Classifying Exceptions in Agent-Based Protocols: A Thin Line Between Violation and Opportunity

Özgür Kafalı

AILAB, Department of Computer Engineering, Boğaziçi University

CSW’10 - February 21, 2010
Outline

1. Introduction
2. Background
   - Commitments
   - Ontologies
3. Framework
   - Protocols
   - Exceptions
   - Classification
4. Case Study
5. Future Directions
Exceptions Everywhere!

- Agents cannot always work together (like humans!)
- Challenges in open systems:
  - No prior knowledge about the environment
  - No guarantees about the behaviors of agents
  - One agent’s misbehavior affects the entire business (exception)
- In the case of exceptions, agents need to
  - Detect the exception
  - Diagnose the exception (find out its cause)
  - Take proper action to recover from the exception
- However, not all exceptions are equally significant
Example: A Delivery Process
Example: A Delivery Process

pay for book

Normal execution

Classifying Exceptions in Agent-Based Protocols
Example: A Delivery Process

Normal execution

pay for book

pay for delivery
Example: A Delivery Process

- pay for book
- Normal execution
- deliver book
- pay for delivery
Example: A Delivery Process

pay for book

Stock exception
Example: A Delivery Process

- Pay for book
- Supply book

Stock exception
Example: A Delivery Process

pay for book

Delivery exception
Example: A Delivery Process

- Pay for book
- Delivery exception
- Pay for delivery
Example: A Delivery Process

- Pay for book
- Delivery exception
- Pay for delivery
- No Truck

Ozgur Kafali
Classifying Exceptions in Agent-Based Protocols
Commitments

- Obligation from debtor to creditor about specific content
- **Conditional Commitment**: represents an initial contract
  e.g., cc(bookstore, customer, pay(book), deliver(book))
- **Base-level Commitment**: represents an actual commitment
  e.g., c(bookstore, customer, deliver(book))
- **Commitment States**: active, fulfilled, violated
Ontologies

- A data model used for representing a domain
- Contains a set of concepts and relations between those concepts

Commitments

- BaseLevelCommitment
  - hasDebtor: Role
  - hasCreditor: Role
  - hasProposition: Predicate
  - hasStatus: \{active, fulfilled, violated\}

- ConditionalCommitment
  - hasCondition: Predicate
Protocols

- Business process composed of a set of sub-processes (or protocols)
- Three types of dependencies between sub-processes
  - Flow: One follows the other
e.g., bookstore delivers after customer pays
  - Fit: Two together produce an output
e.g., deliverer only delivers once two deliveries point to the same street
  - Sharing: Two consume the same input
e.g., two bookstores depend on the same supplier
- Delivery process consists of flow type dependencies
Exceptions

- **Violation**: An event directly conflicting with the goal e.g., the bookstore violating its commitment with the customer for delivering the book in time
Exceptions

- **Violation**: An event directly conflicting with the goal e.g., the bookstore violating its commitment with the customer for delivering the book in time

- **Bad-Fulfillment**: An extra event obstructing the goal e.g., a book being delivered with missing pages
Exceptions

- **Violation**: An event directly conflicting with the goal
e.g., the bookstore violating its commitment with the
customer for delivering the book in time

- **Bad-Fulfillment**: An extra event obstructing the goal
e.g., a book being delivered with missing pages

- **Bonus**: An extra event unrelated to the goal
e.g., a CD being delivered with the purchased book
Classification

- When the agent enters a new state
  - a commitment state is changed
  - a predicate is added
- Compare current state with goal state
  - violated commitments or predicates (violation)
  - extra predicates related to the goal (bad-fulfillment)
  - extra unrelated predicates (bonus)
Protocol, Commitments, Goals

```
cc(bookstore, customer, purchase(z), deliver(z))
cd(deliverer, bookstore, paydelivery(z), deliver(z))
```

\[ G_{customer} = \{purchase(z), deliver(z)\} \]
Exceptions

\[ G_{\text{customer}} = \{\text{purchase}(z), \text{deliver}(z)\} \]
Exceptions

- $\mathcal{G}_{\text{customer}} = \{\text{purchase}(z), \text{deliver}(z)\}$
- Violation: $\mathcal{S}_{\text{customer}} = \{\text{purchase}(\text{book1}), \neg \text{deliver}(\text{book1})\}$
Exceptions

- $G_{\text{customer}} = \{\text{purchase}(z), \text{deliver}(z)\}$
- Violation: $S_{\text{customer}} = \{\text{purchase}(\text{book1}), \neg\text{deliver}(\text{book1})\}$
Exceptions

- $\mathcal{G}_{\text{customer}} = \{\text{purchase}(z), \text{deliver}(z)\}$
- **Violation:** $\mathcal{S}_{\text{customer}} = \{\text{purchase}(\text{book1}), \neg\text{deliver}(\text{book1})\}$
- **Bad-Fulfillment:** $\mathcal{S}_{\text{customer}} = \{\text{purchase}(\text{cd1}), \text{deliver}(\text{cd1}), \text{damaged}(\text{cd1})\}$
Exceptions

- $G_{\text{customer}} = \{ \text{purchase}(z), \text{deliver}(z) \}$

- Violation: $S_{\text{customer}} = \{ \text{purchase}(\text{book1}), \neg \text{deliver}(\text{book1}) \}$

- Bad-Fulfillment: $S_{\text{customer}} = \{ \text{purchase}(\text{cd1}), \text{deliver}(\text{cd1}), \text{damaged}(\text{cd1}) \}$
Exceptions

- $G_{customer} = \{\text{purchase}(z), \text{deliver}(z)\}$
- Violation: $S_{customer} = \{\text{purchase}(book1), \neg\text{deliver}(book1)\}$
- Bad-Fulfillment: $S_{customer} = \{\text{purchase}(cd1), \text{deliver}(cd1), \text{damaged}(cd1)\}$
- Bonus: $S_{customer} = \{\text{purchase}(book1), \text{deliver}(book1), \text{deliver}(book2)\}$
Exceptions

- $\mathcal{G}_{\text{customer}} = \{\text{purchase}(z), \text{deliver}(z)\}$
- **Violation:** $\mathcal{S}_{\text{customer}} = \{\text{purchase}(\text{book1}), \neg\text{deliver}(\text{book1})\}$
- **Bad-Fulfillment:** $\mathcal{S}_{\text{customer}} = \{\text{purchase}(\text{cd}1), \text{deliver}(\text{cd}1), \text{damaged}(\text{cd}1)\}$
- **Bonus:** $\mathcal{S}_{\text{customer}} = \{\text{purchase}(\text{book1}), \text{deliver}(\text{book1}), \text{deliver}(\text{book2})\}$
Exceptions

- $\mathcal{G}_\text{customer} = \{ \text{purchase}(z), \text{deliver}(z) \}$
- **Violation:** $\mathcal{S}_\text{customer} = \{ \text{purchase}(\text{book1}), \neg \text{deliver}(\text{book1}) \}$
- **Bad-Fulfillment:** $\mathcal{S}_\text{customer} = \{ \text{purchase}(\text{cd1}), \text{deliver}(\text{cd1}), \text{damaged}(\text{cd1}) \}$
- **Bonus:** $\mathcal{S}_\text{customer} = \{ \text{purchase}(\text{book1}), \text{deliver}(\text{book1}), \text{deliver}(\text{book2}) \}$
- **How Bad?** $\mathcal{S}_\text{customer} = \{ \text{purchase}(\text{cd1}), \text{deliver}(\text{cd1}), \neg \text{invoice}(\text{cd1}) \}$

---

Özgür Kafalı
Classifying Exceptions in Agent-Based Protocols
Exceptions

- \( G_{customer} = \{purchase(z), \ deliver(z)\} \)
- **Violation:** \( S_{customer} = \{purchase(book1), \neg deliver(book1)\} \)
- **Bad-Fulfillment:** \( S_{customer} = \{purchase(cd1), \ deliver(cd1), \ damaged(cd1)\} \)
- **Bonus:** \( S_{customer} = \{purchase(book1), \ deliver(book1), \ deliver(book2)\} \)
- **How Bad?** \( S_{customer} = \{purchase(cd1), \ deliver(cd1), \neg invoice(cd1)\} \)
Exceptions

- $G_{customer} = \{purchase(z), deliver(z)\}$
- **Violation:** $S_{customer} = \{purchase(book1), \neg deliver(book1)\}$
- **Bad-Fulfillment:** $S_{customer} = \{purchase(cd1), deliver(cd1), damaged(cd1)\}$
- **Bonus:** $S_{customer} = \{purchase(book1), deliver(book1), deliver(book2)\}$
- **How Bad?** $S_{customer} = \{purchase(cd1), deliver(cd1), \neg invoice(cd1)\}$
- **Violation or Opportunity?** $S_{customer} = \{purchase(book1), deliver(book2)\}$
Exceptions

- $G_{\text{customer}} = \{\text{purchase}(z), \text{deliver}(z)\}$
- **Violation**: $S_{\text{customer}} = \{\text{purchase}(\text{book1}), \neg\text{deliver}(\text{book1})\}$
- **Bad-Fulfillment**: $S_{\text{customer}} = \{\text{purchase}(\text{cd1}), \text{deliver}(\text{cd1}), \text{damaged}(\text{cd1})\}$
- **Bonus**: $S_{\text{customer}} = \{\text{purchase}(\text{book1}), \text{deliver}(\text{book1}), \text{deliver}(\text{book2})\}$
- **How Bad?** $S_{\text{customer}} = \{\text{purchase}(\text{cd1}), \text{deliver}(\text{cd1}), \neg\text{invoice}(\text{cd1})\}$
- **Violation or Opportunity?** $S_{\text{customer}} = \{\text{purchase}(\text{book1}), \text{deliver}(\text{book1}), \text{deliver}(\text{book2})\}$
Future Directions

- Exception classification within specified categories
  - Level of violation (e.g., no delivery or late delivery)
  - How bad the goal is fulfilled
  - How significant the bonus is

- Anticipating Exceptions
- Recovering Exceptions