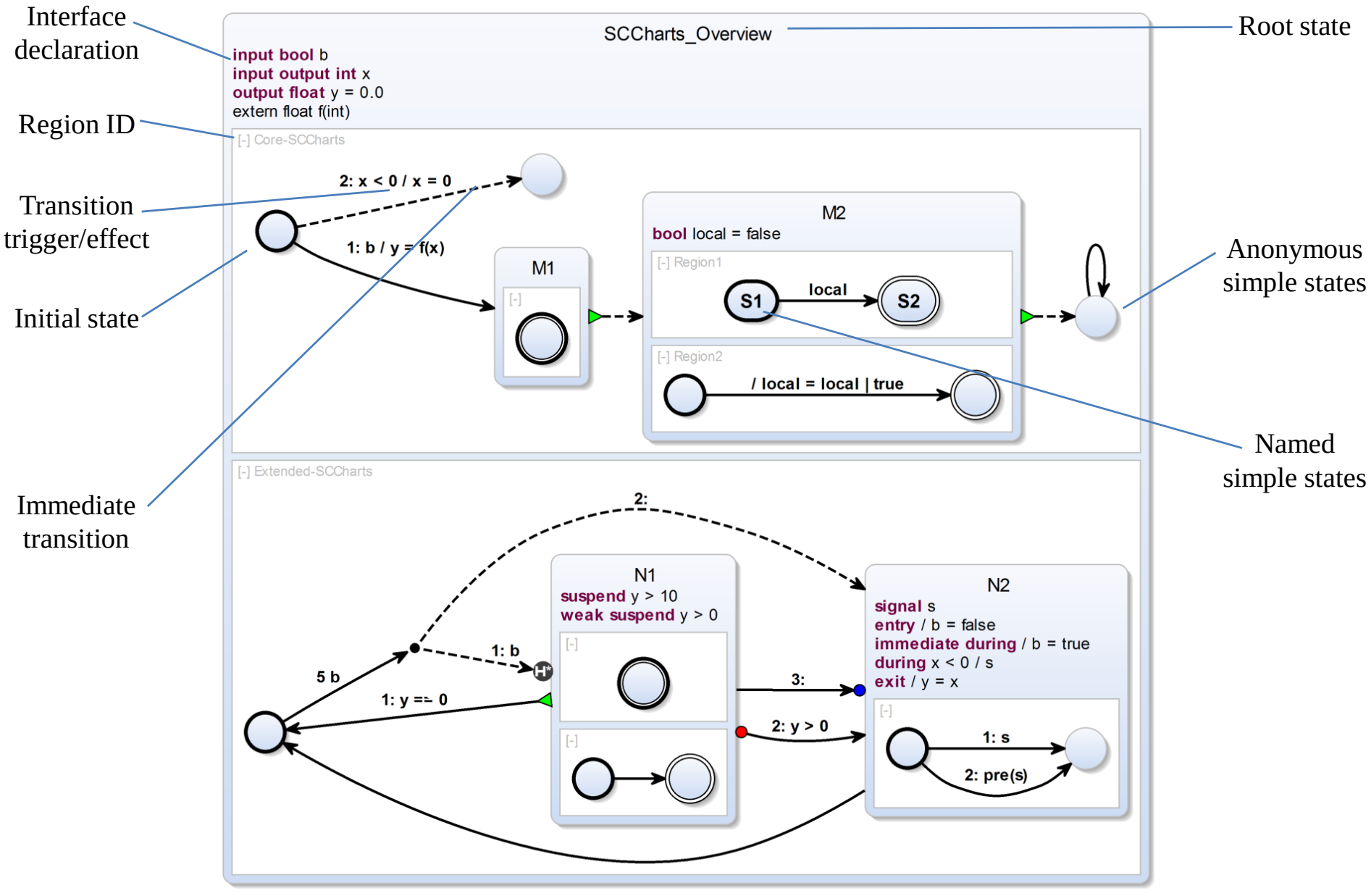

Interface declaration
Region ID
SCCharts_Overview
Root state
input bool b
input output int x
output float y = 0.0
extern float f(int)
[-] Core-SCCharts
2: x < 0 / x = 0
1: b / y = f(x)
M1
[-]
M2
bool local = false
[-] Region1
S1
local
S2
[-] Region2
/ local = local | true
[-] Extended-SCCharts
2:
5 b
1: b
1: y == 0
N1
suspend y > 10
weak suspend y > 0
[-]
[-]
3:
2: y > 0
N2
signal s
entry / b = false
immediate during / b = true
during x < 0 / s
exit / y = x
[-]
1: s
2: pre(s)

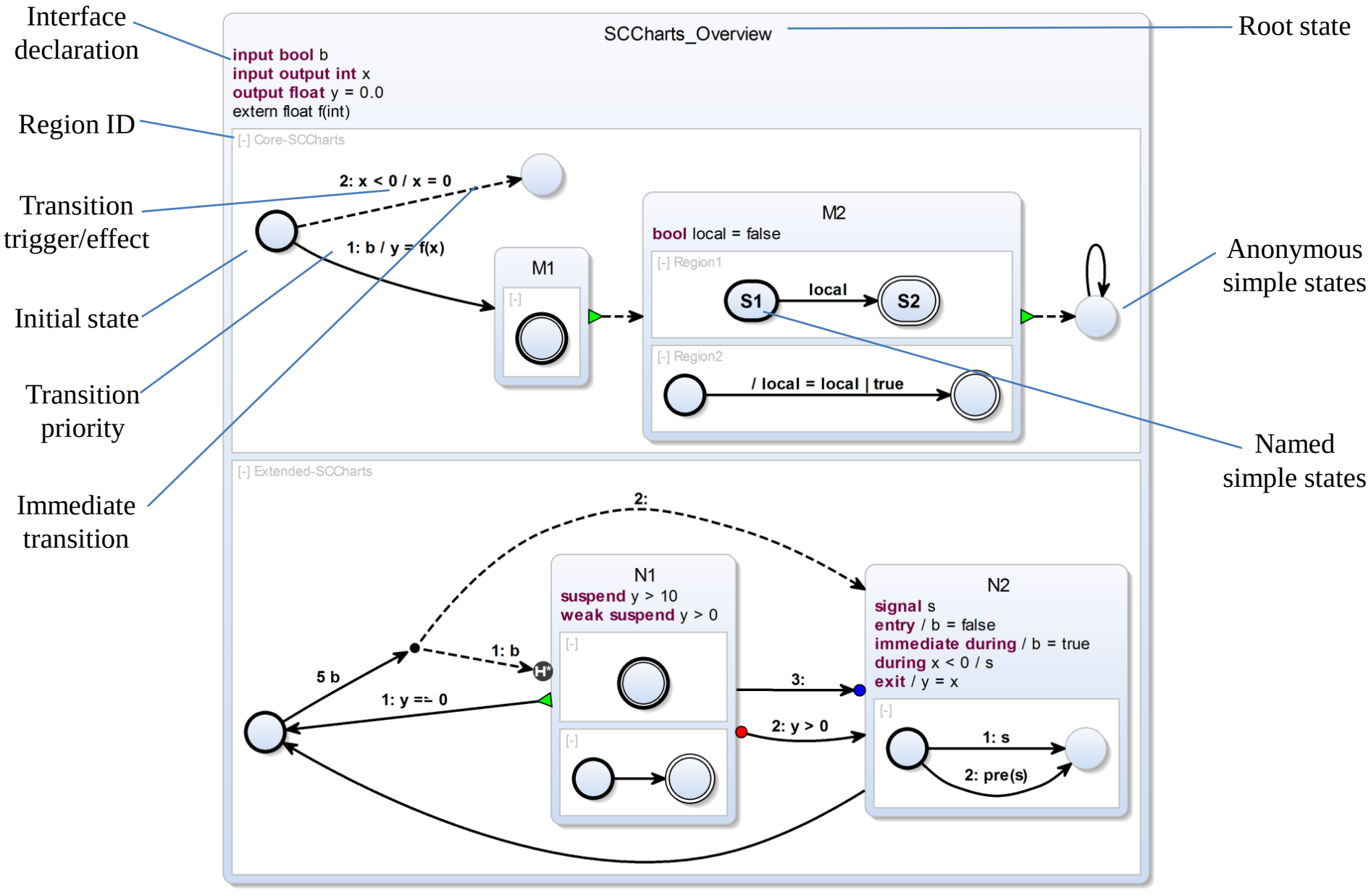

Interface declaration
Region ID
Root state
Named simple states
SCCharts_Overview
input bool b
input output int x
output float y = 0.0
extern float f(int)
[-] Core-SCCharts
2: x < 0 / x = 0
1: b / y = f(x)
M1
[-]
M2
bool local = false
[-] Region1
S1
local
S2
[-] Region2
/ local = local | true
[-] Extended-SCCharts
2:
5 b
1: b
1: y == 0
N1
suspend y > 10
weak suspend y > 0
[-]
[-]
3:
2: y > 0
N2
signal s
entry / b = false
immediate during / b = true
during x < 0 / s
exit / y = x
[-]
1: s
2: pre(s)

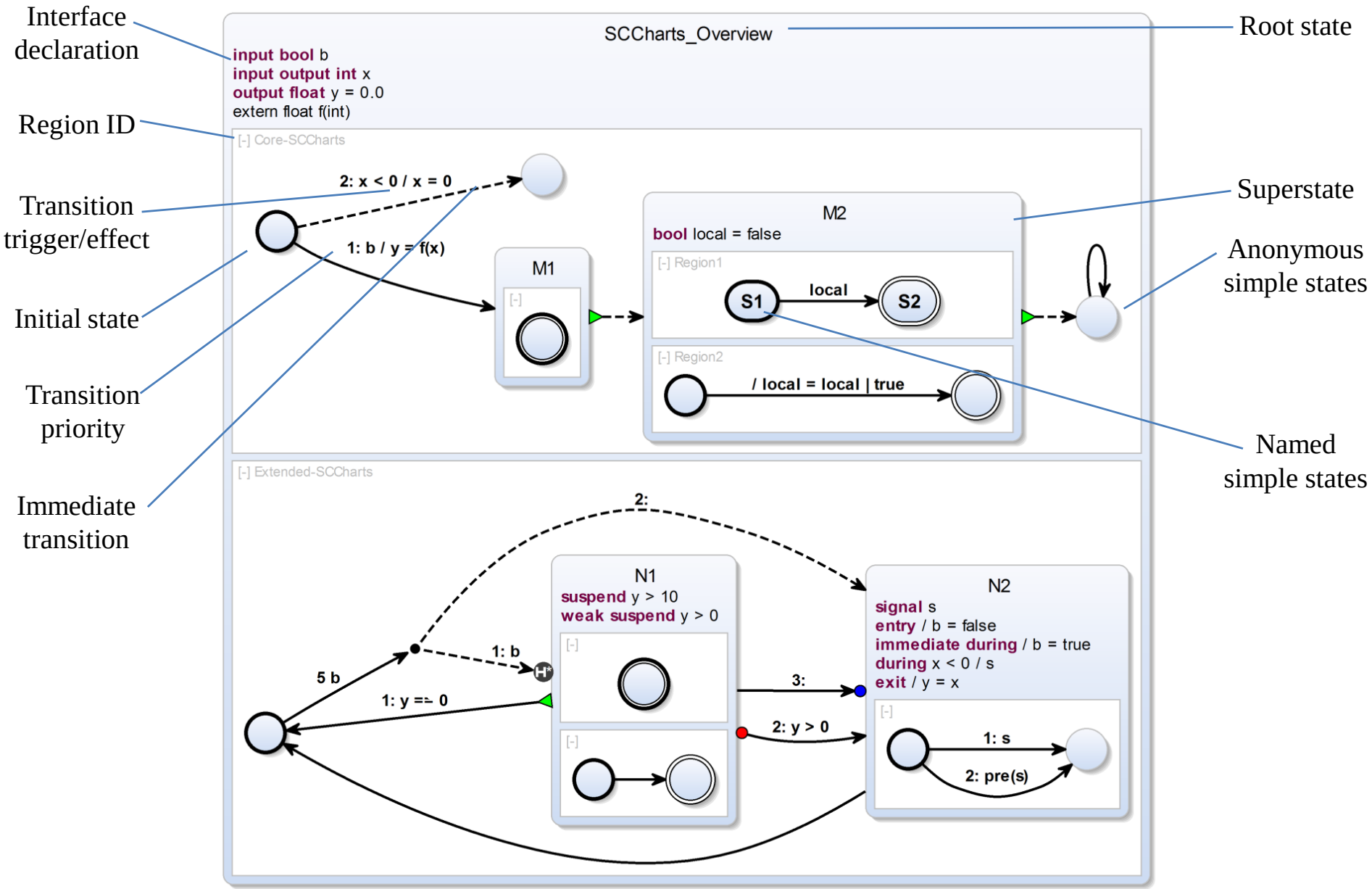

Interface declaration
Region ID
SCCharts_Overview
Root state
input bool b
input output int x
output float y = 0.0
extern float f(int)
[-] Core-SCCharts
2: x < 0 / x = 0
1: b / y = f(x)
M1
[-]
M2
bool local = false
[-] Region1
S1
local
S2
[-] Region2
/ local = local | true
Anonymous simple states
Named simple states
[-] Extended-SCCharts
2:
5 b
1: b
1: y == 0
N1
suspend y > 10
weak suspend y > 0
[-]
[-]
3:
2: y > 0
N2
signal s
entry / b = false
immediate during / b = true
during x < 0 / s
exit / y = x
[-]
1: s
2: pre(s)

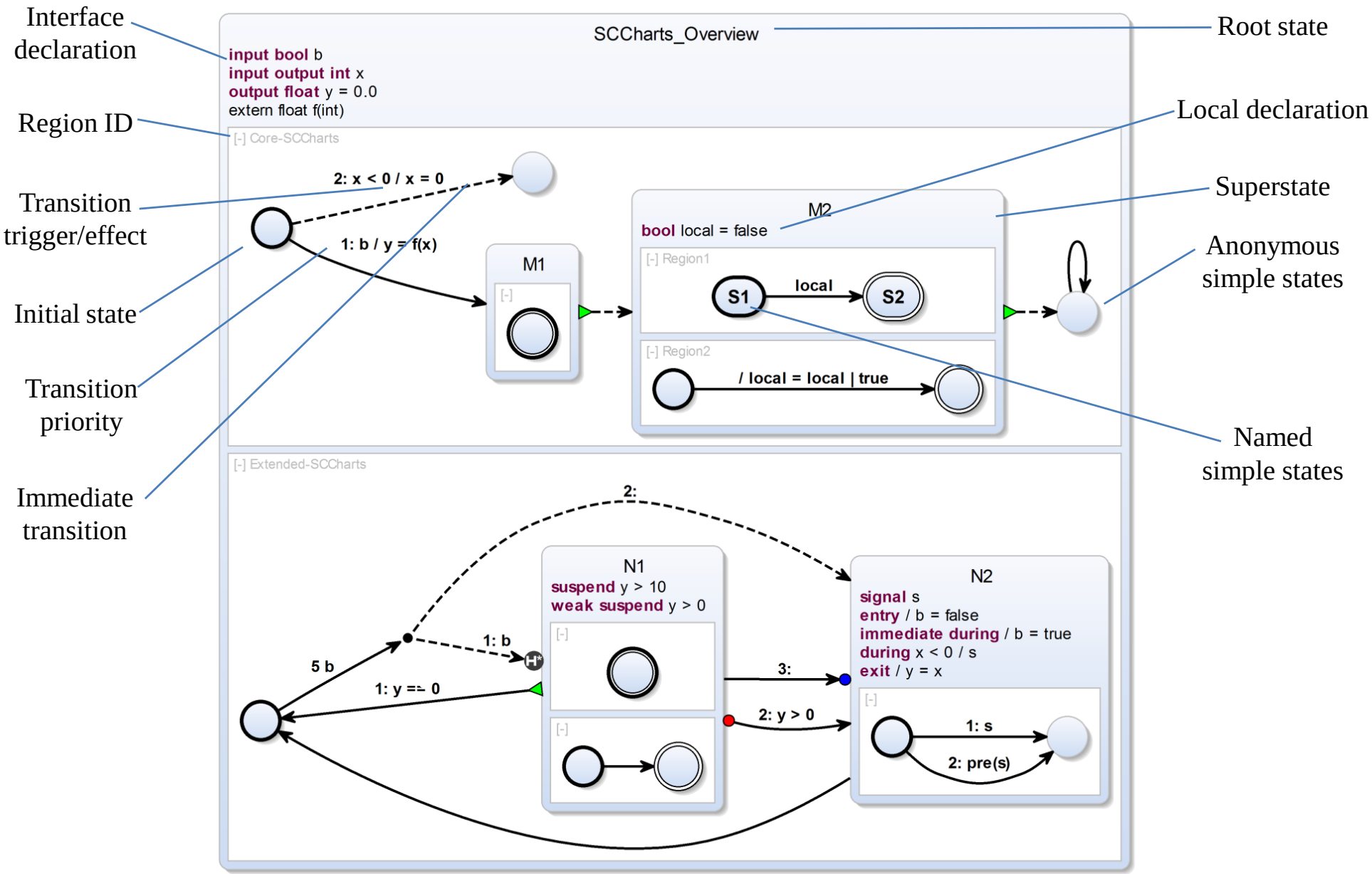

Interface declaration
Region ID
Initial state
SCCharts_Overview
Root state
input bool b
input output int x
output float y = 0.0
extern float f(int)
[-] Core-SCCharts
2: x < 0 / x = 0
1: b / y = f(x)
M1
[-]
M2
bool local = false
[-] Region1
S1
local
S2
[-] Region2
/ local = local | true
Anonymous simple states
Named simple states
[-] Extended-SCCharts
2:
1: b
5 b
1: y == 0
N1
suspend y > 10
weak suspend y > 0
[-]
[-]
3:
2: y > 0
N2
signal s
entry / b = false
immediate during / b = true
during x < 0 / s
exit / y = x
[-]
1: s
2: pre(s)

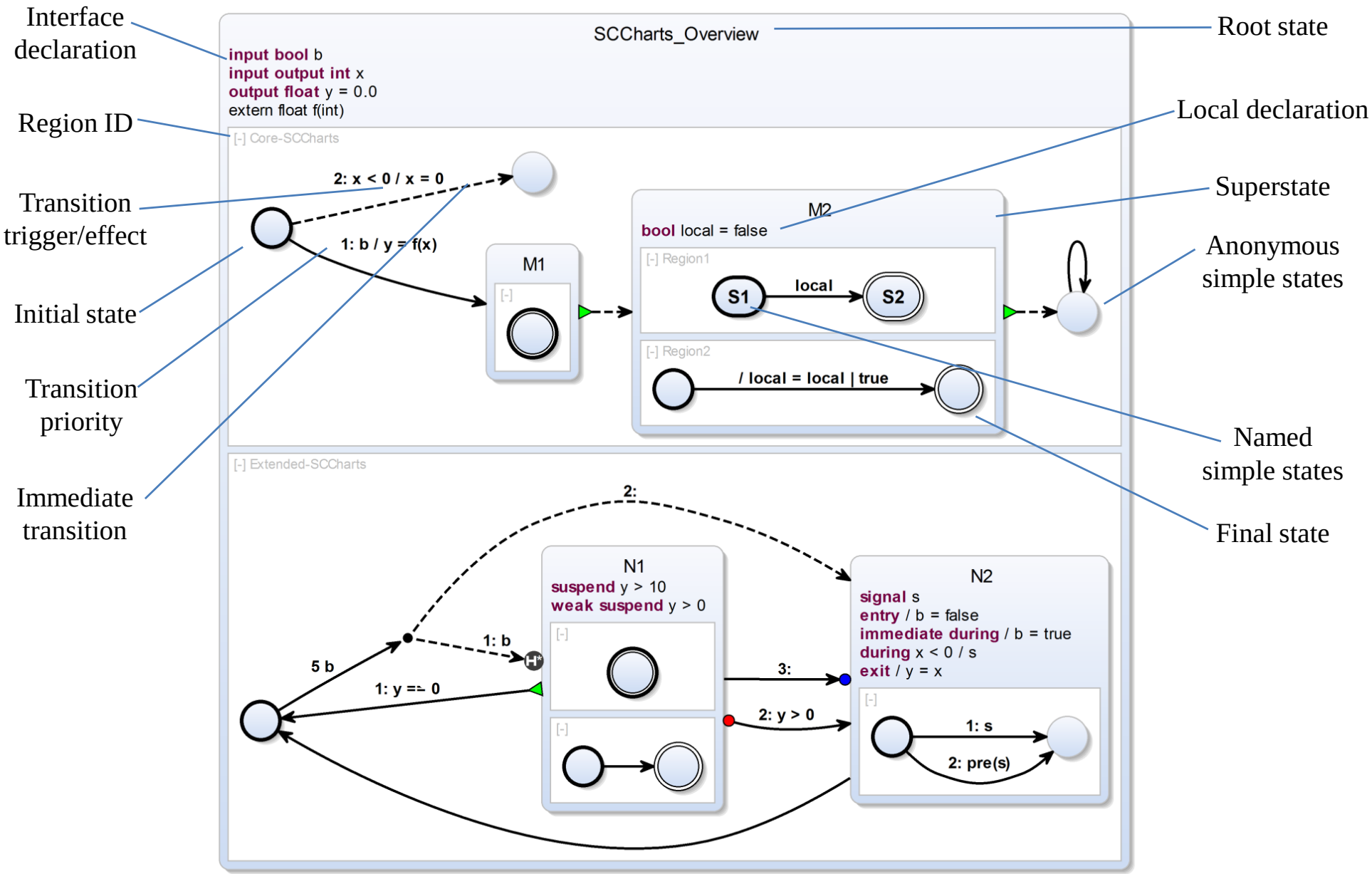

Interface declaration
Region ID
Initial state
Immediate transition
Root state
Anonymous simple states
Named simple states
SCCharts_Overview
input bool b
input output int x
output float y = 0.0
extern float f(int)
[-] Core-SCCharts
2: x < 0 / x = 0
1: b / y = f(x)
M1
[-]
M2
bool local = false
[-] Region1
S1
local
S2
[-] Region2
/ local = local | true
[-] Extended-SCCharts
2:
5 b
1: b
1: y == 0
N1
suspend y > 10
weak suspend y > 0
[-]
[-]
3:
2: y > 0
N2
signal s
entry / b = false
immediate during / b = true
during x < 0 / s
exit / y = x
[-]
1: s
2: pre(s)

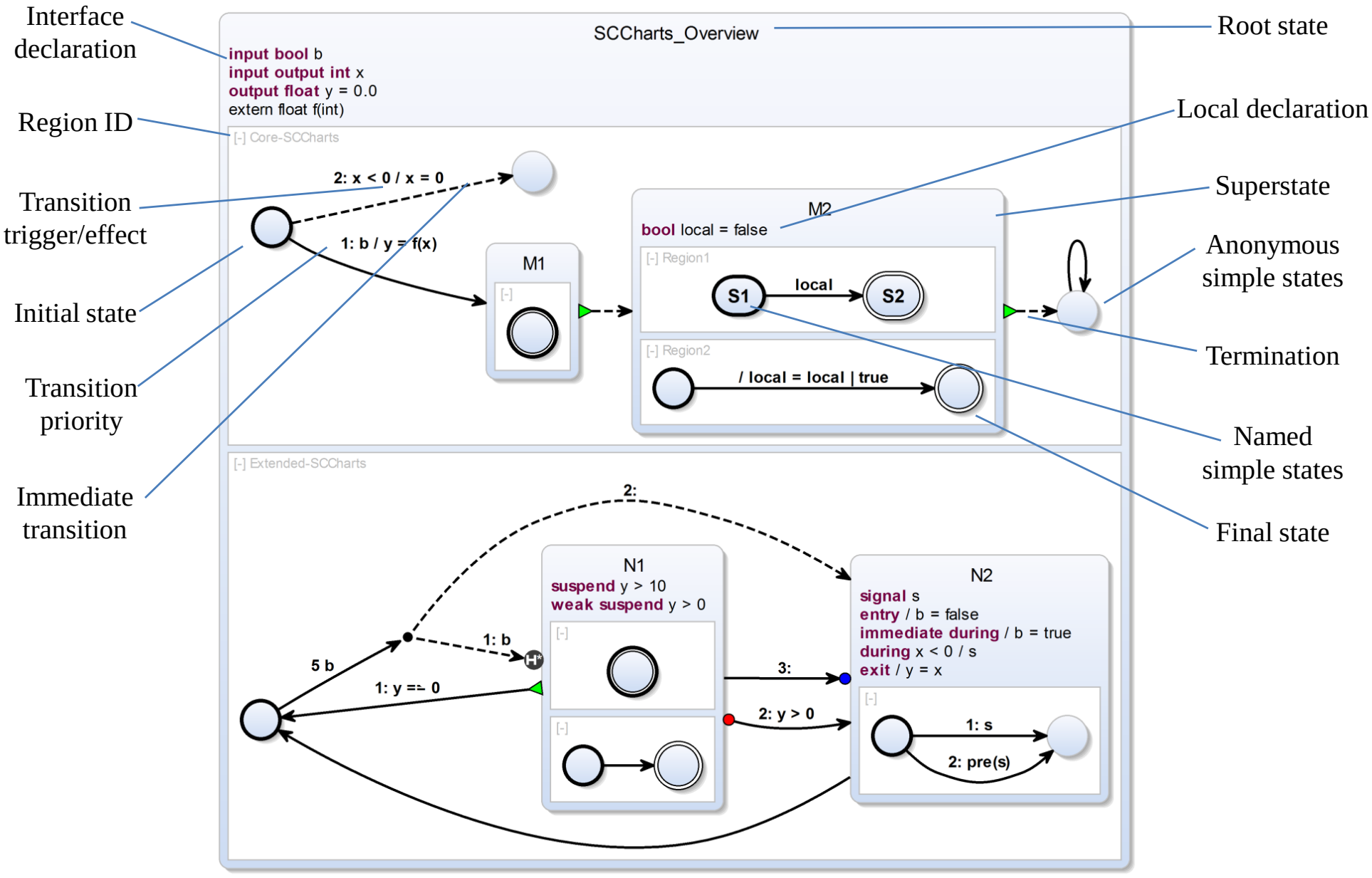

Interface declaration
Region ID
Transition trigger/effect
Initial state
Immediate transition
Root state
Anonymous simple states
Named simple states
SCCharts_Overview
input bool b
input output int x
output float y = 0.0
extern float f(int)
[-] Core-SCCharts
2: x < 0 / x = 0
1: b / y = f(x)
M1
[-]
M2
bool local = false
[-] Region1
S1
local
S2
[-] Region2
/ local = local | true
[-] Extended-SCCharts
2:
5 b
1: b
1: y == 0
N1
suspend y > 10
weak suspend y > 0
[-]
[-]
3:
2: y > 0
N2
signal s
entry / b = false
immediate during / b = true
during x < 0 / s
exit / y = x
[-]
1: s
2: pre(s)

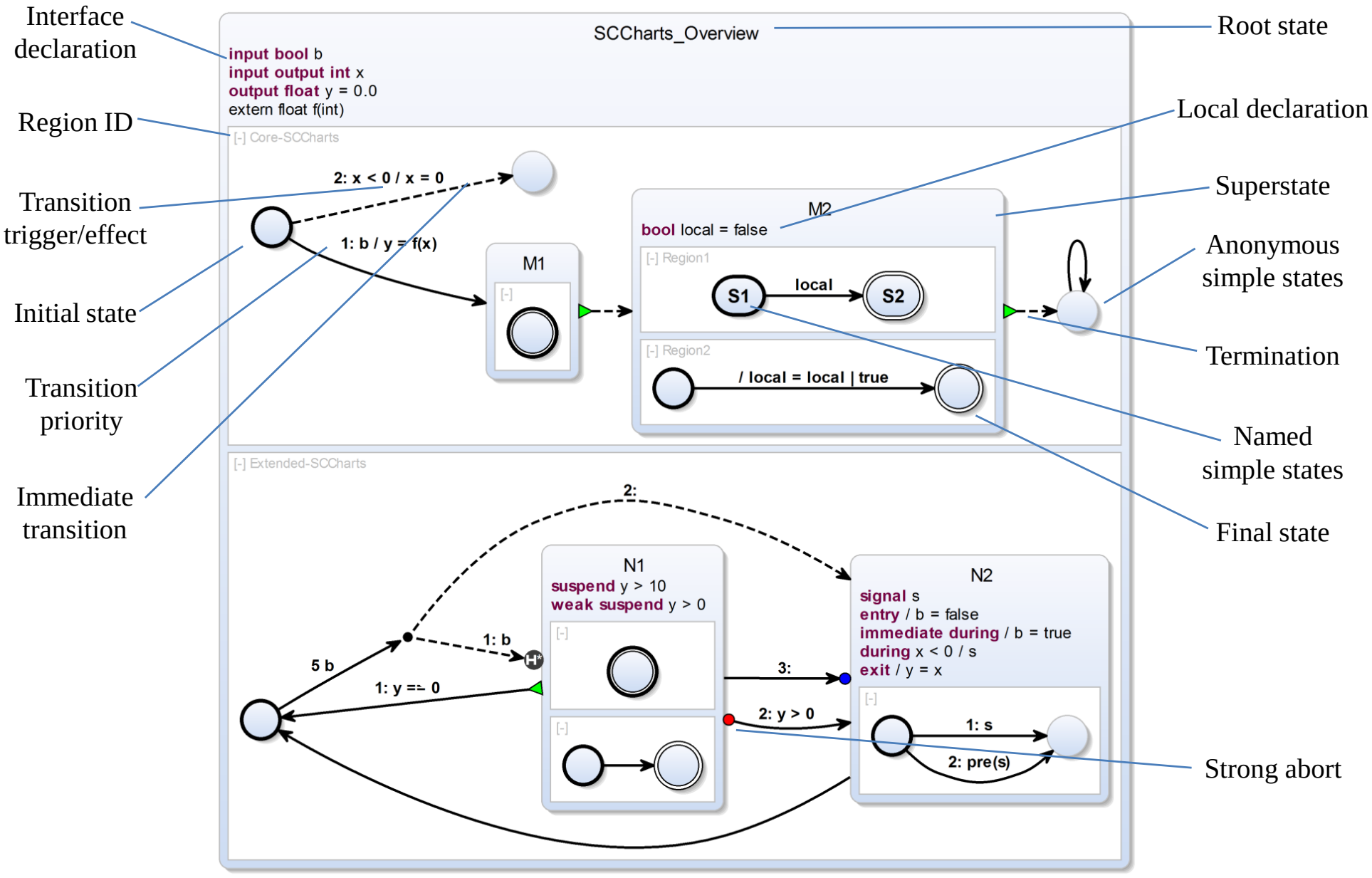

Interface declaration
Region ID
Transition trigger/effect
Initial state
Transition priority
Immediate transition
SCCharts_Overview
Root state
input bool b
input output int x
output float y = 0.0
extern float f(int)
[-] Core-SCCharts
2: x < 0 / x = 0
1: b / y = f(x)
M1
[-]
M2
bool local = false
[-] Region1
S1
local
S2
[-] Region2
/ local = local | true
Anonymous simple states
Named simple states
[-] Extended-SCCharts
2:
1: b
5 b
1: y == 0
N1
suspend y > 10
weak suspend y > 0
[-]
[-]
3:
2: y > 0
N2
signal s
entry / b = false
immediate during / b = true
during x < 0 / s
exit / y = x
[-]
1: s
2: pre(s)

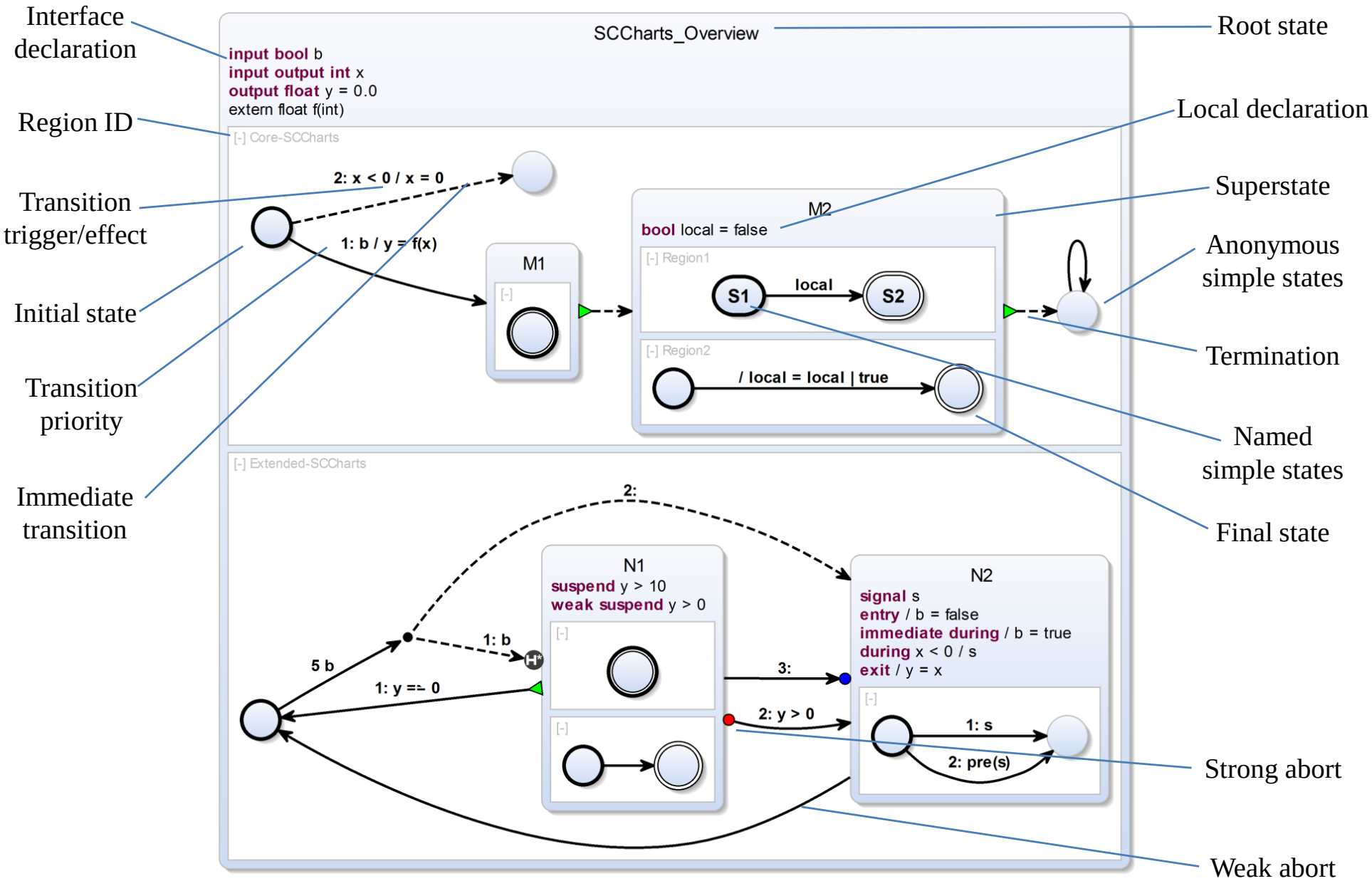

Interface declaration
Region ID
Transition trigger/effect
Initial state
Transition priority
Immediate transition
Root state
Superstate
Anonymous simple states
Named simple states
SCCharts_Overview
input bool b
input output int x
output float y = 0.0
extern float f(int)
[-] Core-SCCharts
2: x < 0 / x = 0
1: b / y = f(x)
M1
M2
bool local = false
[-] Region1
S1
local
S2
[-] Region2
/ local = local | true
[-] Extended-SCCharts
2:
1: b
5 b
1: y == 0
N1
suspend y > 10
weak suspend y > 0
3:
2: y > 0
N2
signal s
entry / b = false
immediate during / b = true
during x < 0 / s
exit / y = x
1: s
2: pre(s)

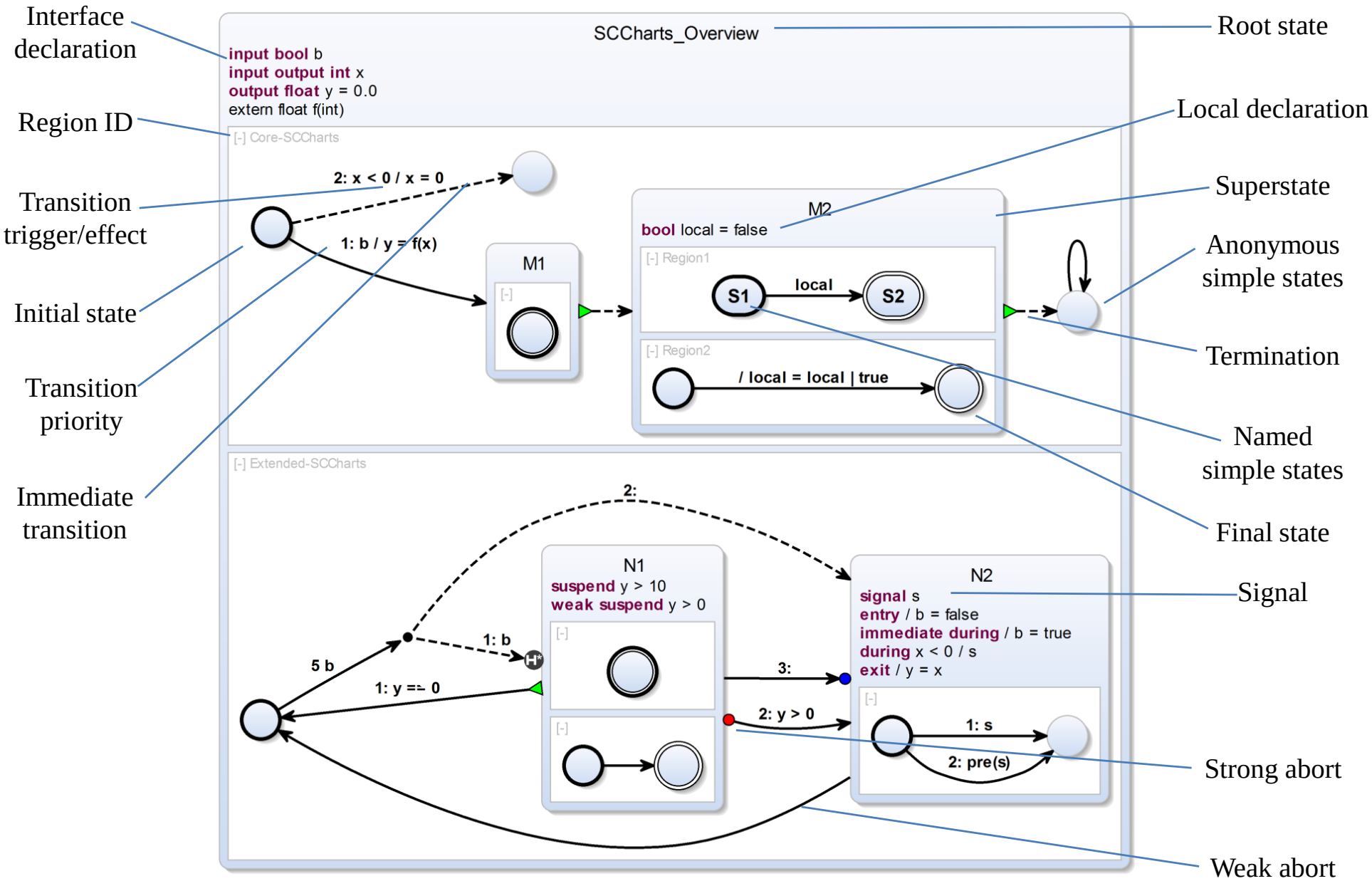

Interface declaration
Region ID
Transition trigger/effect
Initial state
Transition priority
Immediate transition
SCCharts_Overview
Root state
input bool b
input output int x
output float y = 0.0
extern float f(int)
Local declaration
[-] Core-SCCharts
2: x < 0 / x = 0
1: b / y = f(x)
M1
[-]
M2
bool local = false
Superstate
[-] Region1
S1
local
S2
Anonymous simple states
[-] Region2
/ local = local | true
Named simple states
[-] Extended-SCCharts
2:
N1
suspend y > 10
weak suspend y > 0
N2
signal s
entry / b = false
immediate during / b = true
during x < 0 / s
exit / y = x
5 b
1: b
1: y == 0
3:
2: y > 0
1: s
2: pre(s)

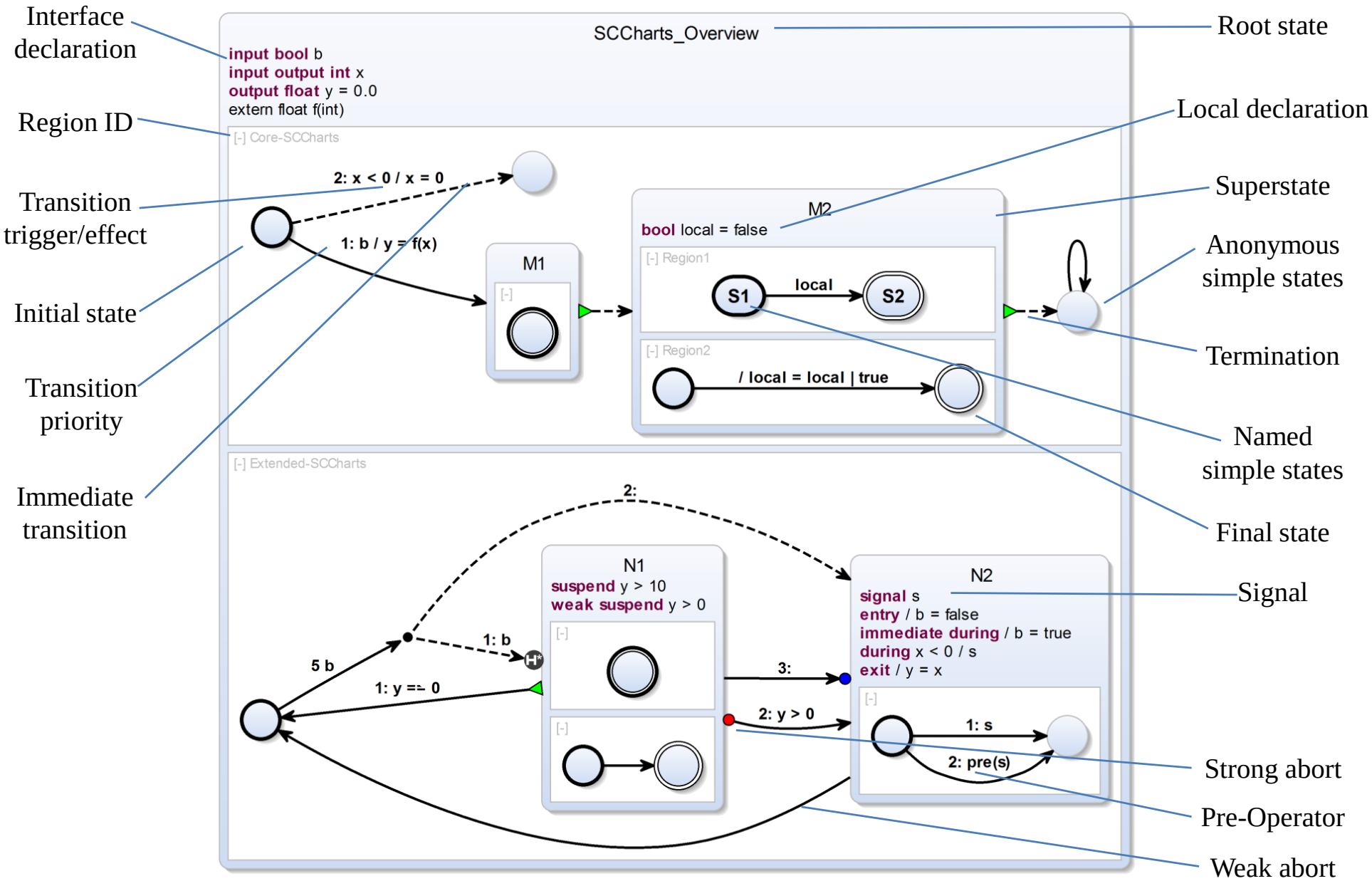

Interface declaration
Region ID
Transition trigger/effect
Initial state
Transition priority
Immediate transition
Root state
Local declaration
Superstate
Anonymous simple states
Named simple states
Final state
SCCharts_Overview
input bool b
input output int x
output float y = 0.0
extern float f(int)
[-] Core-SCCharts
2: x < 0 / x = 0
1: b / y = f(x)
M1
M2
bool local = false
[-] Region1
S1
local
S2
[-] Region2
/ local = local | true
[-] Extended-SCCharts
2:
5 b
1: b
1: y == 0
N1
suspend y > 10
weak suspend y > 0
3:
2: y > 0
N2
signal s
entry / b = false
immediate during / b = true
during x < 0 / s
exit / y = x
1: s
2: pre(s)

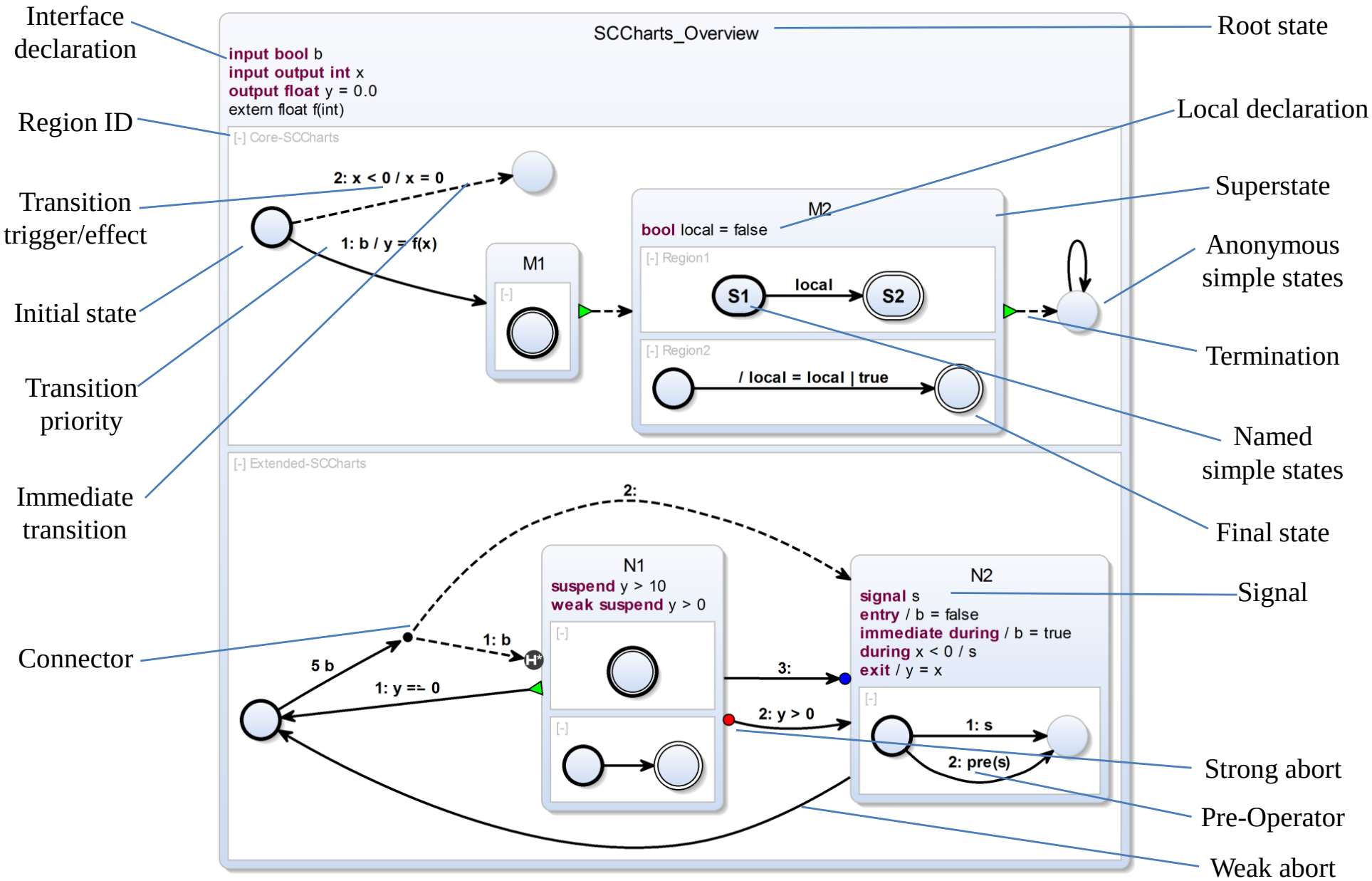

Interface declaration
Region ID
Transition trigger/effect
Initial state
Transition priority
Immediate transition
Root state
Local declaration
Superstate
Anonymous simple states
Termination
Named simple states
Final state
SCCharts_Overview
input bool b
input output int x
output float y = 0.0
extern float f(int)
[-] Core-SCCharts
2: x < 0 / x = 0
1: b / y = f(x)
M1
M2
bool local = false
[-] Region1
S1
local
S2
[-] Region2
/ local = local | true
[-] Extended-SCCharts
2:
1: b
5 b
1: y == 0
N1
suspend y > 10
weak suspend y > 0
3:
2: y > 0
N2
signal s
entry / b = false
immediate during / b = true
during x < 0 / s
exit / y = x
1: s
2: pre(s)

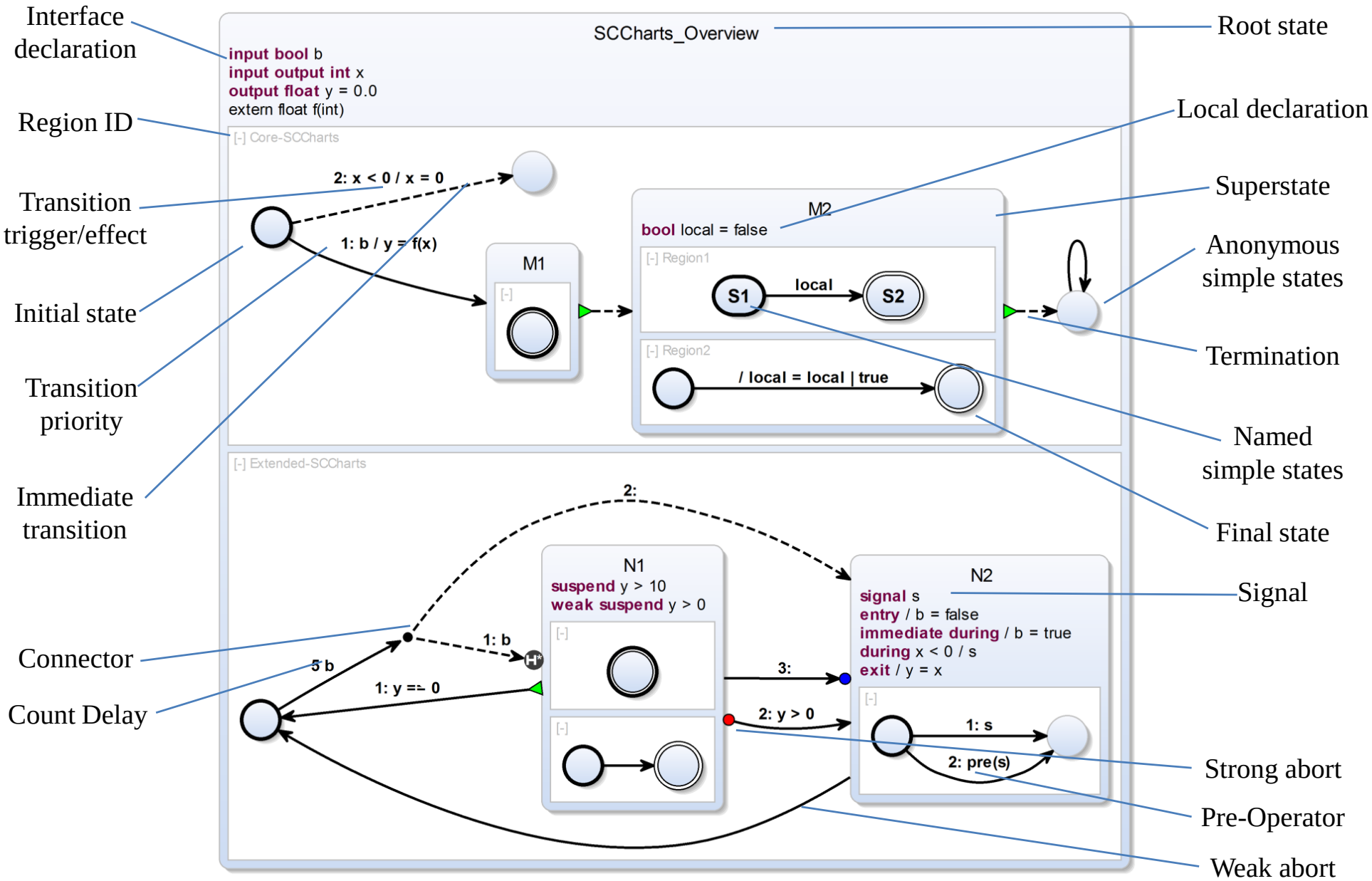

Interface declaration
Region ID
Transition trigger/effect
Initial state
Transition priority
Immediate transition
SCCharts_Overview
input bool b
input output int x
output float y = 0.0
extern float f(int)
[-] Core-SCCharts
2: x < 0 / x = 0
1: b / y = f(x)
M1
[-]
M2
bool local = false
[-] Region1
S1
local
S2
[-] Region2
/ local = local | true
[-] Extended-SCCharts
2:
5 b
1: b
1: y == 0
N1
suspend y > 10
weak suspend y > 0
3:
2: y > 0
N2
signal s
entry / b = false
immediate during / b = true
during x < 0 / s
exit / y = x
1: s
2: pre(s)
Root state
Local declaration
Superstate
Anonymous simple states
Termination
Named simple states
Final state
Strong abort

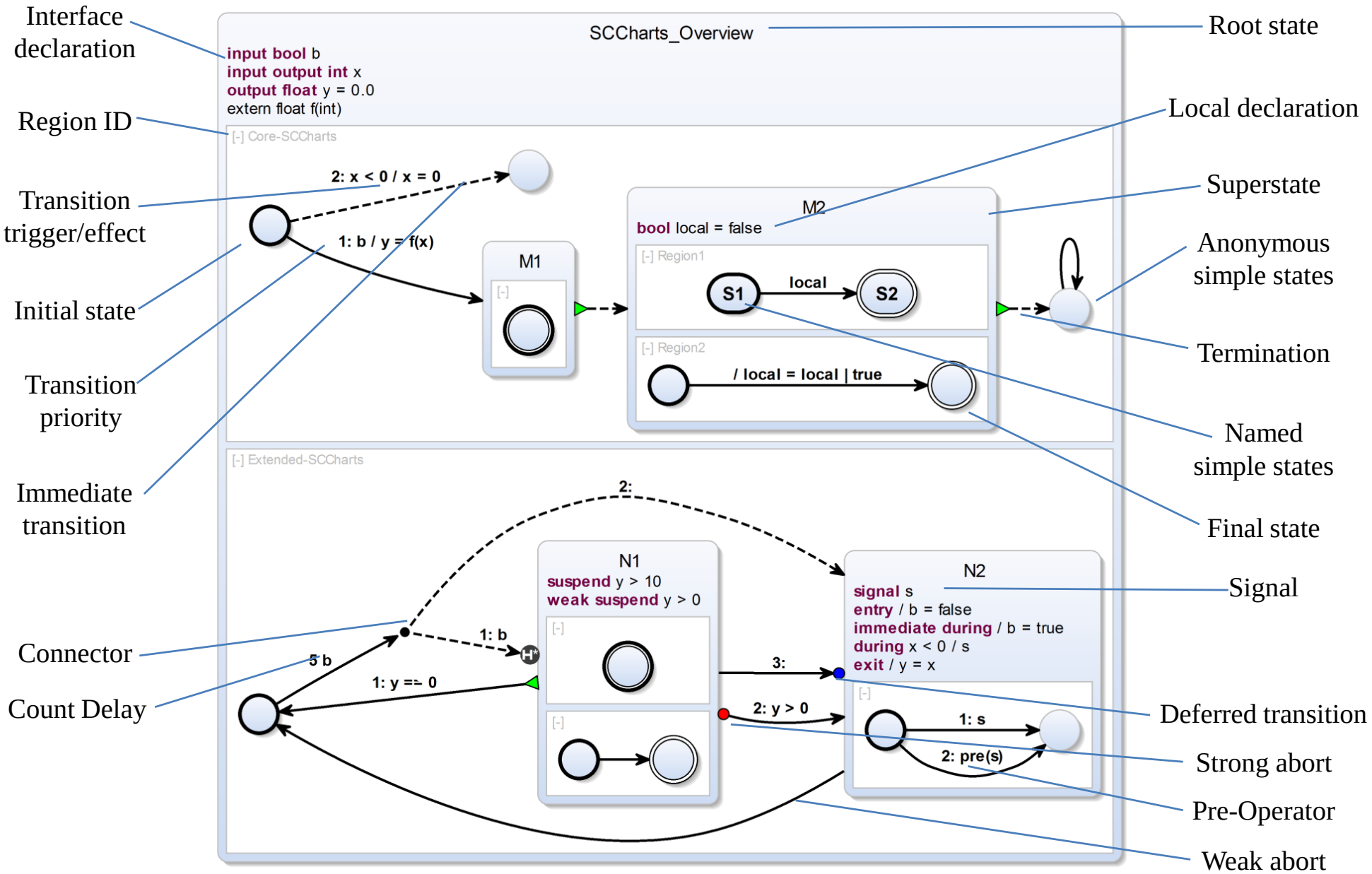

Interface declaration
Region ID
Transition trigger/effect
Initial state
Transition priority
Immediate transition
SCCharts_Overview
input bool b
input output int x
output float y = 0.0
extern float f(int)
[-] Core-SCCharts
2: x < 0 / x = 0
1: b / y = f(x)
M1
M2
bool local = false
[-] Region1
S1
local
S2
[-] Region2
/ local = local | true
[-] Extended-SCCharts
2:
5 b
1: b
1: y == 0
N1
suspend y > 10
weak suspend y > 0
3:
2: y > 0
N2
signal s
entry / b = false
immediate during / b = true
during x < 0 / s
exit / y = x
1: s
2: pre(s)
Root state
Local declaration
Superstate
Anonymous simple states
Termination
Named simple states
Final state
Strong abort
Weak abort

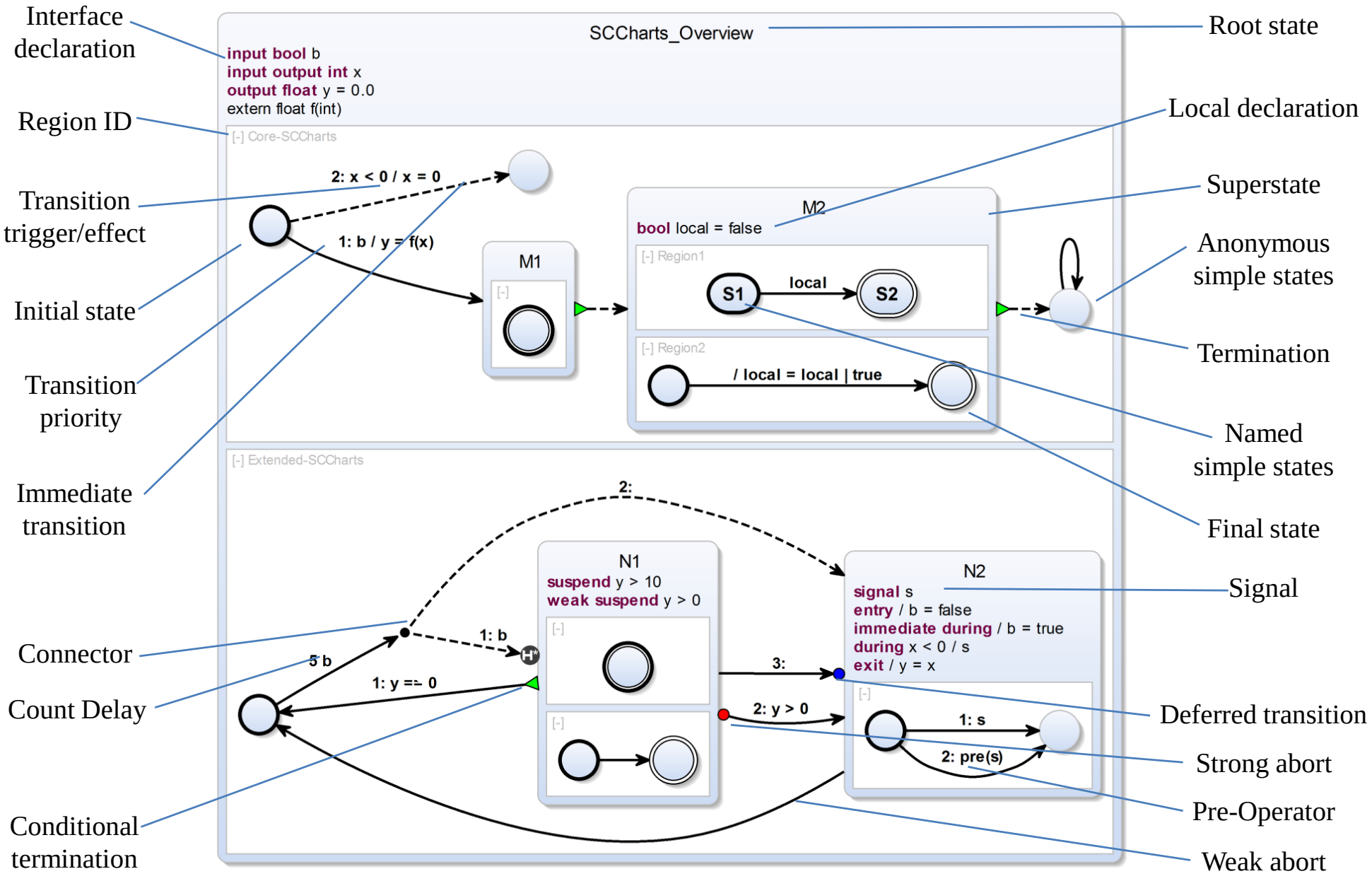

Interface declaration
Region ID
Transition trigger/effect
Initial state
Transition priority
Immediate transition
SCCharts_Overview
input bool b
input output int x
output float y = 0.0
extern float f(int)
[-] Core-SCCharts
2: x < 0 / x = 0
1: b / y = f(x)
M1
M2
bool local = false
[-] Region1
S1
local
S2
[-] Region2
/ local = local | true
[-] Extended-SCCharts
2:
5 b
1: b
1: y == 0
N1
suspend y > 10
weak suspend y > 0
3:
2: y > 0
N2
signal s
entry / b = false
immediate during / b = true
during x < 0 / s
exit / y = x
1: s
2: pre(s)
Root state
Local declaration
Superstate
Anonymous simple states
Termination
Named simple states
Final state
Signal
Strong abort
Weak abort

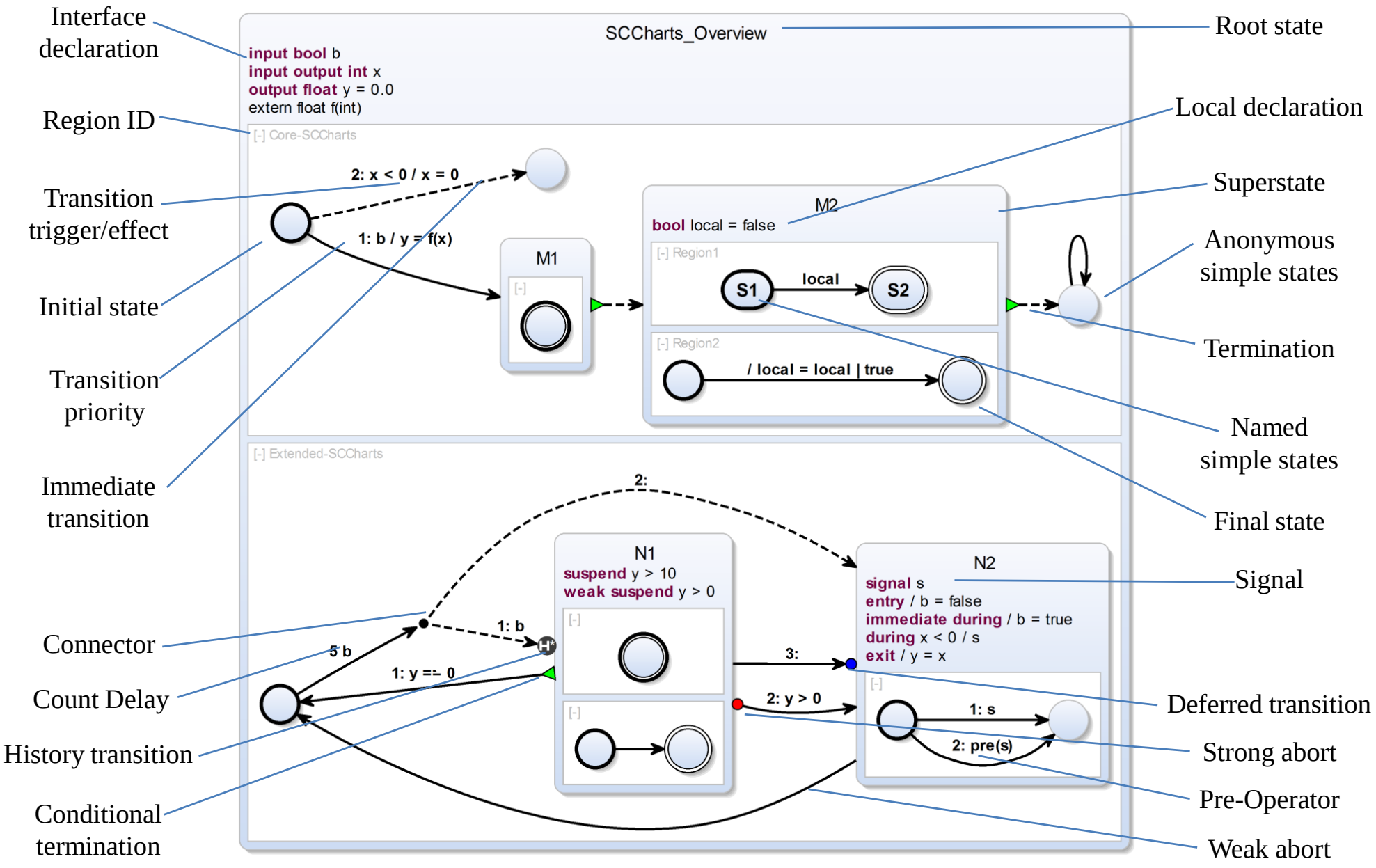

Interface declaration
Region ID
Transition trigger/effect
Initial state
Transition priority
Immediate transition
SCCharts_Overview
input bool b
input output int x
output float y = 0.0
extern float f(int)
[-] Core-SCCharts
2: x < 0 / x = 0
1: b / y = f(x)
M1
M2
bool local = false
[-] Region1
S1
local
S2
[-] Region2
/ local = local | true
[-] Extended-SCCharts
2:
5 b
1: b
1: y == 0
N1
suspend y > 10
weak suspend y > 0
3:
2: y > 0
N2
signal s
entry / b = false
immediate during / b = true
during x < 0 / s
exit / y = x
1: s
2: pre(s)
Root state
Local declaration
Superstate
Anonymous simple states
Termination
Named simple states
Final state
Signal
Strong abort
Pre-Operator
Weak abort

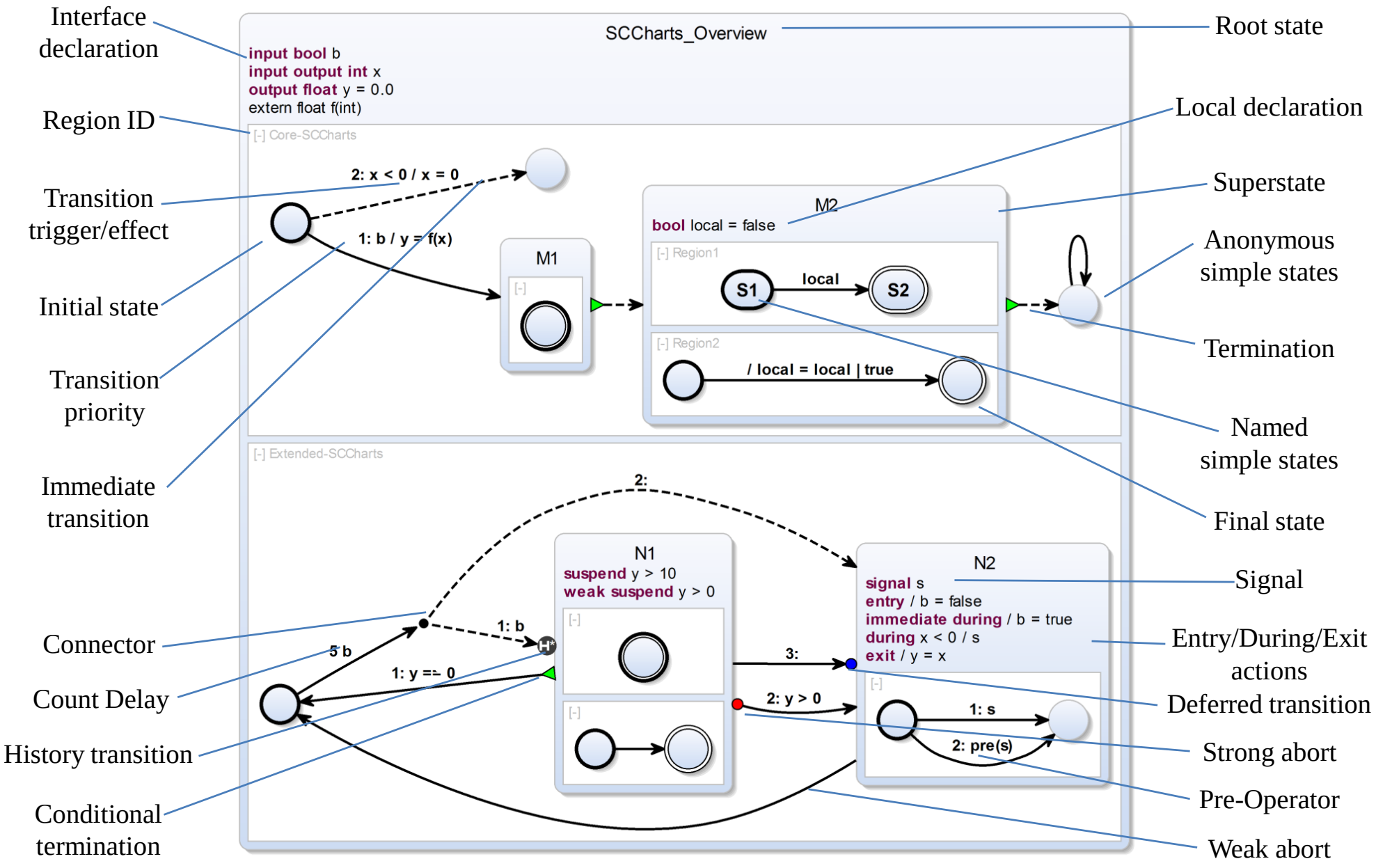

Interface declaration
Region ID
Transition trigger/effect
Initial state
Transition priority
Immediate transition
Connector
SCCharts_Overview
input bool b
input output int x
output float y = 0.0
extern float f(int)
[-] Core-SCCharts
2: x < 0 / x = 0
1: b / y = f(x)
M1
M2
bool local = false
[-] Region1
S1
local
S2
[-] Region2
/ local = local | true
[-] Extended-SCCharts
2:
1: b
5 b
1: y == 0
N1
suspend y > 10
weak suspend y > 0
3:
2: y > 0
N2
signal s
entry / b = false
immediate during / b = true
during x < 0 / s
exit / y = x
1: s
2: pre(s)
Root state
Local declaration
Superstate
Anonymous simple states
Termination
Named simple states
Final state
Signal
Strong abort
Pre-Operator
Weak abort

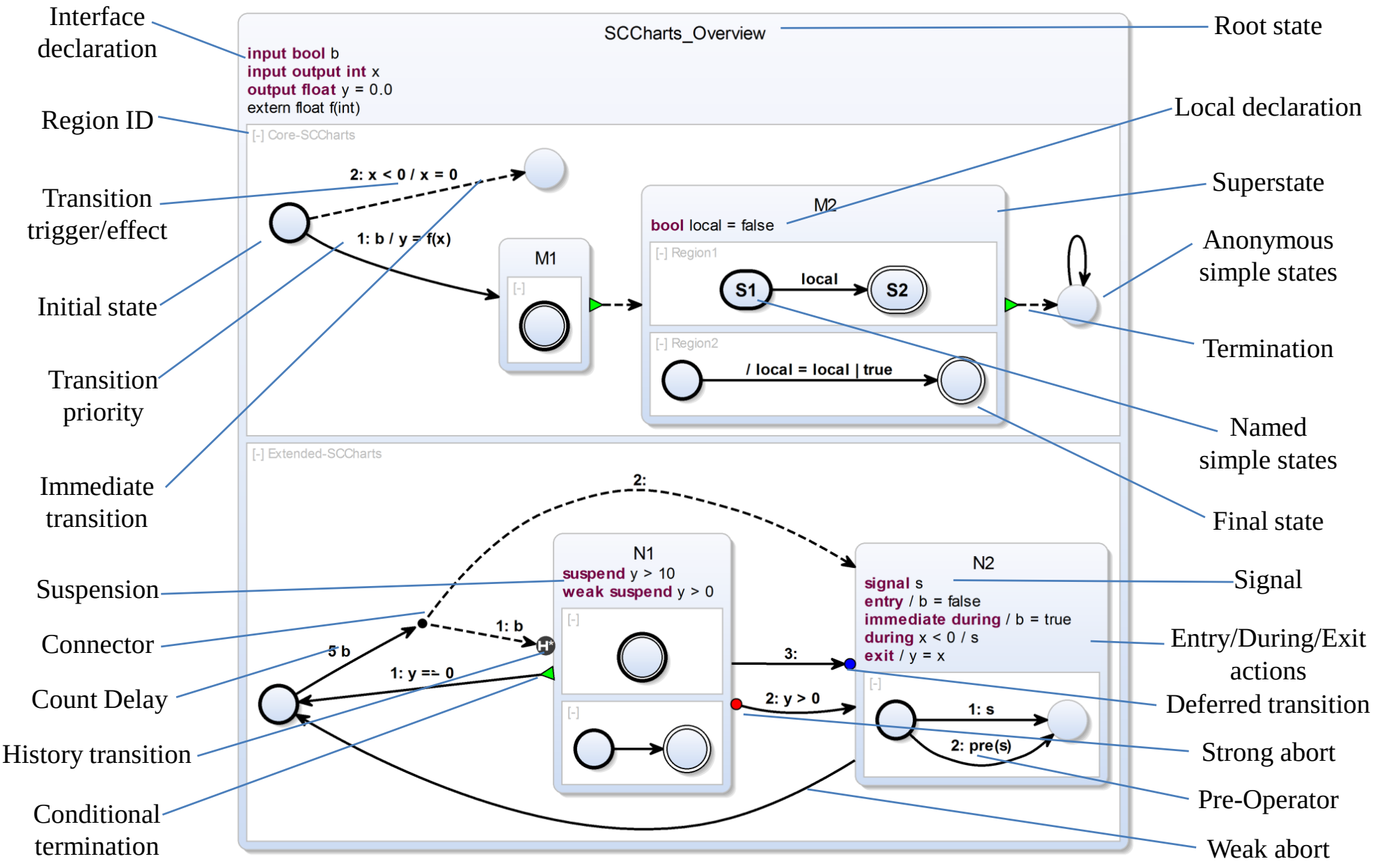

Interface declaration
Region ID
Transition trigger/effect
Initial state
Transition priority
Immediate transition
Connector
Count Delay
SCCharts_Overview
input bool b
input output int x
output float y = 0.0
extern float f(int)
[-] Core-SCCharts
2: x < 0 / x = 0
1: b / y = f(x)
M1
M2
bool local = false
[-] Region1
S1
local
S2
[-] Region2
/ local = local | true
[-] Extended-SCCharts
2:
1: b
5 b
1: y == 0
N1
suspend y > 10
weak suspend y > 0
3:
2: y > 0
N2
signal s
entry / b = false
immediate during / b = true
during x < 0 / s
exit / y = x
1: s
2: pre(s)
Root state
Local declaration
Superstate
Anonymous simple states
Termination
Named simple states
Final state
Signal
Strong abort
Pre-Operator
Weak abort


Interface declaration
Region ID
Transition trigger/effect
Initial state
Transition priority
Immediate transition
Connector
Count Delay
SCCharts_Overview
input bool b
input output int x
output float y = 0.0
extern float f(int)
[-] Core-SCCharts
2: x < 0 / x = 0
1: b / y = f(x)
M1
M2
bool local = false
[-] Region1
S1
local
S2
[-] Region2
/ local = local | true
[-] Extended-SCCharts
2:
1: b
5 b
1: y == 0
N1
suspend y > 10
weak suspend y > 0
3:
2: y > 0
N2
signal s
entry / b = false
immediate during / b = true
during x < 0 / s
exit / y = x
1: s
2: pre(s)
Root state
Local declaration
Superstate
Anonymous simple states
Termination
Named simple states
Final state
Signal
Deferred transition
Strong abort
Pre-Operator
Weak abort

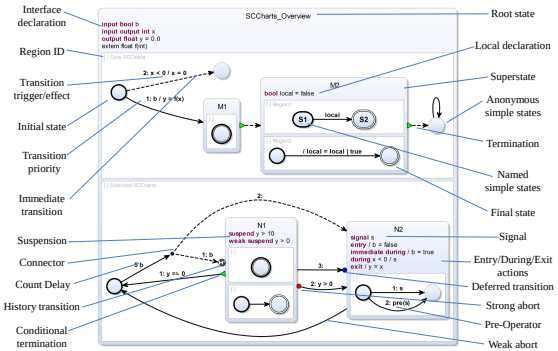

Interface declaration
Region ID
Transition trigger/effect
Initial state
Transition priority
Immediate transition
Connector
Count Delay
Conditional termination
Root state
Local declaration
Superstate
Anonymous simple states
Termination
Named simple states
Final state
Signal
Deferred transition
Strong abort
Pre-Operator
Weak abort
SCCharts_Overview
input bool b
input output int x
output float y = 0.0
extern float f(int)
[-] Core-SCCharts
2: x < 0 / x = 0
1: b / y = f(x)
M1
M2
bool local = false
[-] Region1
S1
local
S2
[-] Region2
/ local = local | true
[-] Extended-SCCharts
2:
1: b
5 b
1: y == 0
N1
suspend y > 10
weak suspend y > 0
3:
2: y > 0
N2
signal s
entry / b = false
immediate during / b = true
during x < 0 / s
exit / y = x
1: s
2: pre(s)


Interface declaration
Region ID
Transition trigger/effect
Initial state
Transition priority
Immediate transition
Connector
Count Delay
History transition
Conditional termination
Root state
Local declaration
Superstate
Anonymous simple states
Termination
Named simple states
Final state
Signal
Deferred transition
Strong abort
Pre-Operator
Weak abort
SCCharts_Overview
input bool b
input output int x
output float y = 0.0
extern float f(int)
[-] Core-SCCharts
2: x < 0 / x = 0
1: b / y = f(x)
M1
M2
bool local = false
[-] Region1
S1
local
S2
[-] Region2
/ local = local | true
[-] Extended-SCCharts
2:
1: b
5 b
1: y == 0
N1
suspend y > 10
weak suspend y > 0
3:
2: y > 0
N2
signal s
entry / b = false
immediate during / b = true
during x < 0 / s
exit / y = x
1: s
2: pre(s)


Interface declaration
Region ID
Transition trigger/effect
Initial state
Transition priority
Immediate transition
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
SCCharts_Overview
input bool b
input output int x
output float y = 0.0
extern float f(int)
[-] Core-SCCharts
2: x < 0 / x = 0
1: b / y = f(x)
M1
M2
bool local = false
[-] Region1
S1
local
S2
[-] Region2
/ local = local | true
[-] Extended-SCCharts
2:
1: b
5 b
1: y == 0
N1
suspend y > 10
weak suspend y > 0
3:
2: y > 0
N2
signal s
entry / b = false
immediate during / b = true
during x < 0 / s
exit / y = x
1: s
2: pre(s)

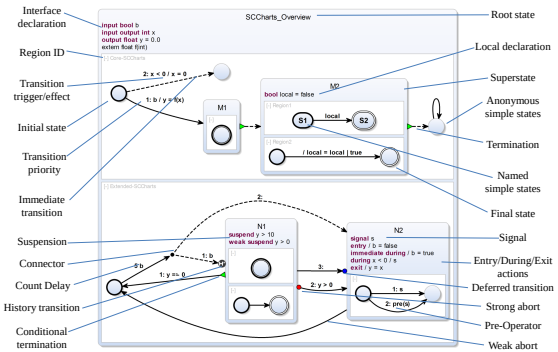

Interface declaration
Region ID
Transition trigger/effect
Initial state
Transition priority
Immediate transition
Suspension
Connector
Count Delay
History transition
Conditional termination
SCCharts_Overview
input bool b
input output int x
output float y = 0.0
extern float f(int)
[-] Core-SCCharts
2: x < 0 / x = 0
1: b / y = f(x)
M1
M2
bool local = false
[-] Region1
S1
local
S2
[-] Region2
/ local = local | true
[-] Extended-SCCharts
2:
1: b
5 b
1: y == 0
N1
suspend y > 10
weak suspend y > 0
3:
2: y > 0
N2
signal s
entry / b = false
immediate during / b = true
during x < 0 / s
exit / y = x
1: s
2: pre(s)
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

Motivation for Core SCCharts



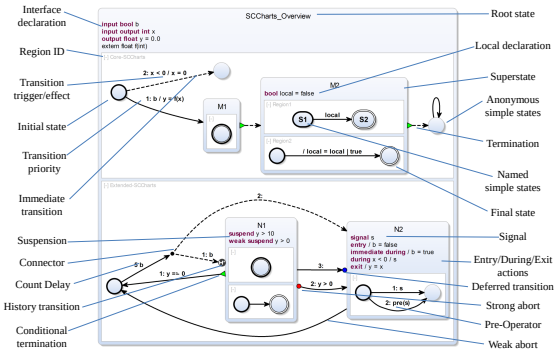
- **Observation I**

Motivation for Core SCCharts



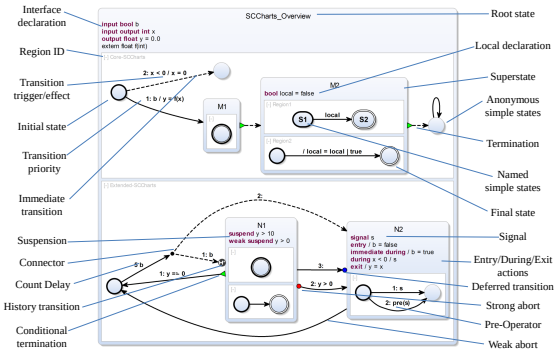
- **Observation I**: Numerous features

Motivation for Core SCCharts



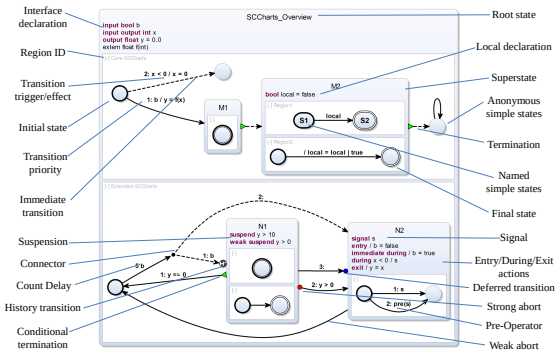
- **Observation I**: Numerous features
  - ☺ Compactness / readability of models

Motivation for Core SCCharts



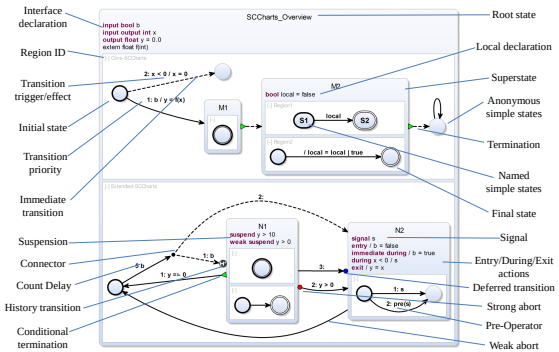
- **Observation I**: Numerous features
  - ☺ Compactness / readability of models
  - ☹ Steeper learning curve

Motivation for Core SCCharts



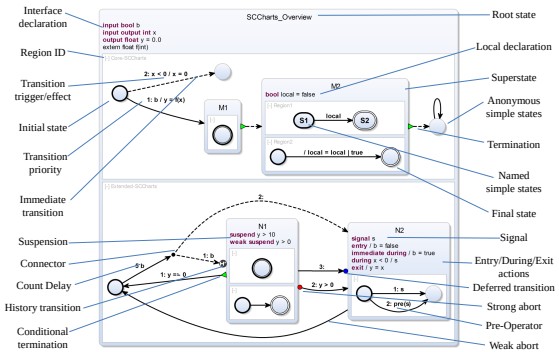
- **Observation I**: Numerous features
  - ☺ Compactness / readability of models
  - ☹ Steeper learning curve
  - ☹ Direct compilation & verification more complex

Motivation for Core SCCharts



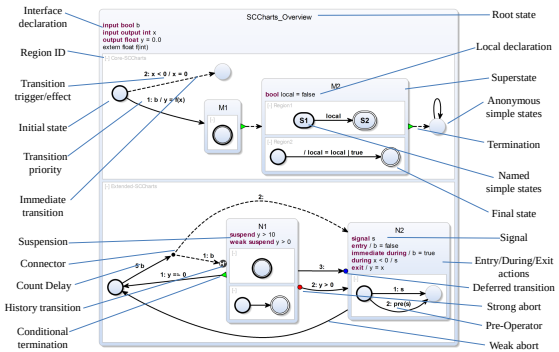
- **Observation I**: Numerous features
  - ☺ Compactness / readability of models
  - ☹ Steeper learning curve
  - ☹ Direct compilation & verification more complex
- **Observation II**

Motivation for Core SCCharts



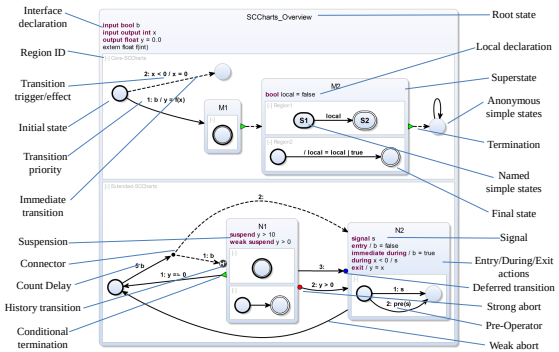
- **Observation I**: Numerous features
  - ☺ Compactness / readability of models
  - ☹ Steeper learning curve
  - ☹ Direct compilation & verification more complex
- **Observation II**: Various features can be expressed by other ones

Motivation for Core SCCharts



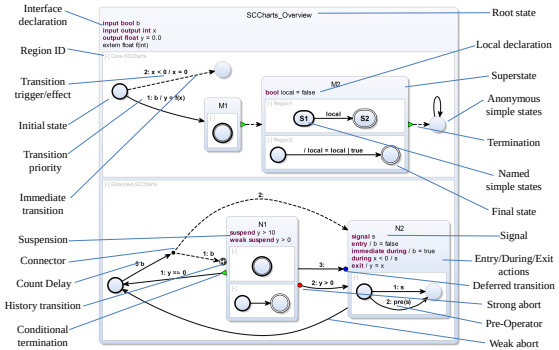
- **Observation I**: Numerous features
  - ☺ Compactness / readability of models
  - ☹ Steeper learning curve
  - ☹ Direct compilation & verification more complex
- **Observation II**: Various features can be expressed by other ones
- **Consequence**

Motivation for Core SCCharts



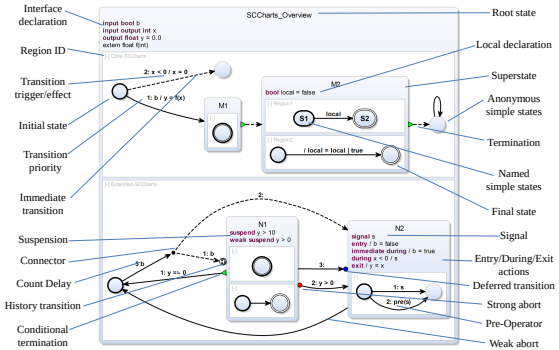
- **Observation I**: Numerous features
  - ☺ Compactness / readability of models
  - ☹ Steeper learning curve
  - ☹ Direct compilation & verification more complex
- **Observation II**: Various features can be expressed by other ones
- **Consequence**: ⇒ Define extended features by means of base features

Motivation (Cont'd)



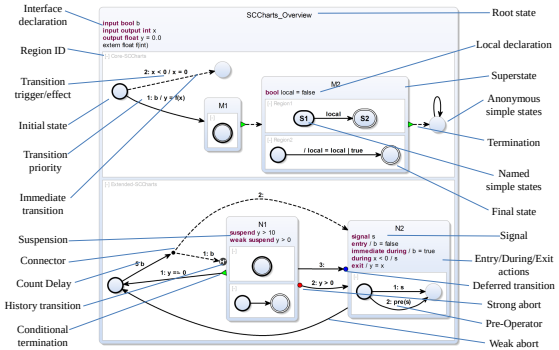
- **Advantages**

Motivation (Cont'd)



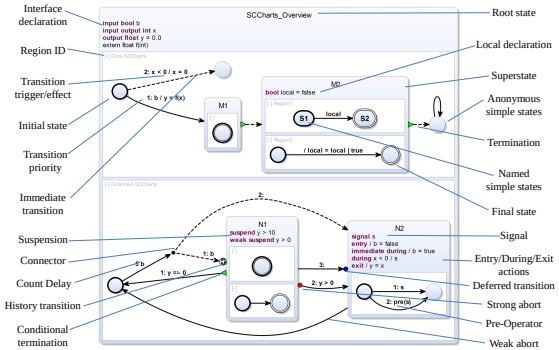
- **Advantages**:
  - Minimal base language (Core SCCharts)
    + advanced features (Extended SCCharts)

Motivation (Cont'd)



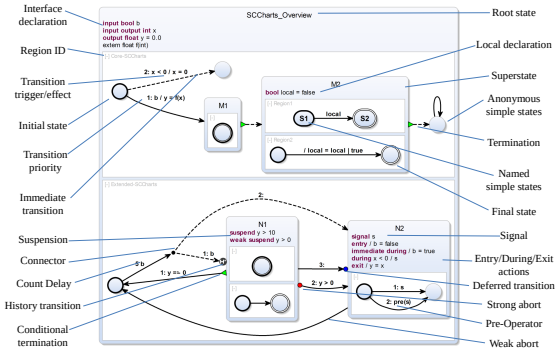
- **Advantages**:
  - Minimal base language (Core SCCharts)
    + advanced features (Extended SCCharts)
    - Similar to Esterel Kernel Statements & Statement Expansion

Motivation (Cont'd)



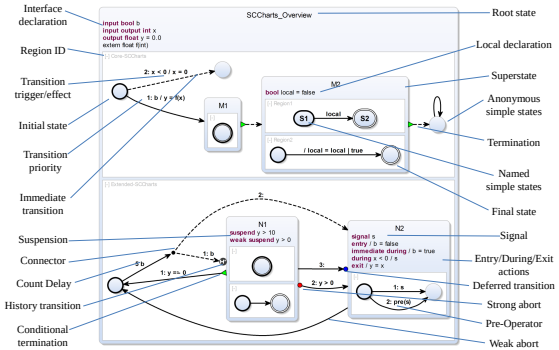
- **Advantages**:
  - Minimal base language (Core SCCharts)
    + advanced features (Extended SCCharts)
    - Similar to Esterel Kernel Statements & Statement Expansion
  - Advanced features are *syntactic sugar*

Motivation (Cont'd)



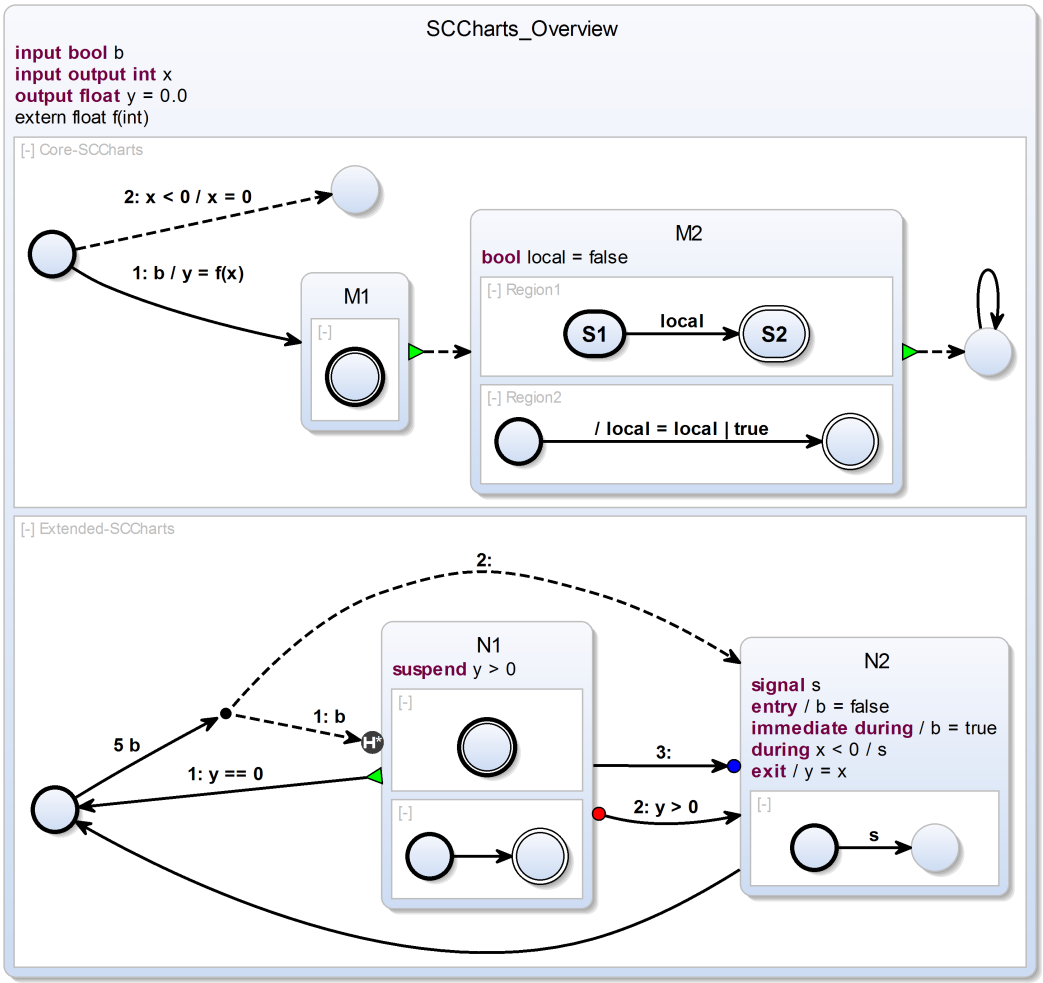
- **Advantages**:
  - Minimal base language (Core SCCharts)
    + advanced features (Extended SCCharts)
    - Similar to Esterel Kernel Statements & Statement Expansion
  - Advanced features are *syntactic sugar*
  - Extensible

Motivation (Cont'd)

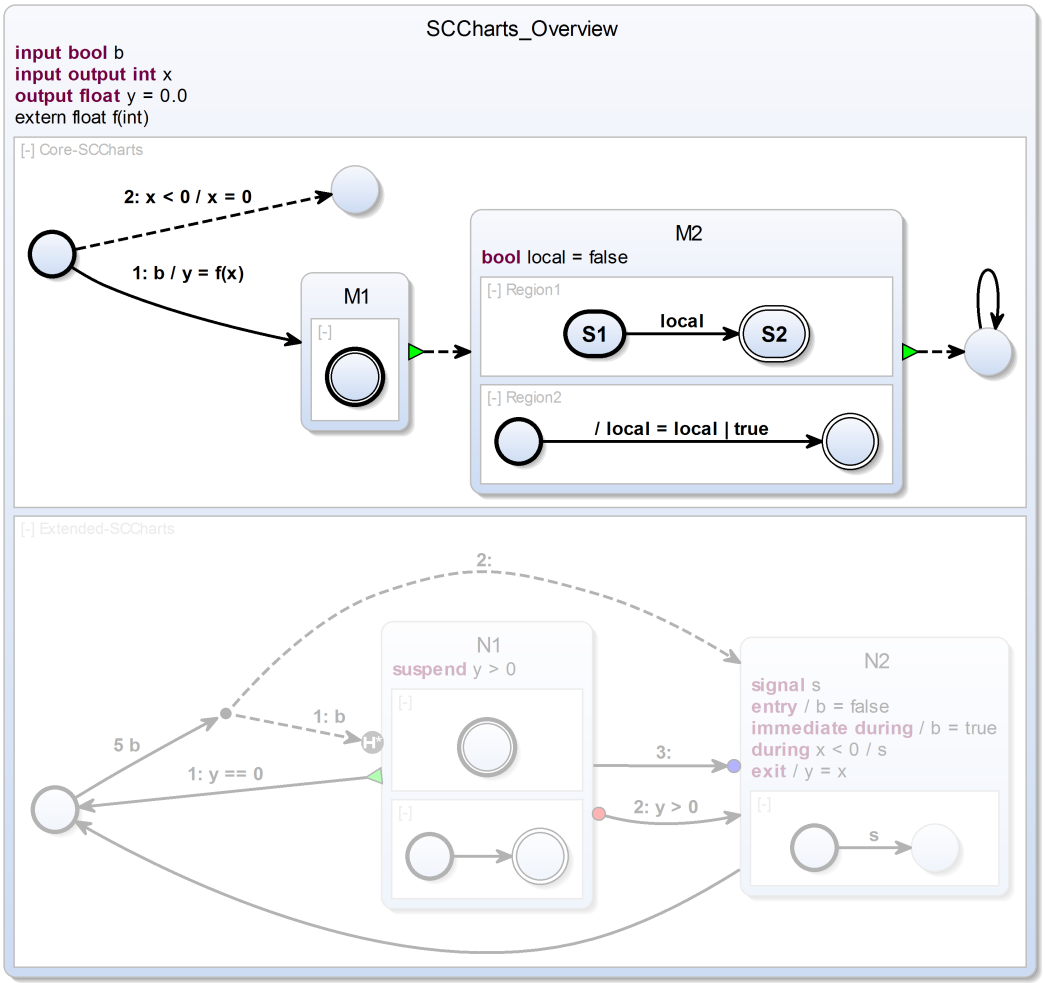


- **Advantages**:
  - Minimal base language (Core SCCharts)
    + advanced features (Extended SCCharts)
    - Similar to Esterel Kernel Statements & Statement Expansion
  - Advanced features are *syntactic sugar*
  - Extensible
  - Compilation (ongoing research)
    - Modular & extensible
    - Less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

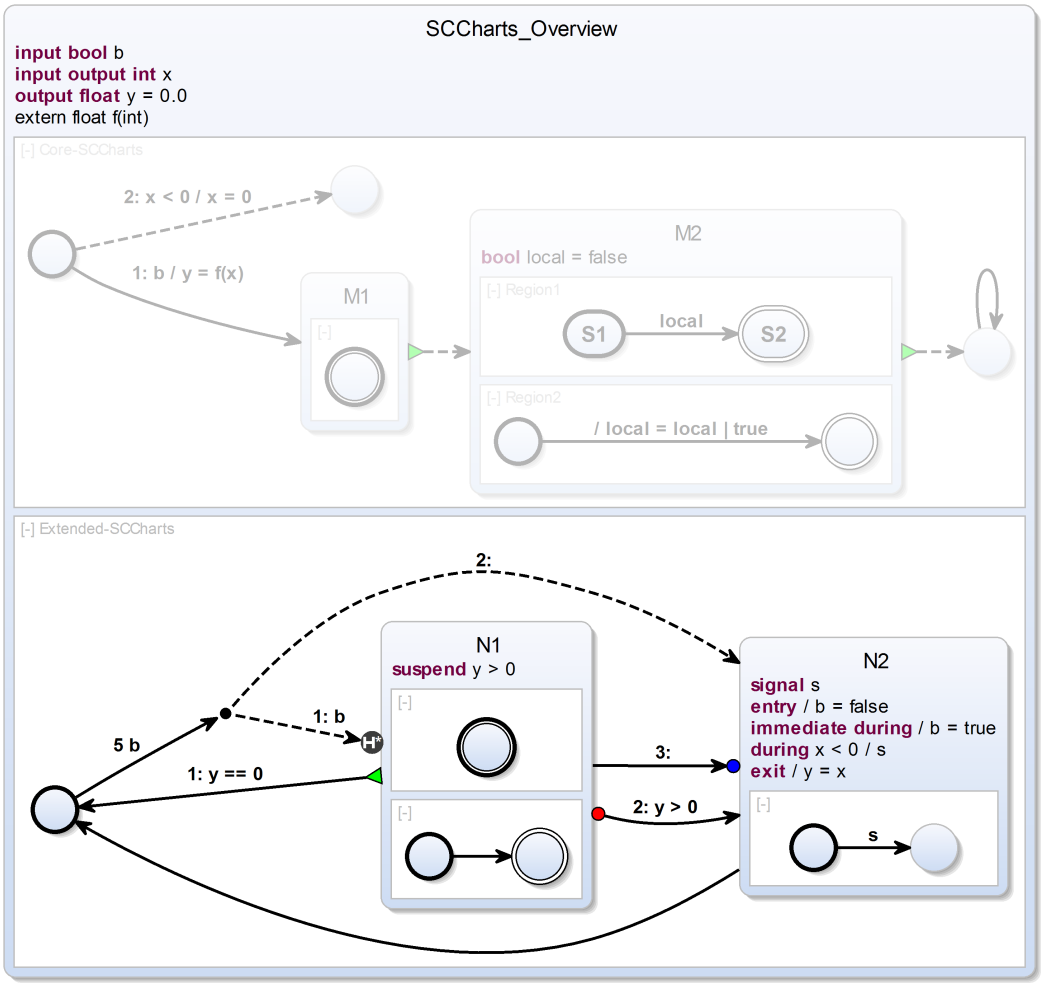
SCCharts — Core & Extended Features



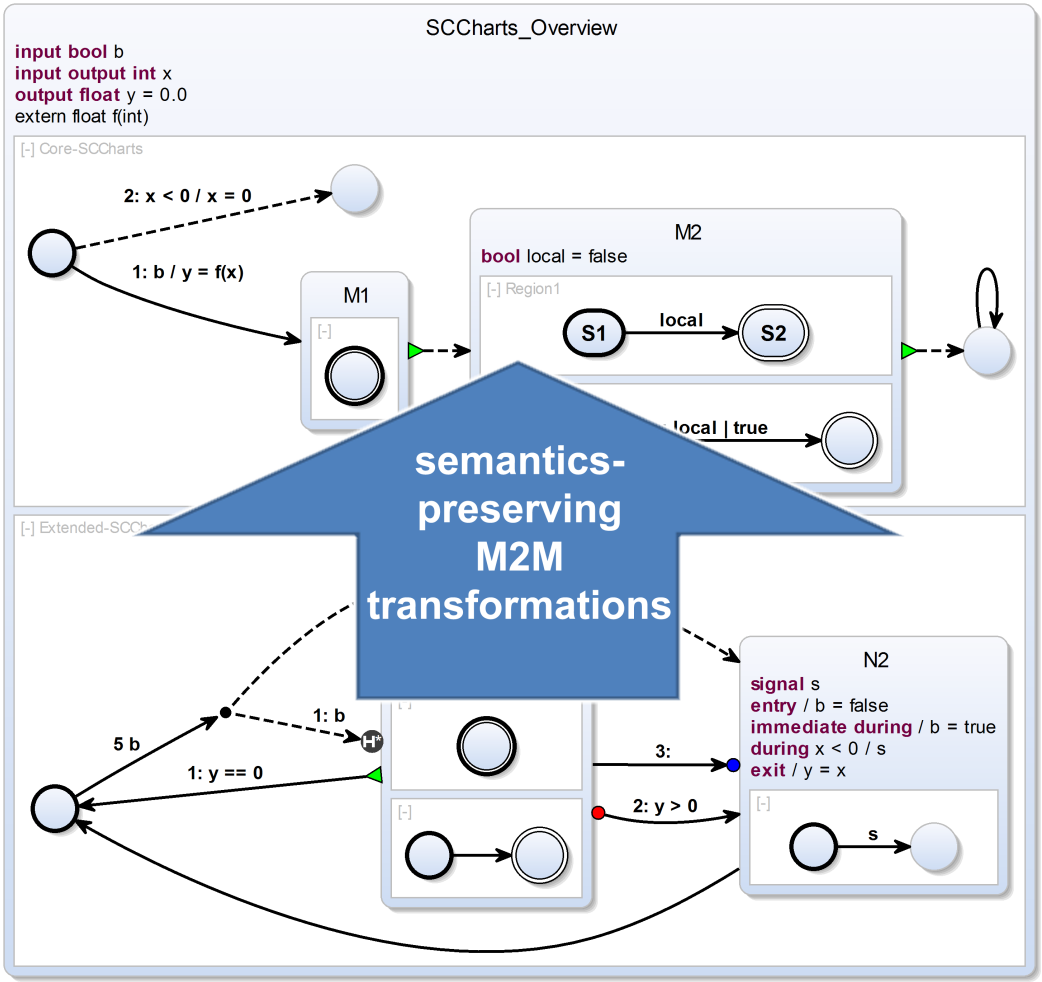
SCCharts — Core & Extended Features



SCCharts — Core & Extended Features



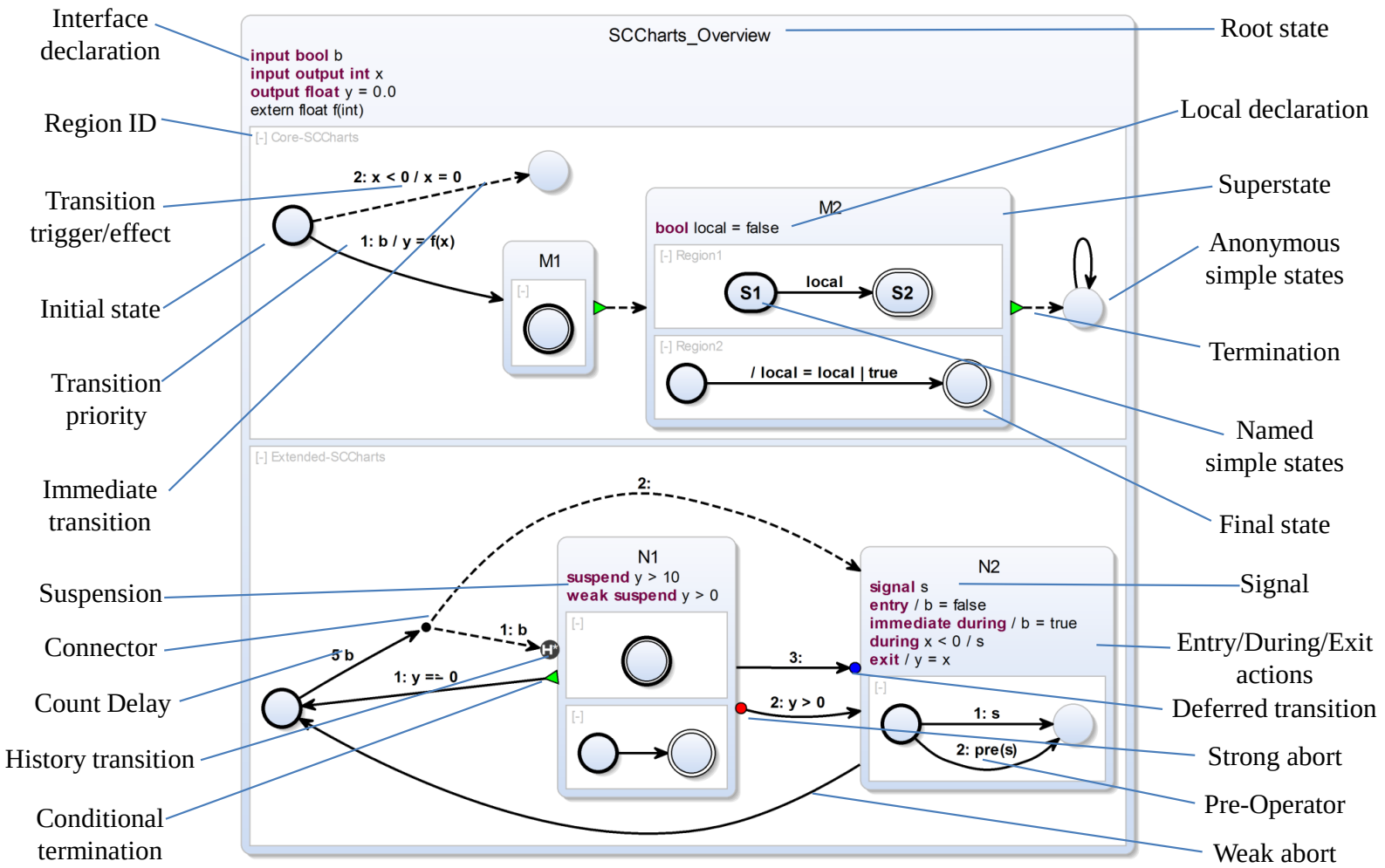
SCCharts — Core & Extended Features



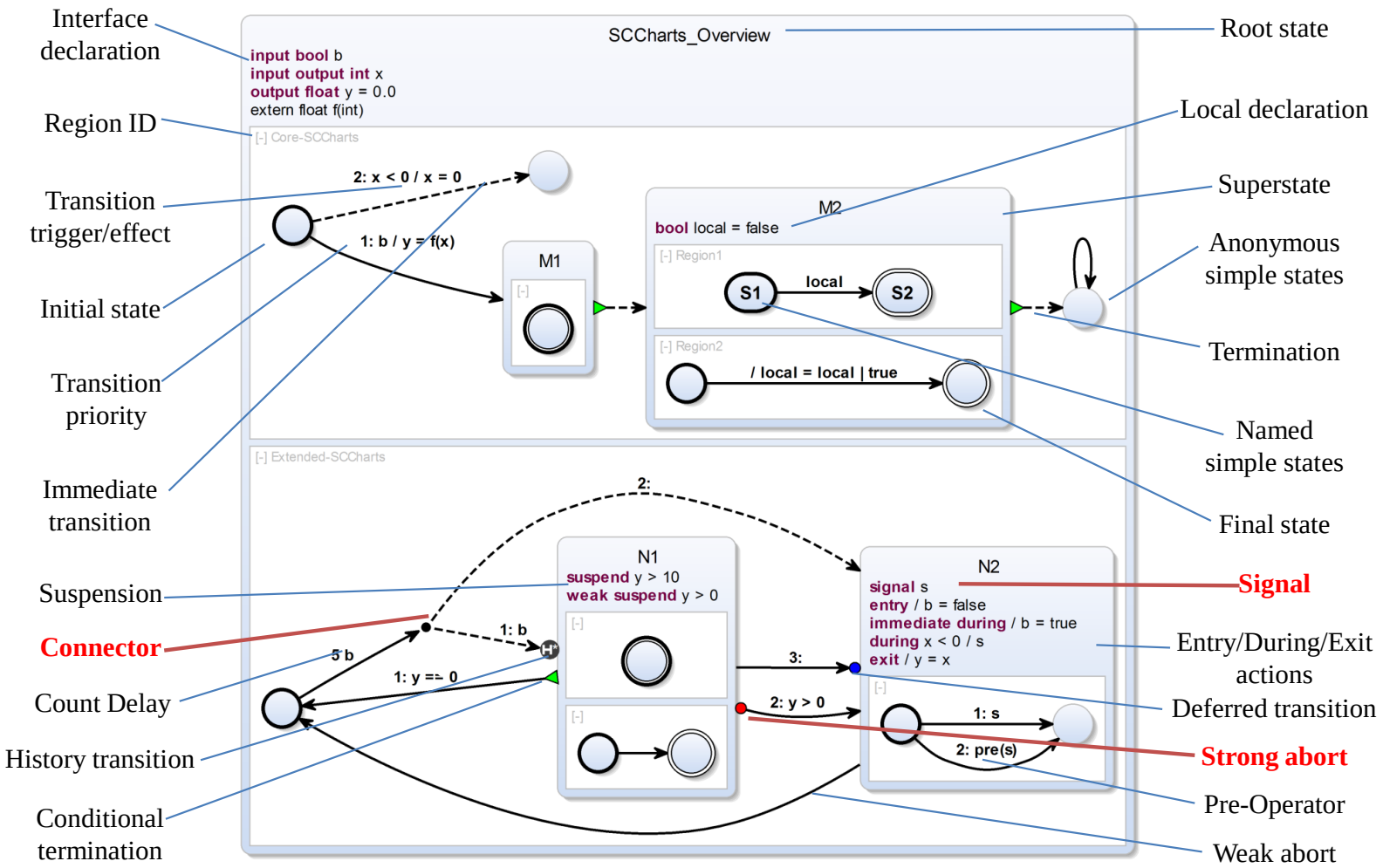
Overview

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

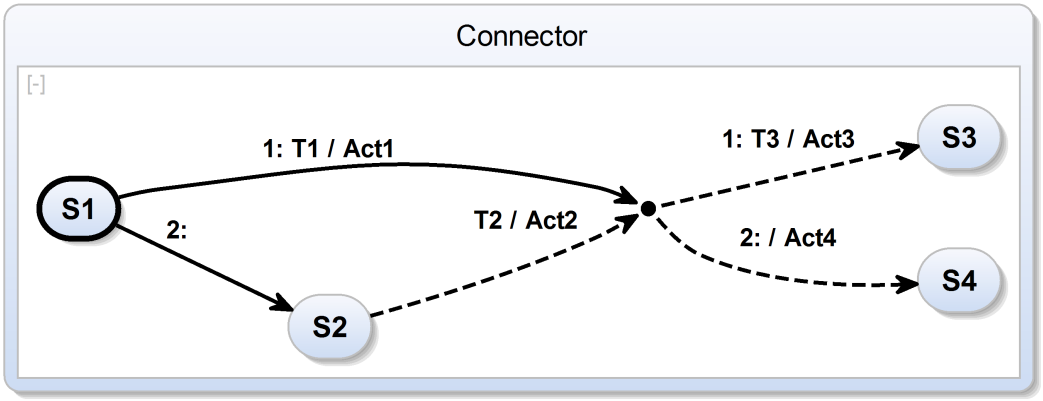
SCCharts — Core Transformations Examples



SCCharts — Core Transformations Examples

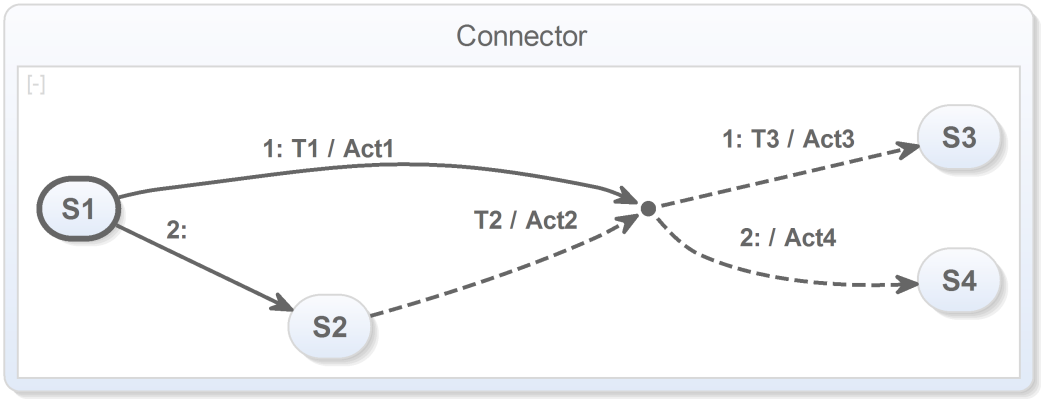


Transforming Connectors

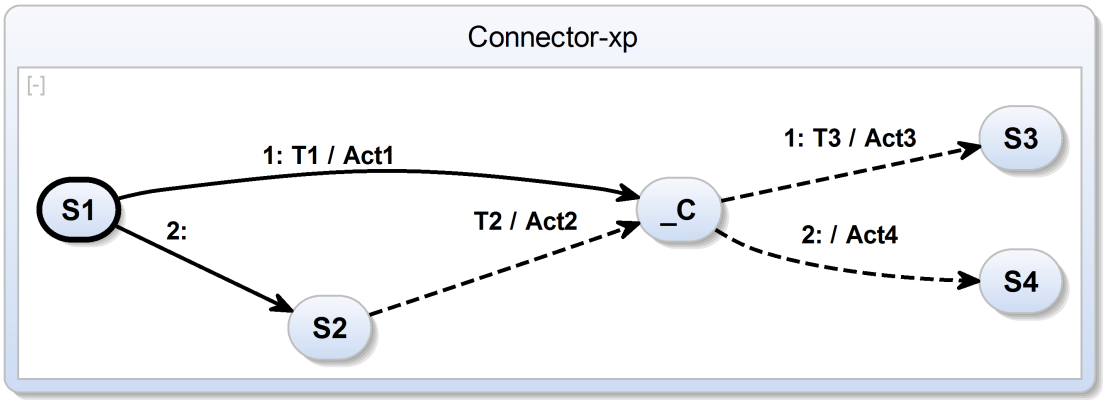


Extended SCCharts with Connectors

Transforming Connectors

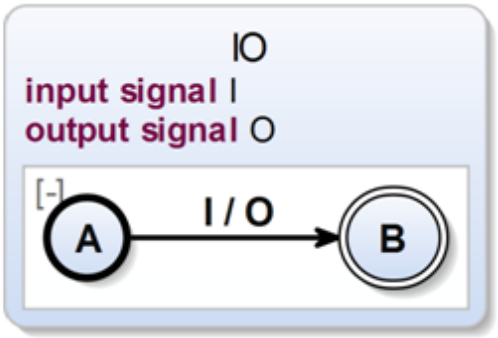


Extended SCCharts with Connectors



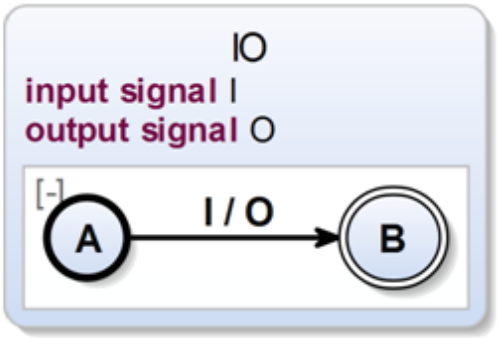
Core SCCharts without Connectors

Transforming Signals



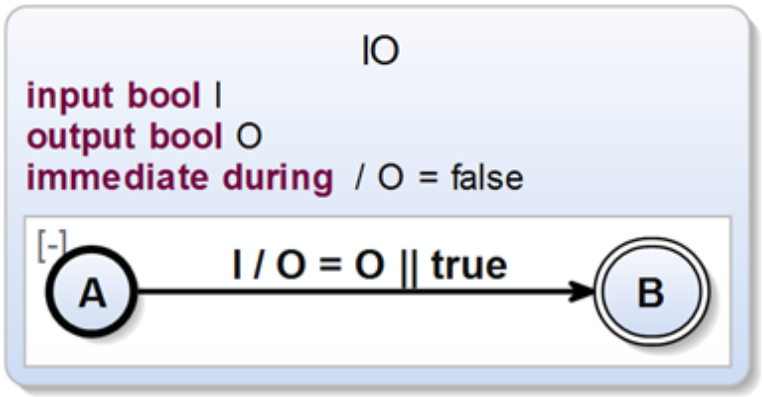
Extended SCCharts with Signals

Transforming Signals



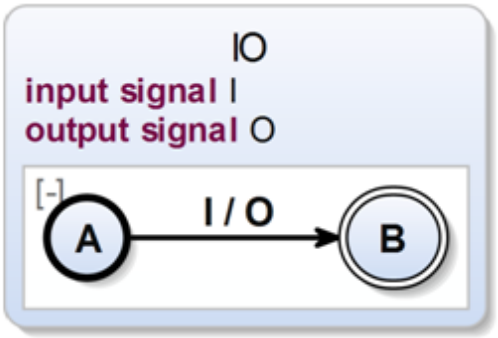
Extended SCCharts with Signals

Signal expansion



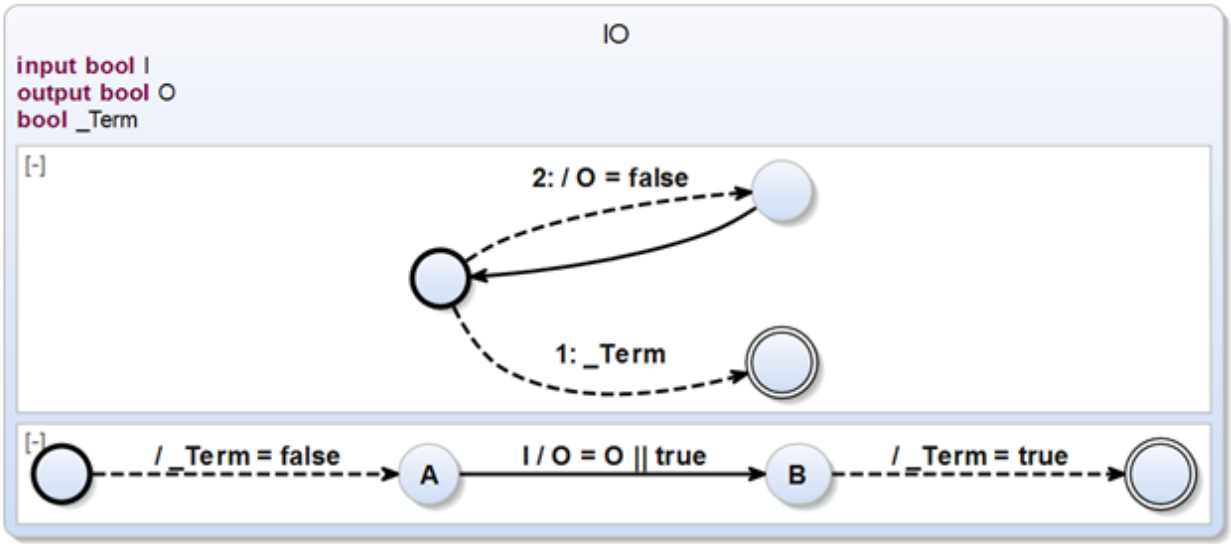
Core SCCharts with During Actions

Transforming Signals



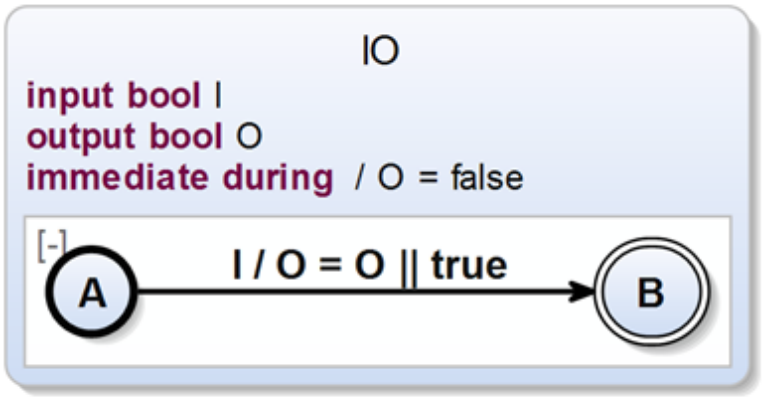
Extended SCCharts with Signals

Signal expansion



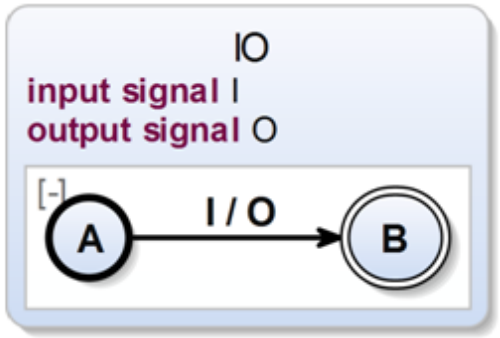
Core SCCharts only

Action expansion



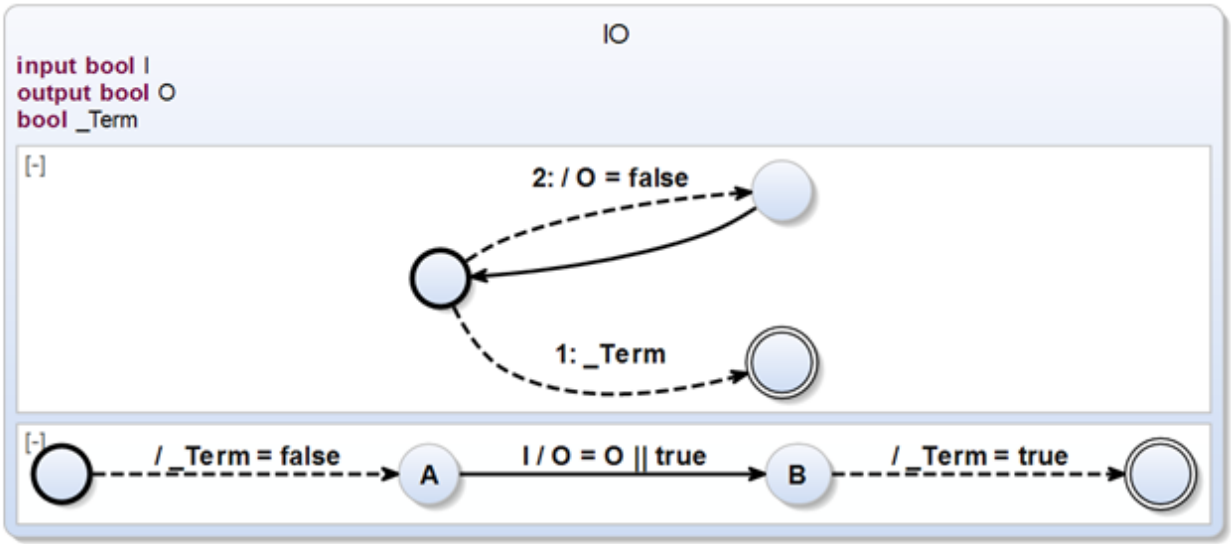
Core SCCharts with During Actions

Transforming Signals



Extended SCCharts with Signals

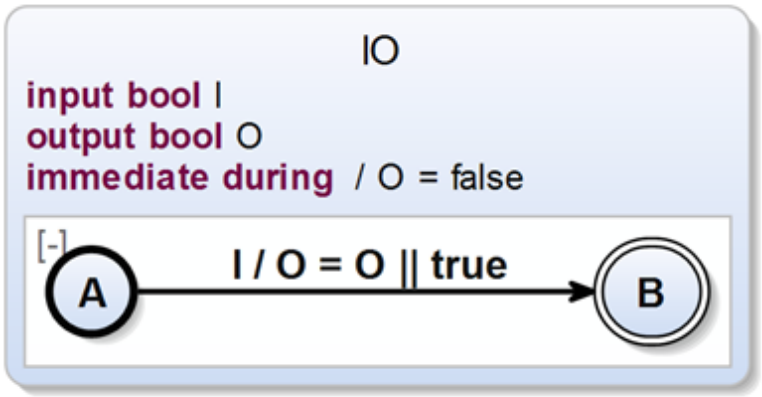
Signal expansion



Core SCCharts only

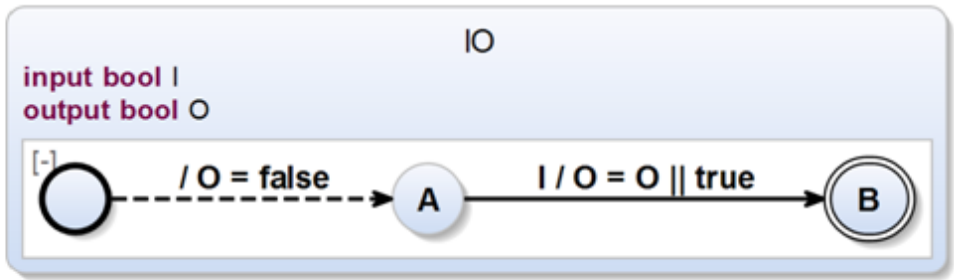
Optimization

Action expansion



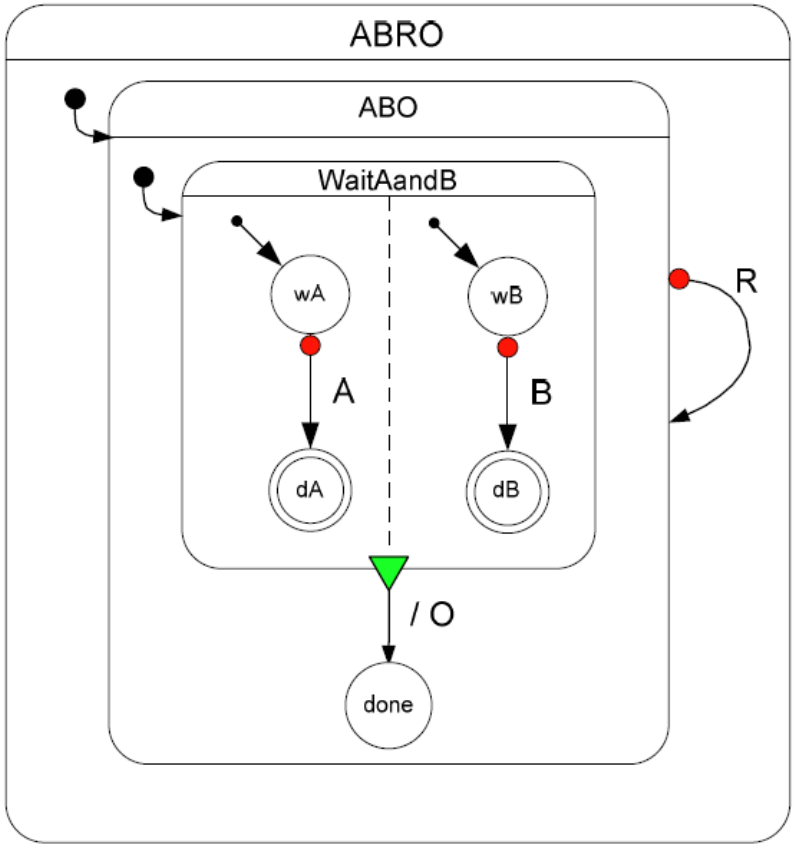
Core SCCharts with During Actions

Action expansion (alternative)



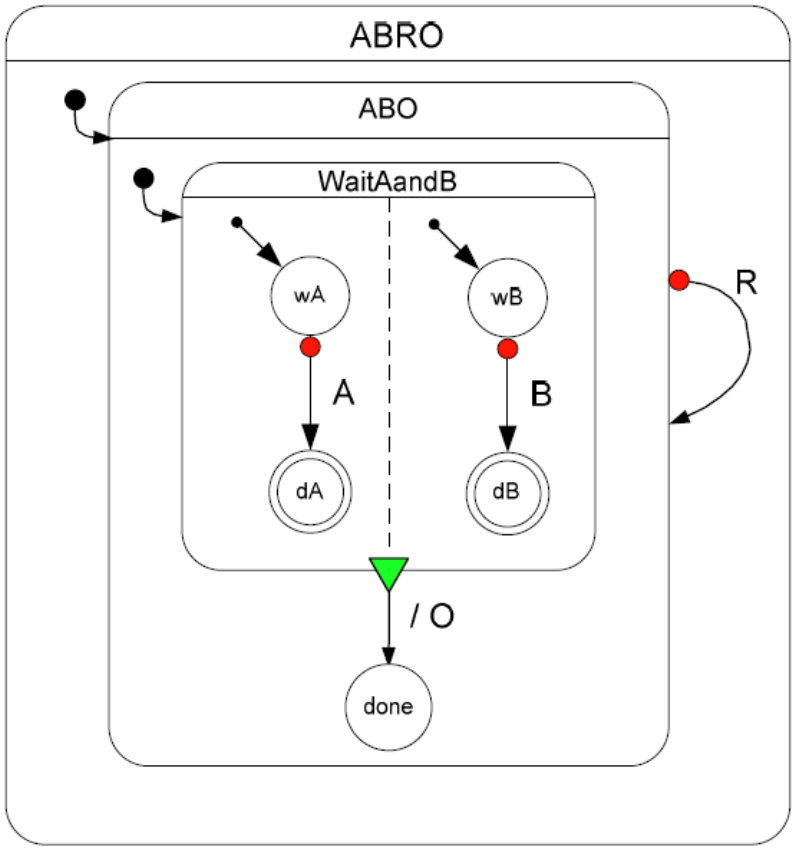
Core SCCharts only (optimized)

SyncChart and SCChart ABRO

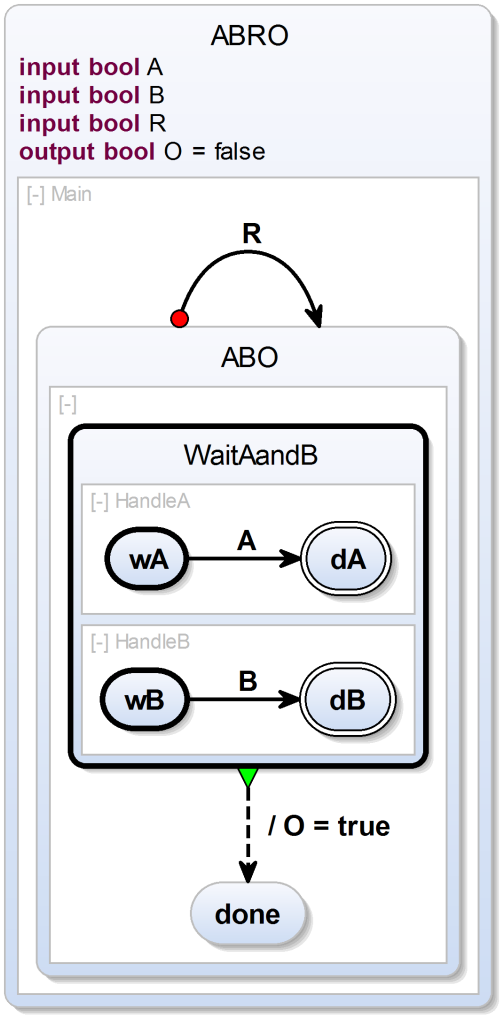


[Charles André, Semantics of SyncCharts, 2003]

SyncChart and SCChart ABRO

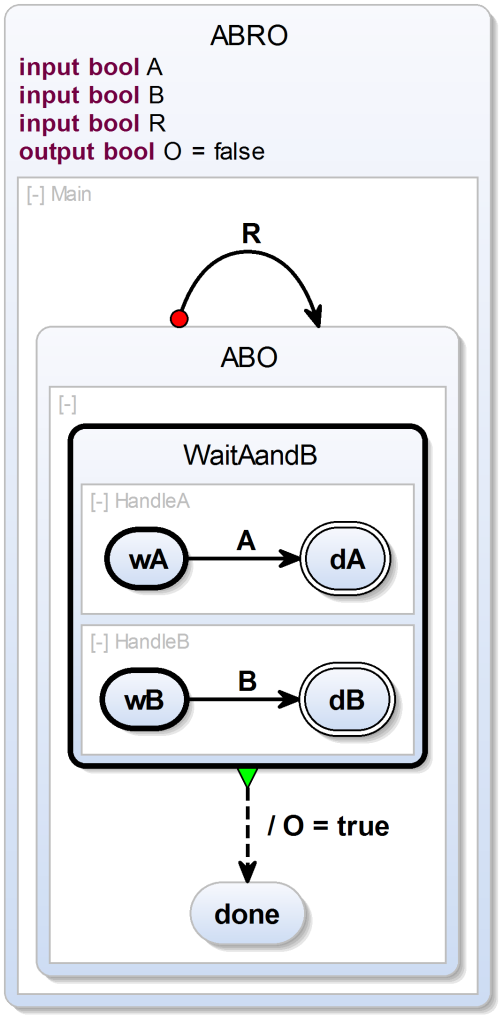


[Charles André, Semantics of SyncCharts, 2003]



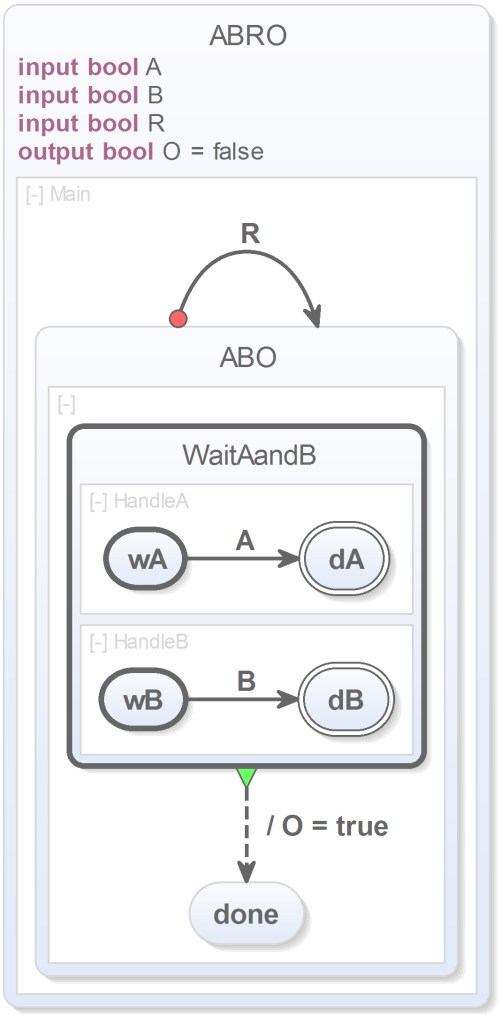
. ABRO SCChart

ABRO — Transforming Strong Aborts

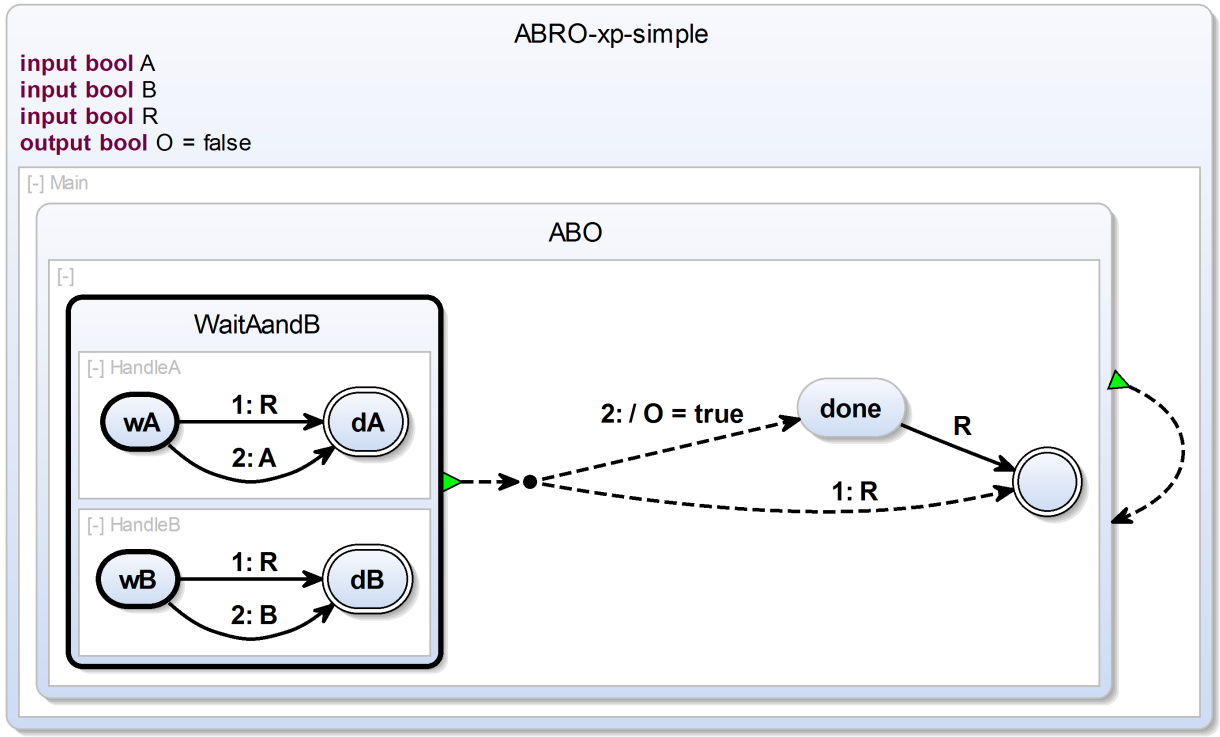


ABRO SCChart with Strong Abort

ABRO — Transforming Strong Aborts

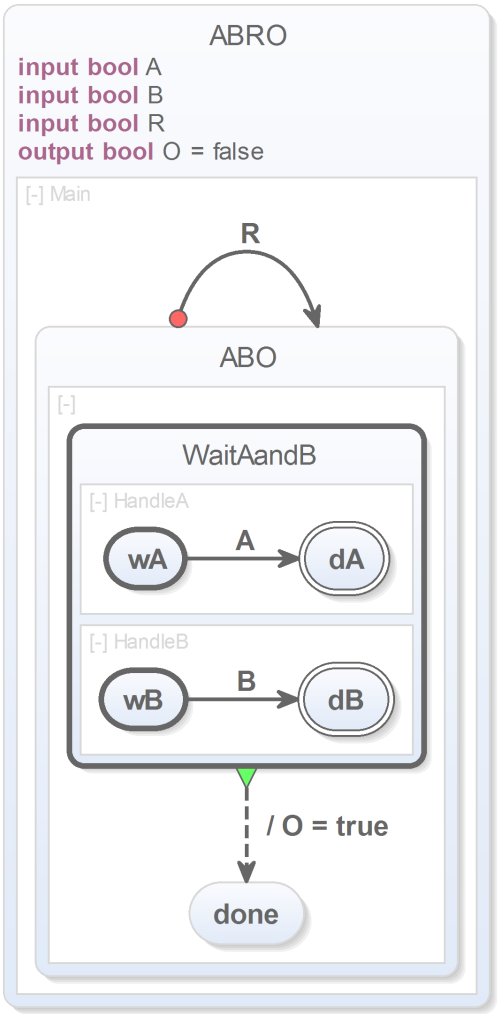


ABRO SCChart with Strong Abort

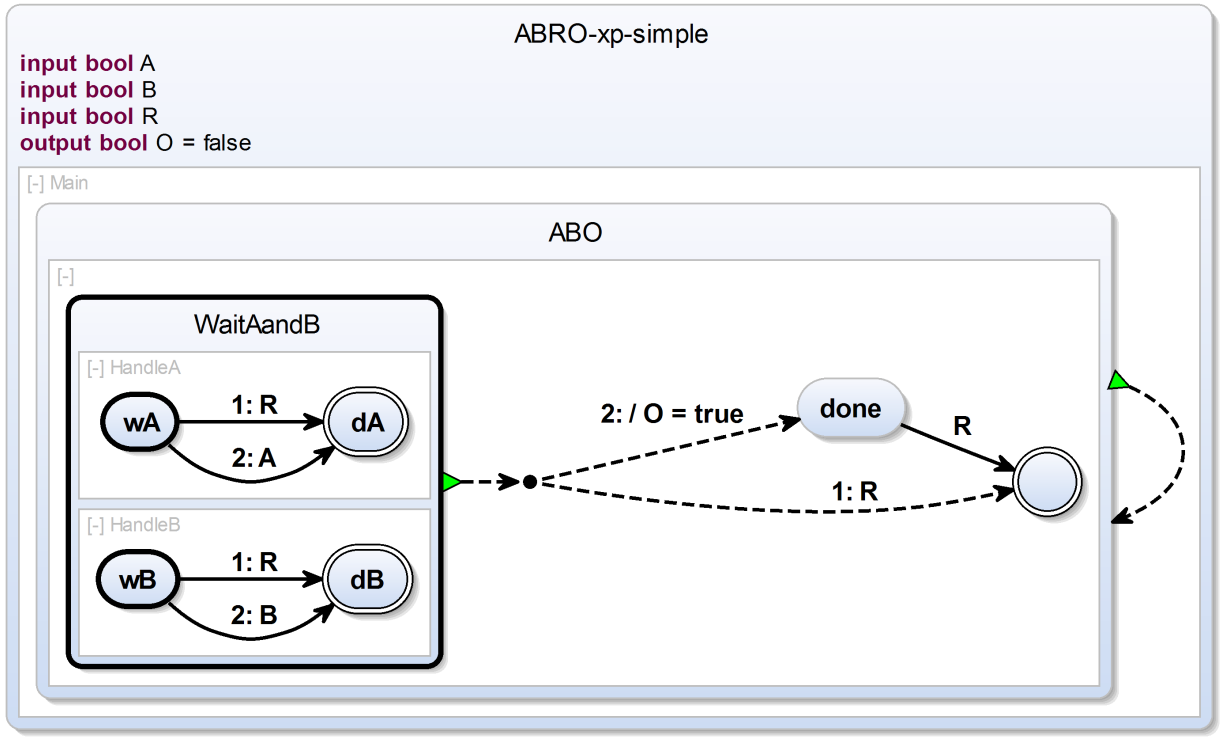


Core SCChart without Strong Abort

ABRO — Transforming Strong Aborts



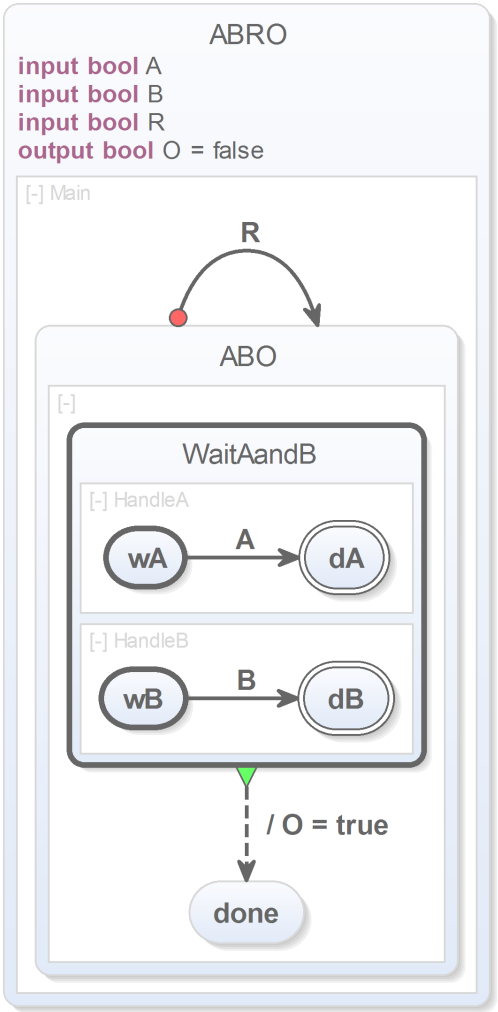
ABRO SCChart with Strong Abort



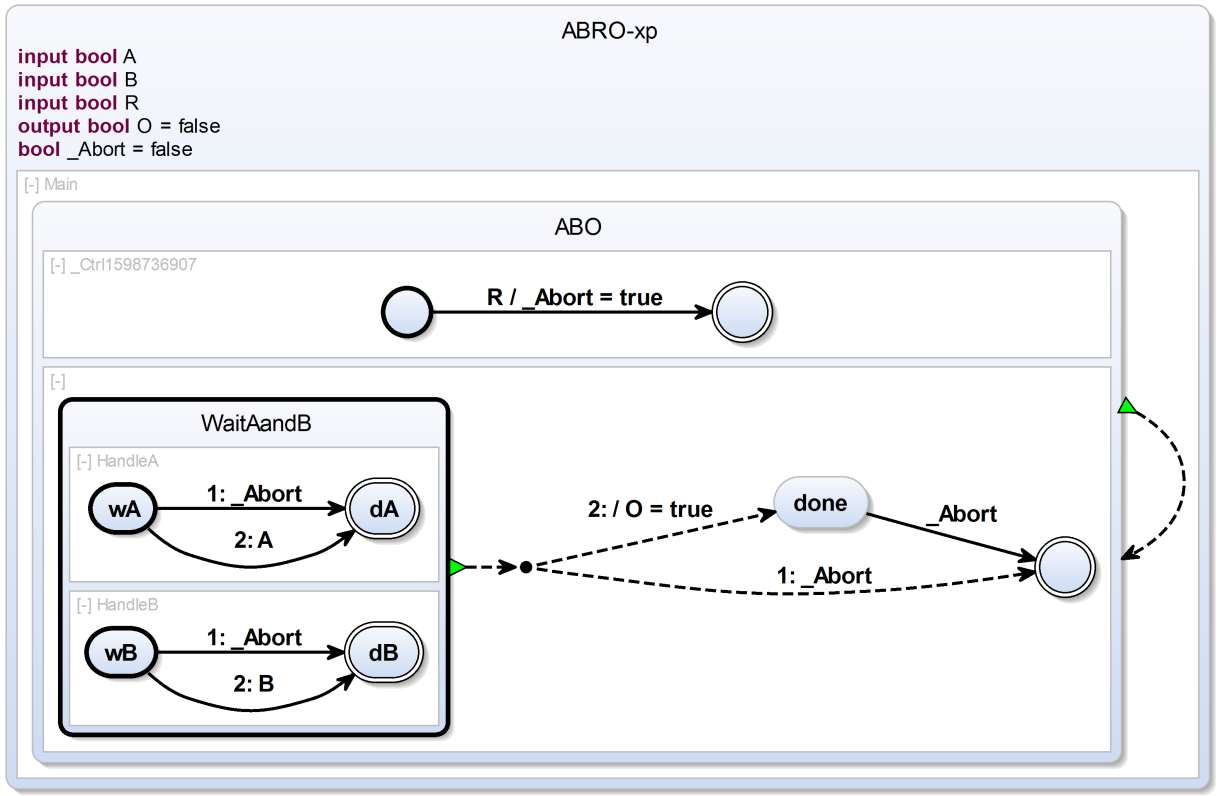
Core SCChart without Strong Abort

→ **Write-Things-Once (WTO) principle violated**

ABRO — Transforming Strong Aborts (cont'd)

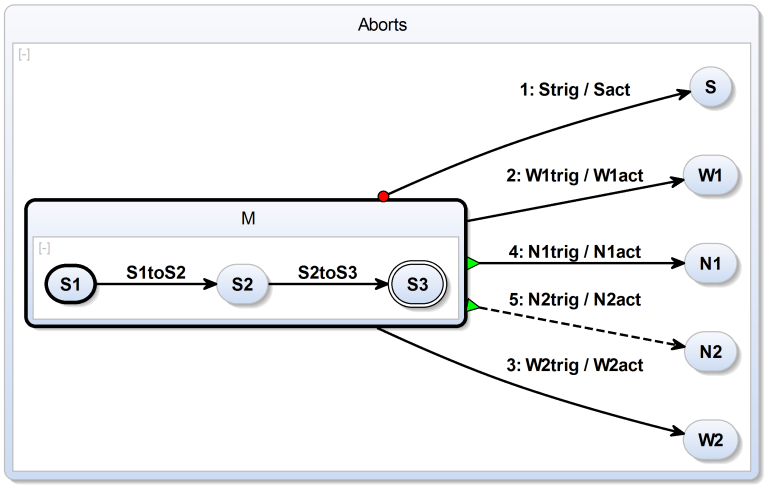


ABRO SCChart with Strong Abort



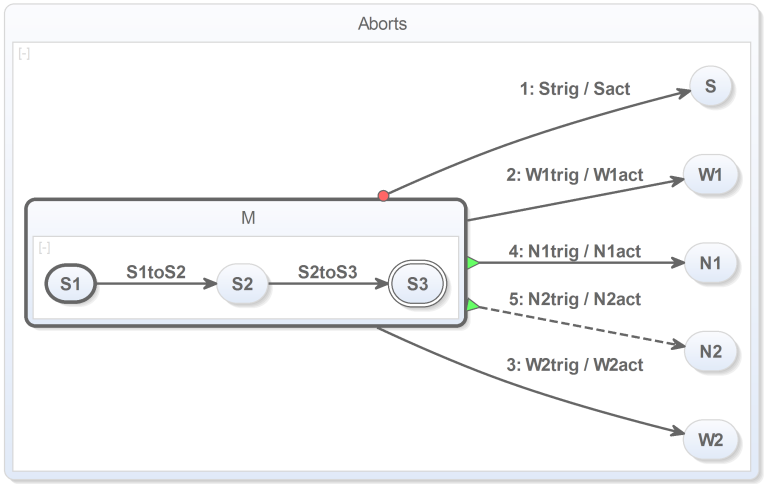
Core SCChart without Strong Abort and WTO

Transforming General Aborts

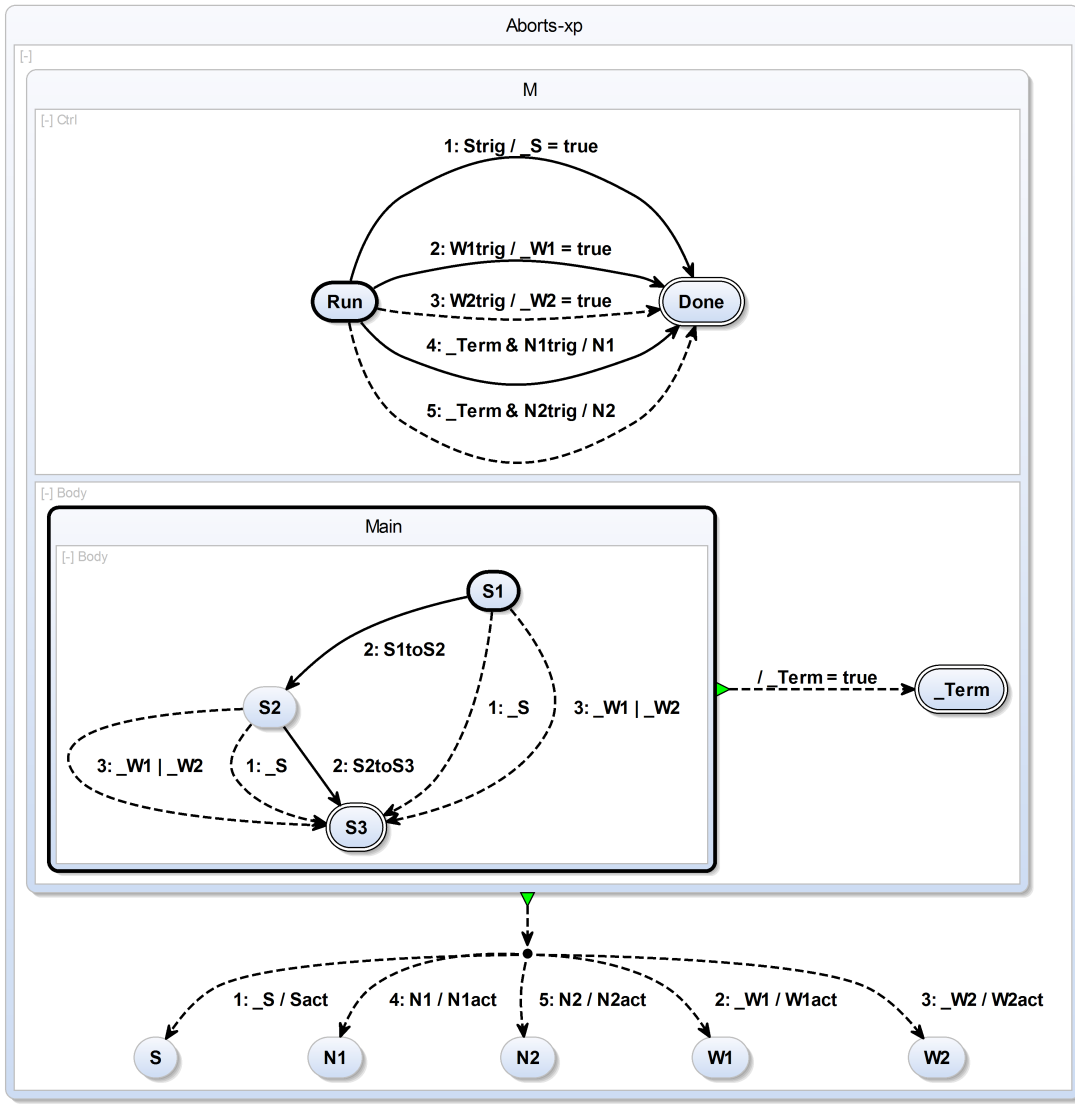


Extended SCCharts with Aborts

Transforming General Aborts



Extended SCCharts with Aborts

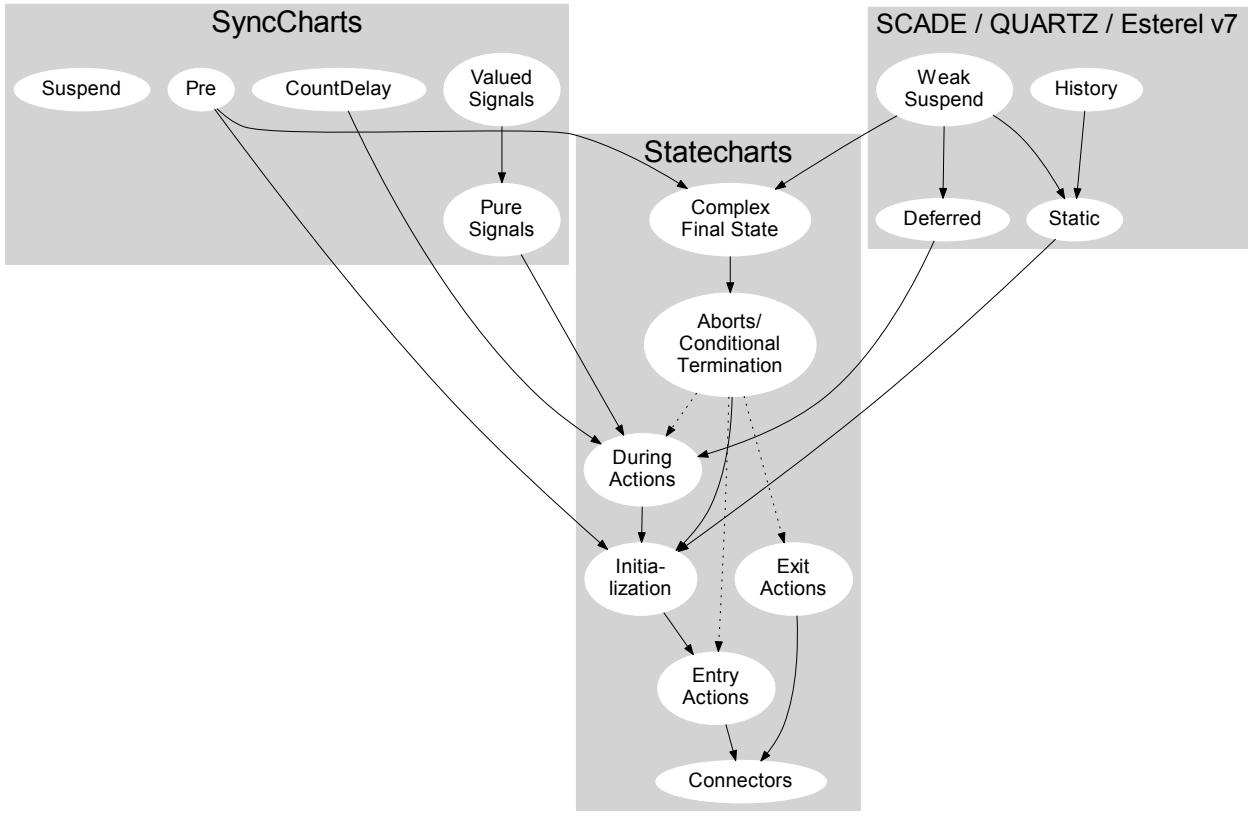


Core SCCharts with one Termination

Overview

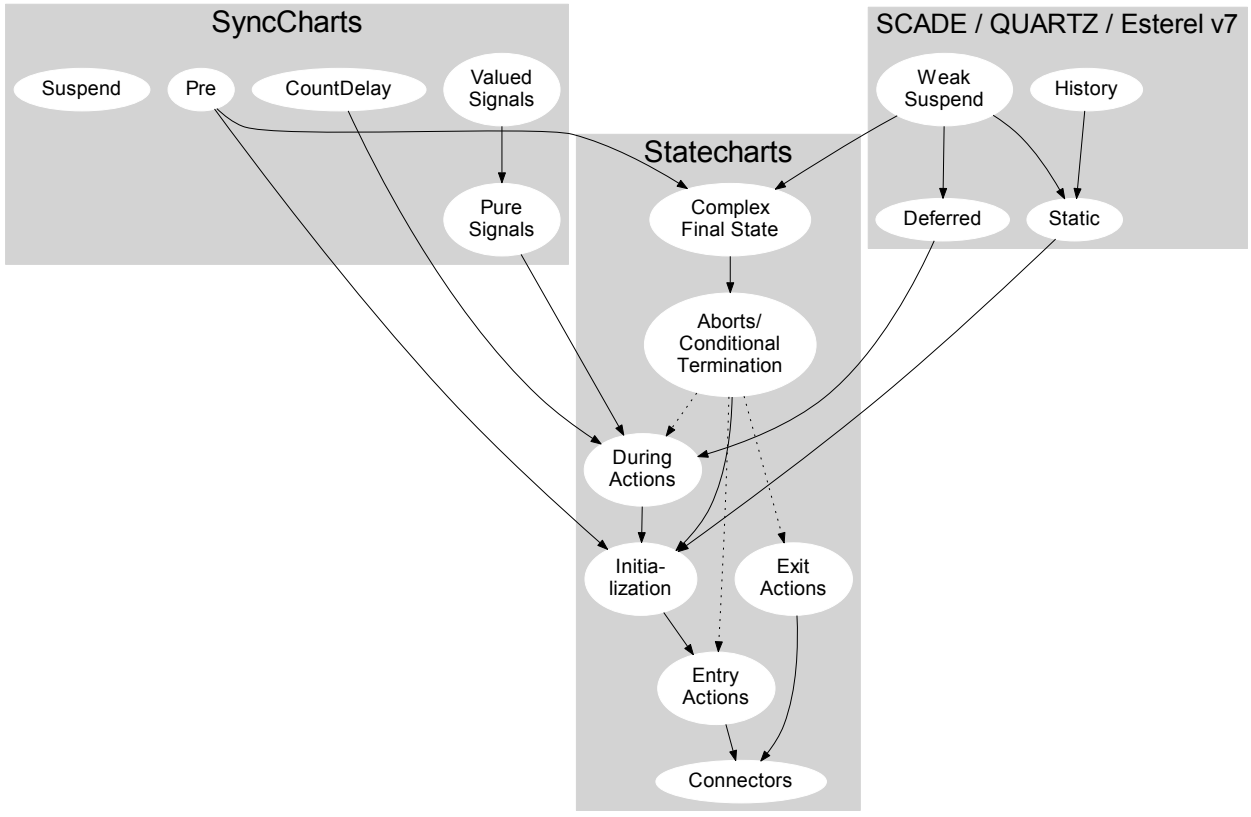
- SCCharts Overview
- Extended SCCharts → Core SCCharts
- Normalizing Core SCCharts
- Implementation in KIELER

Single-Pass Language-Driven Incremental Compilation (SLIC)



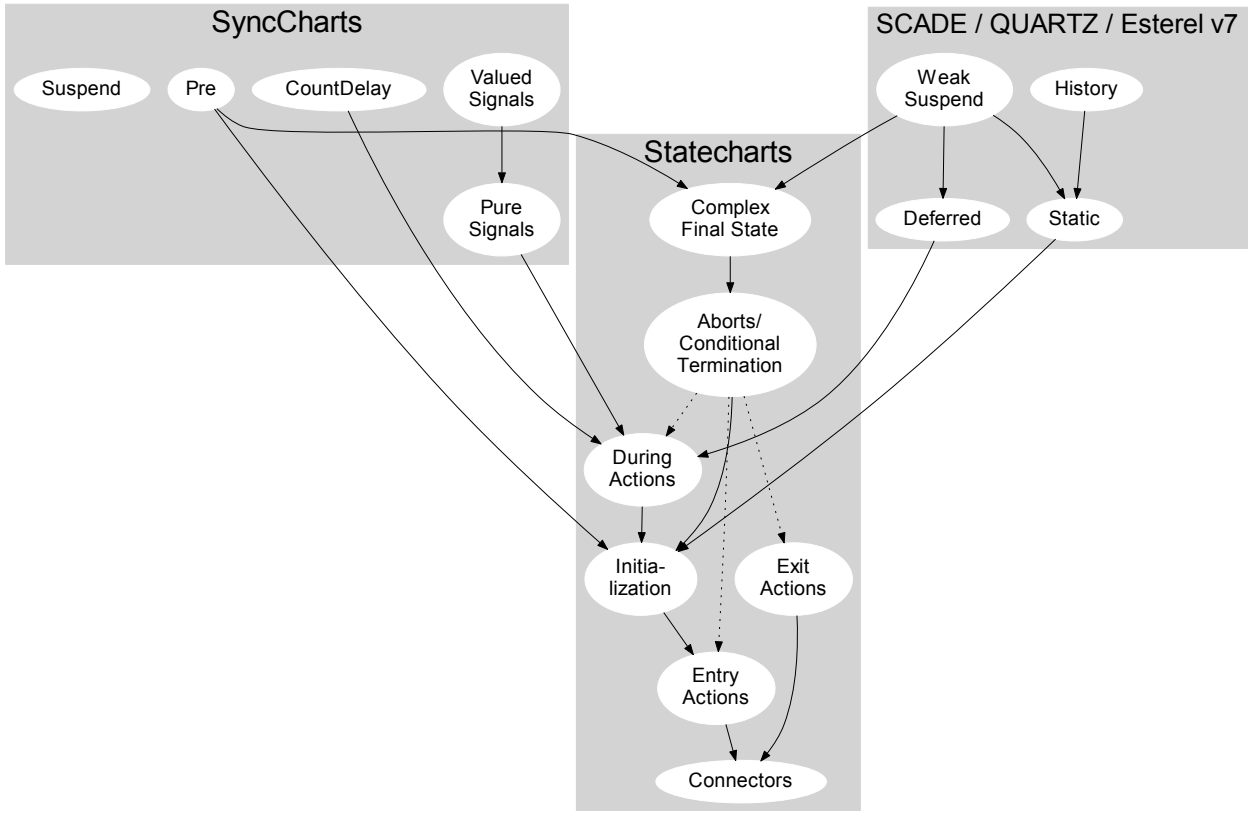
- Some core transformations will produce (use) some other extended features (solid lines)

Single-Pass Language-Driven Incremental Compilation (SLIC)



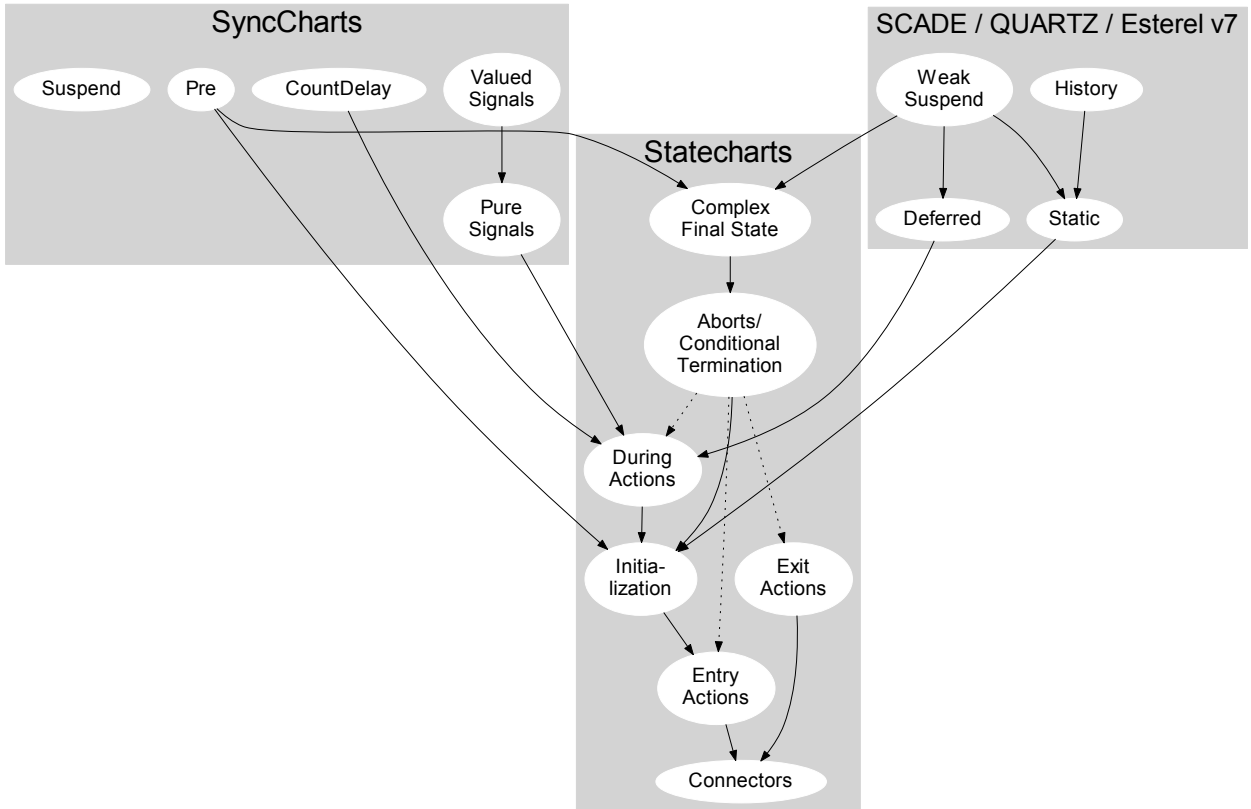
- Some core transformations will produce (use) some other extended features (solid lines)
- Other core transformations cannot handle some extended features (dashed lines)

Single-Pass Language-Driven Incremental Compilation (SLIC)



- Some core transformations will produce (use) some other extended features (solid lines)
- Other core transformations cannot handle some extended features (dashed lines)
- $\rightarrow$ Order in which core transformations are applied is important

Single-Pass Language-Driven Incremental Compilation (SLIC)



- Some core transformations will produce (use) some other extended features (solid lines)
- Other core transformations cannot handle some extended features (dashed lines)
- $\rightarrow$ Order in which core transformations are applied is important
- $\rightarrow$ Dependencies (do not have any cycle, which would be forbidden)

Normalization

- Further simplify compilation process for Core SCCharts

Normalization

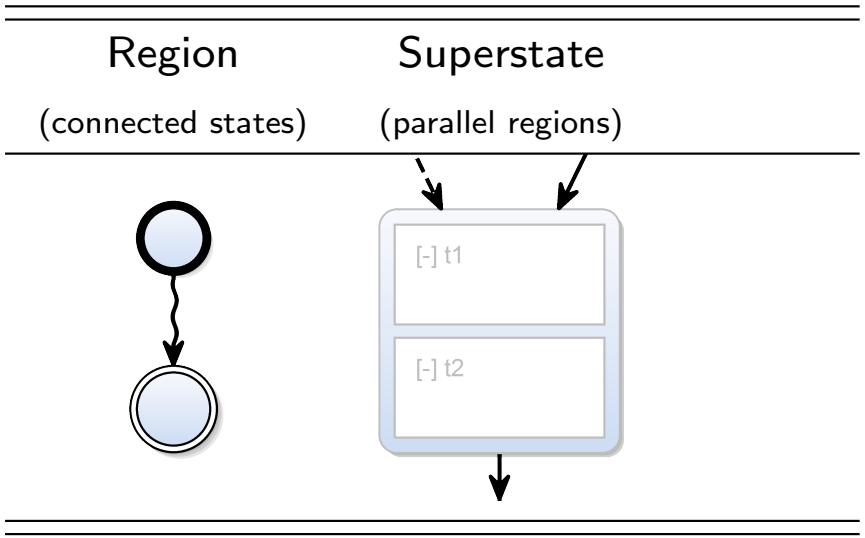
- Further simplify compilation process for Core SCCharts
- Allowed patterns:

Region

(connected states)

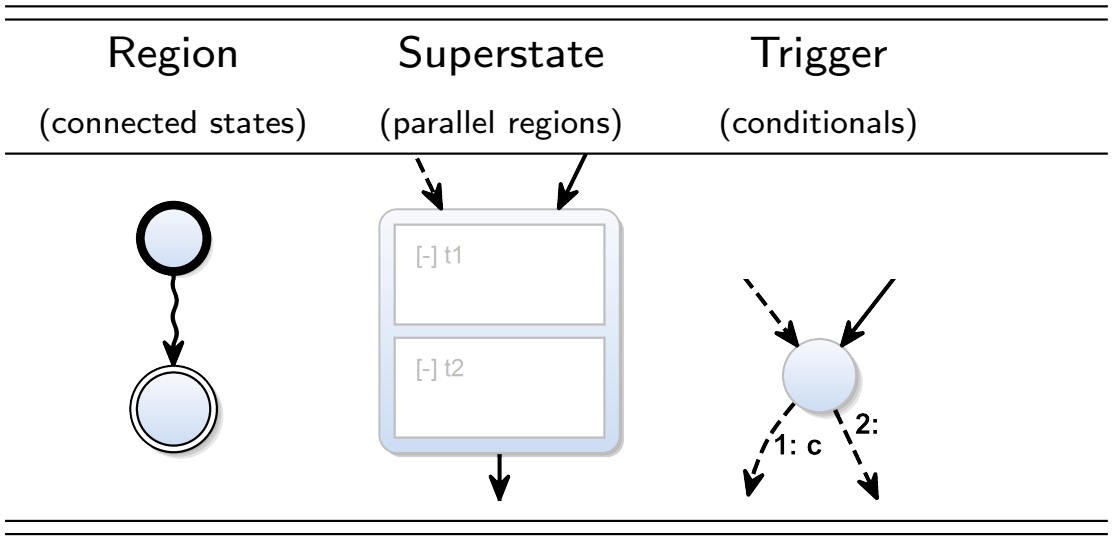
Normalization

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



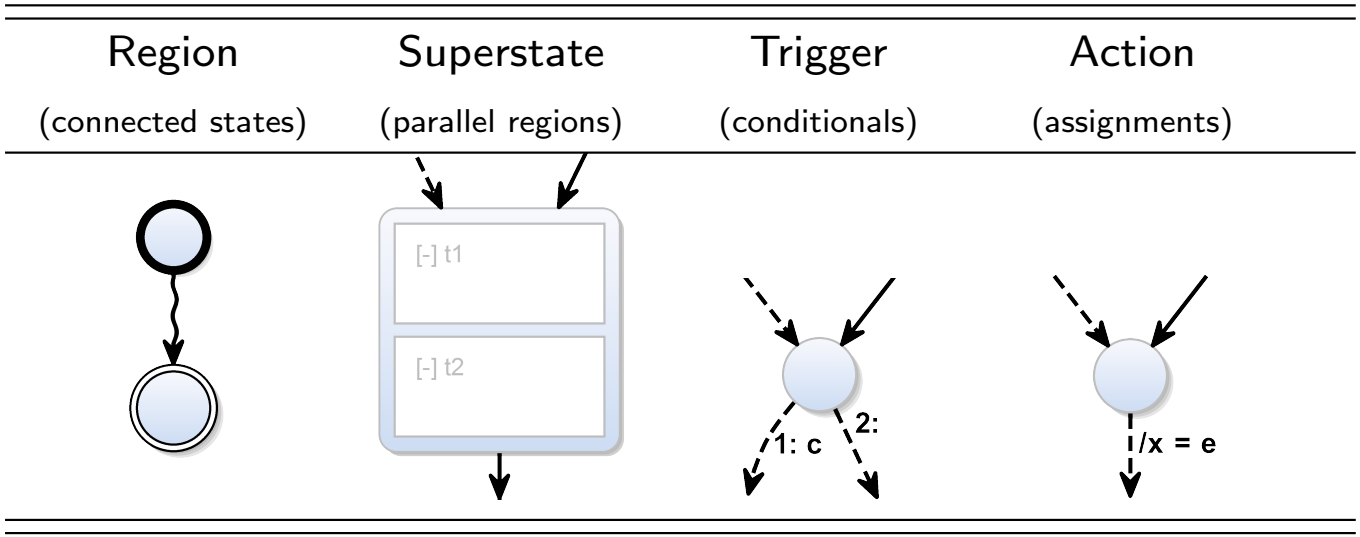
Normalization

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



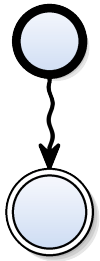
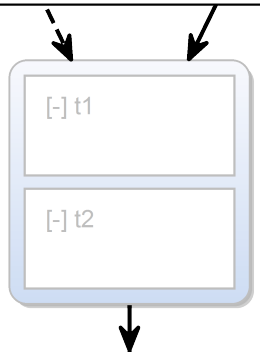
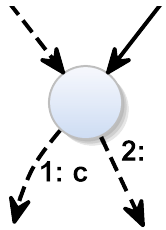
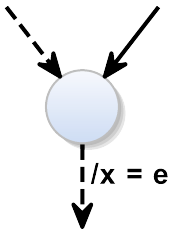
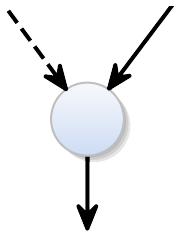
Normalization

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

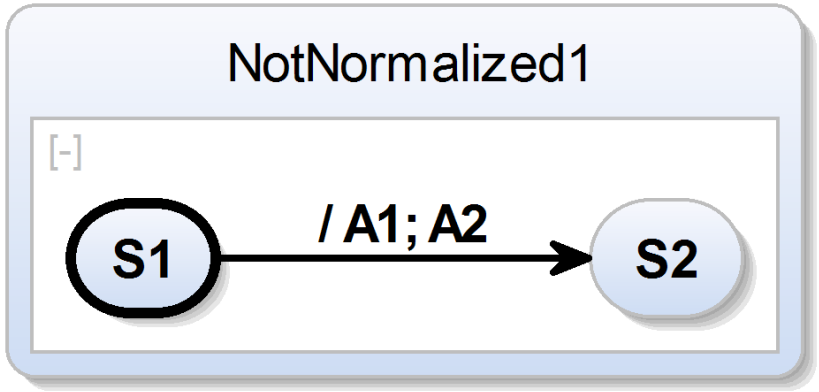


Normalization

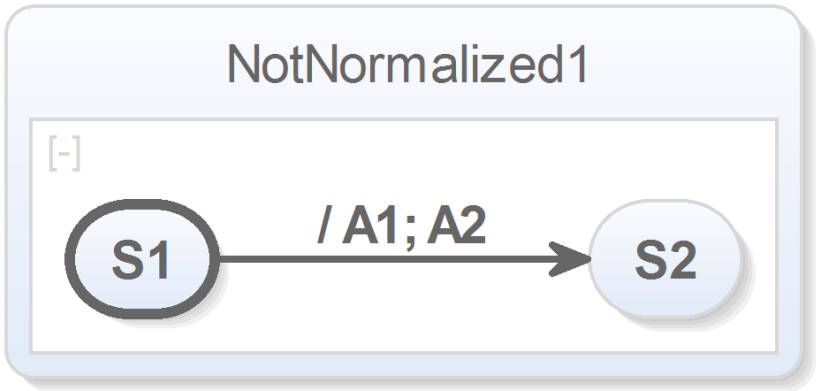
- Further simplify compilation process for Core SCCharts
- Allowed patterns:

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

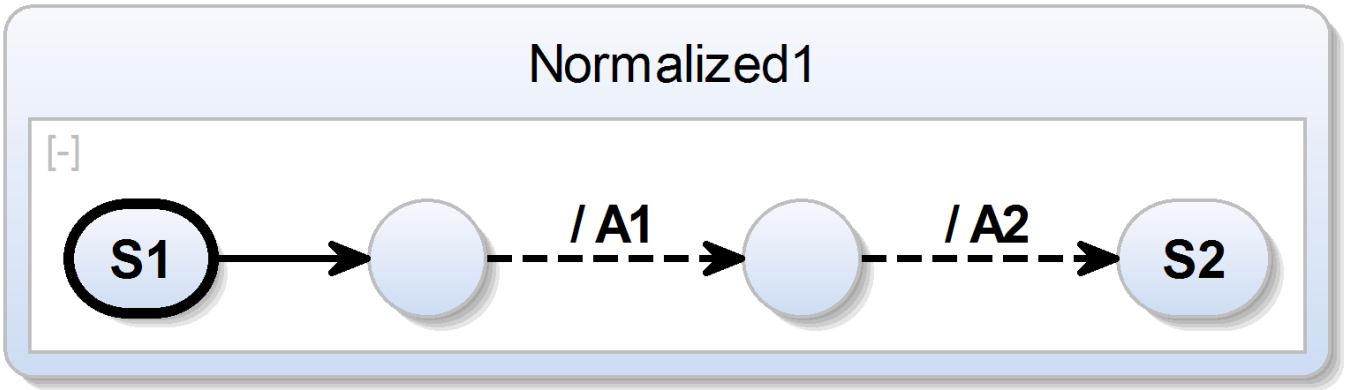
Actions Normalization



Actions Normalization

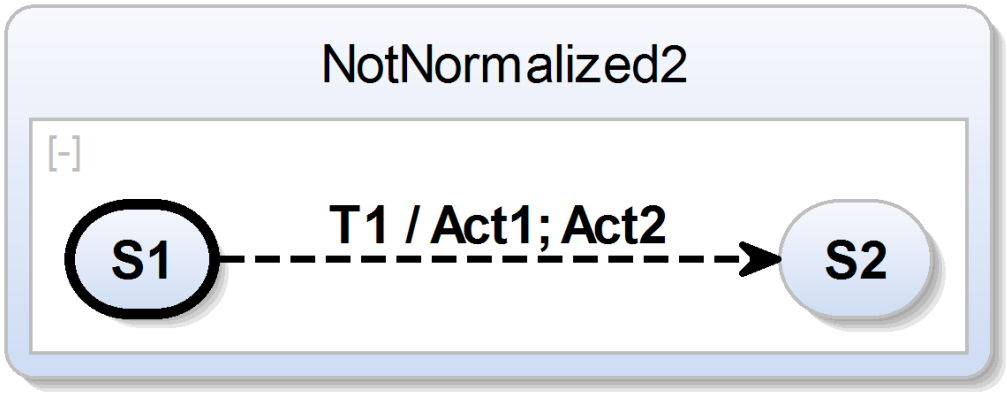


Core SCChart before normalization

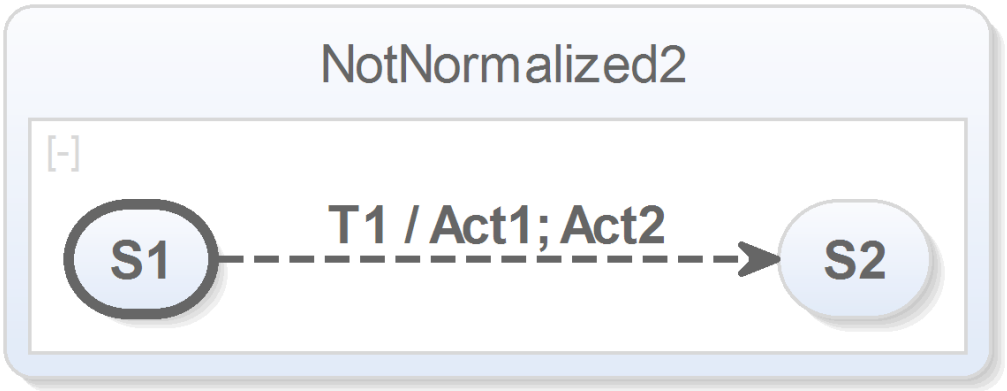


Core SCChart after normalization

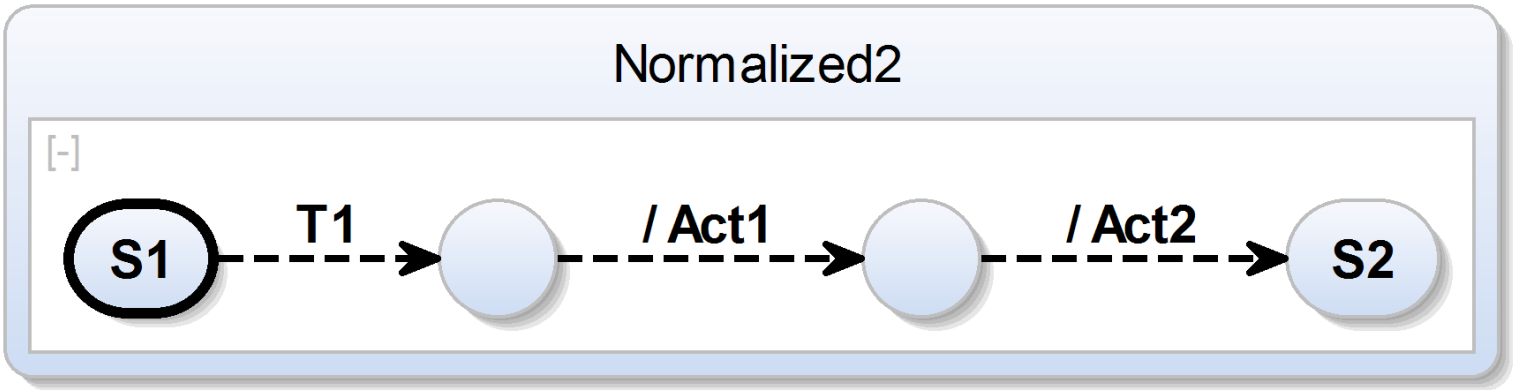
Actions Normalization (cont'd)



Actions Normalization (cont'd)



Core SCChart before normalization

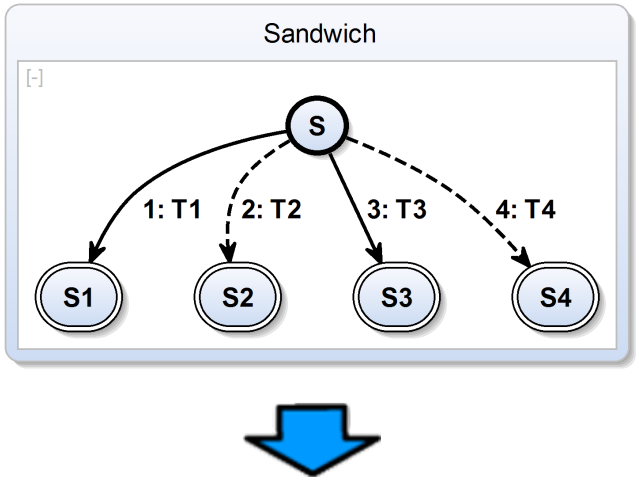


Core SCChart after normalization

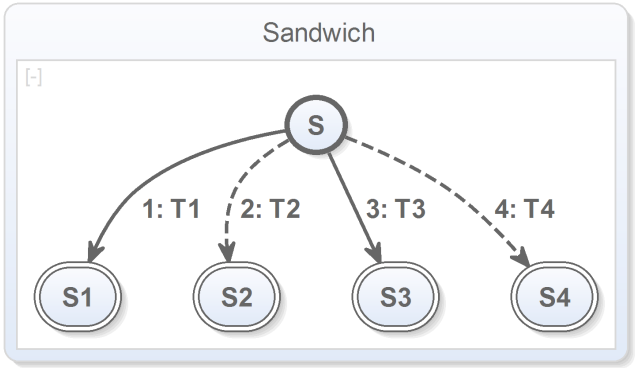
Actions Normalization Implementation Example

```
1  def void transformTriggerActions(Transition transition) {
2    if (((transition.trigger != null || !transition.immediate)
3         && !transition.actions.nullOrEmpty) || transition.actions.size > 1) {
4
5      val targetState = transition.targetState
6      val parentRegion = targetState.parentRegion
7      val transitionOriginalTarget = transition.targetState
8
9      var Transition lastTransition = transition
10
11     for (action : transition.actions.immutableCopy) {
12
13       val actionState = parentRegion.createState(targetState.id + action.id)
14       actionState.setTypeConnector
15
16       val actionTransition = createImmediateTransition.addAction(action)
17       actionTransition.setSourceState(actionState)
18
19       lastTransition.setTargetState(actionState)
20       lastTransition = actionTransition
21      }
22
23      lastTransition.setTargetState(transitionOriginalTarget)
24   }
25 }
```

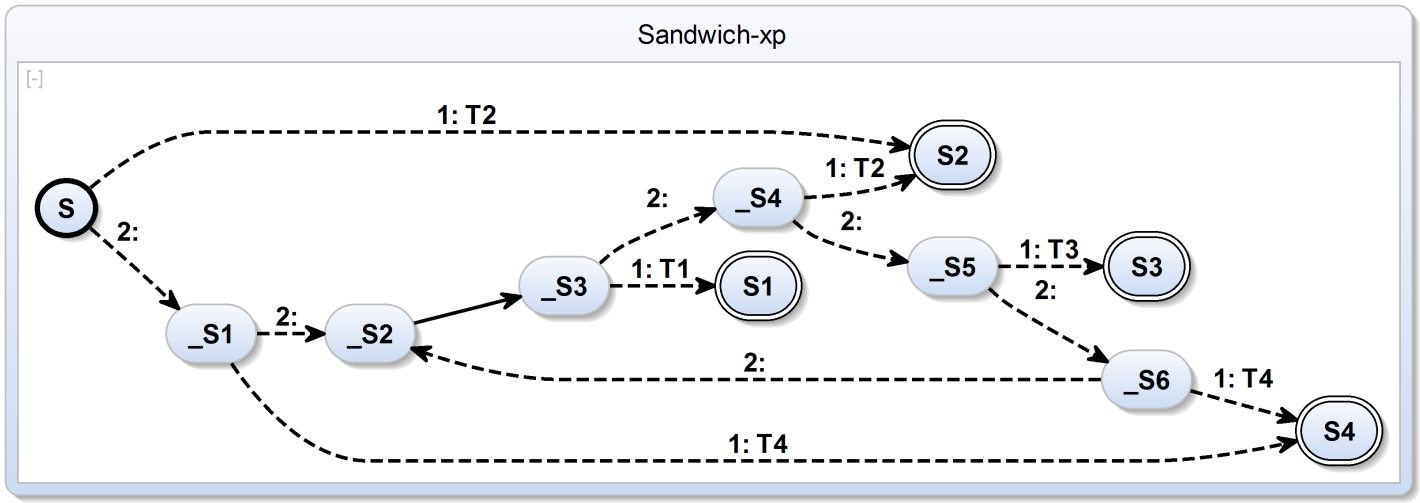
Trigger Normalization



Trigger Normalization

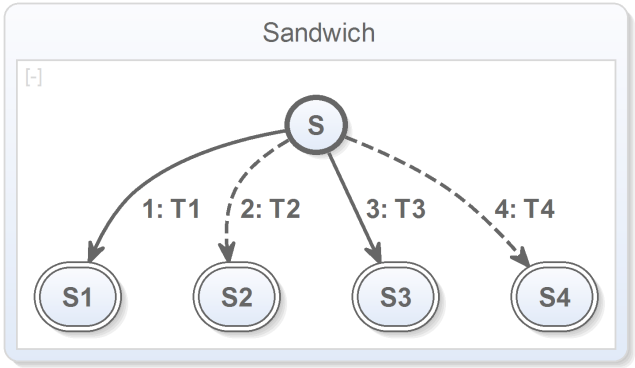


Core SCChart before normalization

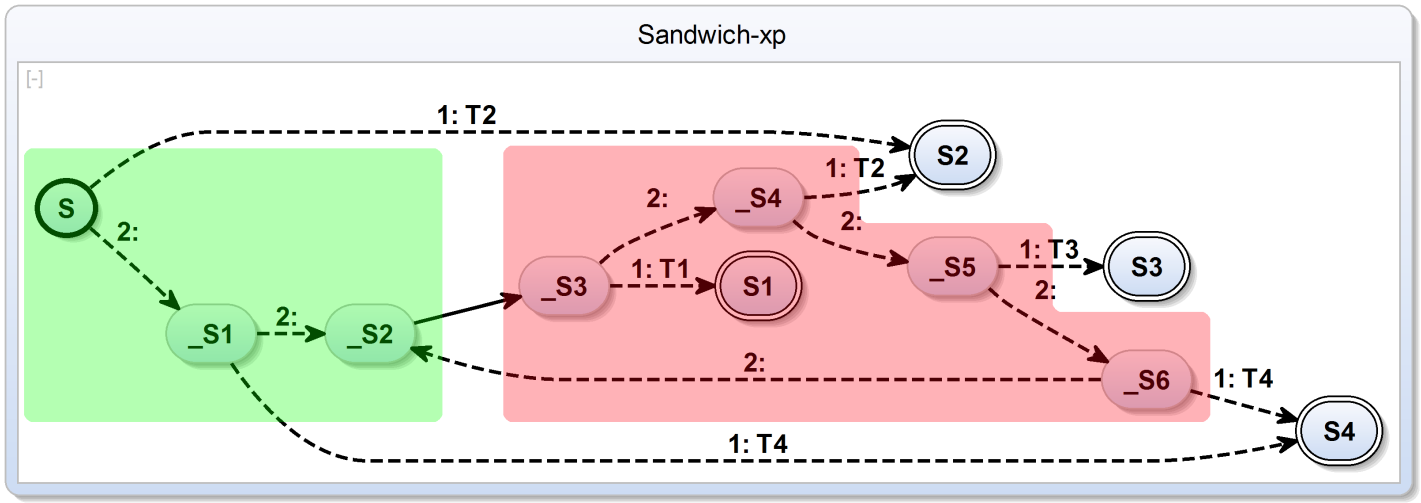


Core SCChart after normalization

Trigger Normalization

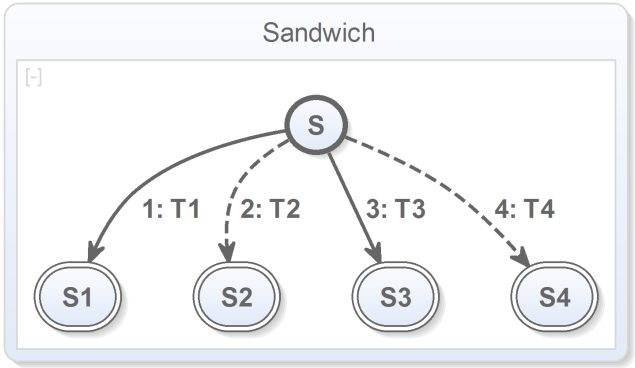


Core SCChart before normalization

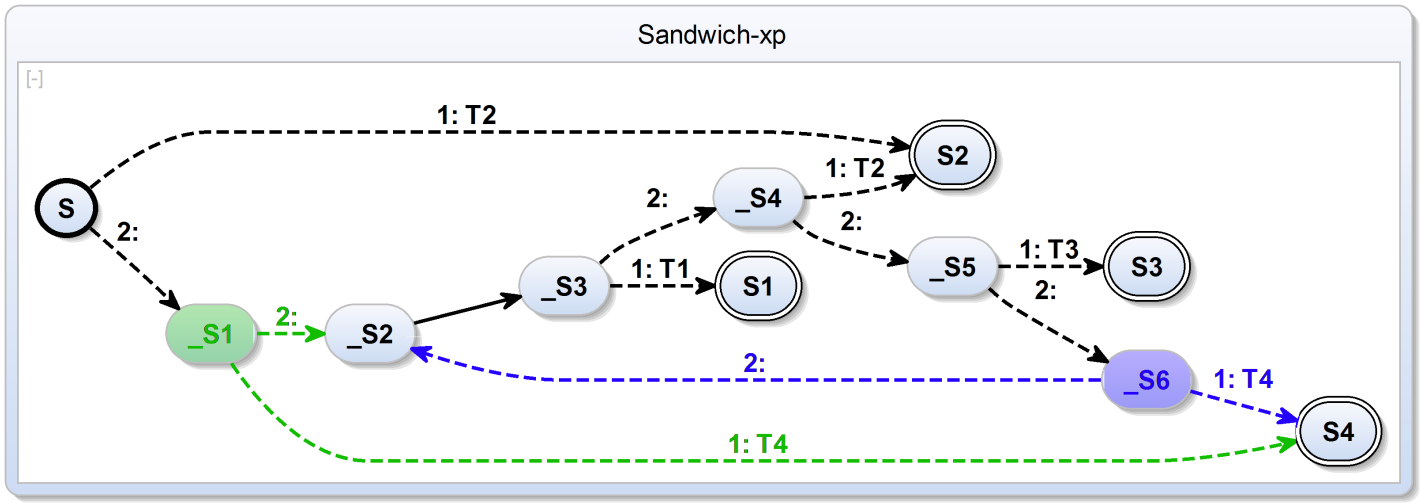


Core SCChart after normalization (Surface & Depth)

Trigger Normalization

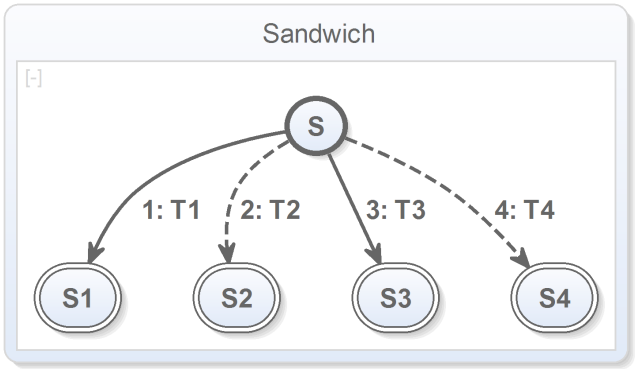


Core SCChart before normalization

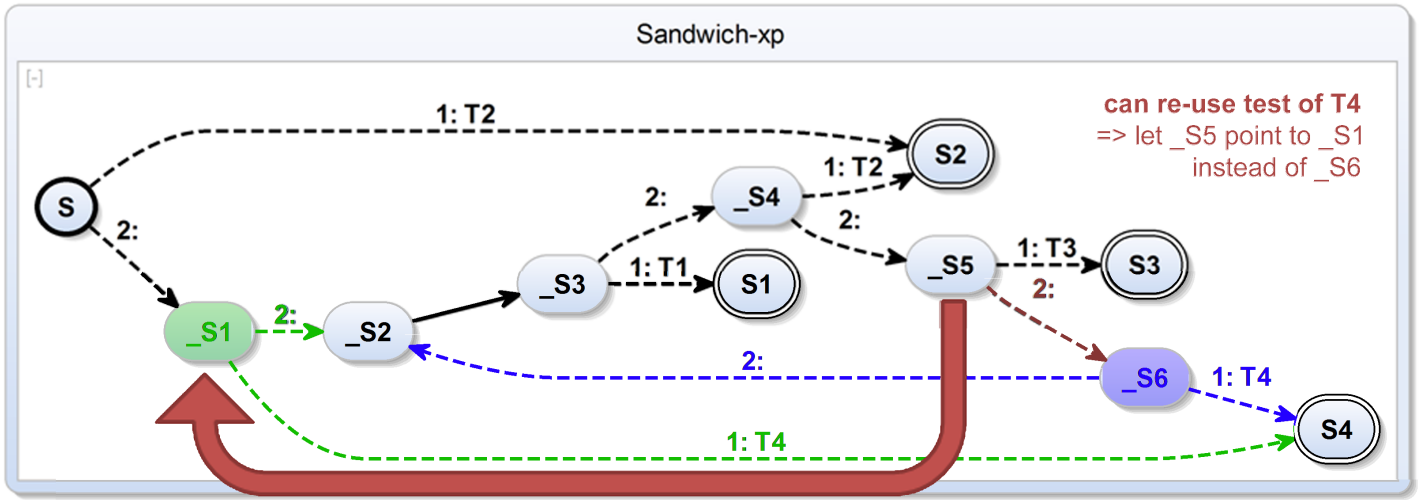


Core SCChart after normalization (potential optimization)

Trigger Normalization

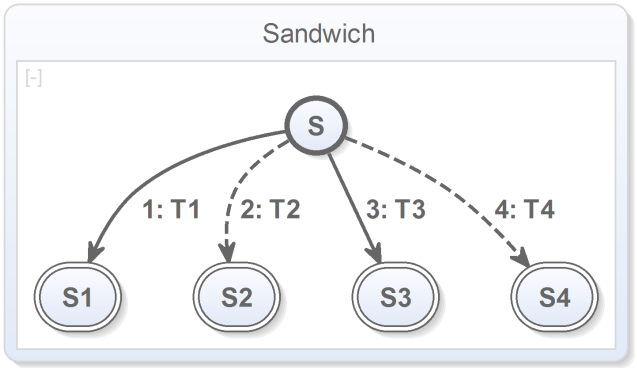


Core SCChart before normalization

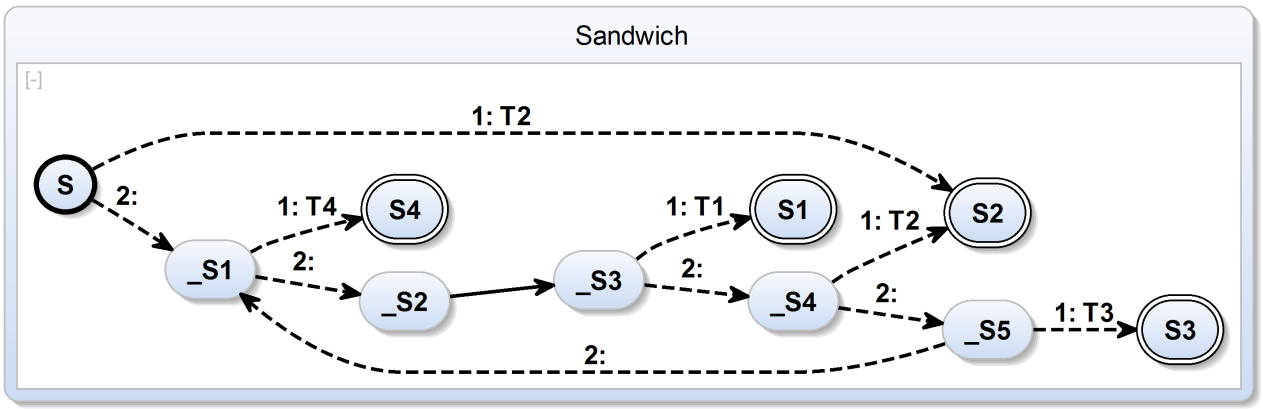


Core SCChart after normalization (potential optimization)

Trigger Normalization (Cont'd)

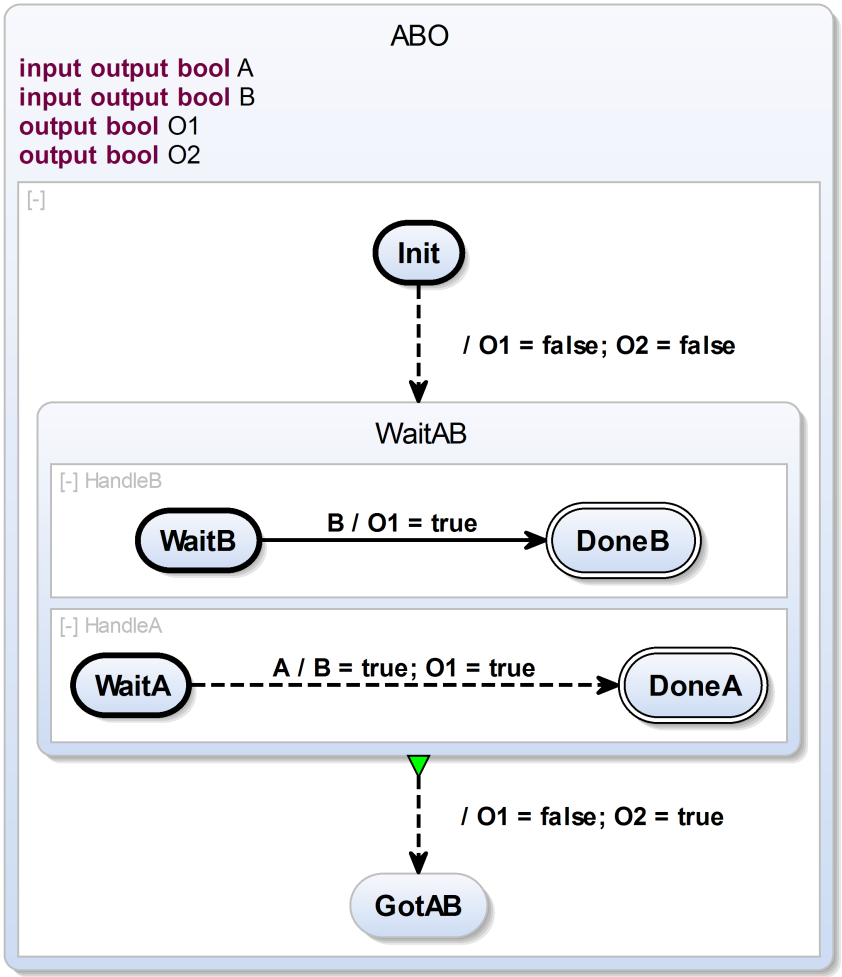


Core SCChart before normalization



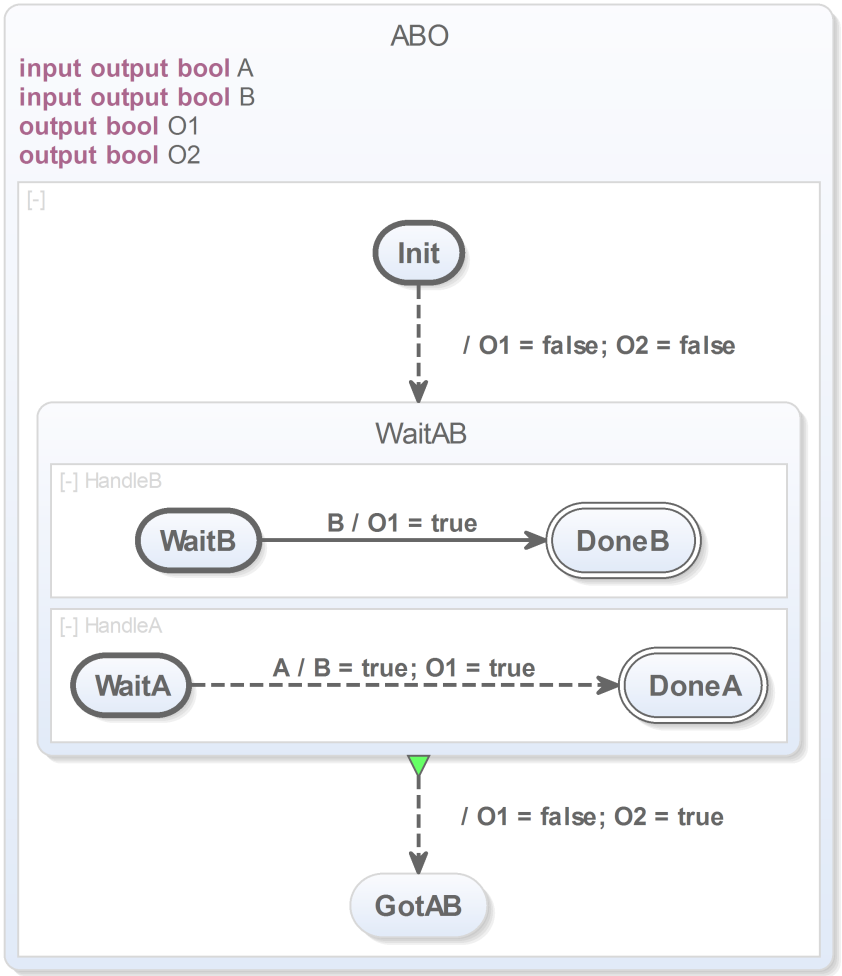
Core SCChart after optimized normalization

ABO — Normalization Example (Actions)

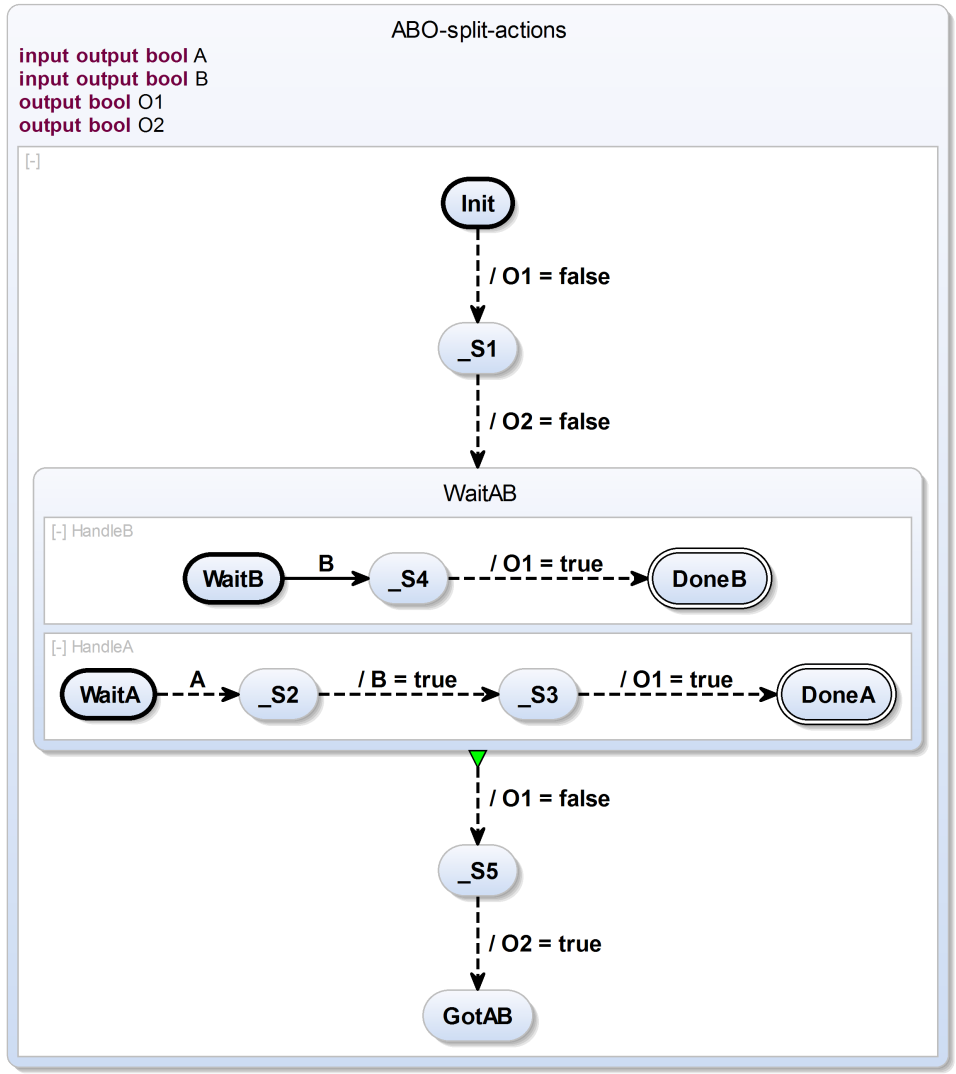


ABO Core SCChart

ABO — Normalization Example (Actions)

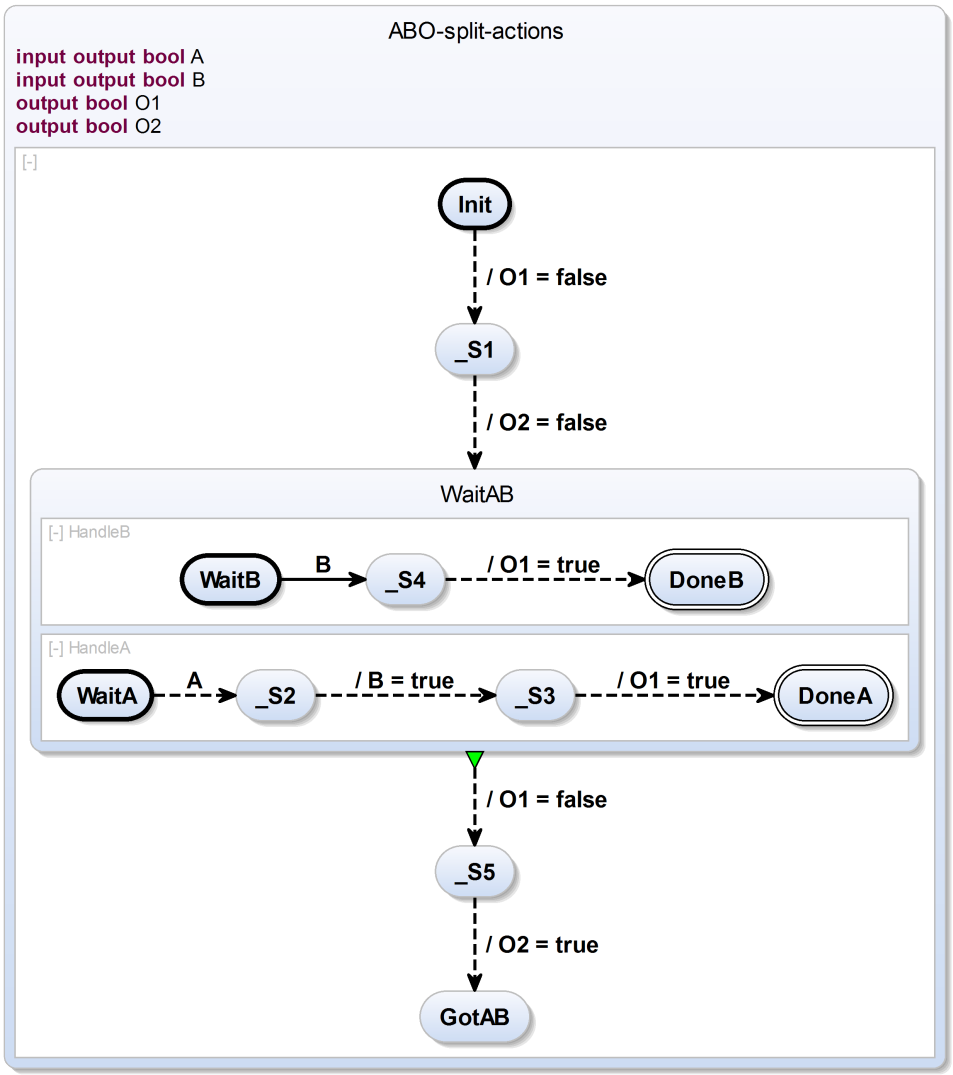


ABO Core SCChart



ABO Core SCChart with normalized actions

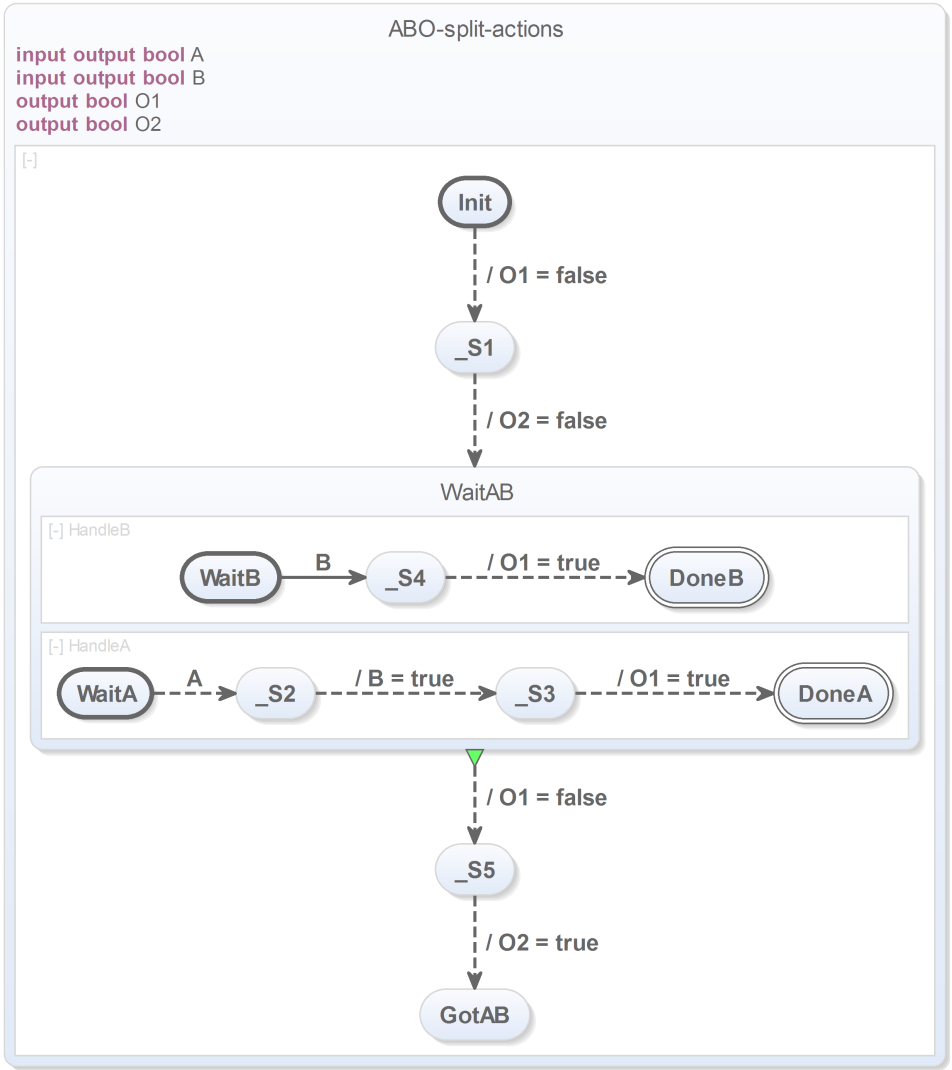
ABO — Normalization Example (Actions & Trigger)



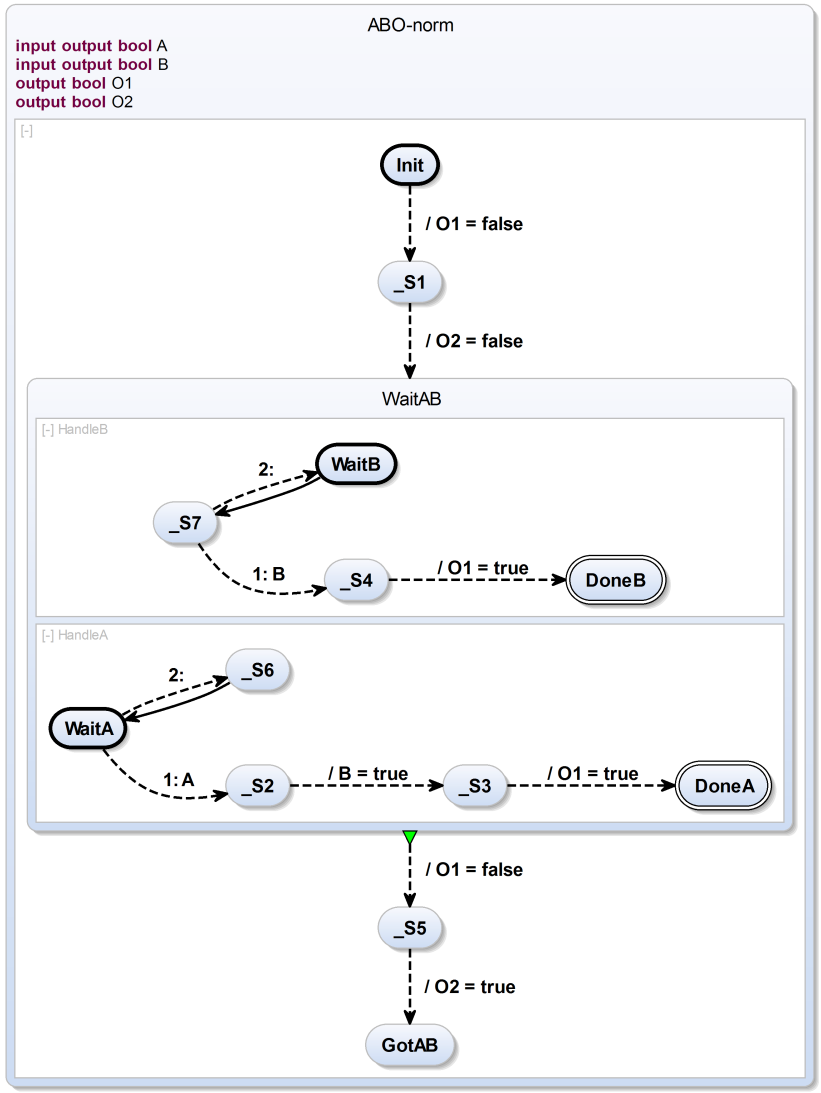
ABO Core SCChart with normalized actions

.

ABO — Normalization Example (Actions & Trigger)



ABO Core SCChart with normalized actions

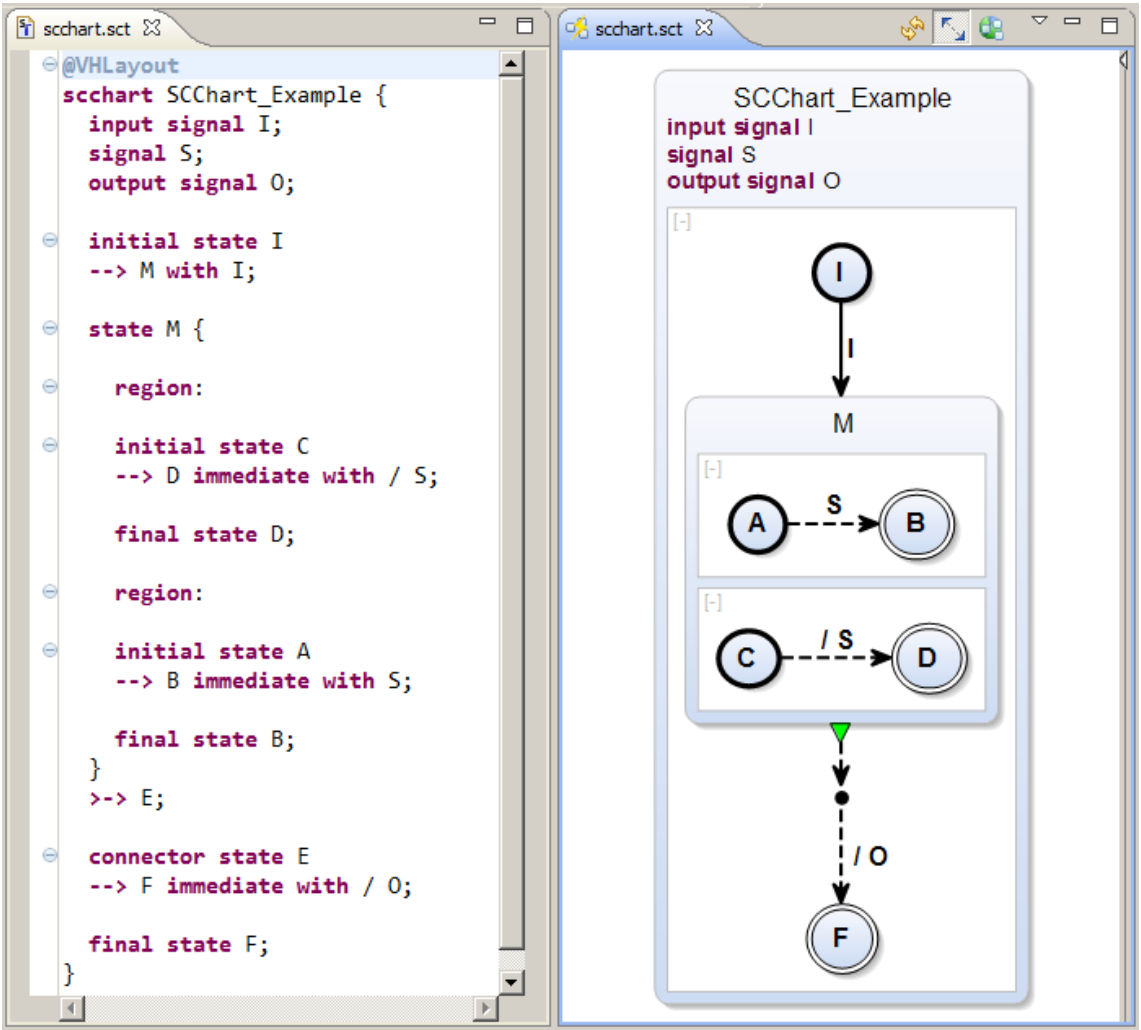


ABO Normalized SCChart

Overview

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

Textual Modeling with KLighD



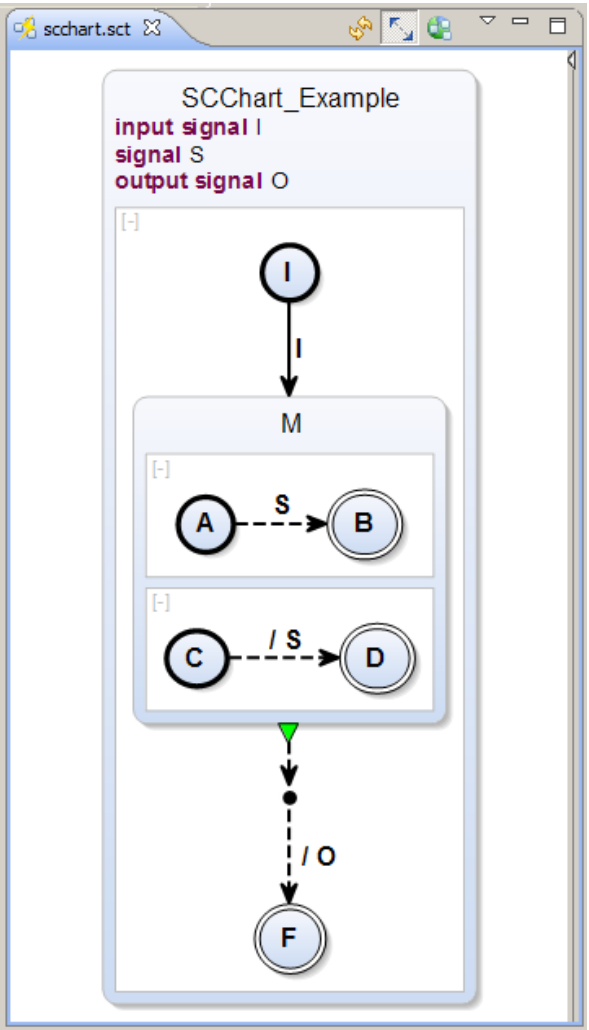
- Eclipse based KIELER framework

Textual Modeling with KLighD

scchart.sct

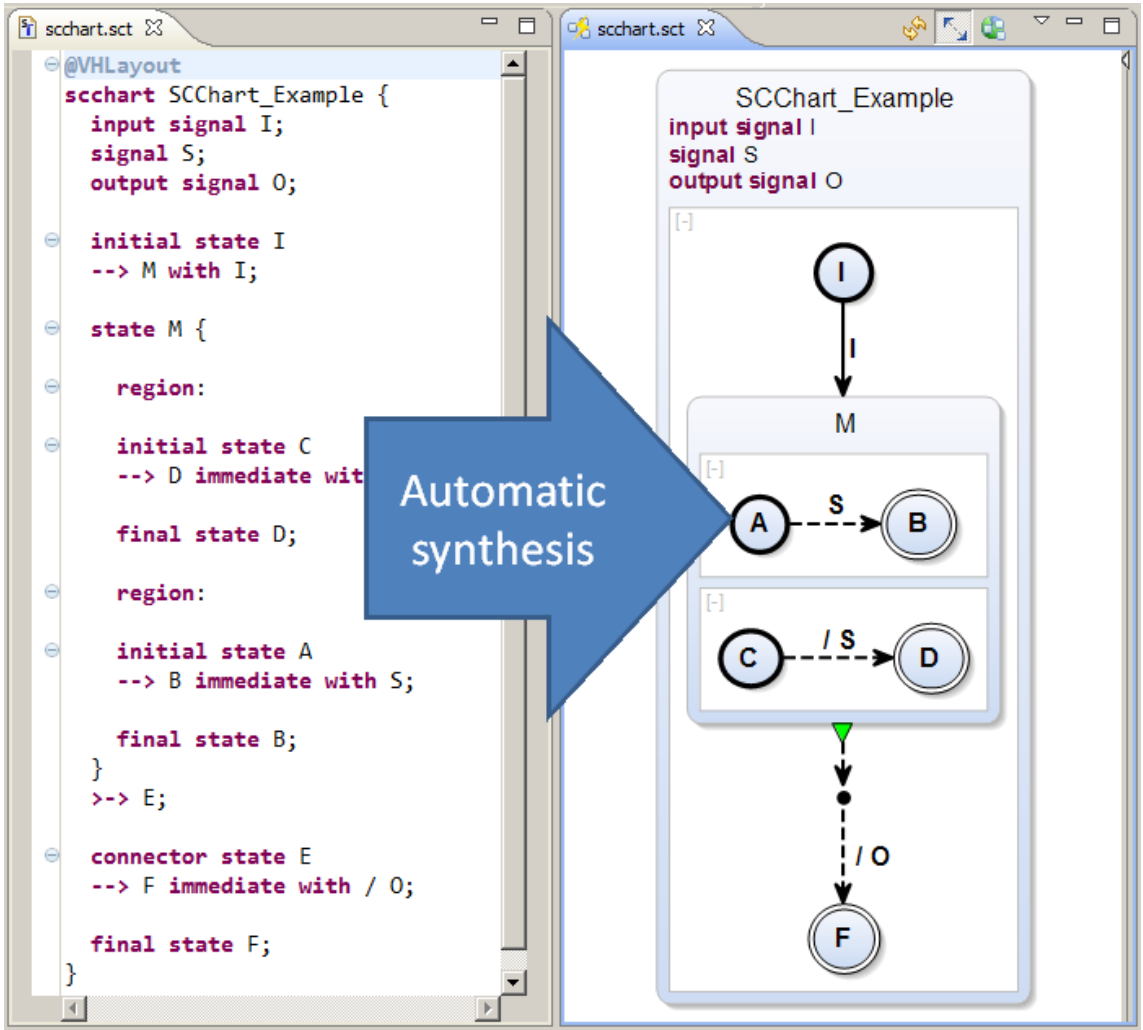
```
@VHLayout
scchart SCChart_Example {
  input signal I;
  signal S;
  output signal O;

  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;
}
```

scchart.sct



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

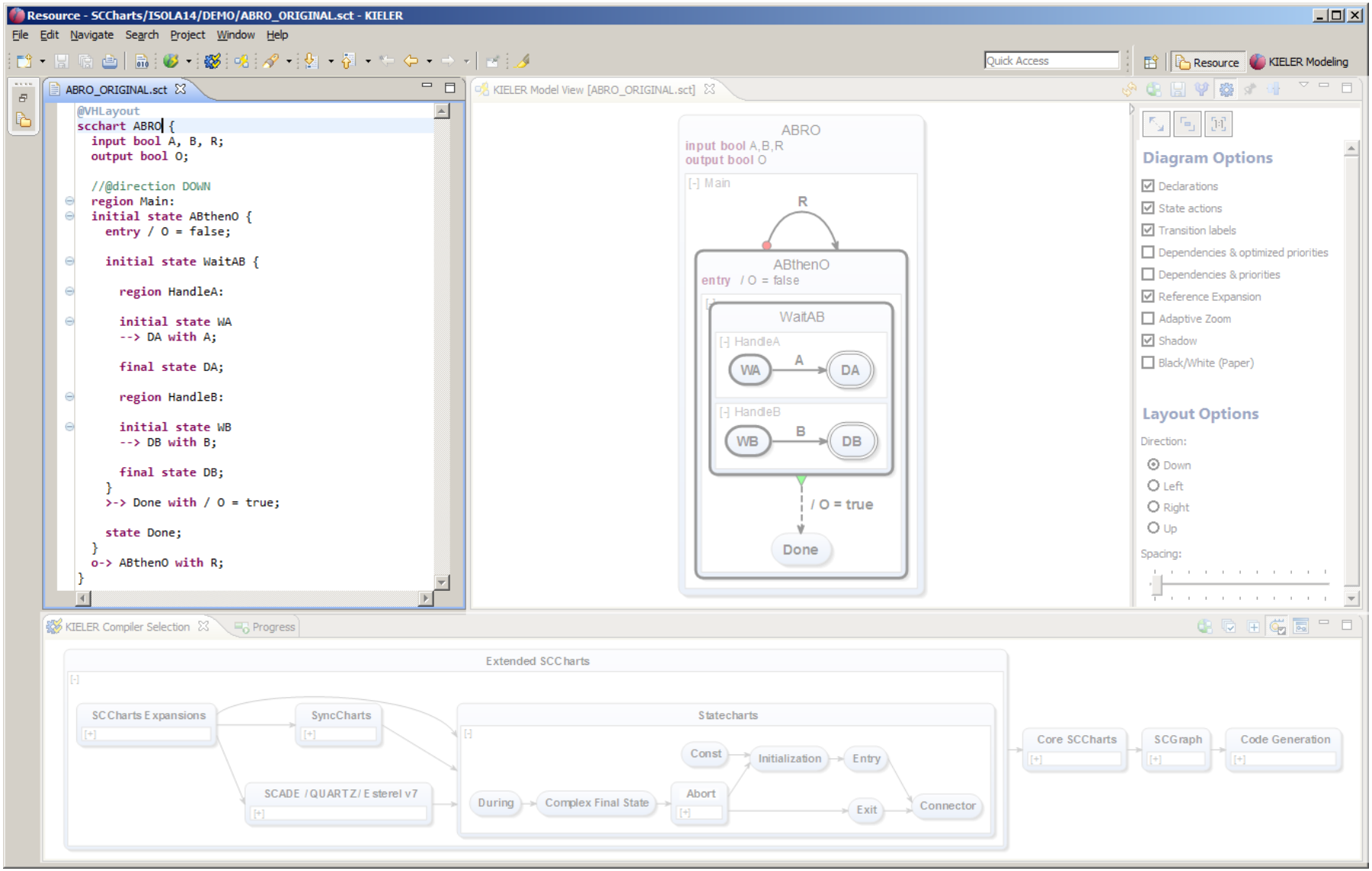
Textual Modeling with KLighD



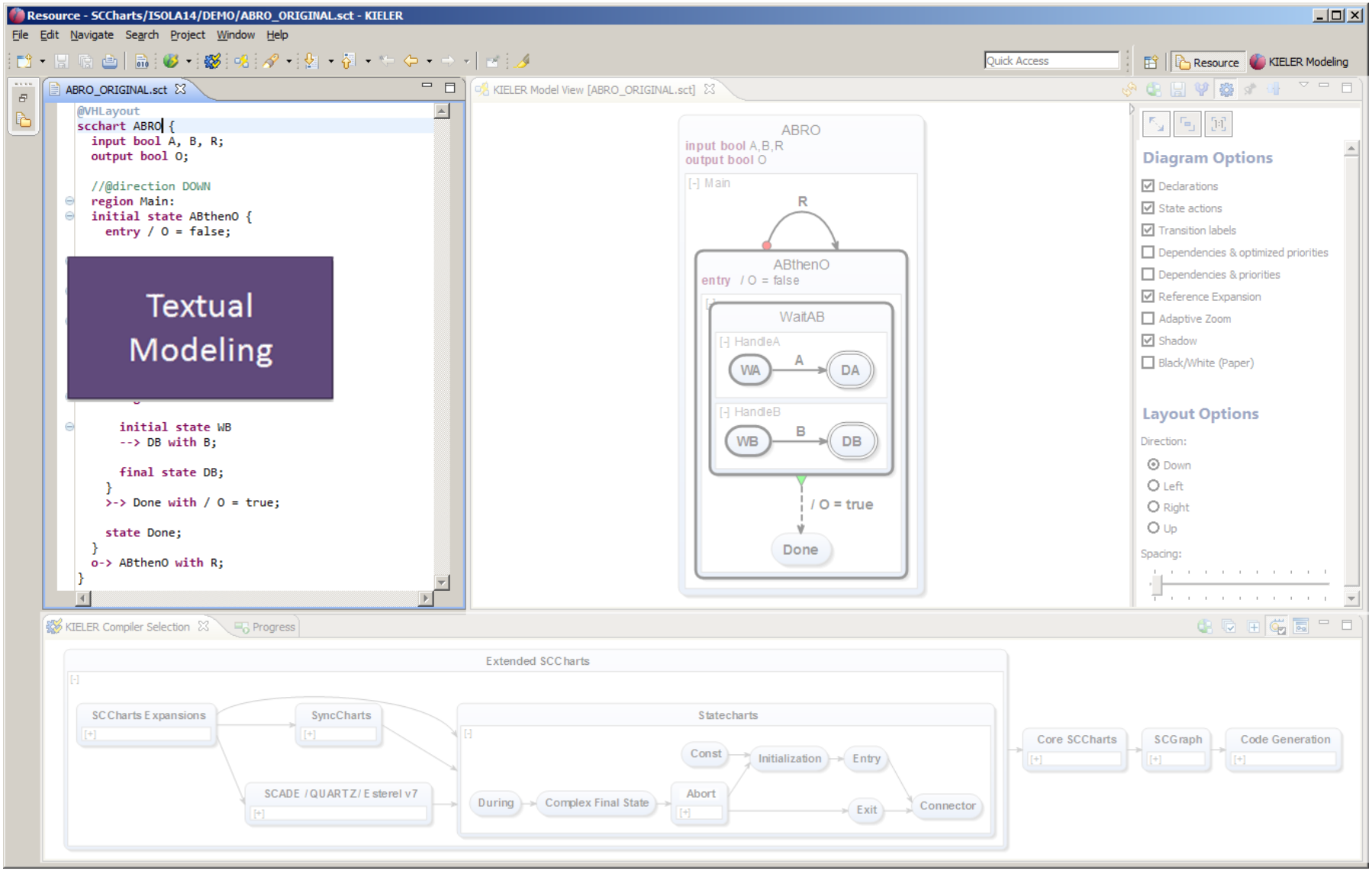
- Eclipse based KIELER framework
- Textual modeling based on Xtext
  - Syntax highlighting
  - Code completion
  - Formatter
- Transient view based on KLighD

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

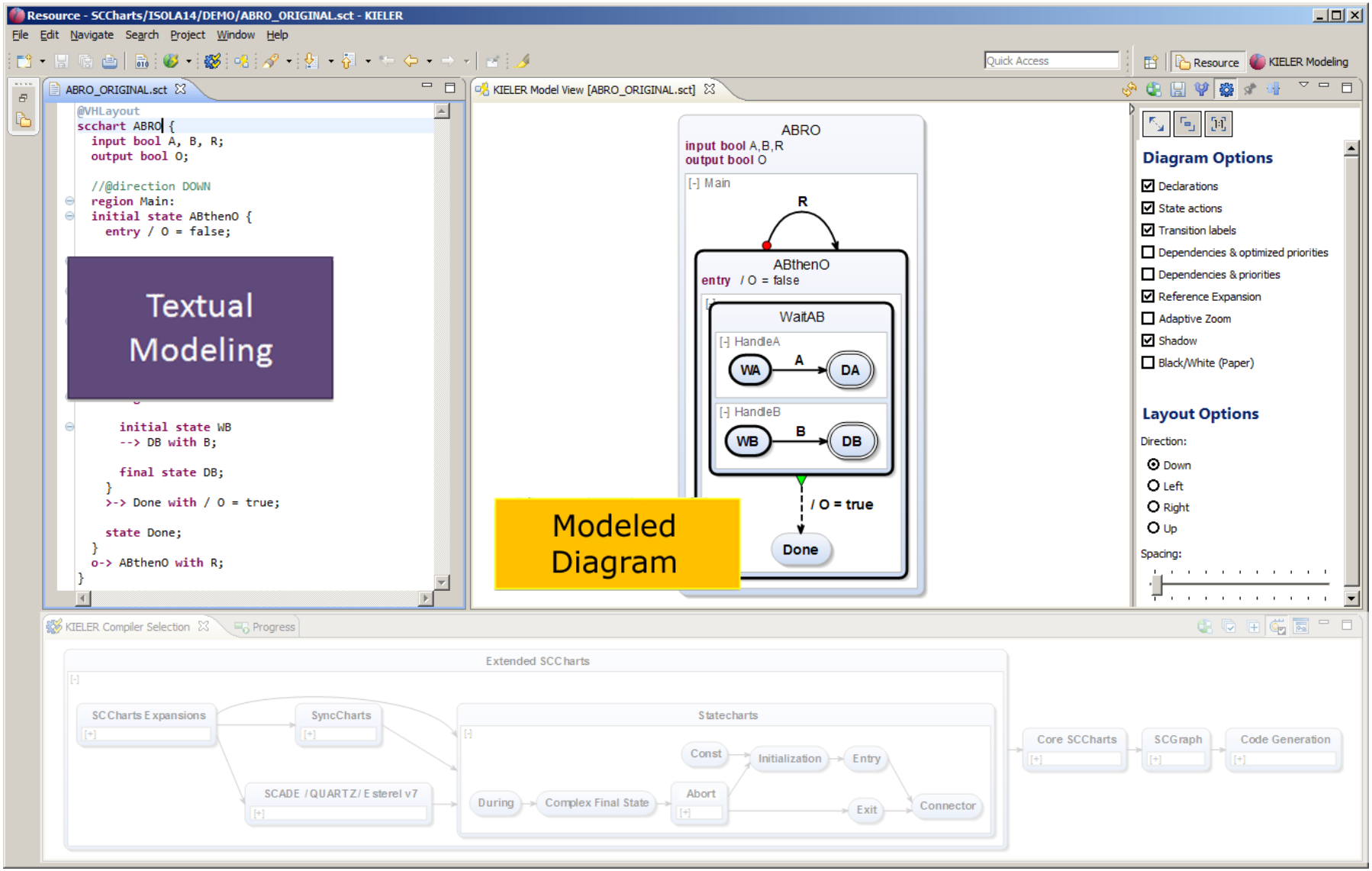
SCCharts Interactive Compilation



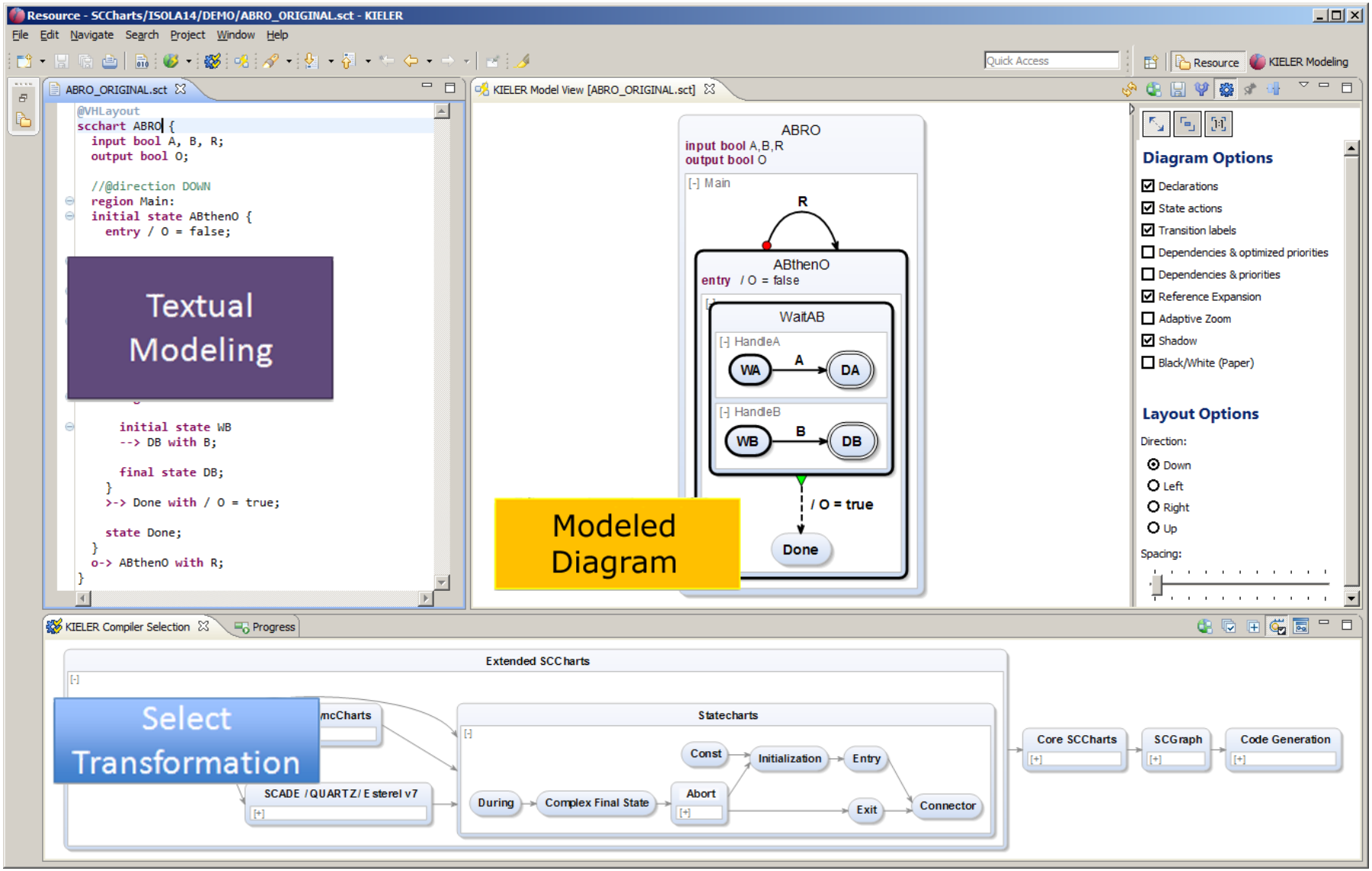
SCCharts Interactive Compilation



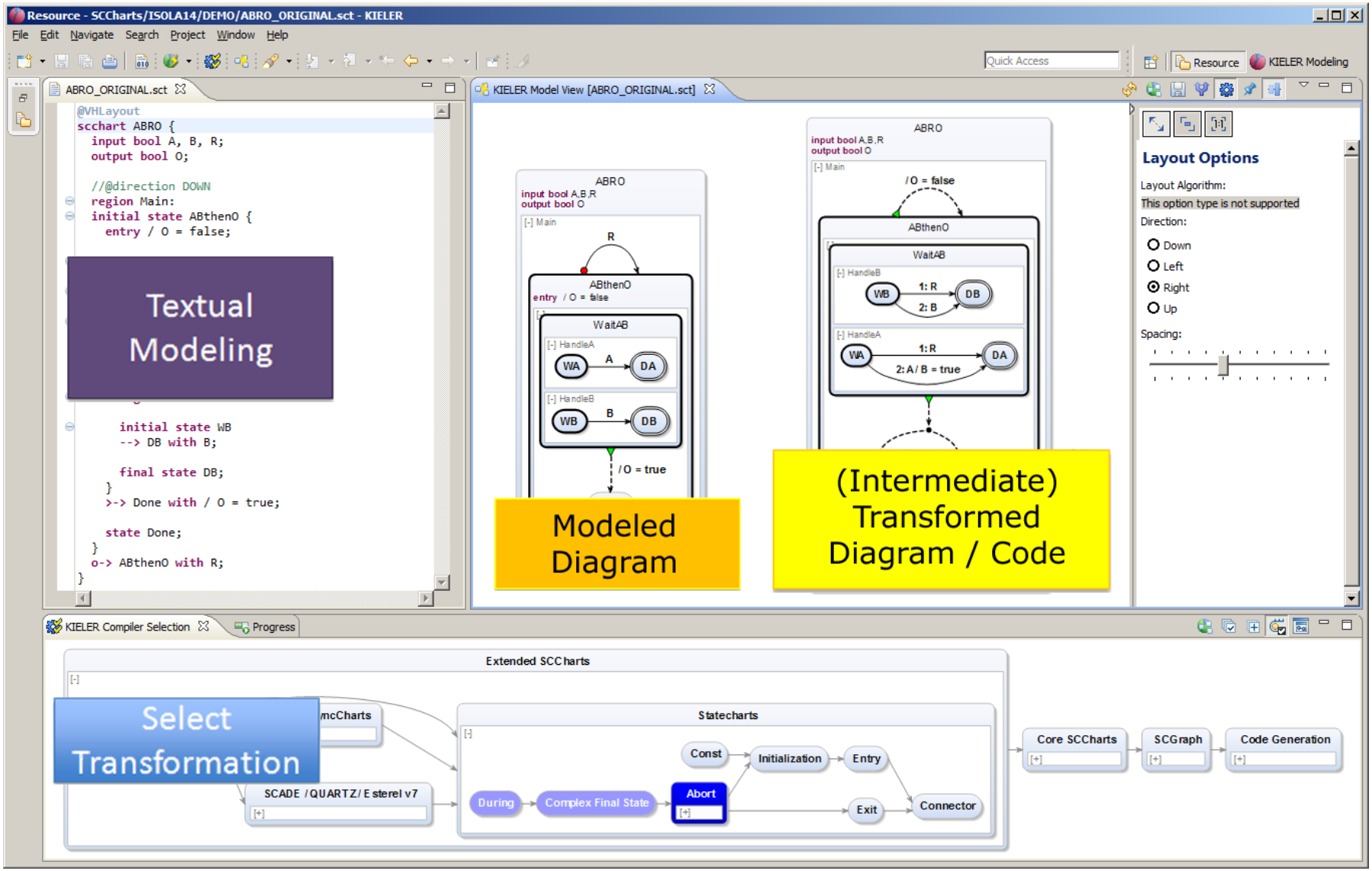
SCCharts Interactive Compilation



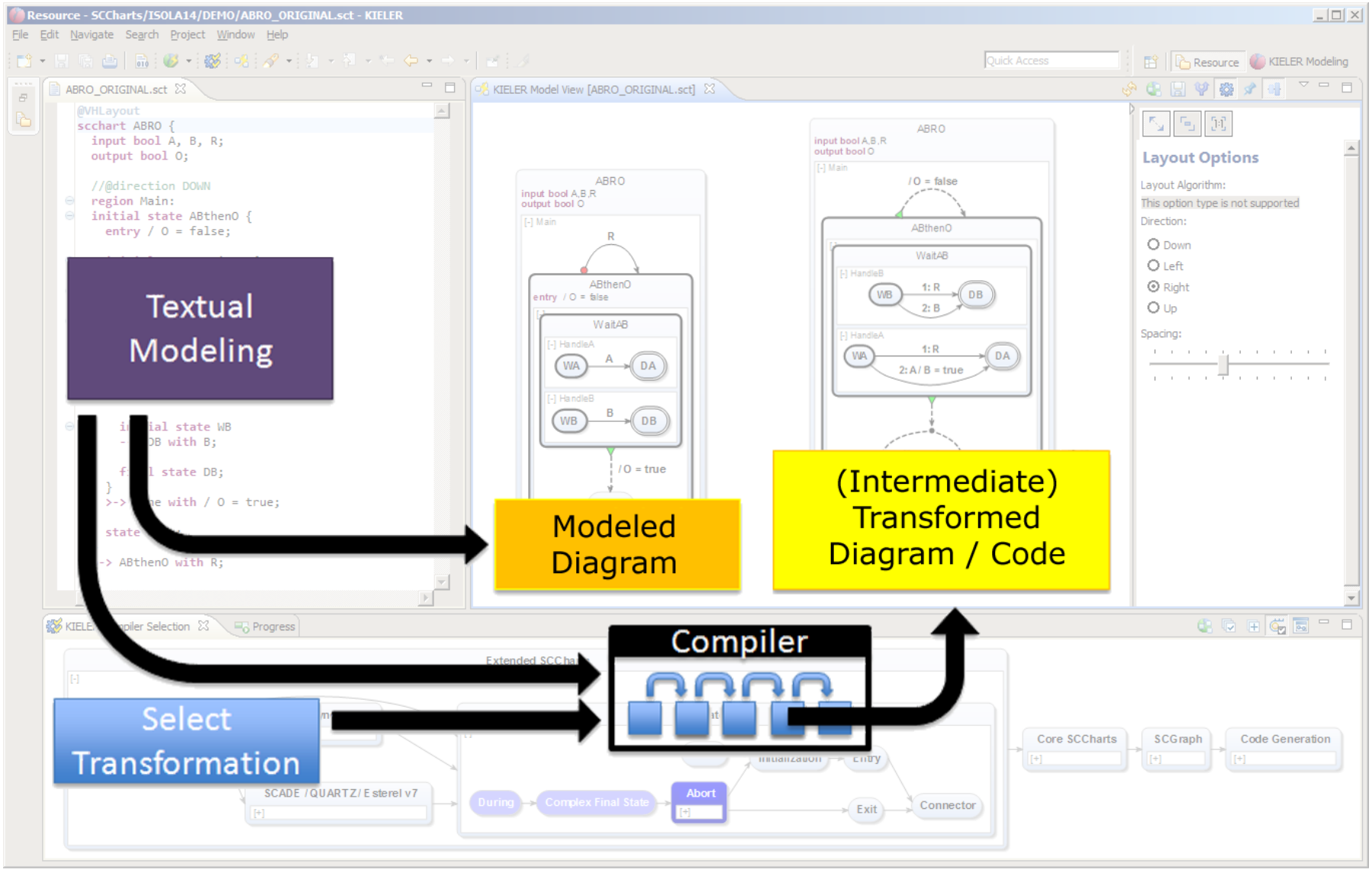
SCCharts Interactive Compilation



SCCharts Interactive Compilation



SCCharts Interactive Compilation



Conclusions

- SyncCharts **are** a great choice for specifying deterministic control-flow behavior

Conclusions

- SyncCharts **are** a great choice for specifying deterministic control-flow behavior...
- ... but do not accept sequentiality

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

Conclusions

- SyncCharts **are** a great choice for specifying deterministic control-flow behavior...
- ... but do not accept sequentiality
  `If (!done) { ... ; done = true;}`
- **SCCharts** extend SyncCharts w.r.t. semantics
  → Sequentially Constructive MoC

Conclusions

- SyncCharts **are** a great choice for specifying deterministic control-flow behavior...
- ... but do 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

Conclusions

- SyncCharts **are** a great choice for specifying deterministic control-flow behavior...
- ... but do 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

Conclusions

- SyncCharts **are** a great choice for specifying deterministic control-flow behavior...
- ... but do 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

- 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*. Proc. ACM SIGPLAN Conference on Programming Language Design and Implementation (PLDI'14), Edinburgh, UK, June 2014. `https://rtsys.informatik.uni-kiel.de/~biblio/downloads/papers/pldi14.pdf`
- C. Motika, S. Smyth and R. von Hanxleden, *Compiling SCCharts—A Case-Study on Interactive Model-Based Compilation*, Proc. 6th International Symposium on Leveraging Applications of Formal Methods, Verification and Validation (ISoLA 2014), Corfu, Greece, LNCS 8802, pp. 443–462 `https://rtsys.informatik.uni-kiel.de/~biblio/downloads/papers/isola14.pdf`