Comparing Workflow Specification Languages: A Matter of Views

Victor Vianu

UCSD/Webdam/INRIA/LSV

- Title: work with Serge and Pierre Bourhis

  ICDT 2011
- More on workflow views: inferring specifications

  preliminary ideas with Luc and Serge
- AXML as a query language

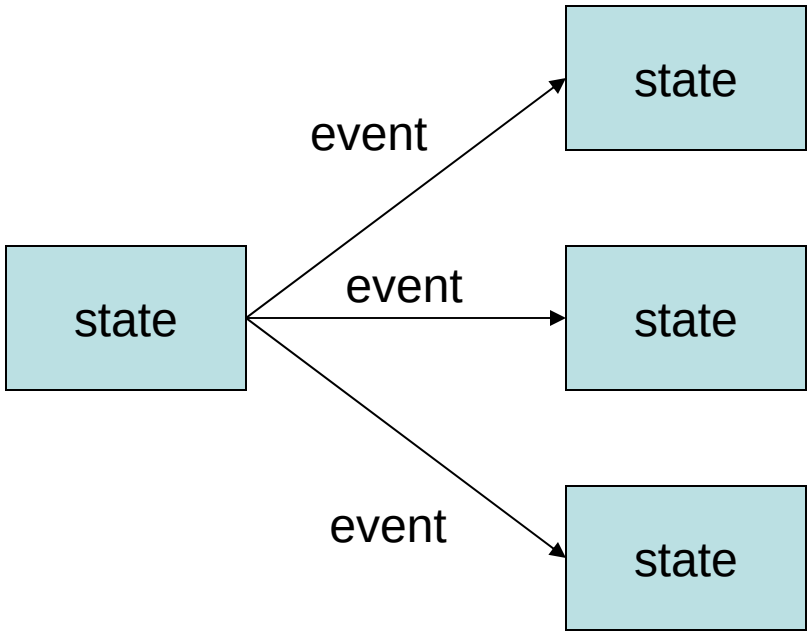
  ongoing work with Serge and Pierre Bourhis

Comparing Workflow Specification Languages: A Matter of Views

- Framework for comparing workflow specifications: based on views
- Specific results for AXML workflow specification mechanisms: guards, automata, temporal logic
- Comparison to IBM's Tuple Atifacts

What is a data-centric workflow?

- States (with data)
- Events (with data)
- Transitions



Specifications of data-centric workflows

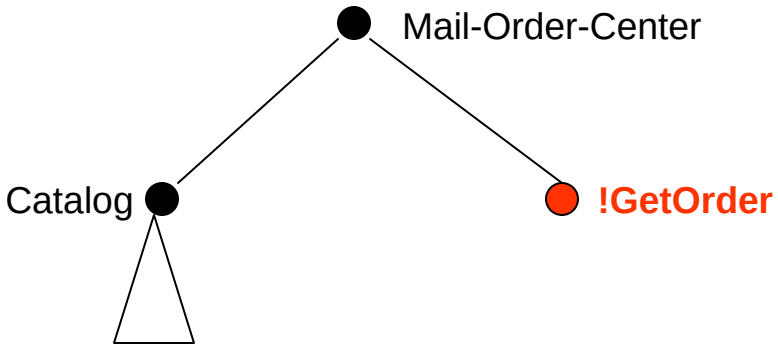
Two examples

- Active XML
- Tuple Artifacts (IBM)

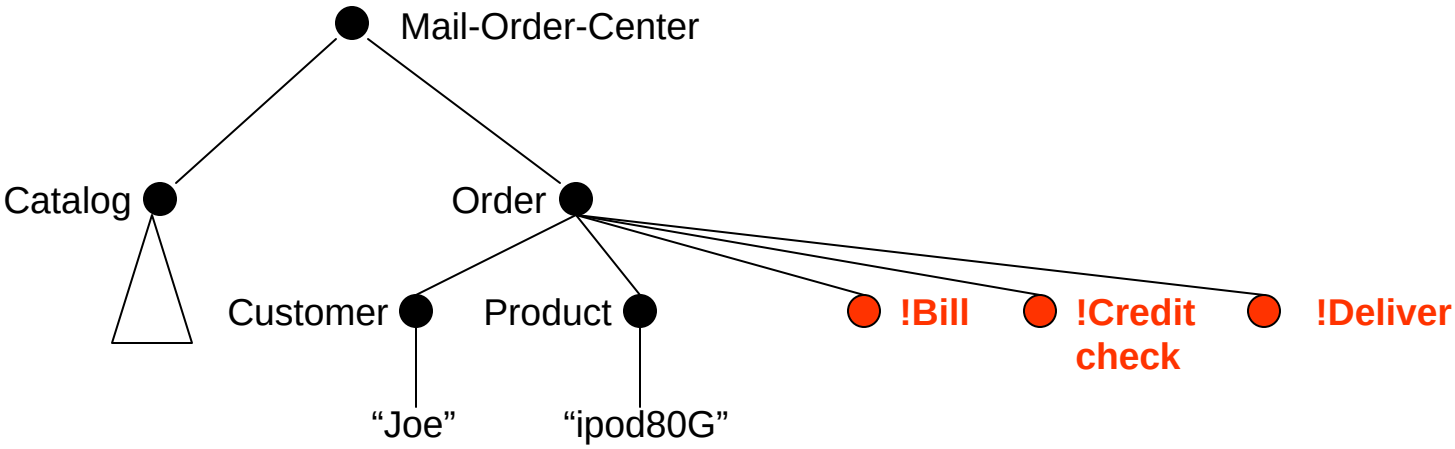
Workflow specification mechanisms for AXML

- **BAXML**: static constraints only
- **GAXML**: function calls and returns controlled by guards
- **AAXML**: allowed transitions described by an automaton
- **TAXML**: workflow constrained by temporal property of history

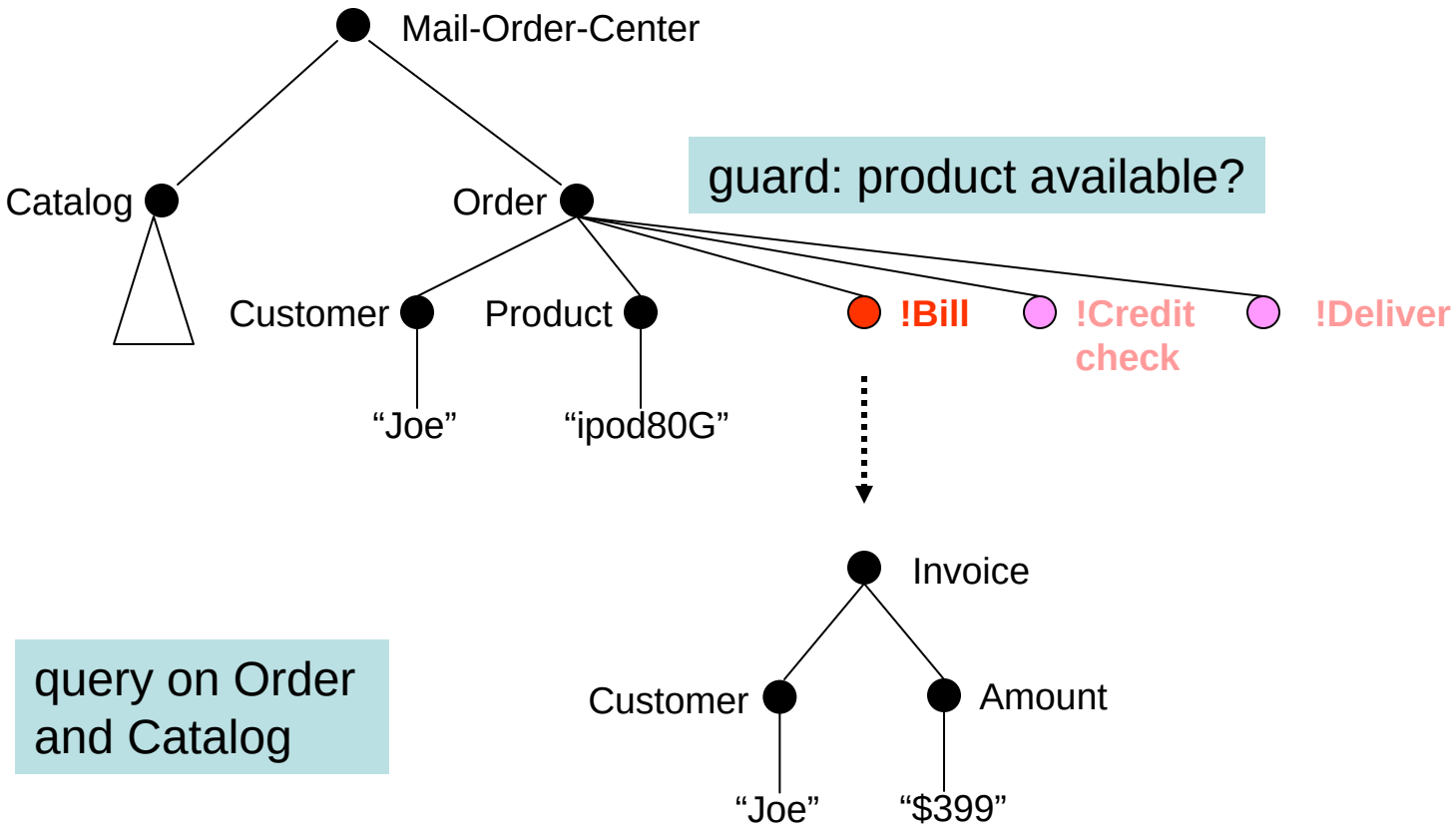
GAXML by example



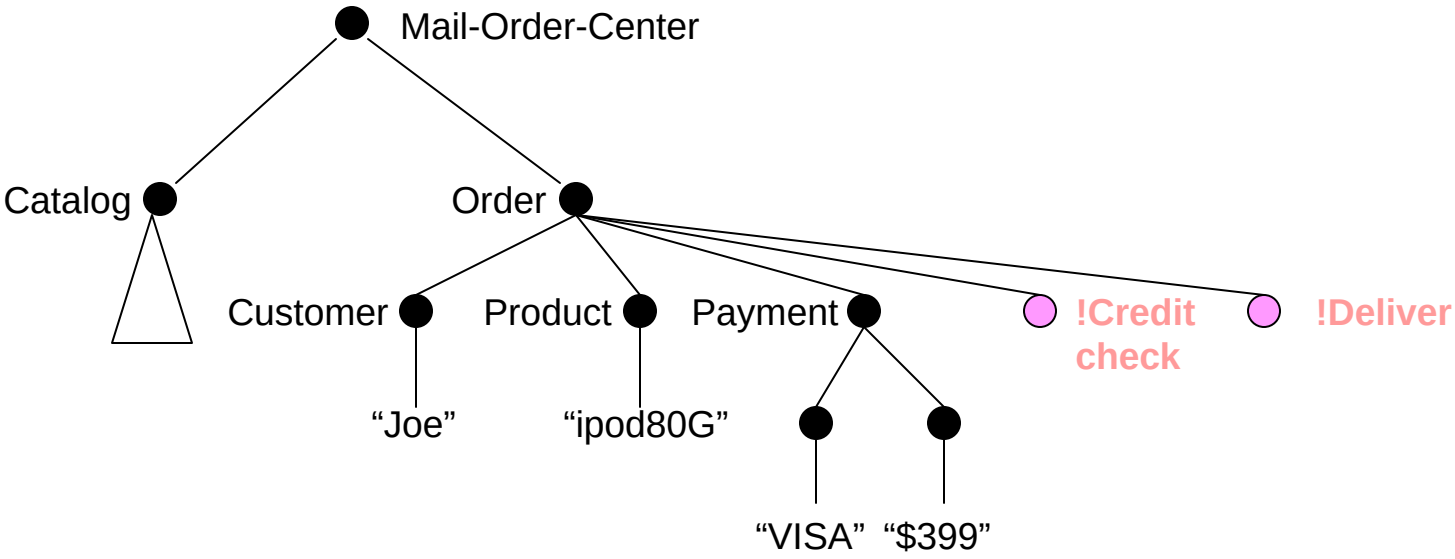
GAXML by example



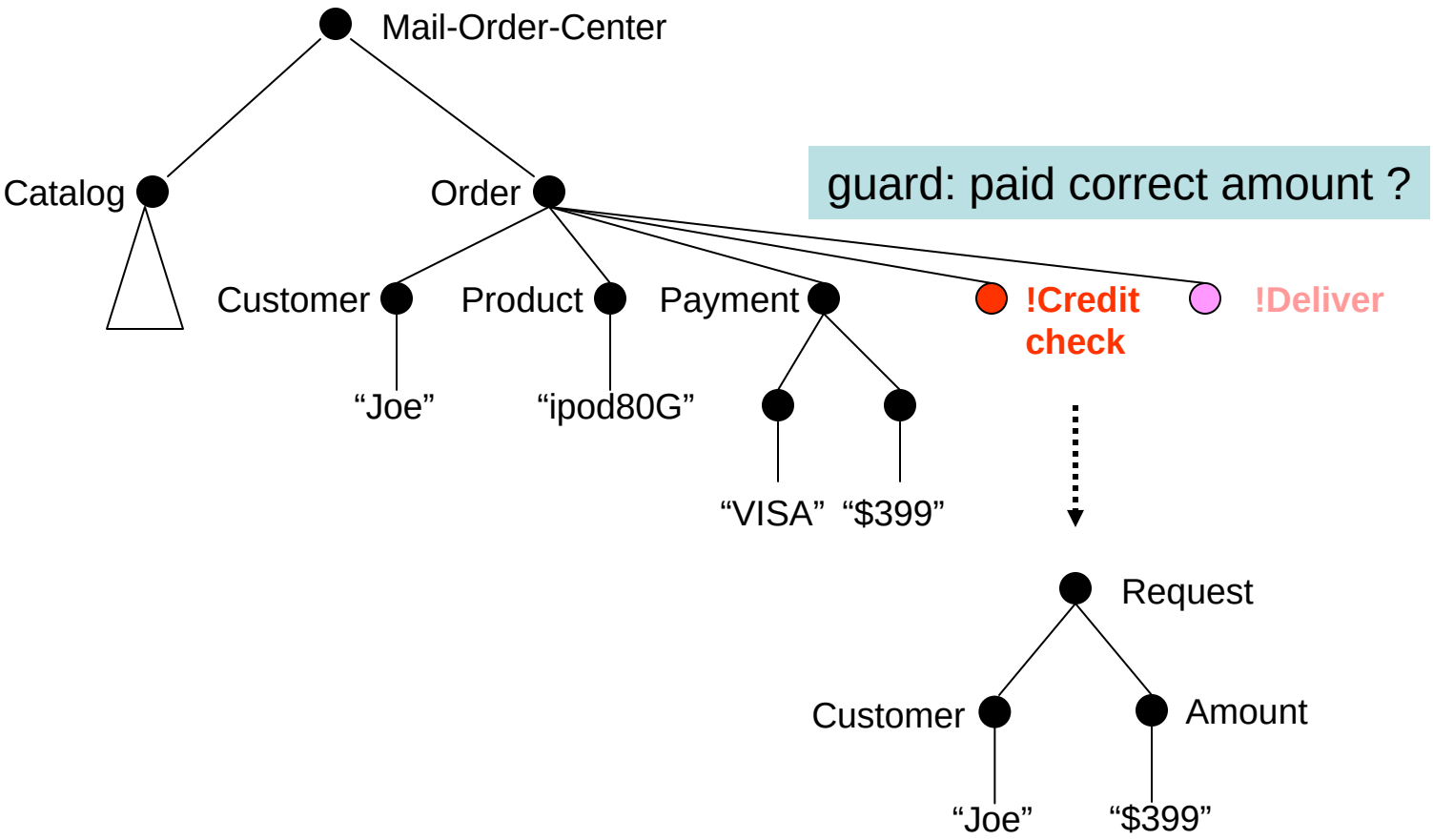
GAXML by example



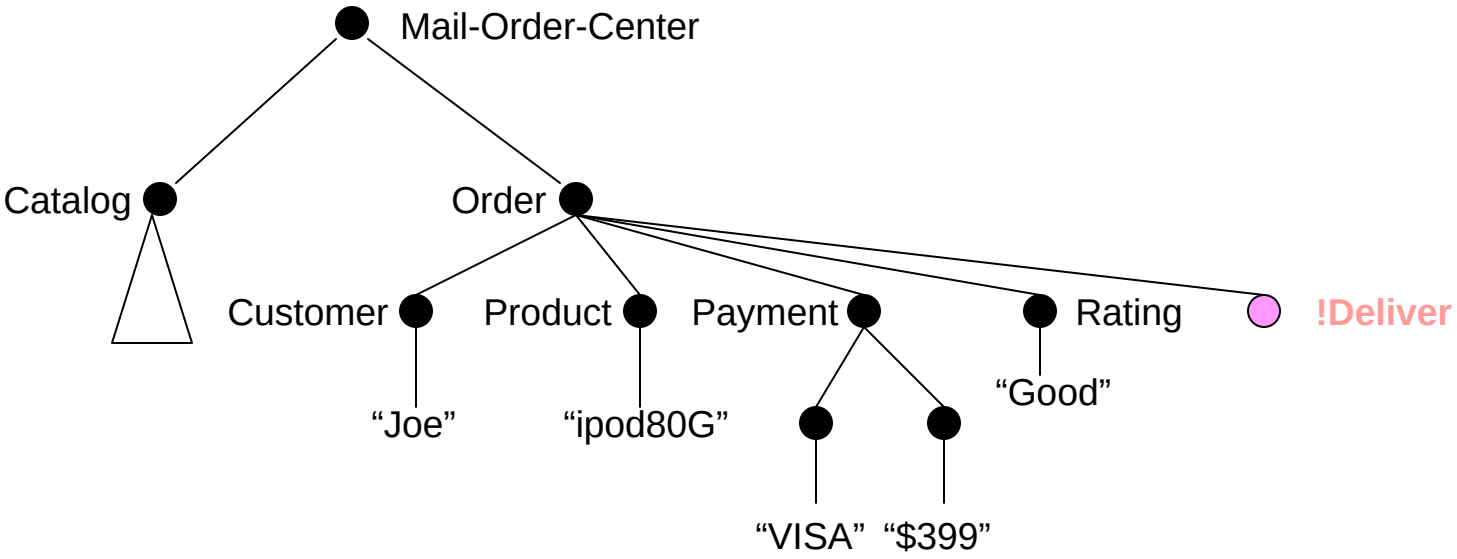
GAXML by example



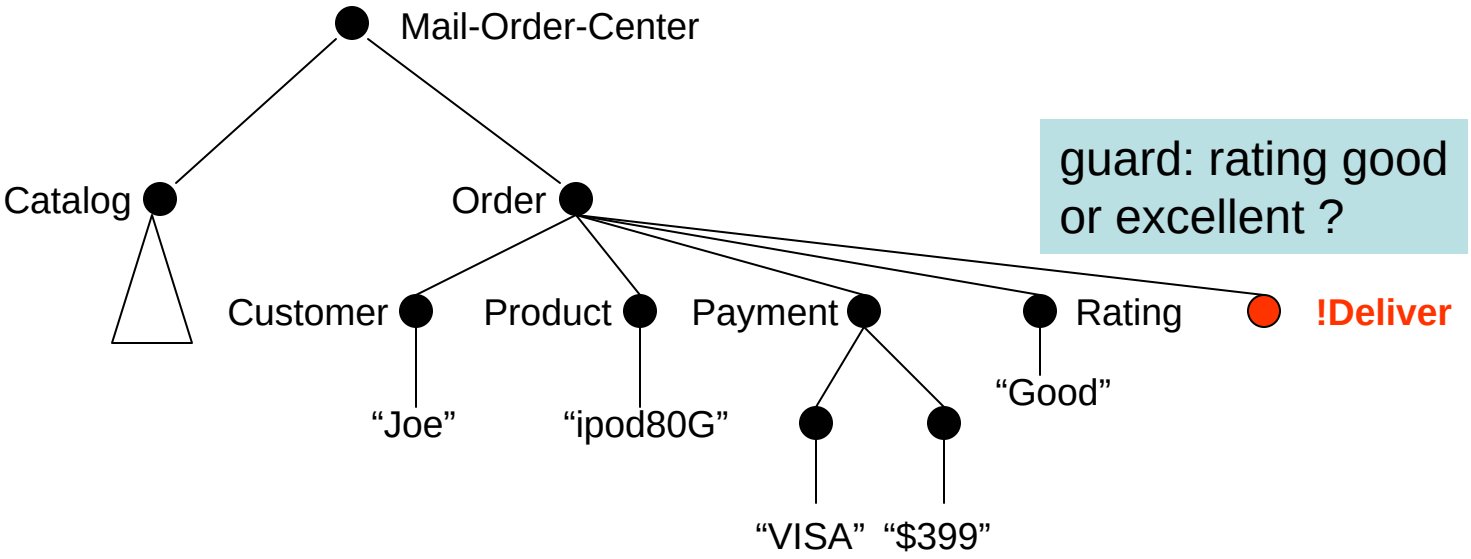
GAXML by example

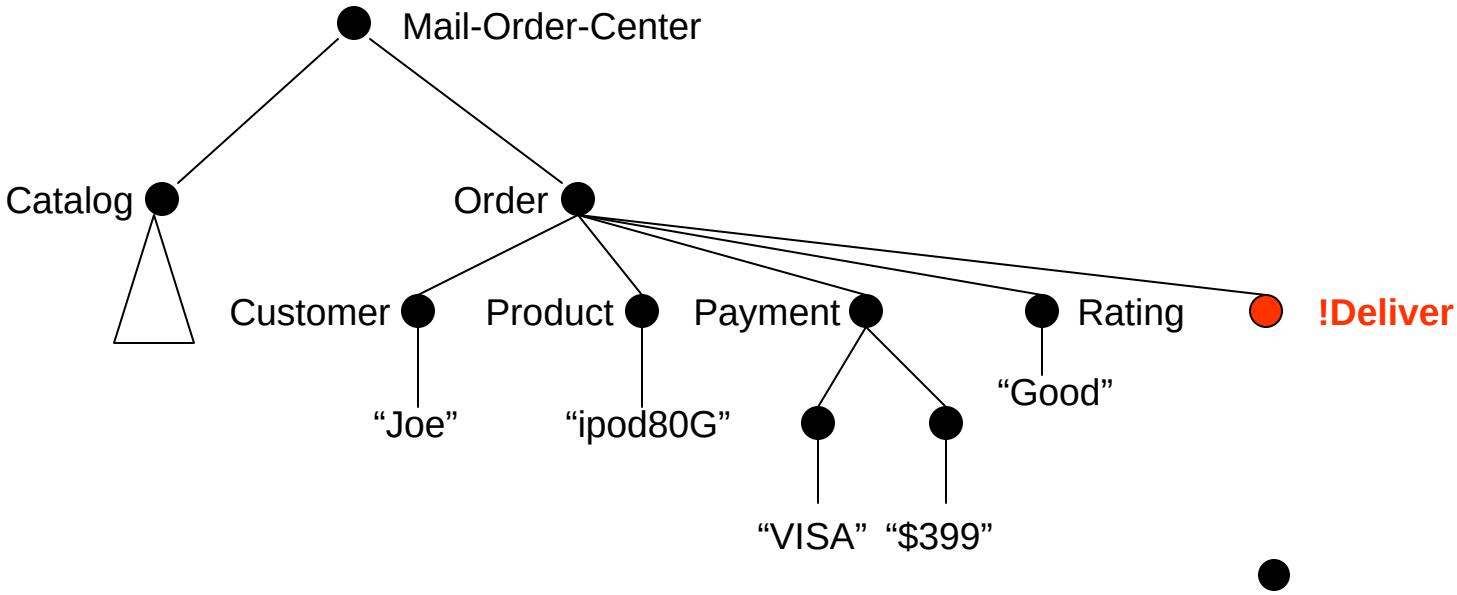


GAXML by example

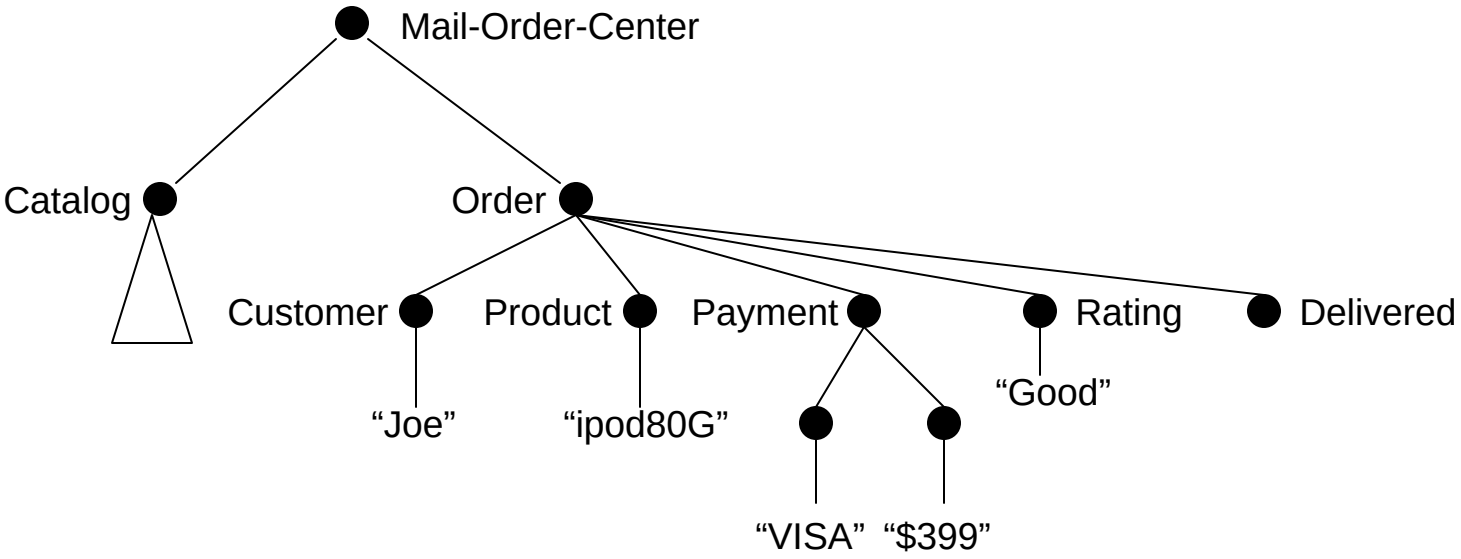


GAXML by example




Mail-Order-Center
Catalog
Order
Customer
Product
Payment
Rating
!Deliver
“Joe”
“ipod80G”
“Good”
“VISA”
“$399”

GAXML by example

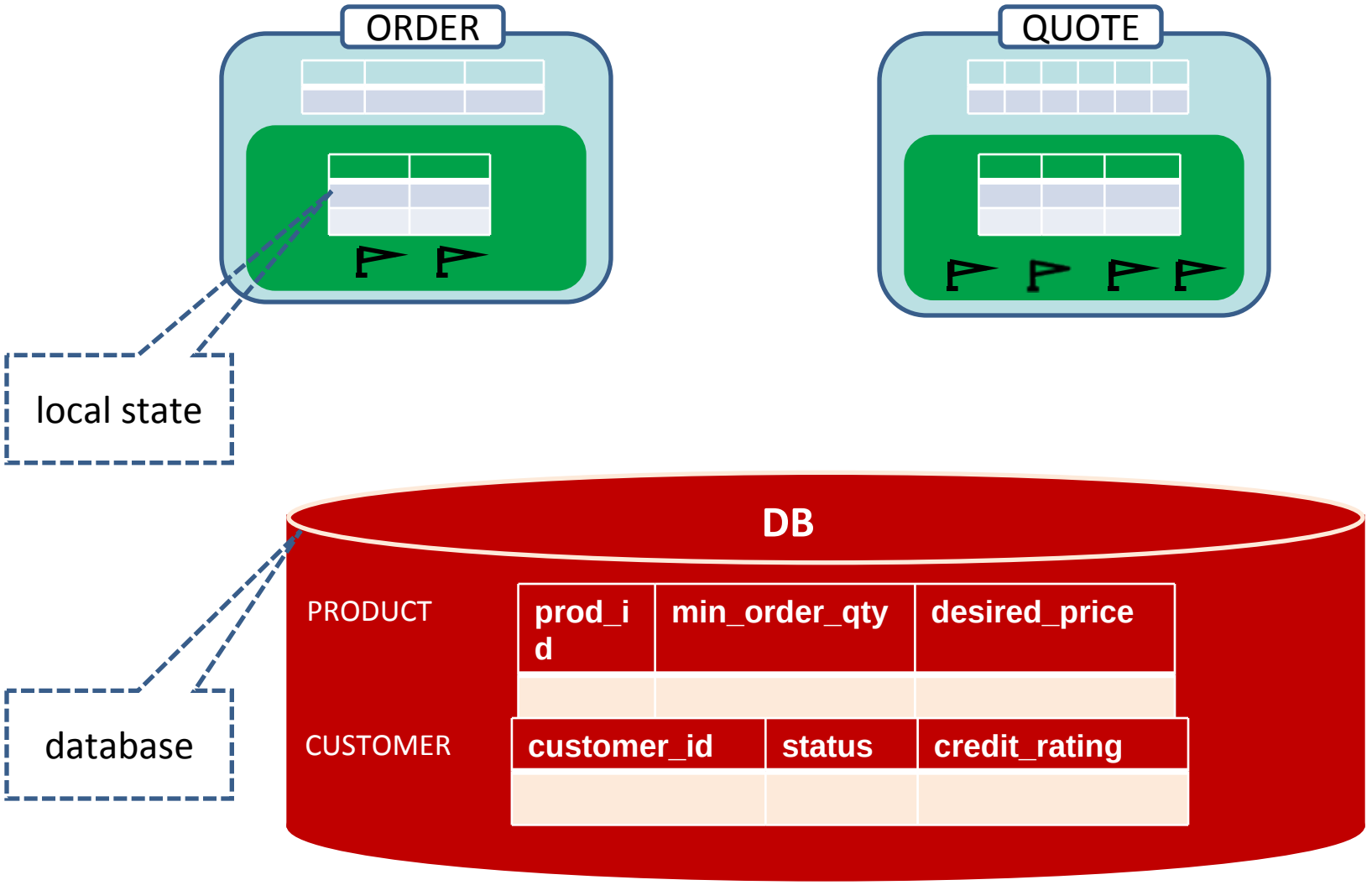


IBM's Tuple Artifacts

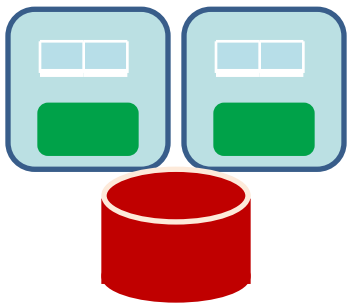
Tuple artifacts:

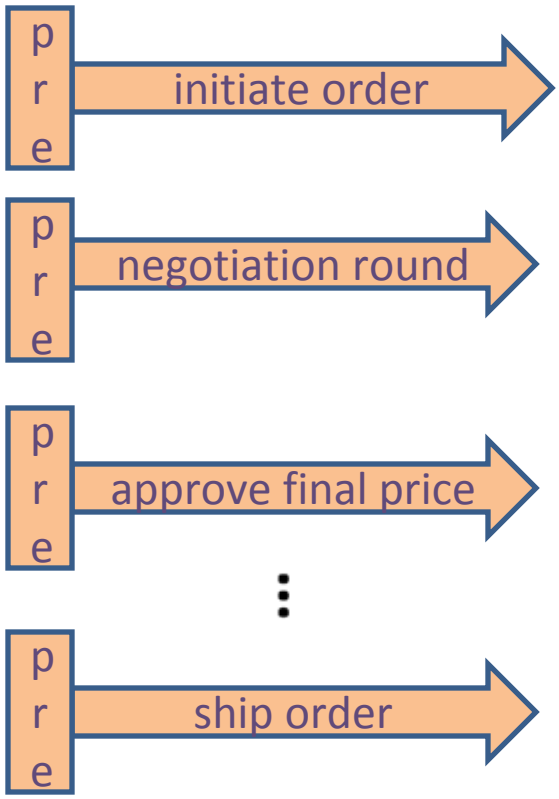
- evolving tuples of data values
- local states: evolving relations
- fixed underlying database
- services (pre/post conditions on
  tuple, local state, and a global relational database)

System with two artifacts



Events: services

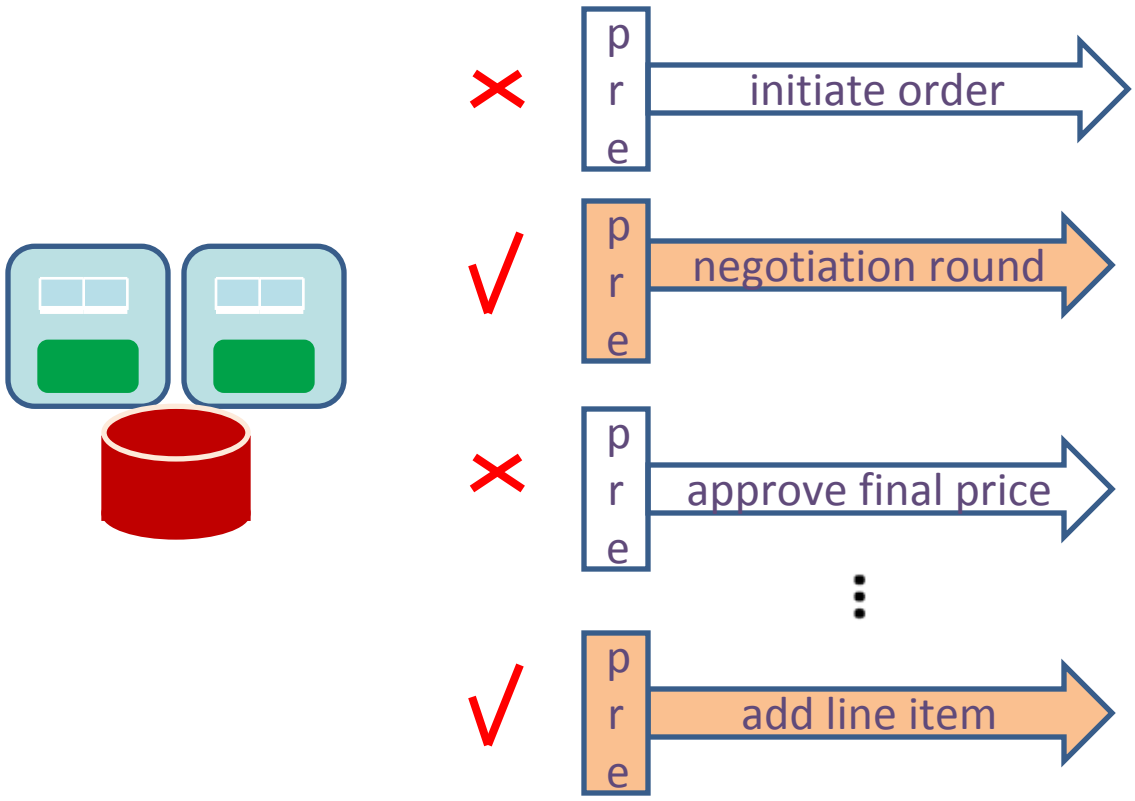




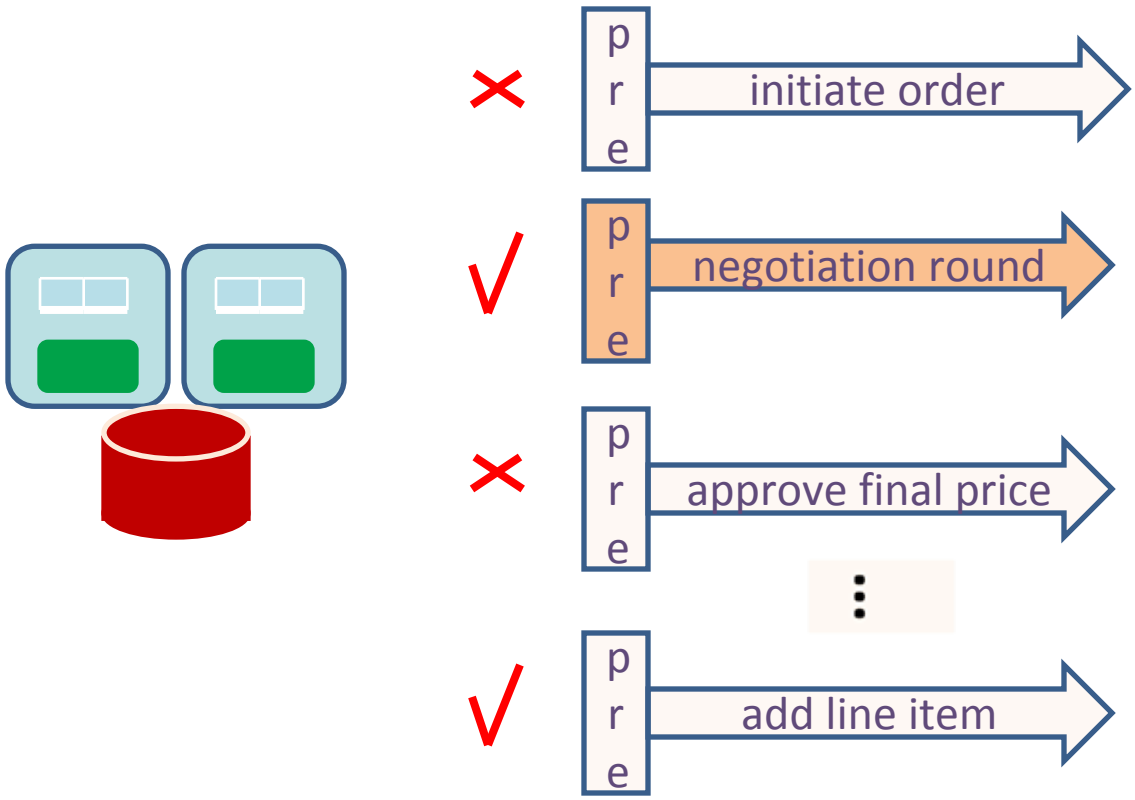
current snapshot of artifact system

available services

Services evaluate their pre-conditions in parallel



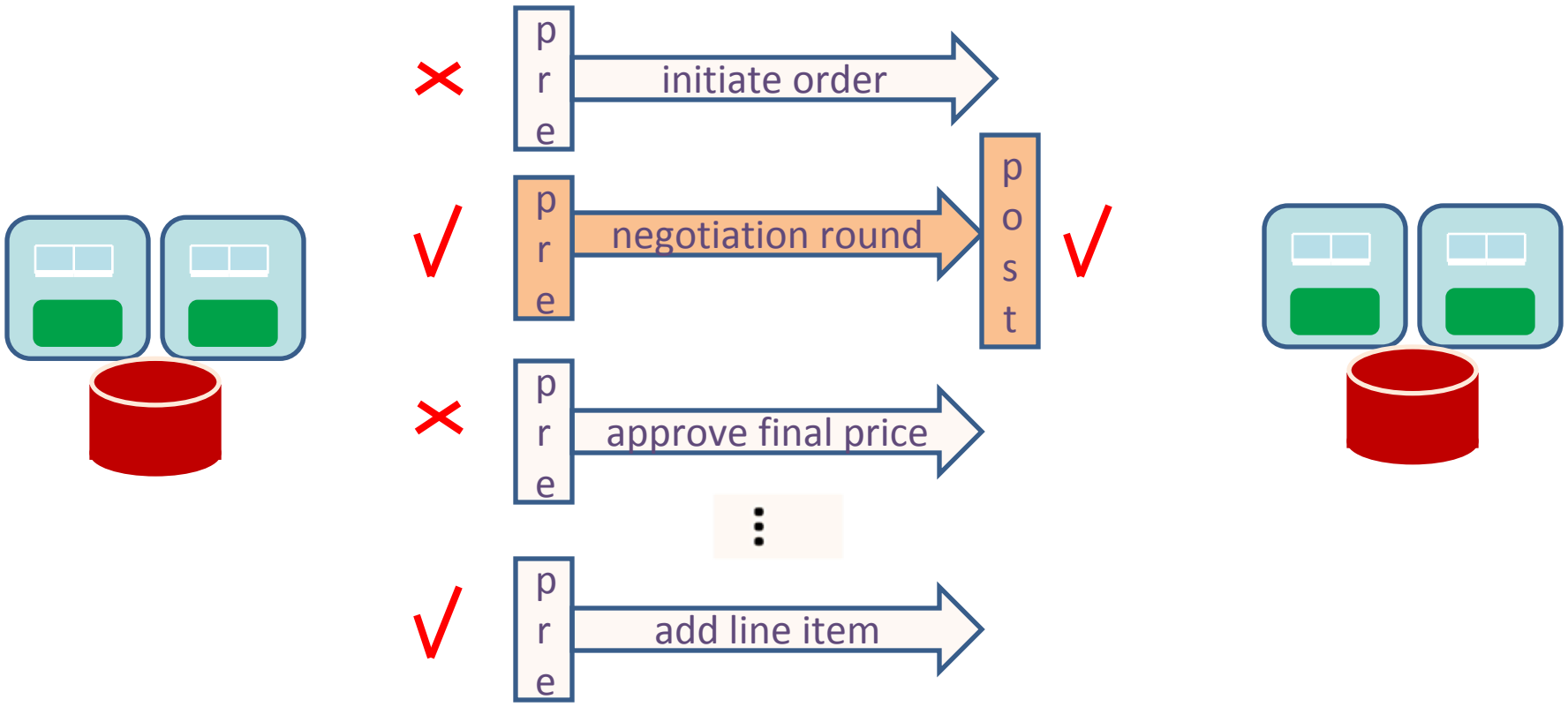
One qualifying service is non-deterministically picked for execution



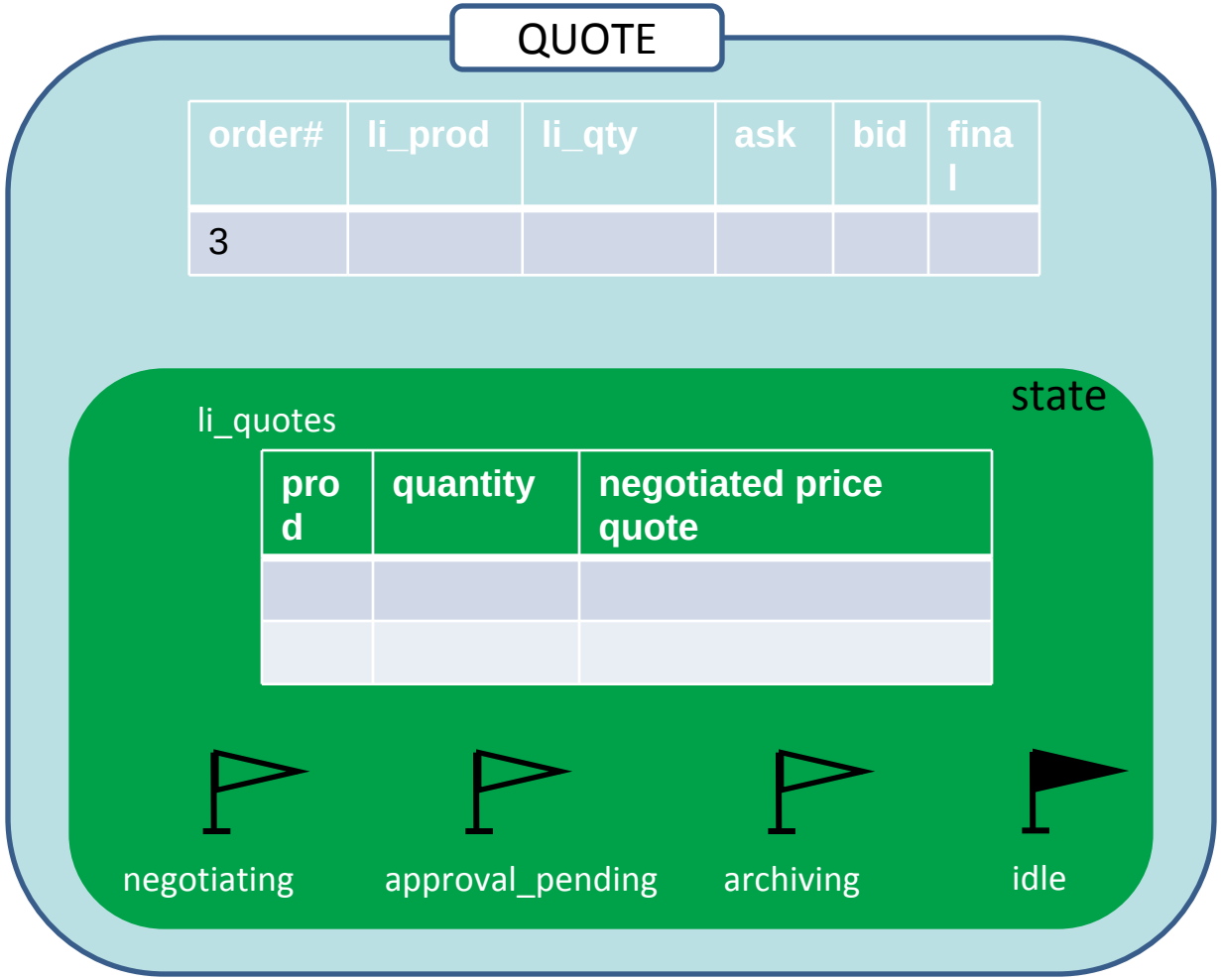
current snapshot of artifact system

available services

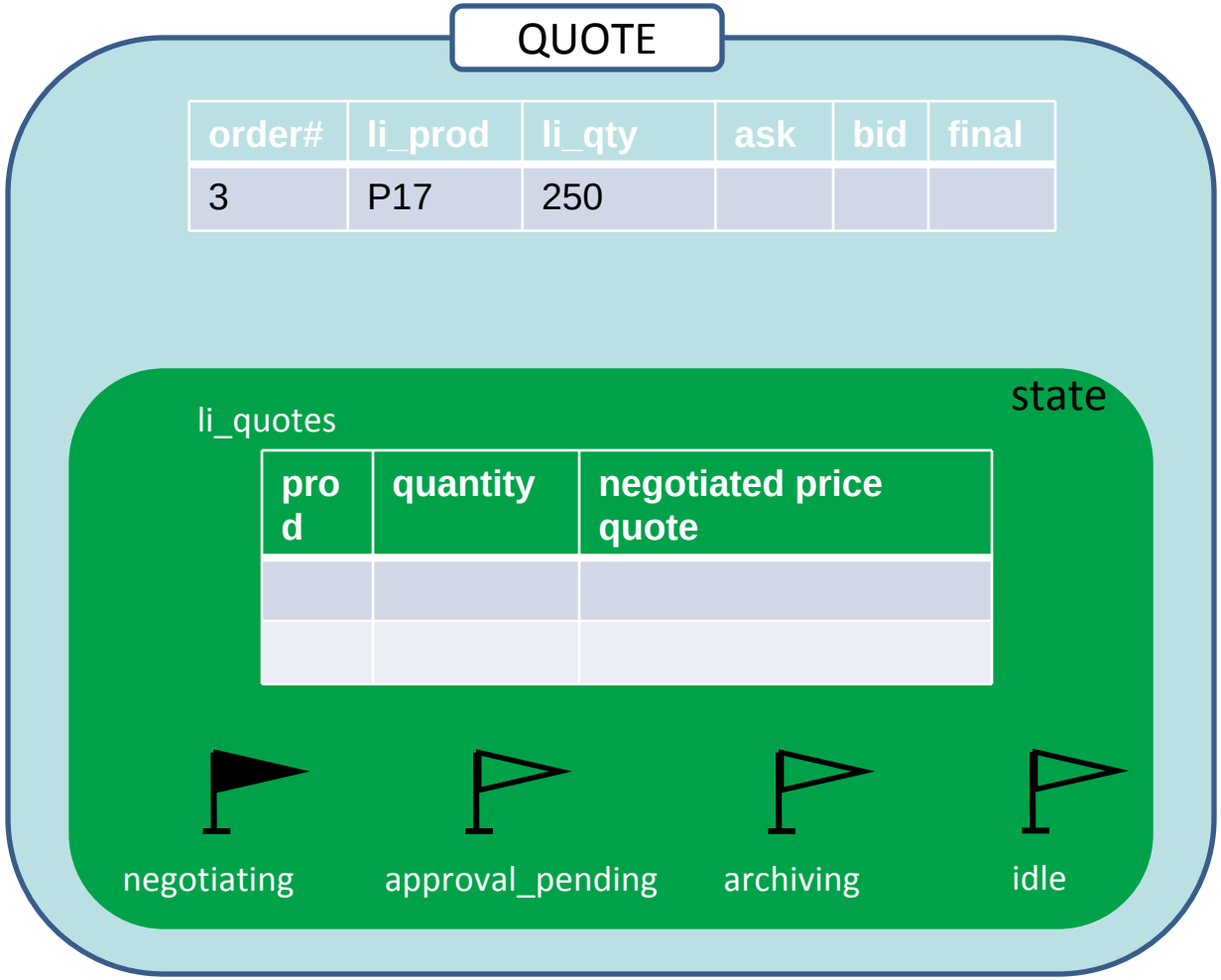
Post-condition Is Satisfied



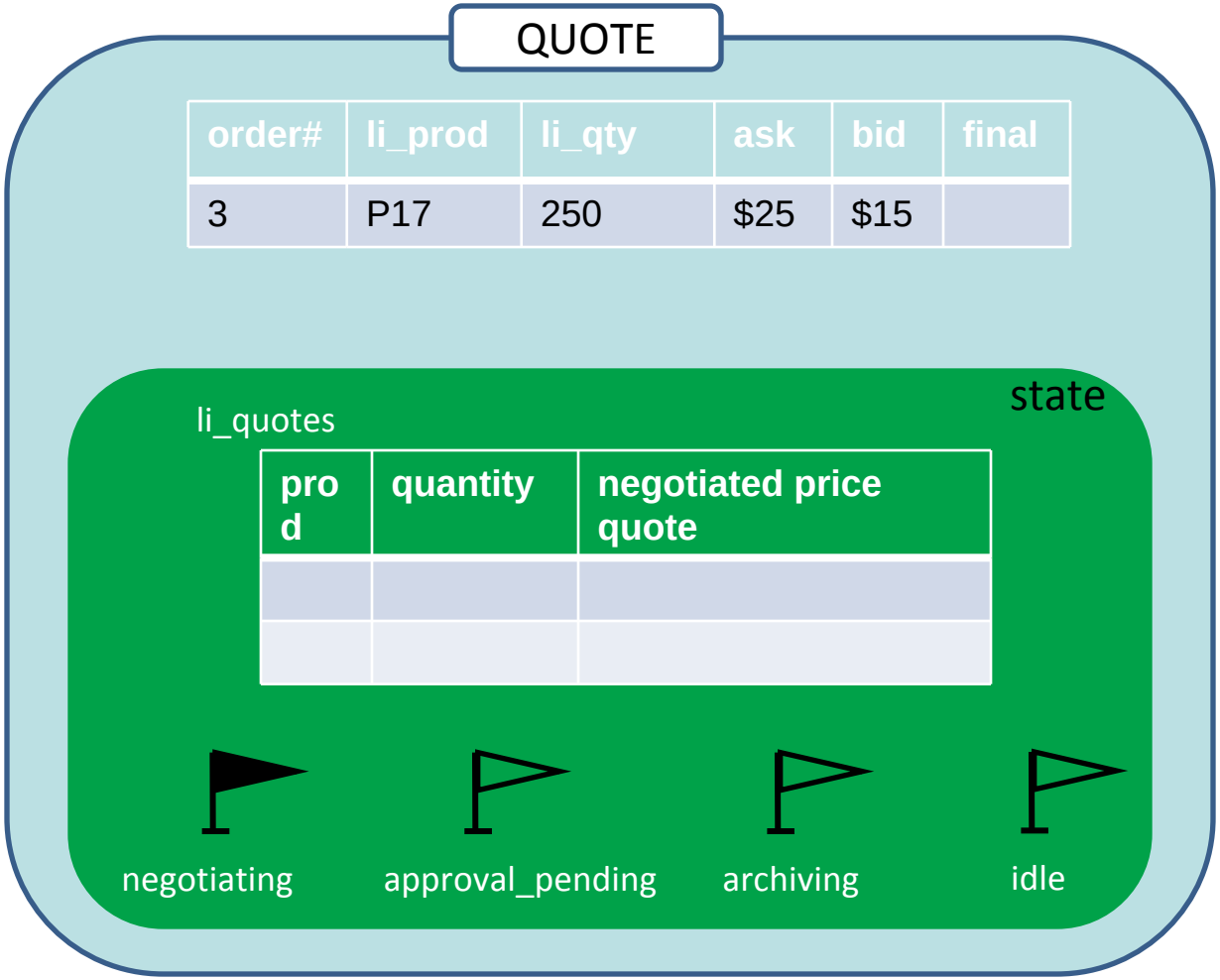
Example: Evolution of QUOTE Artifact



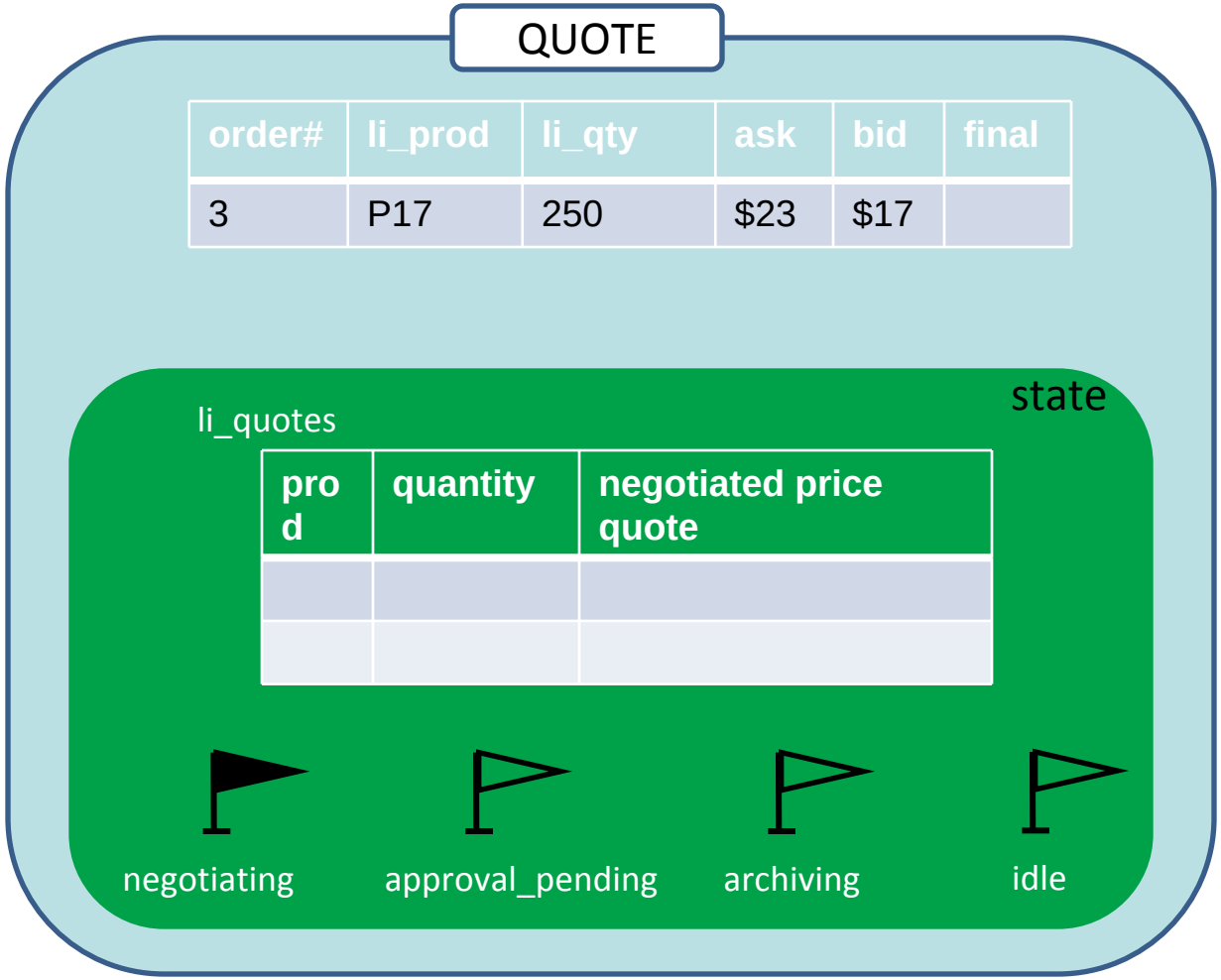
Example: Evolution of QUOTE Artifact



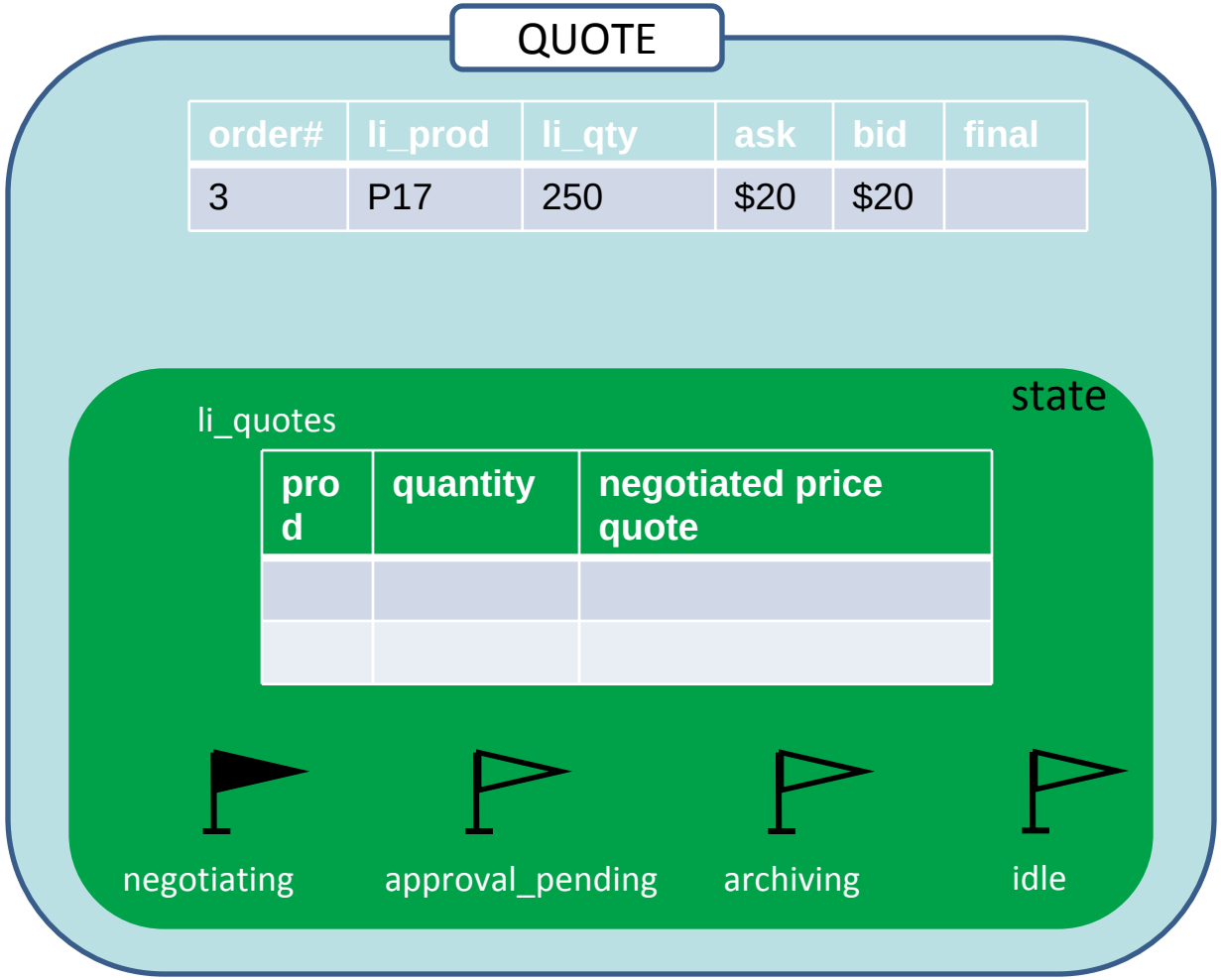
Example: Evolution of QUOTE Artifact



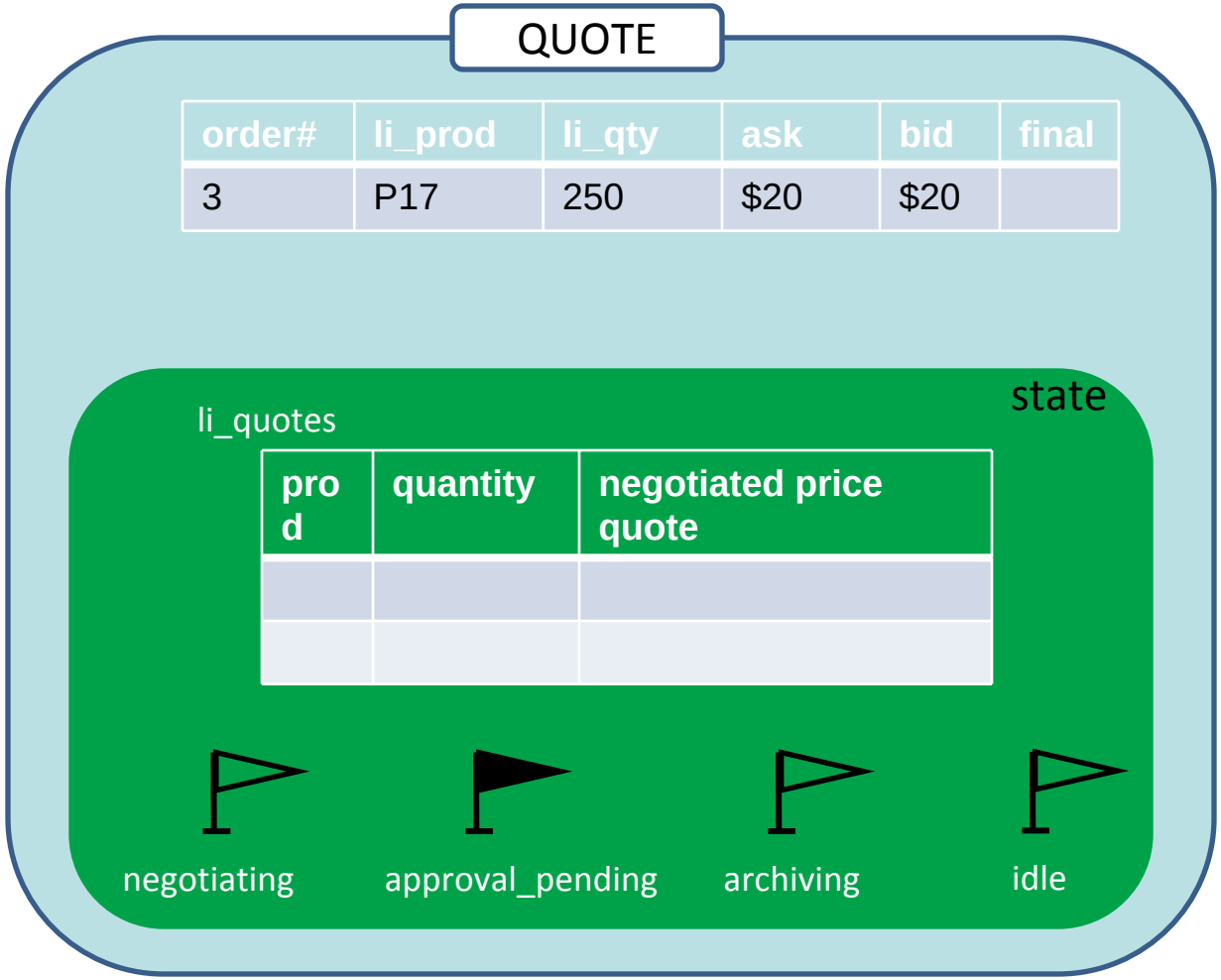
Example: Evolution of QUOTE Artifact



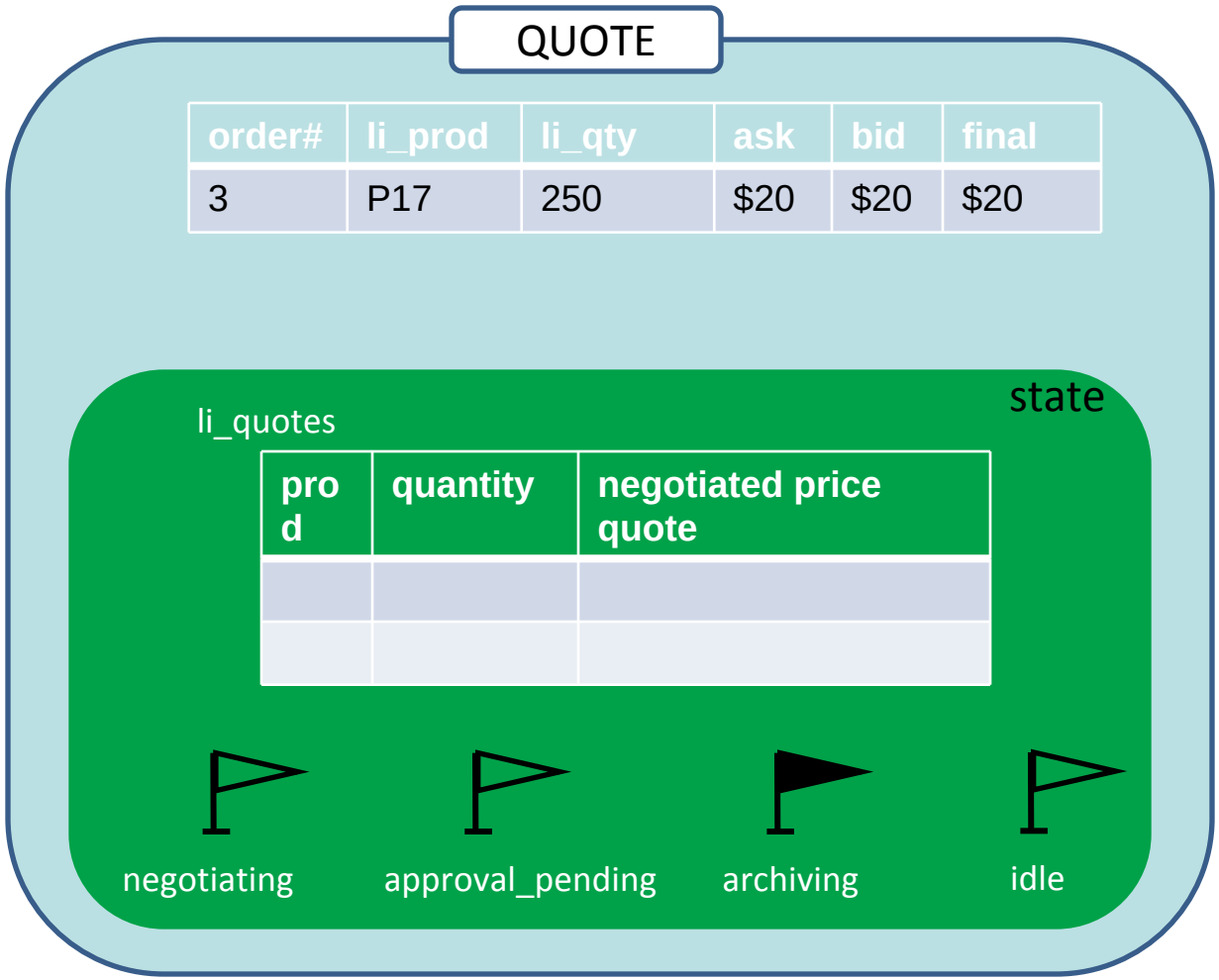
Example: Evolution of QUOTE Artifact



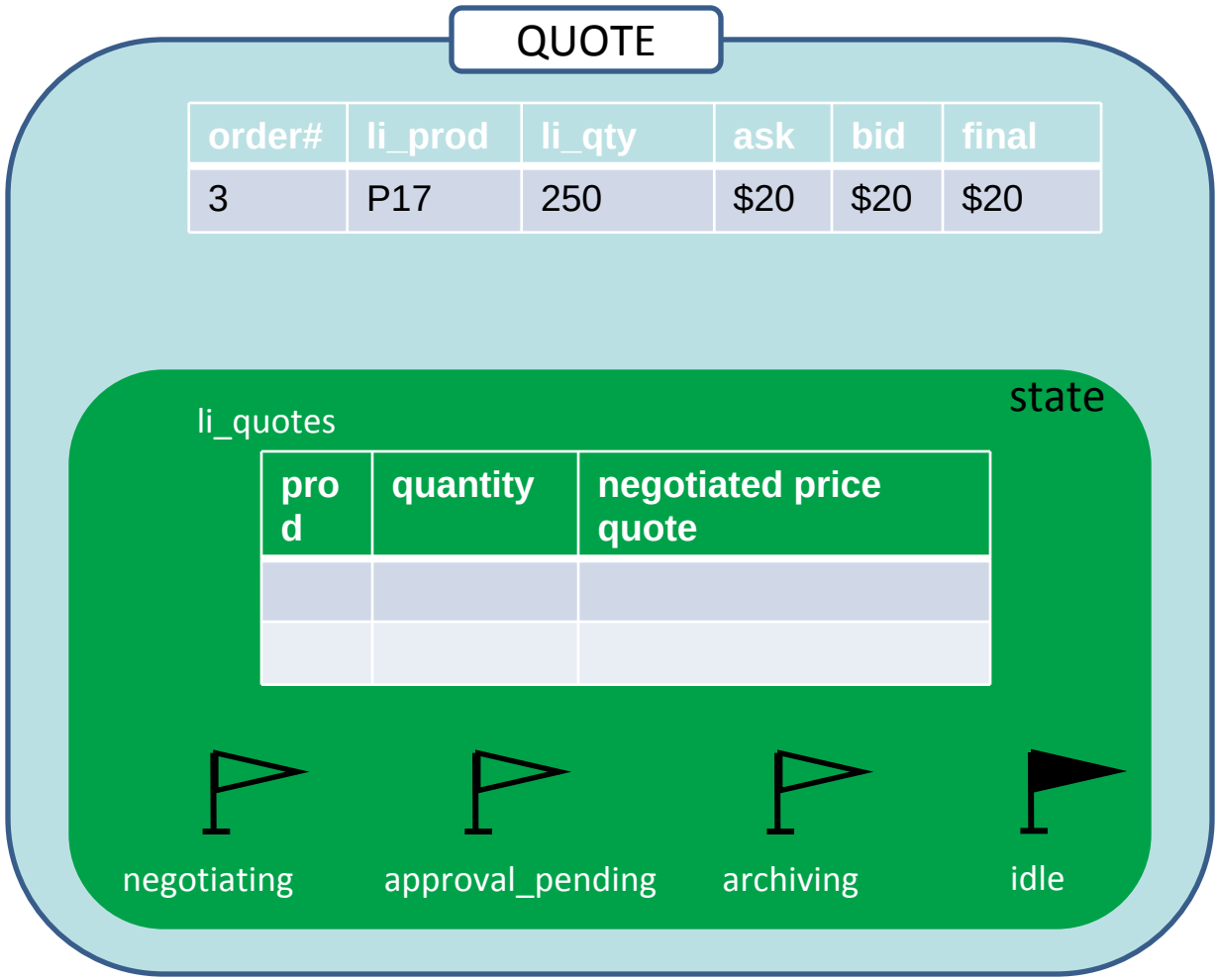
Example: Evolution of QUOTE Artifact



Example: Evolution of QUOTE Artifact

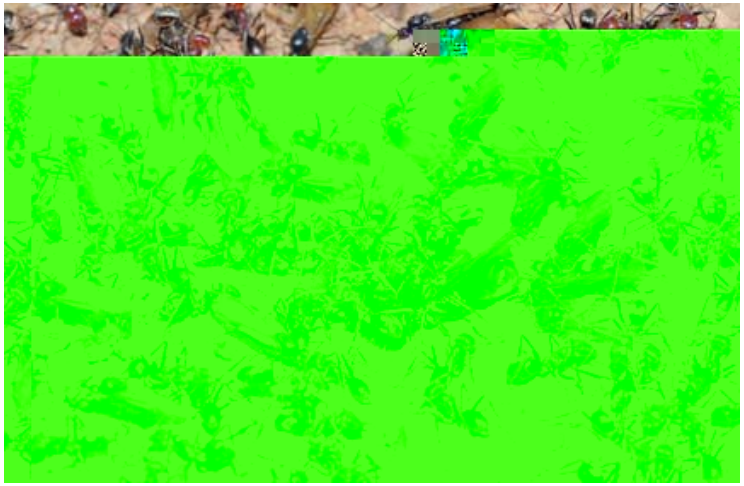


Example: Evolution of QUOTE Artifact



How to compare different workflows?

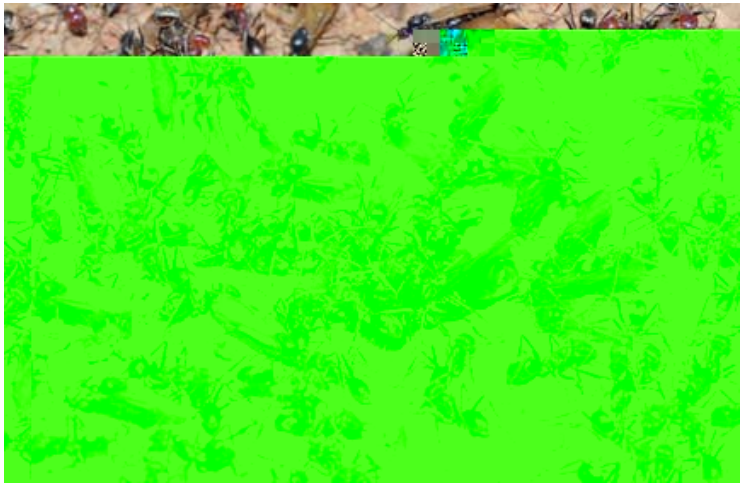
Building a beehive



Building an ant nest

Use Views!

Building a beehive

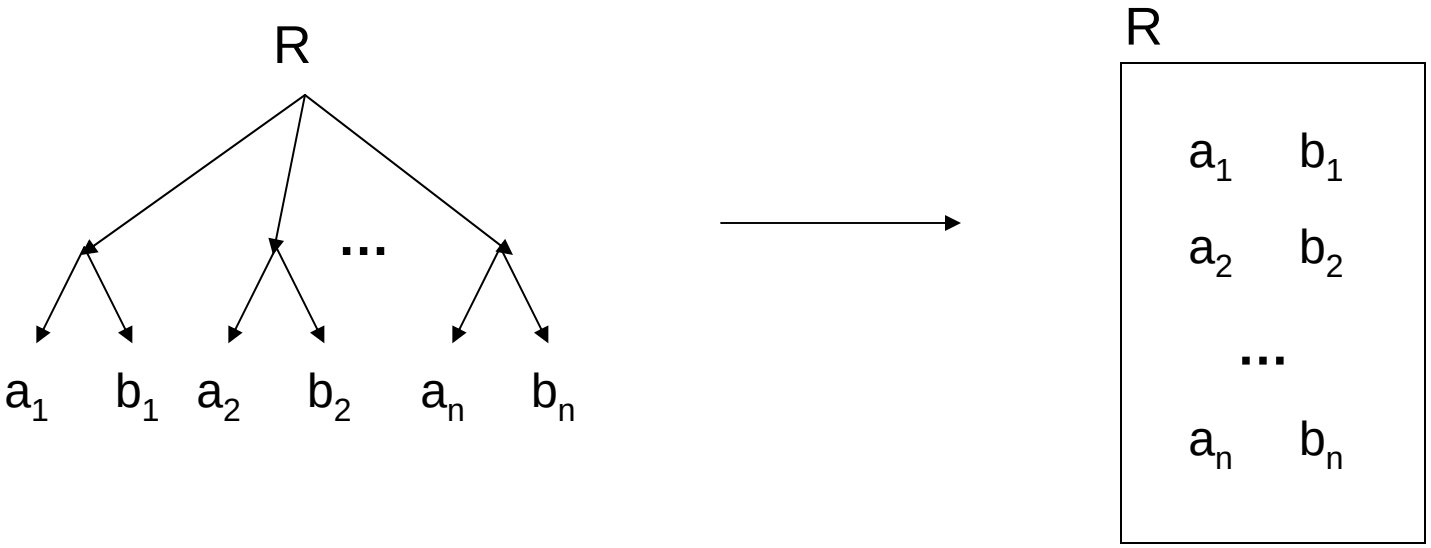


Building an ant nest

Workflow views

- Views on states: restructure, hide
- Views on events: restructure, hide → silent

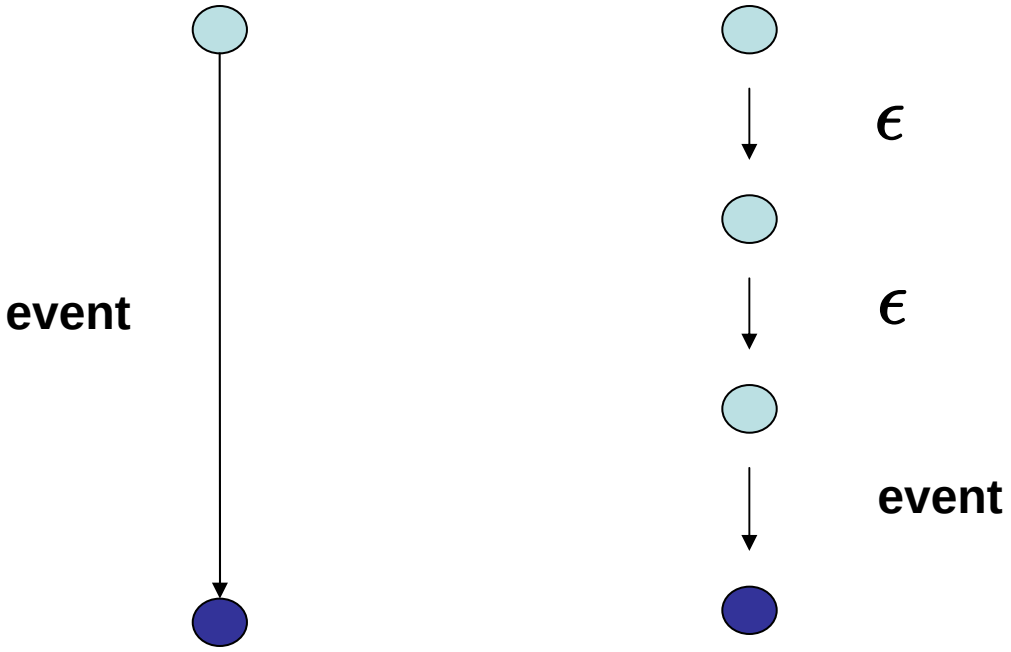
View mapping AXML to Tuple Artifacts



function call !s → service call s

Comparing workflow specification languages

- Define views mapping to a common abstraction
- Define workflow simulation



A bit more complicated for tree of runs

Workflow specification mechanisms for AXML

- **BAXML**: static constraints only
- **GAXML**: function calls and return controlled by guards
- **AAXML**: allowed transitions described by an automaton
- **TAXML**: workflow constrained by temporal property of history

Main result on AXML

BAXML can simulate GAXML, AAXML*, TAXML*

In other words:
static constraints can simulate all other mechanisms

* Modulo very minor restrictions

BAXML vs Tuple Artifacts

- BAXML can simulate Tuple Artifacts
  with respect to previous view

- Tuple Artifacts cannot simulate BAXML
  view: keep just names of services/function calls

Note: the more information is kept in the view, the harder to simulate

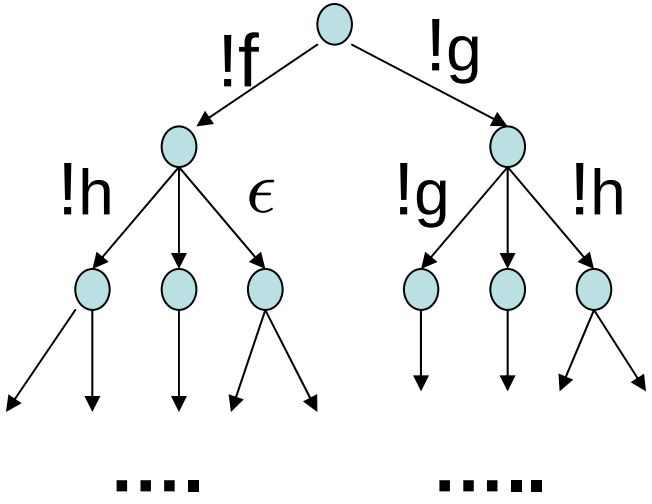
Another use of views: controlled exposure of specification

- Adapt complex specifications to needs of users
- Hide private information about internal workflow

Issue: how to explain a view to its users

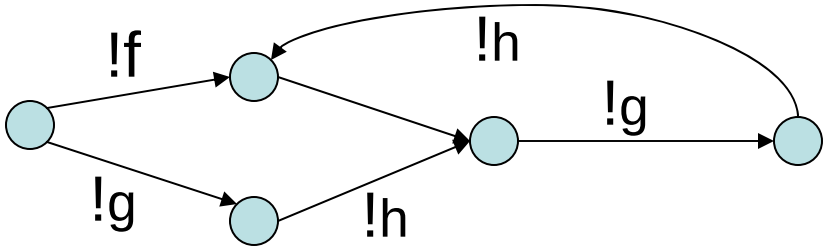
Example

- BAXML view: keep just tree of function calls



How to explain the view

- Ideally: if it is regular, finite-state transition system whose unfolding is the infinite tree



- If the tree of runs is not regular, can it be specified in a more powerful but still reasonable way? Can it be approximated? Is the *set* of infinite runs regular?
- How about more complex views with data?

Conclusion

Views rule (once again)!

- Allow comparing different workflow models
- Customization of workflows
- Abstraction that can be used in verification