

**The meaning of
concurrent states in
UML state diagrams**

SAC 2003, Melbourne, Florida

Egon Börger, Alessandra Cavarra, Elvinia Riccobene

Contents

- Unified Modeling Language
- State Diagrams
- Abstract State Machines
- Language issues

Unified Modeling Language

- a single **framework** in which to place the sketches, diagrams, pictures, and formulae that shape and communicate our understanding of complex systems;
- a **welcome** development for advocates of precise, mathematical, techniques:
 - integrating formal methods;
 - linking theory to practice.
- a hybrid language with an agreed **formal syntax**, but without a formal semantics.

“...a completely formal specification... would have added significant complexity without clear benefit.

In addition, the state of the practice in formal specifications does not yet address some of the more difficult language issues that UML introduces.”

UML 1.4 with Action Semantics
Object Management Group, 2002

Fidelity

“Every model may be expressed at different levels of fidelity.”

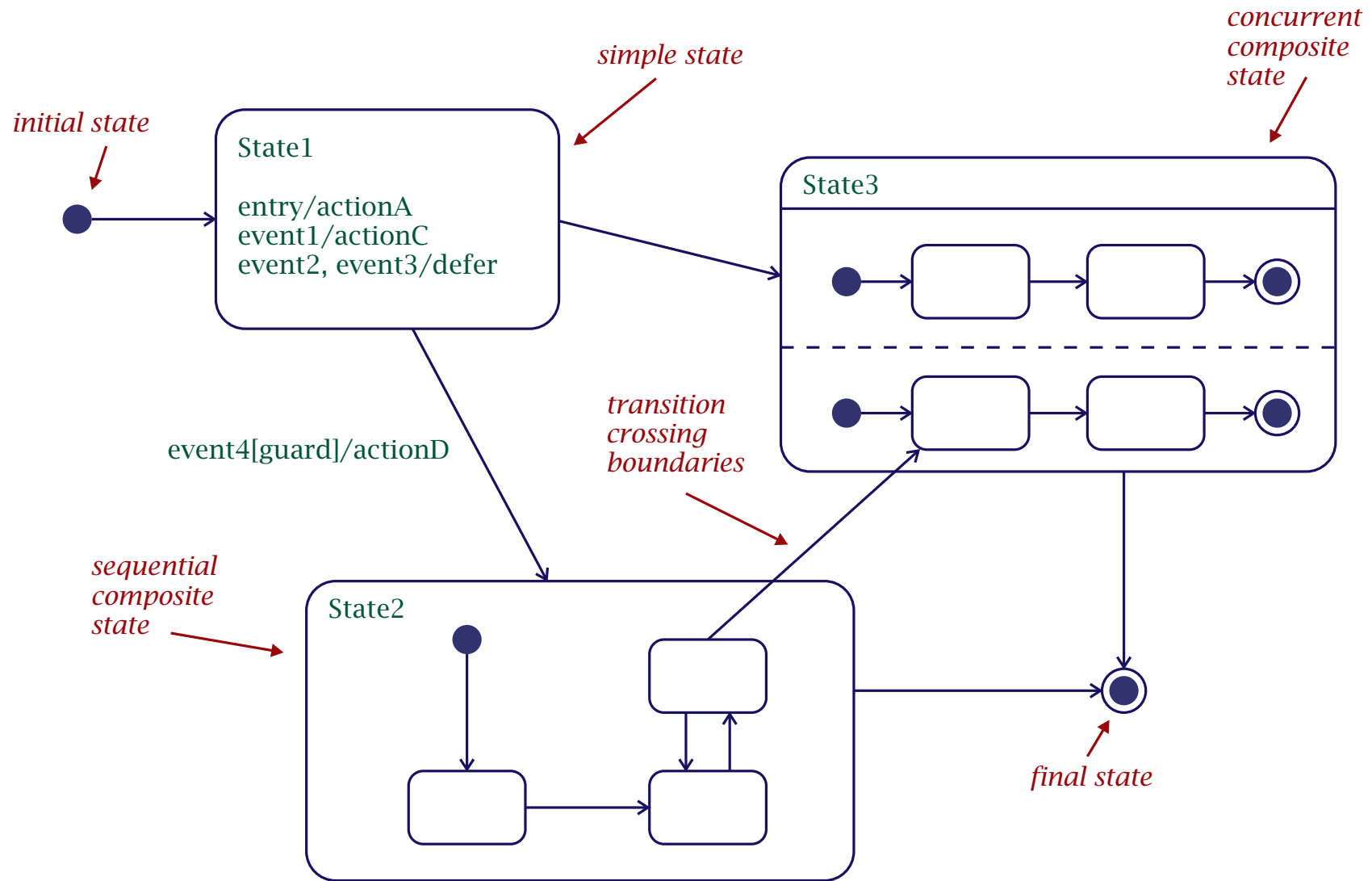
Formal semantics

If a model is intended to be precise—or **faithful**—then a formal semantics can be used:

- to check that the various components of the model are mutually consistent;
- to check that the system as described would exhibit particular properties;
- to compare two different models of the same system, or system component;
- to generate, automatically, a suite of tests

State Diagrams

- describe an abstraction of the **state space** for objects of a particular class;
- the state space is divided into regions linked by guarded, labelled **transitions**;
- the diagram explains the sequence of **actions** produced in response to stimuli, or processed **events**.



A UML state diagram

Formal semantics for state diagrams

There are two reasons for giving a formal semantics to the (sub-) language of UML state diagrams:

1. giving a semantics to **object models**—being able to calculate the consequences of design decisions taken by **users**;
2. providing a critique of the **language** itself—being able to calculate the consequences of decisions taken by the **OMG**.

Abstract State Machines

- collections of attributes...
- updated by imperative rules...
- fired in combination as and when enabled.

Semantics

We represent:

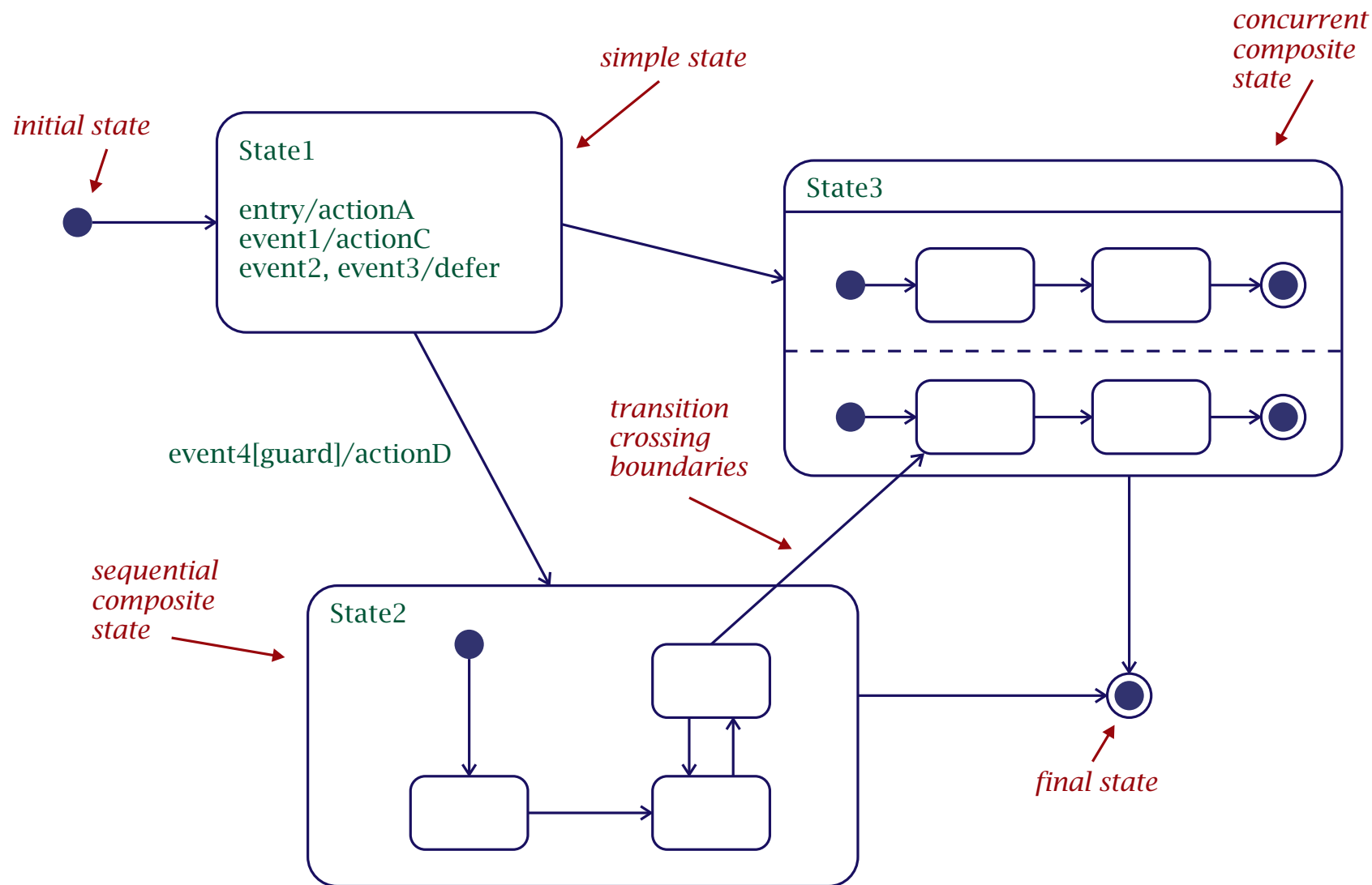
- the hierarchy of composite states;
- the currently active states;
- the various queues of events;
- the transitions.

Example

```
stateMachineExecution(trans) ≡  
  if internal(trans) then action(trans)  
  else  
    seq exitState(source(trans), ToS)  
    action(trans)  
    enterState(FromS, target(trans))  
    case target(trans)  
      SequentialState : enterInitialState(target(trans))  
      ConcurrentState : startConcurrComput(target(trans))  
      HistoryState : restoreConfig(target(trans))  
    endcase  
  where  
    anc = lca(source(trans), target(trans))  
    ToS = directSubState(anc, UpChain(source(trans), anc))  
    FromS = directSubState(anc, DownChain(anc, target(trans)))
```

Language issues

- run-to-completion;
- deferred events;
- transition priority;
- internal activity.



A UML state diagram

Footnote

“The dynamic semantics are described using natural language, although in a precise way so that they can easily be understood.

Currently, the dynamic semantics are not considered essential for the development of tools; however, this will probably change in the future.”

Indeed, the beta-release version of the new UML documentation includes a trace semantics for state and interaction diagrams.

AGEDIS

www.agedis.de