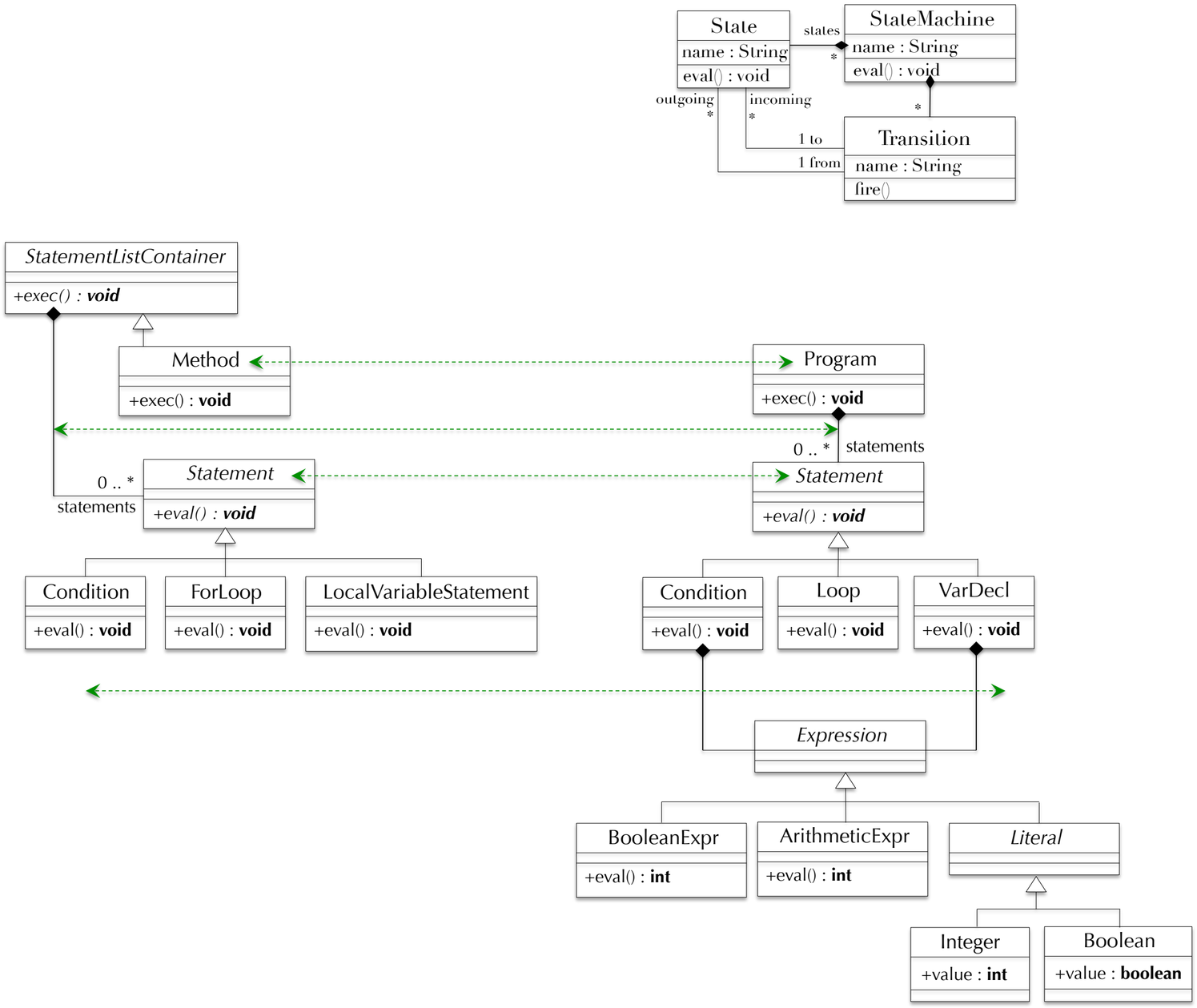
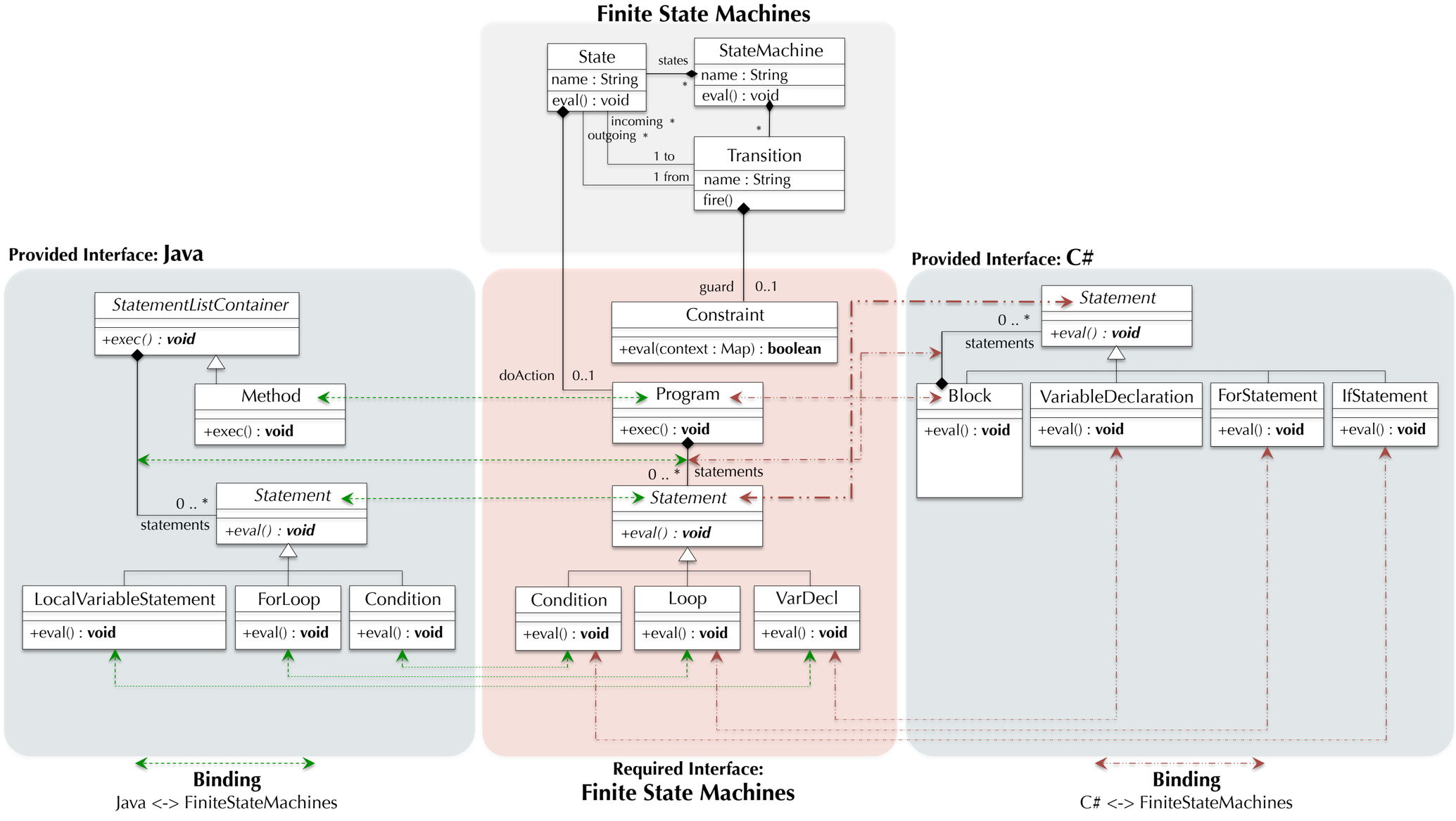
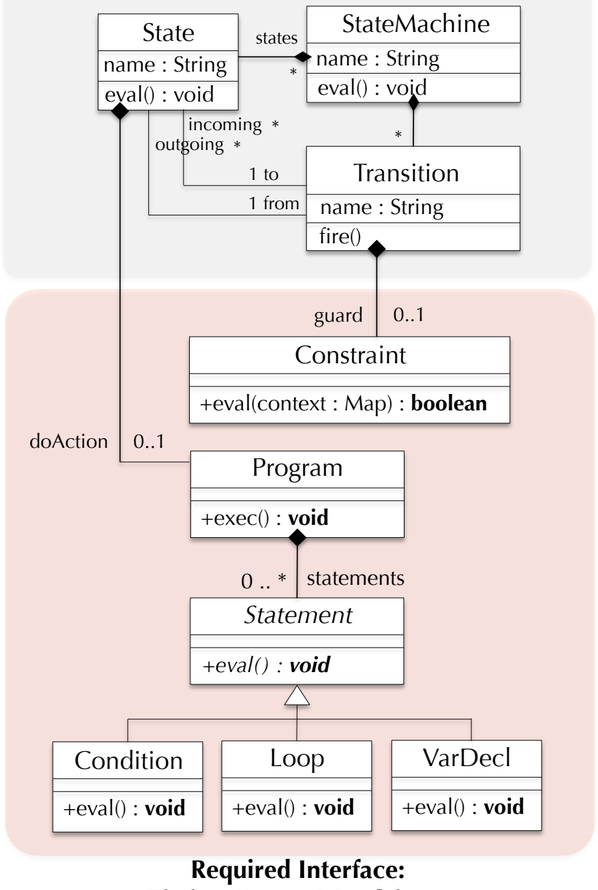

State
name : String
eval() : void
states
*
StateMachine
name : String
eval() : void
*
outgoing
*
incoming
*
1 to
1 from
Transition
name : String
fire()
StatementListContainer
+exec() : void
Method
+exec() : void
Program
+exec() : void
0 .. *
statements
0 .. *
statements
Statement
+eval() : void
Statement
+eval() : void
Condition
+eval() : void
ForLoop
+eval() : void
LocalVariableStatement
+eval() : void
Condition
+eval() : void
Loop
+eval() : void
VarDecl
+eval() : void
Expression
BooleanExpr
+eval() : int
ArithmeticExpr
+eval() : int
Literal
Integer
+value : int
Boolean
+value : boolean

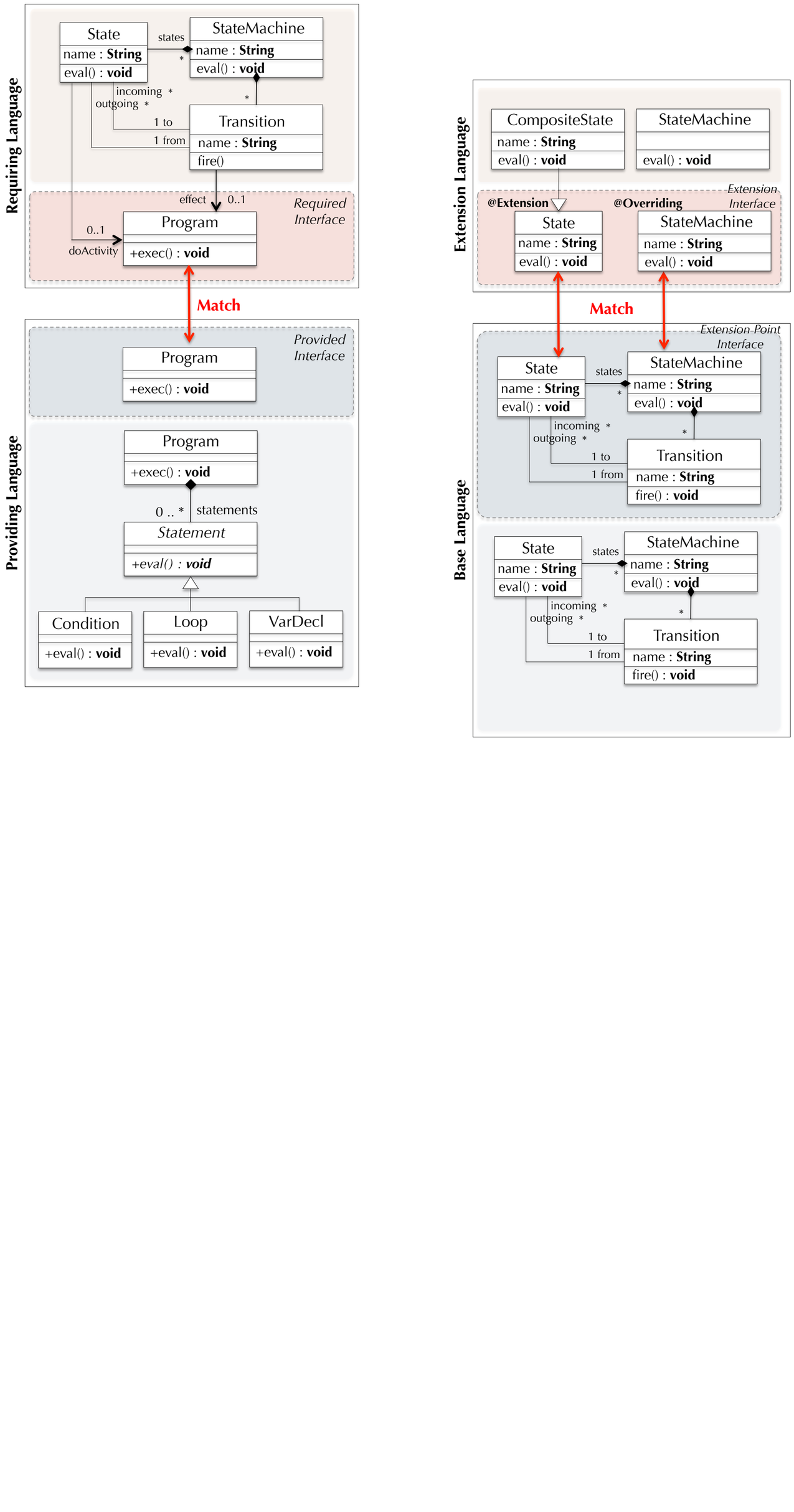

Finite State Machines
State
name : String
eval() : void
states
*
StateMachine
name : String
eval() : void
*
incoming *
outgoing *
1 to
1 from
Transition
name : String
fire()
guard
0..1
Provided Interface: Java
StatementListContainer
+exec() : void
Method
+exec() : void
0 .. *
statements
Statement
+eval() : void
LocalVariableStatement
+eval() : void
ForLoop
+eval() : void
Condition
+eval() : void
Constraint
+eval(context : Map) : boolean
doAction
0..1
Program
+exec() : void
0 .. *
statements
Statement
+eval() : void
Condition
+eval() : void
Loop
+eval() : void
VarDecl
+eval() : void
Provided Interface: C#
Statement
+eval() : void
0 .. *
statements
Block
+eval() : void
VariableDeclaration
+eval() : void
ForStatement
+eval() : void
IfStatement
+eval() : void
Binding
Java <-> FiniteStateMachines
Required Interface:
Finite State Machines
Binding
C# <-> FiniteStateMachines

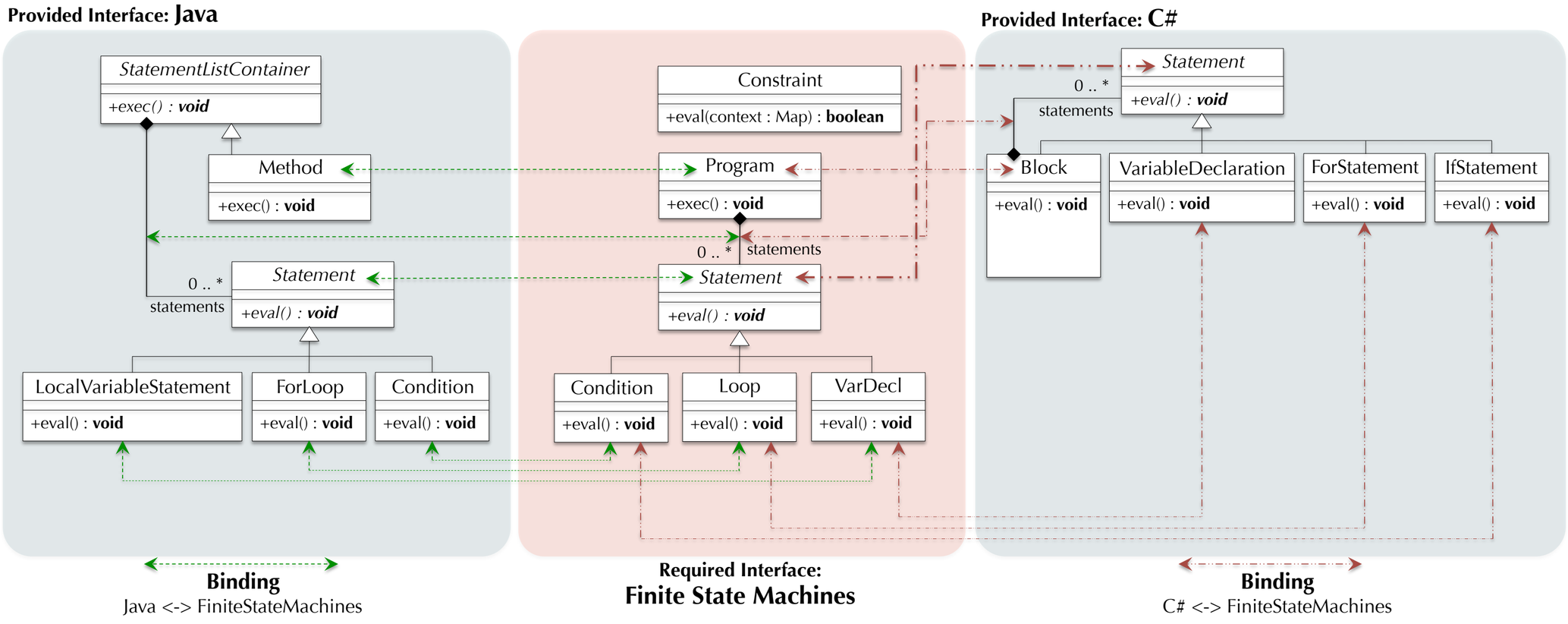
Finite State Machines

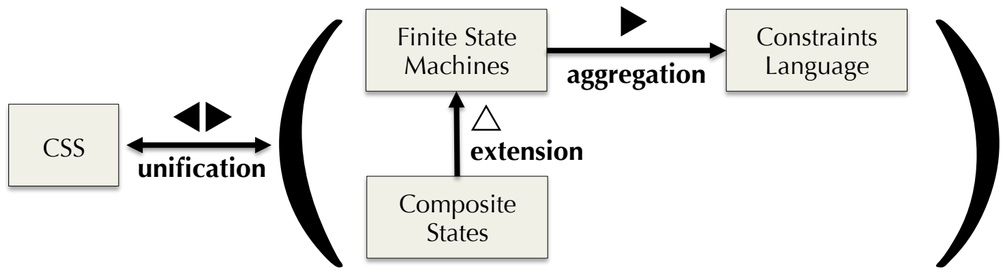


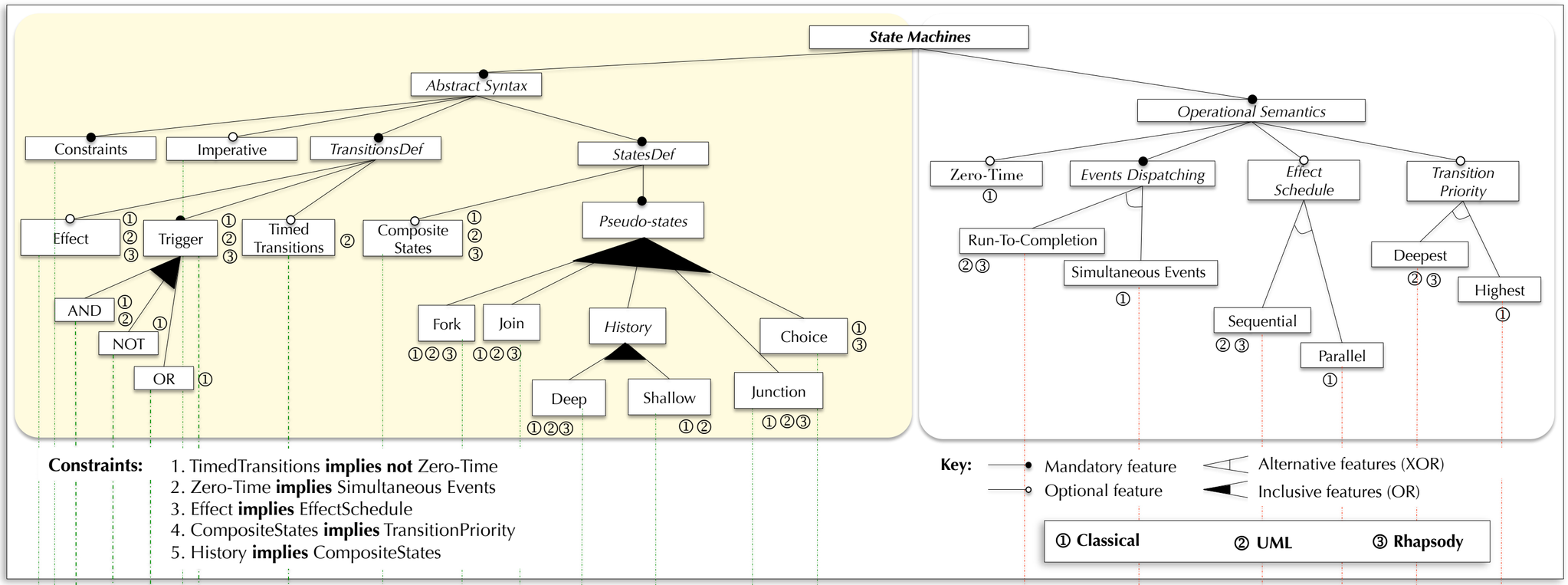
Required Interface:

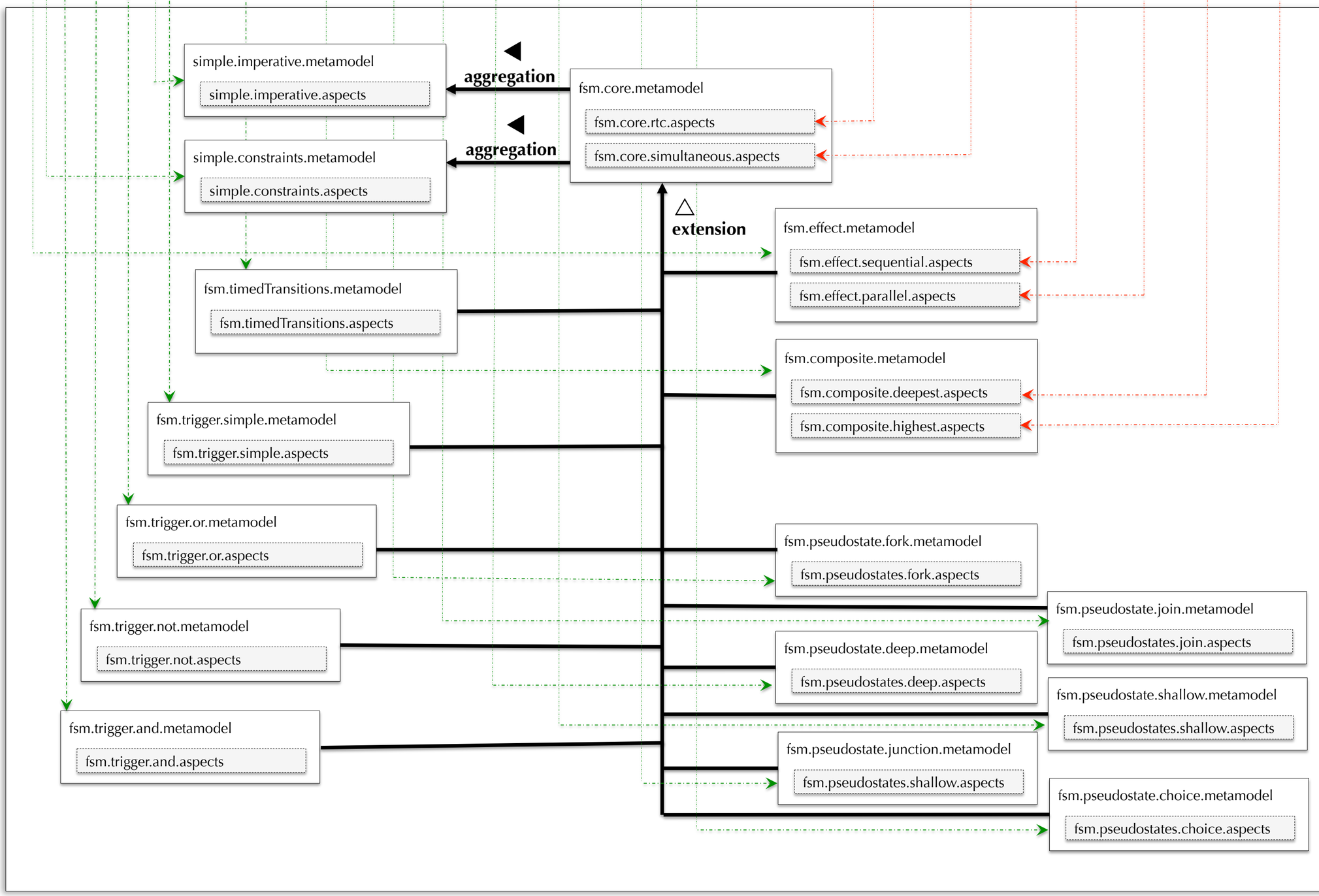
Finite State Machines

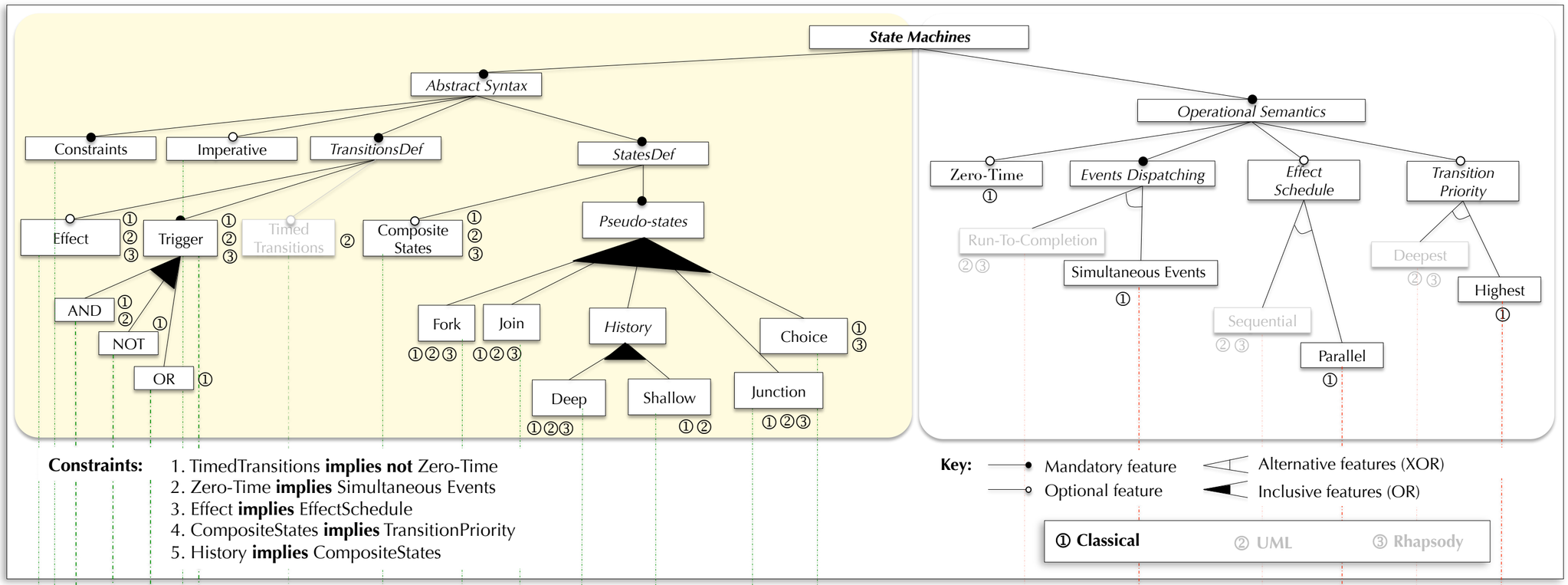

Requiring Language
State
name : String
eval() : void
states
*
StateMachine
name : String
eval() : void
*
incoming *
outgoing *
1 to
1 from
Transition
name : String
fire()
effect
0..1
Required Interface
0..1
doActivity
Program
+exec() : void
Match
Provided Interface
Program
+exec() : void
Providing Language
Program
+exec() : void
0 .. *
statements
Statement
+eval() : void
Condition
+eval() : void
Loop
+eval() : void
VarDecl
+eval() : void
Extension Language
CompositeState
name : String
eval() : void
StateMachine
eval() : void
@Extension
@Overriding
Extension Interface
State
name : String
eval() : void
StateMachine
name : String
eval() : void
Match
Base Language
Extension Point Interface
State
name : String
eval() : void
states
*
StateMachine
name : String
eval() : void
*
incoming *
outgoing *
1 to
1 from
Transition
name : String
fire() : void
State
name : String
eval() : void
states
*
StateMachine
name : String
eval() : void
*
incoming *
outgoing *
1 to
1 from
Transition
name : String
fire() : void

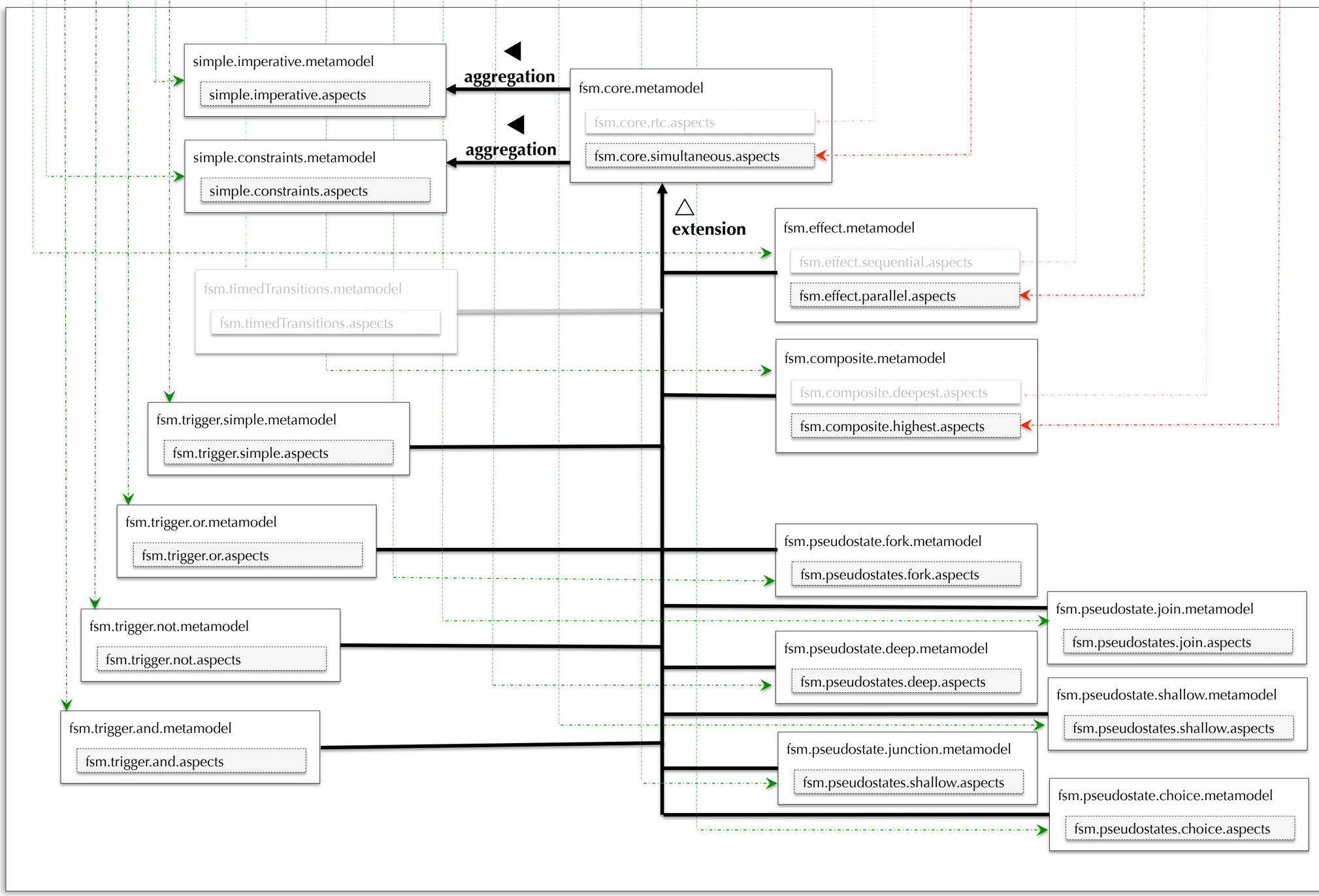

Provided Interface: Java
StatementListContainer
+exec() : void
Method
+exec() : void
0 .. *
statements
Statement
+eval() : void
LocalVariableStatement
+eval() : void
ForLoop
+eval() : void
Condition
+eval() : void
Constraint
+eval(context : Map) : boolean
Program
+exec() : void
0 .. *
statements
Statement
+eval() : void
Condition
+eval() : void
Loop
+eval() : void
VarDecl
+eval() : void
Provided Interface: C#
Statement
+eval() : void
0 .. *
statements
Block
+eval() : void
VariableDeclaration
+eval() : void
ForStatement
+eval() : void
IfStatement
+eval() : void
Binding
Java <-> FiniteStateMachines
Required Interface:
Finite State Machines
Binding
C# <-> FiniteStateMachines

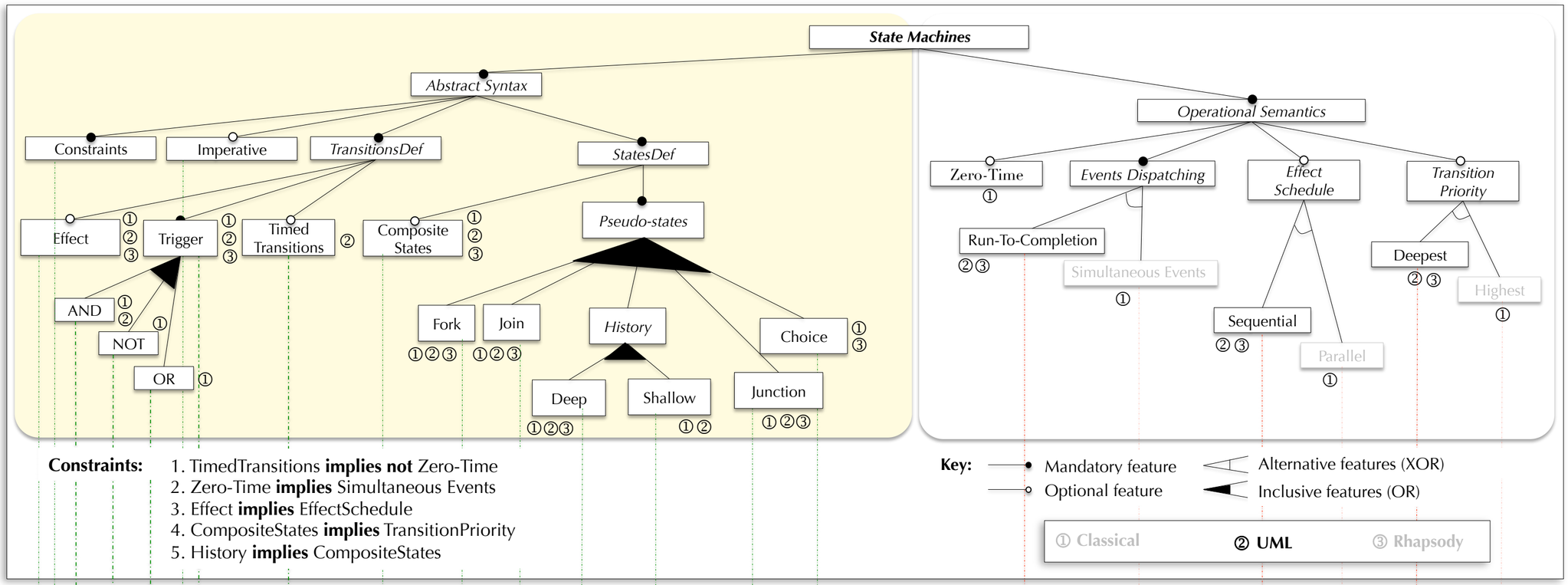

CSS
unification
Finite State Machines
aggregation
Constraints Language
extension
Composite States

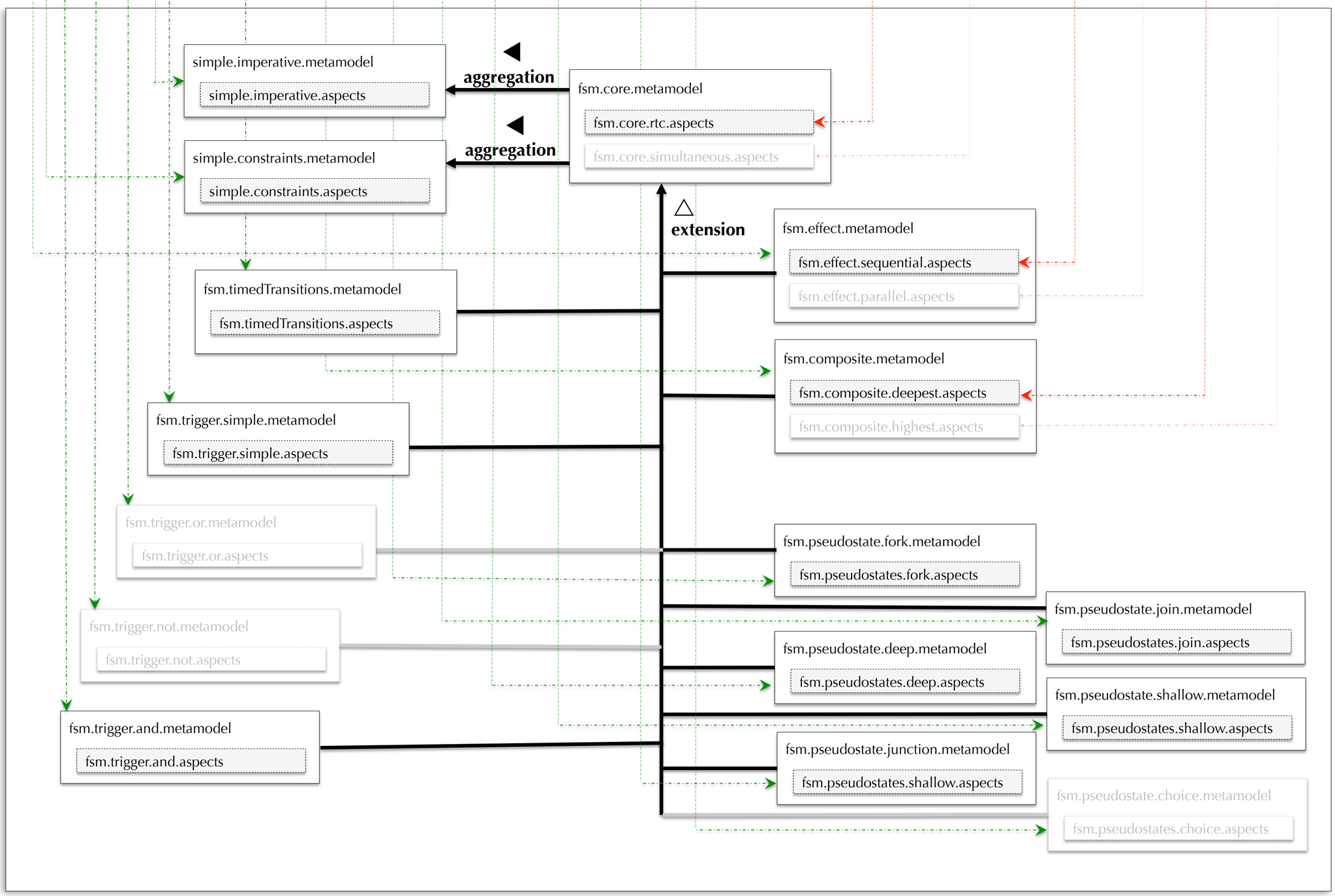

State Machines
Abstract Syntax
Constraints
Imperative
TransitionsDef
StatesDef
Effect
Trigger
Timed Transitions
Composite States
Pseudo-states
AND
NOT
OR
Fork
Join
History
Choice
Deep
Shallow
Junction
Operational Semantics
Zero-Time
Events Dispatching
Effect Schedule
Transition Priority
Run-To-Completion
Simultaneous Events
Sequential
Parallel
Deepest
Highest
Constraints:
1. TimedTransitions implies not Zero-Time
2. Zero-Time implies Simultaneous Events
3. Effect implies EffectSchedule
4. CompositeStates implies TransitionPriority
5. History implies CompositeStates
Key:
Mandatory feature
Optional feature
Alternative features (XOR)
Inclusive features (OR)
① Classical
② UML
③ Rhapsody

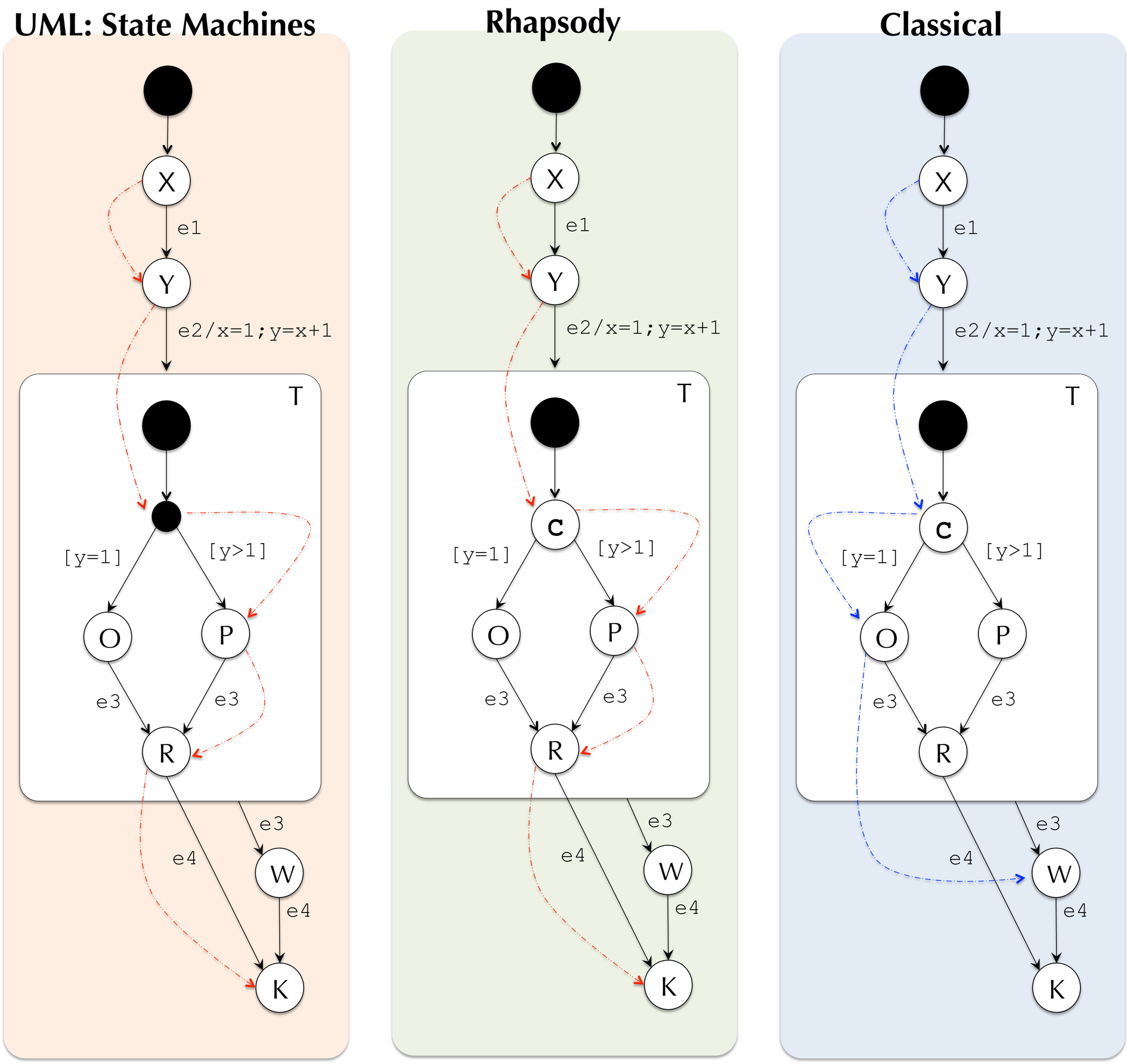

simple.imperative.metamodel
simple.imperative.aspects
aggregation
fsm.core.metamodel
fsm.core.rtc.aspects
fsm.core.simultaneous.aspects
simple.constraints.metamodel
simple.constraints.aspects
aggregation
extension
fsm.effect.metamodel
fsm.effect.sequential.aspects
fsm.effect.parallel.aspects
fsm.timedTransitions.metamodel
fsm.timedTransitions.aspects
fsm.composite.metamodel
fsm.composite.deepest.aspects
fsm.composite.highest.aspects
fsm.trigger.simple.metamodel
fsm.trigger.simple.aspects
fsm.trigger.or.metamodel
fsm.trigger.or.aspects
fsm.pseudostate.fork.metamodel
fsm.pseudostates.fork.aspects
fsm.pseudostate.join.metamodel
fsm.pseudostates.join.aspects
fsm.trigger.not.metamodel
fsm.trigger.not.aspects
fsm.pseudostate.deep.metamodel
fsm.pseudostates.deep.aspects
fsm.pseudostate.shallow.metamodel
fsm.pseudostates.shallow.aspects
fsm.trigger.and.metamodel
fsm.trigger.and.aspects
fsm.pseudostate.junction.metamodel
fsm.pseudostates.shallow.aspects
fsm.pseudostate.choice.metamodel
fsm.pseudostates.choice.aspects

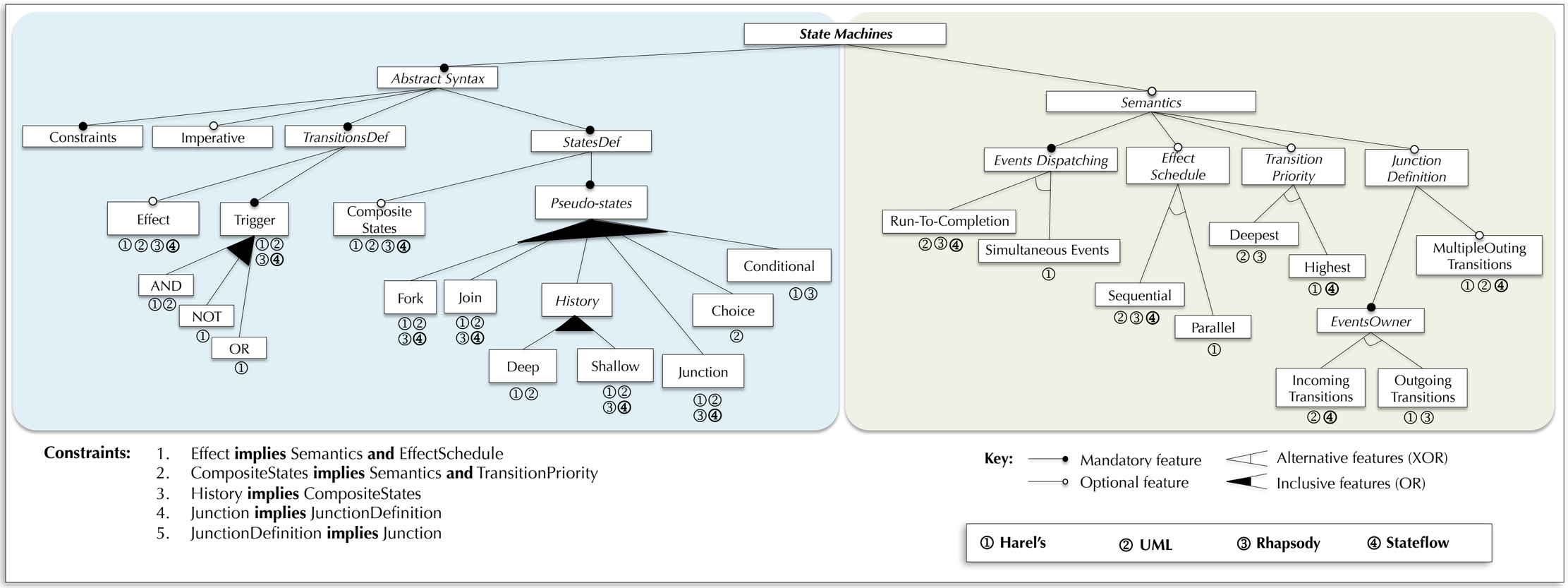

State Machines
Abstract Syntax
Constraints
Imperative
TransitionsDef
StatesDef
Effect
Trigger
Timed Transitions
Composite States
Pseudo-states
AND
NOT
OR
Fork
Join
History
Choice
Deep
Shallow
Junction
Operational Semantics
Zero-Time
Events Dispatching
Effect Schedule
Transition Priority
Run-To-Completion
Simultaneous Events
Sequential
Parallel
Deepest
Highest
Constraints:
1. TimedTransitions implies not Zero-Time
2. Zero-Time implies Simultaneous Events
3. Effect implies EffectSchedule
4. CompositeStates implies TransitionPriority
5. History implies CompositeStates
Key:
Mandatory feature
Optional feature
Alternative features (XOR)
Inclusive features (OR)
① Classical
② UML
③ Rhapsody

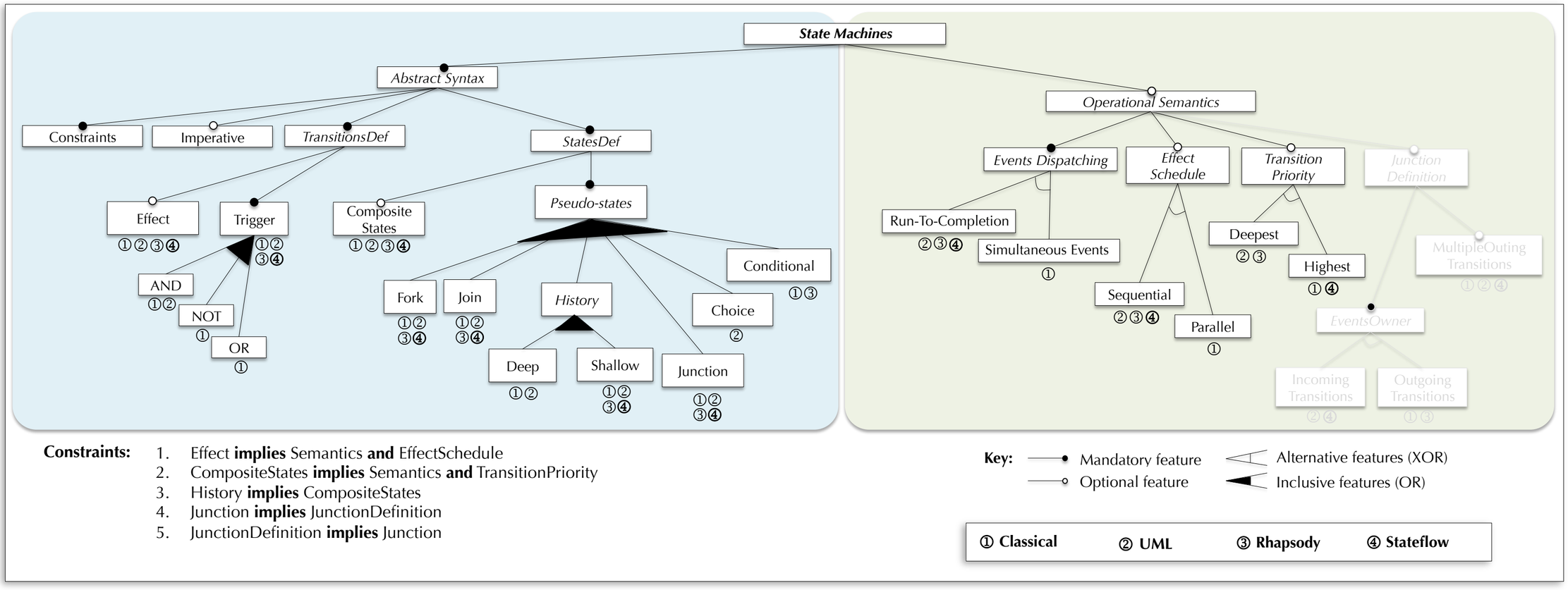

simple.imperative.metamodel
simple.imperative.aspects
aggregation
fsm.core.metamodel
fsm.core.rtc.aspects
fsm.core.simultaneous.aspects
simple.constraints.metamodel
simple.constraints.aspects
aggregation
extension
fsm.effect.metamodel
fsm.effect.sequential.aspects
fsm.effect.parallel.aspects
fsm.timedTransitions.metamodel
fsm.timedTransitions.aspects
fsm.composite.metamodel
fsm.composite.deepest.aspects
fsm.composite.highest.aspects
fsm.trigger.simple.metamodel
fsm.trigger.simple.aspects
fsm.trigger.or.metamodel
fsm.trigger.or.aspects
fsm.pseudostate.fork.metamodel
fsm.pseudostates.fork.aspects
fsm.pseudostate.join.metamodel
fsm.pseudostates.join.aspects
fsm.trigger.not.metamodel
fsm.trigger.not.aspects
fsm.pseudostate.deep.metamodel
fsm.pseudostates.deep.aspects
fsm.pseudostate.shallow.metamodel
fsm.pseudostates.shallow.aspects
fsm.trigger.and.metamodel
fsm.trigger.and.aspects
fsm.pseudostate.junction.metamodel
fsm.pseudostates.shallow.aspects
fsm.pseudostate.choice.metamodel
fsm.pseudostates.choice.aspects

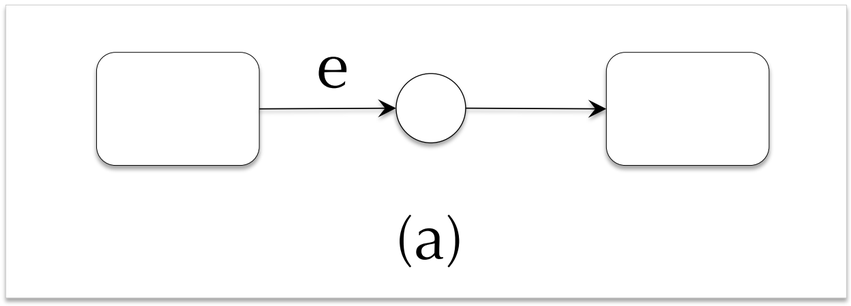

State Machines
Abstract Syntax
Constraints
Imperative
TransitionsDef
StatesDef
Effect
Trigger
Timed Transitions
Composite States
Pseudo-states
AND
NOT
OR
Fork
Join
History
Choice
Deep
Shallow
Junction
Operational Semantics
Zero-Time
Events Dispatching
Effect Schedule
Transition Priority
Run-To-Completion
Simultaneous Events
Sequential
Parallel
Deepest
Highest
Constraints:
1. TimedTransitions implies not Zero-Time
2. Zero-Time implies Simultaneous Events
3. Effect implies EffectSchedule
4. CompositeStates implies TransitionPriority
5. History implies CompositeStates
Key:
Mandatory feature
Optional feature
Alternative features (XOR)
Inclusive features (OR)
① Classical
② UML
③ Rhapsody


simple.imperative.metamodel
simple.imperative.aspects
aggregation
fsm.core.metamodel
fsm.core.rtc.aspects
fsm.core.simultaneous.aspects
simple.constraints.metamodel
simple.constraints.aspects
aggregation
extension
fsm.effect.metamodel
fsm.effect.sequential.aspects
fsm.effect.parallel.aspects
fsm.timedTransitions.metamodel
fsm.timedTransitions.aspects
fsm.composite.metamodel
fsm.composite.deepest.aspects
fsm.composite.highest.aspects
fsm.trigger.simple.metamodel
fsm.trigger.simple.aspects
fsm.trigger.or.metamodel
fsm.trigger.or.aspects
fsm.pseudostate.fork.metamodel
fsm.pseudostates.fork.aspects
fsm.pseudostate.join.metamodel
fsm.pseudostates.join.aspects
fsm.trigger.not.metamodel
fsm.trigger.not.aspects
fsm.pseudostate.deep.metamodel
fsm.pseudostates.deep.aspects
fsm.pseudostate.shallow.metamodel
fsm.pseudostates.shallow.aspects
fsm.trigger.and.metamodel
fsm.trigger.and.aspects
fsm.pseudostate.junction.metamodel
fsm.pseudostates.shallow.aspects
fsm.pseudostate.choice.metamodel
fsm.pseudostates.choice.aspects

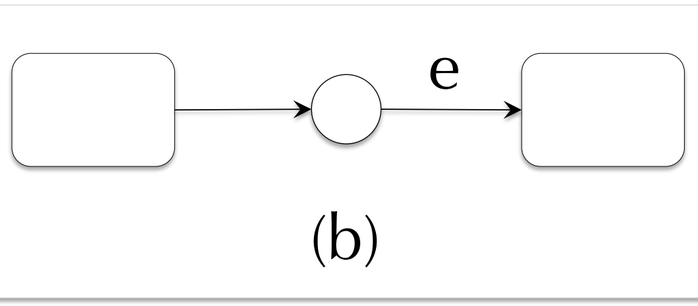

UML: State Machines
Rhapsody
Classical
X
e1
Y
e2/x=1;y=x+1
T
C
[y=1]
[y>1]
O
P
e3
R
e4
W
K


State Machines
Abstract Syntax
Constraints
Imperative
TransitionsDef
StatesDef
Effect ①②③④
Trigger ①② ③④
AND ①②
NOT ①
OR ①
Composite States ①②③④
Pseudo-states
Fork ①② ③④
Join ①② ③④
History
Deep ①②
Shallow ①② ③④
Junction ①② ③④
Choice ②
Conditional ①③
Semantics
Events Dispatching
Effect Schedule
Transition Priority
Junction Definition
Run-To-Completion ②③④
Simultaneous Events ①
Sequential ②③④
Parallel ①
Deepest ②③
Highest ①④
MultipleOuting Transitions ①②④
EventsOwner
Incoming Transitions ②④
Outgoing Transitions ①③
Constraints:
1. Effect implies Semantics and EffectSchedule
2. CompositeStates implies Semantics and TransitionPriority
3. History implies CompositeStates
4. Junction implies JunctionDefinition
5. JunctionDefinition implies Junction
Key:
Mandatory feature
Optional feature
Alternative features (XOR)
Inclusive features (OR)
① Harel's
② UML
③ Rhapsody
④ Stateflow

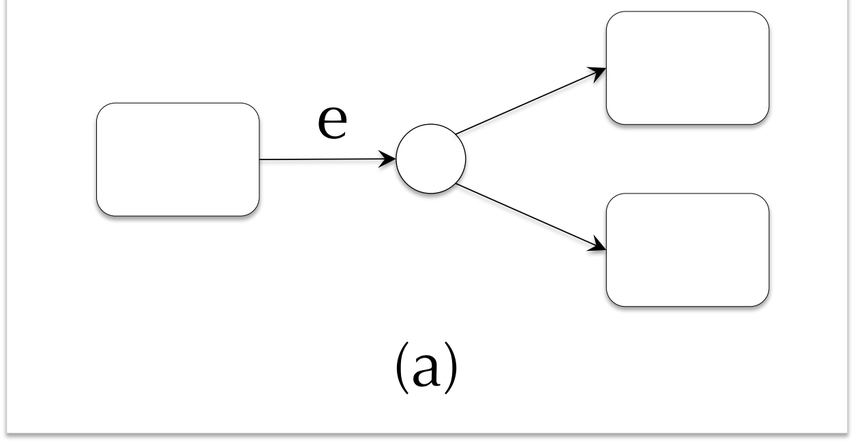

State Machines
Abstract Syntax
Constraints
Imperative
TransitionsDef
StatesDef
Effect ①②③④
Trigger ①② ③④
AND ①②
NOT ①
OR ①
Composite States ①②③④
Pseudo-states
Fork ①② ③④
Join ①② ③④
History
Deep ①②
Shallow ①② ③④
Junction ①② ③④
Choice ②
Conditional ①③
Operational Semantics
Events Dispatching
Effect Schedule
Transition Priority
Junction Definition
Run-To-Completion ②③④
Simultaneous Events ①
Sequential ②③④
Parallel ①
Deepest ②③
Highest ①④
MultipleOuting Transitions ①②④
EventsOwner
Incoming Transitions ②④
Outgoing Transitions ①③
Constraints:
1. Effect implies Semantics and EffectSchedule
2. CompositeStates implies Semantics and TransitionPriority
3. History implies CompositeStates
4. Junction implies JunctionDefinition
5. JunctionDefinition implies Junction
Key:
Mandatory feature
Optional feature
Alternative features (XOR)
Inclusive features (OR)
① Classical
② UML
③ Rhapsody
④ Stateflow



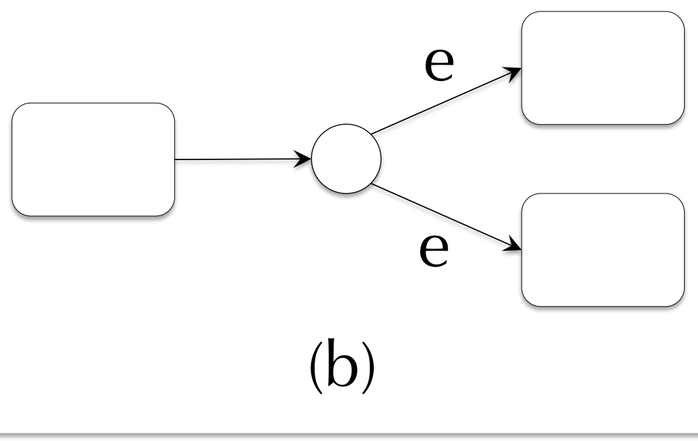
(a)



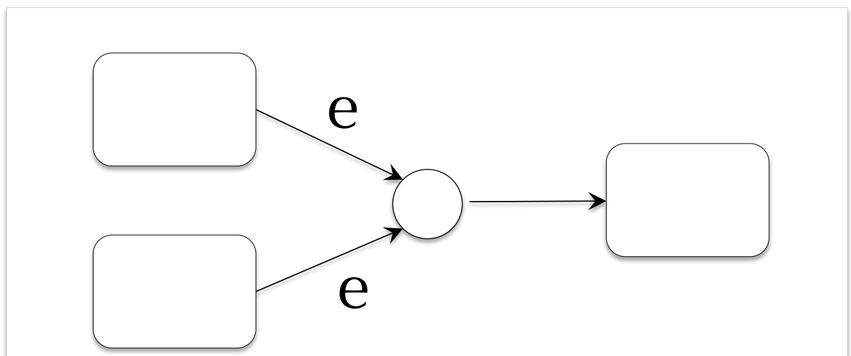
(b)



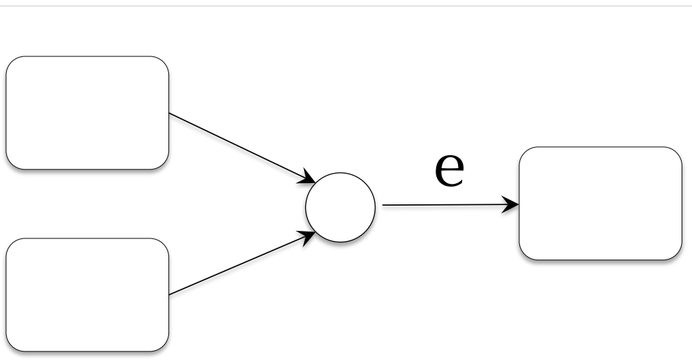
(a)



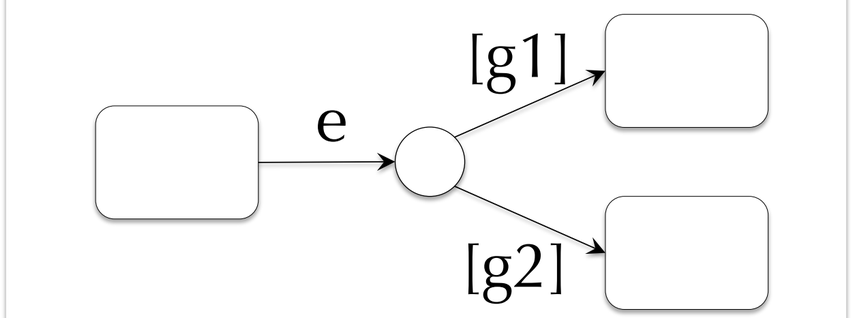
(b)



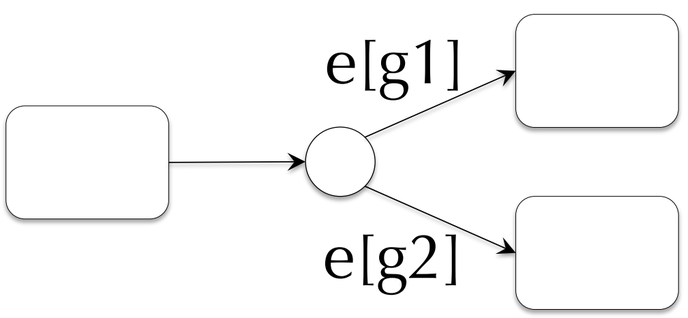
(c)



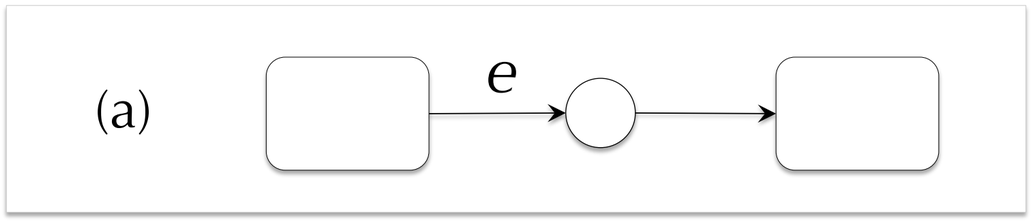
(d)

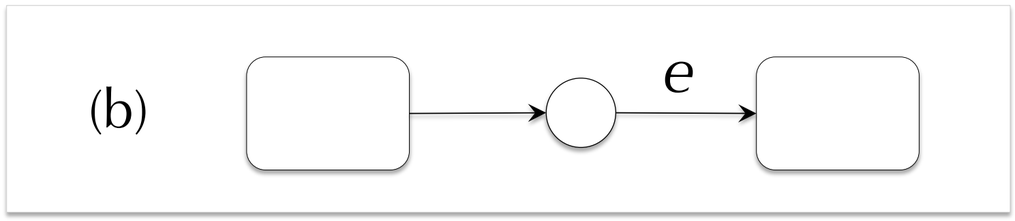


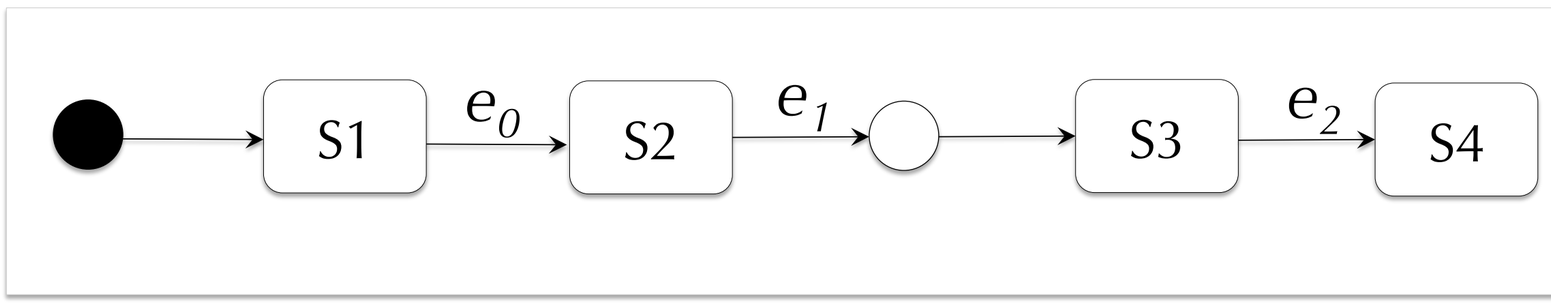
(e)



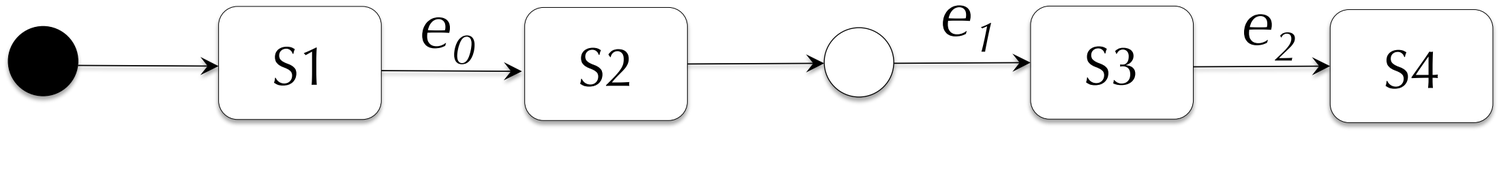
(f)



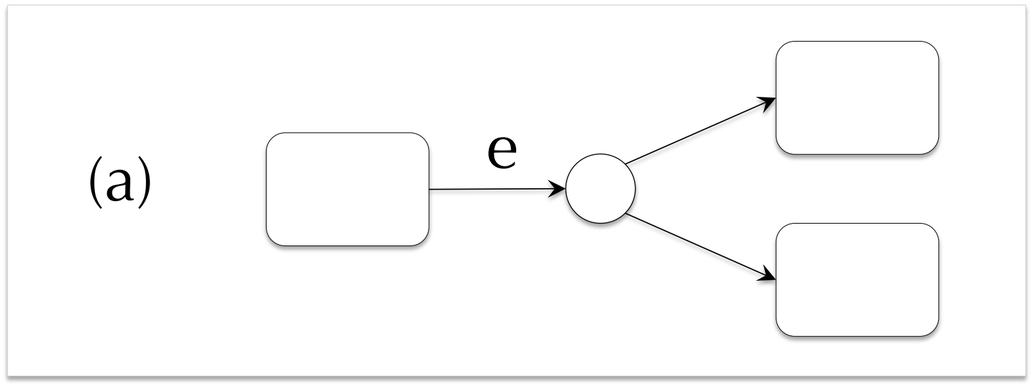


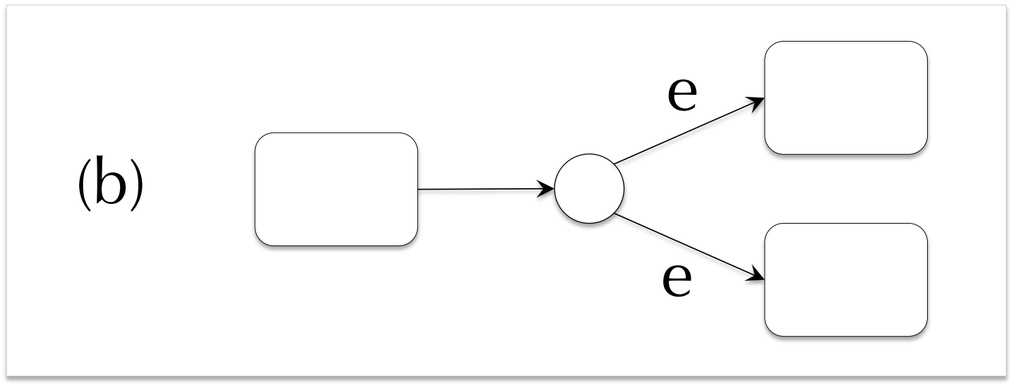


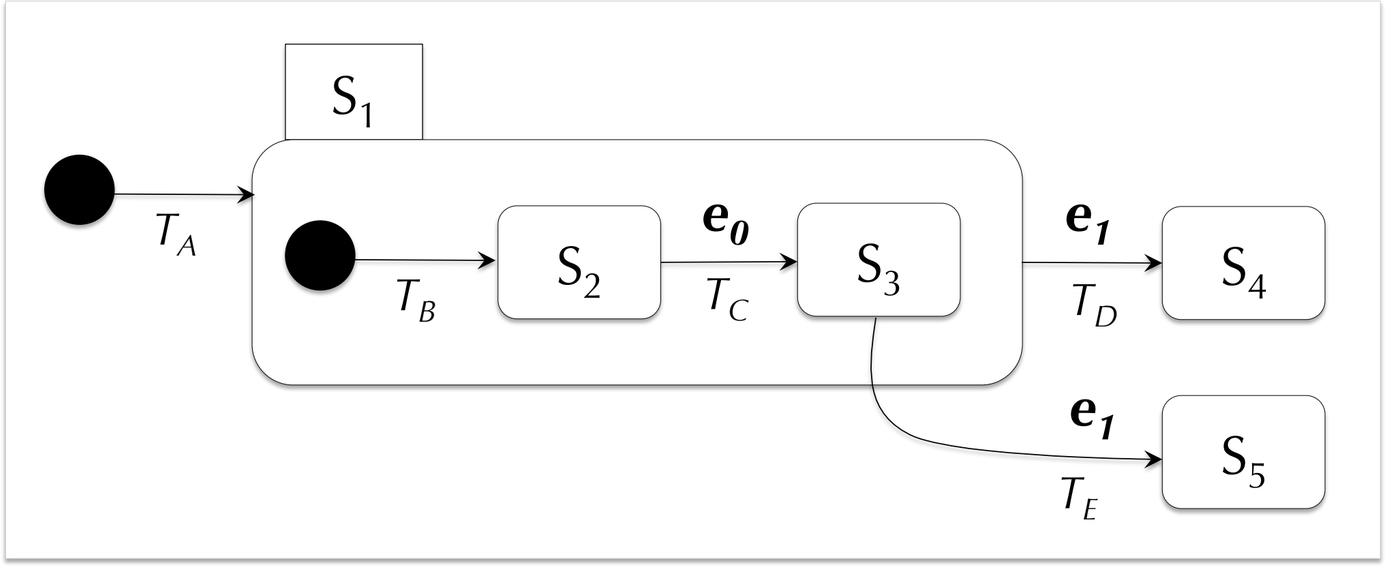
step_0 **[S1]**
step_1 {e_0} **[S2]**
step_2 {e_1} [j],**[S3]**
step_4 {e_3} **[S4]**

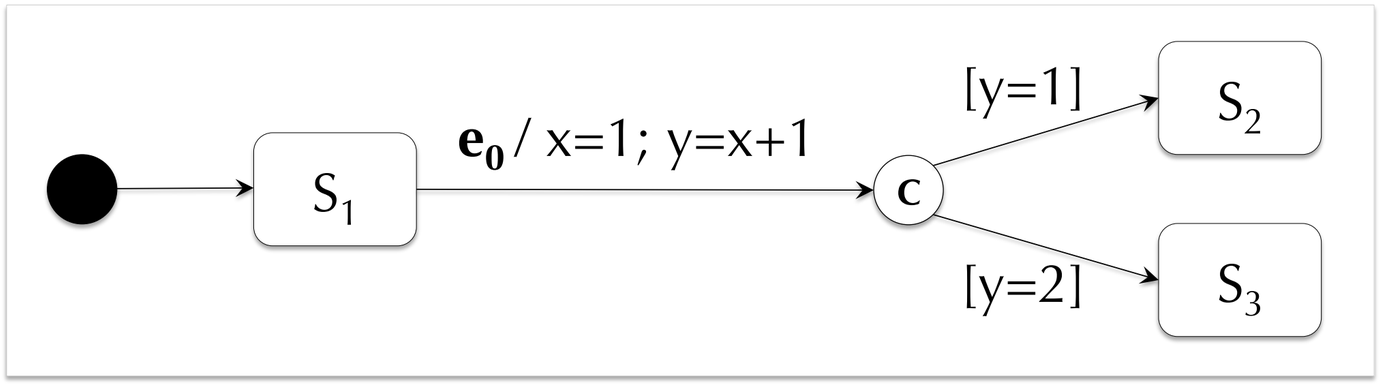


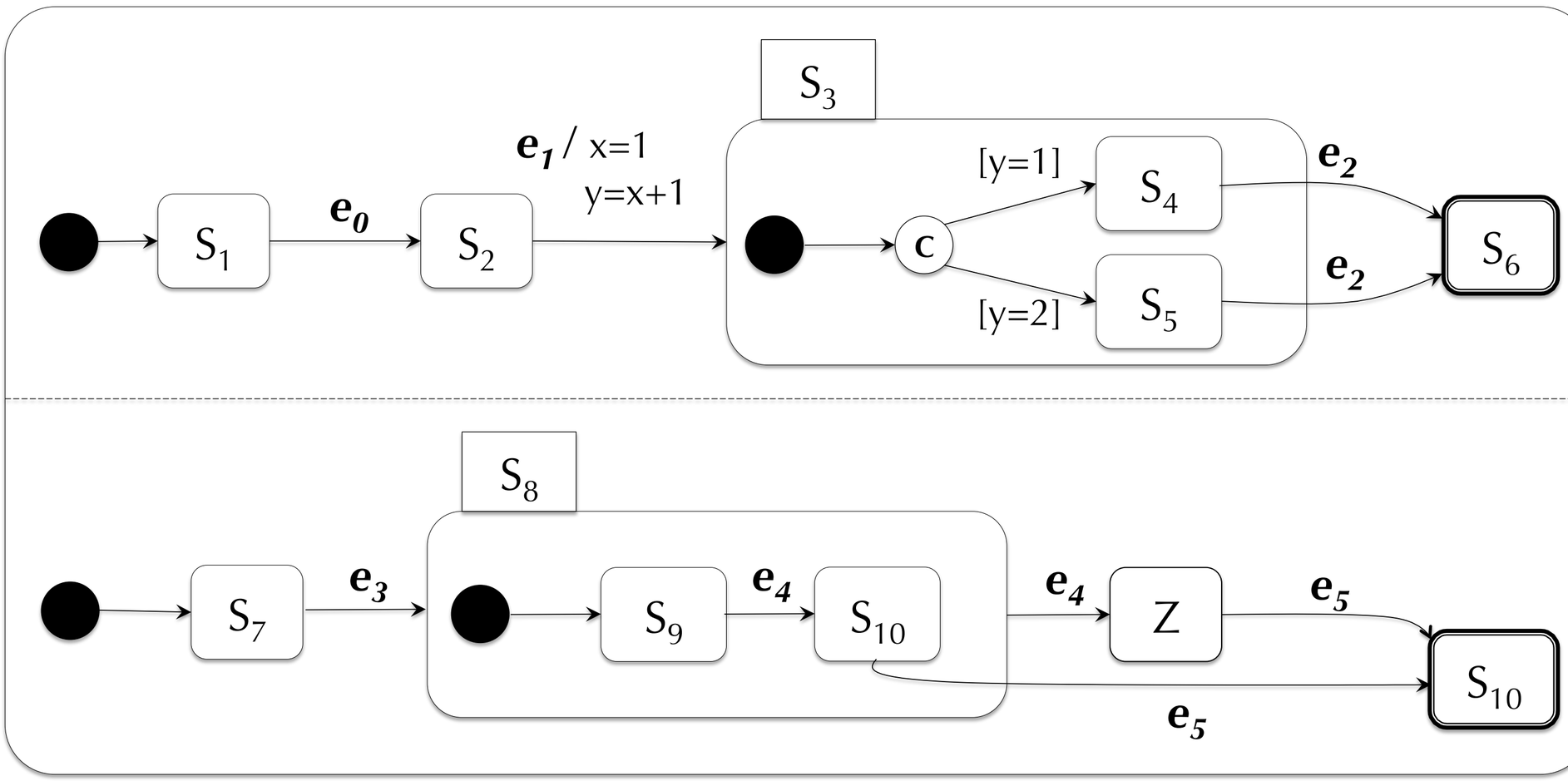
step_0 **[S1]**
step_1 {e_0} [S2],**[j]**
step_2 {e_1} **[S3]**
step_4 {e_3} **[S4]**











Execution traces

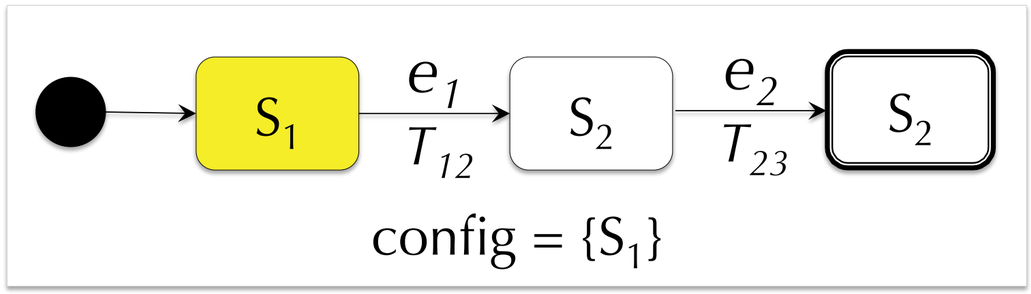
step_0	$[S_1\ S_2\ S_{10}]$
step_1 $\{e_0, e_4\}$	$[S_1\ S_3\ S_{11}]$
step_2 $\{e_1\}$	$[S_1\ S_5\ S_{11}]$
step_3 $\{e_2, e_5\}$	$[S_1\ S_8\ S_{12}]$
step_4 $\{e_3\}$	$[S_1\ S_9\ S_{12}]$

step_0	$[S_1\ S_2\ S_{10}]$
step_1 $\{\mathbf{e_0}, e_4\}$	$[S_1\ S_3\ S_{10}]$
step_1′ $\{e_0, \mathbf{e_4}\}$	$[S_1\ S_3\ S_{11}]$
step_2 $\{e_1\}$	$[S_1\ S_6\ S_{11}]$
step_3 $\{\mathbf{e_2}, e_5\}$	$[S_1\ S_7\ S_{11}]$
step_3′ $\{e_2, \mathbf{e_5}\}$	$[S_1\ S_7\ S_{12}]$
step_4 $\{e_3\}$	$[S_1\ S_9\ S_{12}]$

step_0	$[S_1\ S_2\ S_{10}]$
step_1 $\{\mathbf{e_0}, e_4\}$	$[S_1\ S_3\ S_{10}]$
step_1′ $\{e_0, \mathbf{e_4}\}$	$[S_1\ S_3\ S_{11}]$
step_2 $\{e_1\}$	$[S_1\ S_6\ S_{11}]$
step_3 $\{\mathbf{e_2}, e_5\}$	$[S_1\ S_7\ S_{11}]$
step_3′ $\{e_2, \mathbf{e_5}\}$	$[S_1\ S_7\ S_{12}]$
step_4 $\{e_3\}$	$[S_1\ S_9\ S_{12}]$

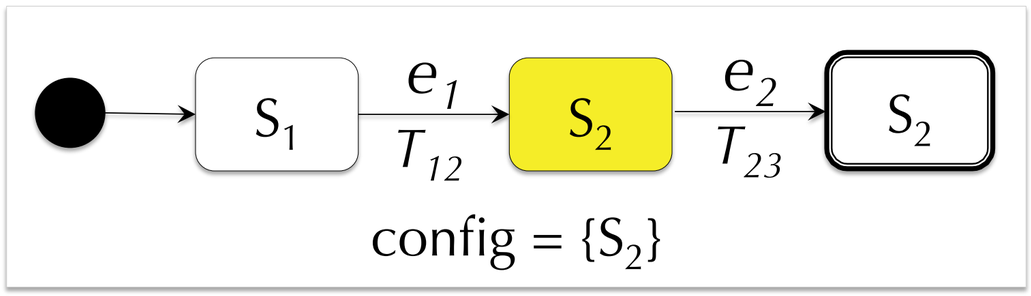
step_0	$[S_1\ S_2\ S_{10}]$
step_1 $\{\mathbf{e_0}, e_4\}$	$[S_1\ S_3\ S_{10}]$
step_1′ $\{e_0, \mathbf{e_4}\}$	$[S_1\ S_3\ S_{11}]$
step_2 $\{e_1\}$	$[S_1\ S_6\ S_{11}]$
step_3 $\{\mathbf{e_2}, e_5\}$	$[S_1\ S_8\ S_{11}]$
step_3′ $\{e_2, \mathbf{e_5}\}$	$[S_1\ S_8\ S_{12}]$
step_4 $\{e_3\}$	$[S_1\ S_9\ S_{12}]$

Step 0
(Initialization)



Current configuration (t_0)

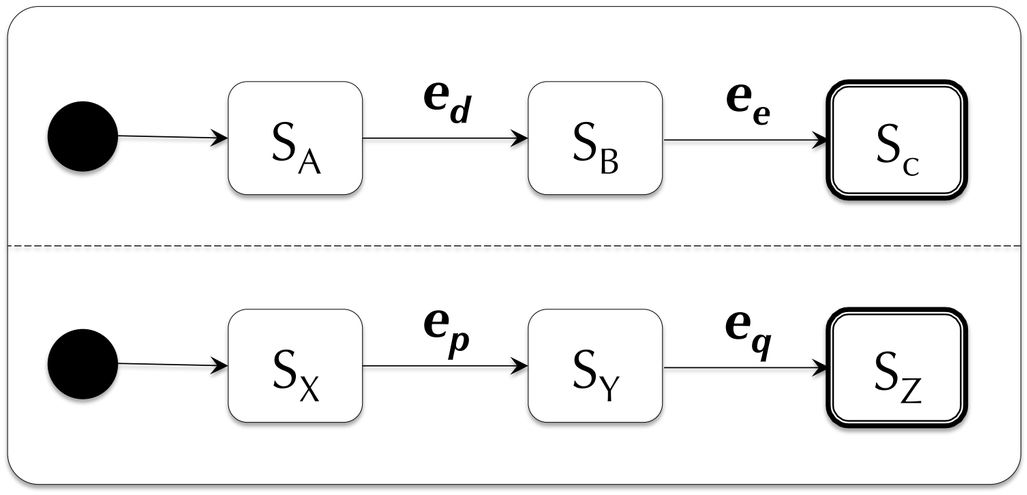
Step 1 {e_1}

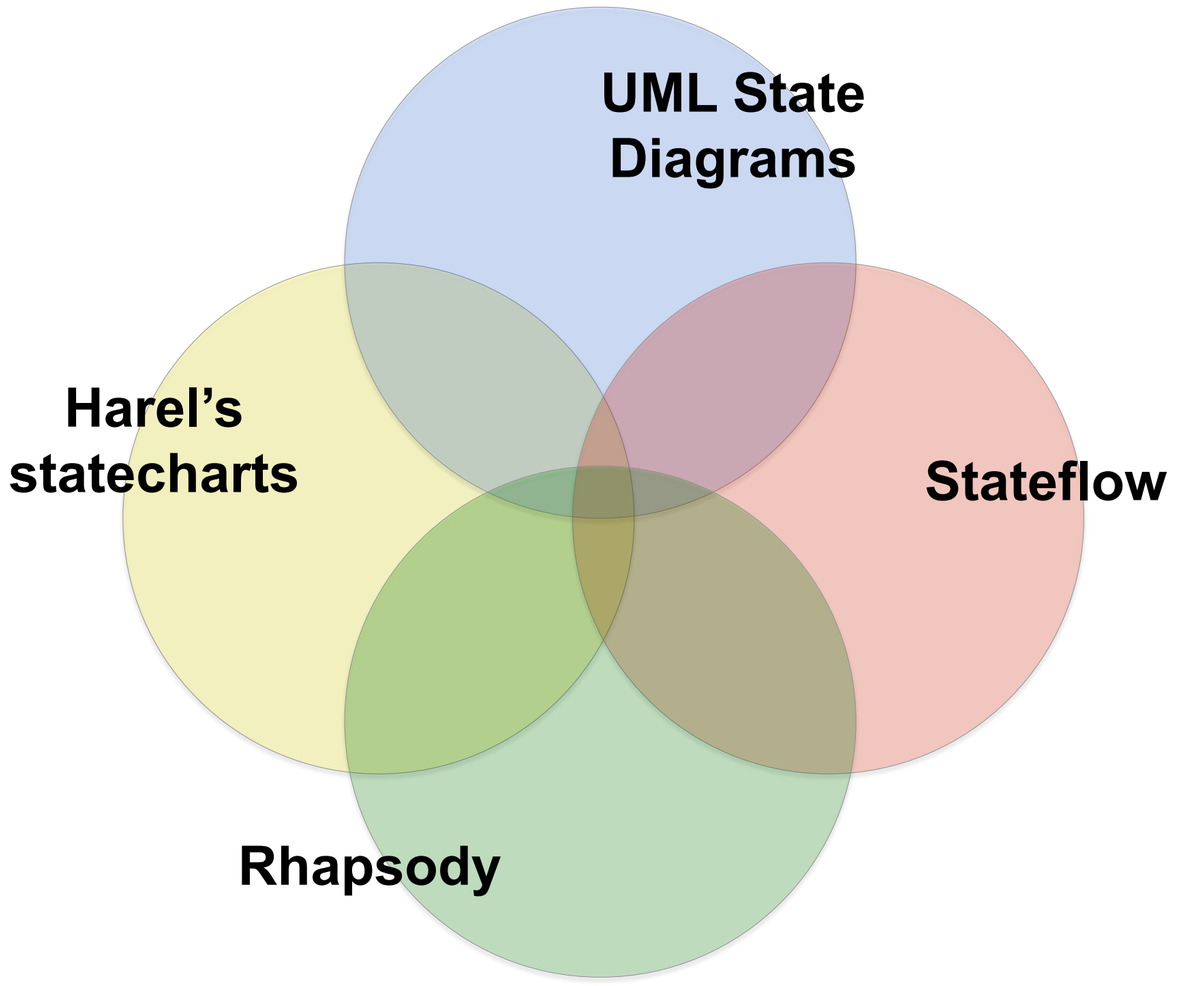


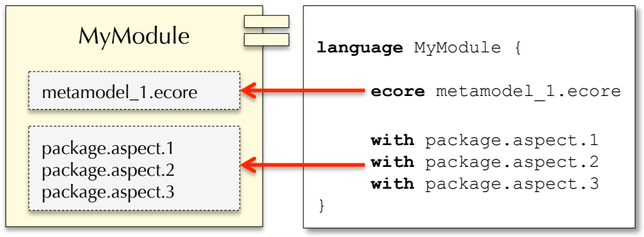
Current configuration (t_1)

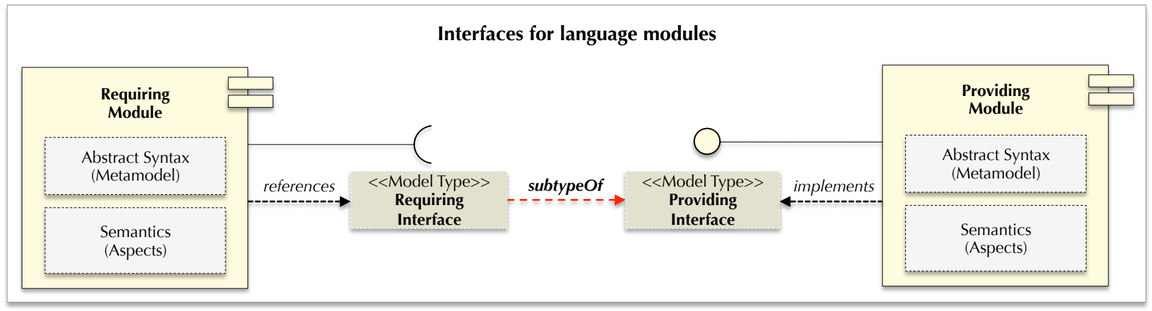
Step 2 {e_2}
(Final step)

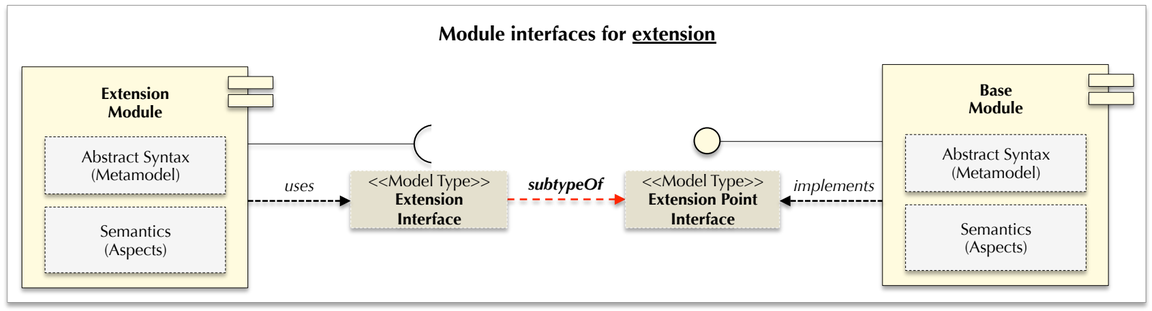
Current configuration (t_2)

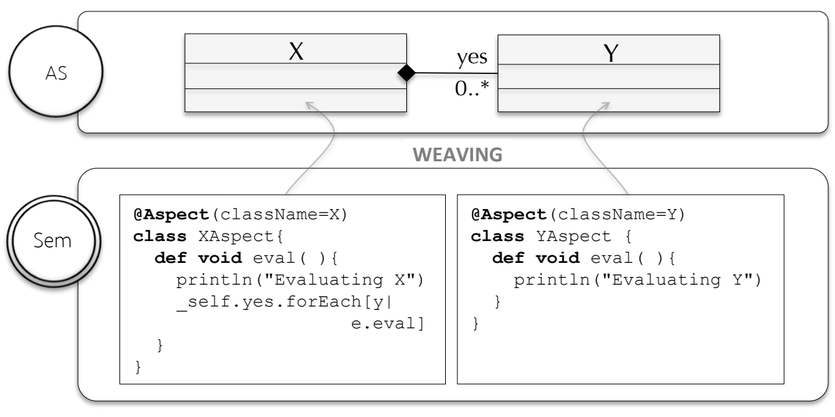


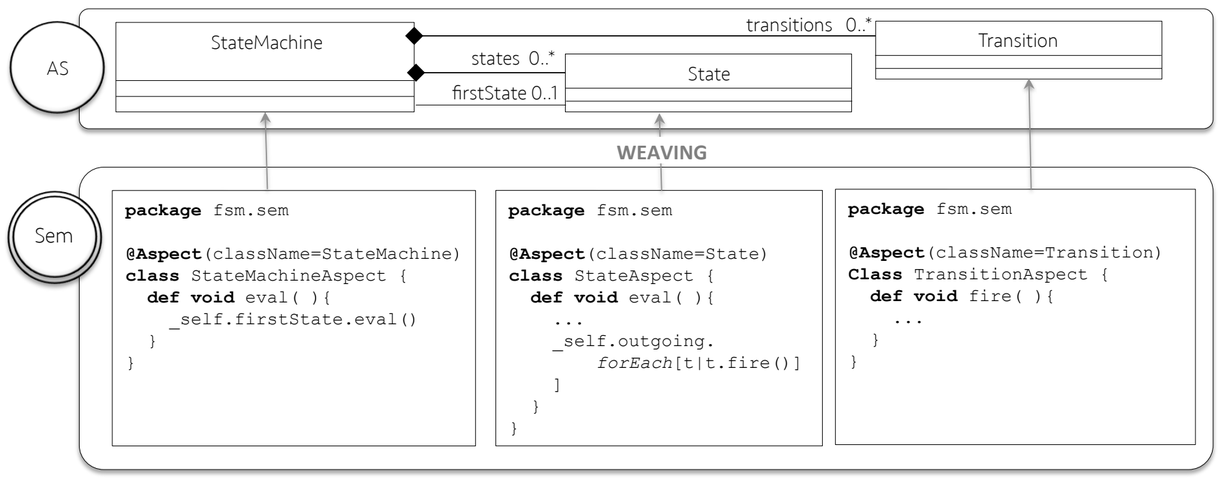

UML State Diagrams
Harel's statecharts
Stateflow
Rhapsody

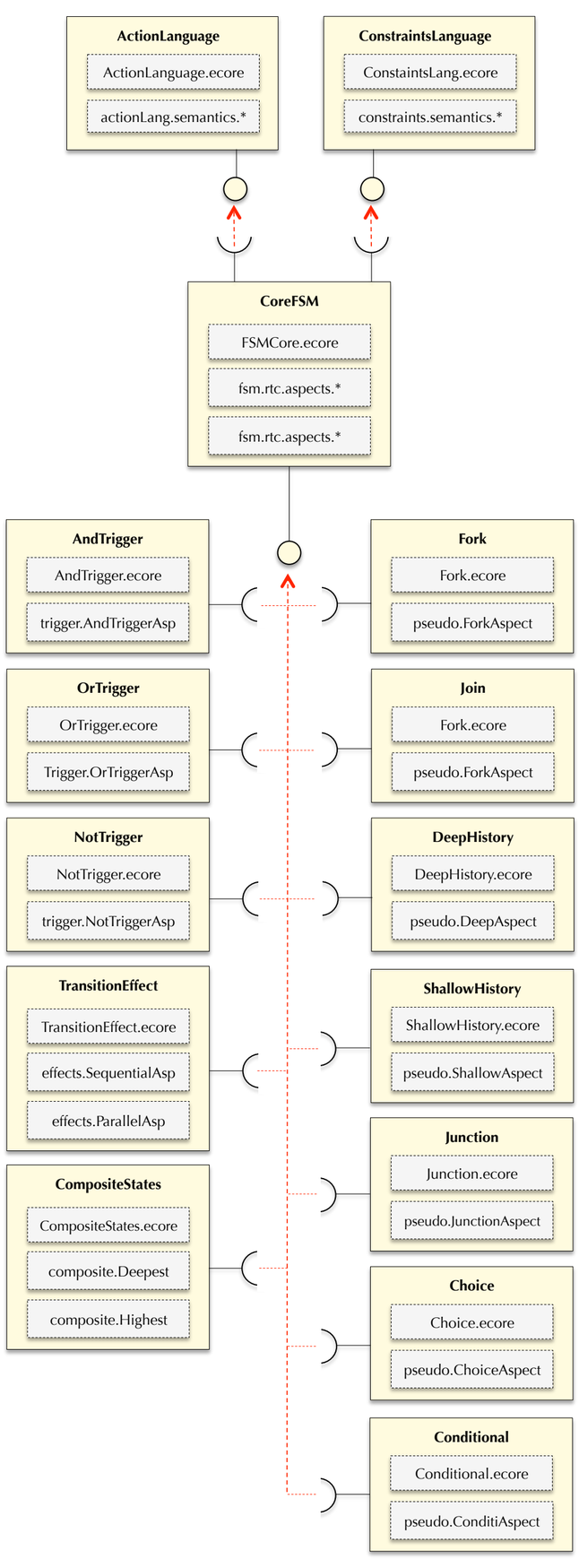

MyModule
metamodel_1.ecore
package.aspect.1
package.aspect.2
package.aspect.3
language MyModule {
ecore metamodel_1.ecore
with package.aspect.1
with package.aspect.2
with package.aspect.3
}


Interfaces for language modules
Requiring Module
Abstract Syntax (Metamodel)
Semantics (Aspects)
references
<<Model Type>> Requiring Interface
subtypeOf
<<Model Type>> Providing Interface
implements
Providing Module
Abstract Syntax (Metamodel)
Semantics (Aspects)


Module interfaces for extension
Extension Module
Abstract Syntax (Metamodel)
Semantics (Aspects)
uses
<<Model Type>> Extension Interface
subtypeOf
<<Model Type>> Extension Point Interface
implements
Base Module
Abstract Syntax (Metamodel)
Semantics (Aspects)


AS
X
yes
0..*
Y
WEAVING
Sem
@Aspect(className=X)
class XAspect{
def void eval( ){
println("Evaluating X")
_self.yes.forEach[y|
e.eval]
}
}
@Aspect(className=Y)
class YAspect {
def void eval( ){
println("Evaluating Y")
}
}

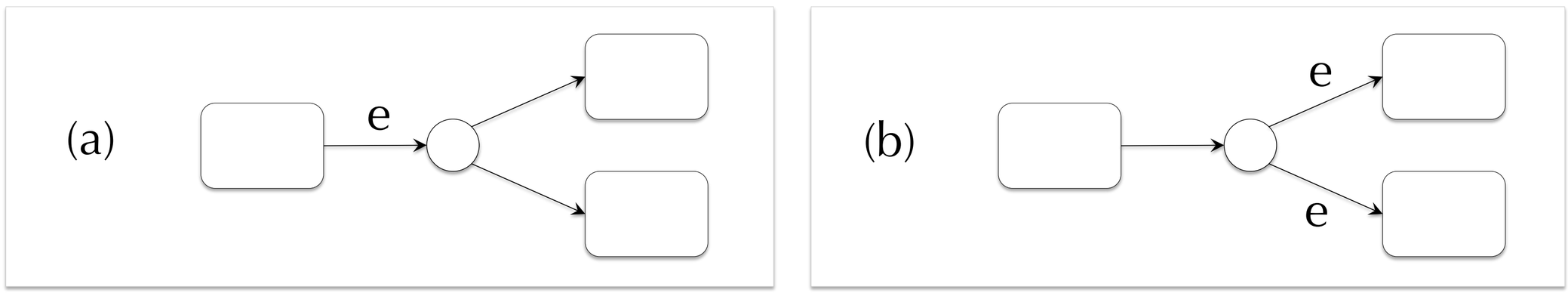

AS
StateMachine
transitions 0..*
Transition
states 0..*
State
firstState 0..1
WEAVING
Sem
package fsm.sem
@Aspect(className=StateMachine)
class StateMachineAspect {
def void eval( ){
_self.firstState.eval()
}
}
package fsm.sem
@Aspect(className=State)
class StateAspect {
def void eval( ){
...
_self.outgoing.
forEach[t|t.fire()]
]
}
}
package fsm.sem
@Aspect(className=Transition)
Class TransitionAspect {
def void fire( ){
...
}
}


ActionLanguage
ActionLanguage.ecore
actionLang.semantics.*
ConstraintsLanguage
ConstaintsLang.ecore
constraints.semantics.*
CoreFSM
FSMCore.ecore
fsm.rtc.aspects.*
fsm.rtc.aspects.*
AndTrigger
AndTrigger.ecore
trigger.AndTriggerAsp
Fork
Fork.ecore
pseudo.ForkAspect
OrTrigger
OrTrigger.ecore
Trigger.OrTriggerAsp
Join
Fork.ecore
pseudo.ForkAspect
NotTrigger
NotTrigger.ecore
trigger.NotTriggerAsp
DeepHistory
DeepHistory.ecore
pseudo.DeepAspect
TransitionEffect
TransitionEffect.ecore
effects.SequentialAsp
effects.ParallelAsp
ShallowHistory
ShallowHistory.ecore
pseudo.ShallowAspect
CompositeStates
CompositeStates.ecore
composite.Deepest
composite.Highest
Junction
Junction.ecore
pseudo.JunctionAspect
Choice
Choice.ecore
pseudo.ChoiceAspect
Conditional
Conditional.ecore
pseudo.ConditiAspect

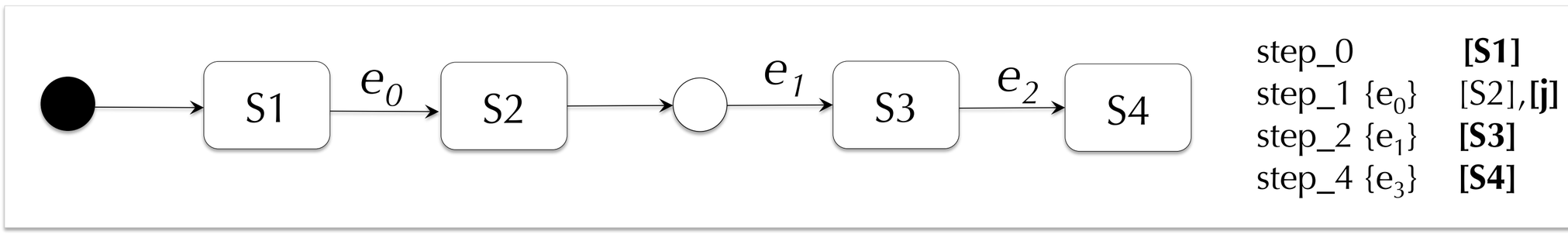
1


(a)
e
(b)
e

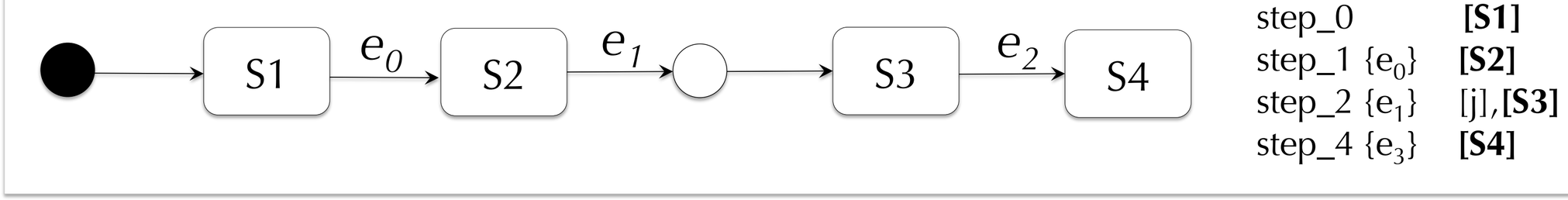
2

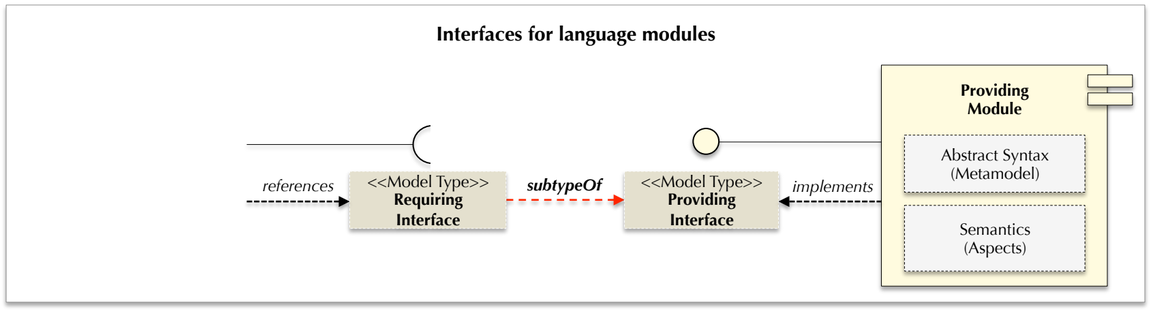

(a)
e
(b)
e
e

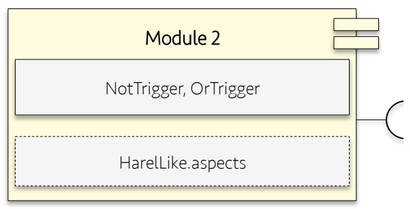
3

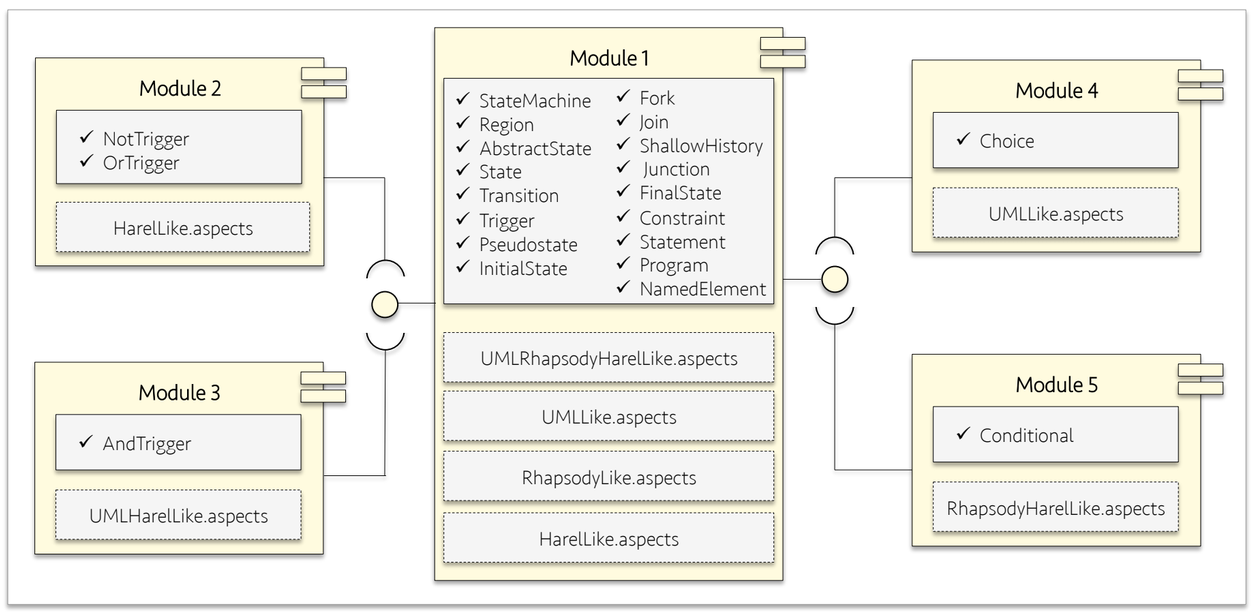

S1
e0
S2
e1
S3
e2
S4
step_0 [S1]
step_1 {e0} [S2],[j]
step_2 {e1} [S3]
step_4 {e3} [S4]

4


S1
e0
S2
e1
S3
e2
S4
step_0 [S1]
step_1 {e0} [S2]
step_2 {e1} [j],[S3]
step_4 {e3} [S4]


Interfaces for language modules
references
<<Model Type>>
Requiring
Interface
subtypeOf
<<Model Type>>
Providing
Interface
implements
Providing
Module
Abstract Syntax
(Metamodel)
Semantics
(Aspects)


Module 2
NotTrigger, OrTrigger
HarelLike.aspects


Module 2
✓ NotTrigger
✓ OrTrigger
HarelLike.aspects
Module 3
✓ AndTrigger
UMLHarelLike.aspects
Module 1
✓ StateMachine
✓ Region
✓ AbstractState
✓ State
✓ Transition
✓ Trigger
✓ Pseudostate
✓ InitialState
✓ Fork
✓ Join
✓ ShallowHistory
✓ Junction
✓ FinalState
✓ Constraint
✓ Statement
✓ Program
✓ NamedElement
UMLRhapsodyHarelLike.aspects
UMLLike.aspects
RhapsodyLike.aspects
HarelLike.aspects
Module 4
✓ Choice
UMLLike.aspects
Module 5
✓ Conditional
RhapsodyHarelLike.aspects