# The STATEMATE Semantics of Statecharts

## by David Harel

Presentation by:
John Finn
October 5, 2010

# Outline

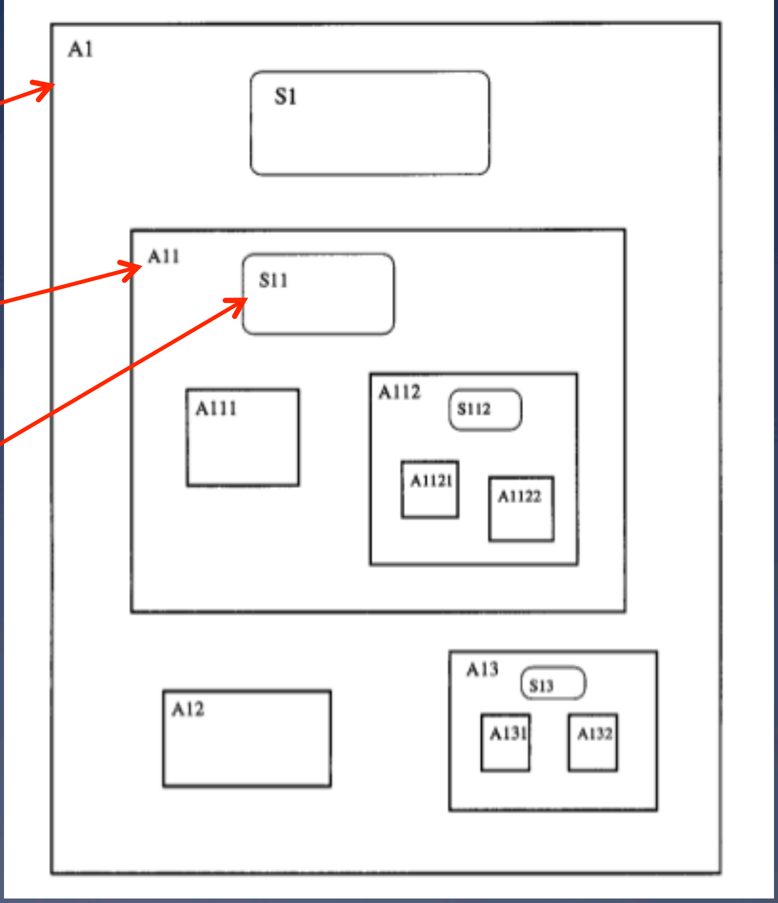
- Introduction
- The Basics
- System Reactions
- Compound Transitions
- History
- Scope of Transitions
- Conflicting Transitions

# Introduction

- No official semantics
- Nearly 20 variants [von der Beek 1994]
- Clarity and Simplicity
- STATEMATE semantics, which is a commercial tool for the specification and design of complex systems

# The Basics: Activity Chart

- Hierarchy
- Root
- Activities
- Control Activities
- OR/AND/Basic States

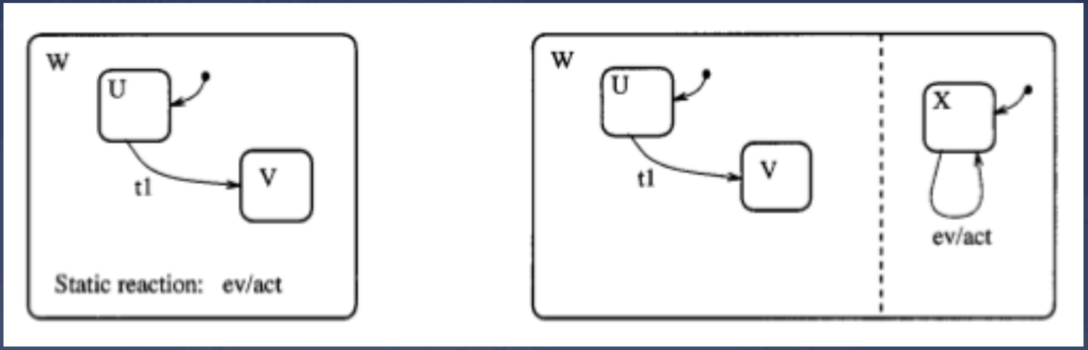


# The Basics: Syntax

- e[c]/a
- e: event, which triggers a transition
- c: condition, which enables the transition if true
- a: action, which is carried out if the transition is triggered and its condition is true
- Special Events: enter(S), exit(S)

# The Basics: States

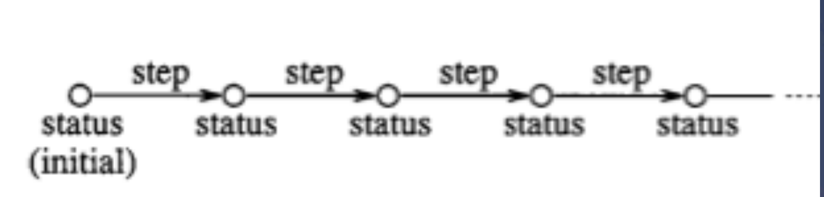
- Static Reactions have the e[c]/a syntax, and can be carried out if the system is in the state
- Virtual State



- Activities can be active “within” or “throughout” a state

# The Basics: System

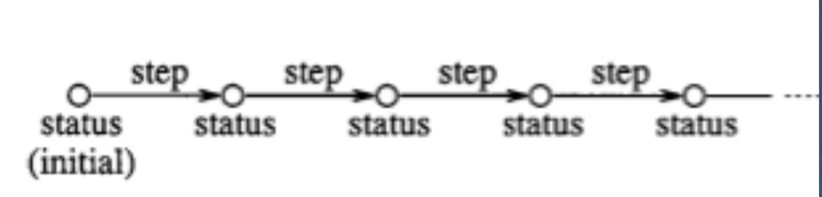
- Runs represent “snapshots” of the system’s response to an external stimuli
- Each snapshot is called a Status, which includes:
  - Active states
  - Activities
  - Data and conditional values
  - Generated events
  - Scheduled actions
  - Past behavior



- System changes status by executing a Step

# The Basics: Semantics

- Reactions to events and system changes can only be sensed after the step is complete
- Events only "live" for the step following the one in which they occur
- Calculations in one step are based on the status at the start of that step
- The maximal subset of non-conflicting transitions and static reactions are always executed
- A step takes zero time



# System Reactions: Configuration

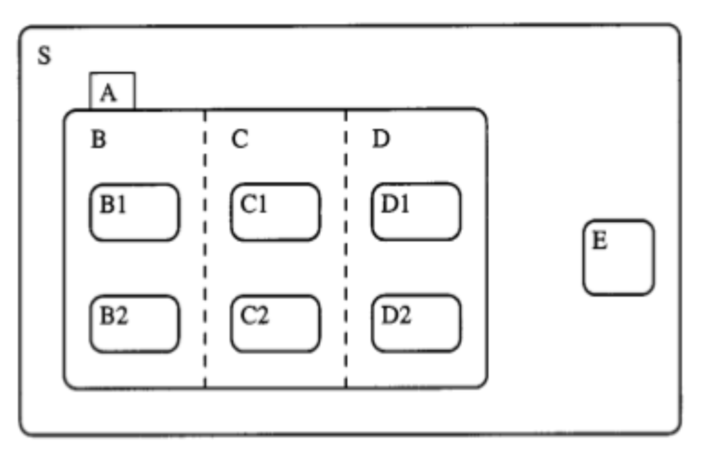
- Configuration is the maximal set of states a system can be in simultaneously
- Consider a root state, R and a configuration, C
  - C must contain R
  - If C contains an OR state A, it must contain one of A's sub-states
  - If C contains a AND state A, it must contain all of A's sub-states
  - No extraneous states, all states must be require by the rules above

# System Reactions: Configuration

- If the system is in state A, it must also be in A's parent state, unless the current state is the root
- Basic configurations consist of only basic states
- For example:
  - Basic Config: {B1, C1, D1}, {E}
  - Full Config: {B1, C1, D1, B, C, D, A, S}
  - Can you spot another Full Config?
  - Illegal Config:
    - {B1, B2, C1, D1}
  - Non-maximal Config:
    - {B1, C1}
  - What about {B2, C1, D2}?

{E, S}

Basic Configuration

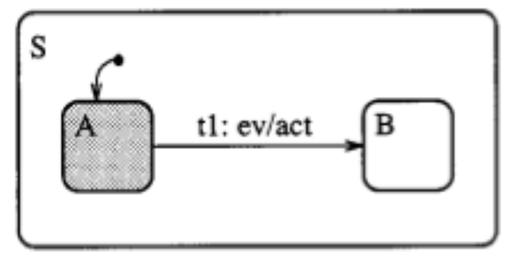


# System Reactions: Operations

- How does a system change its status:
  - Transitions
  - Static Reactions
  - Actions performed when entering a state
  - Actions performed when exiting a state

# System Reactions: Transitions

- Transition becomes enabled when within the transition's source state and the event becomes true
- For example: Exit A and Enter B
  - exit(A) and enter(B) are generated
  - in(A) becomes false, in(B) become true
  - Exiting A actions take place
  - Entering B actions take place
  - State S's Static Reactions are executed
  - Activities within or throughout A are deactivated, while activities within (not necessarily) or throughout B are activated



# System Reactions: Transitions

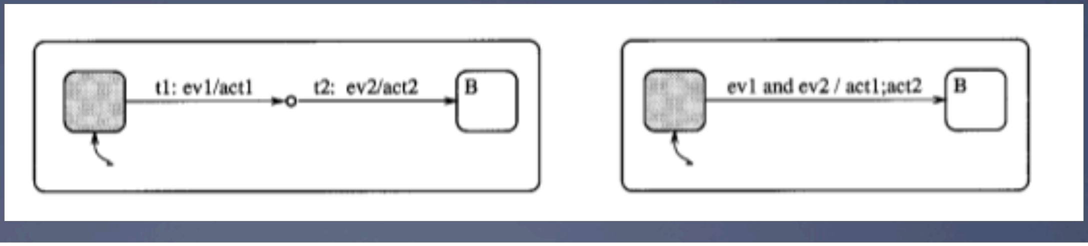
- All of the mentioned changes are sensed in the next step
- For example, For the step below, which act is executed if X is initialized to 4? 5?
  - X := X + 1;
  - **if** X = 5 **then** act1 **else** act2 **end**

act2; act1

- Racing Condition: when two or more actions attempt to change a variable in the same step, the outcome is unpredictable

# Compound Transitions: Rules

- Each step must lead the system into a legal configuration
- A system cannot be in a non-basic state without the ability to enter a sub-state
- Transition Segment: labeled arrow which can connect states and other transitions
- Basic Compound Transition: maximal chain of transition segments that are executed simultaneously

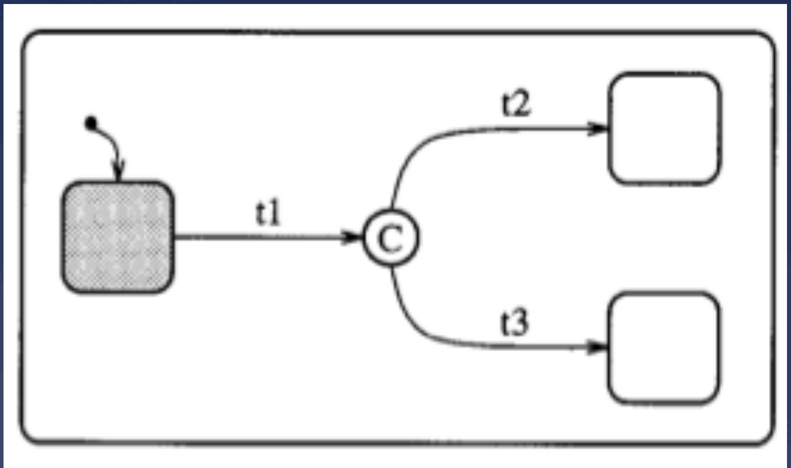


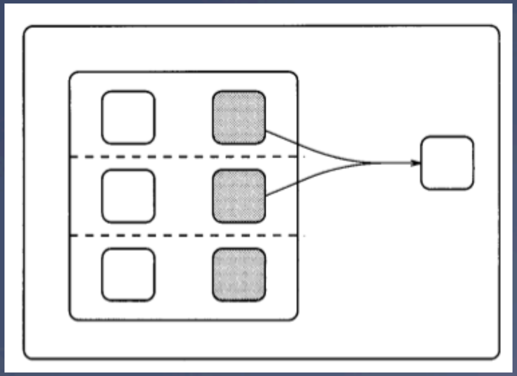
# Compound Transitions

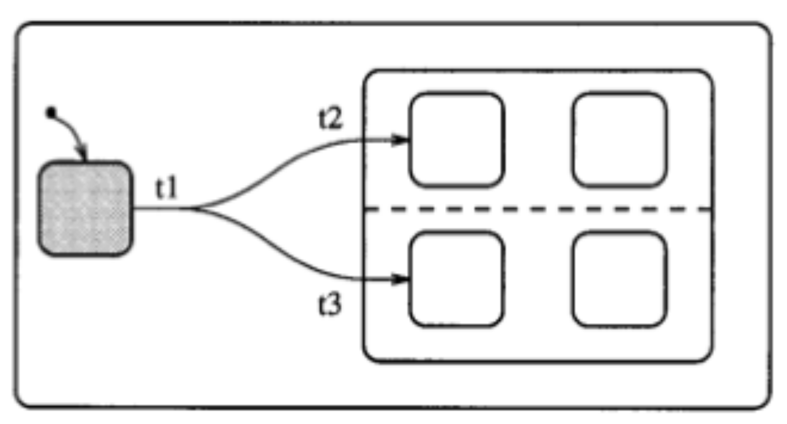
- Joint/Fork are AND connectors
- Condition/Selection/Junction are OR connectors
- Initial CT: source of the CT is a state
- Continuation CT: source is a default or history connector
- Full CT: Contains one initial CT and potentially several continuation CTs

# Compound Transitions: Examples

- OR connectors
  - Two CTs:
    - {t1, t2}
    - {t1, t3}
- AND connectors
  - {t1, t2, t3}

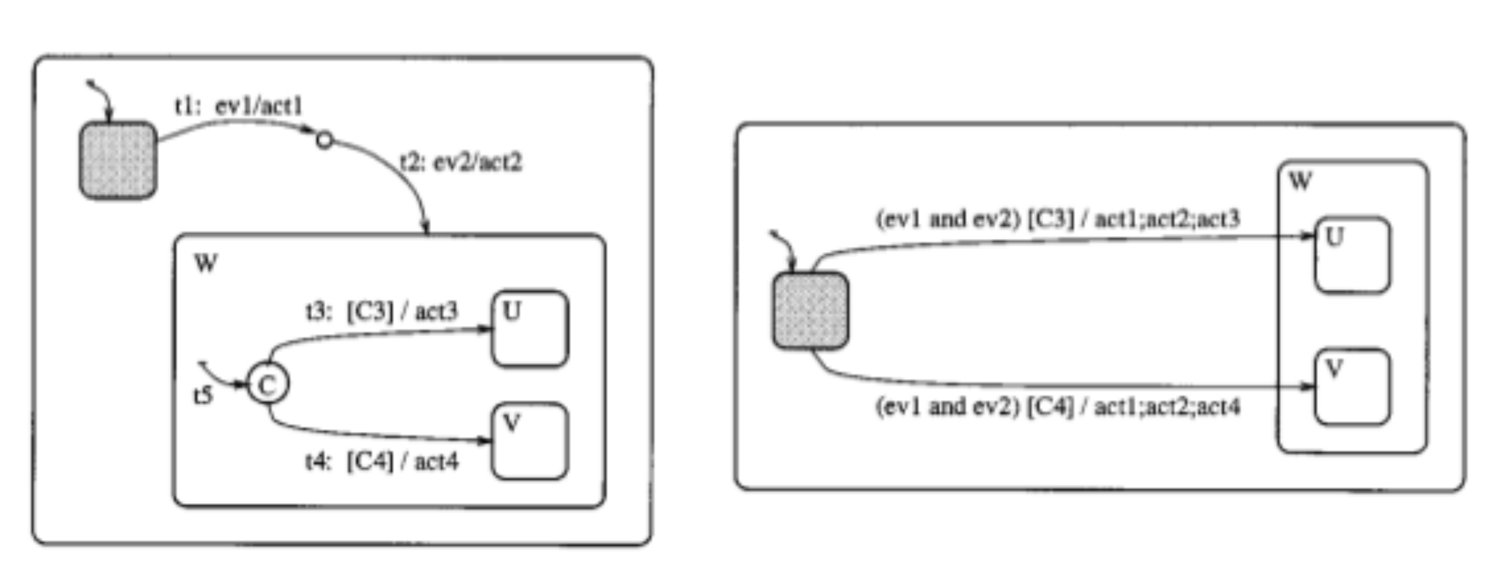






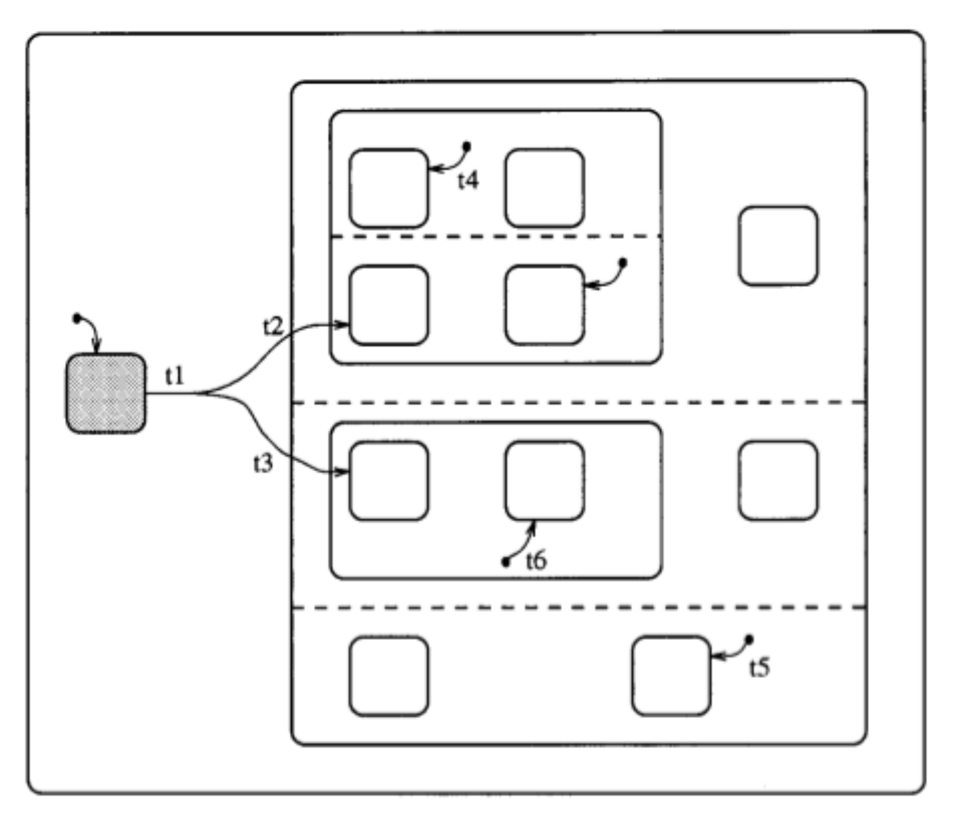
# Compound Transitions: Examples

- More complicated…
  - t1 and t2 must be executed together, which leads into t5
  - Then, t3 OR t4
  - Full CTs: {t1, t2, t5, t3} or {t1, t2, t5, t4}


t1: ev1/act1
t2: ev2/act2
W
t3: [C3] / act3
U
t5
C
t4: [C4] / act4
V
W
(ev1 and ev2) [C3] / act1;act2;act3
U
V
(ev1 and ev2) [C4] / act1;act2;act4

# Compound Transitions: Examples

- Initial CT
  - {t1, t2, t3}
- Full CT
  - {t1, t2, t3, t4, t5}
- Why not t6?



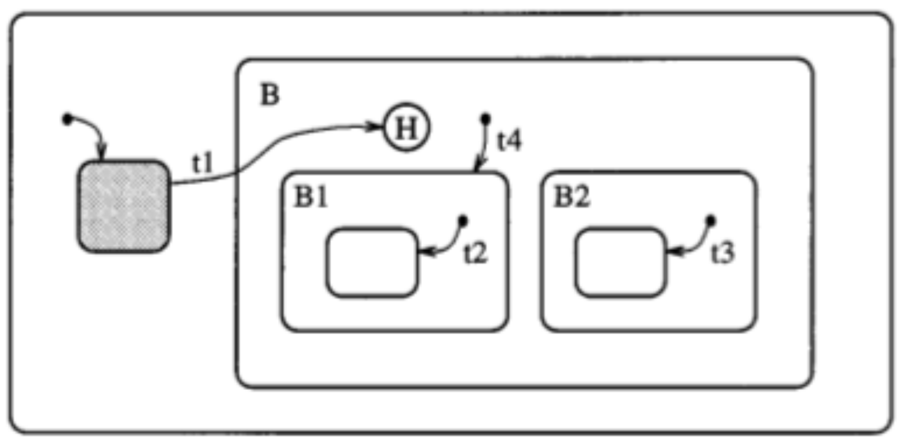
# History

- Two types of history connectors
- Suppose we are executing a CT, t1 to state S
  - H Connector
    - Let S' be the sub-state of S which the system was in when most recently in S
    - t1 is treated as if its target is S' instead of S
  - H* Connector
    - Let S' be the basic configuration relative to S which the system was in when most recently in S
    - t1 is targets all of the states in S'
  - If entering S for the first time, t1 is treated as if it is targeting S

# History: Example

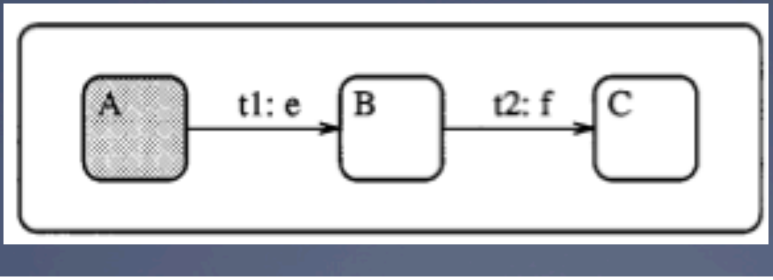
- Transition t1 is taken
  - If B was last in B1 the last time in B, then B' = B1
    - The full transition become {t1, t2}
  - If B was last in B2  the last time in B, then B' = B2
    - {t1, t3}
  - If entering B for the first time?

{t1, t4, t2}



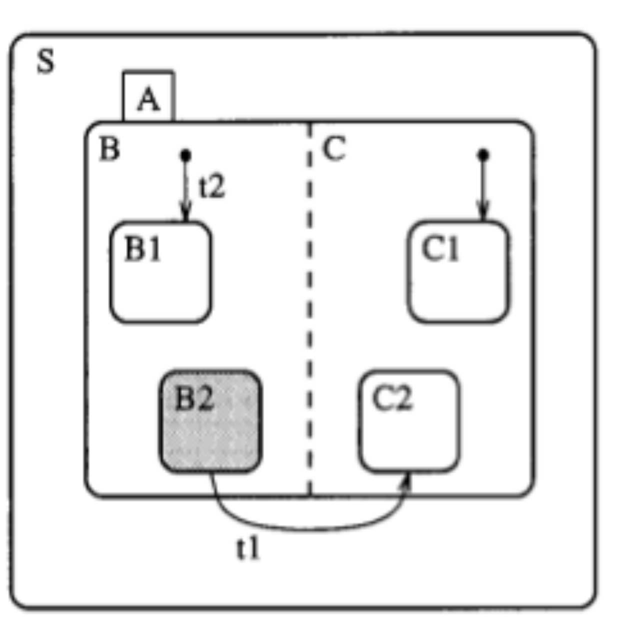
# Scope of Transitions

- If the system is in A to start and events e and f are triggered during the previous step
  - Transition t1 become active but not t2
  - The system is now in state B, but it does not know f was triggered previously, and therefore, it will only go to C if f is triggered again
- CT is enabled in a step if at the beginning of the step the system is in all the states of its source and if its trigger is true



# Scope of Transitions

- The previous example seems simple, however, consider this example
- When executing t1, should we exit and reenter A?
- Similarly, should events that trigger from exiting or entering A be executed?
- Transition Scope answers these questions



# Scope of Transitions

- The scope of a transition is the lowest OR state in the hierarchy of states that is a proper common ancestor of all the sources and targets of that transition, including non-basic states

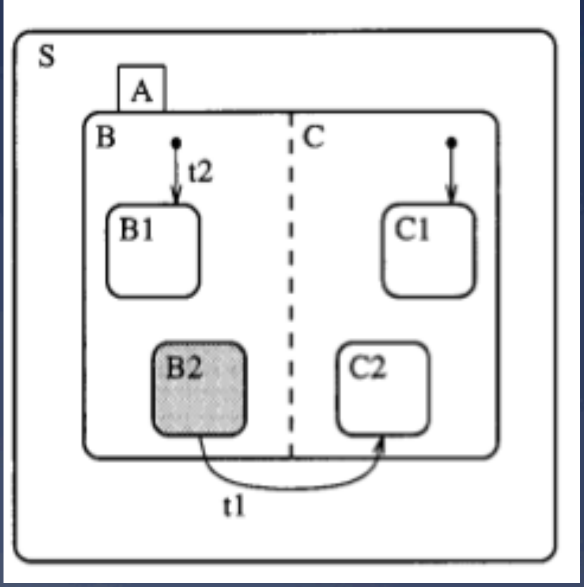
# Scope of Transitions

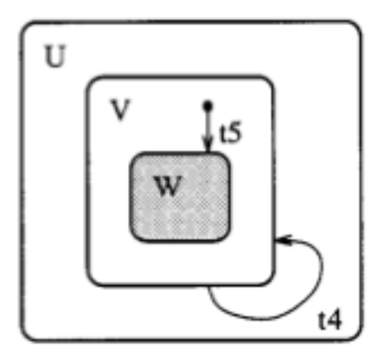
- For example, the scope of t1 is S
- Execution of t1 implies
  - Exiting B2, B, A, C, and C1 or C2
  - Entering A, B, B1, C, C2
- What about t4?

U

Exiting W and V

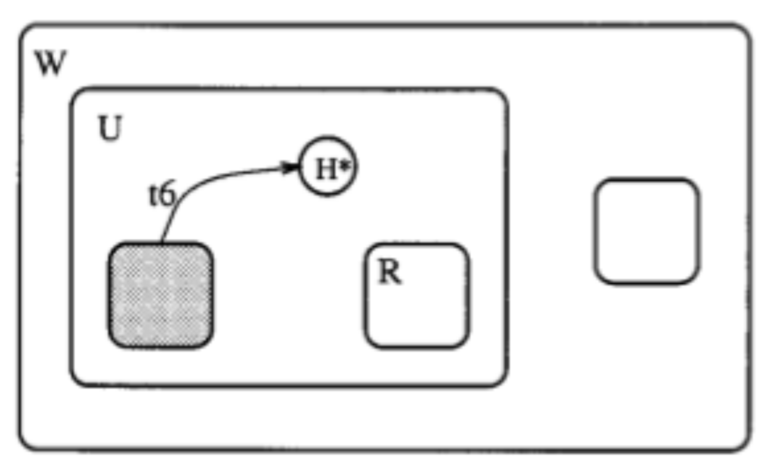
Entering V and W





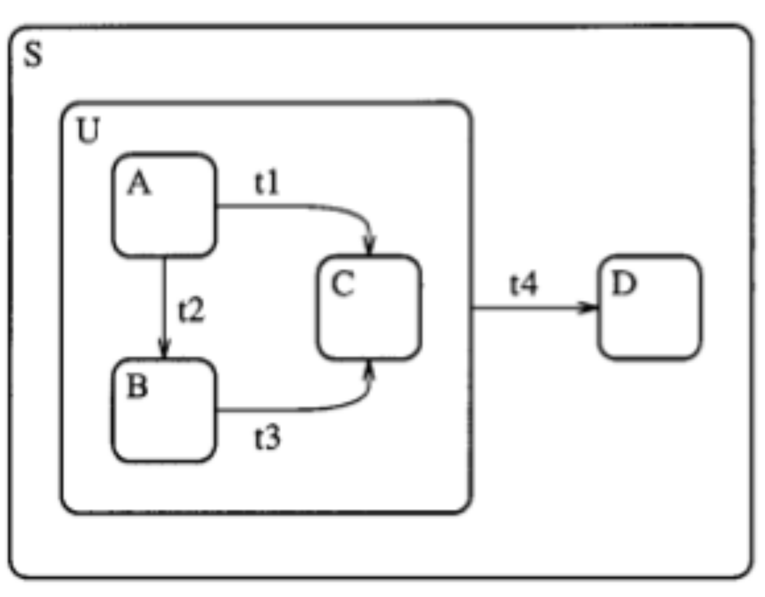
# Scope of Transitions

- What is the scope of t6? W



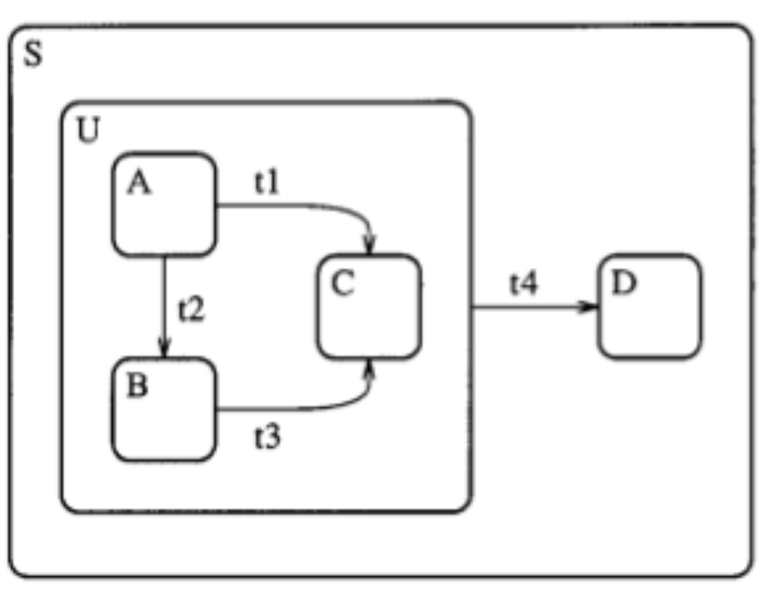
# Conflicting Transitions

- Two transitions are conflicting if there is some common state that would be exited if any one of them were to be taken
- Transitions t1 and t2 are conflicting
- Also, t4 is in conflict with t1, t2 and t3, why?



# Conflicting Transitions

- Non-determinism: there is no reason to take t1 over t2 or vice versa
- However, in the second case, t4 has priority over t1, t2 and t3
- The transition with the highest scope has priority
- If same scope a Non-determinism occurs



# Conflicting Transitions

- Dealing with non-determinisms
  - Simulation Tool waits for one of the possibilities to be selected by the user
  - Dynamic test tool will try all possibilities
  - The code synthesized by the software generator will select the first possibility
  - The hardware code generator behaves similarly, but can report non-determinisms

# Summary

- Introduction
- The Basics
- System Reactions
- Compound Transitions
- History
- Scope of Transitions
- Conflicting Transitions

# Next Time

- Jonathan Kotker will present the remainder of the article
  - Basic Step Algorithm
  - Models of Time
  - Racing Conditions
  - Multiple State Charts