Automatic Verification of Data-Centric Business Processes

Alin Deutsch (UC San Diego)
Rick Hull (IBM TJ Watson)
Fabio Patrizi (Sapienza)
Victor Vianu (UC San Diego)

Motivation

- Common BP modeling paradigm: process-centric workflow models
  - Data either completely abstracted away or added as an afterthought annotation
- Emerging trend: Data-centric workflow models
  - see Siena (IBM), WebML (Milan), WAVE (UCSD), Kepler(UCDavis), Hilda(Cornell)
- Need for data-aware verification
- Opportunity created by yet another recent trend:
  high-level specification formalisms
  - Specification becomes implementation: compiled to workflow code

Data Awareness Means Infinite-state Verification

- Infinite-state verification, notoriously difficult
- Conventional approaches to infinite-state verification:
  - abstract to finite state first, then apply model checking (loses semantics)
  - use theorem provers (incomplete, require expert user feedback)
- Our approach:
  - an alternative to general yet incomplete methods: find expressive classes of business processes and properties with decidable data-aware verification.

Artifact Systems

- "Business artifact" model introduced by IBM
  - demonstrated in practice to yield substantial improvements in the implementation of business processes

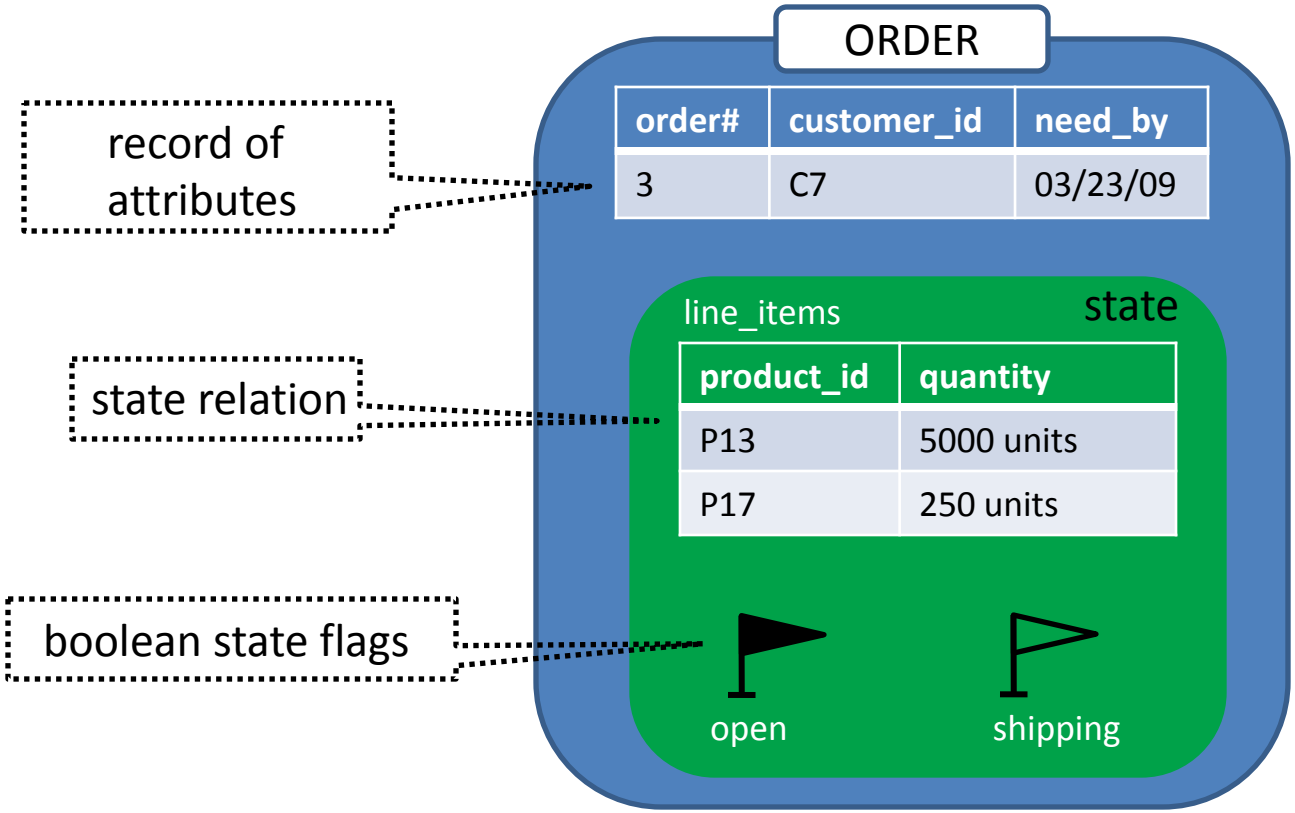
- **Artifact** = an **attribute record** annotated with **state**
  - state = collection of
    - **state relations** and
    - **boolean states** (flags) that record stages in artifact life cycle
  - **type** = schema of attribute record and states

- **Artifact System** consists of
  - collection of artifact types, one instance per type
  - underlying **database**
  - collection of **services**
    - model tasks that update artifacts
    - can be performed automatically or by various human operators

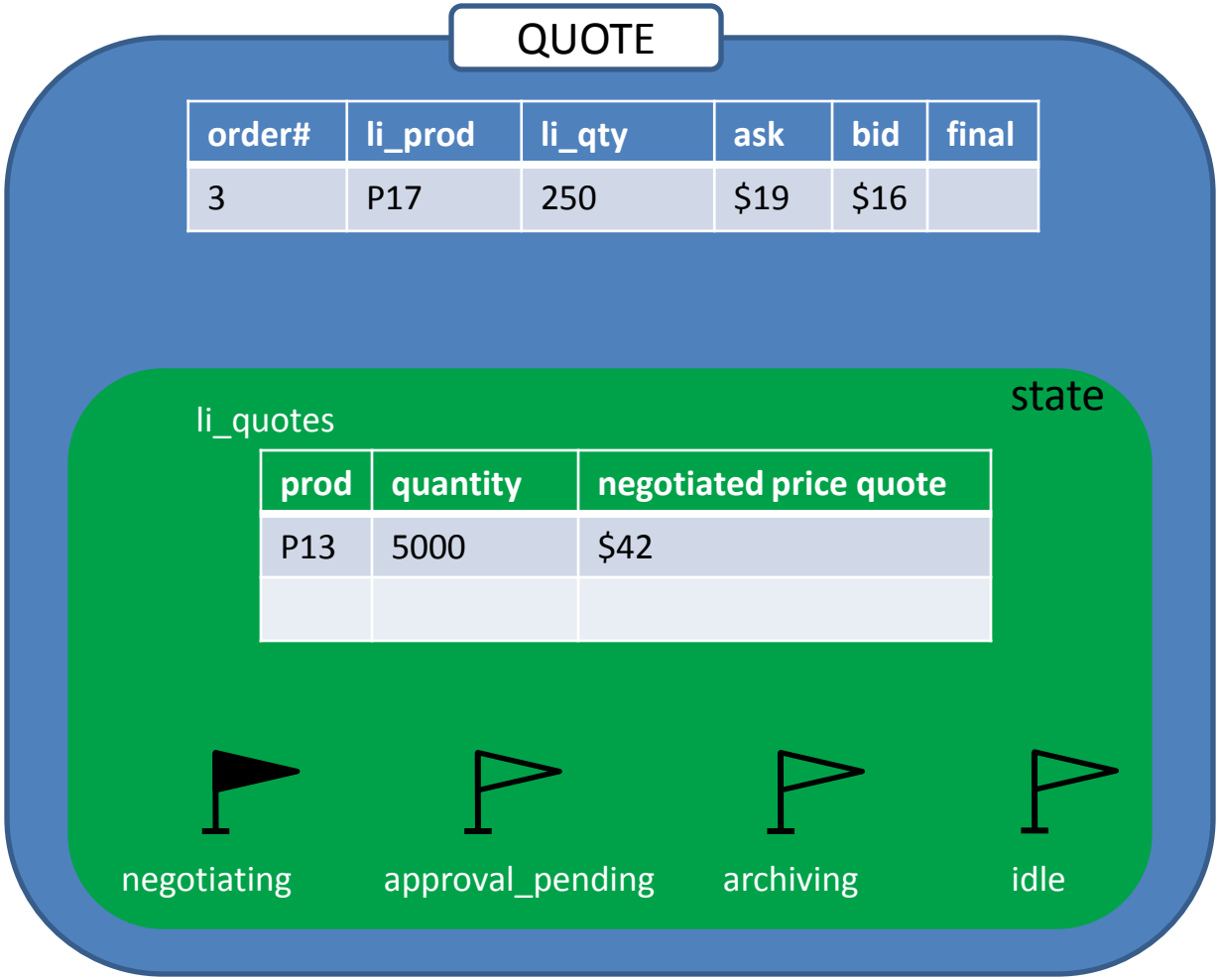
Running Example

- manufacturer
  - fills customer purchase orders
- customer
  - repeatedly adds new line items into the purchase order
- price of each line item is negotiated individually
- both parties consult an underlying database
  - customer to select available products from catalog
  - manufacturer to check customer status, desired price, etc.

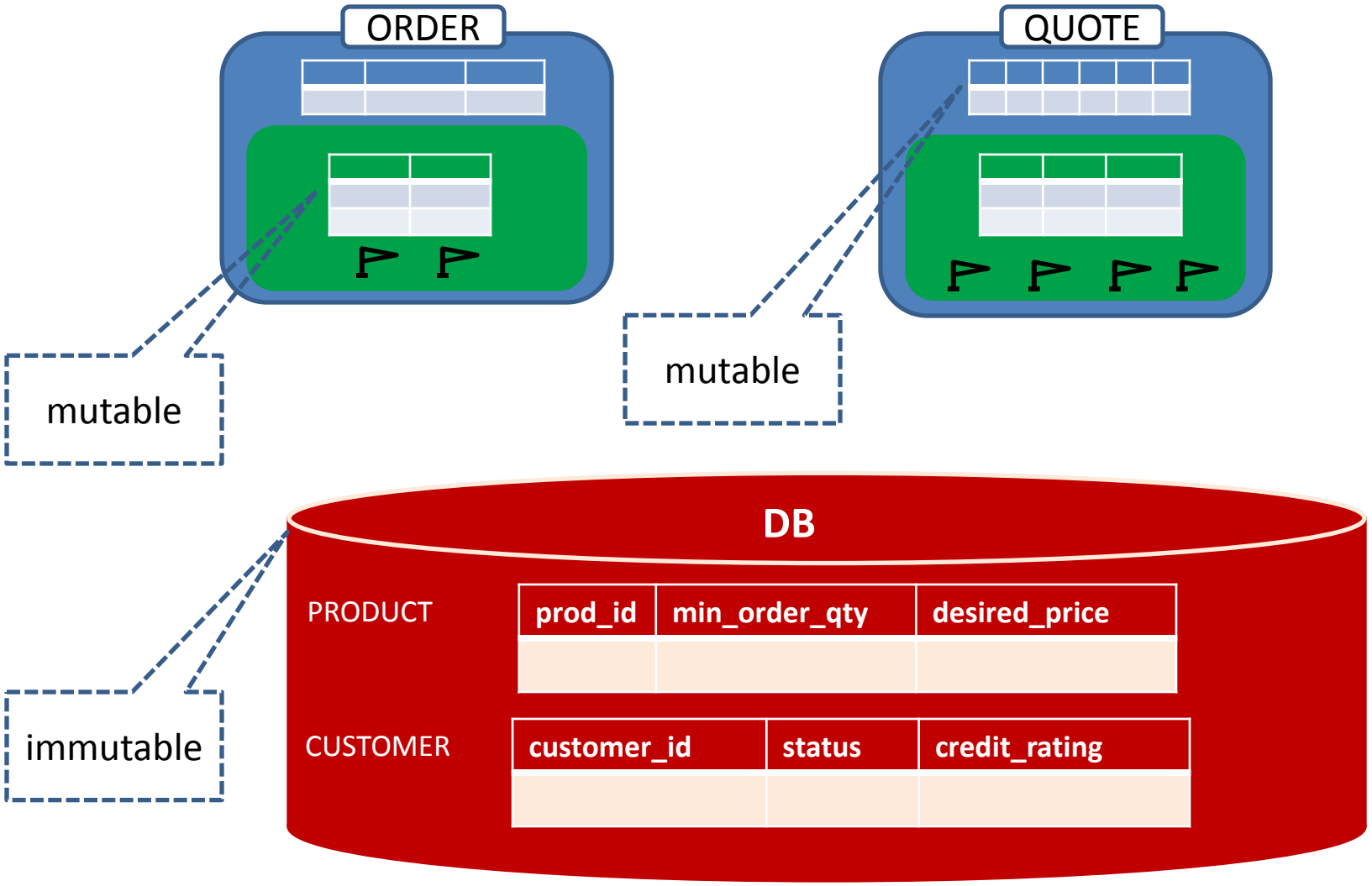
The ORDER Artifact



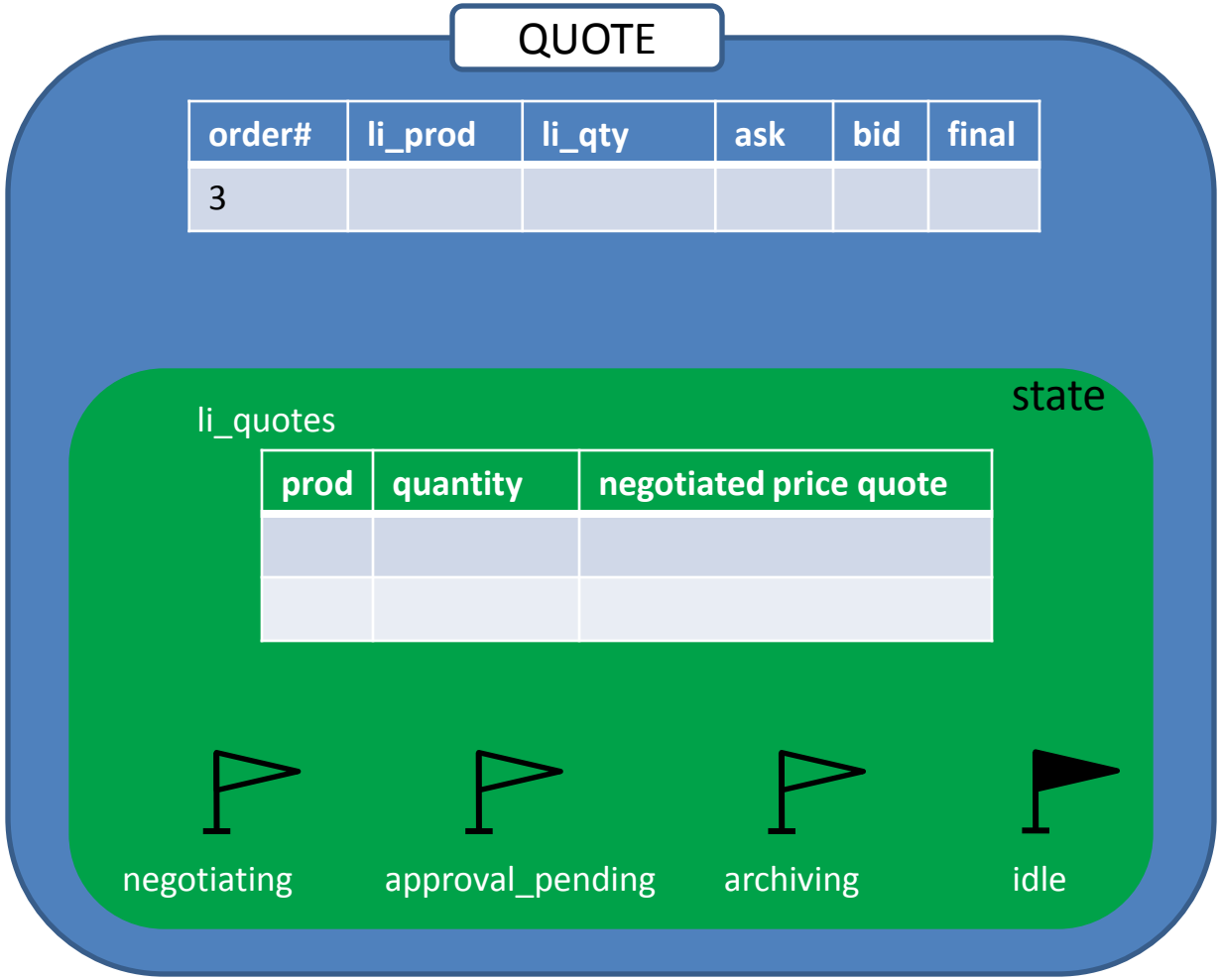
The QUOTE Artifact



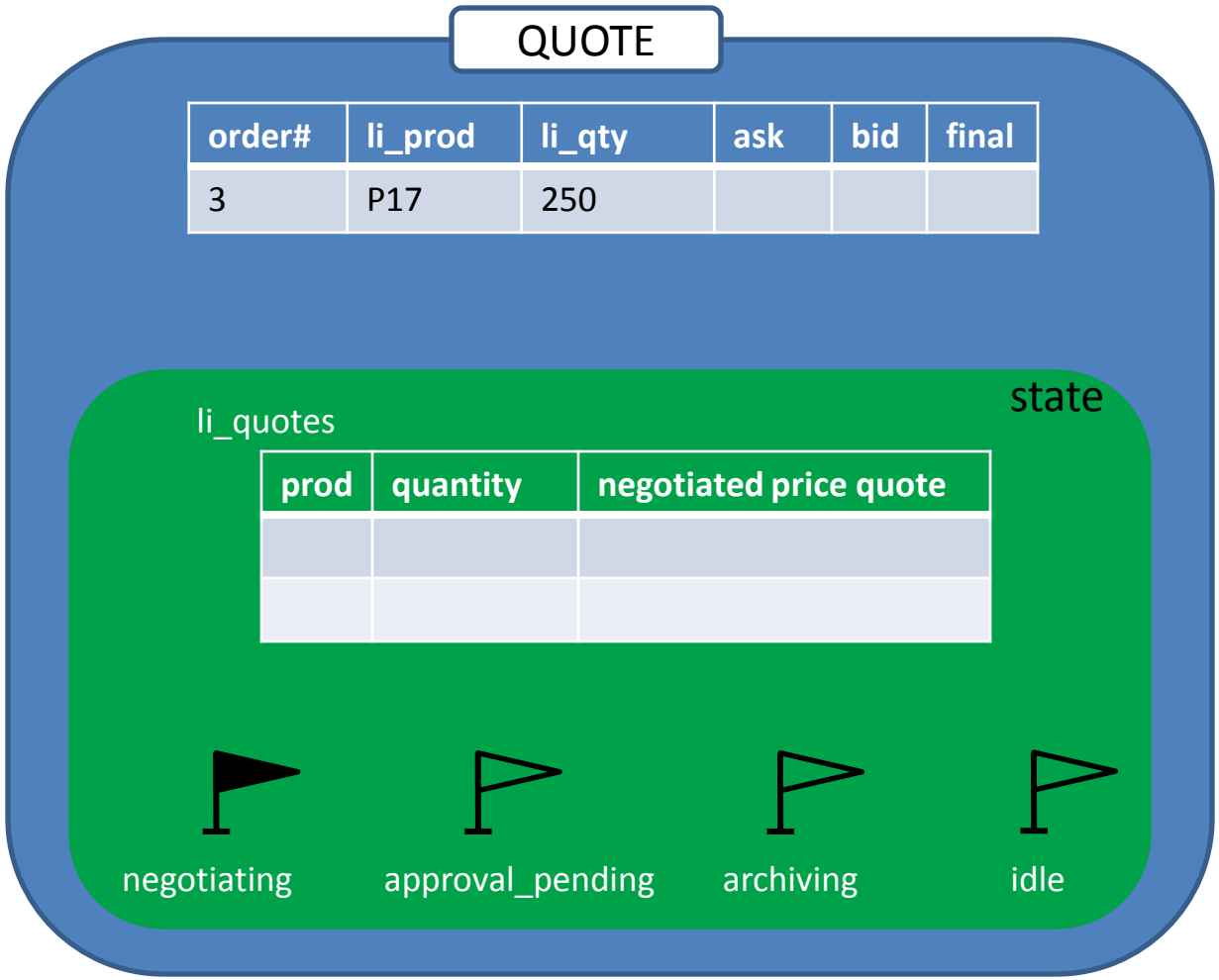
The Underlying Database



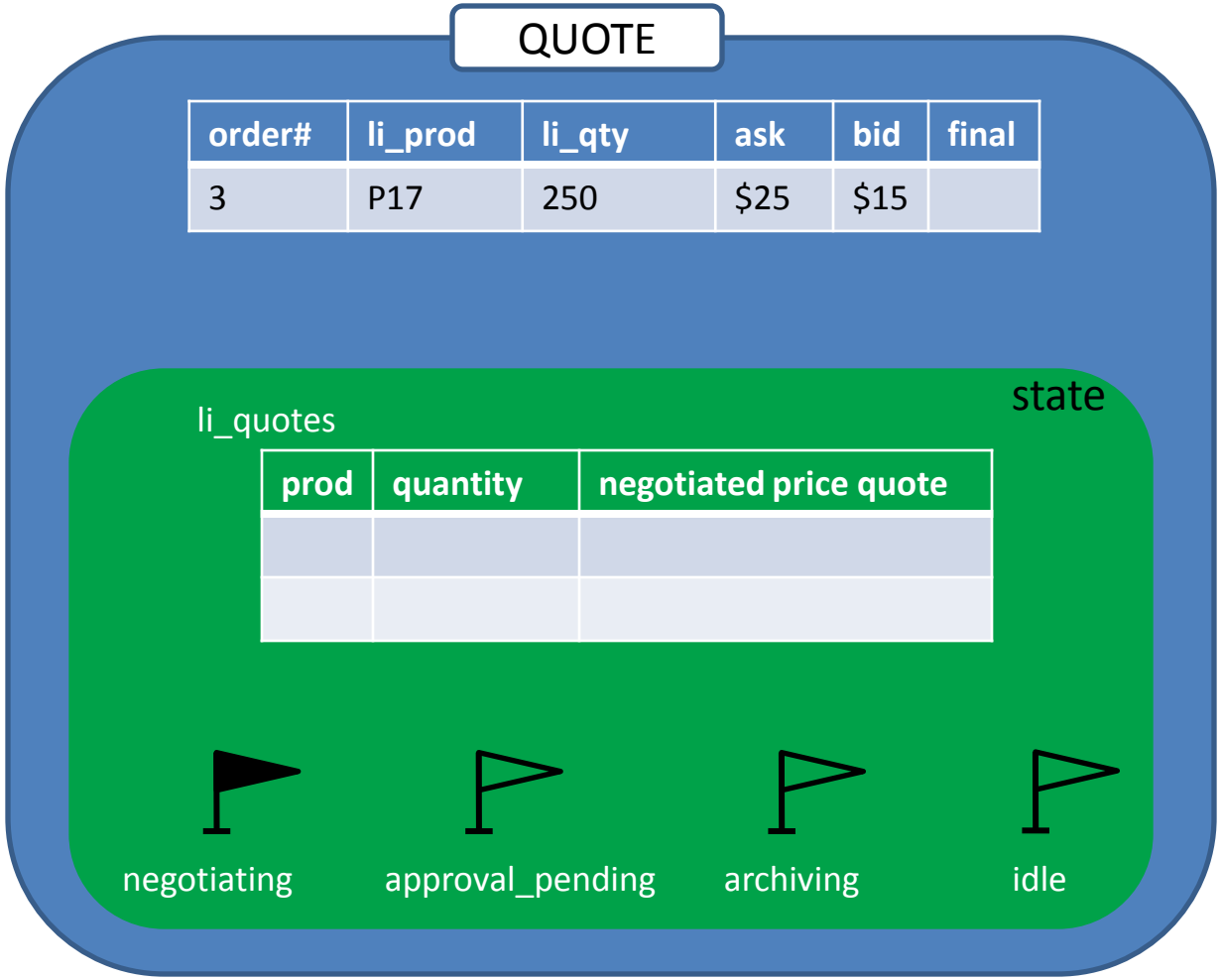
Example: Evolution of QUOTE Artifact



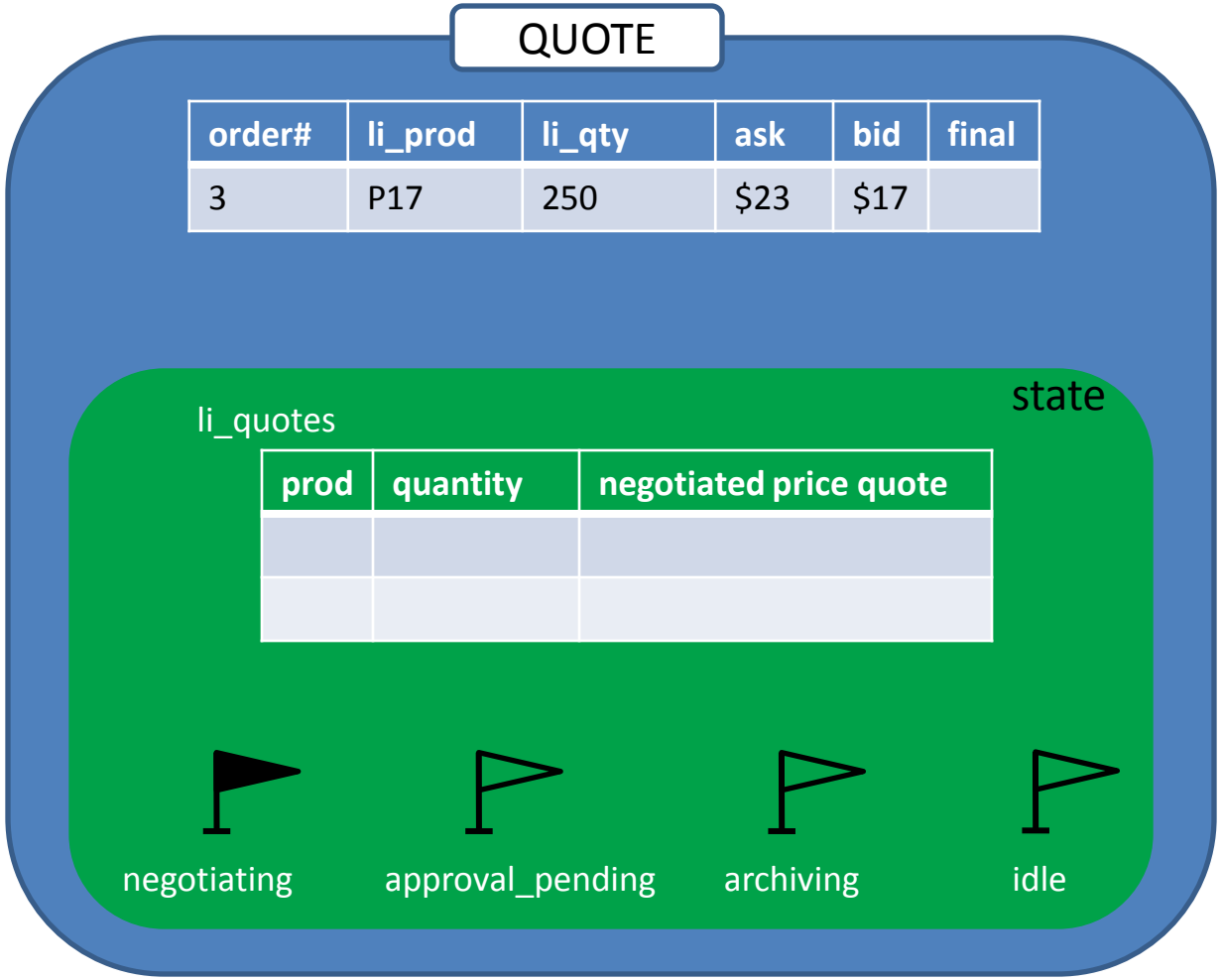
Example: Evolution of QUOTE Artifact



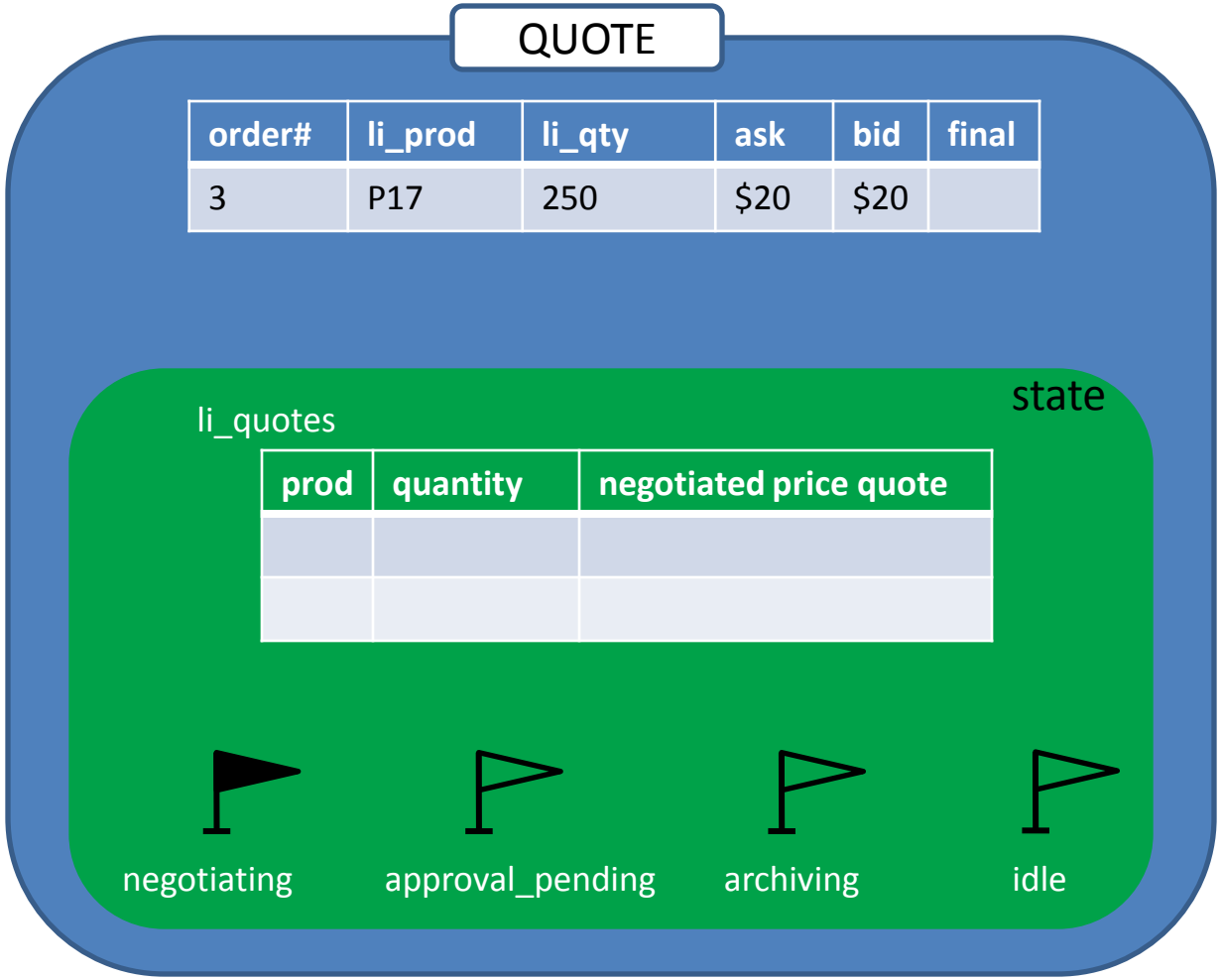
Example: Evolution of QUOTE Artifact



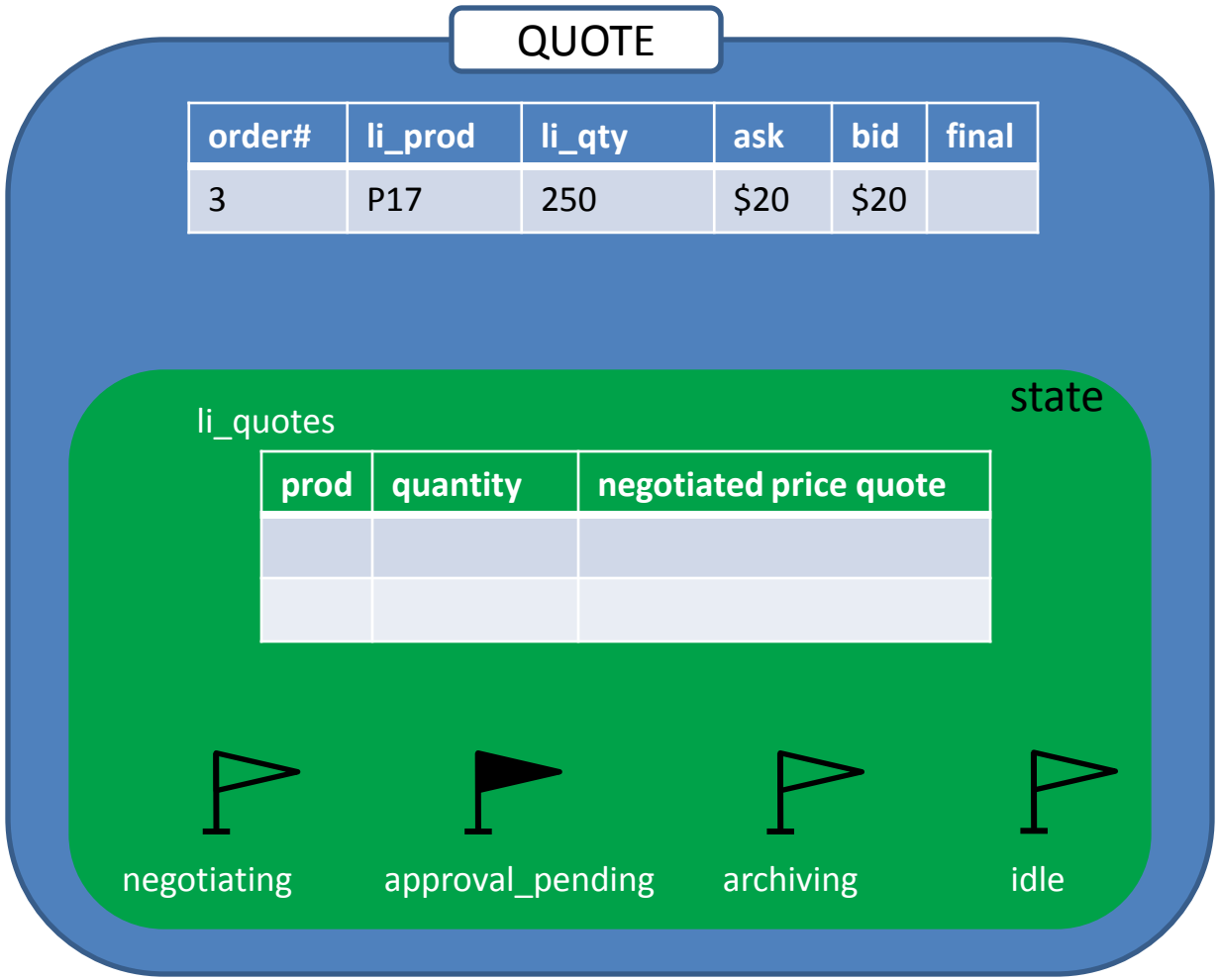
Example: Evolution of QUOTE Artifact



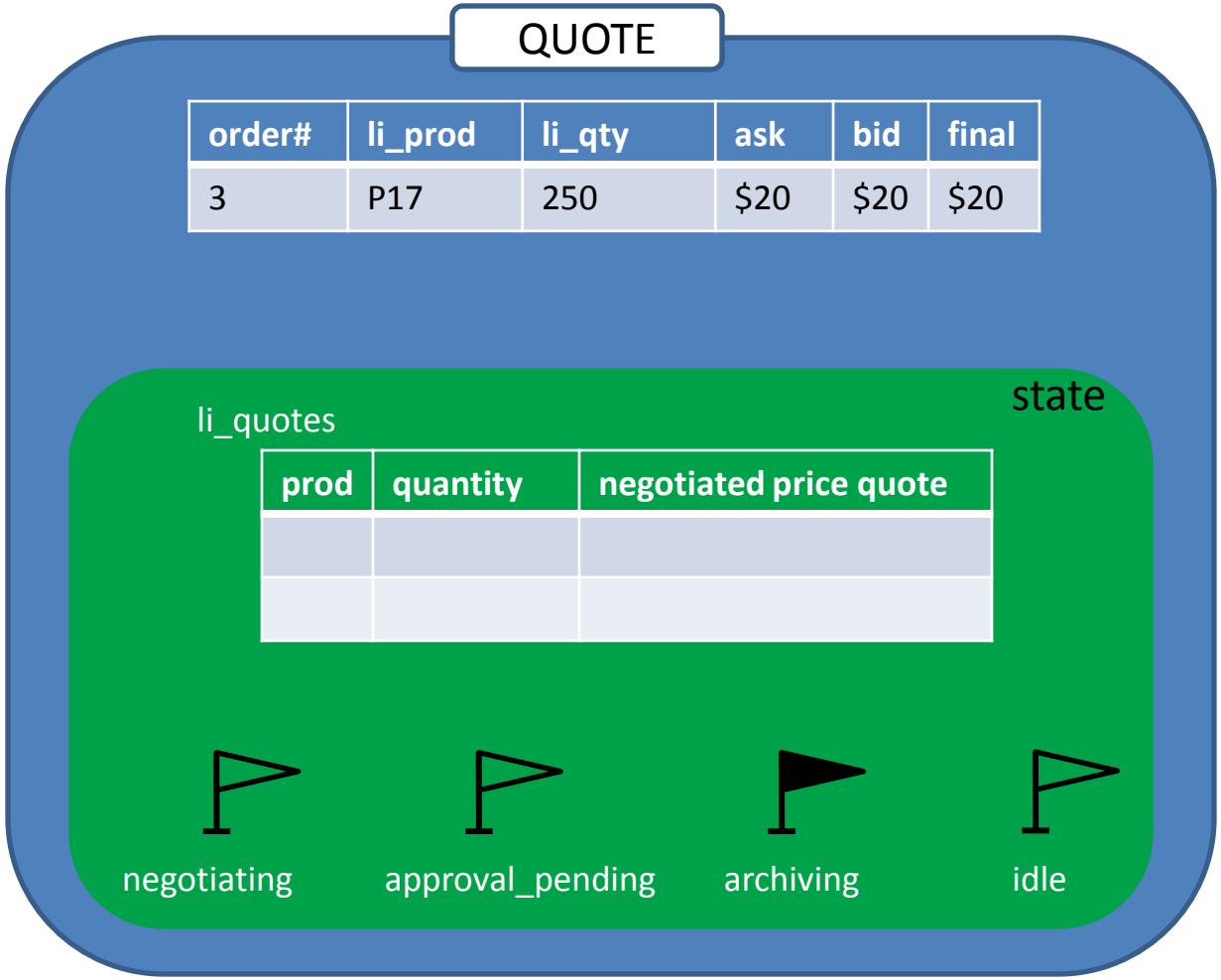
Example: Evolution of QUOTE Artifact



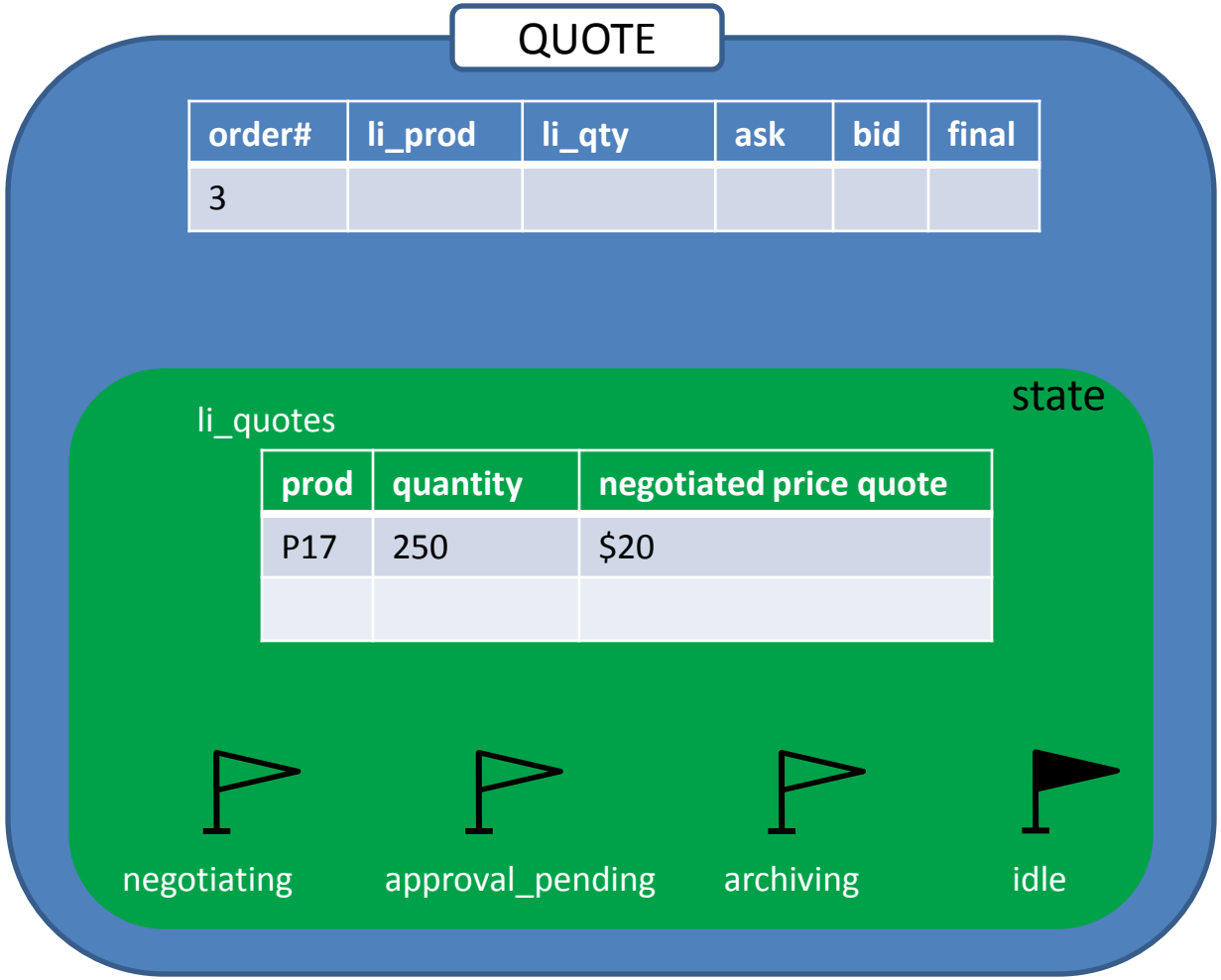
Example: Evolution of QUOTE Artifact



Example: Evolution of QUOTE Artifact



Example: Evolution of QUOTE Artifact



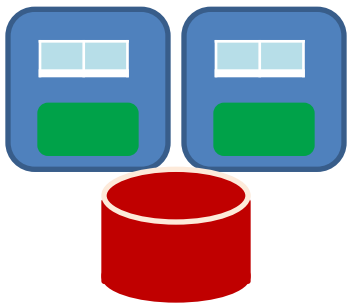
Services

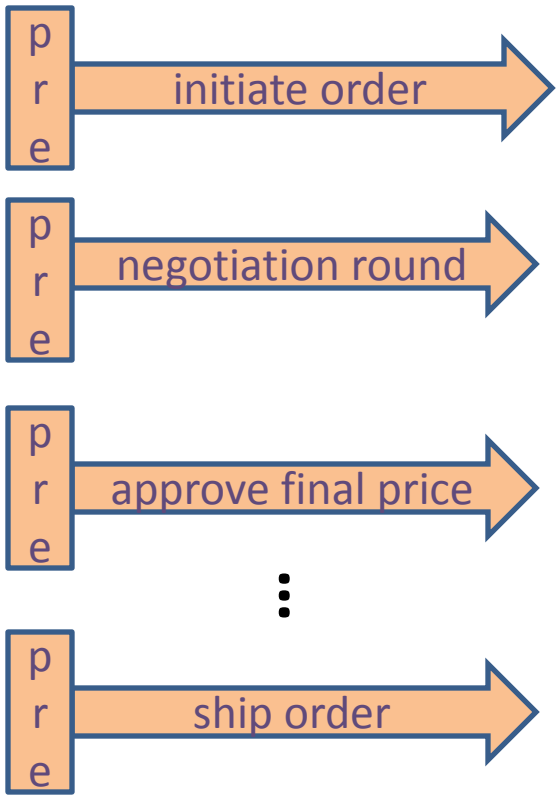
- Model operations on artifacts
  - updates of the artifact attributes
  - insertions/deletions in artifact states
  - setting/resetting of state flags

- Insertions & updates can draw values from …
  - active domain of current snapshot

  - rest of domain, which is *infinite*, to support modeling of
    - arbitrary external inputs (by programs or humans), and
    - computation that returns new values.

One Evolution Step

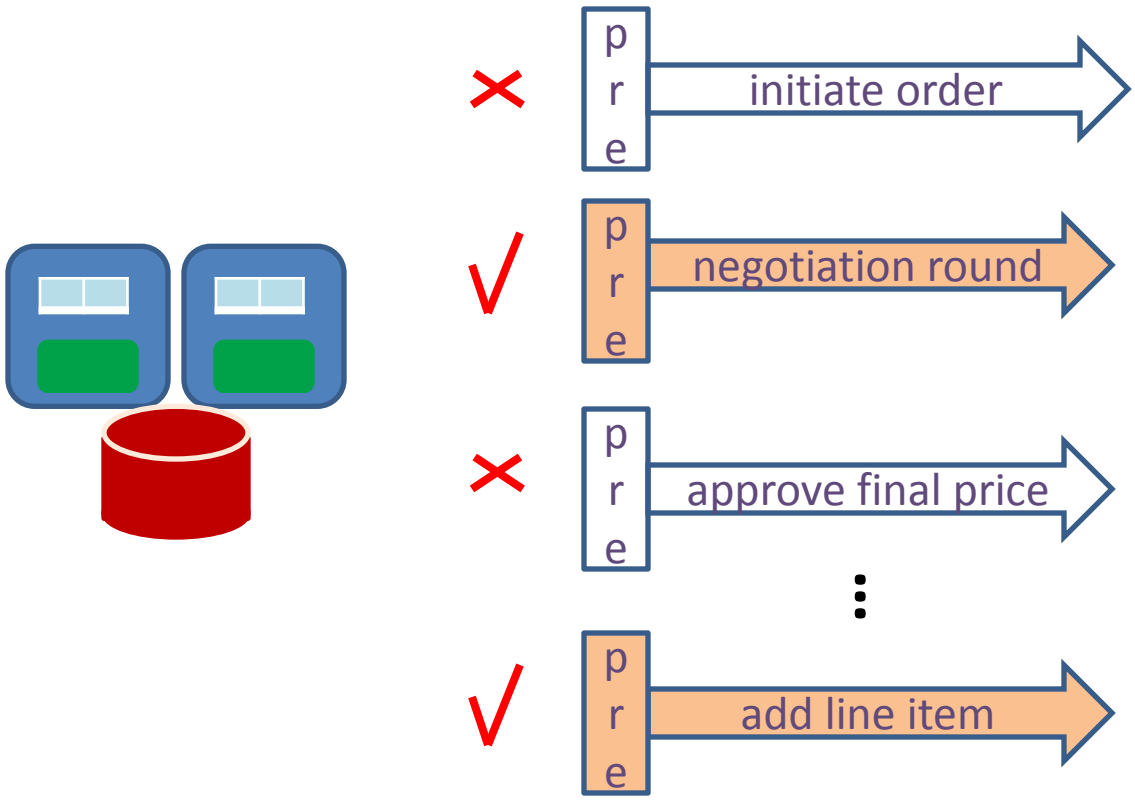




current snapshot of artifact system

available services

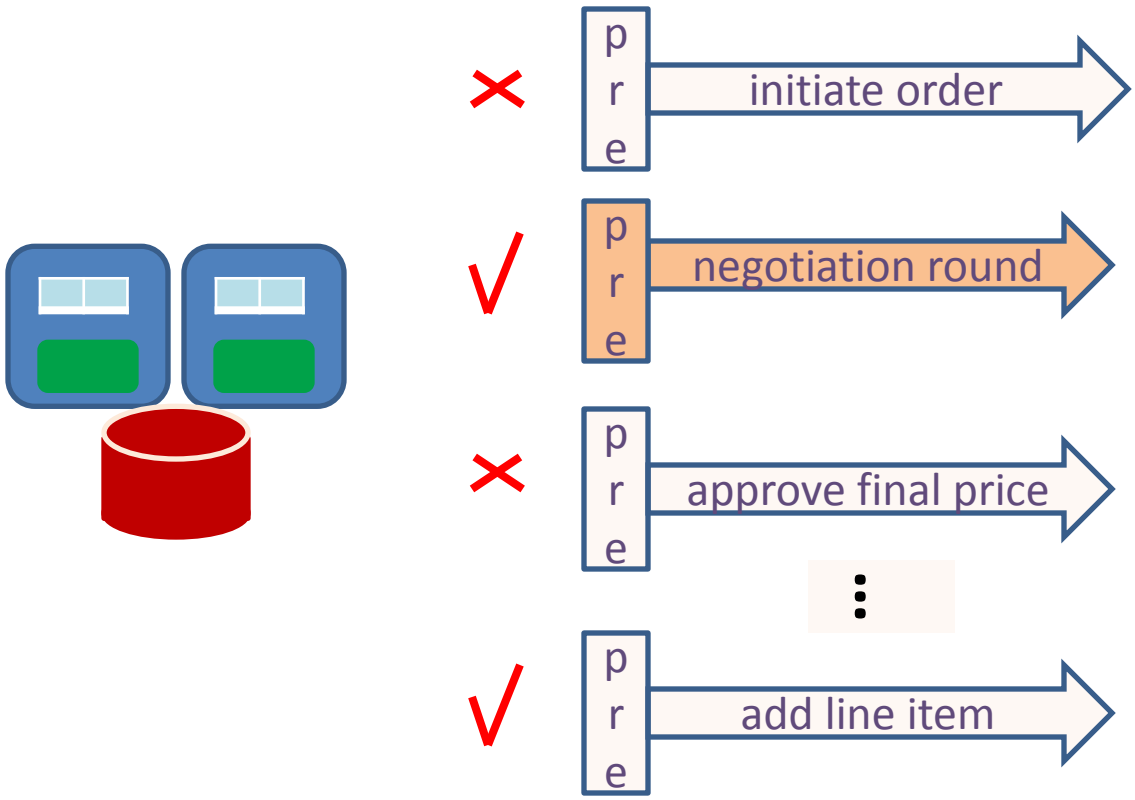
Services evaluate their pre-conditions in parallel



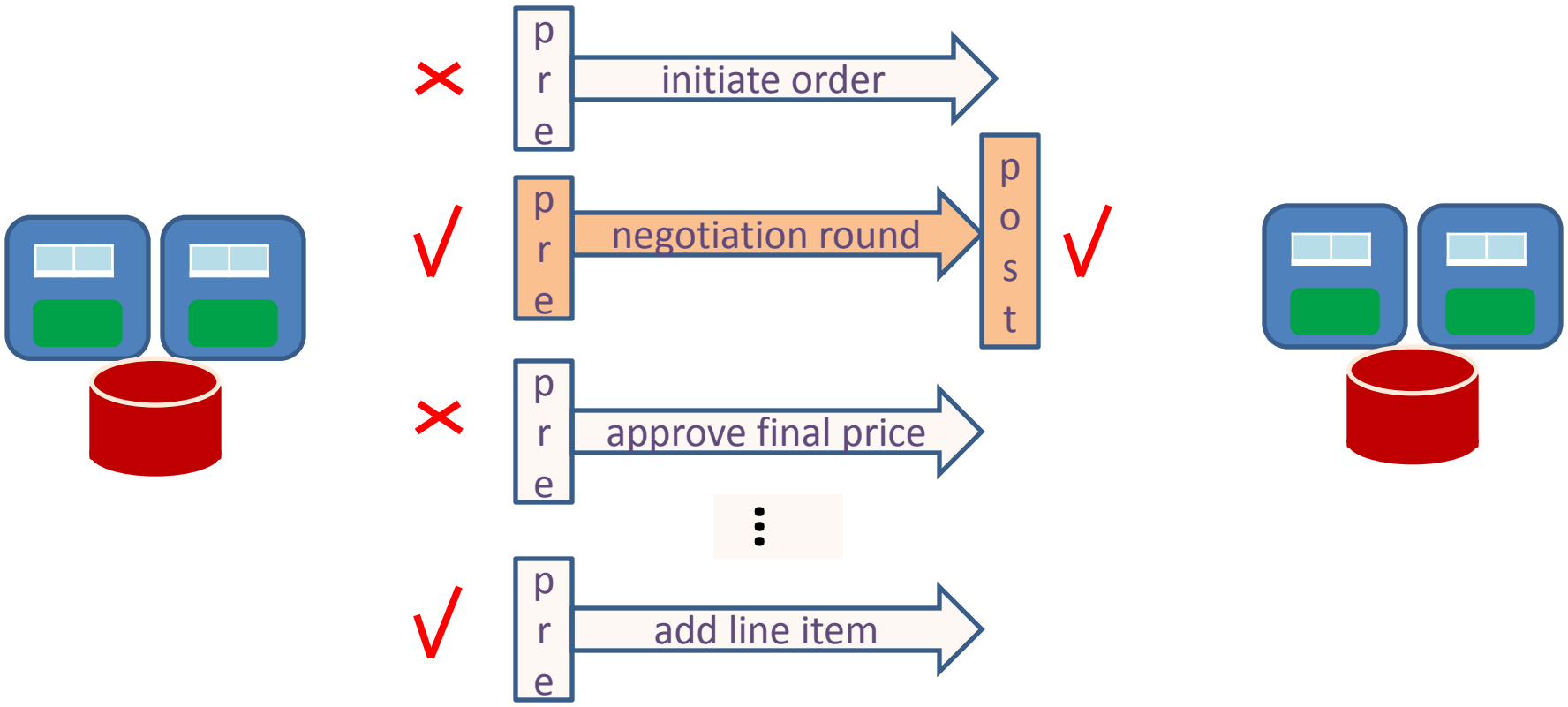
current snapshot of artifact system

available services

One qualifying service is non-deterministically picked for execution



Post-condition Is Satisfied



Desirable Correctness Properties

- Basic consistency of specification

  E.g. *QUOTE flags* **negotiating** *and* **approval_pending** *are always mutually exclusive.*

- Semantic properties relating to business model

  E.g. *If customer status is not preferred, and credit rating is worse than good, then before accepting order for a product with final negotiated price lower than desired price, explicit approval from human executive must be requested.*

Service and Property Specification

Service Specification

A service specification comprises

- a ***pre-condition***
  - boolean query on current snapshot of artifact system
- a ***post-condition***
  - specifies constraints on the updates to artifact attributes
  - relates current and successor snapshot of artifact system
- for each state relation, optional ***state insertion/deletion rules***
  - rules specify sets of tuples to insert into (delete from) state relations
  - rule = query(current snapshot)

"query", "constraint": expressed by FO formulas over ***Artifact Schema***

Artifact Schema

Artifact schema $\mathcal{A}$ comprises

- database schema
- schemas of state relations of all artifacts
  - boolean state flags modeled by singleton nullary relations
- schemas of attribute records of all artifacts
  - attribute record modeled by singleton relation ***(attribute relation)***
  - convention: $\mathbf{R_O}$ is attribute relation for artifact **O**
- built-in relations **=, ≤** on domain

Example: Negotiation Round as Service

- Pre-condition:

  a new round can start only if **negotiating** flag is set in QUOTE artifact

- State rules (one of them):

  once negotiation converges, **negotiating** flag is reset

Example: Negotiation Round as Service

- Pre-condition:

- State rules (one of them):

once negotiation converges, **negotiating** flag is reset

Legend: database attribute state

Example: Negotiation Round as Service

- Pre-condition:

deletion rule

negotiating

- State rules (one of them):

ω constant models uninitialized value

$\neg$**negotiating** $\leftarrow \exists o,p,q,a\ \mathbf{R_{QUOTE}}(o,p,q,a,a,\omega)$

evaluated against successor snapshot

evaluated against current snapshot

Legend: database attribute state

Cont'd: Negotiation Round as Service

- Post-condition:

evaluated against successor snapshot

evaluated against current snapshot

$$R_{QUOTE}(o,p,q,a,b,f) := \exists\, a',b'\ \ R_{QUOTE}(o,p,q,a',b',f) \wedge a' \neq b' \wedge f=\omega \wedge a' \geq a \wedge b \geq b'$$

Ask and bid values a,b in successor instance are unknown at design time
(if negotiation has not converged in current instance).

What we do know:
- new values (not necessarily drawn from active domain of current instance).
- respect well-formed negotiation rules.

Legend: database attribute state

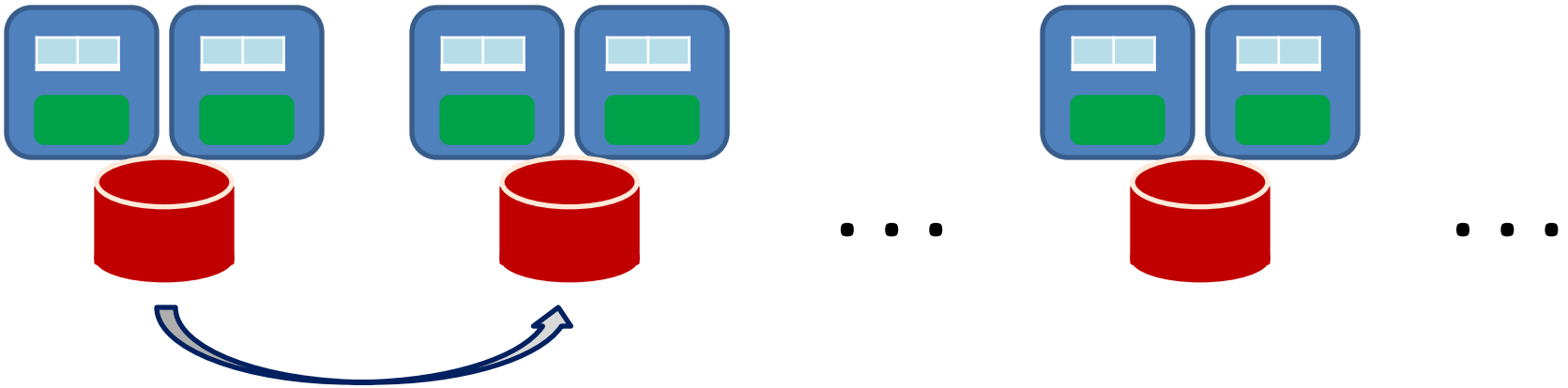
Service Semantics

We say that snapshot **A'** is a ***possible successor*** of **A** w.r.t. service σ if

- **A** satisfies precondition of σ
- the database is preserved from **A** to **A'**
- for each state S, **A'**(S) is obtained from **A**(S) by
  - inserting tuples in result on **A** of insertion rule of **S** (if any),
  - deleting tuples in result on **A** of deletion rule of **S** (if any),
- A,A' satisfy post-condition of σ

Runs of an Artifact System

Correctness properties pertain to runs = infinite sequences of snapshots



possible successor
w.r.t. some service

Note: despite running on a finite db, the entire run may use infinitely many fresh values drawn from domain.

LTL(FO) Properties

- Properties expressed in ***LTL(FO)***
  = standard linear temporal logic
  with propositions replaced by FO formulas
  (make statements about individual snapshots in run)

- Classic LTL temporal operators

X*p*	*p* holds in **neXt** (successor) snapshot
*p***U***q*	*p* holds in every snapshot **Until** *q* does
F*p*	*p* holds eventually (**Finally**)
G*p*	*p* holds **Generally** (in all snapshots)
*p***B***q*	*p* holds **Before** *q*

Example Properties Revisited

- Basic consistency of specification

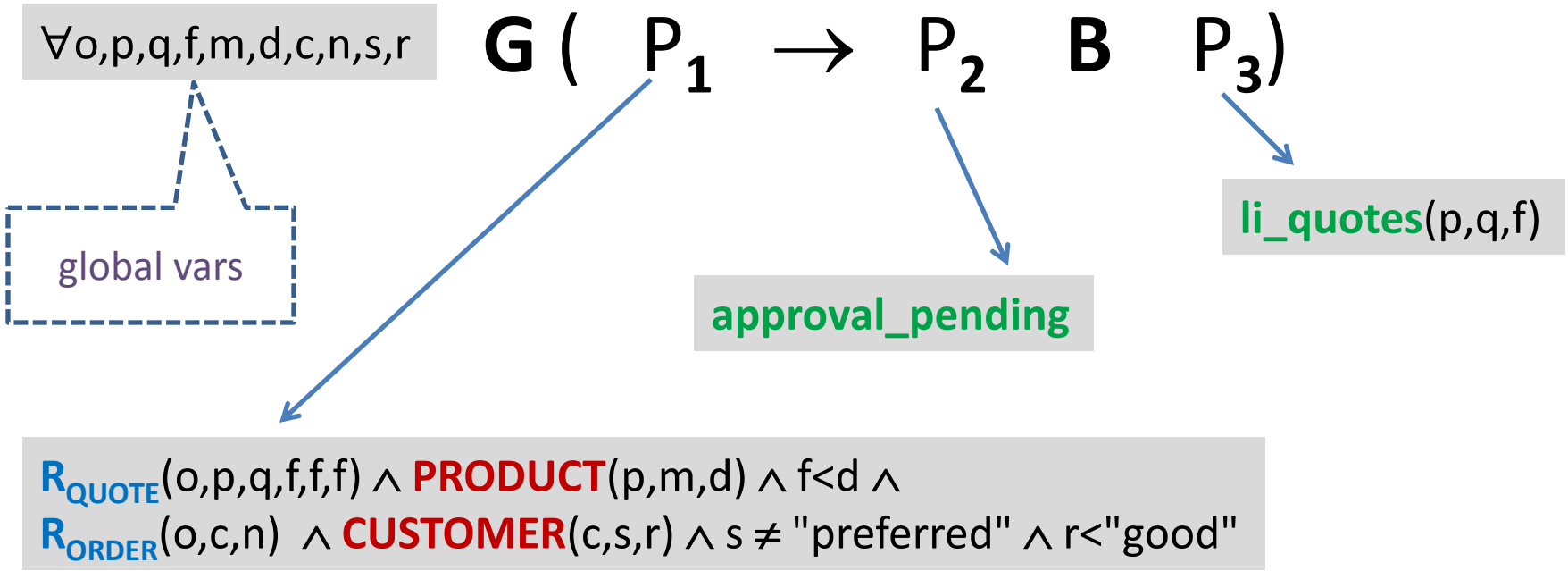
QUOTE flags negotiating*,* approval_pending *always mutually exclusive:*

G $\neg$(**negotiating** $\wedge$ **approval_pending**)

Legend: database attribute state

Semantic Property Revisited

If customer status is not preferred, and credit rating is worse than good, then before accepting order for a product with final negotiated price lower than desired price, explicit approval from human executive must be requested.



Legend: database attribute state

Verification

Verification Problem

Do all runs of a given artifact system satisfy a given LTL(FO) property?

- *Undecidable* even if property uses *no* temporal operators.
  - follows immediately from Trakhtenbrot's theorem
- Restrictions needed for decidable verification...

Guarded Restriction

- Main idea: quantification is restricted

- Related to *input-bounded quantification* for
  - ASM transducers [Spielmann, PODS'00], and
  - extended ASM transducers [D,Sui,Vianu, PODS'04, JCSS'07]

Guarded Restriction

Restrictions on quantification:

- ***Guarded FO formulas***: replace quantification rule as follows

  ~~$\exists x\, \varphi(x)$~~ $\quad\quad$ $\exists x\, (\, \mathbf{R_O}(...,x,...) \wedge \varphi(x)\,)$

  ~~$\forall x\, \varphi(x)$~~ $\quad\quad$ $\forall x\, (\, \mathbf{R_O}(...,x,...) \rightarrow \varphi(x)\,)$

  where $\mathbf{R_O}$ is the attribute relation for some artifact **O**,
  and x cannot appear in state atoms in φ.

- ***Guarded artifact system***:
  - all formulas used in state rules are guarded
  - all pre- and post-conditions are $\exists$***FO** formulas, with all state atoms *ground*
- ***Guarded LTL-FO sentence***: all its FO components are guarded

Not too restrictive: detailed spec of running example is guarded (see paper).

Guarded State Rule

Recall a previous state rule

$$\neg \textbf{negotiating} \leftarrow \exists o,p,q,a \; \mathbf{R_{QUOTE}}(o,p,q,a,a,\omega)$$

Guarded, since:

- all quantified vars appear in attribute relation atoms, and
- do not appear in state atoms

Legend: database attribute state

Main Result

Given: a guarded artifact system Γ and a guarded LTL(FO) property ϕ.

Theorem: It is decidable whether every run of Γ satisfies ϕ.
PSPACE-complete for fixed-arity schemas, EXPSPACE otherwise.

- Couldn't hope for better: finite-state model checking is PSPACE-complete
- Fundamental reasons why proof does not reduce to previous work on verification of ASM [Spielmann] and WAVE model [D,Sui,Vianu]:
  - even for fixed, finite db, one run can accumulate infinitely many values
  - presence of total order ≤ (dense, countable, no-endpoints)

Applications

Business Rules

- Proposed in [Bhattacharaya, Gerede, Hull, Liu, Su. BPM'07]
  - Conditions super-imposed on pre-condition of existing services without changing their implementation

- Support reuse & customization of 3rd party services
  - Providers strive for wide applicability and pick as unrestrictive pre-conditions as possible
  - Consumer requires more control

- Our modeling of a business rules:
  - sentence associated to a service, same syntax as pre-conditions.

Static Analysis for Business Rules

- Verification under business rules:
  - Is property satisfied by all runs that obey the business rules?

Verification under business rules is decidable for guarded artifact systems, properties and business rules.

- Redundant business rules:
  - Does a set of business rules imply another rule β for artifact system Γ?
    (i.e. does β rule out any run of Γ?)

Business rule redundancy is decidable for guarded artifact systems and business rules.

Redundant Attributes

Often useful to streamline specification by removing artifact attributes.

- Rick has anecdote involving an IBM customer.

Optimization question:

*Is there a way to satisfy property ϕ without using attribute a of Artifact **A**?*

i.e. Can ϕ be satisfied while attribute remains uninitialized?

$$\Gamma \models \phi \rightarrow \mathbf{F}\,(\ \exists x\, \exists a\ R_A(x,a)\ \wedge a \neq \omega\ )$$

If Γ and ϕ are guarded, and ϕ has no global variables, then checking attribute redundancy is decidable.

Boundaries of Decidability

Boundaries of Decidability

The following relaxations of the guarded restriction lead to undecidability:

- Relax restrictions on states by modeling them as set-valued attributes.
  Reduction from PCP
- Allow non-ground state atoms in pre- and post-conditions.
  PCP
- Allow state projections in state rules (simple form of unguarded quantif.)
  implication for functional and inclusion dependencies
- Allow un-guarded quantification in property
  implication for functional and inclusion dependencies
- Allowing path quantification in properties: CTL(FO)
  implication for functional and inclusion dependencies

Boundaries of Decidability (2)

The following are undecidable even under the guarded restriction.

- Add even a single key constraint.
  Reduction from PCP

- Existence of blocking pre-run.
  PCP

- Replace ≤ (dense, no-endpoint total order) by a successor relation.
  PCP

Summary

We formalize a business process modeling paradigm that has attracted attention in industry and research

The flavor in this talk extends prior artifact-centric models

- Introduces underlying database that is consulted by services
- Services equipped with updateable state relations
- Service and property allow sophisticated data manipulation
- Infinite, ordered domain, critical for appropriately modeling
  - arbitrary external input
  - results of computation generating new values
  - partially specified sub-processes

- We explore boundaries of decidability for verification
  - identify expressive and tight class of guarded systems and properties
  - decidability in PSPACE, no worse than finite-state model checking

Conclusion

Our results were enabled by a mix of data-aware modeling with techniques from model checking and logic.

We believe it is of interest to the database, verification and business process communities.

Thank you!

Example: Negotiation Round as Service

- Pre-condition:

negotiating

- State rules:

$\neg$**negotiating** $\leftarrow \exists$ o,p,q,a R_{QUOTE}(o,p,q,a,a,ω)

approval_pending $\leftarrow$
$\exists$ o,p,q,a R_{QUOTE}(o,p,q,a,a,ω) $\wedge$
$\exists$ c,n R_{ORDER}(o,c,n) $\wedge \neg$**CUSTOMER**(c, ”preferred”, ”excellent”)

insertion rule

Legend: database attribute state

Unguarded State Rule

- For preferred customers with better than good credit, no approval is required once the negotiation converges:

$$\textbf{no\_approval\_needed} \leftarrow \exists\, o,p,q,a\ \ R_{QUOTE}(o,p,q,a,a,\omega) \wedge \exists\, c,n,r\ \ R_{ORDER}(o,c,n) \wedge \textbf{CUSTOMER}(c, \text{"preferred"}, r) \wedge\ \ r > \text{"good"}$$

unguarded usage of r

- This particular case can be "fixed" by adjusting spec: add customer rating as artifact attribute and fill its value in via a service.

- Cannot be systematically done, due to undecidability result.

Blocking Pre-runs

A system with only blocking pre-runs has no infinite runs.
➔Verification becomes meaningless: all properties vacuously satisfied.

It is decidable whether all maximal-length pre-runs of Γ are blocking.
PSPACE-complete for fixed-arity schemas, EXPSPACE otherwise.

Immediately from Main Result:

equivalent to asking whether Γ has no (proper, i.e. infinite) runs
equivalent to asking $\Gamma \models$ **false**

Example Business Rule

Assume:

- ORDER artifact extended with flag **paid**, set by a service in charge of collecting payment.

- Shipment service **ship**: original implementation only checks that ORDER flag **done** is set.

- Additional business rule **βship** associated to ship service: implements policy that

  "only platinum customers with excellent credit may get order shipped before payment is received":

βship :

$$\exists\, o,c,n \;\; R_{ORDER}(o,c,n) \wedge \textbf{CUSTOMER}(c, \text{"preferred"}, \text{"platinum"}) \;\vee\; \textbf{paid}$$

Verification Under Business Rules

Given: Artifact system Γ, property ϕ, set **B** of business rules.

We say that

- Γ satisfies ϕ **under B** iff Γ' satisfies ϕ in the standard way
  where Γ' is obtained from Γ by adding each business rule as a pre-condition conjunct to its service.

- A rule is **guarded** if it is guarded when viewed as pre-condition.

Corollary: Verification under business rules is decidable for guarded artifact systems, properties and business rules.

Redundant Business Rules

Given artifact system Γ and business rule set **B**:

Does rule $\beta_s \in B$ "have any effect" on Γ?

i.e. Does β_s exclude any run of Γ?

Corollary: Checking business rule redundancy is decidable for guarded artifact systems and business rules.