

UML vs. Classical vs. Rhapsody Statecharts: Not All Models Are Created Equal

Michelle L. Crane

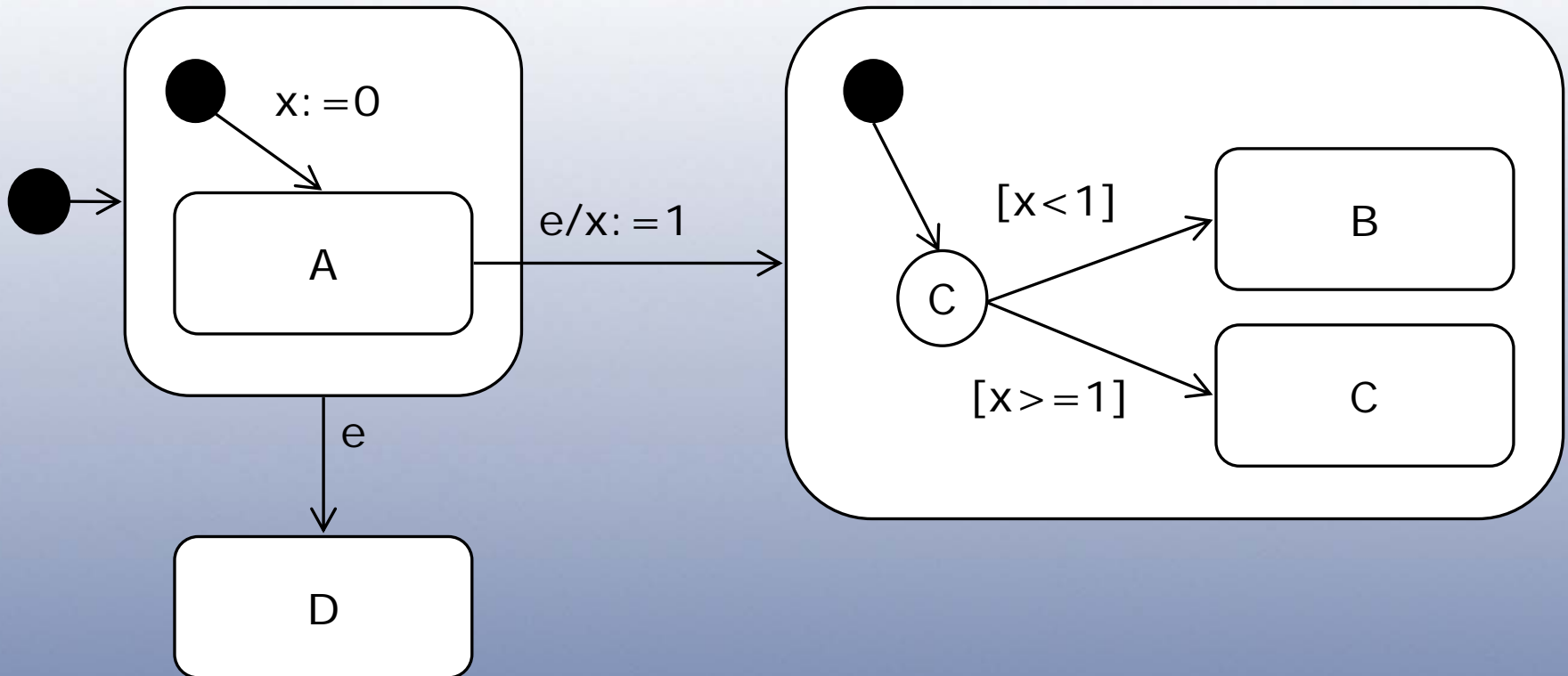
Juergen Dingel

Queen's University

Kingston, Ontario, Canada

Same State Machine

Machine starts: In A with $x=0$.
Event e occurs...



So?



- lack of standardization
 - confusion when communicating
 - modeler ↔ modeler
 - modeler ↔ customer
 - tool development
 - implement one or more formalisms
 - porting from one formalism to another may **not** be straightforward
 - may not even be possible!
- differences are syntactic and semantic

Formalisms/Notations

- Classical statecharts
 - Harel's original syntax (1987) with more recent semantics (1996-1998)
- UML 2.0
 - "industry standard" for visual modelling; statemachine diagrams are 1 of 13 diagram types
 - object-based variant of classical
- Rhapsody
 - another object-based variant of classical
 - closer to UML than to classical now

What Kind of Differences?

□ Notation

- common construct, but different notation
- e.g., junction  ↔  conditional
- least critical
 - after simple translation, model is compatible

What Kind of Differences?

□ Well-Formedness

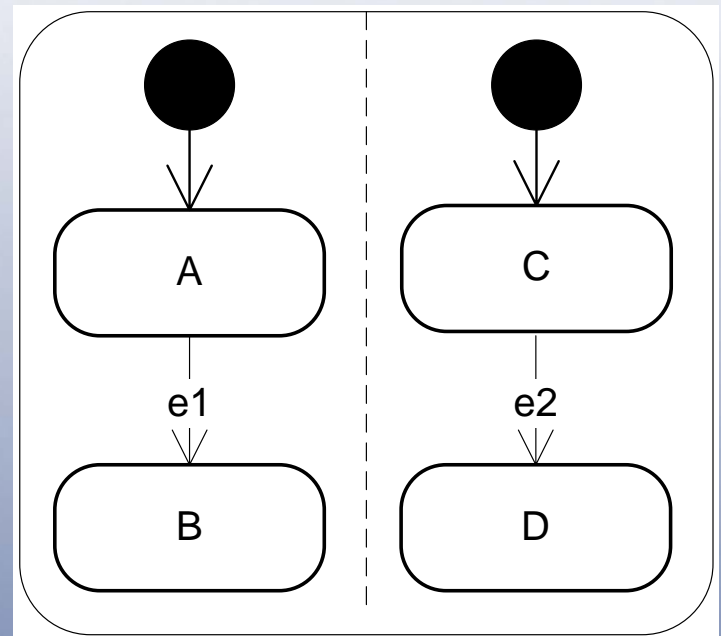
- model well-formed in one or two formalisms, but not all three
 - construct not available in a formalism
 - additional/different constraints on a construct
- translation may be possible, but not in every situation

What Kind of Differences?

- Executable Behaviour
 - most critical
 - model well-formed in different formalism, so no errors on compile
 - but behaviour different
 - e.g., priority scheme

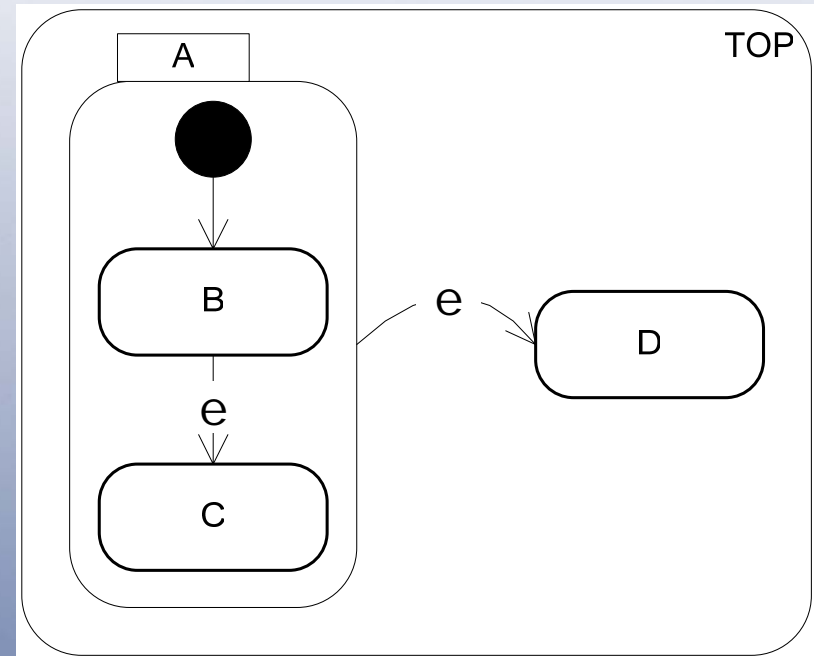
Simultaneous Events

- handled only by Classical
 - $(A,C) \rightarrow (B,D)$
- in UML, Rhapsody
 - if $e1 \neq e2$
 - $(A,C) \rightarrow (x,y) \rightarrow (B,D)$



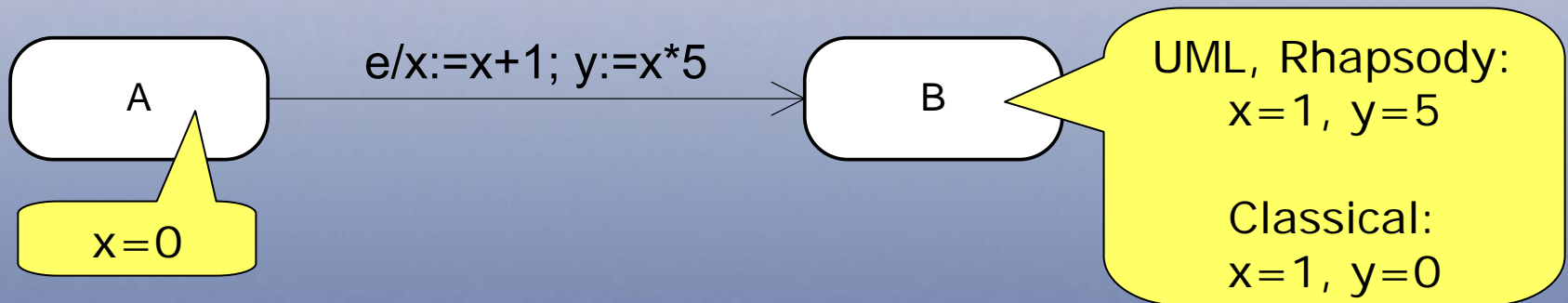
Priority

- priority of conflicting transitions
 - classical
 - scope = lowest OR-state neither exited nor entered
 - priority goes to highest scope
 - UML, Rhapsody
 - priority goes to lowest source state

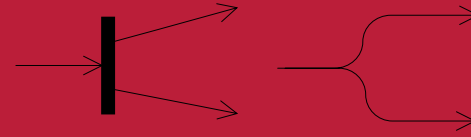


Order of Execution

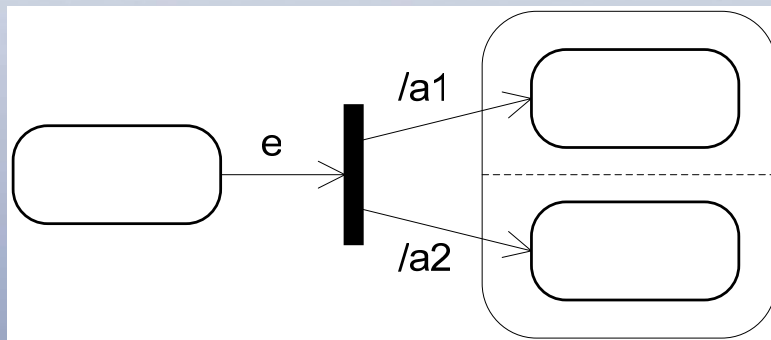
- multiple actions along transition
- UML, Rhapsody
 - executed in sequence
- Classical
 - executed in parallel



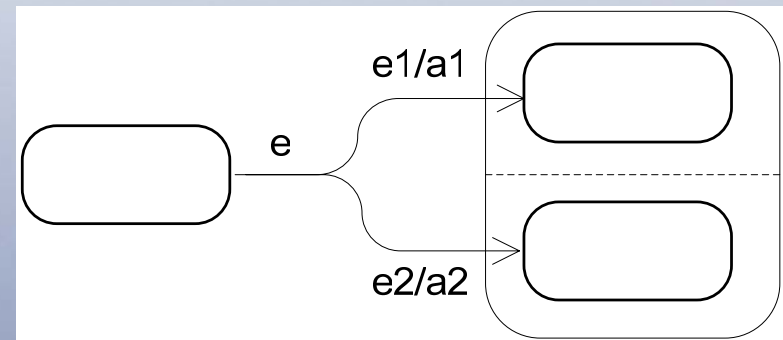
Fork



- ❑ Rhapsody: no labelling after fork
- ❑ UML: no events after a fork



- ✗ Rhapsody
- ✓ UML

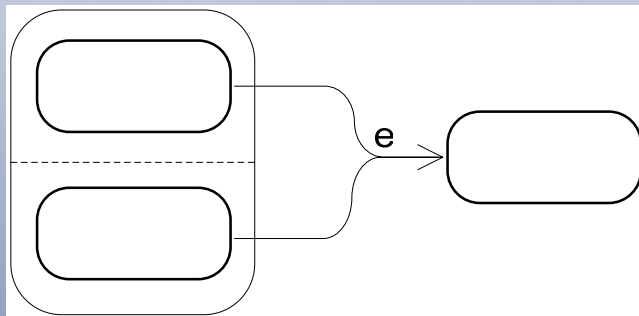


- ✗ UML
 - ✗ Rhapsody
 - ✓ Classical
- Classical: only if
e, e1, e2
simultaneous

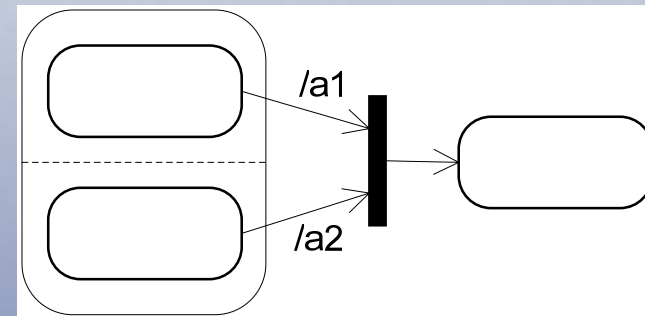
Join



- ❑ UML: no events before or after a join
- ❑ Rhapsody: no actions before a join

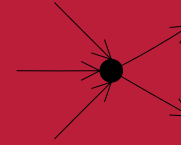


- ✗ UML
- ✓ Classical

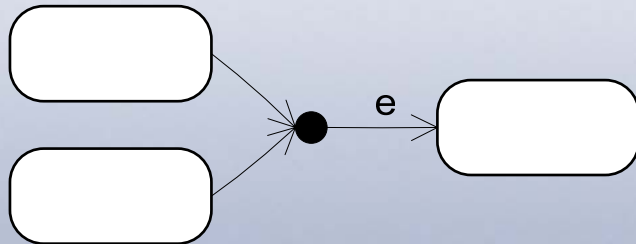


- ✓ UML
- ✗ Rhapsody

Junction

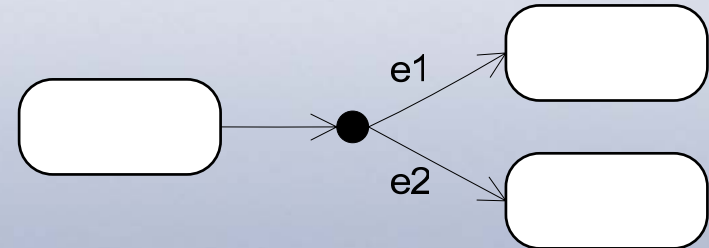


- ❑ UML: no events after junction
- ❑ Rhapsody: only one exit



- ✓ Classical
- ✗ UML

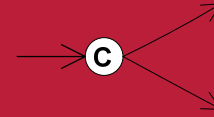
Make UML compatible by moving event



- ✓ Classical
- ✗ UML
- ✗ Rhapsody

Can only be triggered if e1, e2 simultaneous.

Conditional



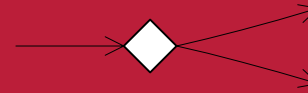
static choice

- guards evaluated before the transition is taken

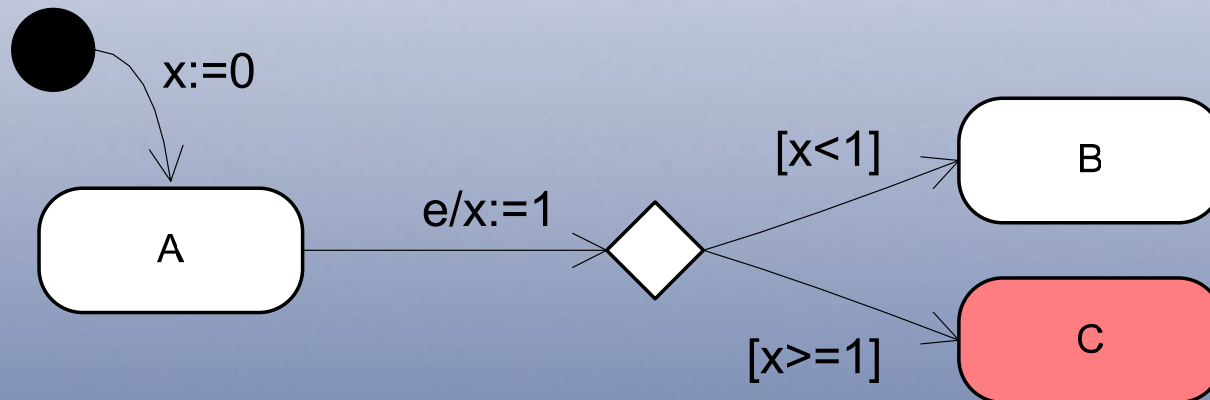
UML

- removed from UML 1.3
- equivalent to junction

Choice



- dynamic choice
 - not equivalent to junction
 - guards evaluated as the choice is reached
- only in UML



Comparison Summary

Construct/Concept	UML	Class.	RHAP.	Note
Syntax				
States				
entry/exit actions	●	⊙	●	1
do-activity	●	⊙	⊙	2
deferred events	●	⊗	⊗	
Pseudo-states				
initial	●	●	●	3
final	●	●	●	4
fork	●	⊙	⊙	5
join	●	⊙	⊙	5
shallow history	●	⊙	⊗	6
deep history	●	⊙	⊙	6
junction (static)	●	●	●	7
conditional (static)	N/A	+	+	8
choice (dynamic)	●	⊗	⊗	
Transitions				
event trigger	●	⊙	●	9
action (behaviour)	●	⊙	●	1
completion	●	⊘	⊘	10
Semantics				
simultaneous events	N/A	+	⊗	11
simultaneous actions	N/A	+	⊗	12
priority	●	⊙	●	13

●	little difference
⊙	considerable difference
⊗	not supported
+	not by UML but by other



Comparison Summary

Construct/Concept	Notation	Well-Form.	Behaviour
Syntax			
States			
entry/exit actions			✓
do-activity		✓	
deferred events		✓	
Pseudo-states			
initial			
final	✓		
fork	✓	✓	
join	✓	✓	
shallow history		✓	
deep history		✓	
junction (static)		✓	✓
conditional (static)	✓		✓
choice (dynamic)		✓	
Transitions			
event trigger		✓	
action (behaviour)			✓
completion			
Semantics			
simultaneous events		✓	✓
simultaneous actions			✓
priority			

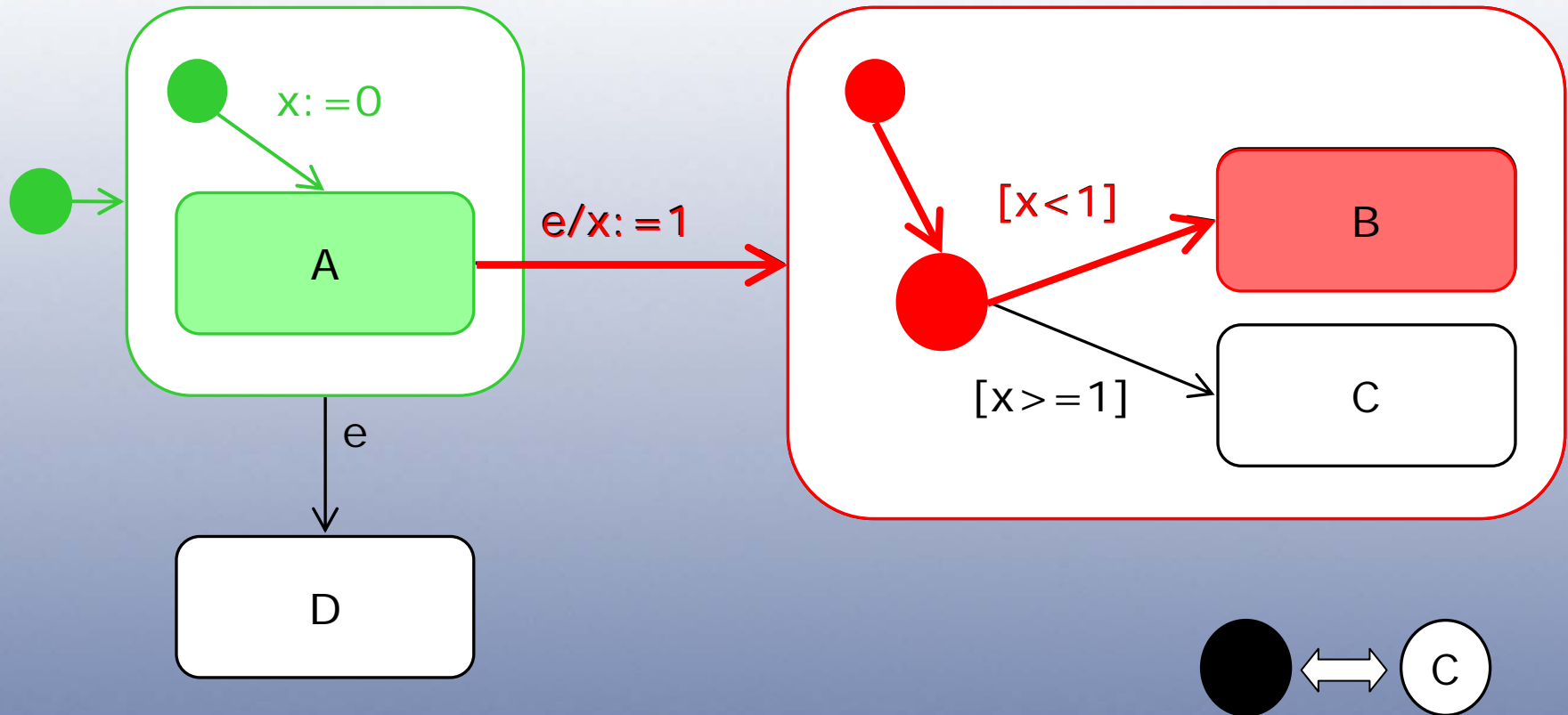
●	little difference
⊙	considerable difference
⊗	not supported
+	not by UML but by other

Comparison Conclusions

- Rhapsody and UML closer than either is to Classical
 - porting
- dynamic choice
- simultaneous events
 - many of the well-formedness and execution behaviour differences
 - do-activities, forks, joins, junctions, event triggers
- priority scheme
 - does not cause notation or well-formedness problems
 - model that would behave differently due to priority scheme would **not** be found by syntax or well-formedness check

Different Effects – UML

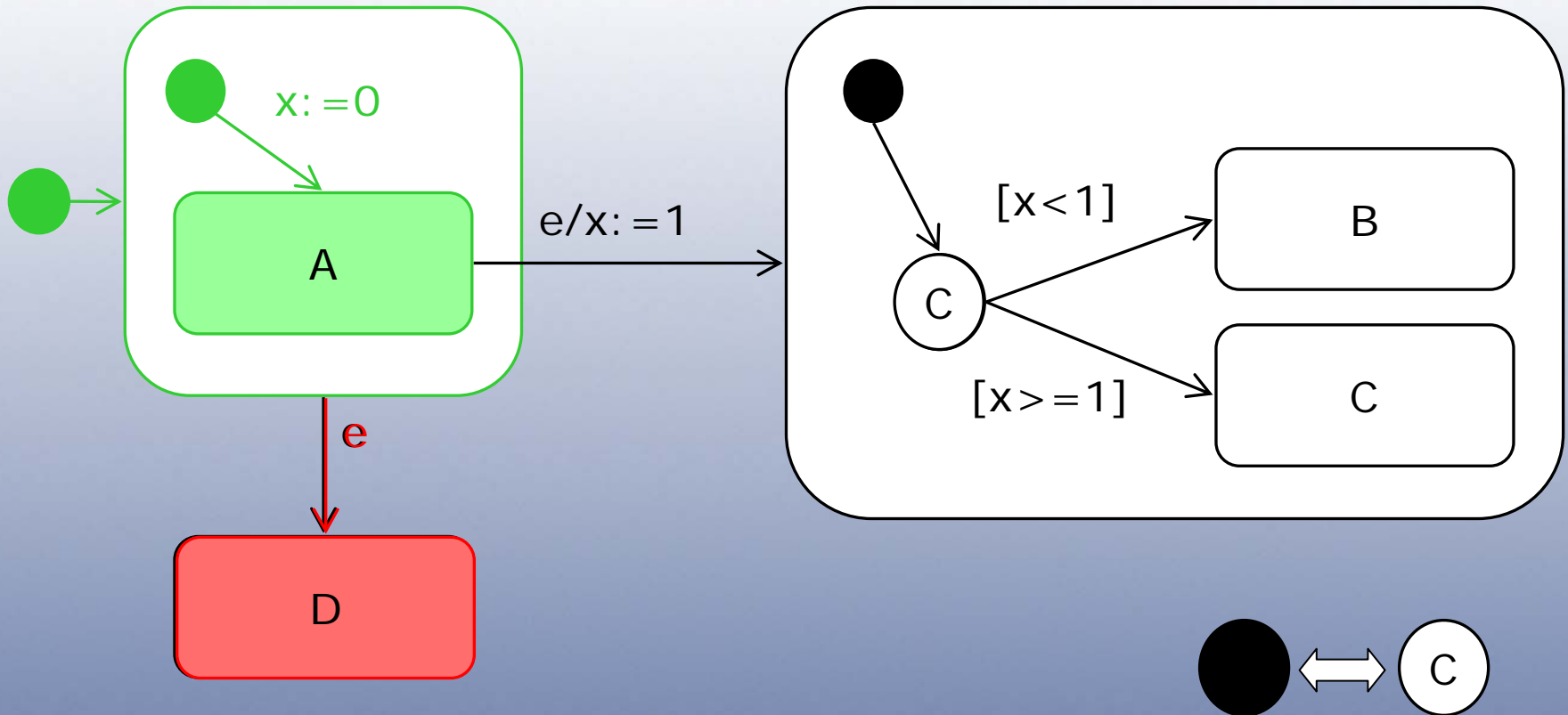
Machine starts: In A with $x=0$.
Event e occurs...





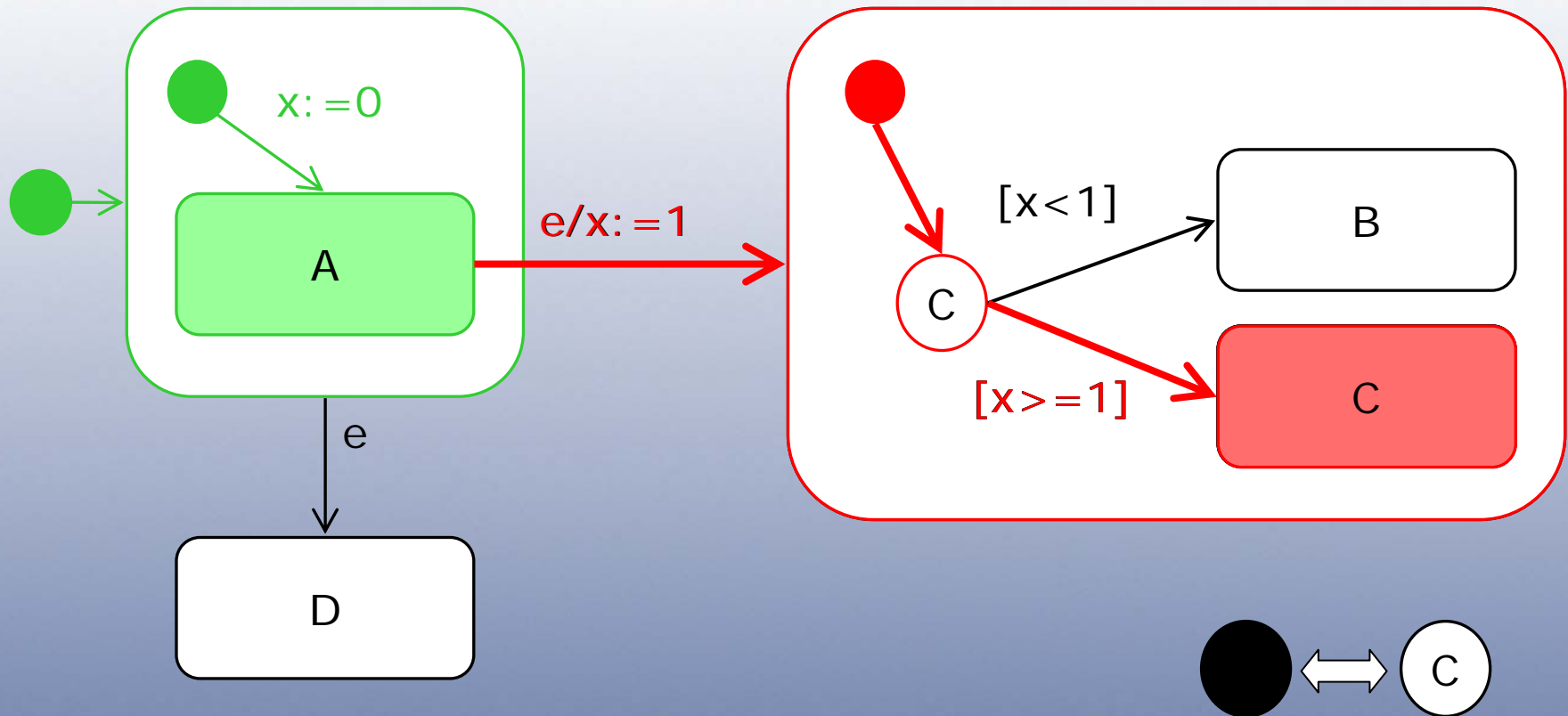
Different Effects – Classical

Machine starts: In A with $x=0$.
Event e occurs...



Different Effects – Rhapsody

Machine starts: In A with $x=0$.
Event e occurs...



Conclusion

simple state machines

→ complicated interpretation

UML vs. Classical vs. Rhapsody Statecharts: Not All Models Are Created Equal

Michelle L. Crane
Juergen Dingel

Queen's University
Kingston, Ontario, Canada

