Outline

Specification Approaches
Outline

Specification Approaches

- Message Sequence Diagrams
- Protocols and Policies
- State Diagrams
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
Method Invocation in Object-Oriented Programming

Only one thread of control; objects exchange messages
Message Emission and Reception

Independent threads of control; autonomous parties exchange messages, asynchronously sending and receiving

- Customer (c) requests a quote from Merchant (m).

Diagram:
- Customer (c) sends a Request for Quote to Merchant (m).
- Merchant responds with a quote.
The Alternative Block

Nondeterministically choose and execute any fragment whose guard is true

\[ \text{alt} \]

\[ \text{[Yes]} \quad \text{Accept Quote} \]

\[ \text{[\neg Yes]} \quad \text{Reject Quote} \]

\[ \text{Provide Quote} \]

\[ \text{Provide Quote} \]

\[ \text{c:Customer} \quad \text{m:Merchant} \]
The Optional Block

Modeling error here: Showing internal detail (free (spare time)) in a protocol
The Loop Block

Usually bounded in our examples

\[\text{c:Customer} \rightarrow \text{m:Merchant}\]

Provide Goods

Pay Charges

[5 times]

Offer

Counter Offer

Munindar P. Singh (NCSU)
Purchase (Just the Happy Path)

Notice the hand off pattern, indicative of delegation
The Parallel Block

Provide Goods

Pay Charges

Deliver Goods

Request Payment
Exercise: Diagramming Precedence

- Four roles: $A$, $B$, $C$, $D$ (could map to the same parties)
- Two messages: $m_{AB}$ and $m_{CD}$ (sender to receiver: distinct parties)
- We would like to assert that $m_{AB}$ precedes $m_{CD}$
All Possible Sequence Diagrams

Given messages from $a$ to $b$ and from $c$ to $d$

$\begin{align*}
a &\neq b \\
c &\neq d
\end{align*}$
Exercise: Which of the Precedence Diagrams are Compatible with Asynchrony?

Invariant outcomes regardless of relative execution speed, communication delays, and no global clock.
Exercise: Diagramming Occurrence and Exclusion

Use guards that refer to message occurrence
If \([m_{AB}]\) occurs then so does \([m_{CD}]\)

- Four roles: \(A, B, C, D\) (could map to the same parties)
- Two messages: \(m_{AB}\) and \(m_{CD}\) (sender to receiver)
- We would like to assert that
  - \(m_{AB}\) excludes \(m_{CD}\)
  - \(m_{AB}\) and \(m_{CD}\) mutually exclude each other
  - \(m_{AB}\) requires \(m_{CD}\)
Properties of a (Point-to-Point) Message Channel

Consider these questions

**Noncreative:** Must a message that is received have been sent before?
- Can we take a system snapshot that violates this property?

**Reliable:** Must a message that is sent be received?
- Can we take a system snapshot that violates this property?

**Ordered:** Must the messages received from the same sender be received in the order in which they were sent?
- In which direction does the information flow?

**Causal:** Must the messages received from different senders be received in the order in which they were sent?
- Can we take a system snapshot that violates this property?
Challenges to Correctness of Protocols

Not specific to sequence diagrams

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

**Asynchrony**: different parties observe messages in inconsistent orders
  - Despite FIFO channels

**Intuitions about correctness**
  - If each party interacts correctly, is the overall behavior correct?
  - If not, our sequence diagram is not *realizable* or *enactable*
  - Is the design of each party obvious?
  - Does the design of the parties preclude some legal enactments?
Outline

Specification Approaches
Message Sequence Diagrams
Protocols and Policies
State Diagrams
Business Protocols

Interactions among autonomous parties, understood at the business level

- *Conversation*: An instance of a protocol
- Operational representations: steps taken
  - Procedural
    - Sequence diagrams
    - State diagrams
    - Activity diagrams
    - Petri Nets
  - Declarative
    - Temporal logic
    - Dynamic logic
    - Information-based specifications
- Meaning-based representations: underlying business transaction
  - Declarative, if captured formally at all
    - Commitment machines
    - Constitutive specifications
Exercise: Identify the Public and Private Components

Process = Protocol + Policies

- Request for Quotes
- Quote
- Accept
- Ship
- Deliver
Exercise: How Might we Modularize Protocols?
Consider Purchase
Modular Business Protocols

- Identify small, well-defined interactions with clear business meanings
- Improve flexibility and concurrency
- Possibly lead to invalid executions
- How can we ensure good properties despite modularity?
  - Begin from a constraint language
  - Standardize modular fragments as patterns, e.g., RosettaNet
Sequence Diagrams for Business Modeling

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
Outline

Specification Approaches
Message Sequence Diagrams
Protocols and Policies
State Diagrams
State Diagrams
Formalized in UML 2.0 from Harel’s statecharts

Generalize over finite state machines

- Condition or guard on a transition
- Superstate (OR-state): being in a substate entails being in the superstate
  - Natural for summarizing states that bear similar meanings and support similar transitions
- Parallel states indicate being in the each of the states at the same time (AND-state)
  - Cartesian product of the individual states
  - Natural for expressing mutually independent components of the state
State Diagram Syntax

- **State 1**
  - **Substate 1.1**
  - **Substate 1.2**

- **Init**
  - Event [guard]
  - Action

- **Substate 2**
Exercise: Diagram the Purchase Protocol
First as we specified and second with concurrent Pay and Ship subprotocols
Exercise: Diagram the Purchase Protocol with Return and Refund
Exercise: Diagram Precedence, Occurrence, Exclusion

Across two messages, $m_1$ and $m_2$
Applying State Diagrams in Our Setting

Behavior descriptions, but of social behavior

- 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 service engagement
  - 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 service engagement, its social state ought to progress consistently