Event-Based Architecture Definition Language

- Luckham/Vera
- Abstract
  - requirements for architecture description languages
  - Subset of Rapide which satisfies requirements
  - Rapide: modelling the architecture of software and hardware
  - causal event simulations

Testing Applications for Conformance to Reference Architectures

- X/Open Distributed Transaction Processing Architecture for UNIX, still evolving.
- Industry standard, promotes interoperability and easy integration of products from different vendors
- More than 750 products are branded (conformance tested). Rapide simplifies branding.

Event-Based Architecture Definition Language

- causal event simulations
  - events and time stamps
  - causal relationships
- Rapide architecture description features
  - event patterns
  - interfaces
  - architectures
  - event pattern mappings

Rapide

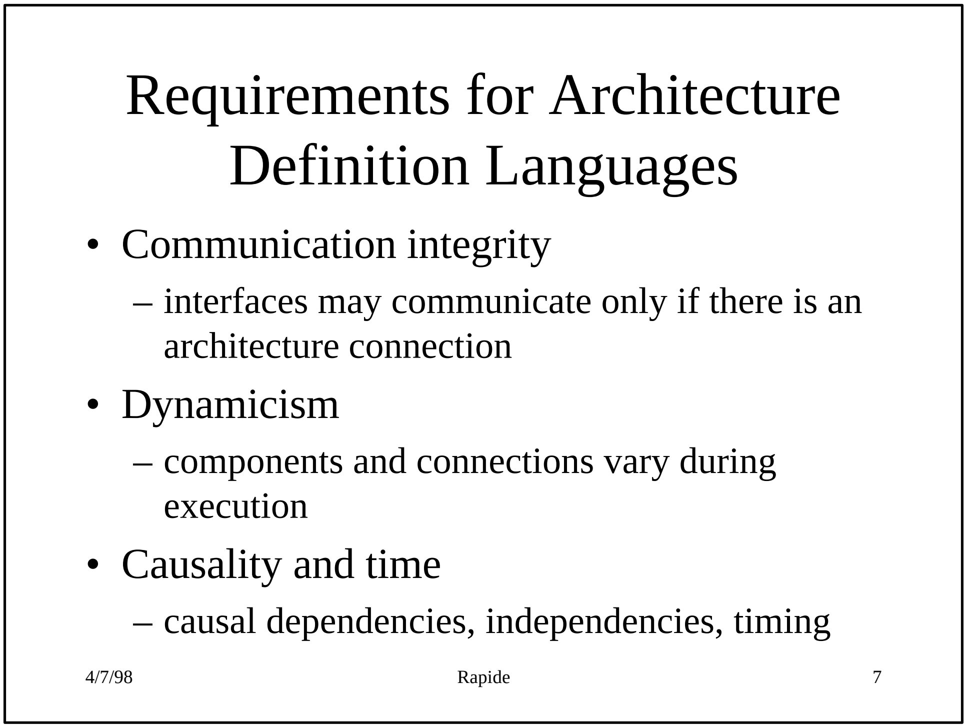
- Executable architecture definition language (EADL)
- Interface Connection Architecture
  - Interfaces: behavior of components
  - Connections: communication between components
  - Constraints: restrict behavior of interfaces and connections

Rapide

- When architecture is executed: causal event history which is automatically checked for conformance to constraints

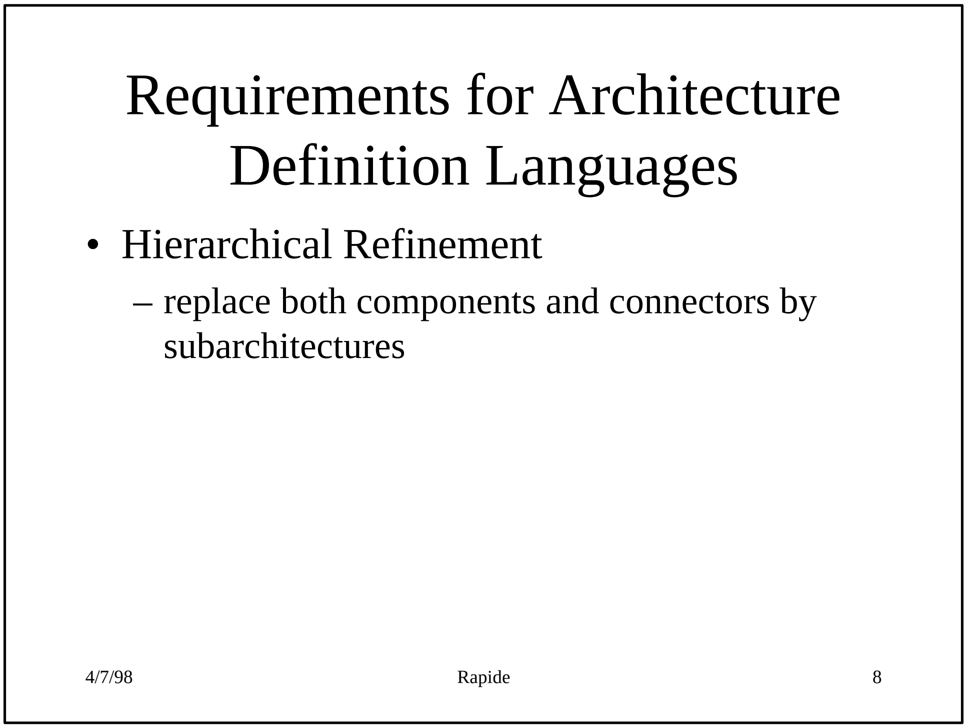
Requirements for Architecture Definition Languages

- Component abstraction: Interfaces:
  - – facilities
    - provided
    - required
  - – behavior allowing execution and analysis
- Communication abstraction
  - – connections
    - use only interfaces, allow execution and analysis



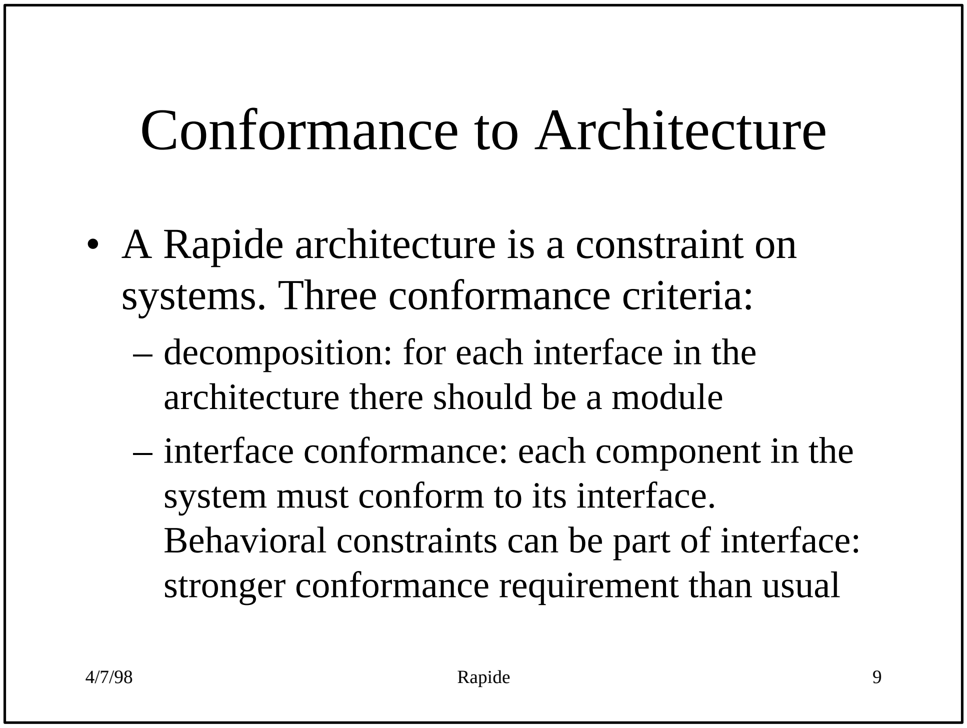
Requirements for Architecture Definition Languages

- Communication integrity
  - interfaces may communicate only if there is an architecture connection
- Dynamicism
  - components and connections vary during execution
- Causality and time
  - causal dependencies, independencies, timing



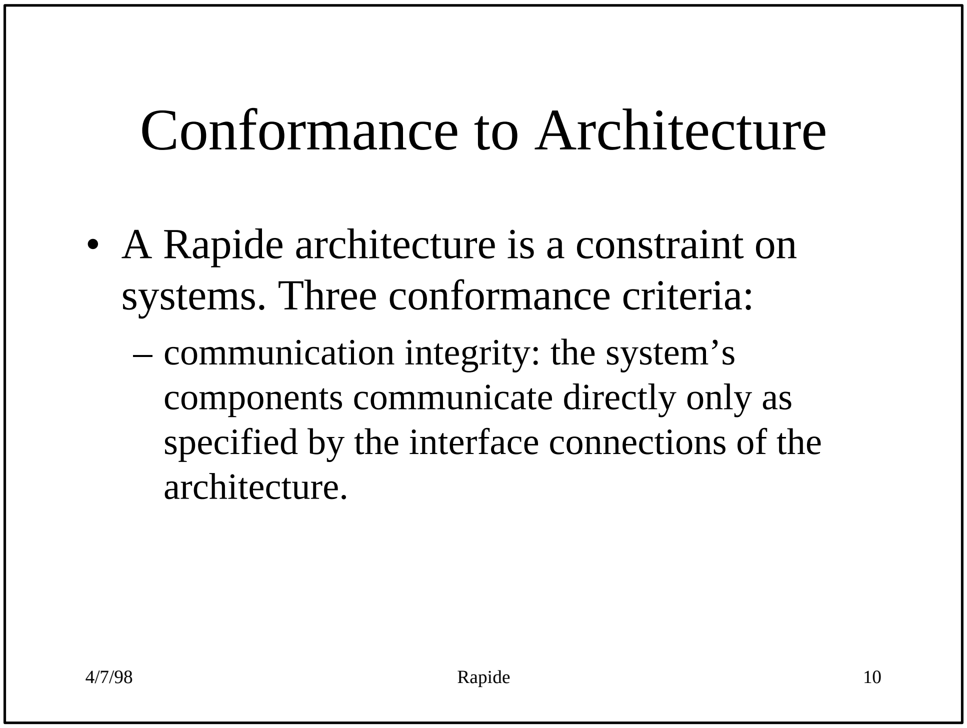
Requirements for Architecture Definition Languages

- Hierarchical Refinement
  - replace both components and connectors by subarchitectures



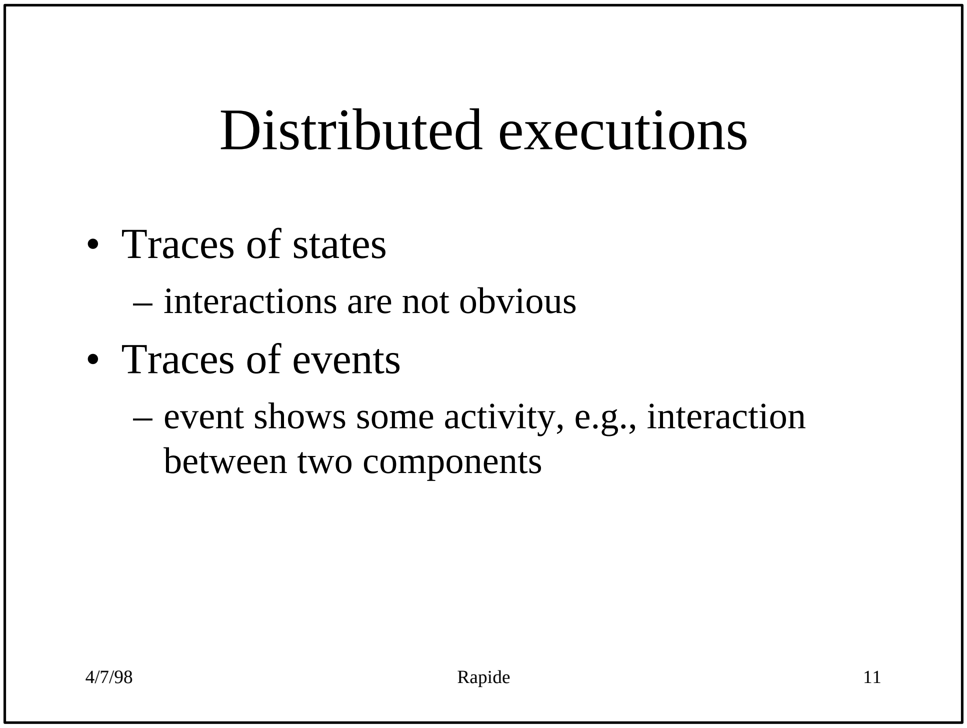
Conformance to Architecture

- A Rapide architecture is a constraint on systems. Three conformance criteria:
  - – decomposition: for each interface in the architecture there should be a module
  - – interface conformance: each component in the system must conform to its interface. Behavioral constraints can be part of interface: stronger conformance requirement than usual



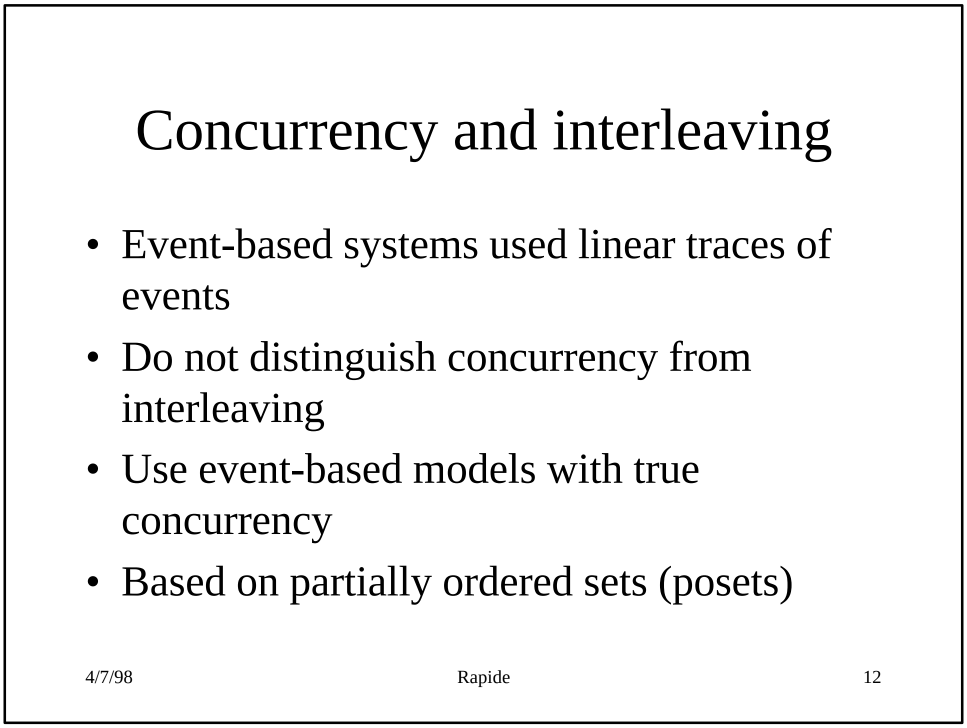
Conformance to Architecture

- A Rapide architecture is a constraint on systems. Three conformance criteria:
  - – communication integrity: the system’s components communicate directly only as specified by the interface connections of the architecture.



Distributed executions

- Traces of states
  - interactions are not obvious
- Traces of events
  - event shows some activity, e.g., interaction between two components



Concurrency and interleaving

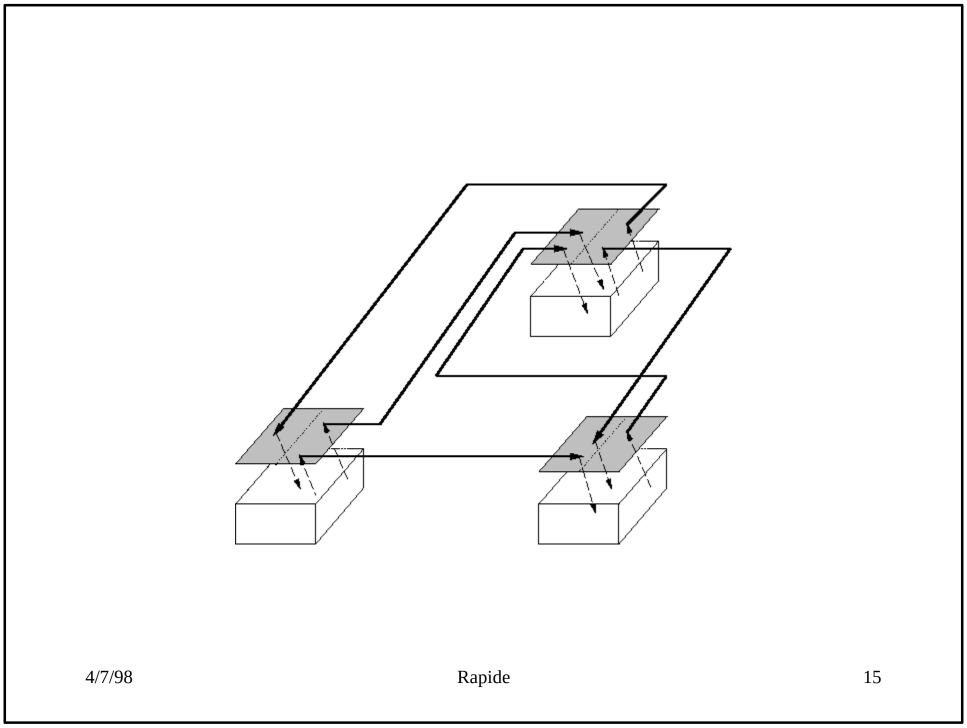
- Event-based systems used linear traces of events
- Do not distinguish concurrency from interleaving
- Use event-based models with true concurrency
- Based on partially ordered sets (posets)

Causality Partial Order

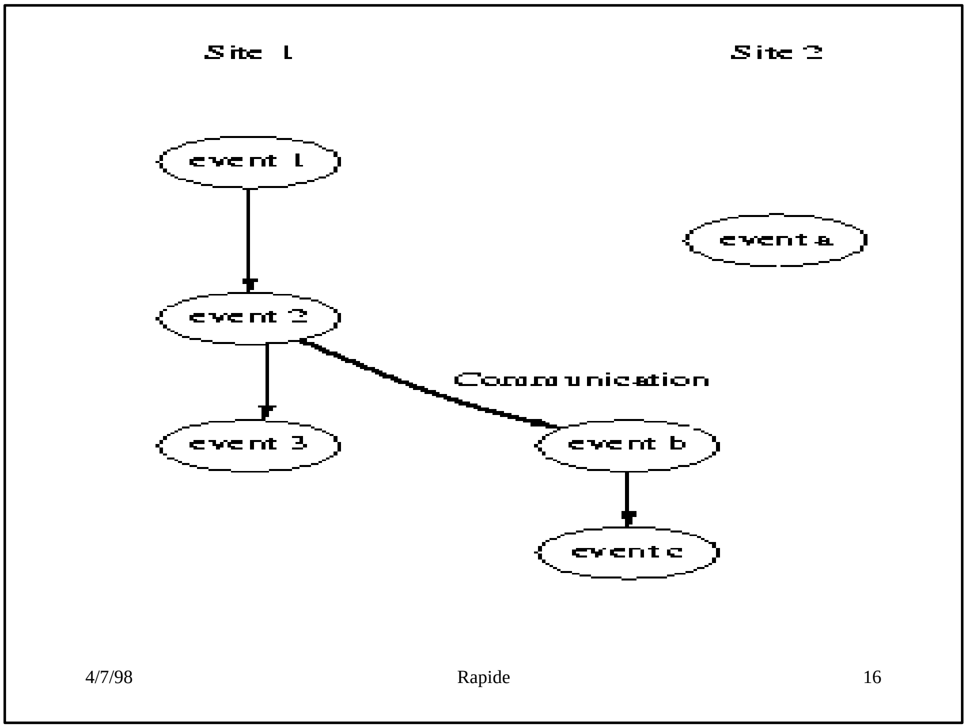
- An event is caused by another event if the first event could not have occurred without the occurrence of the second event.
- Independent events: did not cause each other.
- Concurrency is causal independence
- A partial order on a set S is an irreflexive, anti-symmetric and transitive relation « on the elements of S.

Causal dependency partial order

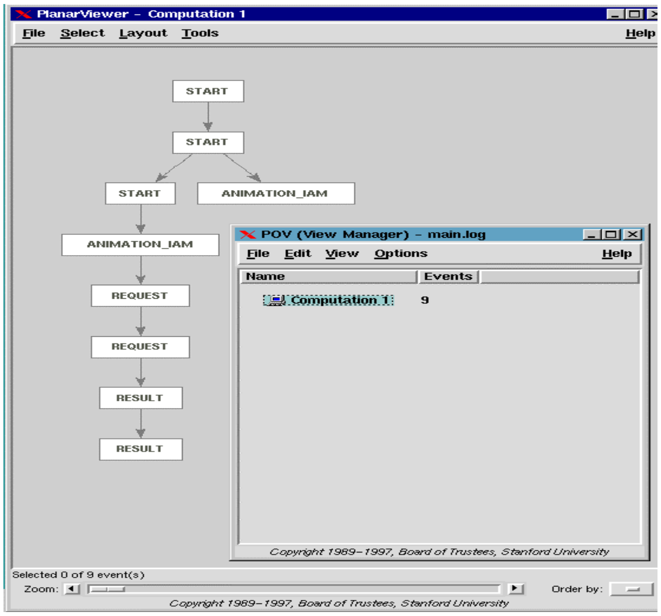
- An event B depends on an event A, written (A -> B) if and only if:
  – A and B are generated by the same process, and A is generated before B (total order per process).
  – A process is triggered by A and then generates B.
  – A process generates A and then assigns to a variable v. Another process reads v and then generates B.
  – Transitivity



4/7/98 Rapide 15



4/7/98 Rapide 16

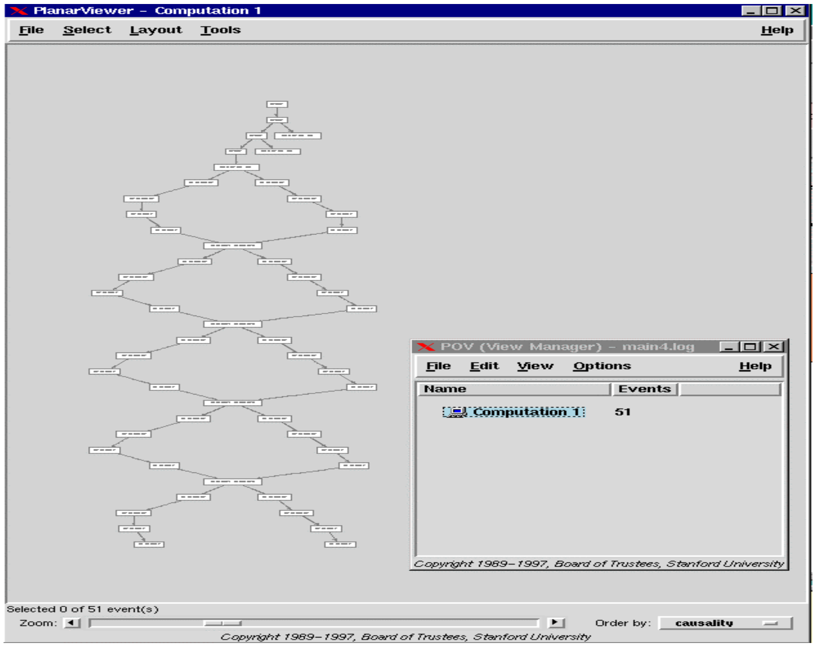

PlanarViewer – Computation 1
File Select Layout Tools Help
START
START
START
ANIMATION_IAM
ANIMATION_IAM
REQUEST
REQUEST
RESULT
RESULT
POV (View Manager) – main.log
File Edit View Options Help
Zoom:
Order by:

Name	Events
Computation 1	9

Copyright 1989–1997, Board of Trustees, Stanford University

Selected 0 of 9 event(s)

Copyright 1989–1997, Board of Trustees, Stanford University

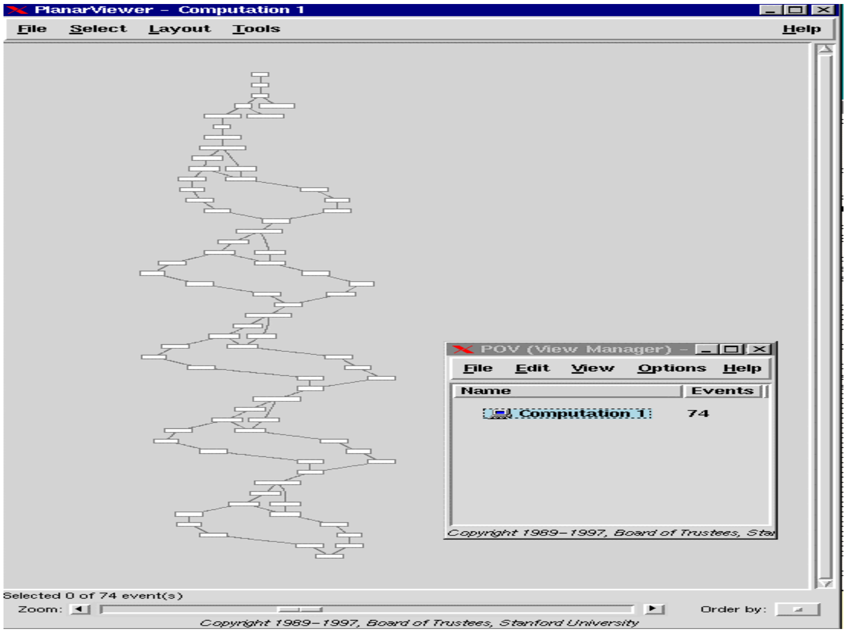

PlanarViewer – Computation 1
File Select Layout Tools Help
POV (View Manager) – main4.log
File Edit View Options Help
Zoom:
Order by: causality

Name	Events
Computation 1	51

Copyright 1989–1997, Board of Trustees, Stanford University

Selected 0 of 51 event(s)

Copyright 1989–1997, Board of Trustees, Stanford University


PlanarViewer – Computation 1
File Select Layout Tools Help
POV (View Manager) –
File Edit View Options Help
Name Events
Computation 1 74
Copyright 1989–1997, Board of Trustees, Sta
Selected 0 of 74 event(s)
Zoom:
Order by:
Copyright 1989–1997, Board of Trustees, Stanford University

Event Patterns

- Used in several ways:
  - to recognize (or trigger on) posets with particular features.
  - To specify constraints on posets, thus constraining behaviors.
  - To generate posets (for generating behaviors)
    - only restricted pattern can be used
  - connections, mappings between architectures

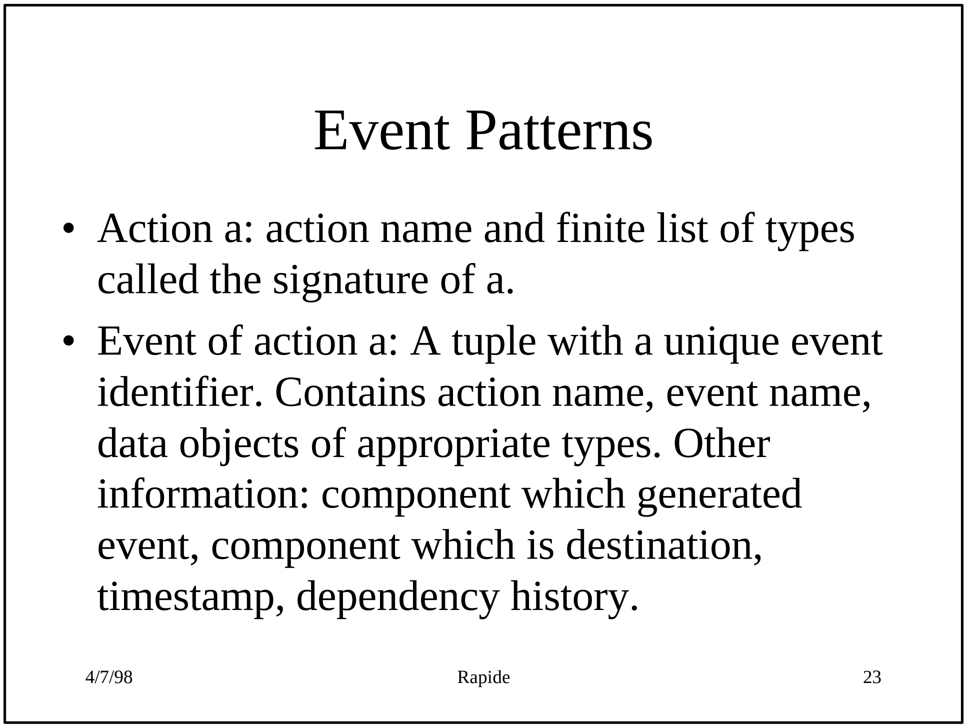
Event Patterns

- A generalization of regular expressions for event posets instead of for sequences of events.
- Note: posets are special kinds of graphs: why not consider a generalization of regular expressions for graphs?

?I place holder variable: bound by pattern matching

Event Patterns

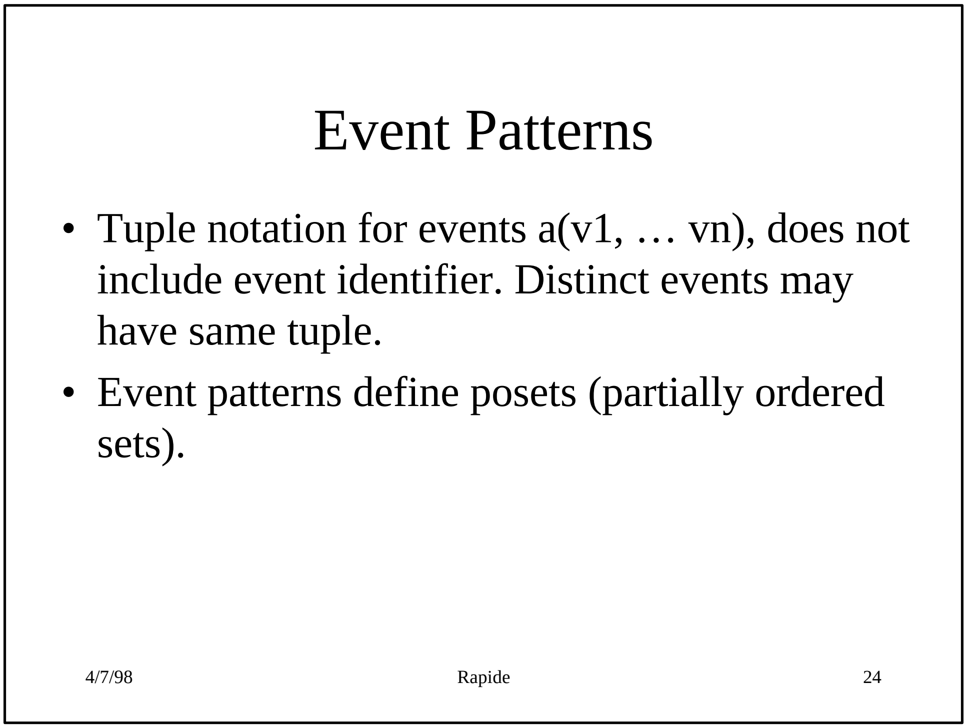
- Examples
  - A(?I) and B(?I) : An A event and a B event with the same parameter.
  - A -> B: An A event and a B event which depends on the A event.
  - A(?I) where ?I>4: An A event whose parameter is greater than 4.



Event Patterns

- Action a: action name and finite list of types called the signature of a.
- Event of action a: A tuple with a unique event identifier. Contains action name, event name, data objects of appropriate types. Other information: component which generated event, component which is destination, timestamp, dependency history.

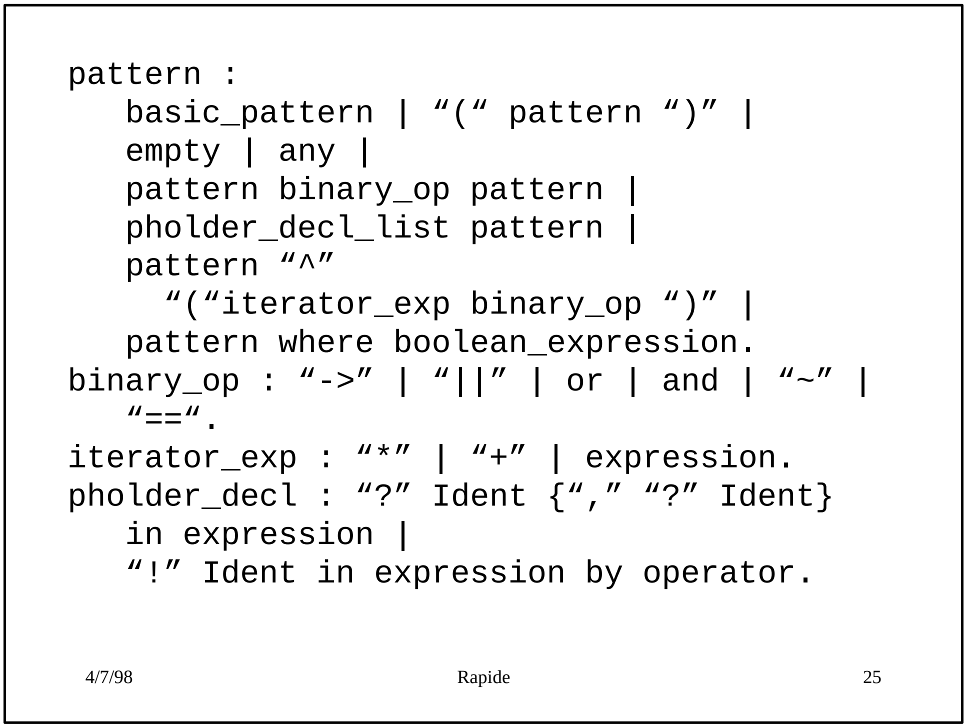
4/7/98 Rapide 23



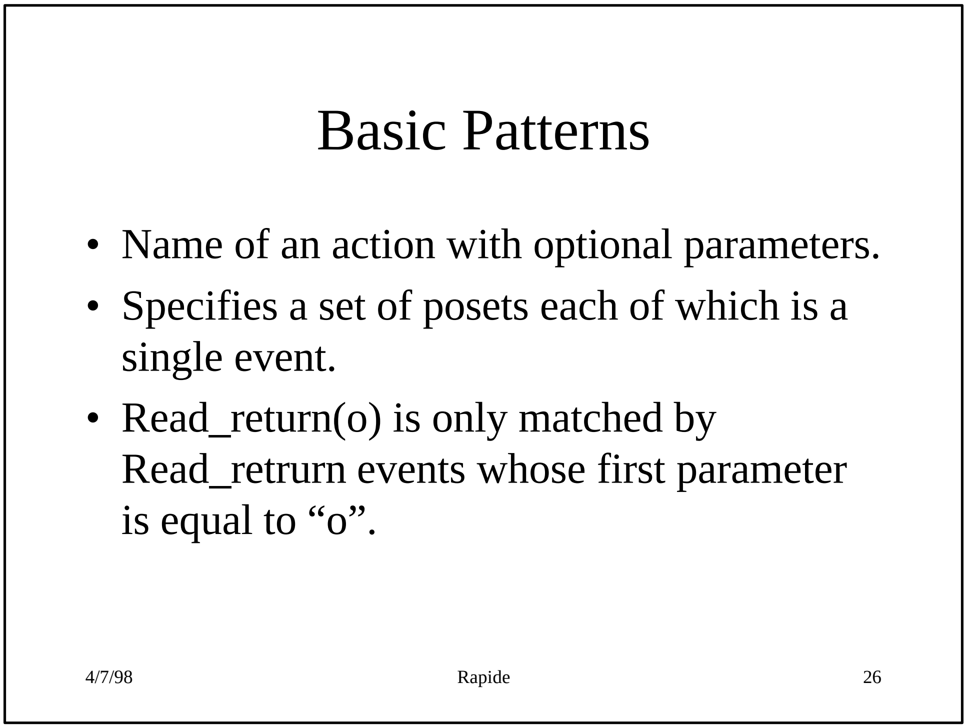
Event Patterns

- Tuple notation for events a(v1, … vn), does not include event identifier. Distinct events may have same tuple.
- Event patterns define posets (partially ordered sets).

4/7/98 Rapide 24



```
pattern :
   basic_pattern | “(“ pattern “)” |
   empty | any |
   pattern binary_op pattern |
   pholder_decl_list pattern |
   pattern “^”
     “(“iterator_exp binary_op “)” |
   pattern where boolean_expression.
binary_op : “->” | “||” | or | and | “~” |
   “==“.
iterator_exp : “*” | “+” | expression.
pholder_decl : “?” Ident {“,” “?” Ident}
   in expression |
   “!” Ident in expression by operator.
```



Basic Patterns

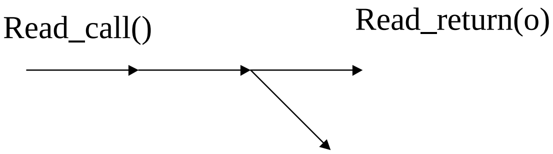
- Name of an action with optional parameters.
- Specifies a set of posets each of which is a single event.
- Read_return(o) is only matched by Read_retrurn events whose first parameter is equal to “o”.

Constants

- Two pattern constants: empty and any.
- Empty pattern is matched only by the empty poset.
- Any pattern is matched by any single event. That is, by any poset consisting of one event.

Composite Patterns

- Dependent: `P->P'.` A match of patterns P and P' where all of the events that matched P' depend on all the events that matched P.
- Read_call() ->Read_return(o)

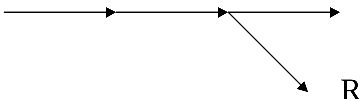


Read_call()

Read_return(o)

Composite Patterns

- Independent: `P || P'.` A match of patterns P and P’ where none of the events that matched P’ are dependent on any of the events that matched P and vice versa.
- Read_call() || Read_return(o)

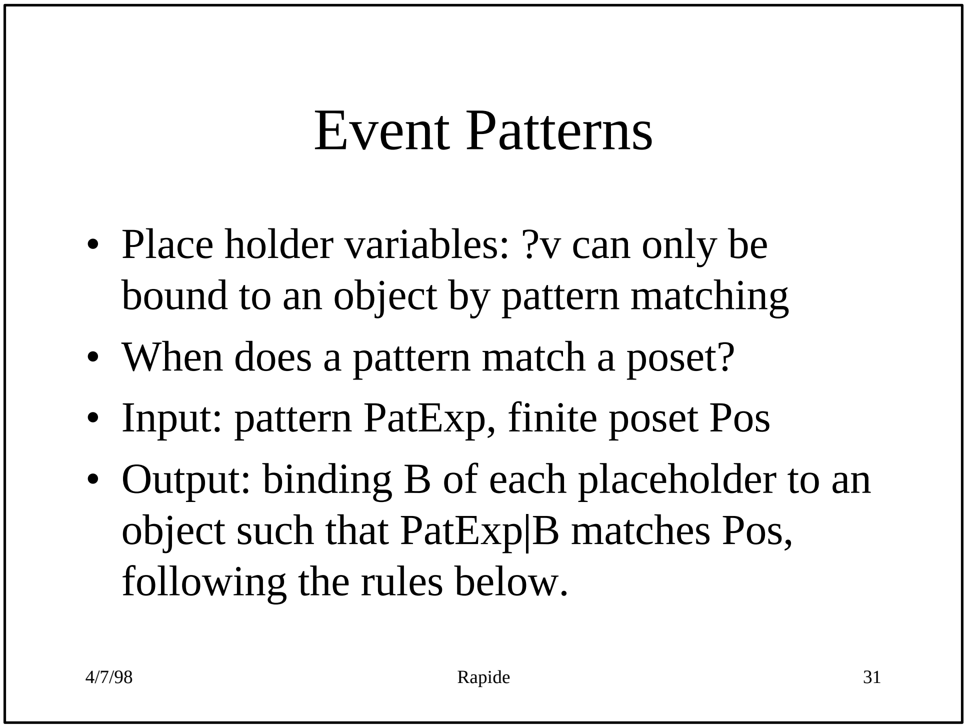


Read_return(o)

Read_call()

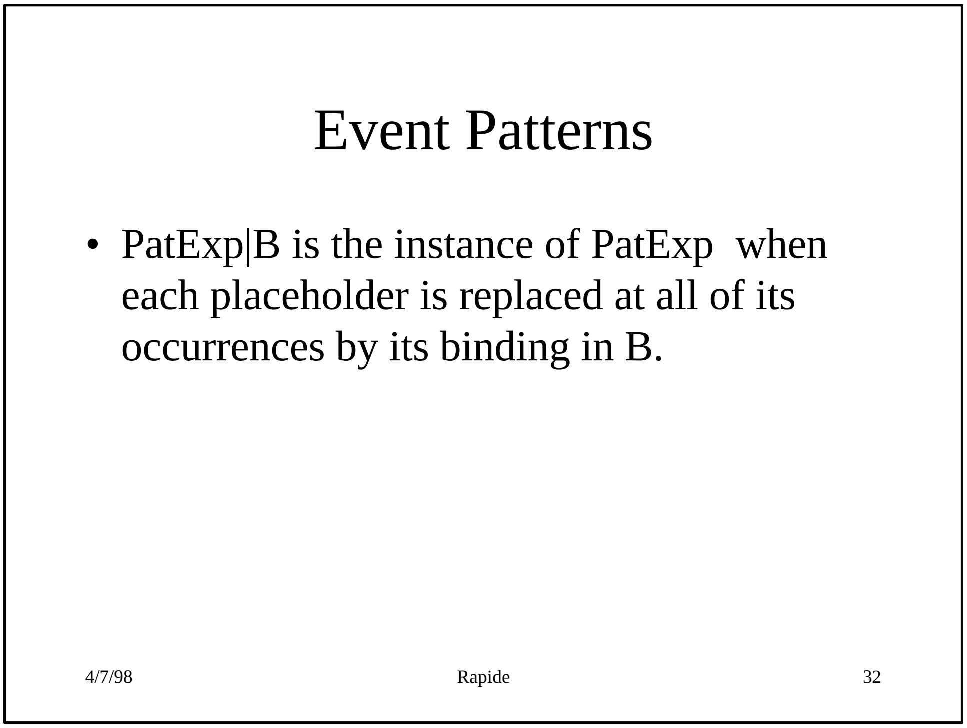
Composite Patterns

- Disjunction: `P or P'.` A match of patterns P or a match of pattern P’.



Event Patterns

- Place holder variables: ?v can only be bound to an object by pattern matching
- When does a pattern match a poset?
- Input: pattern PatExp, finite poset Pos
- Output: binding B of each placeholder to an object such that PatExp|B matches Pos, following the rules below.



Event Patterns

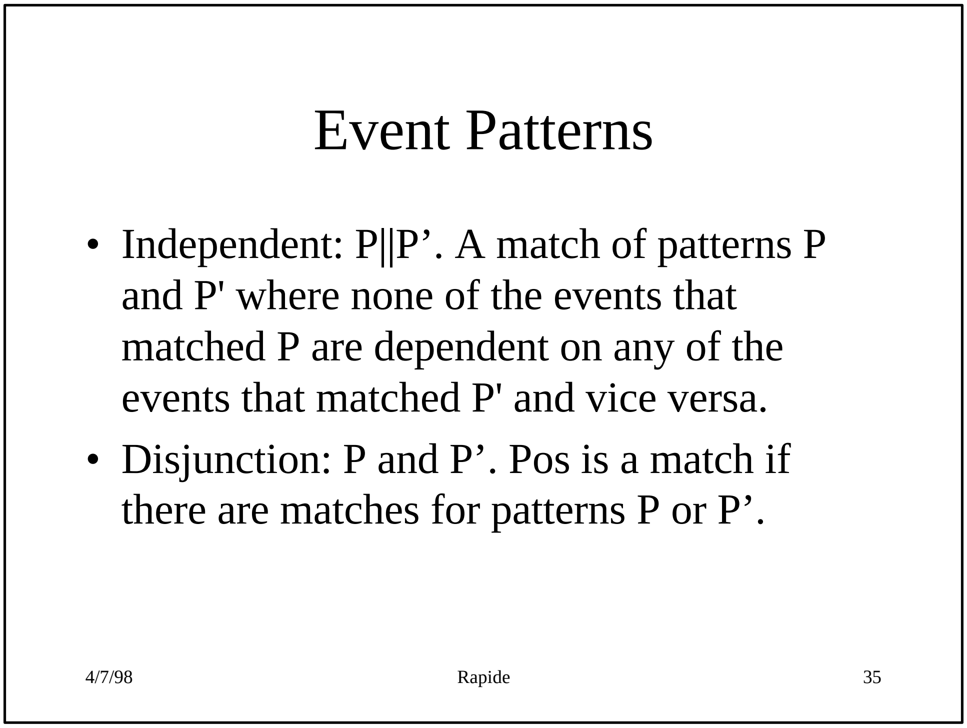
- PatExp|B is the instance of PatExp when each placeholder is replaced at all of its occurrences by its binding in B.

Event Patterns

- If PatExp is a basic pattern a(v1, …, vn), then B is chosen such that Pos = PatExp|B
- If PatExp is a composite pattern, built up from event patterns, P, P’ and a pattern operation, then Pos matches PatExp if Pos is the union of the matches for P and P’ under the same binding of placeholders B, as follows:

Event Patterns

- Dependent: P->P’. Pos is a match if all the events in the match for P’ depend on all the events in the match for P.
- Conjunction: P and P’. Pos is a match if there are matches for patterns P and P’.
- Disjoint Conjunction: P ~ P’. Pos is a match if all the events which matched P are distinct from the events which matched P’.



Event Patterns

- Independent: P||P’. A match of patterns P and P' where none of the events that matched P are dependent on any of the events that matched P' and vice versa.
- Disjunction: P and P’. Pos is a match if there are matches for patterns P or P’.