Lecture 5

Safe State Machines (SSM)

Daniel Kästner
AbsInt GmbH

2012

Embedded Systems

- Typically, embedded systems are reactive systems:

  „A reactive system is one which is in continual interaction with is environment and executes at a pace determined by that environment“ [Bergé, 1995]

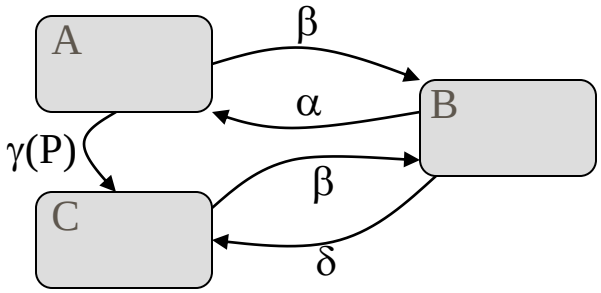
  Behavior depends on input and current state.

➢ automaton model appropriate

➢ SCADE language
  - ✓ Data flow kernel (extended LUSTRE dialect)
  - State Machines (SSM)

State Transition Diagrams

- State transition: When event γ occurs in state A, if Condition P is true at the time, the system executes action *a* and transfers to state C.
- State diagrams are directed graphs with nodes denoting states, and arrows (labelled with the triggering event, guarding conditions and action to be executed) denoting transitions.
- Example:



- Problem: all combinations of states have to be represented explicitly, leading to exponential blow-up.

State Transition Diagrams

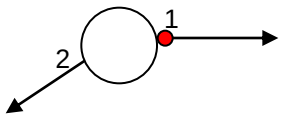
- Disadvantages:
  - No structure (no strategy for bottom-up of top-down development)
  - State-transition diagrams are flat, ie without hierarchy
  - Uneconomical wrt transitions (eg interrupt): exponential blow-up
  - Uneconomical wrt states: exponential blow-up
  - Uneconomical wrt parallel composition: exponential blow-up
  - Inherently sequential; parallelism cannot be expressed in a natural way

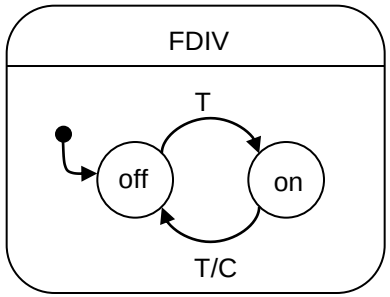
SyncCharts

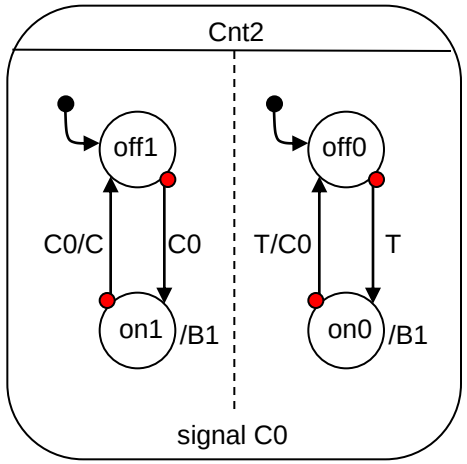
- Visual formalism for describing states and transitions of a system in a modular fashion.
- Extension of state-transition diagrams (Mealy/Moore automata):
  - Hierarchy
  - Modularity
  - Parallelism
- Is fully deterministic.
- Tailored to control-oriented applications (drivers, protocols).
- Implements sychronous principle.

SyncCharts

- States (circles and rectangles):
  - can be named
  - two types:
    - simple state (circle)
    - macrostate (rounded rectangle): contain a hierarchy of other states
  - are optionally labelled*: /<effect>
- Transitions (arrows):
  - are labelled*: <trigger>/<effect>
    - All components are optional.
  - three types:
    - strong abort
    - weak abort
    - normal termination
  - can have priorities (-> determinism)

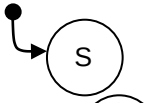


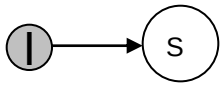




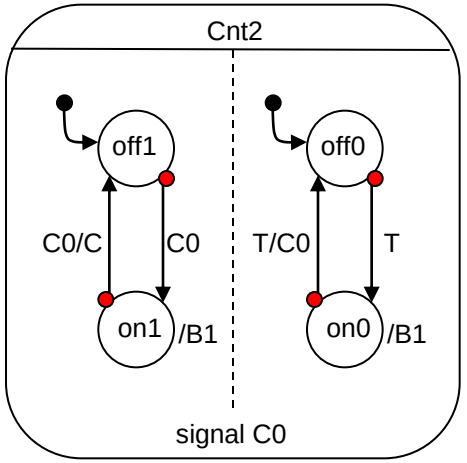
*Triggers and effects are signals, or combinations of signals using boolean operations *or*, *and* and *not.*

States & State Transition Graphs

- Special states:
  - Initial state: 

(alternative notation: 

)
  - Terminal state: 
- State Transition Graph: connected labeled graph made of states connected by transitions, with an initial state.
- Two types of states:
  - Simple state: just carries a label.
  - Macrostate: contains at least one state transition graph.



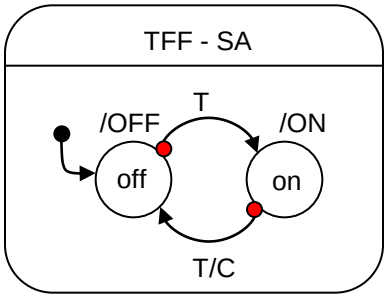
- At each instant there is one and only one active state.
- An active state waits for the satisfaction of the trigger of one of its outgoing transitions, at an instant strictly posterior to its entering (activation).

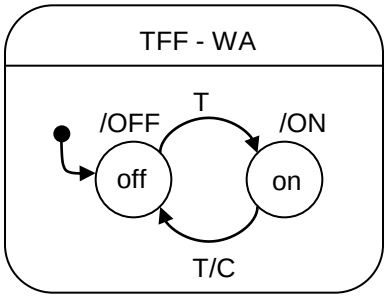
State and Transition Labels

- Signals are characterized by their presence status (+,-, ⊥).
  - Valued signal: signals conveys a value of a given type.
  - Pure signal: no value conveyed.
- tick: implicit signal present at every instant.
- A trigger is satisfied ⇔ associated signal is present.
- Transition labels:
  - When the trigger is satisfied, the transition is said to be enabled.
  - The transition is immediately taken and emits the associated signals.
  - The firing of a transition is fully deterministic and takes no time.
- Node labels:
  - Signal emission depends on transition type (strong/weak abort)
  - Signals are emitted when…

	…entering	…in	…exiting
Weak abort	Yes	Yes	Yes
Strong abort	Yes	Yes	No

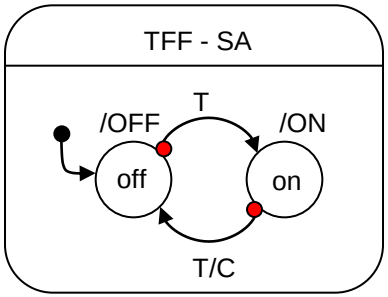
Example: Strong vs. Weak Abort

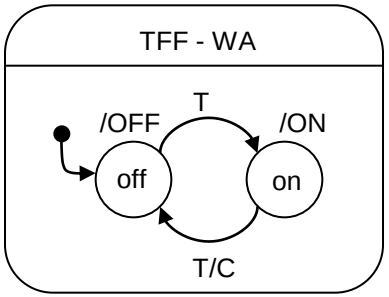




Instant	Input	TFF-SA Output	TFF-WA Output
1			
2	T		
3			
4	T		
5			
6	T		
7	T		
8	T		
9			

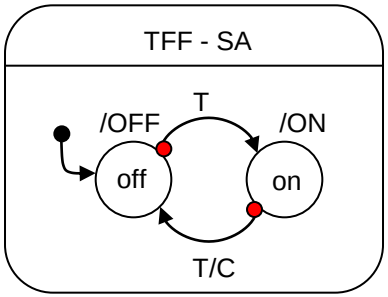
Example: Strong vs. Weak Abort

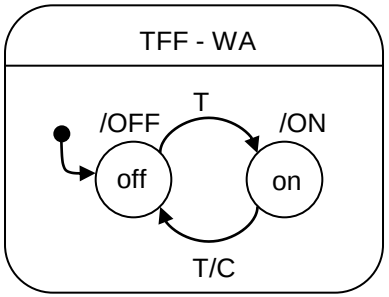




Instant	Input	TFF-SA Output	TFF-WA Output
1		OFF	
2	T	ON	
3		ON	
4	T	C, OFF	
5		OFF	
6	T	ON	
7	T	C,OFF	
8	T	ON	
9		ON	

Example: Strong vs. Weak Abort

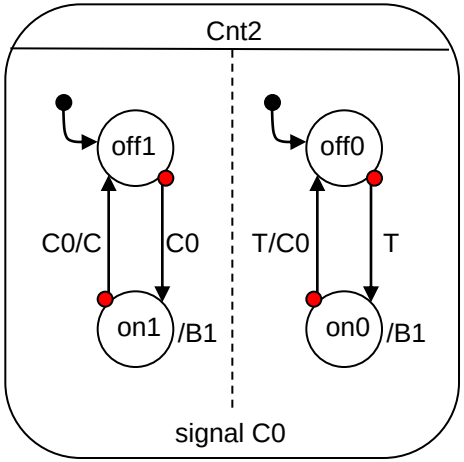




Instant	Input	TFF-SA Output	TFF-WA Output
1		OFF	OFF
2	T	ON	OFF,ON
3		ON	ON
4	T	C, OFF	ON,C,OFF
5		OFF	OFF
6	T	ON	OFF,ON
7	T	C,OFF	ON,C,OFF
8	T	ON	OFF,ON
9		ON	ON

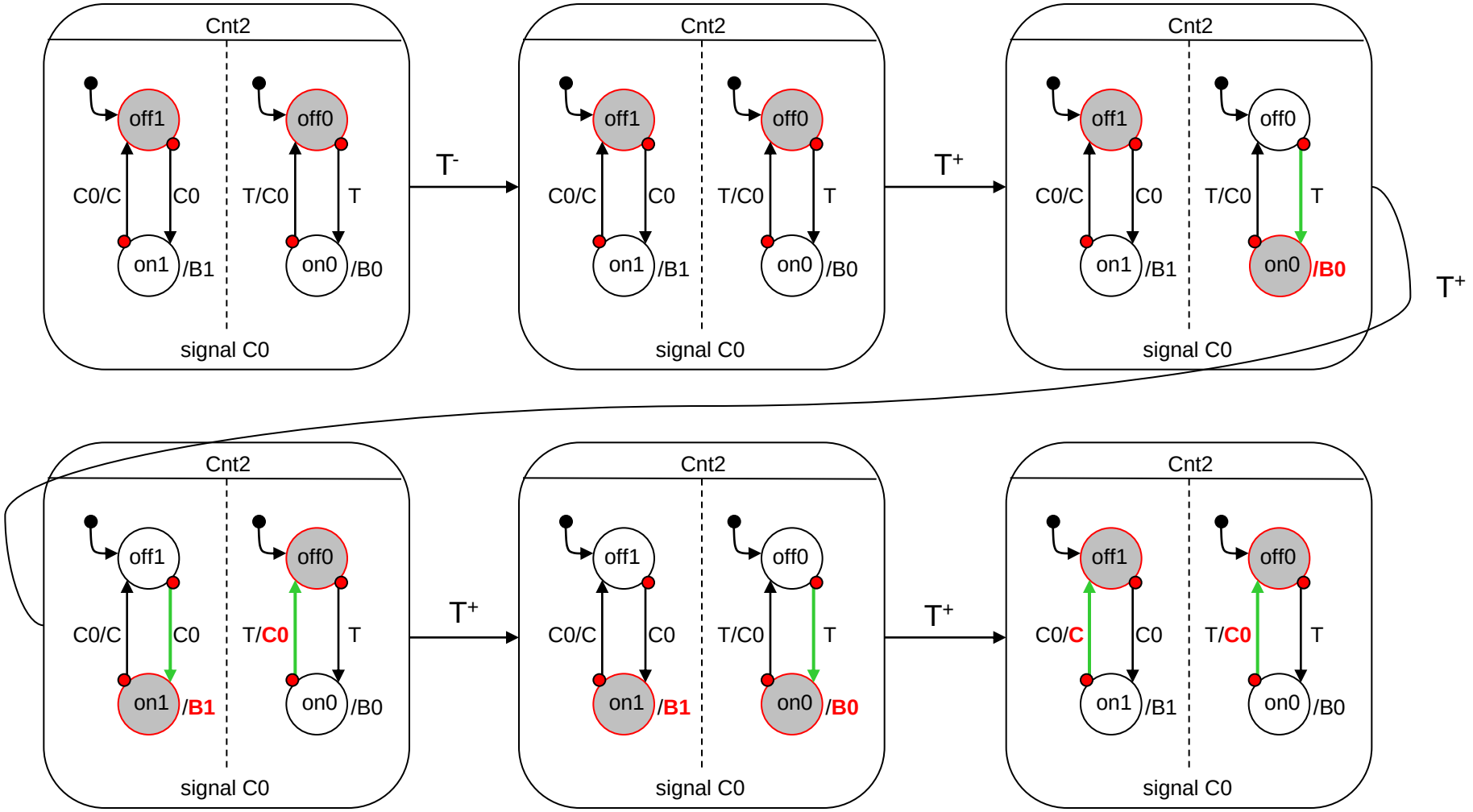
Concurrency

- A macrostate can contain a parallel composition of separate concurrent STGs. Graphical notation: dashed separation line.
- STGs are coupled by shared signals.
- A local signal is declared by the keyword signal and its scope is the containing macrostate.
- A set of (concurrent) active states is called a configuration.



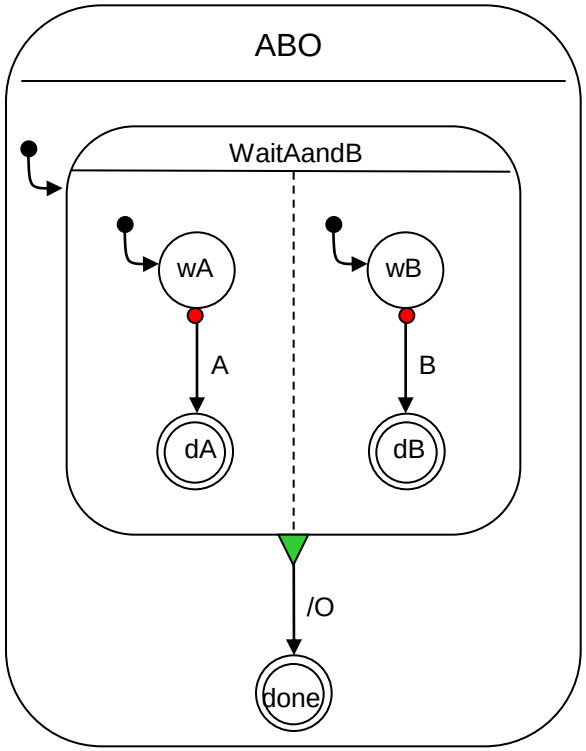
- Notation:
  - Active state
  - Taken transition
  - Emitted signal **S**
  - S^+: presence of signal S
  - S^-: absence of signal S

Example Reaction

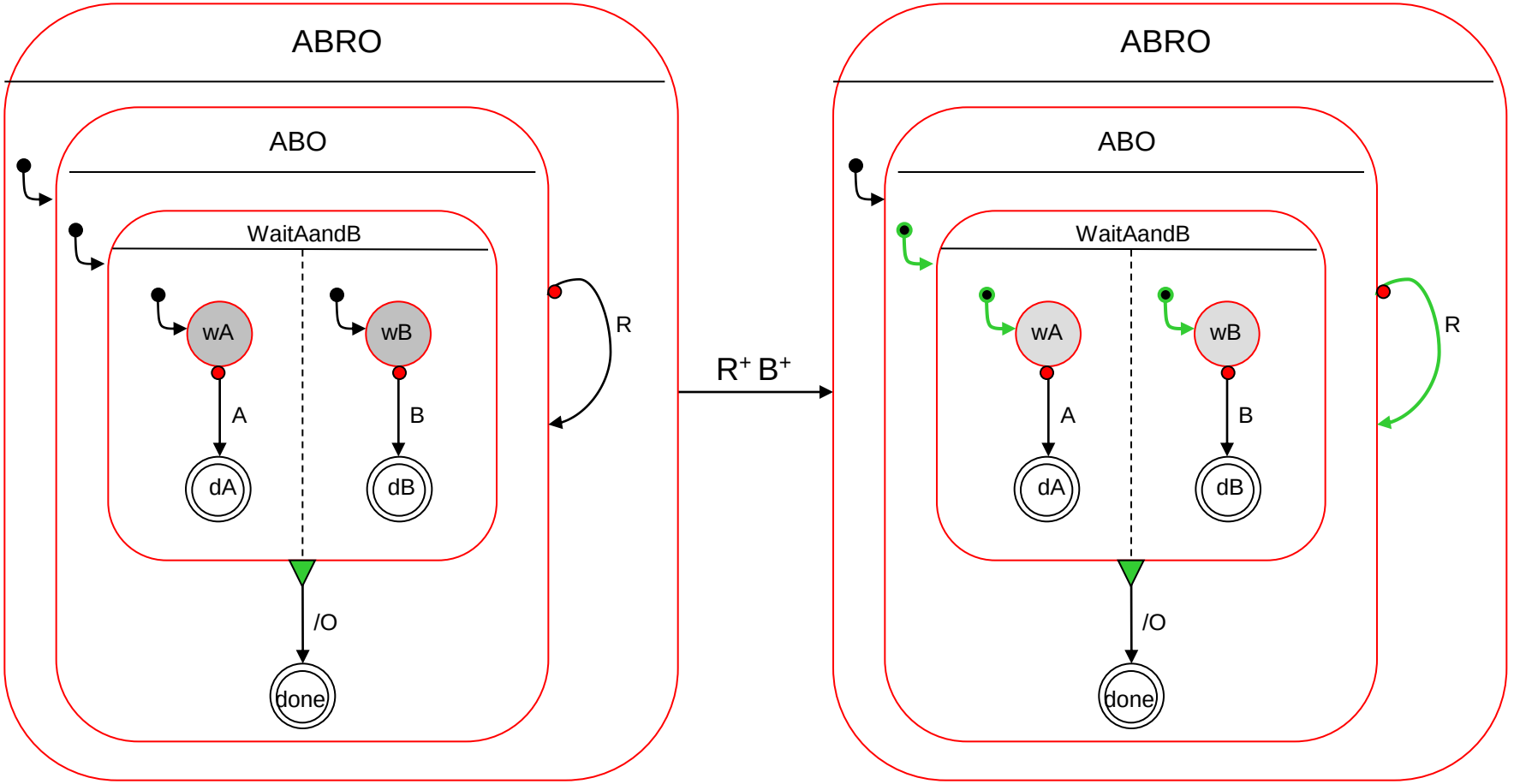


Concurrency and Normal Termination

- When each concurrent STG in a macrostate reaches a final state, then the macrostate is immediately exited by its normal termination transition.



Concurrency and Abort



Transitions

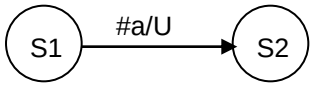
- A strong abort prevents any execution in the preempted state.

- For any state
  - every outgoing transition has a different priority
  - any strong abort transition has priority over any weak abort transition
  - any weak abort transition has priority over a normal termination transition

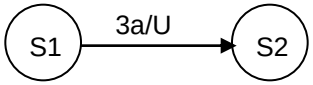
- There are no inter-level transitions.

SyncCharts: Advanced Constructs

- Immediate transition
  - Syntax: #<trigger>/<effect>



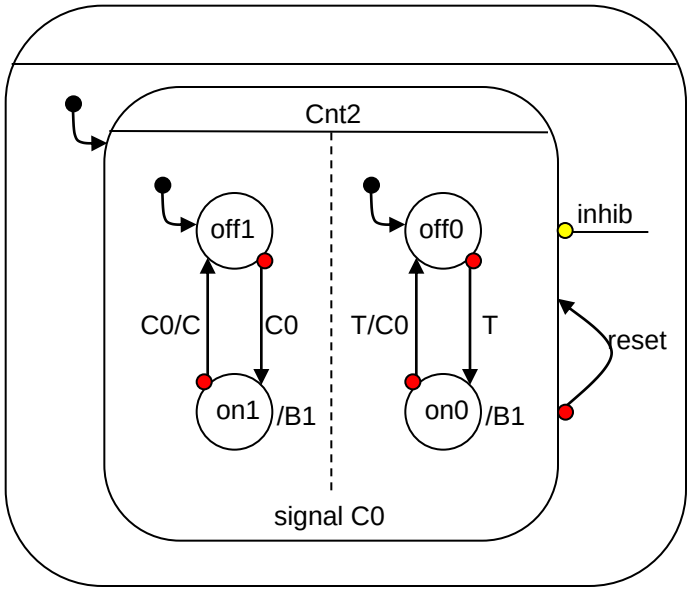
  - The trigger may be satisfied as soon as the state is entered: An active state waits for the satisfaction of the trigger of one of its outgoing transitions, at an instant strictly posterior to its entering, or immediately in case of an immediate transition.
- Count delays for transitions
  - Syntax: <factor><trigger>/<effect>



  - <factor> is the natural number of instants a transition must be active before it is executed. These active instants need not be consecutive, but the source state (S1) must be active all the time.

Suspension

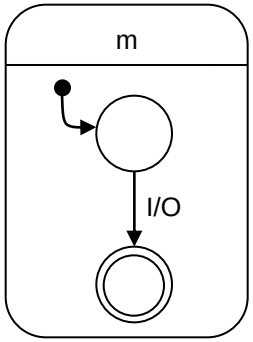
- A suspension is associated with a trigger. If the trigger is satisfied the reaction is suspended in the target state: the execution of the preempted state is frozen.
- Notation: T ,or: S T
- Note: aborts take priority over suspensions.



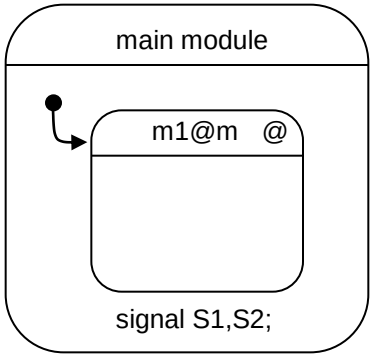
Run Modules

- Macrostates are states that have their own behavior (also called processes). They can be abstracted as modules, similarly to procedures in programming languages.
- Modules are instantiated by using run modules (corresponding to procedure calls).
- A signal interface renaming has to be defined for run modules.

Module M with interface
input I;
output O;

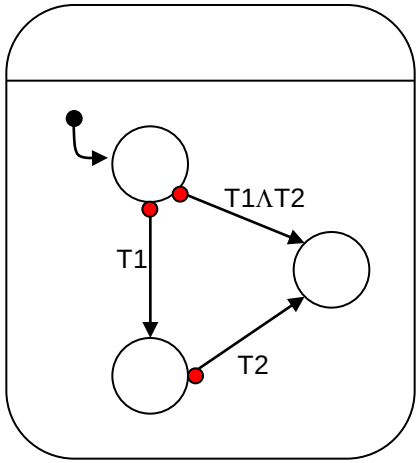


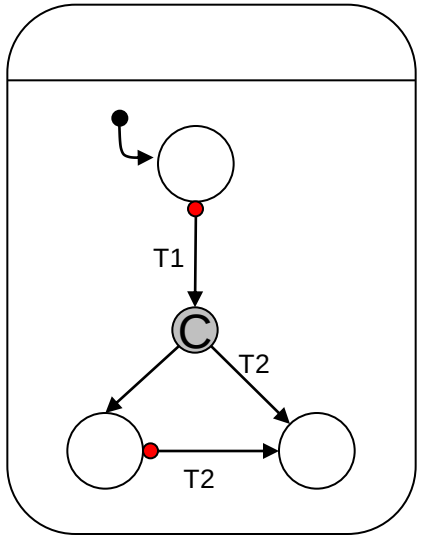
… used as a run module with the following signal binding:
signal S1 / I;
signal S2 / O;



Conditional Connector

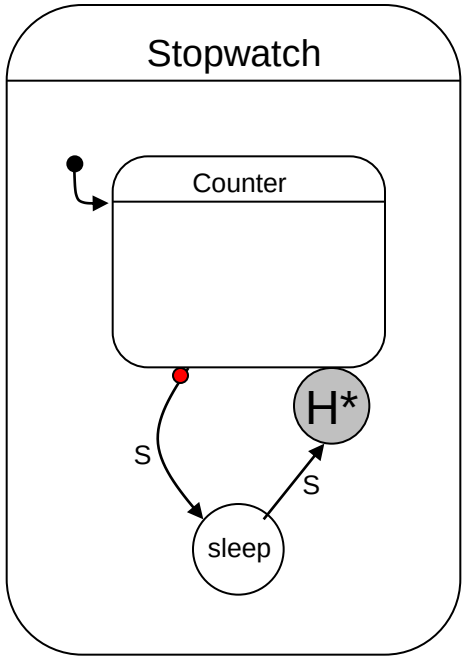
- Introduction of conditional pseudostate ©
- Concise representation of scenarios where a common trigger is shared by several outgoing transitions.
- All departing transitions are immediate.
- One departing "default" transition without condition must be present.



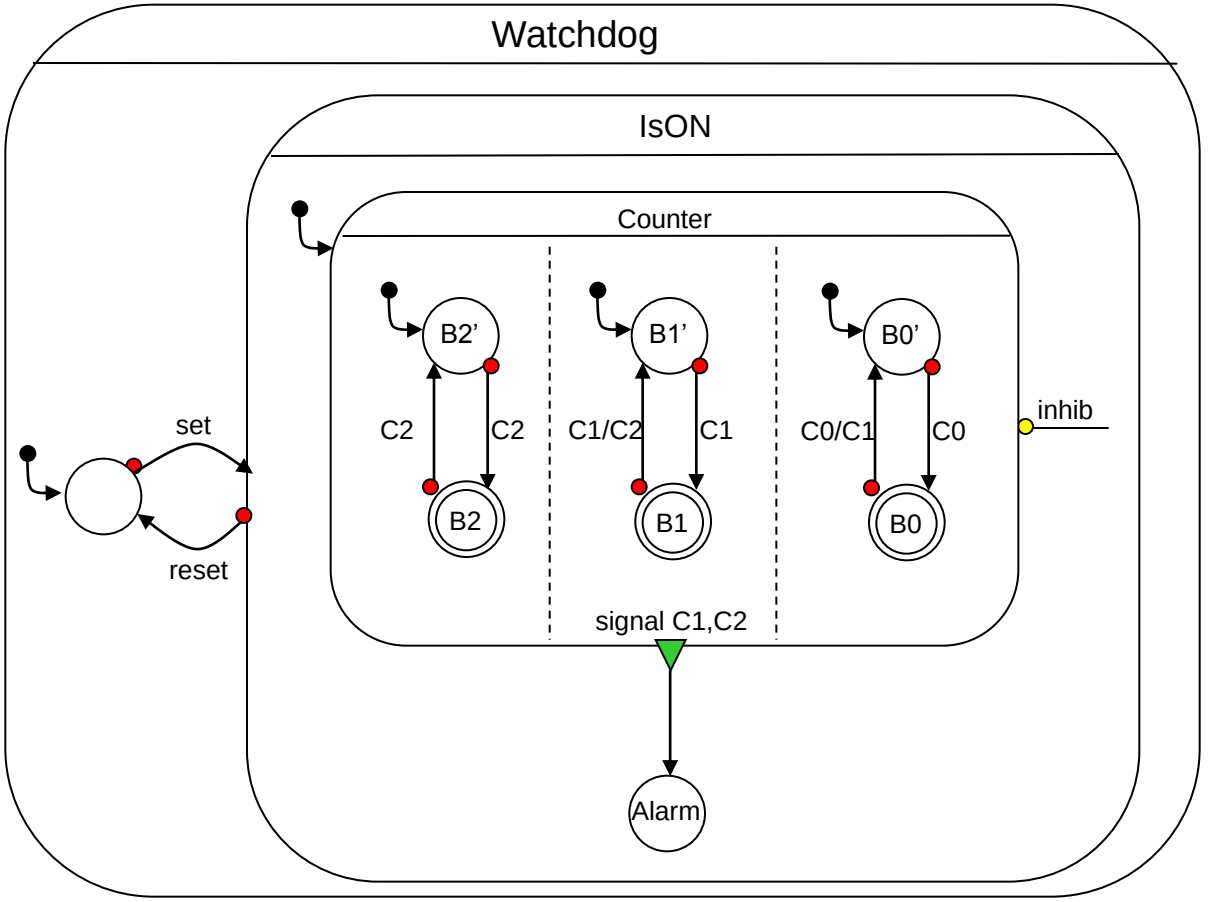


History Connector

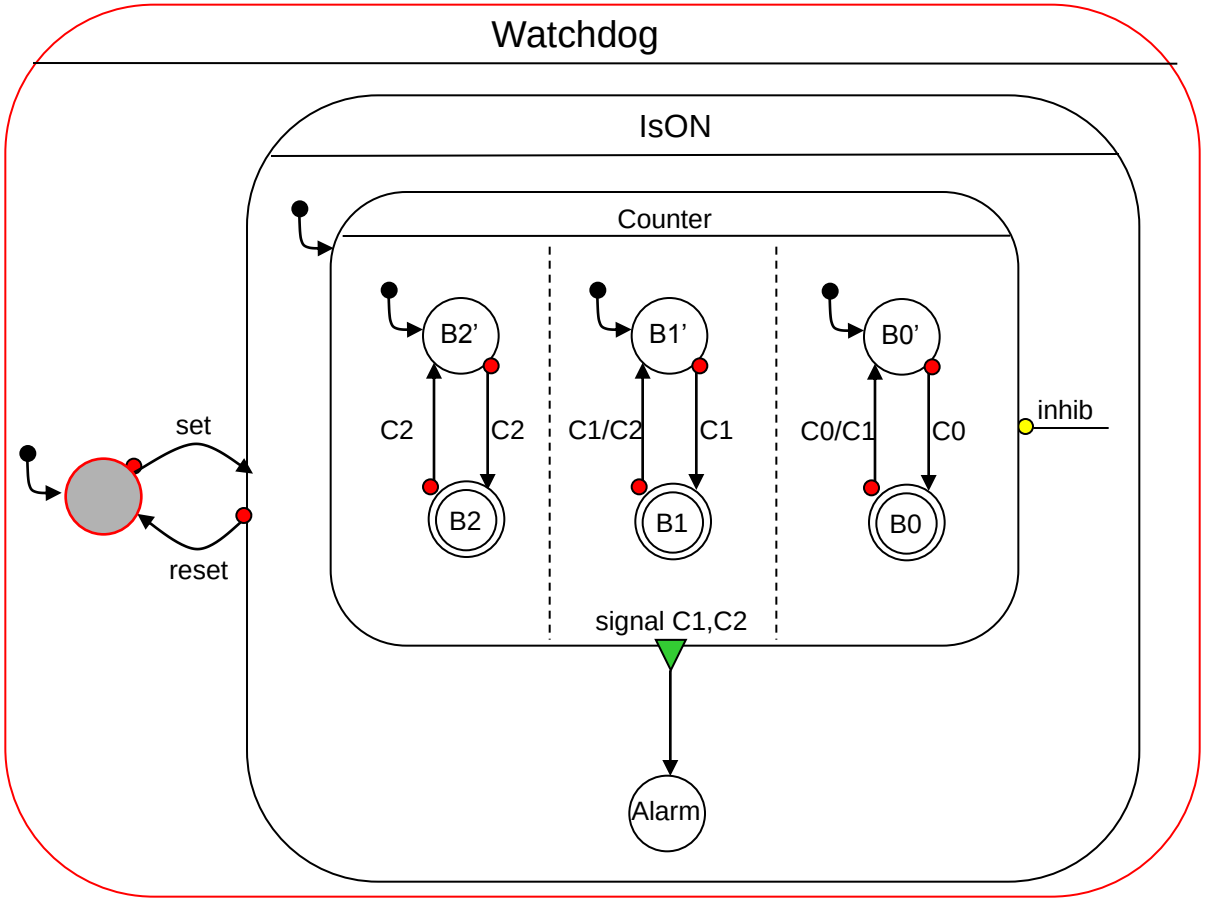
- Directly attached to macrostates
- Only incoming transitions can connect
- The previous state of the macrostate is restored when it is entered through a history connector.



Example: Watchdog

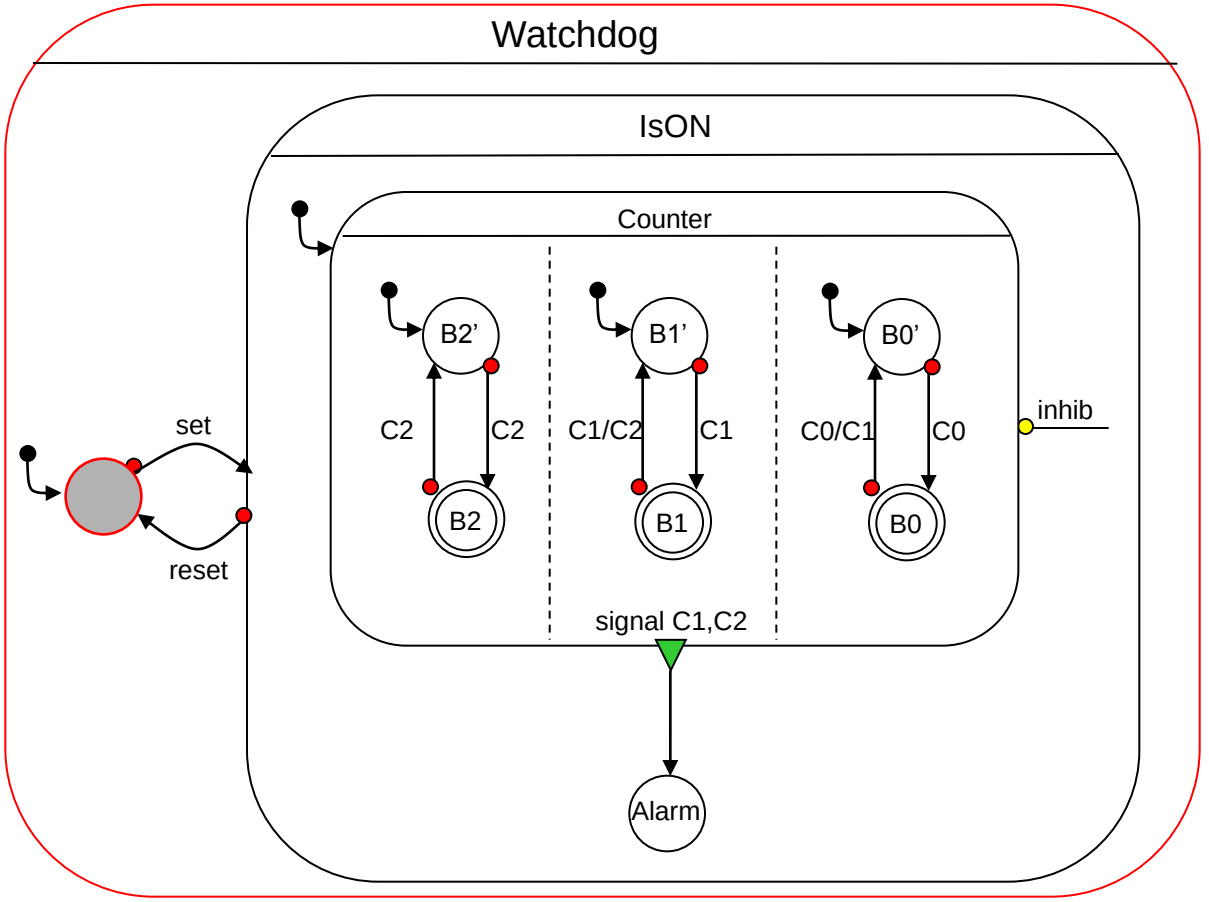


Example: Watchdog



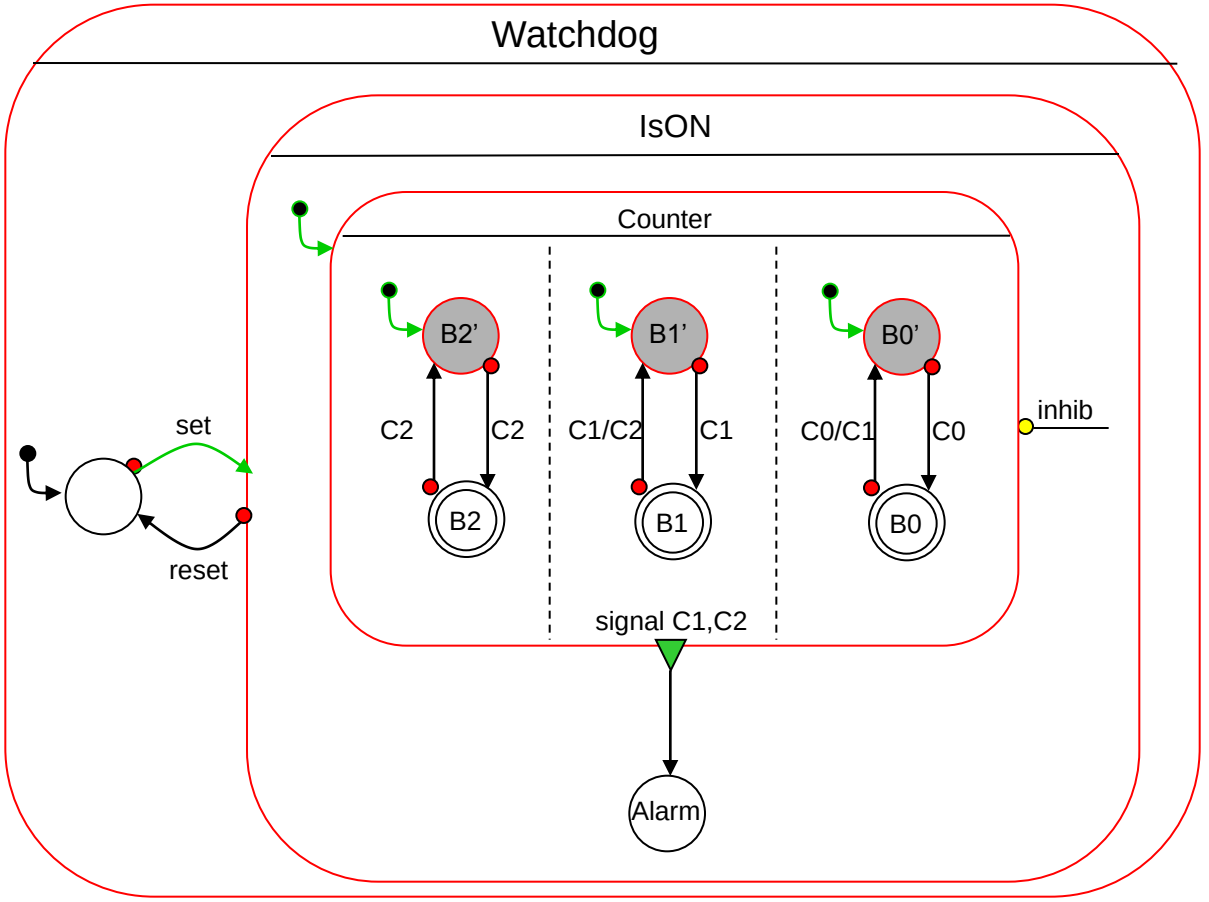
Example: Watchdog

$C0^+$



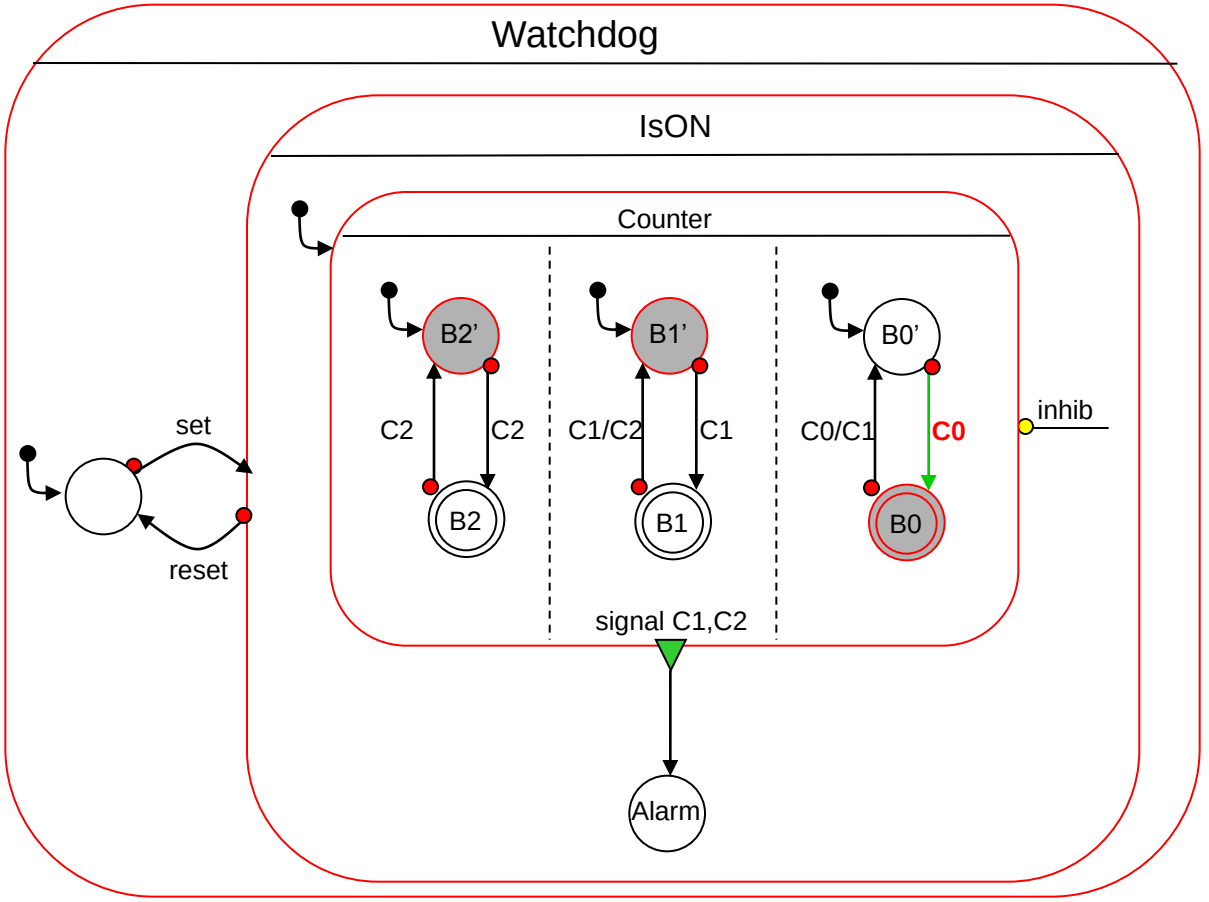
Example: Watchdog

set+



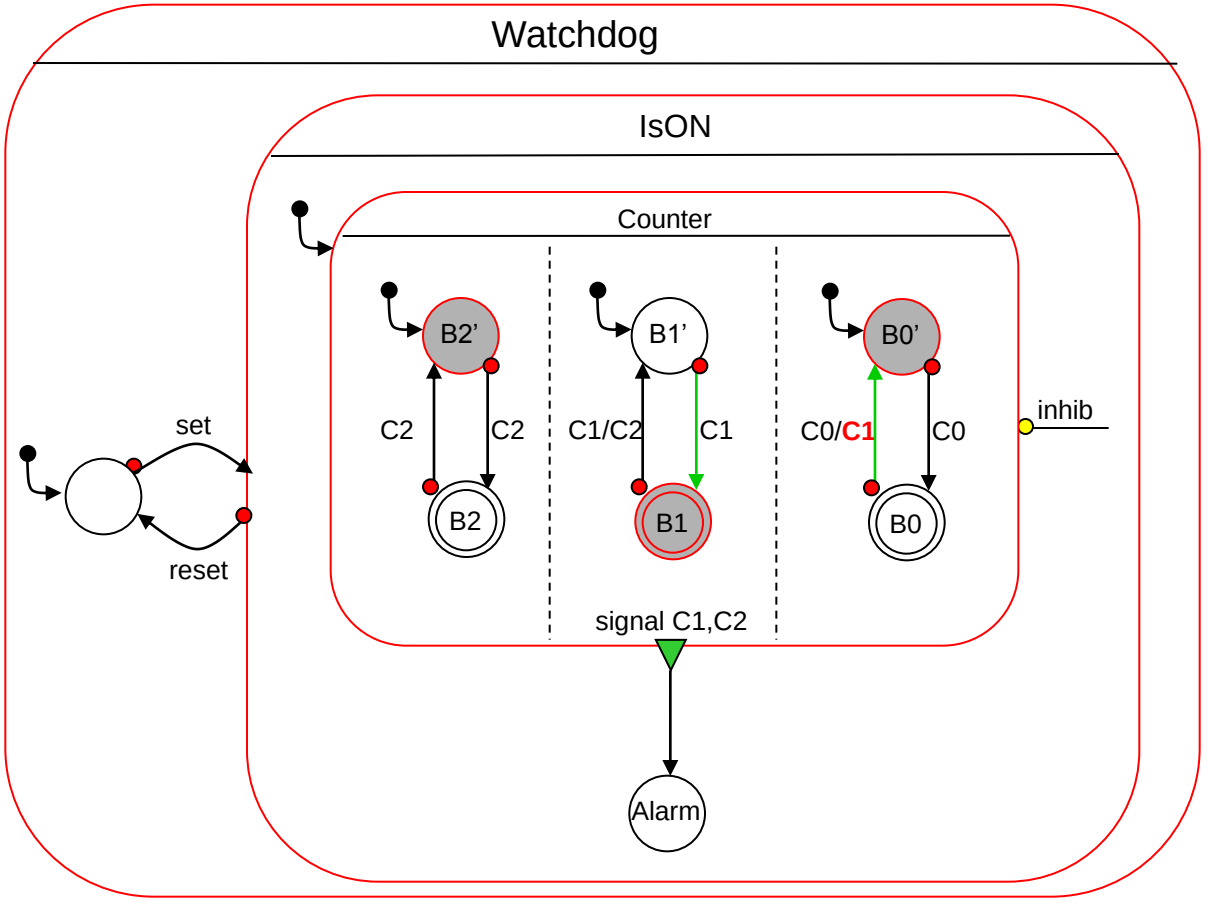
Example: Watchdog

$C0^+$



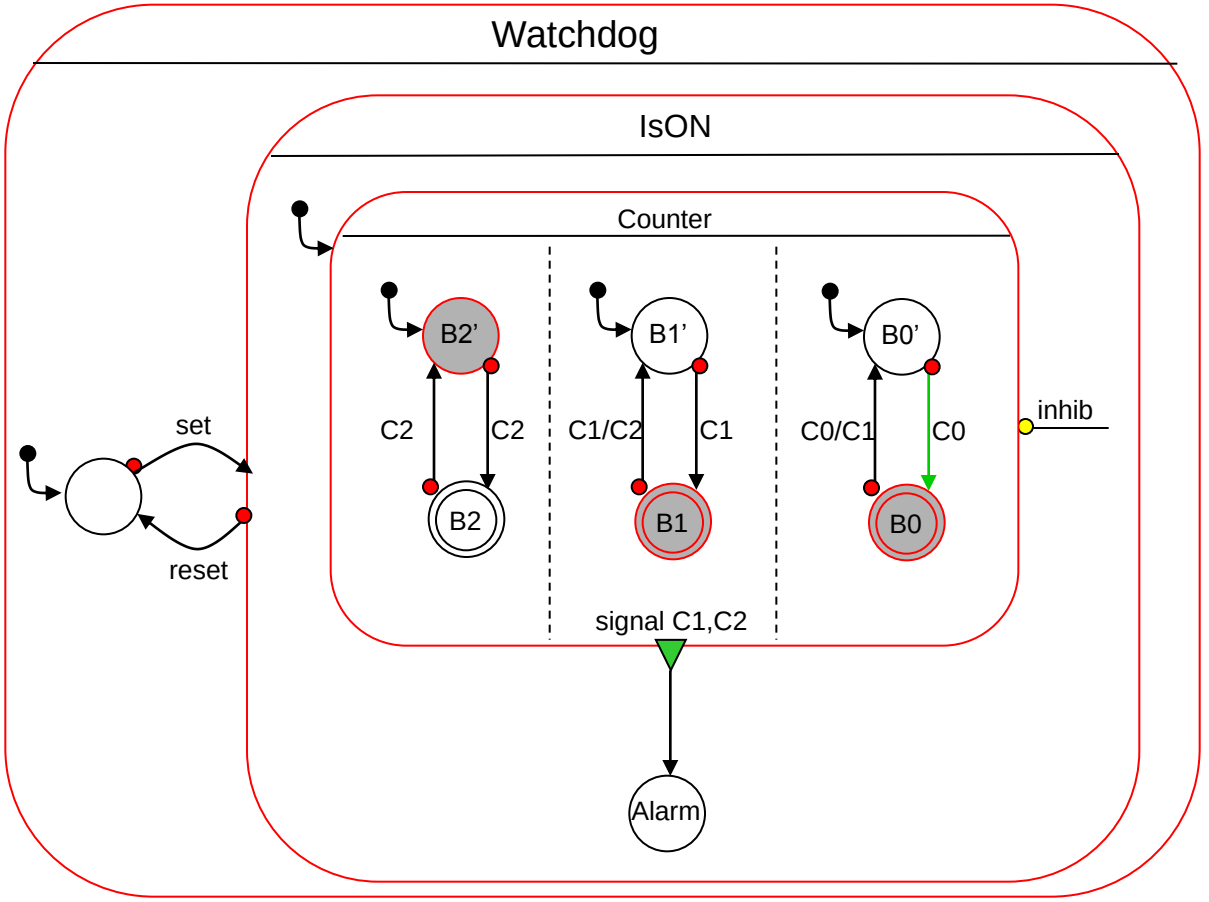
Example: Watchdog

C0+

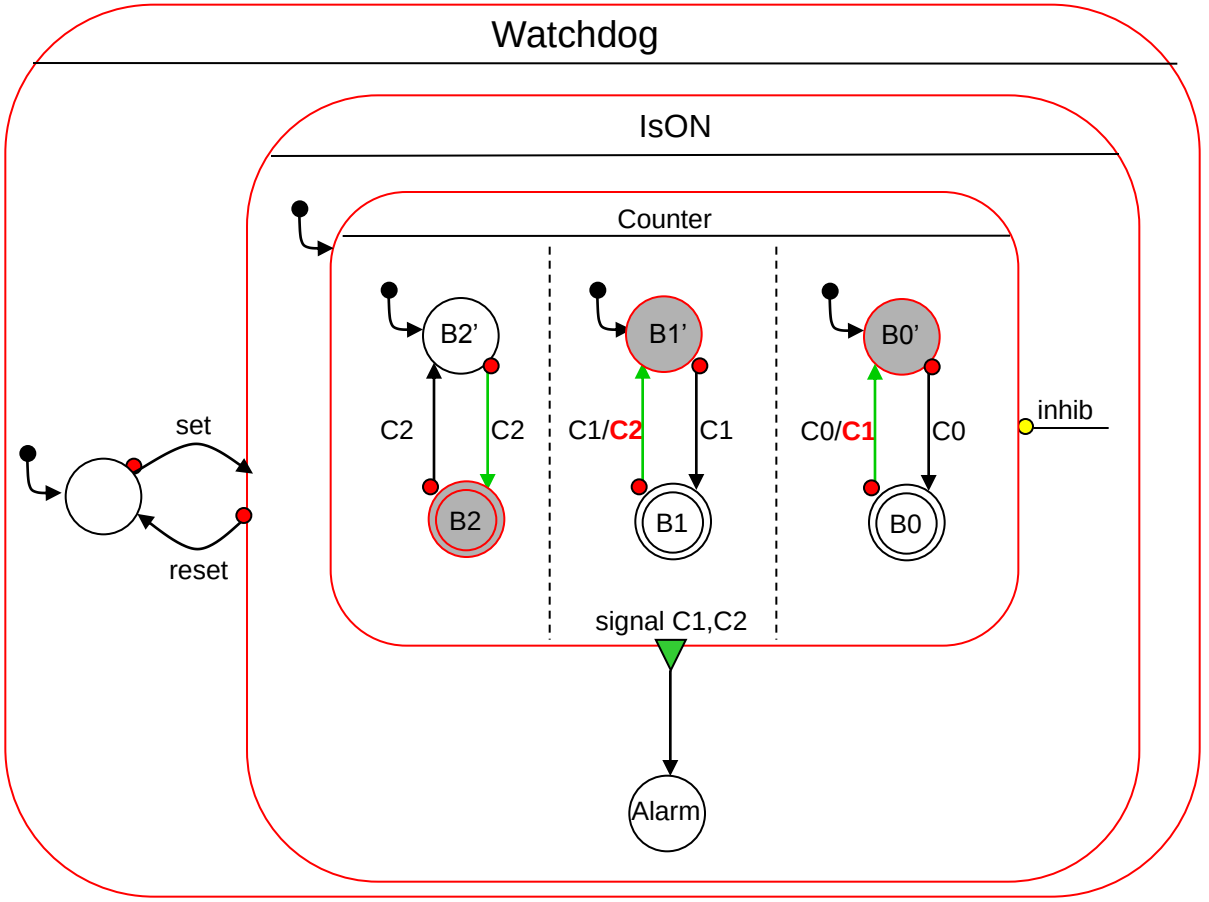


Example: Watchdog

C0+



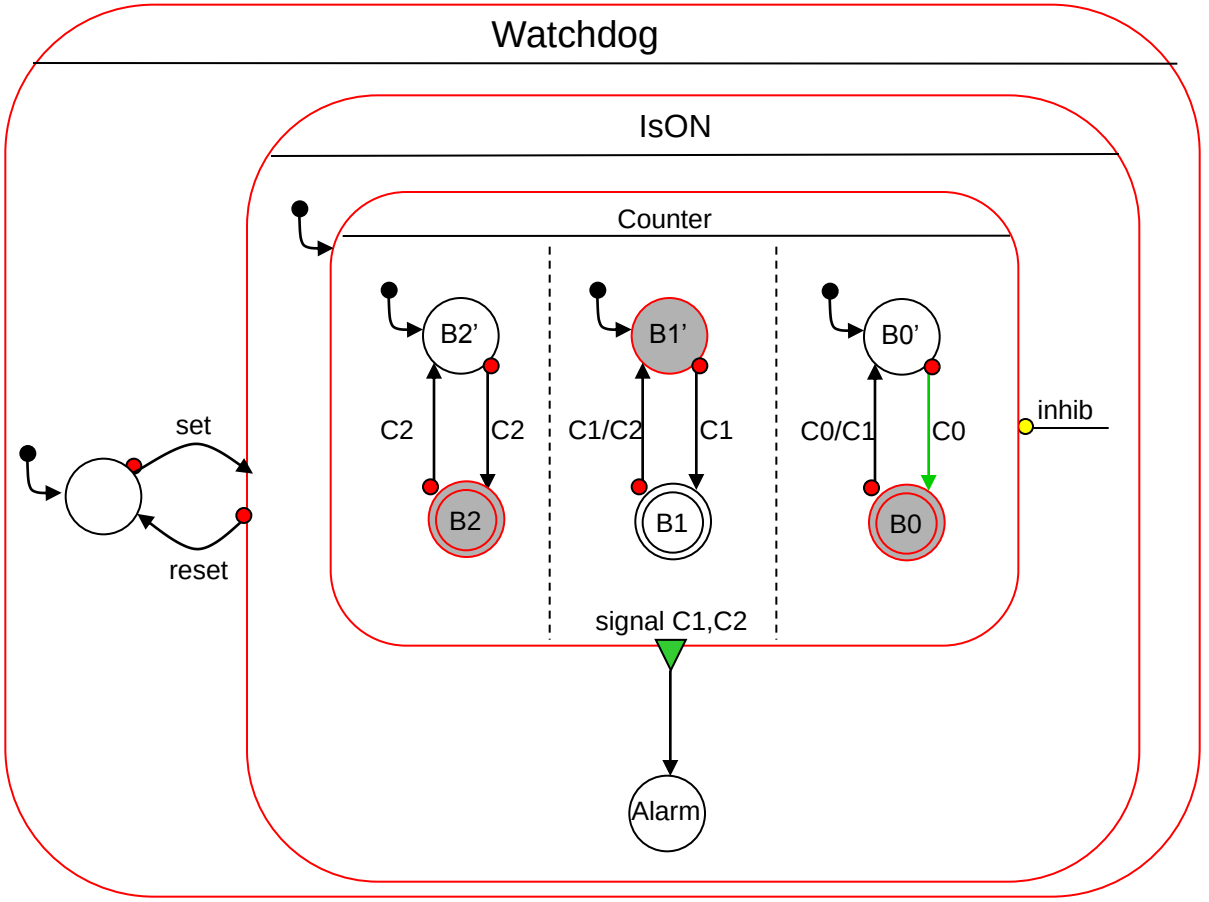
Example: Watchdog



C0⁺

Example: Watchdog

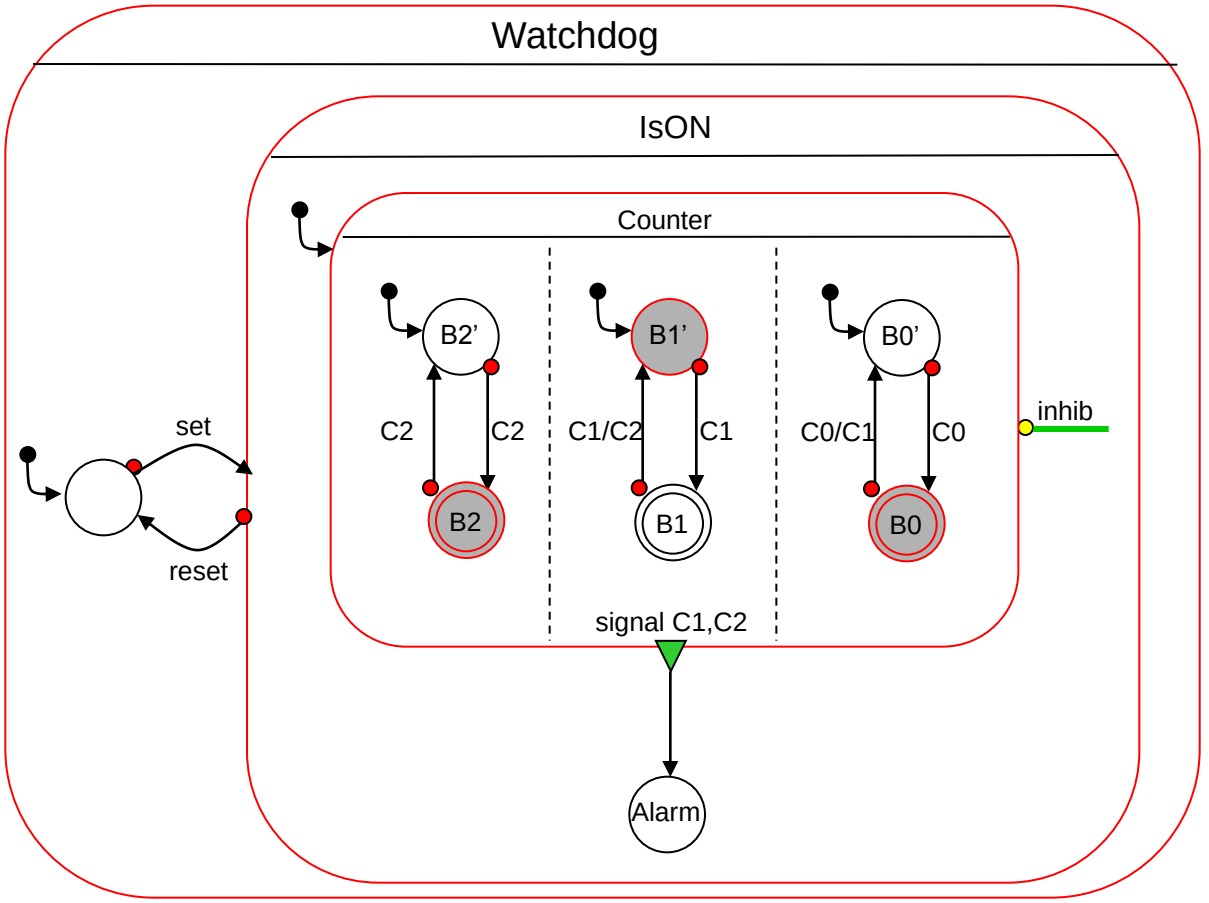
$C0^+$



Example: Watchdog

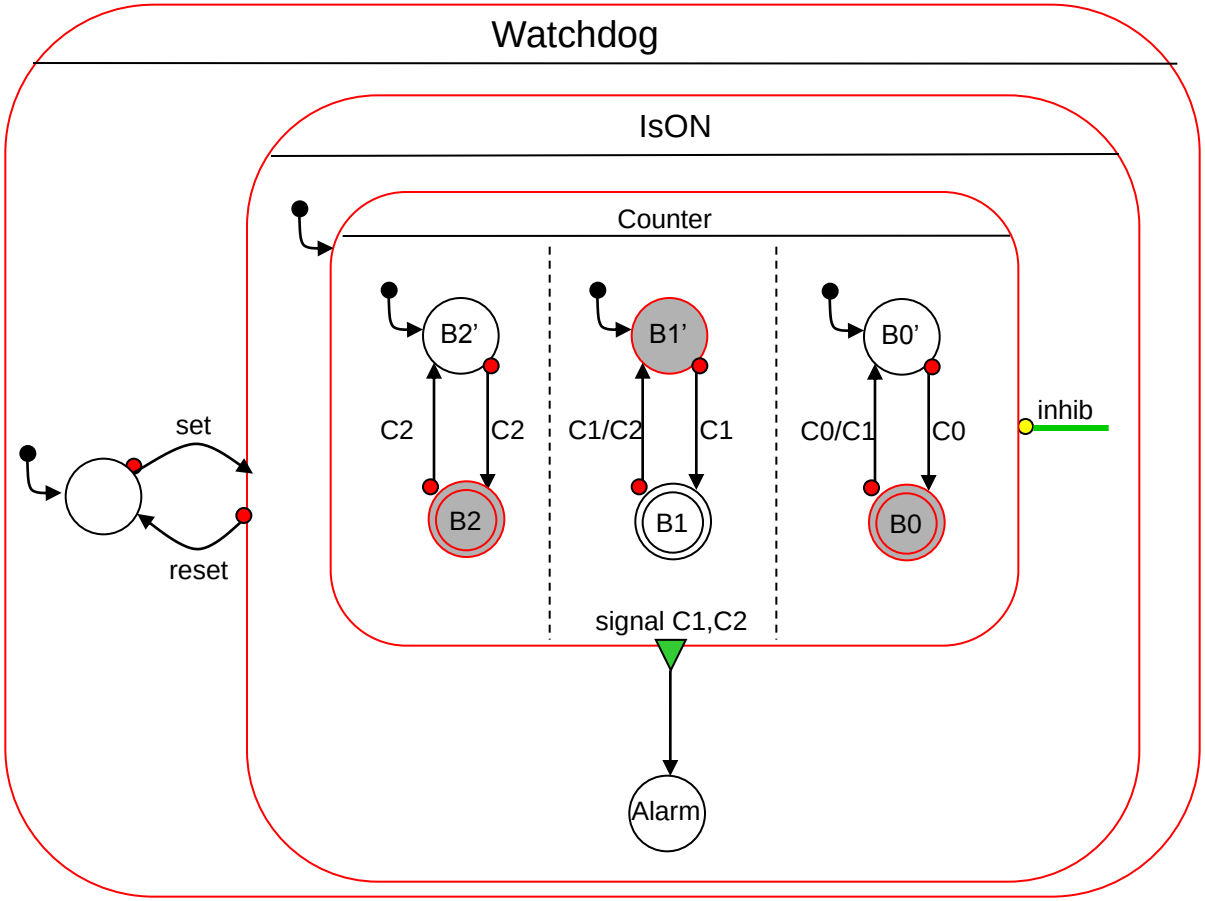
inhib+

C0+



Example: Watchdog

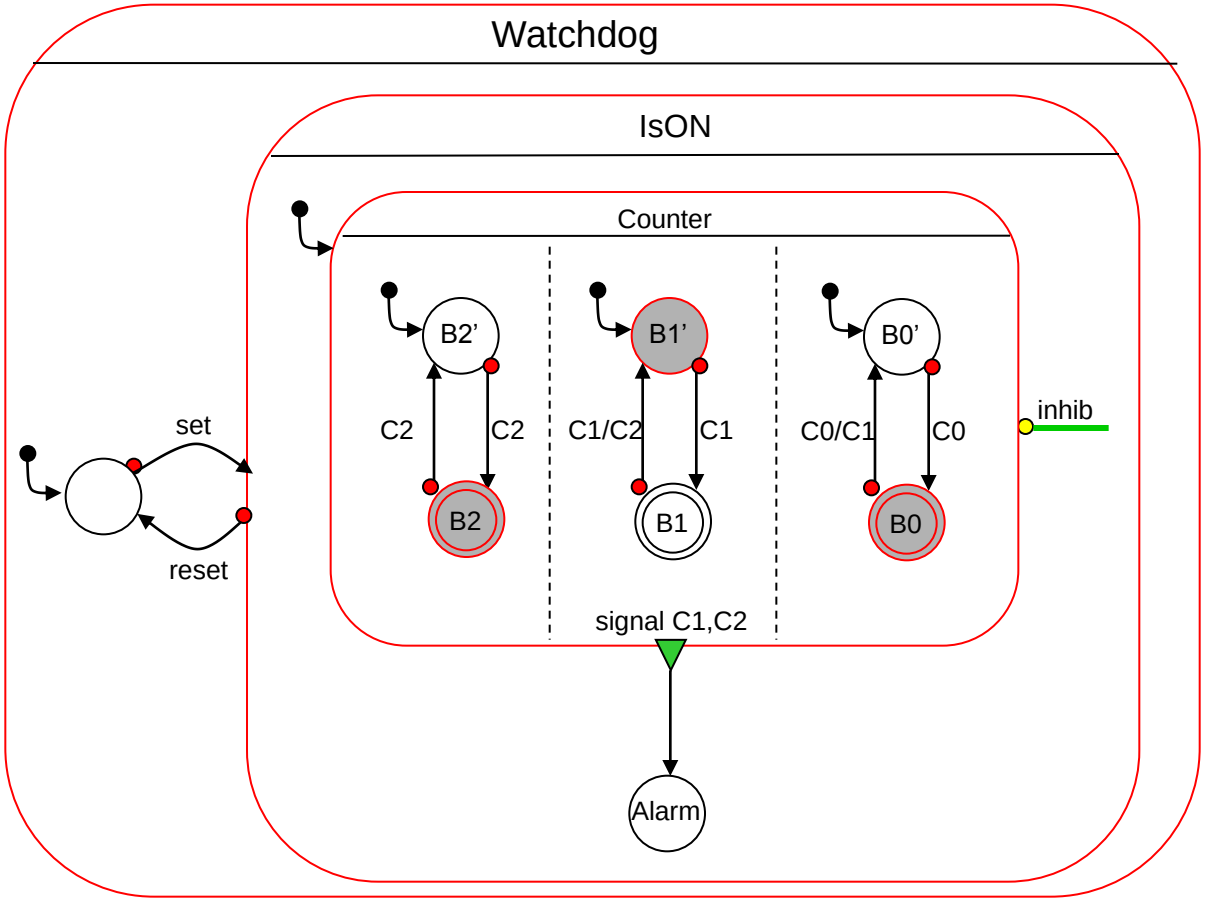
inhib+



Example: Watchdog

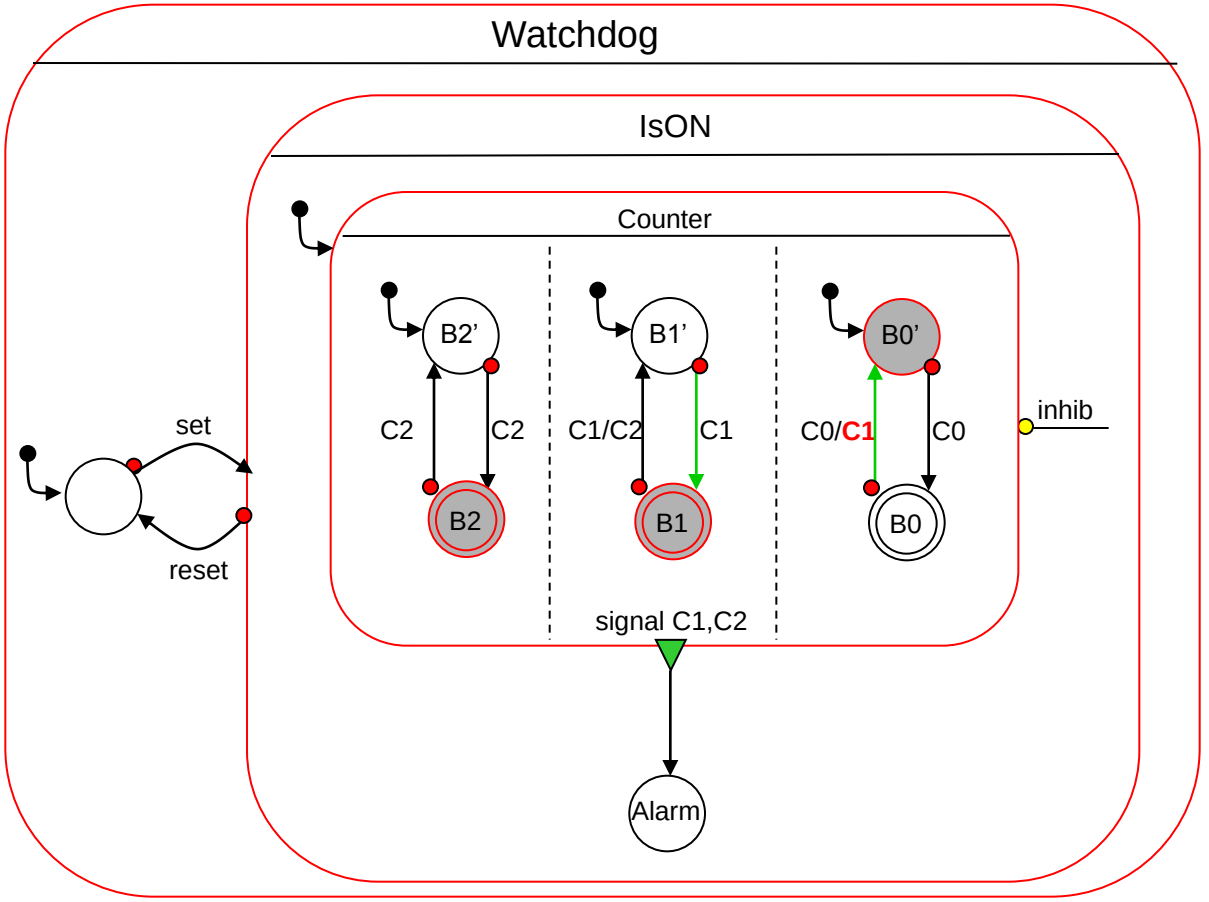
inhib+

C0+



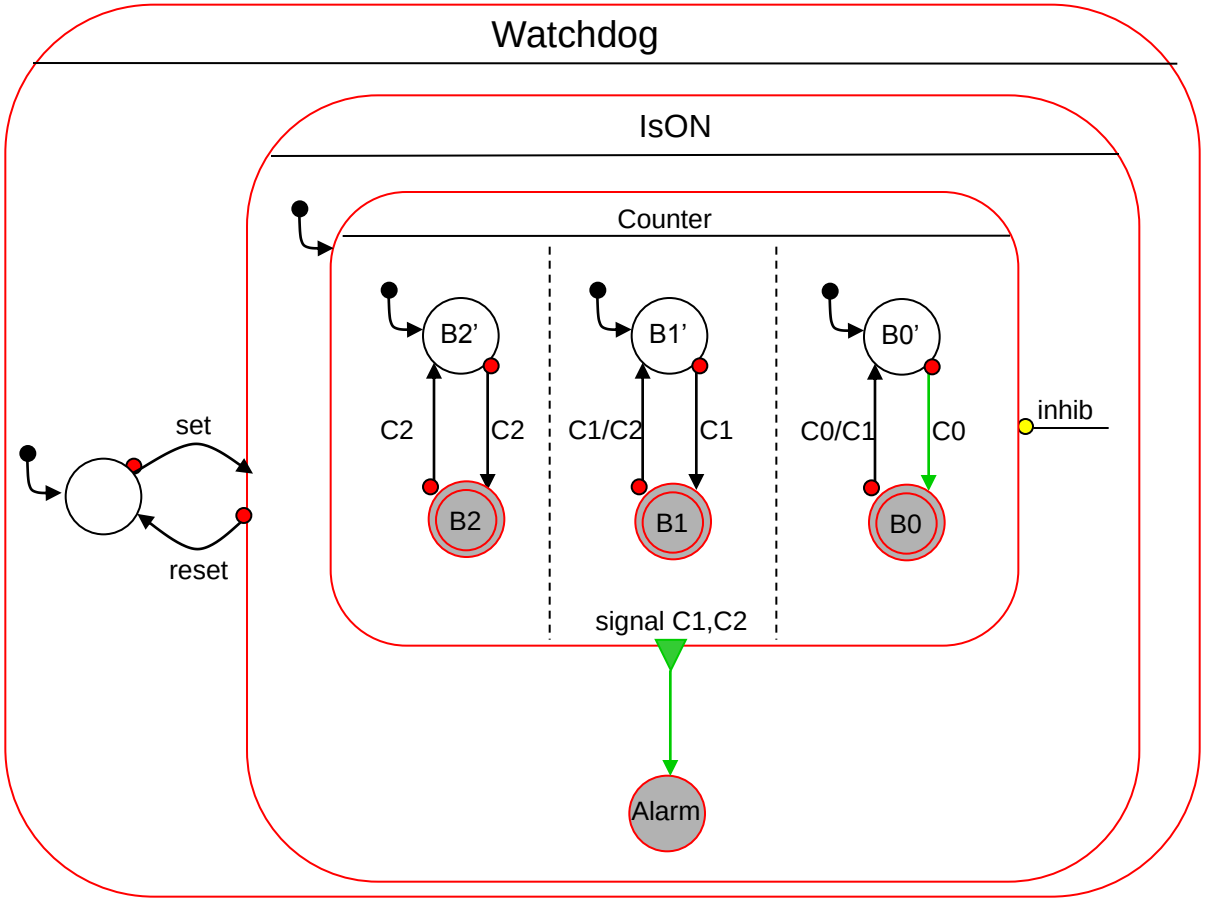
Example: Watchdog

C0+



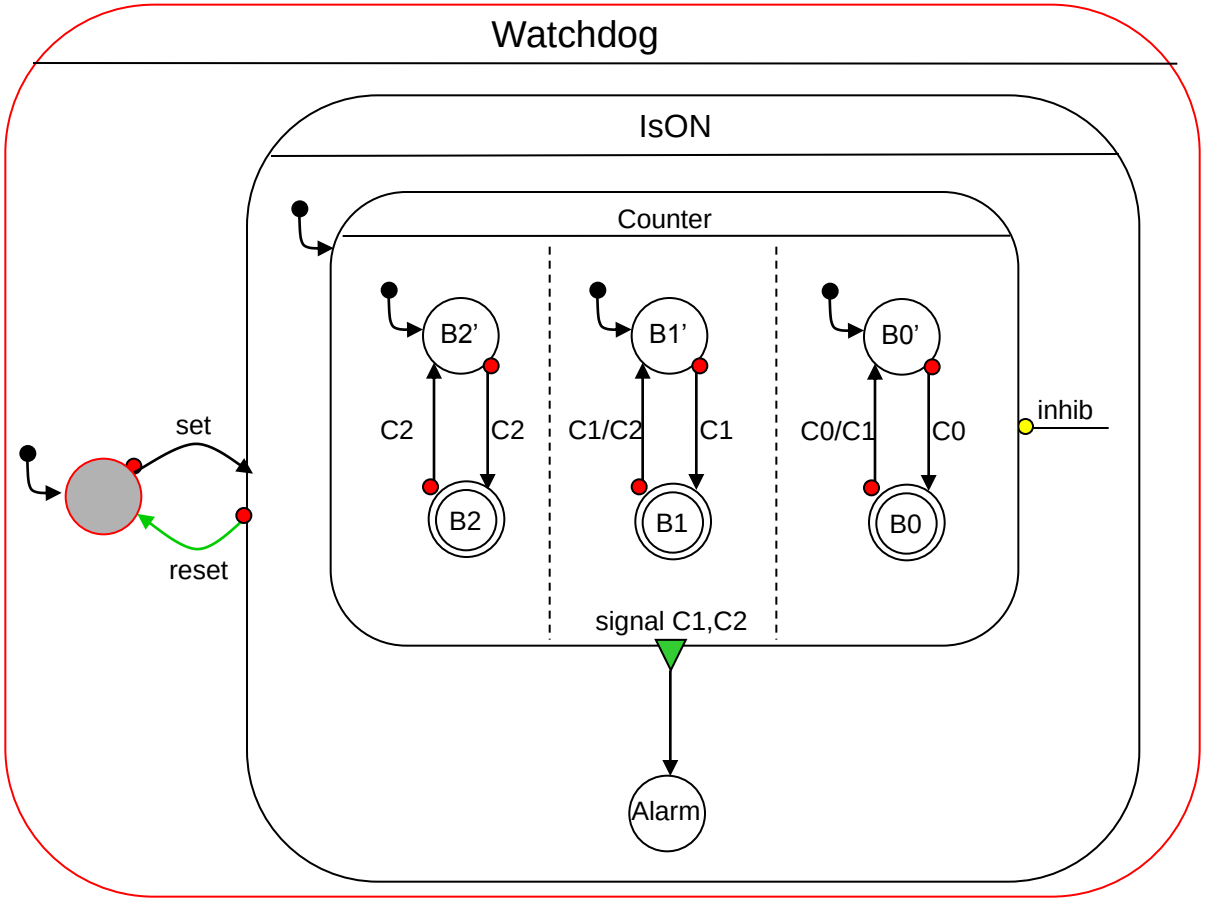
Example: Watchdog

C0+



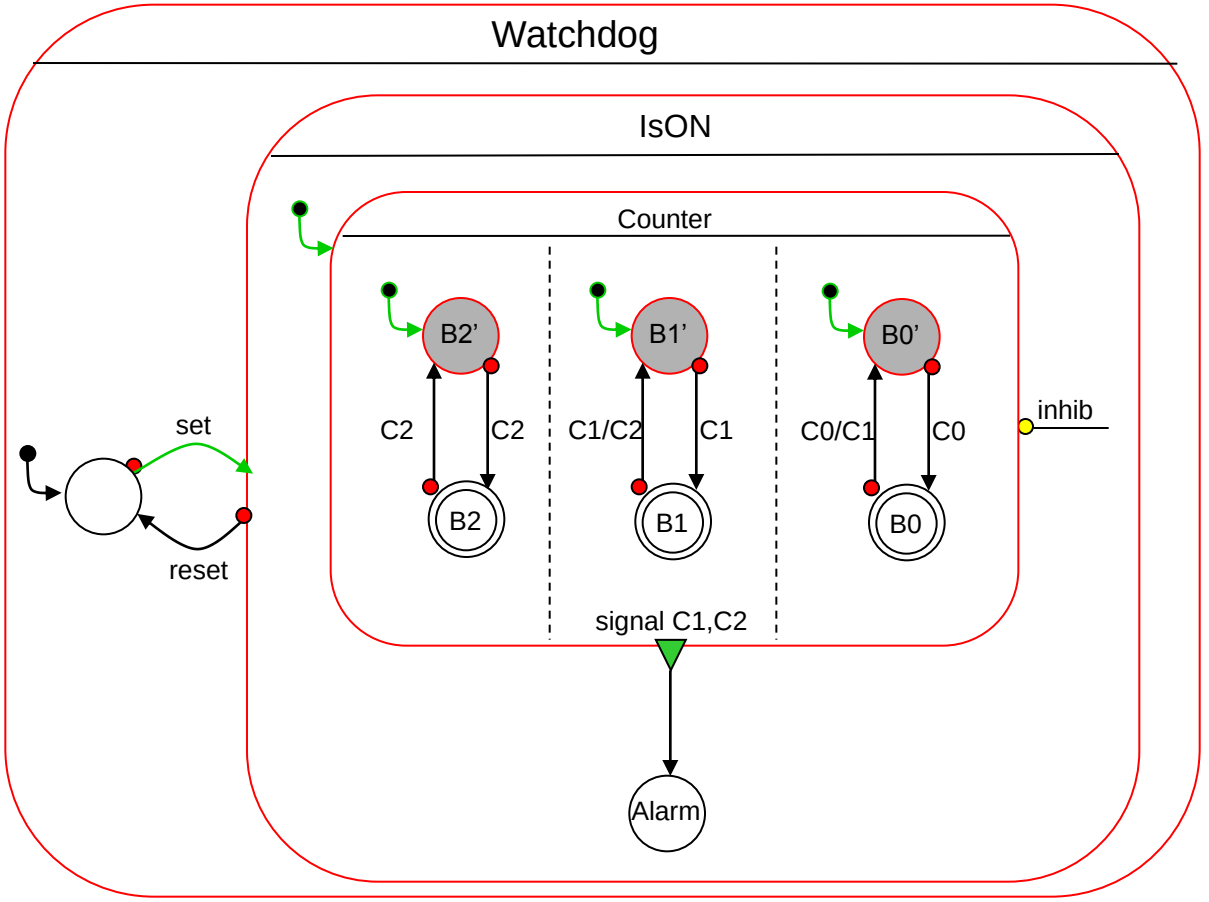
Example: Watchdog

reset+

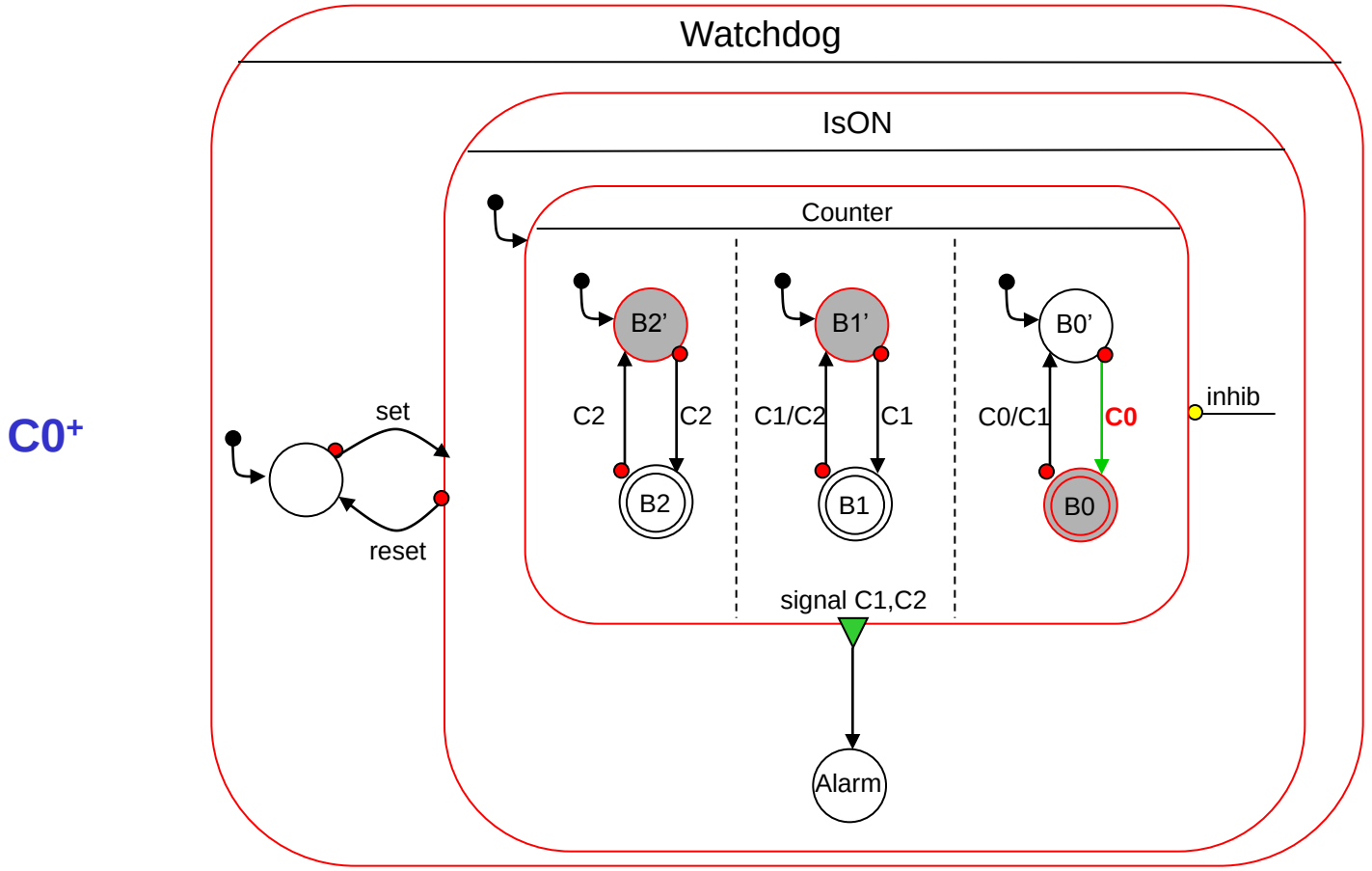


Example: Watchdog

set+



Example: Watchdog

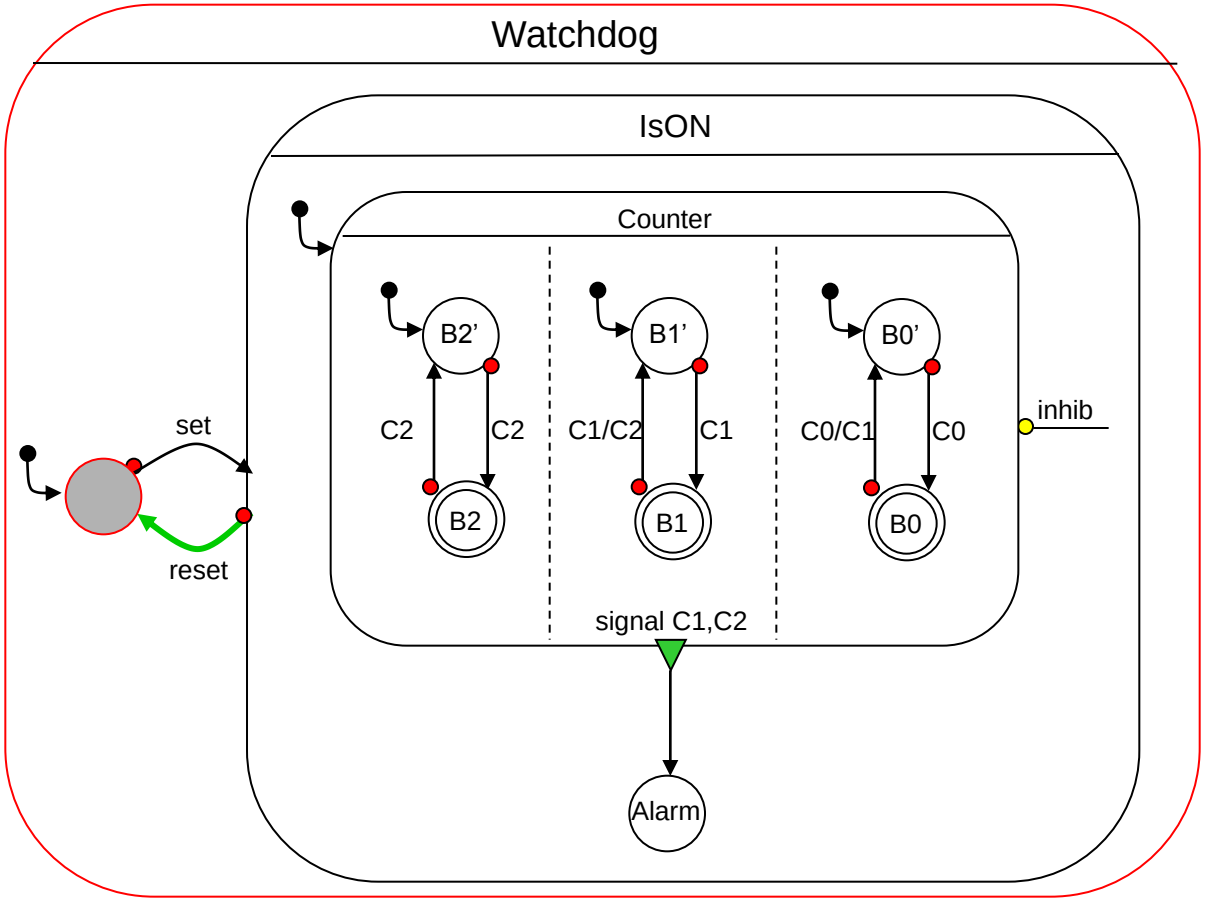


Example: Watchdog

reset⁺

inhib⁺

C0⁺

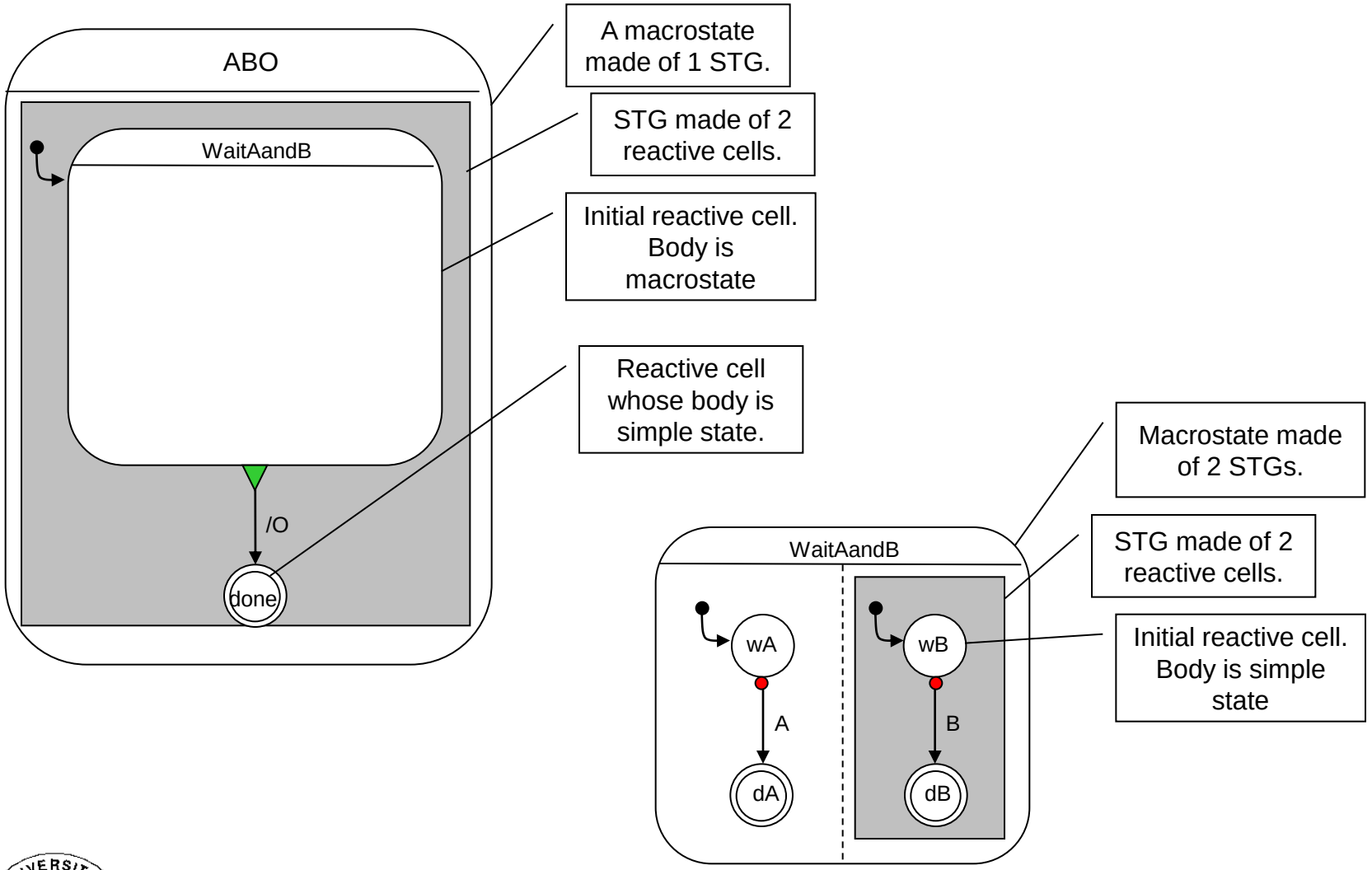


Computing a Reaction – Definitions

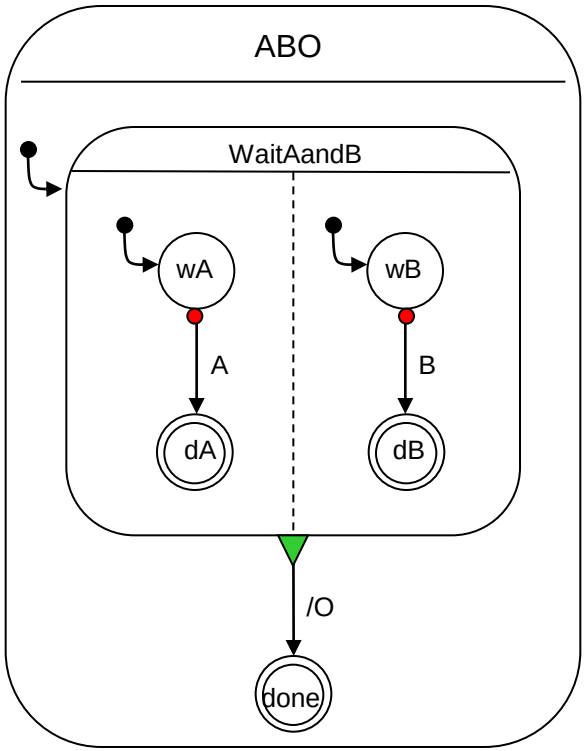
- Each SSM can be represented by unique macrostate, called *Top*, which designates the root of the state containment hierarchy.

- A state-transition graph (STG) *G* is a tuple *G = (S, ini)* where *S* is a non-empty set of reactive cells, and *ini* is the initial reactive cell.
- A reactive cell is a tuple *C=(B,R,S)* such that its body *B* is a state (simple state or macrostate) and *R* the set of all its outgoing transitions. The status *S* of a reactive cell is either *IDLE* or *ACTIVE*.
- A macrostate *M* is a quadruple *M=(G,I,O,L)* composed from a non-empty set *G* of STGs, and three possibly empty sets of signals: input signals (*I*), output signals (*O*), local signals (*L*).
- A transition has a destination and a label. The destination is a reactive cell, the label is composed of three optional fields: a trigger, a guard, an effect. A transition is denoted as a quadruple *R=<type, trigger, effect, targetID>.* Feasible types are *sA* (strong abort), *wA* (weak abort), *nT* (normal termination).

Illustration

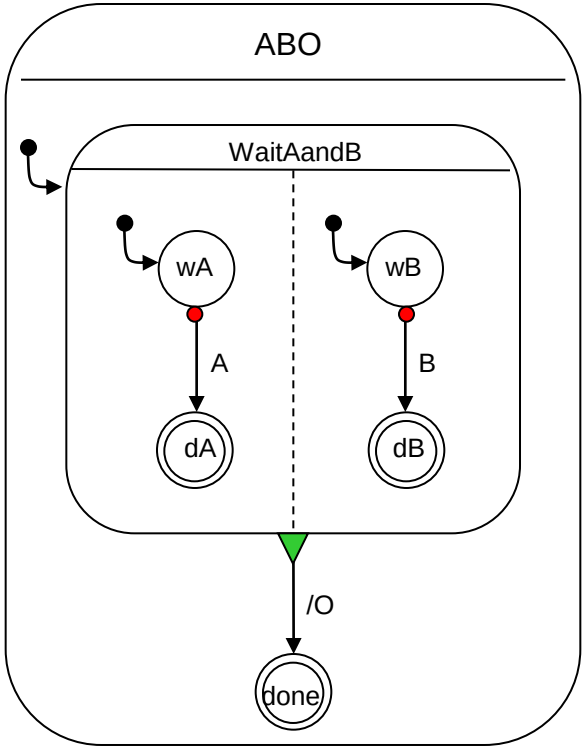


Detailed Example



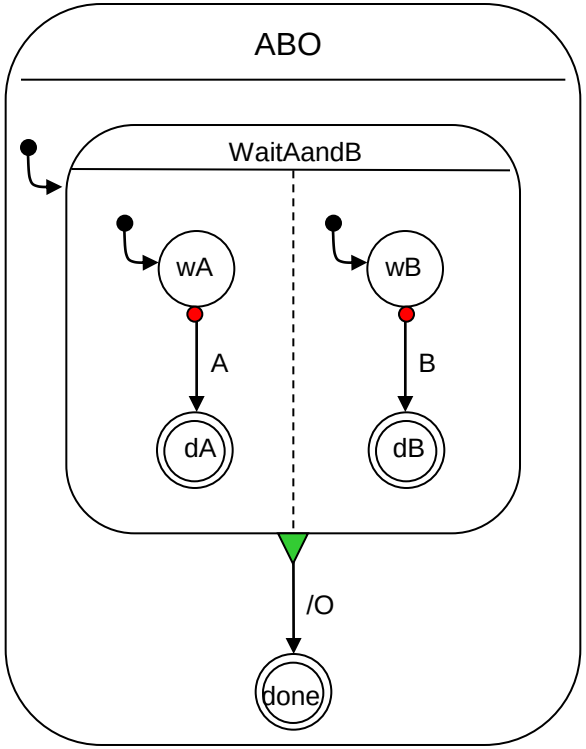
- Macrostate ABO=TOP:
  - ABO.G = {ABO.G_1}
  - ABO.I={A,B}
  - ABO.O={O}
  - ABO.L= Ø
- State-Transition Graph ABO.G_1:
  - ABO.G_1.S = {$RC_{WaitAandB}$, RC_{done}}
  - ABO.G_1.ini = $RC_{WaitAandB}$
- Reactive cell $RC_{WaitAandB}$
  - $RC_{WaitAandB}$.B={WaitAandB}
  - $RC_{WaitAandB}$.R={<nT,,O,RC_{done}>}
- Reactive cell RC_{done}
  - RC_{done}.B={done}
  - RC_{done}.R= Ø

Detailed Example



- Macrostate WaitAandB:
  - WaitAandB.G={WaitAandB.G_1, WaitAandB.G_2}
  - WaitAandB.I={A,B}
  - WaitAandB.O= Ø
  - WaitAandB.L= Ø
- State-Transition Graph WaitAandB.G_1:
  - WaitAandB.G_1.S = {RC_{wA}, RC_{dA}}
  - WaitAandB.G_1.ini = RC_{wA}
- State-Transition Graph WaitAandB.G_2:
  - WaitAandB.G_1.S = {RC_{wB}, RC_{dB}}
  - WaitAandB.G_1.ini = RC_{wB}

Detailed Example

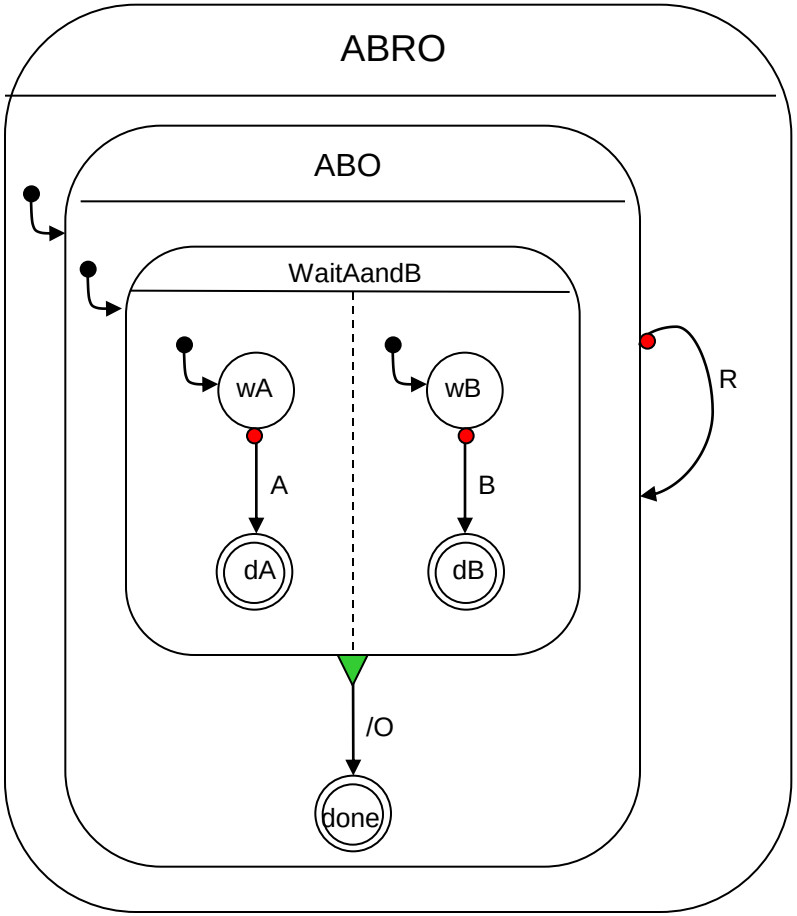


- Reactive cell RC_{wA}
  - $RC_{wA}.B = \{wA\}$
  - $RC_{wA}.R = \{<sA,A,,RC_{dA}>\}$
- Reactive cell RC_{dA}
  - $RC_{dA}.B = \{dA\}$
  - $RC_{dA}.R = \emptyset$
- Reactive cell RC_{wB}
  - $RC_{wB}.B = \{wB\}$
  - $RC_{wB}.R = \{<sA,B,,RC_{dB}>\}$
- Reactive cell RC_{dB}
  - $RC_{dB}.B = \{dB\}$
  - $RC_{dB}.R = \emptyset$
- Simple states done, wA,dA,wB,dB

Configurations

- A configuration is a maximal set of states (macrostates or simple states) the system could be in simultaneously. (Note that formally status is associated with reactive cells.)
- Let T be the top macrostate associated with an SSM. A legal configuration C for T must satisfy the following rules:
  1. T in C
  2. If a macrostate M is in C, then from each STG G directly contained in M there is exactly one state directly contained in G.
  3. C is maximal and contains only states satisfying rules 1 and 2.
- A stable configuration is a legal configuration that the SSM can reach after a sequence of reactions. Only the stable configurations are of interest for the user.

Example

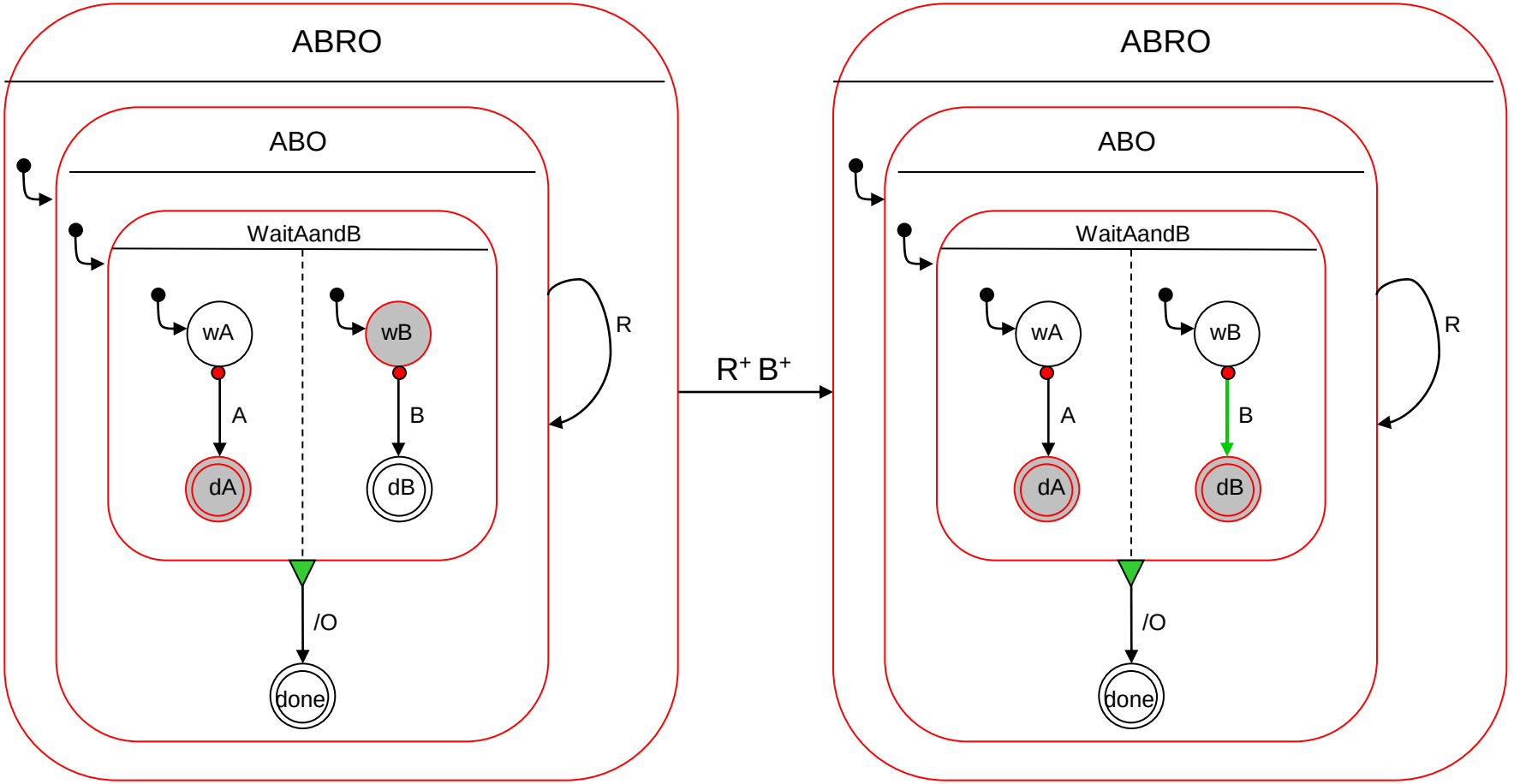


- Legal configurations:
  - {ABRO,ABO,done}
  - {ABRO,ABO,WaitAandB,wA,wB}
  - {ABRO,ABO,WaitAandB,wA,dB}
  - {ABRO,ABO,WaitAandB,dA,wB}
  - {ABRO,ABO,WaitAandB,dA,dB}
- Stable configurations:
  - all legal configurations except {ABRO,ABO,WaitAandB,dA,dB}

Computing a reaction

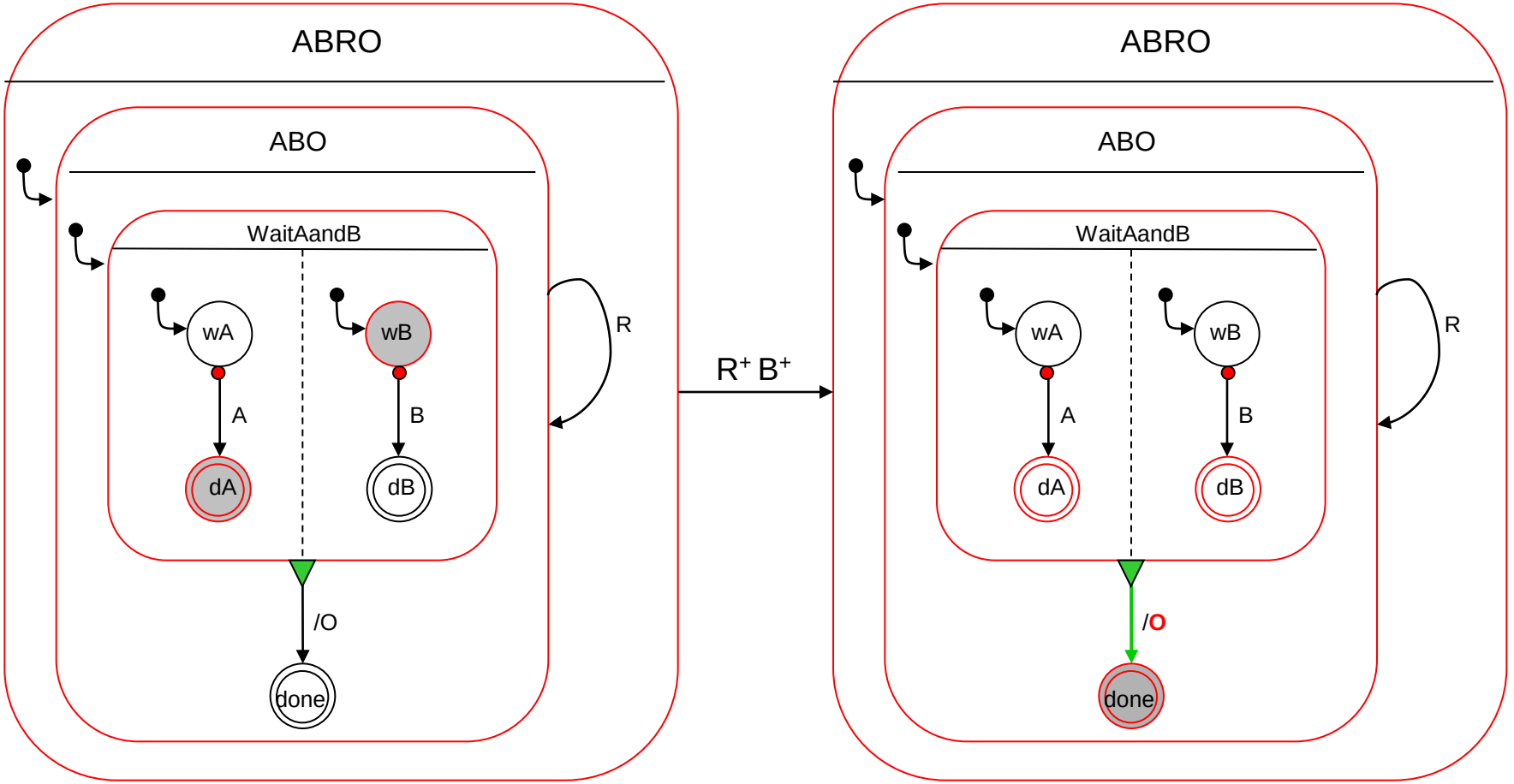
- Computing a reaction is done by concurrent threads which suspend their execution when a trigger cannot be evaluated and can resume when new signal statuses are broadcast.
- Reactions are computed as a sequence of microsteps, all executed during the same instant but in the order that respects causality.
- A transition is taken (ie a microstep is executed) only when its trigger is surely satisfied (no possibility of backtracking).
- Termination codes of components (reactive cell, STG, macrostate, simple state):
  - DONE: execution has been terminated
  - DEAD: nothing left to do at the current instant and in the future (final state); component is candidate to join a normal termination.
  - PAUSE: nothing left to do until next instant
  - Partial order: DEAD < PAUSE

Concurrency and Weak Abort



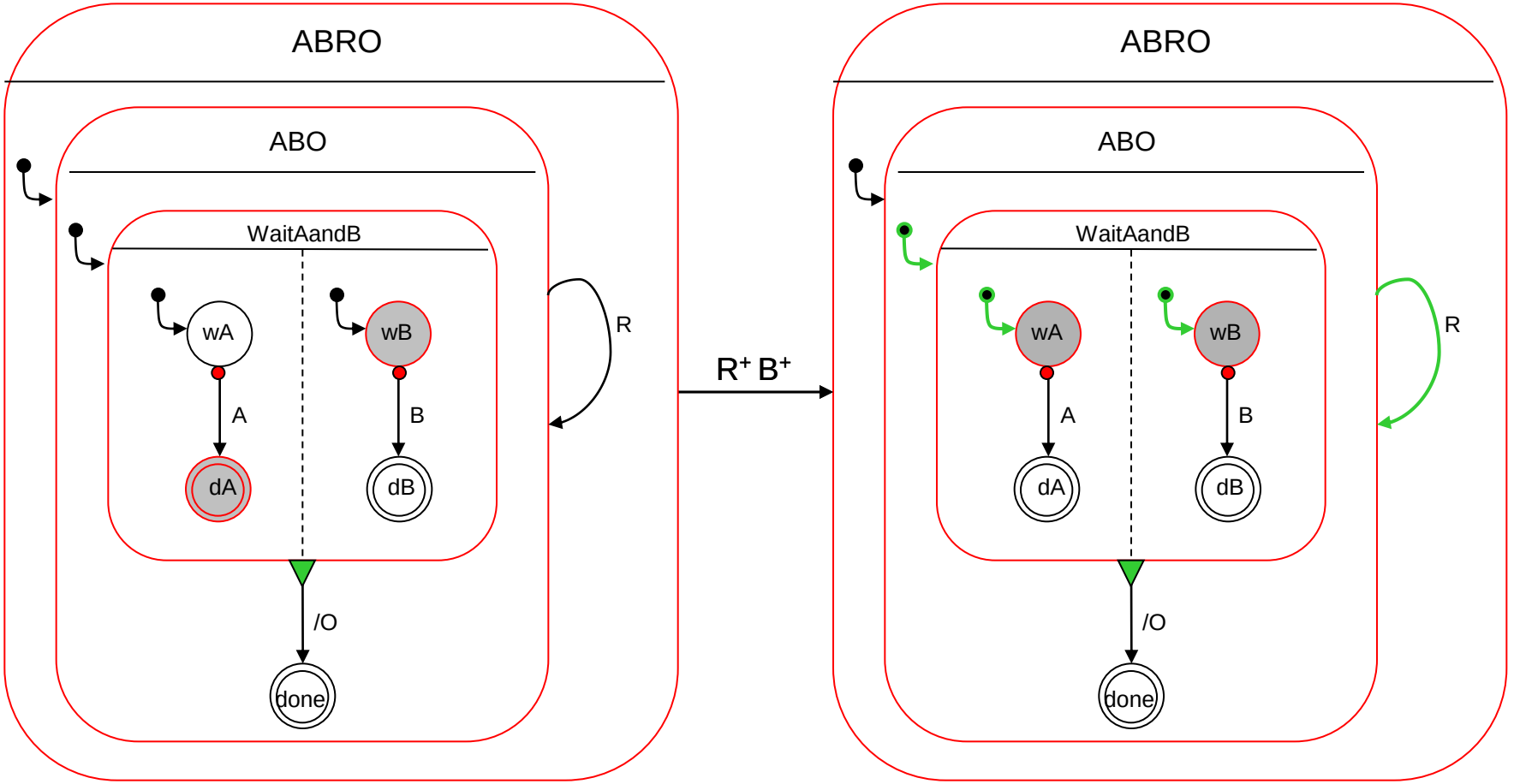
Microstep 1

Concurrency and Weak Abort



Microstep 2

Concurrency and Weak Abort



Microstep 3

Computing the reaction of an SSM

- Reaction of an SSM. Given a stable configuration, a reaction is computed by:
  1. Read input signals (presence status of all input signals is known)
  2. Set all output signals to the unknown presence status (⊥)
  3. Compute reaction of the top macrostate:
     react(T)
     /* yields emitted signals and the next stable configuration */

State Reaction

- Reaction of macrostate M:
  1. Set all local signals to ⊥
  2. For each STG G directly contained in M do in parallel
     - Compute reaction of STG G and store the termination code in c(g): c(G) = react(G)
  3. When all parallel executions are done
     - Compute C=maximum of c(G) for all STGs G in M
  4. Return C

- Reaction of a simple state S
  1. If S is a final state return DEAD.
  2. If an effect is associated with S, then emit all signals of the effect.
  3. return PAUSE

State Transition Graph Reaction

- Reaction of a STG G
  1. If there is no current reactive cell in G then set current-RC to the initial RC: G.current = G.ini;
  2. Compute the reaction of the current :
     r = react(G.current)
  3. If r =DONE then G.current = G.current.nextState; goto 2.
  4. return r /* here r cannot be DONE */

- Comments:
  - When entering a macrostate the current RC of each STG is undefined. If the STG is already active, the current RC is the (unique) currently active RC.
  - Reactions of all STGs from a macrostate are computed in parallel (fork).

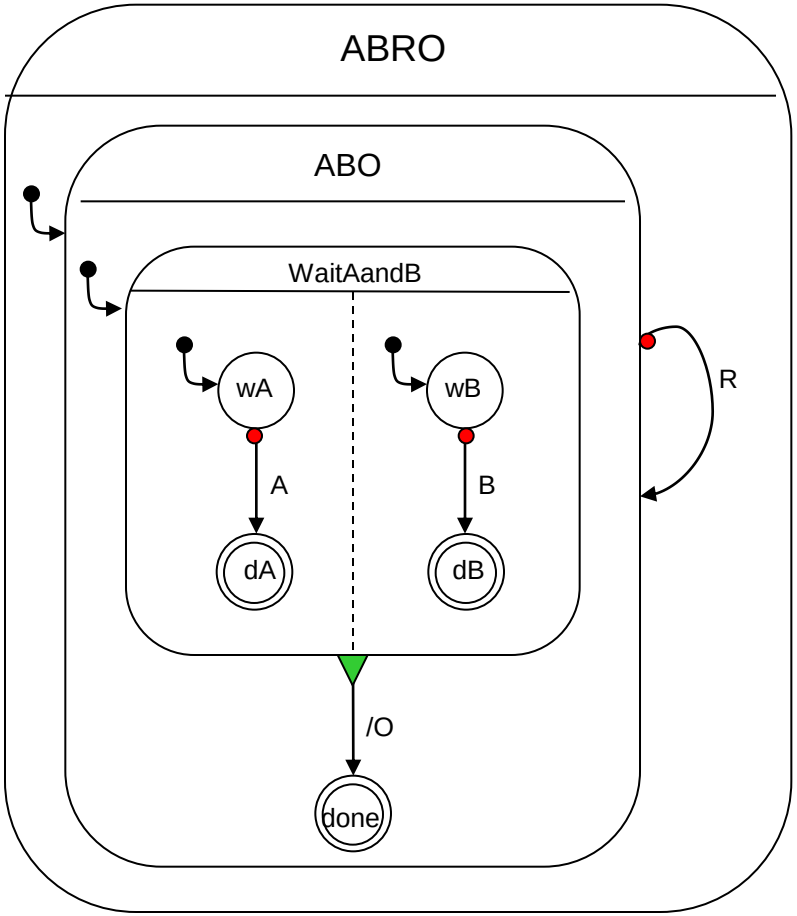
Reactive Cell Reaction

- Reaction of a reactive cell RC:
  1. if (!firstInstant) Strong abort test:
     - If a strong abort transition is enabled then take this transition and return its termination code
  2. Execute the body of the reactive cell
     - If it is a macrostate M, then recursive call: B = react(M)
     - If it is a simple state S, then terminal call: B = react(S)
  3. if (!firstInstant) Weak abort test:
     - If a weak abort transition is enabled then take this transition and return its termination code
  4. Normal termination test:
     - If (B == DEAD) then take normal termination transition and return its termination code
  5. End of reaction:
     - Set RC.status=ACTIVE
     - return PAUSE

Reactive Cell Reaction

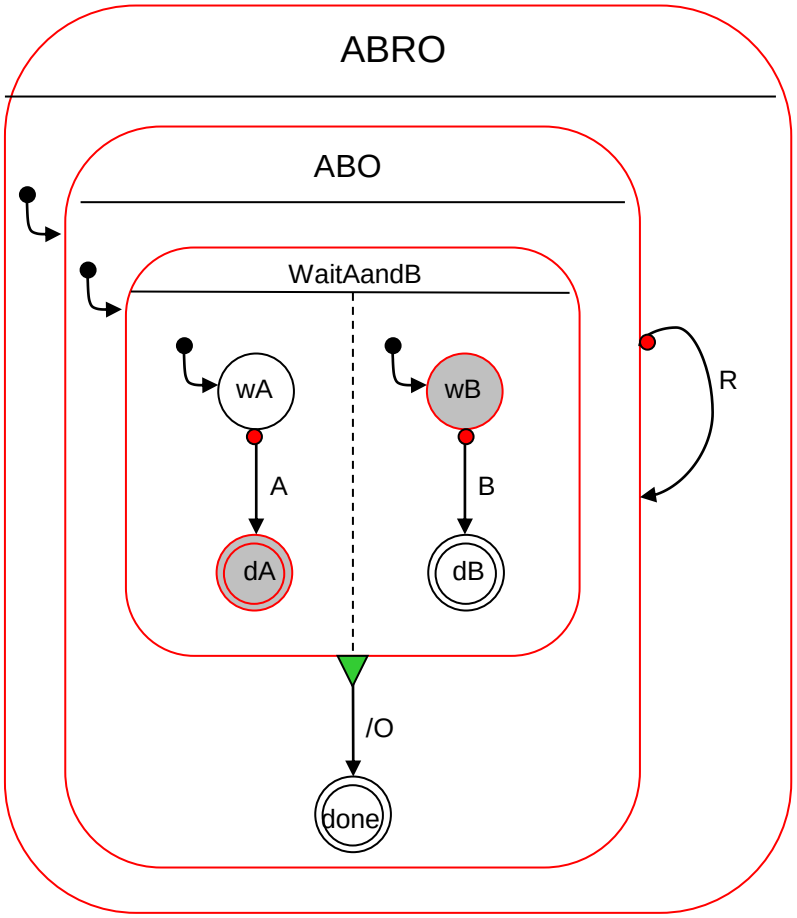
- The triggers are not tested at the first instant when the reactive cell is activated (strong/weak abort test).
- If the presence status of a triggering signal is unknown, the execution is suspended till another concurrent execution thread will fix the status of the tested signal.
- Taking a transition t (strong/weak abort, or normal termination) means:
  - Recursively "kill" the body of the reactive cell RC:
    - set k.status=IDLE for all transitively contained reactive cells k
    - reset G.current for all transitively contained STGs G
  - Execute the effect associated with t and set RC.nextRC = t.target;
  - Set RC.status = IDLE;
  - return DONE;

Example



- Start with stable configuration {ABRO,ABO,WaitAandB,dA,wB}
- Input: R^+,B^+

Example



- Start with stable configuration {ABRO,ABO,WaitAandB,dA,wB}
- Input: R^+,B^+
- Reactive cells with status ACTIVE: RC_{ABRO}, RC_{ABO}, $RC_{WaitAandB}$, RC_{dA}, RC_{wB}
- current-RC setting of state transition graphs:
  $ABRO.G1.current=RC_{ABO}$
  $ABO.G1.current=RC_{WaitAandB}$
  $WaitAandB.G1.current=RC_{dA}$
  $WaitAandB.G2.current=RC_{wB}$

Computing the SSM Reaction

- Compute reaction of SSM
  - Read input signals: R+,B+,A-
  - Set output signals to $\perp$
  - call react(Top)=react(ABRO) **1**
- **1** react(ABRO): macrostate reaction
  - Set local signals to $\perp$
  - Only one STG contained => Execute c(ABRO.G_1)=react(ABRO.G_1) **2**
- **2** react(ABRO.G_1): state transition graph reaction
  - ABRO.G1.current is defined and set to RC_{ABO}
  - Execute r = react(RC_{ABO}) **3**
- **3** react(RC_{ABO}): reactive cell reaction
  - RC_{ABO}.S==ACTIVE => not 1st instant
  - Execute SA-Check (Strong Abort Check): transition t=<Sa,R,,RC_{ABO}> is enabled => take transition t **4**

Computing the SSM Reaction

- **4** Execute transition t=<sA,R,,RC_{ABO}>
  - Recursively kill body of t
    - Set q.status to IDLE for all recursively contained reactive cells g not already set to IDLE: $RC_{WaitAandB}$, RC_{dA}, RC_{wB}
    - Reset G.current for all recursively contained STGs G: ABO.G_1, WaitAandB.G_1, WaitAandB.G_2
  - Execute the effect of the transition (none)
  - RC_{ABO}.nextRC = RC_{ABO}
  - RC_{ABO}.status = IDLE
  - return DONE
- back to **4** (react(RC_{ABO}) continued):
  - return termination code of t = DONE
- back to **3** (react(ABRO.G_1) continued):
  - r==DONE => G1.current= RC_{ABO}.nextRC= RC_{ABO}
  - r = react(RC_{ABO}) **5**

Computing the SSM Reaction

- **5** react(RC_{ABO}): reactive cell reaction
  - RC_{ABO}.S==IDLE => 1st instant is true
  - no SA-Check
  - Execute reaction of macrostate ABO: B = react(ABO) **6**
- **6** react(ABO): macrostate reaction
  - Set all local signals to ⊥
  - Only one STG contained => Execute c($ABO.G_1$)=react($ABO.G_1$) **7**
- **7** react($ABO.G_1$): state transition graph reaction
  - ABO.G1.current is undefined, so set ABO.G1.current=ABO.G1.ini=$RC_{WaitAandB}$
  - Execute r = react($RC_{WaitAandB}$) **8**
- **8** react($RC_{WaitAandB}$): reactive cell reaction
  - $RC_{WaitAandB}$.S==IDLE => 1st instant is true
  - no SA-Check
  - Execute reaction of macrostate WaitAandB: B = react(WaitAandB) **9**

Computing the SSM Reaction

- **9** react(WaitAandB): macrostate reaction
  - Set all local signals to ⊥
  - Two STGs contained => Execute in parallel:
    c(WaitAandB.G_1)=react(WaitAandB.G_1) **10**
    || c(WaitAandB.G_1)=react(WaitAandB.G_1) **11**
- **10** react(WaitAandB.G_1): state transition graph reaction
  - WaitAandB.G1.current is undefined, so set
    WaitAandB.G1.current=WaitAandB.G1.ini=RC_{wA}
  - Execute r = react(RC_{wA}) **12**
- **12** react(RC_{wA}): reactive cell reaction
  - RC_{wA}.S==IDLE => 1st instant is true
  - no SA-Check
  - Execute reaction of simple state wA: B = react(wA) **13**
- **13** react(wA): simple state reaction
  - not final, no effect
  - return PAUSE

Computing the SSM Reaction

- back to **13** (react(RC_{wA}) continued):
  - B=PAUSE
  - 1st instant is true => no WA-check
  - B!=DEAD => no NT-check
  - RC_{wA}.status=ACTIVE
  - return PAUSE
- back to **12** (react(WaitAandB.G_1) continued):
  - r==PAUSE != DONE, so return PAUSE
- back to **10** (react(WaitAandB) continued):
  - Thread 1 yields c(WaitAandB.G_1)=PAUSE
  - Consider execution of Thread 2.
- **11** react(WaitAandB.G_2): state transition graph reaction
  - WaitAandB.G_2.current is undefined, so set WaitAandB.G_2.current=WaitAandB.G_2.ini=RC_{wB}
  - Execute r = react(RC_{wB}) **14**

Computing the SSM Reaction

- **14** react(RC_{wB}): reactive cell reaction
  - RC_{wB}.S==IDLE => 1st instant is true
  - no SA-Check
  - Execute reaction of simple state wB: B = react(wB) **15**
- **15** react(wB): simple state reaction
  - not final
  - no effect
  - return PAUSE
- back to **15** (react(RC_{wB}) continued):
  - B=PAUSE
  - 1st instant is true => no WA-check
  - B!=DEAD => no NT-check
  - RC_{wB}.status=ACTIVE
  - return PAUSE

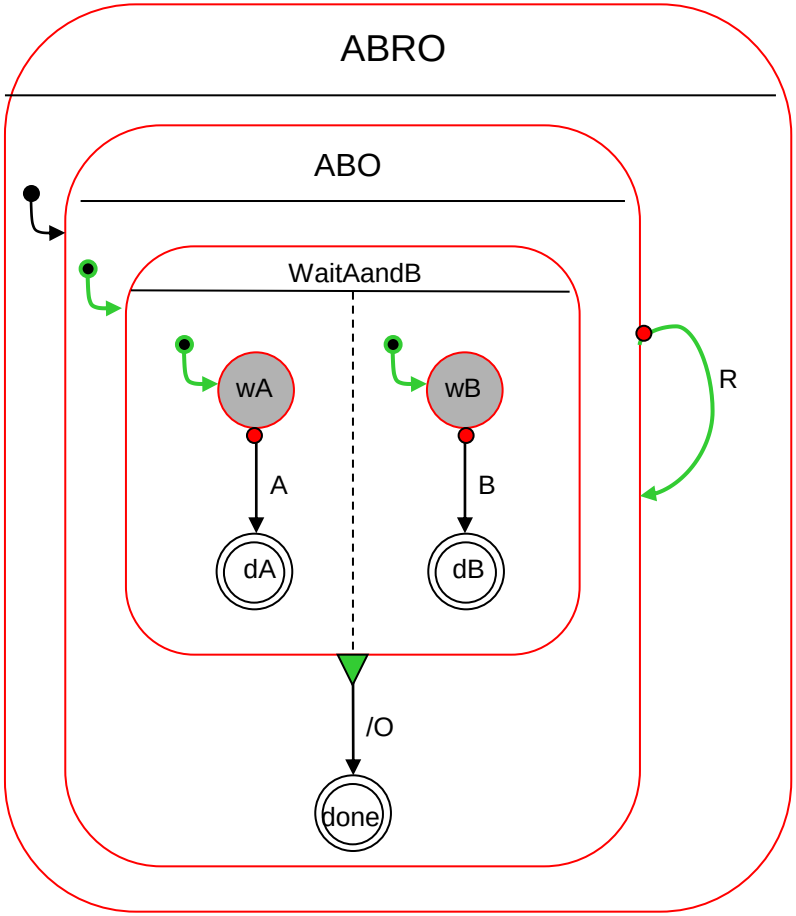
Computing the SSM Reaction

- back to **14** (react(WaitAandB.G_2) continued):
  - r==PAUSE != DONE, so return PAUSE
- back to **11** (react(WaitAandB) continued):
  - Thread 2 yields c(WaitAandB.G_2)=PAUSE
  - C = max(PAUSE,PAUSE)=PAUSE
  - return PAUSE
- back to **9** (react($RC_{WaitAandB}$) continued):
  - B=PAUSE
  - 1st instant is true => no WA-check
  - B!=DEAD => no NT-check
  - $RC_{WaitAandB}$.status=ACTIVE
  - return PAUSE
- back to **8** (react(ABO.G_1) continued):
  - r==PAUSE != DONE, so return PAUSE

Computing the SSM Reaction

- back to **7** (react(ABO) continued):
  - C = PAUSE
  - return PAUSE
- back to **6** (react(RC_{ABO}) continued):
  - B=PAUSE
  - 1st instant is true => no WA-check
  - B!=DEAD => no NT-check
  - RC_{ABO}.status=ACTIVE
  - return PAUSE
- back to **5** (react(ABRO.G_1) continued):
  - r==PAUSE != DONE, so return PAUSE
- back to **2** (react(ABRO) continued):
  - C = PAUSE
  - return PAUSE

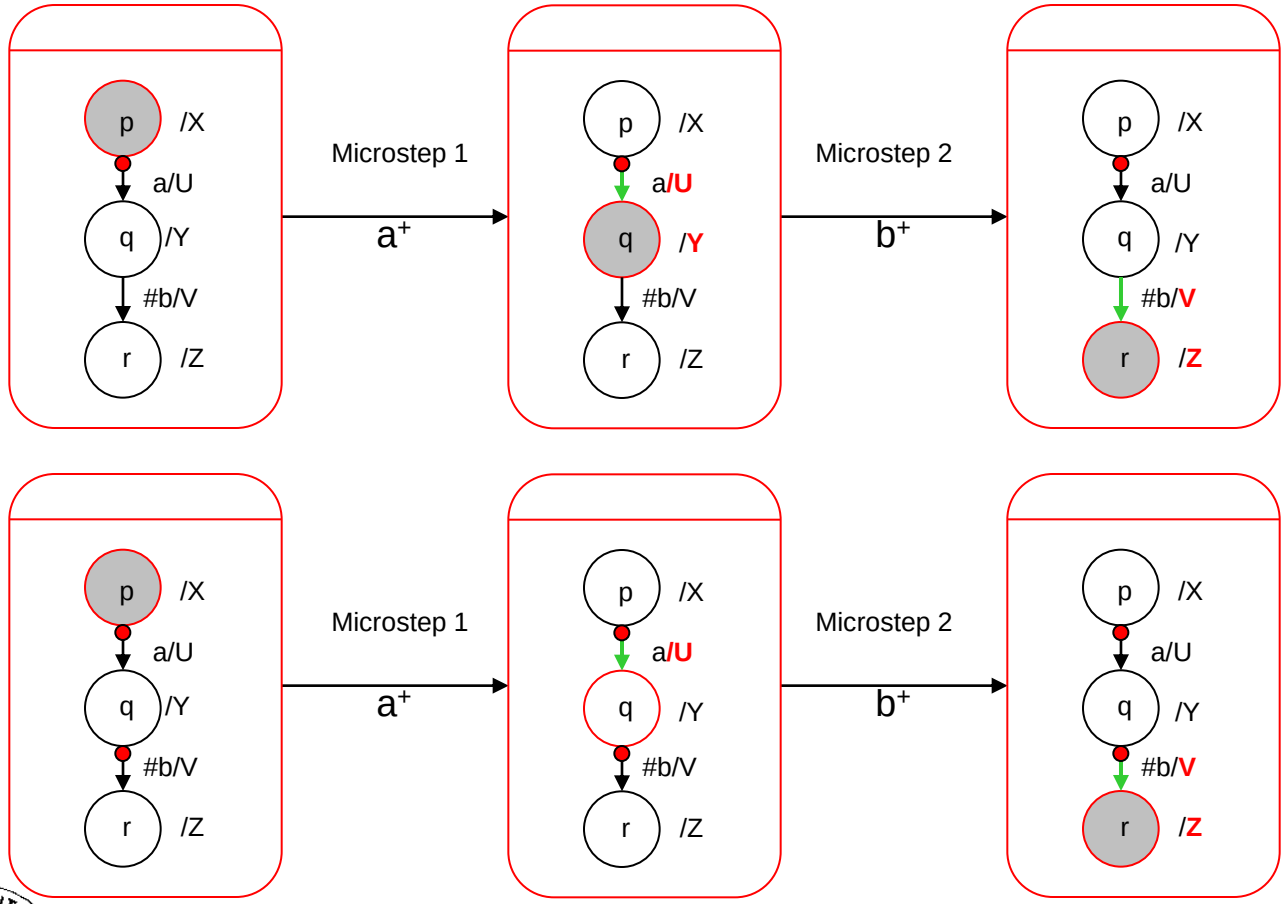
Computing the SSM Reaction



- New stable configuration {ABRO,ABO,WaitAandB,wA,wB}
- Reactive cells with status ACTIVE: RC_{ABRO}, RC_{ABO}, $RC_{WaitAandB}$, RC_{wA}, RC_{wB}
- current-RC setting of state transition graphs:
  $ABRO.G1.current=RC_{ABO}$
  $ABO.G1.current=RC_{WaitAandB}$
  $WaitAandB.G1.current=RC_{wA}$
  $WaitAandB.G2.current=RC_{wB}$

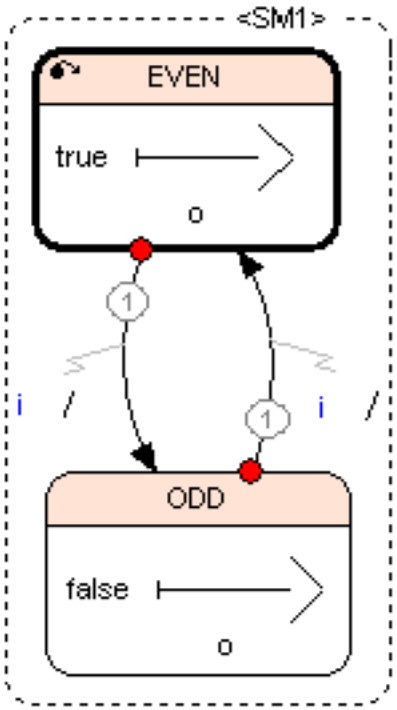
Immediate Weak/Strong Aborts

- What happens for a^+b^+ ?



Textual Representation

```
automaton SM1
     initial state EVEN
     unless if i restart ODD;
     let
     o = true;
     tel
     state ODD
     unless if i restart EVEN;
     let
     o = false;
     tel
returns o ;
```



Textual Representation

```
automaton SM1
     initial state A
     let
     automaton SM2
     initial state B
     until if 2 times 'S1 restart C;
     final state C
     returns .. ;
     automaton SM3
     initial state D
     until if 2 times c restart E;
     final state E
     returns .. ;
     tel
     until
     synchro do let emit 'S2; tel restart F;
     state B
returns .. ;
```