Interaction-Oriented Programming

Concepts, Theories, and Results on Commitment Protocols

Munindar P. Singh

singh@ncsu.edu

Department of Computer Science
North Carolina State University
http://www.csc.ncsu.edu/faculty/mpsingh/

The Basic Problem

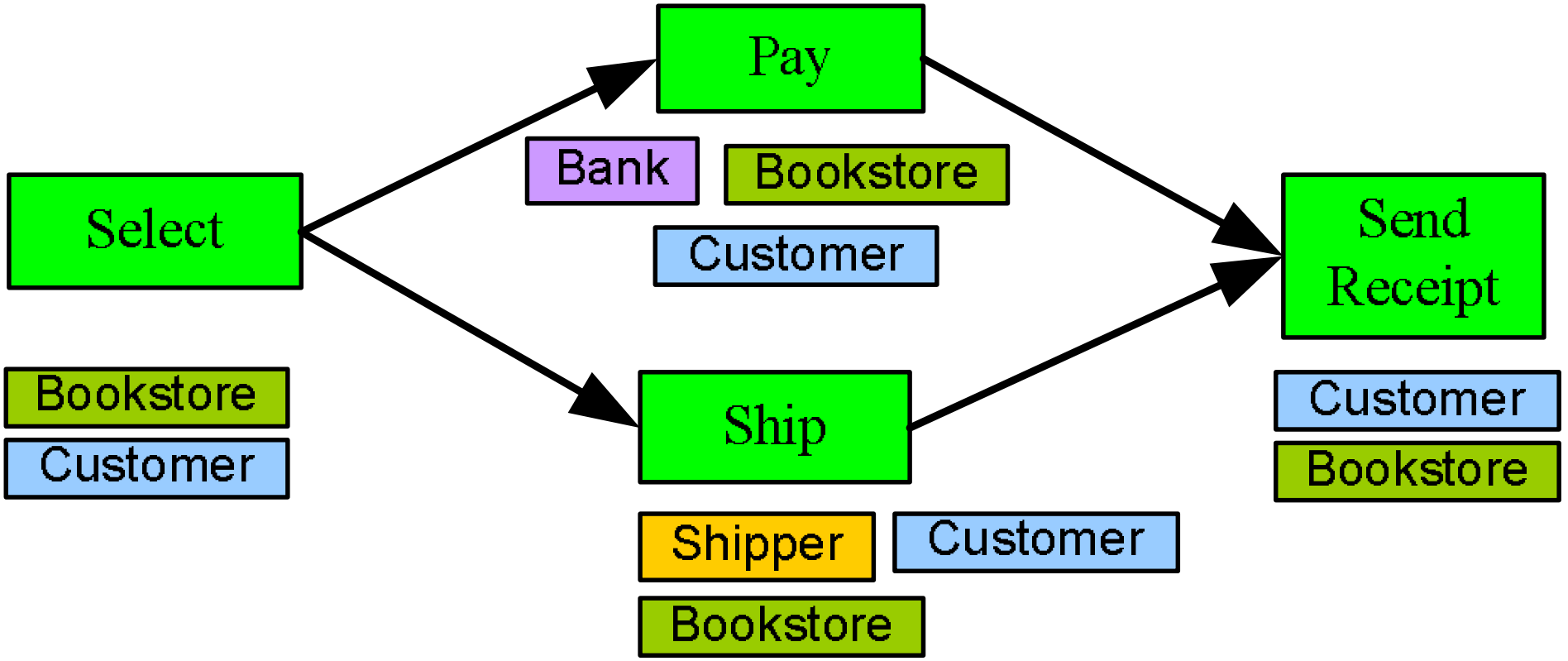
- Everyone acknowledges the importance of open systems
  - Autonomous
  - Heterogeneous
  - Dynamic
- Yet dominant methods assume
  - Closed environments
  - Fixed administrative domains

A Brief History of Programming

Key abstractions

- **Applications:** Procedures
- **Workflows:** Directed graphs
- **Messages:** Objects
- **Conversations:** Agents
- Contentful **Interactions:** Protocols

Traditional View of Process

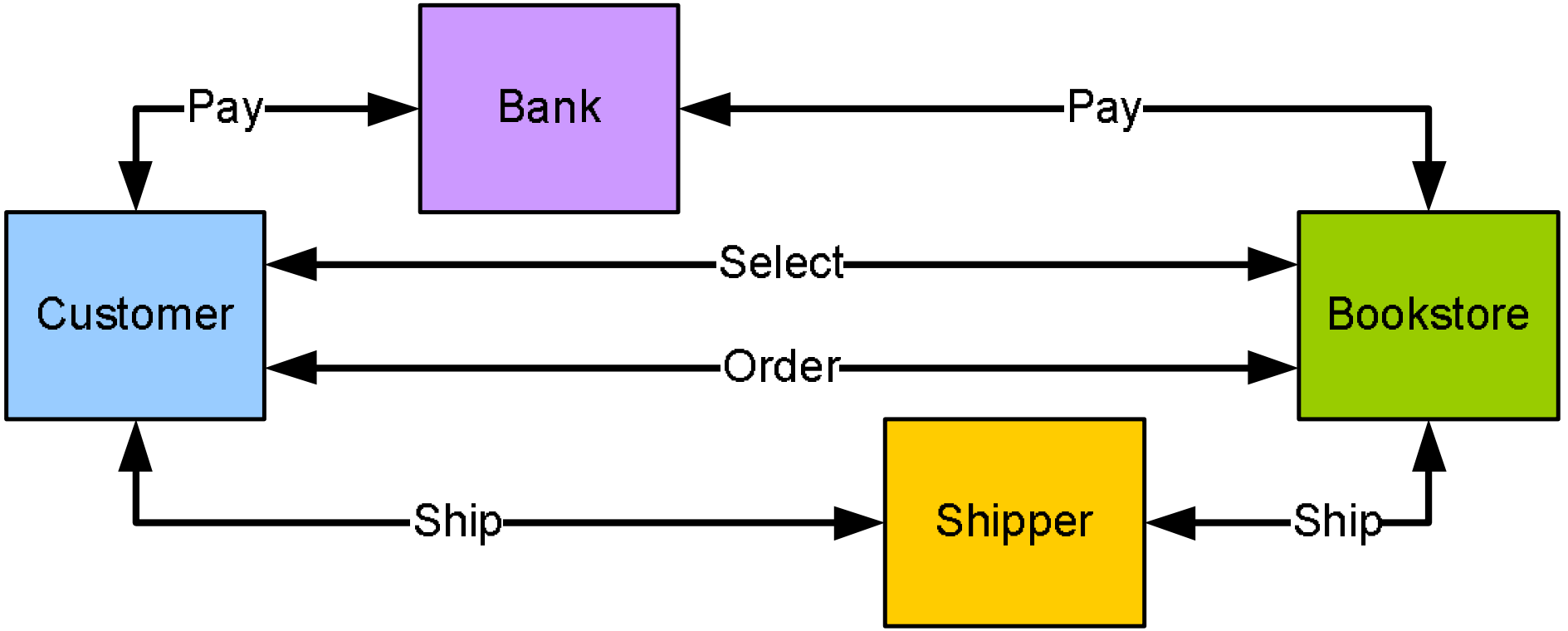


The Essential Tension

- *Reusability*
  - Context freedom
  - Encapsulation
- *Usability* (usefulness)
  - Context sensitivity (organizations, laws, ...)
- Main idea
  - The components have a life of their own
  - Interactions matter

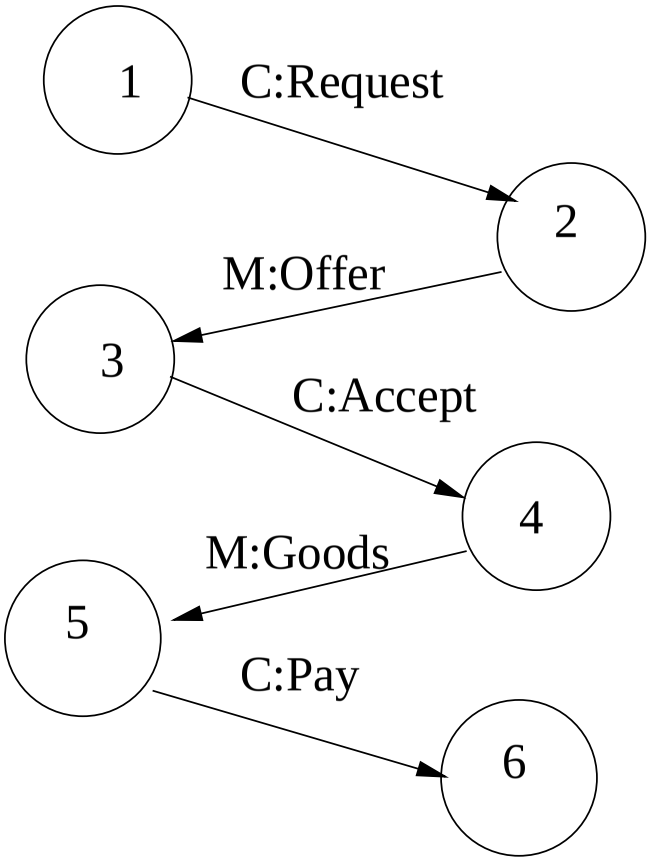
Process = Protocol + Policies

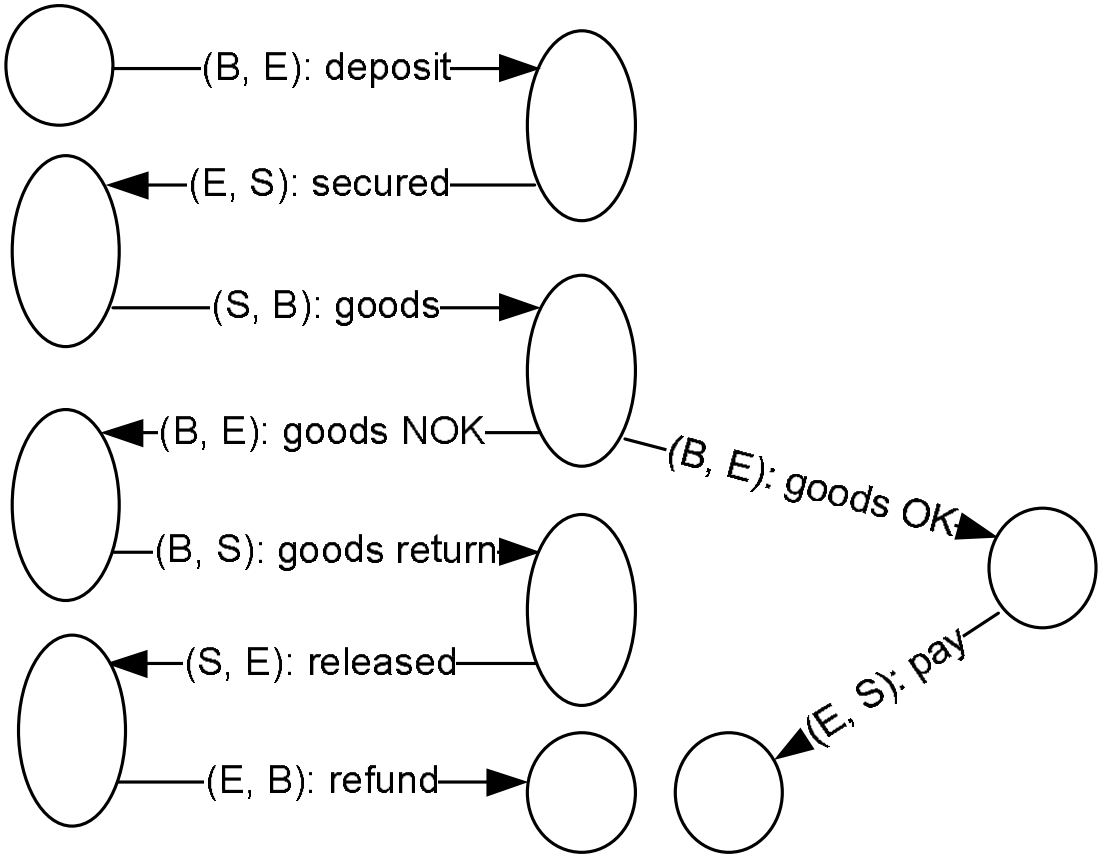
Each partner's policies



Protocols: NetBill and Escrow

Roles; message vocabulary





Protocols in Industry

- Increasing interest
  - IOTP, Escrow, SET, NetBill, ...
  - RosettaNet: 107
  - ebXML
  - TWIST, FIX
- Legally binding
- Shortcomings
  - Two party, request-response
  - No formal semantics
  - Rigid enactment

Commitments: Atoms of Contracts

- Like a directed obligation
  - Debtor
  - Creditor
  - Condition
  - (Precondition)
- Contextual
- Manipulable: assign, delegate, release . . .

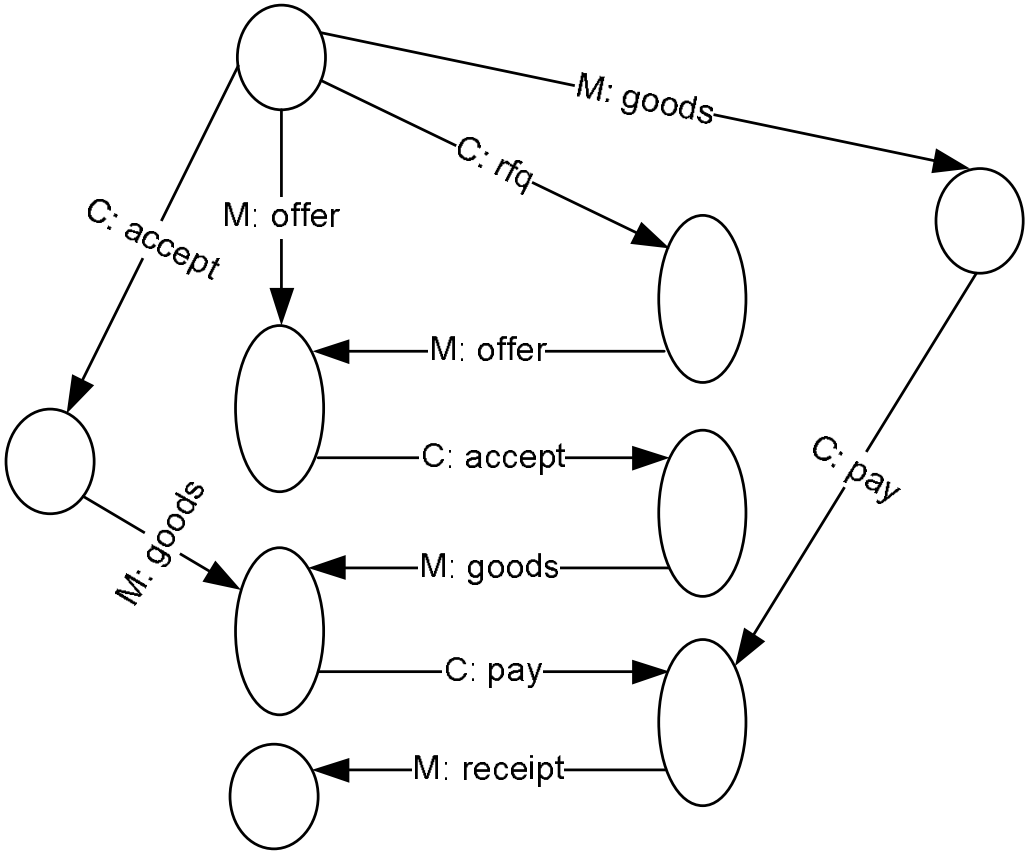
Commitment Protocols

- Roles
- **States**
  - Domain propositions
  - Commitments
- Messages
  - **Action theory:** effects on states
  - Transitions by logical inference

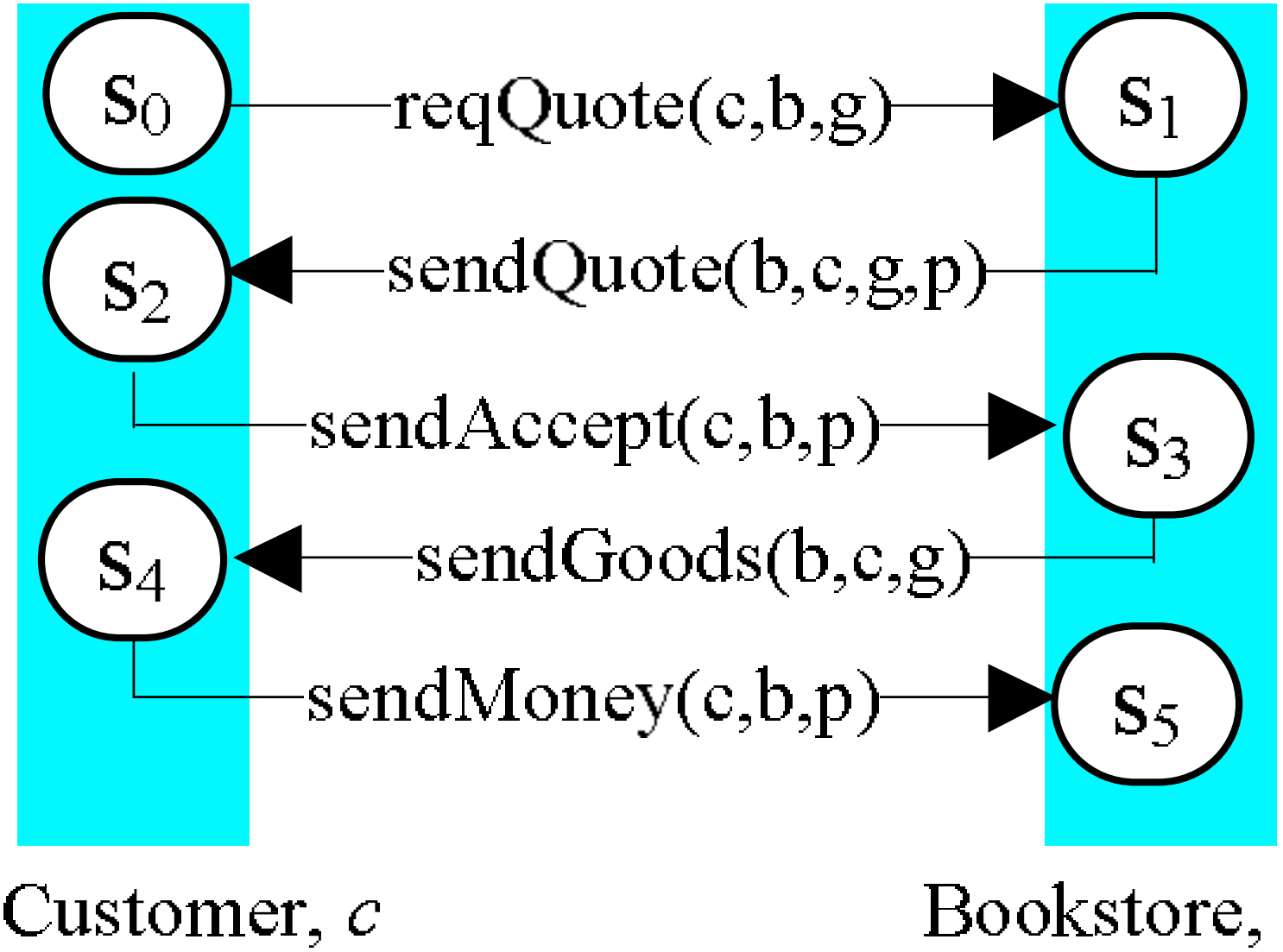
Reusable assets

Compiled Commitment Machine

More flexible than hand-made FSM



Simple Scenario and Example Run

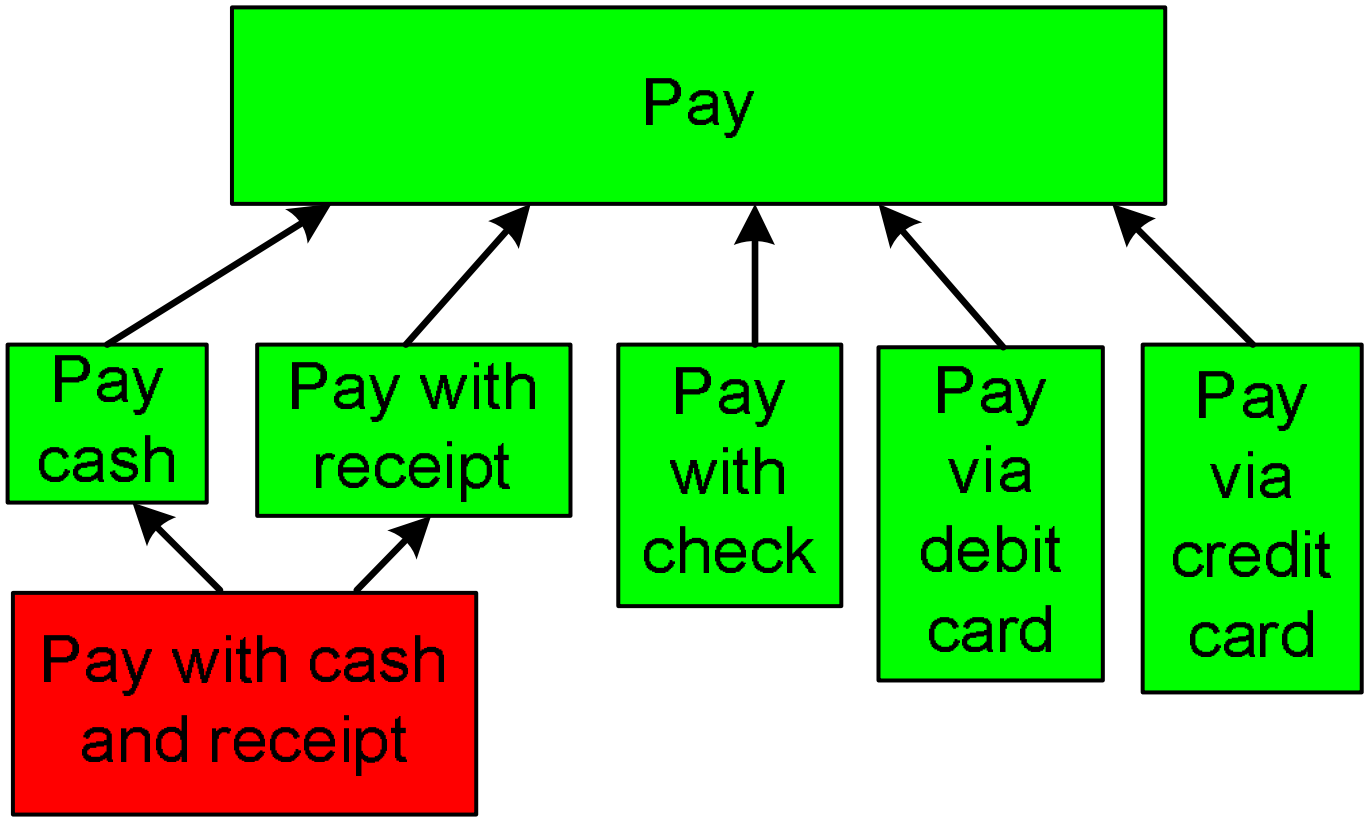


Challenge: Modeling

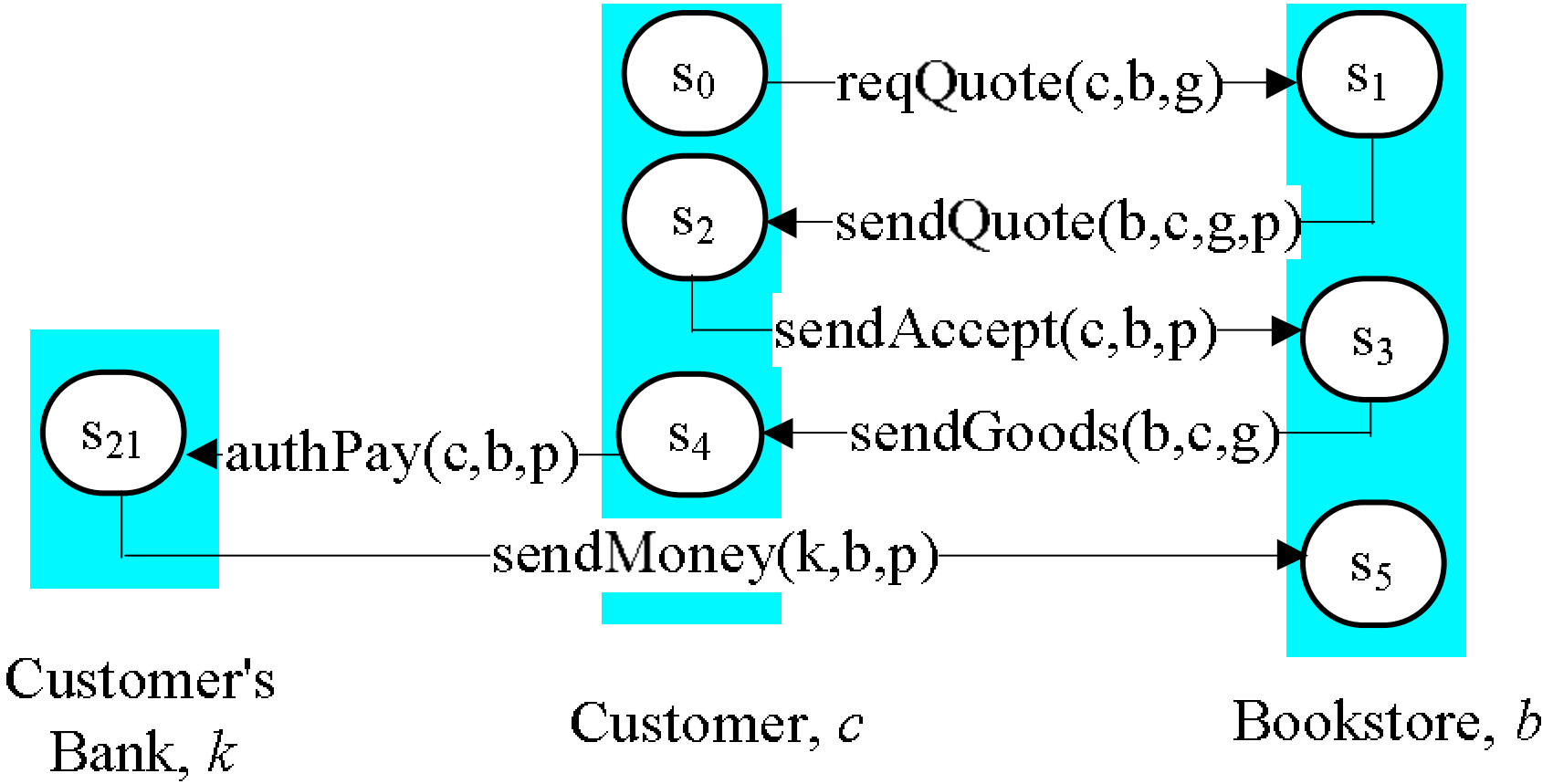
- *Refinement:* pay by credit card
- *Extensibility:* verify C's age
- *Adjustment:* payment before shipping
- *Restructure:* introduce shipper or bank

Refinement of Protocols

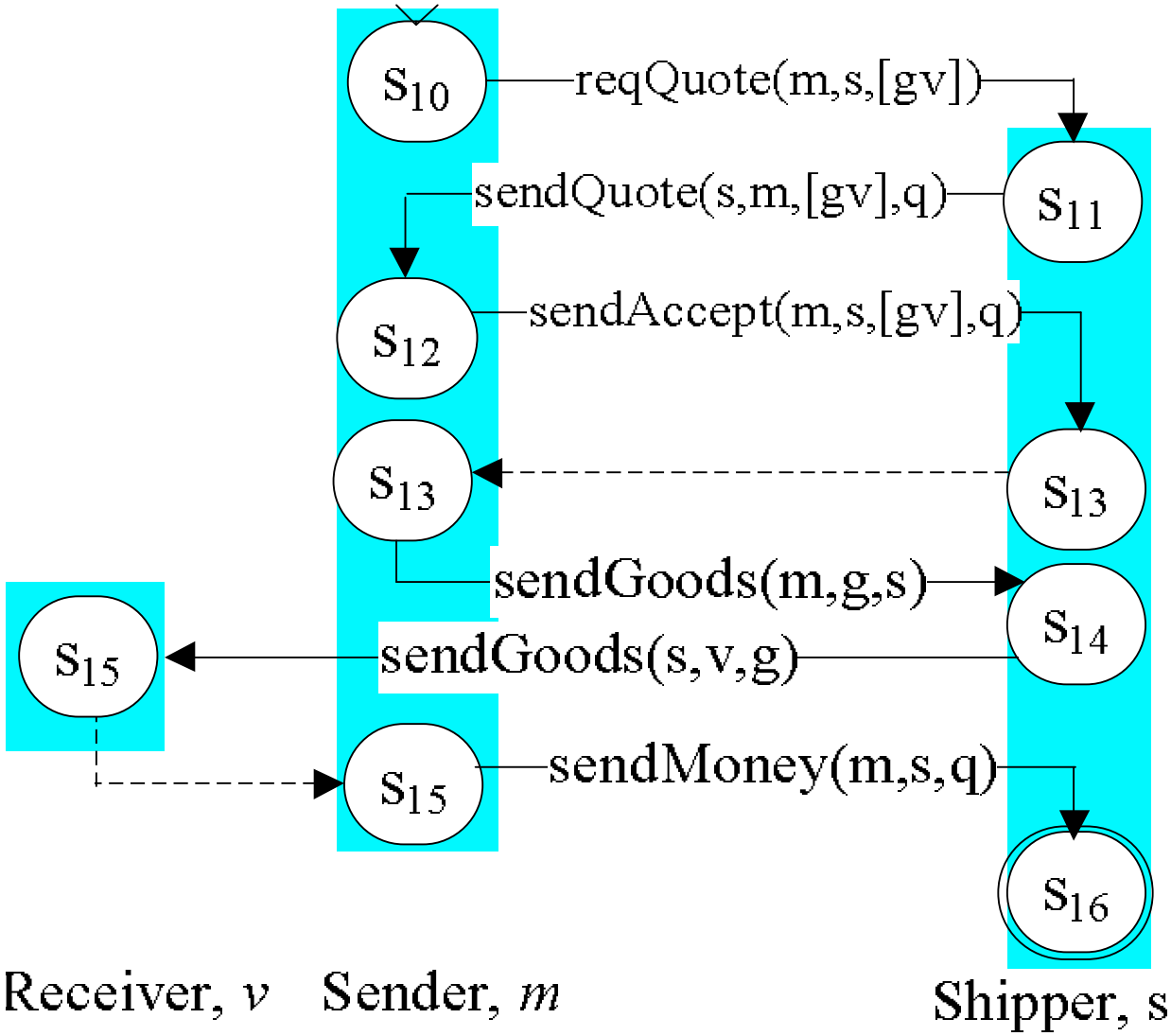
- *Functional:* pay versus ship
- *Nonfunctional:* payer trusts payee or not



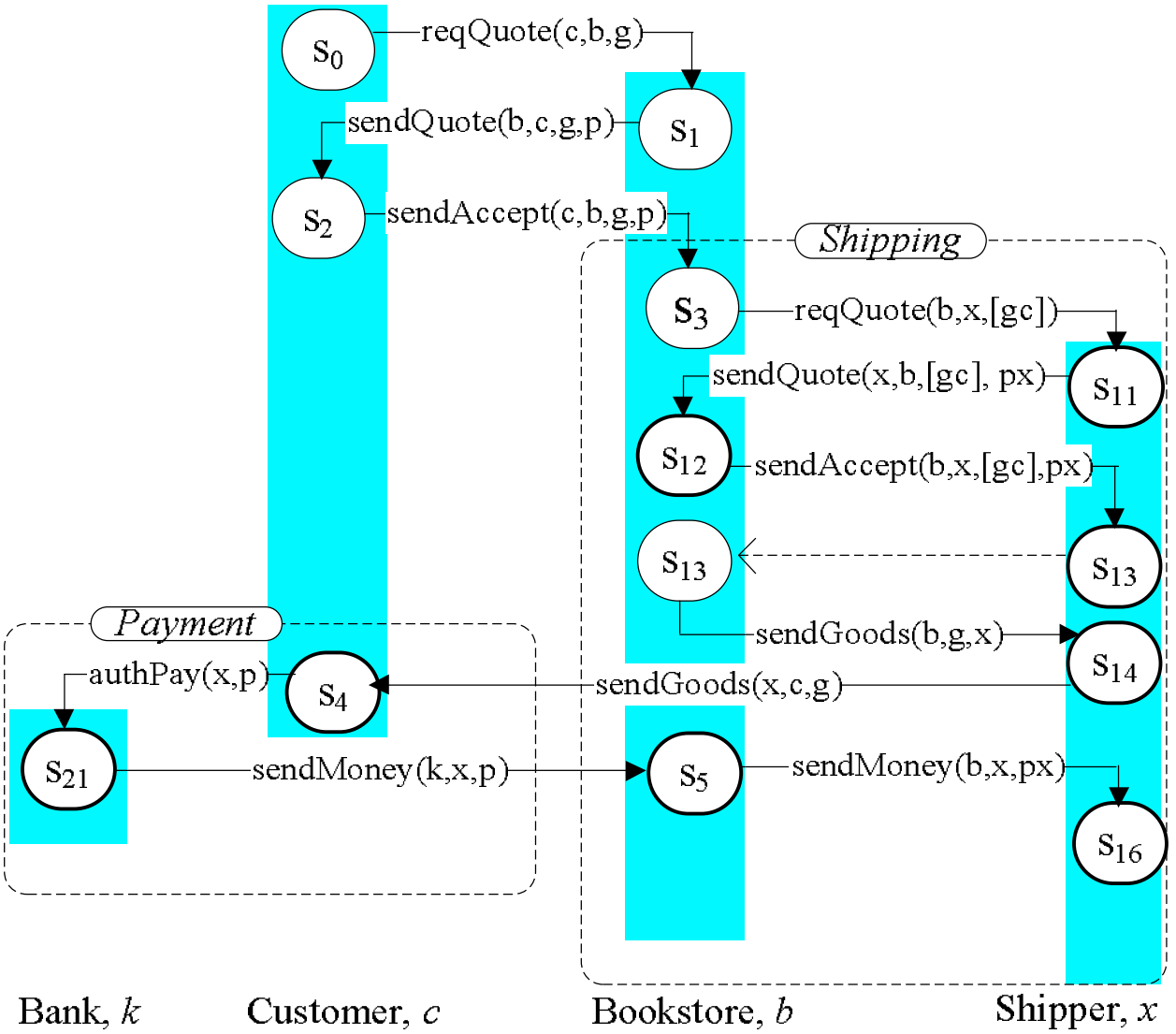
Example Run: Pay via Bank



Example Run: Shipper Protocol



Example Run: Composed Purchase



Challenge: Enactment

- *Adaptivity*
  - Ship before payment, if C trusted
- *Exceptions*
  - Remind, complain, sue, refund, . . .
- *Exploiting opportunities*

Challenge: Contextualization

- Correctness depends on the context
- Context can be social, legal, institutional
  - Advance payment if low trust
  - Check age for tobacco
  - Uniform Commercial Code

Specification in C+: Purchase

Elaboration tolerance is key

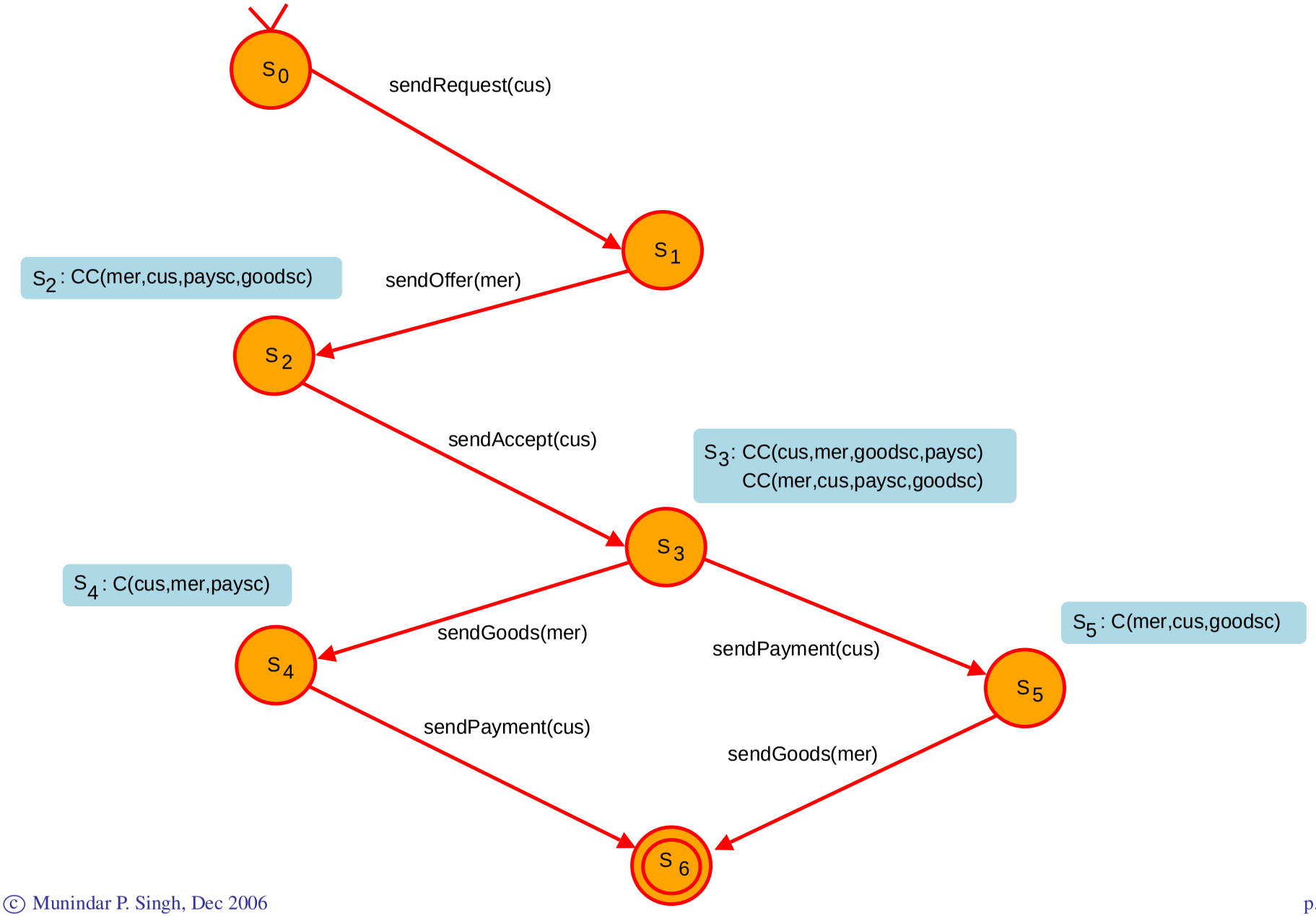
```
sendRequest(role), sendOffer(role),
sendPayment(role) :: exogenousAction;

goods(role, role), pay(role, role) :: inertialFluer

initial, final :: sdFluent.

sendGoods(mer) causes goods(mer, cus).
sendGoods(mer) causes
  cdischarge(cus, mer, goodsc, payc)
  if ccommitment(cus, mer, goodsc, payc).
...
```

Purchase (P_{base})



Transformers

Encode handling specific element of context

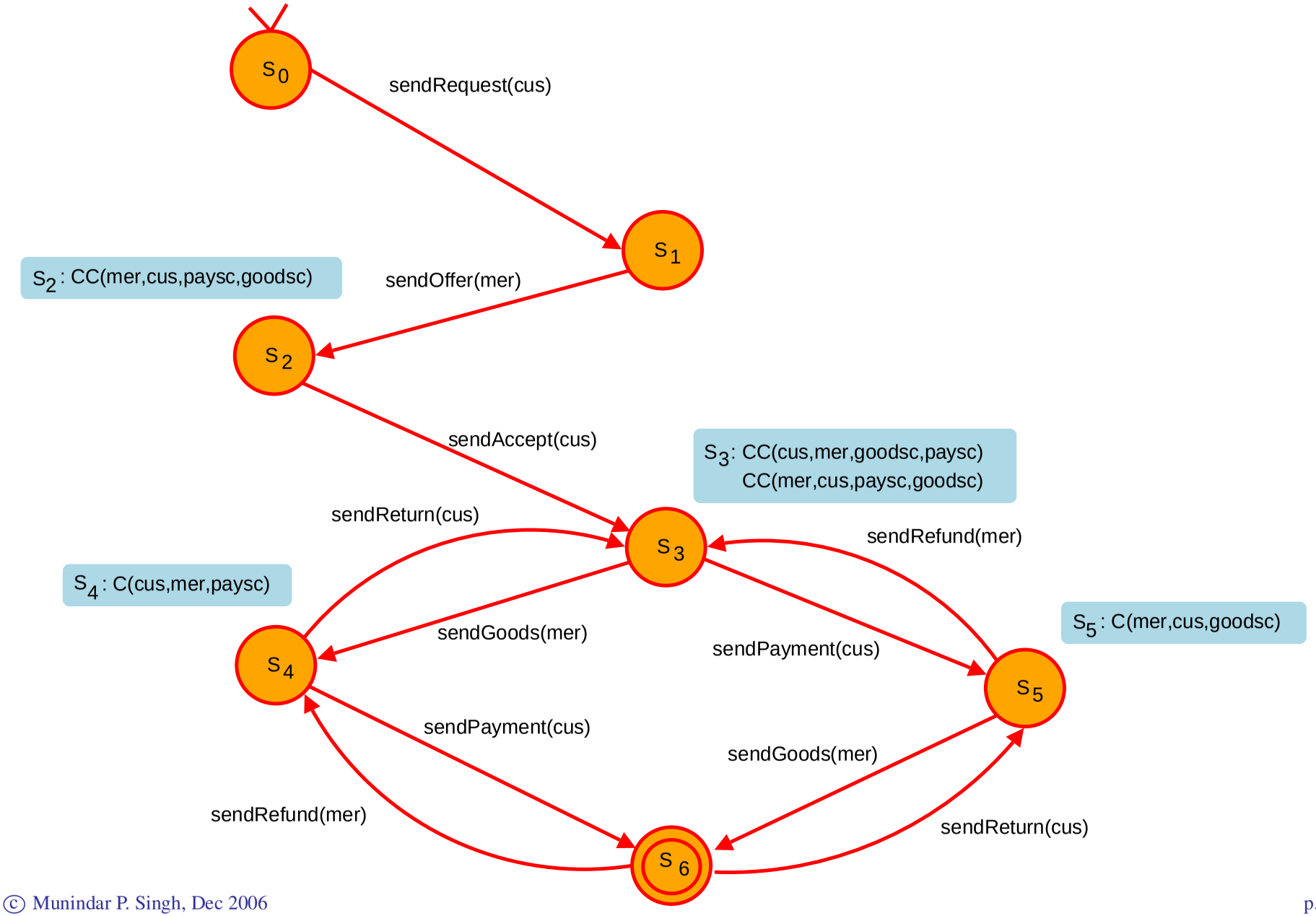
- Like protocols, specified in *C+*
- Append to protocol
- Multiple transformers can be applied

Return-Refund Transformer

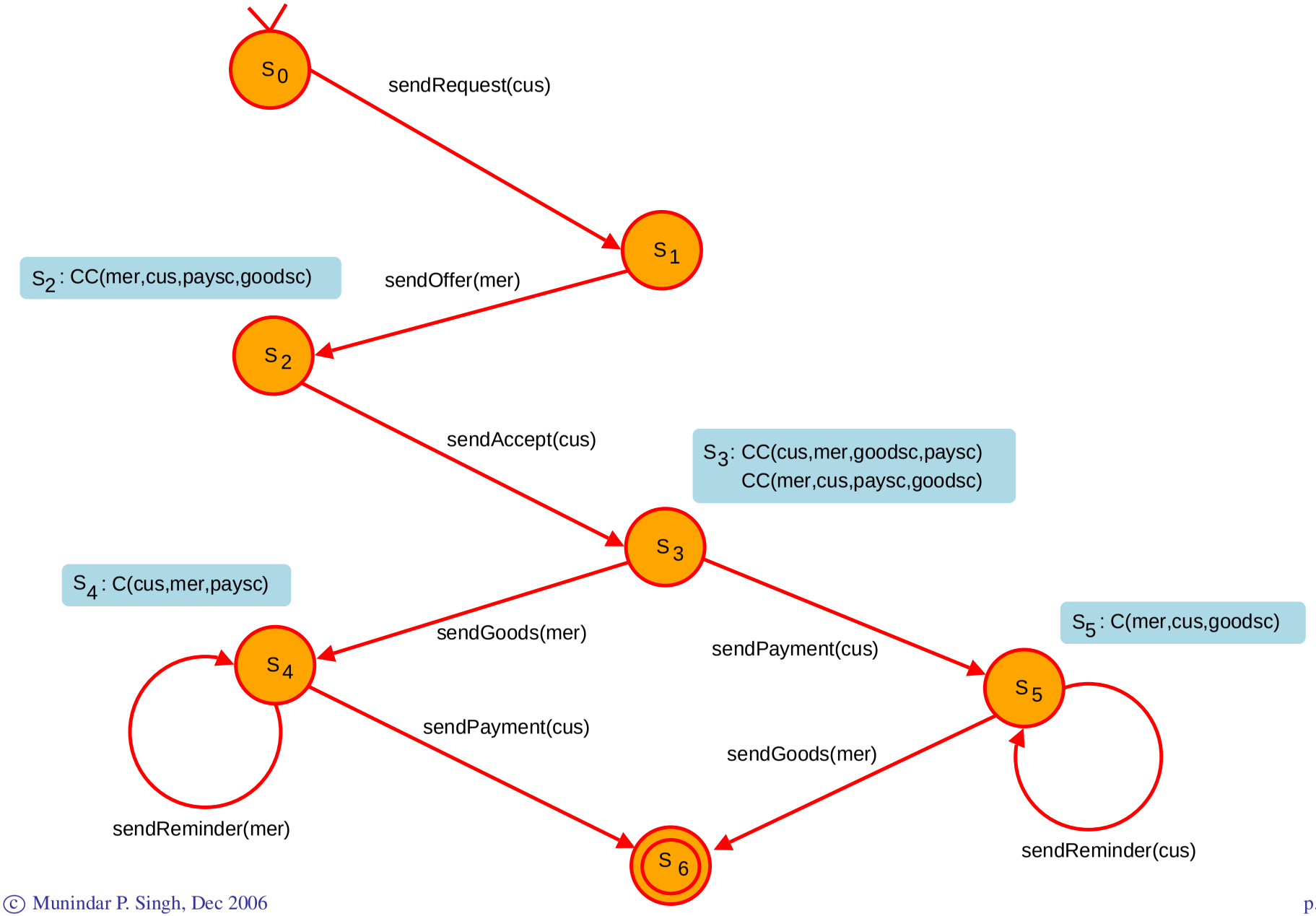
```
sendReturn(role) :: exogenousAction;

sendReturn(customer) causes
  create(merchant, customer, goodsc)
  if pay(customer, merchant)
     & goods(merchant, customer).
sendReturn(customer) causes
  ccreate(customer, merchant, goodsc, payc)
  if commitment(customer, merchant, payc)
  & goods(merchant, customer).
...
```

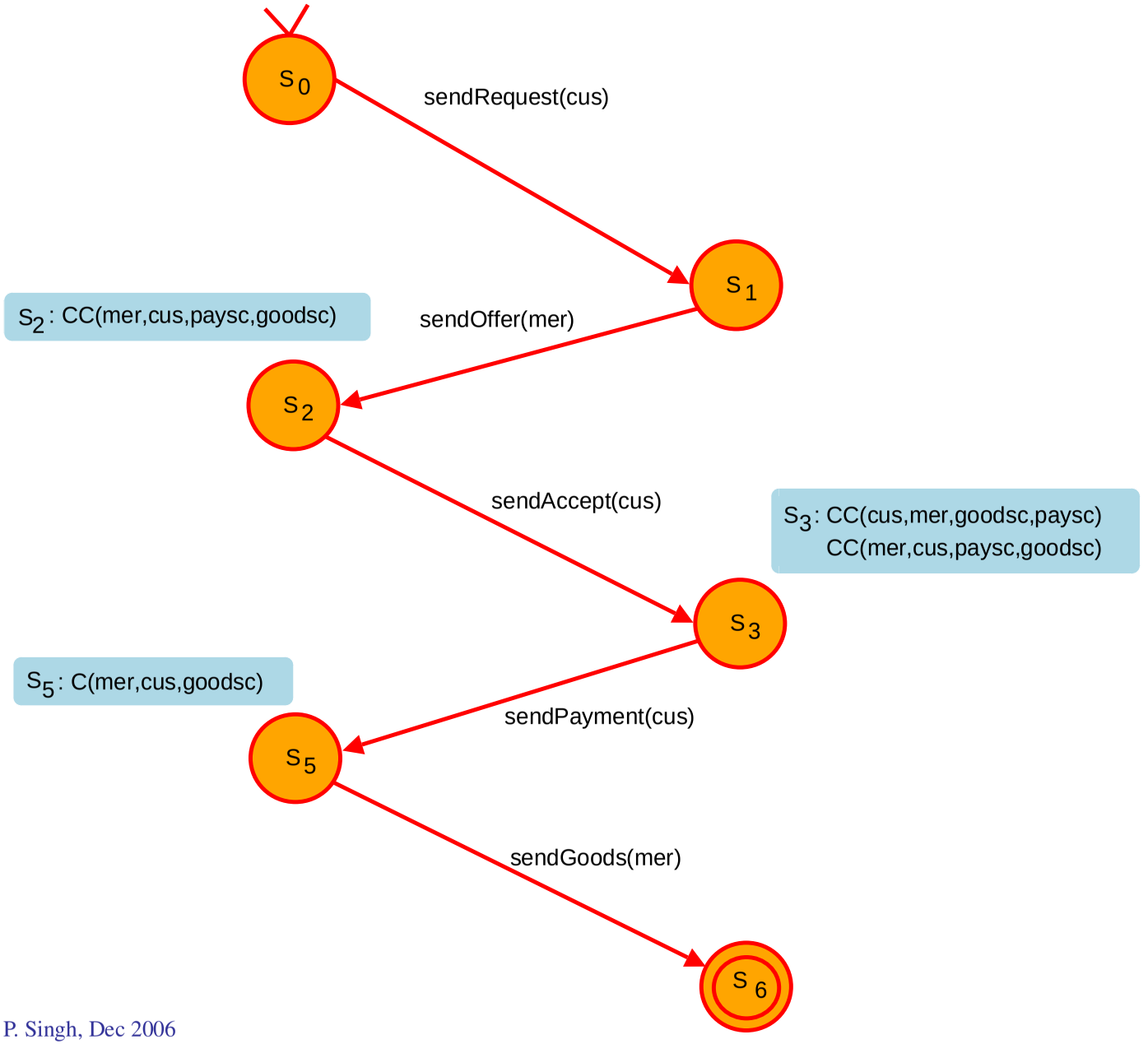
Purchase With Return-Refund (P_{rr})



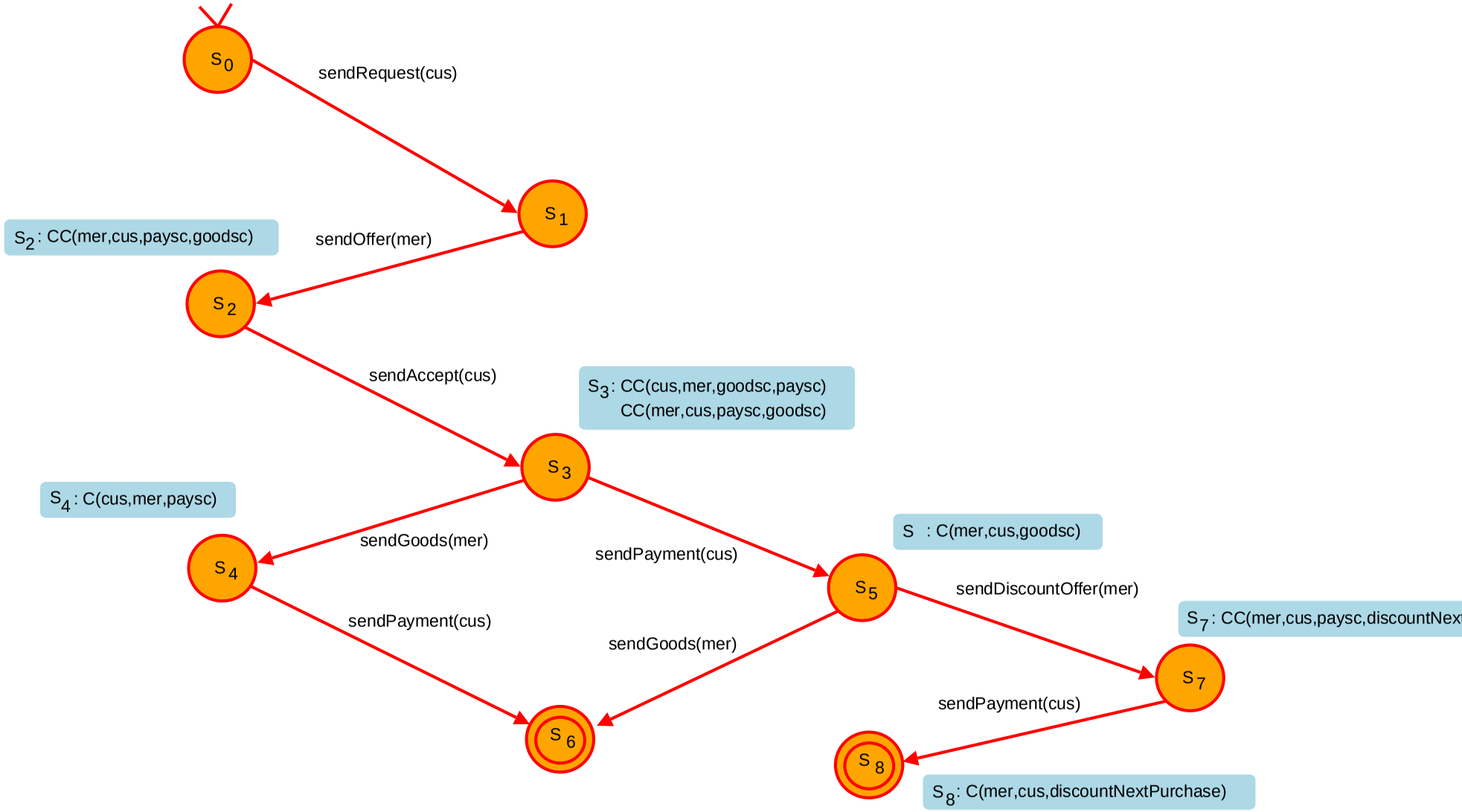
Purchase With Reminders (P_{rem})



Purchase With Pay First (P_{pf})



Purchase With Discount (P_d)



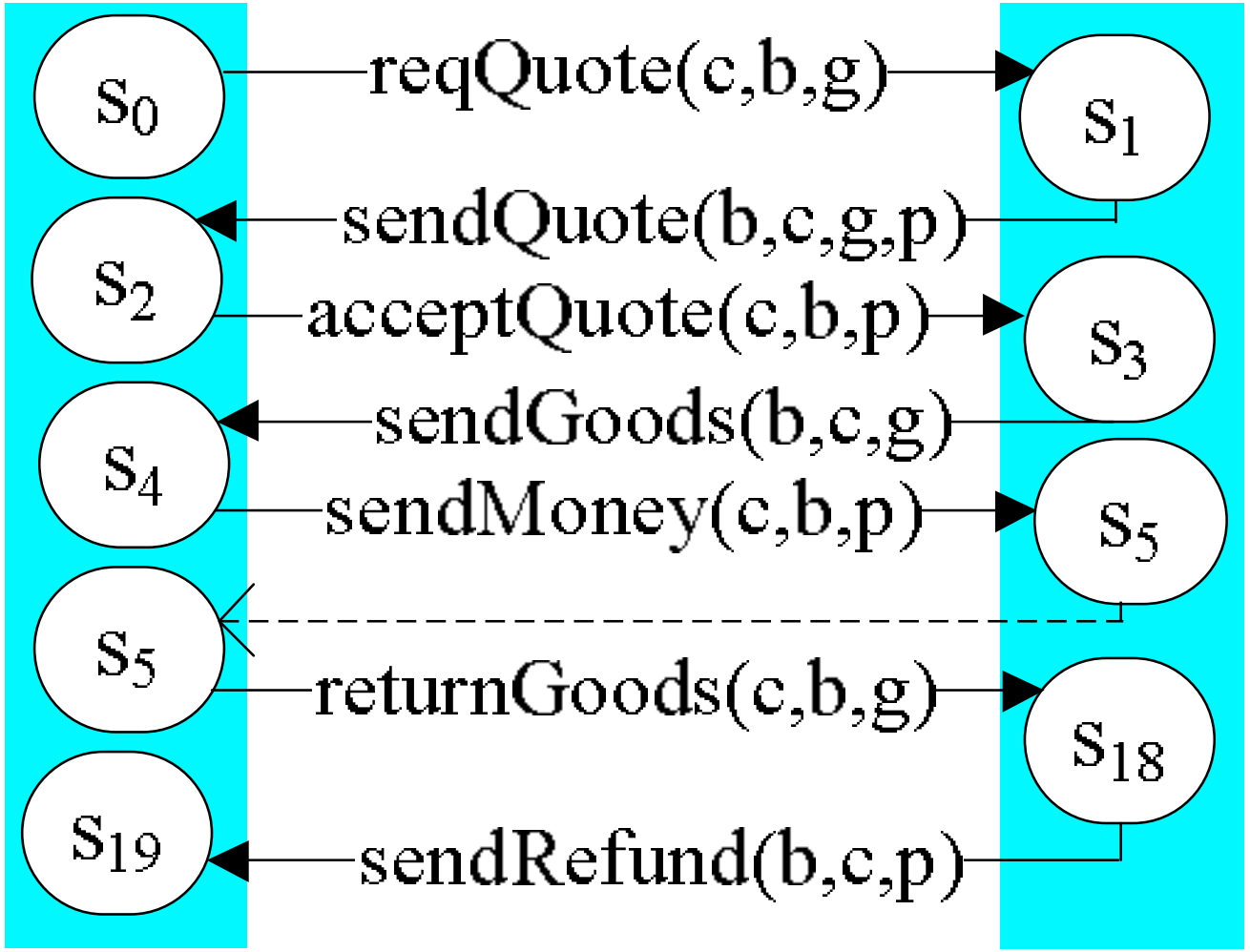
Conformance and Interoperability

Agents adapt protocols and apply local policies

- Conformance wrt to a protocol
  - Generate no runs not in the protocol
- Interoperation wrt each other
  - Take no action that may cause peers to hang
  - Make compatible choices

Enacting Protocols in Organizations

Organizations specify, monitor, verify compliance; enable responses to exceptions



Directions

- Operational patterns
  - Time out, remind, garbage collect, . . .
  - Manipulate commitments
- Methodologies, e.g., enhancing Tropos:
  - Cover functional reqs via protocols
  - Refine protocols for nonfunctional reqs
  - Identify agent policies and context

Protocol Semantics and Pragmatics

Combining the Platonic and the Aristotlean

- *Repository* and tools
- *Recombination*
  - Exploit semantics
  - Add back to repository
- *Collective intelligence:* social knowledge about
  - Correctness
  - Contexts where (in)appropriate

Summary

- Interaction as first class citizen
- Protocols versus policies
- Semantics enables
  - Refinement
  - Composition
  - Contextualization
- Subtle notions of correctness: conformance, interoperability

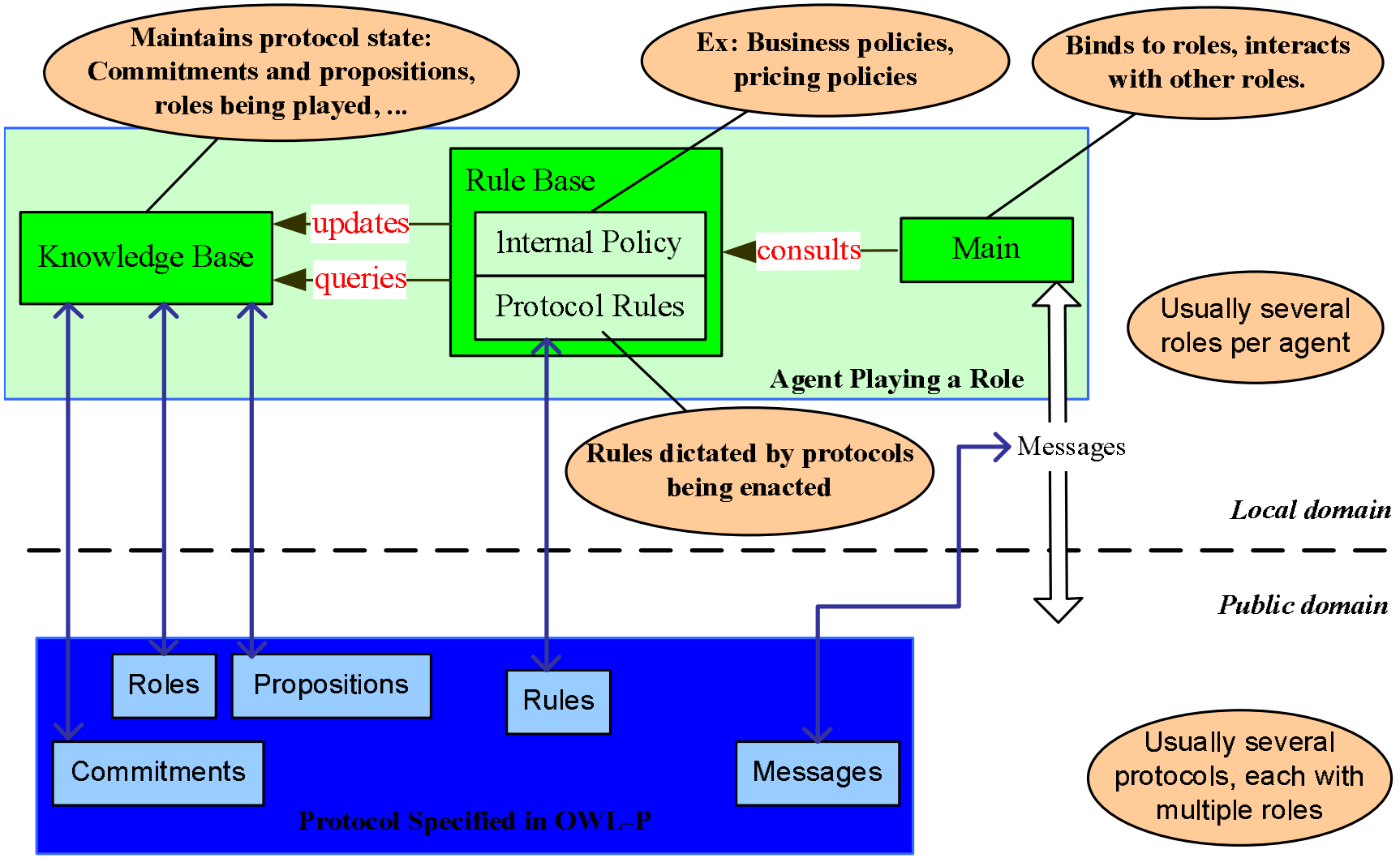
Papers on this Topic

- Recent papers in AAAI, AAMAS, ICWS, ICSOC, IJCAI, SCC address parts of the above vision
- "Agent Communication Languages: Rethinking the Principles." *IEEE Computer*, 31(12):40–47, Dec 1998
- "Reasoning About Commitments in the Event Calculus: An Approach for Specifying and Executing Protocols." *Annals Math & AI*, 42(1-3), 2004
- "Verifying Compliance with Commitment Protocols." *J. Autonomous Agents & MAS*, 2(3):217–236, Sep 1999
- "An Ontology for Commitments in Multiagent Systems." *AI & Law*, 7:97–113, 1999

Thanks!

http://www.csc.ncsu.edu/faculty/mpsingh/

Architecture



Implementation Agenda

Bridging the gap between current architectures (e.g., ESB or enterprise service bus) and user needs

- Capture and generalize scenarios known to be of user interest
- Develop a repository of validated protocols
- Extend and incorporate current tools: OWL-P (protocols) and MAVOS (multiagent virtual organization system)

Conceptual Model for Organizations