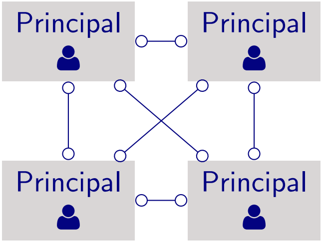
Before Computing: Decentralization was Natural

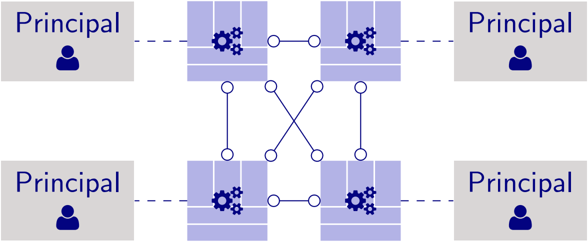
Long-lived engagements between autonomous principals



- In business, health, finance, ...
- Conceptually decentralized

Multiagent Systems: Agents Help Principals

Realize a decentralized, loosely coupled system to promote flexibility
Heterogeneous agents encode decision making of their respective principals



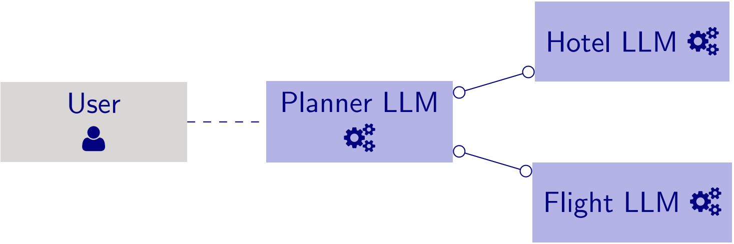
What we need

(Operational) protocols Constraints on the ordering and occurrence of interactions (messages)

Meaning The import of an interaction on the social tier

Agentic AI: Multiagent Paradigm

Flexible, generative AI-powered agents that make real-world decisions



Inflexible coordination via workflows (task graphs) defeats flexibility

- Task graphs: obsolete, rigid notation
- Need flexible operational models
- Need models based on *interaction meaning*

Interaction-Oriented Programming (IOP)

Empower stakeholders and programmers

Method

- Model a multiagent system in terms of interactions
- Compose and verify models
- Implement agents independently on the basis of models

High-level abstractions that

- Reflect stakeholder intuitions and
- Let programmers focus on the business logic

Communication Protocols

A protocol defines how the agents ought to communicate with one another

- What are the main requirements for protocol specifications?
- How can we specify a communication protocol?
  - Roles (abstracting over agents)
  - Message schemas, i.e., allowed content
  - Message emission and reception, point-to-point or multicast, between specified roles
  - Constraints on message occurrence
  - Constraints on message ordering
- Agents participate in a protocol by playing a role in it
- How can we develop agents suitable for a role?

Challenges to the Correctness of Protocols

Regardless of specification language

Distribution: different parties observe different messages, i.e., each lacks remote knowledge

Asynchrony: different parties observe messages in inconsistent orders

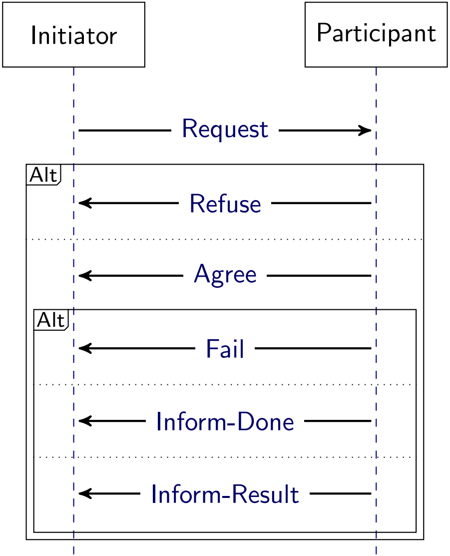
- FIFO channels don't preclude such inconsistency
- FIFO channels are a restrictive assumption

Sequence Diagrams

Well-known specification approach

- Originally used for object-oriented programming
- Our needs: closest to message sequence charts
- An intuitive way to express interactions
  - Expresses global view consolidating local perspectives
  - Excellent for describing possible interaction instances
  - But beware the pitfalls ...
- Support (potential) validation checks
  - Formalizing semantics is not obvious: multiple approaches
- Standardized in UML 2.0 as Sequence Diagrams
  - Caveat: Arrowheads and other details of these notes don't necessarily match UML

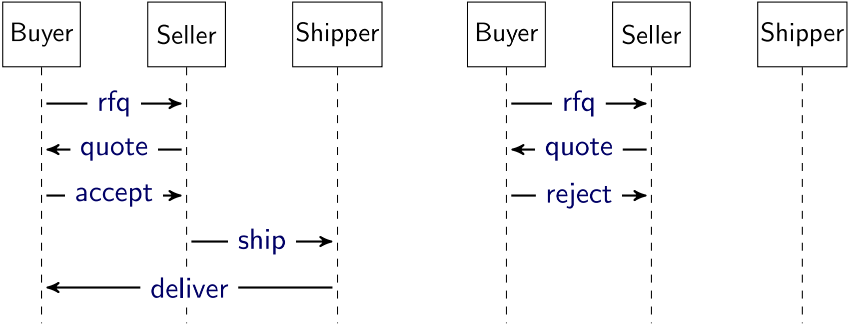
FIPA Request Interaction Protocol as a Sequence Diagram



- Roles: INITIATOR and PARTICIPANT
- Messages
  - *request*, *agree*, *refuse*, *failure*, an *inform-done*, or an *inform-result*
- Ordering and occurrence
  - *refuse* or an *agree*
  - *agree* followed by a detailed response: *failure*, *inform-done*, or *inform-result*
  - *agree* is required only if the INITIATOR asked for a notification

Purchase: Example Protocol

Notice the hand off pattern, indicative of delegation (revisited later)



Exercise: Sequence Diagrams for Possible Enactments

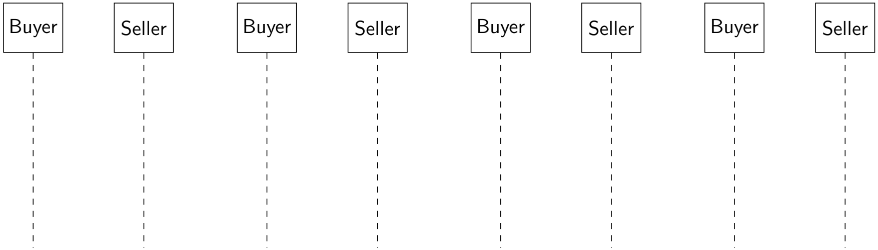
Show crossing messages

Intuition about protocols

A protocol is the set of its possible enactments

Scenario

- Buyer sends a *purchase order* to Seller, specifying an item and price
- Seller sends the *item* to Buyer
- Buyer sends a *payment* to Seller

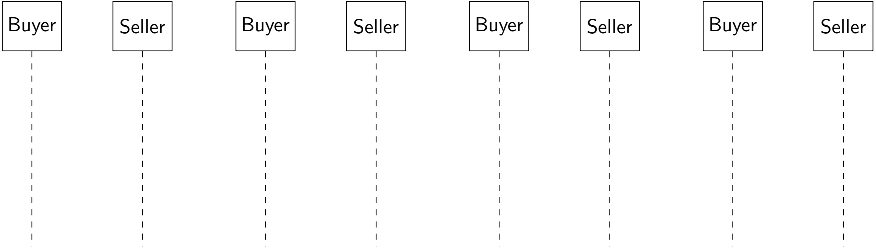


Exercise: Sequence Diagrams for Possible Enactments

Race conditions

Scenario

- Buyer sends a *purchase order* to Seller, specifying an item and price
- Seller sends the *item* to Buyer
- Buyer sends a *payment* or a *cancel* to Seller



Sequence Diagrams for Multiagent Systems

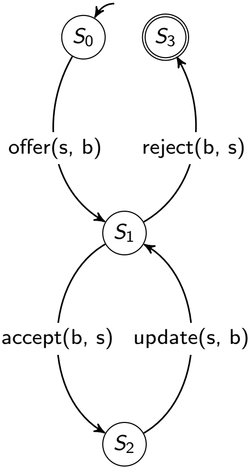
No!

- No internal reasoning
  - No private predicates in guards
- No method calls
  - No self calls
- No synchronous messages
  - No business puts itself on indefinite hold waiting for its partner to proceed
- No causally invalid expectations
  - No *nonlocal* choice
    - No nonlocal choice that matters
  - No control of incoming message occurrence or ordering
  - No dependence on occurrence or ordering of remote message emission or reception
  - No reliance on ordering across channels
    - No reliance on ordering within a channel unless warranted

Example Finite State Machine Representation

Part of a purchase protocol that deals with making offers

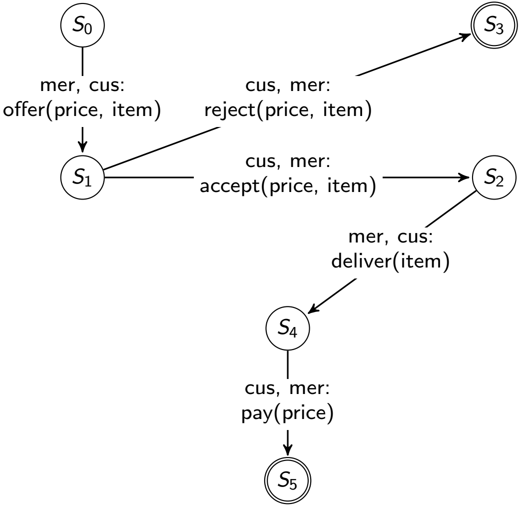
- Roles: buyer (b) and seller (s)
- Initial state, with arrow
- Final state, double barred
- Transitions labeled with messages
    - Specify legal message flows



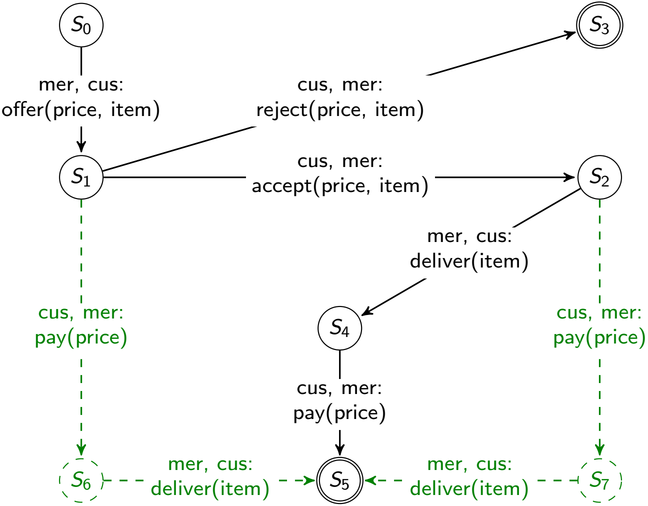
Finite State Machine (NetBill Protocol)

Legitimate protocol: specifies interactions, not internal decision making

- Roles: merchant (mer) and customer (cus)
- Transitions: messages sender, receiver
- Enactment: *reject*
- Enactment: *accept*, *deliver*, *pay*
- Correctness: purely operational terms (sequences of messages, not meanings)
  - Excludes legitimate enactments (next page)



State Machine Example: Generalized



Exercise: FSM for a Protocol

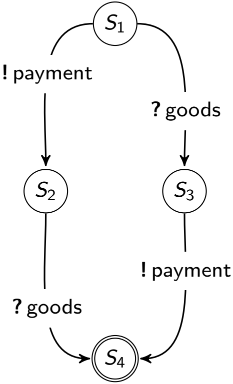
Scenario

- Buyer sends a *purchase order* to Seller, specifying an item and price
- Seller sends the *item* to Buyer
- Buyer sends a *payment* or a *cancel* to Seller

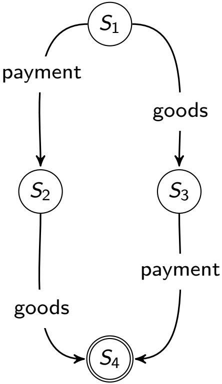
Protocols and Roles

Protocol: shared view; roles: each local view
Here, roles are *communicating* state machines

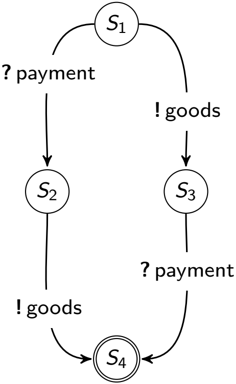
The Buyer Role



Trade Protocol



The Seller Role



Evaluation of the FSM Representation

Does not account for meanings of messages

- Flexibility: limited by over-specifying operations (message order and occurrence)
- Compliance checking: easy since the protocol is explicit about operations
  - Failure to comply may not indicate an application-level problem

Need for reasoning about interaction meaning

- Implicit meanings invite inconsistent interpretations
- To capture meanings requires declarative model of operations

Applying State Diagrams in Our Setting

Behavior descriptions, but of *social behavior*—to be introduced

- In general, sequence diagrams should describe interactions whereas state diagrams should describe internal behaviors
  - Traditional sequence diagrams often step into internal details
  - Traditional state diagrams are low-level, just as traditional sequence diagrams are, only more so
- Our state diagrams apply to a *social* state, which can be affected through messages described by sequence diagrams
- Consider state diagrams as describing the progression of the social state of a multiagent system
  - We can express this from an outside, i.e., a public or an institutional, as opposed to an implementation perspective
  - A research challenge is to ensure the social state remains sufficiently aligned across the interacting parties
  - For a properly designed multiagent system, its social state ought to progress consistently