Eventuation properties and interaction contracts

Mario Südholt

Ascola research team; Mines Nantes, Inria, Lina

SCRIPT WS

Vrije Universiteit Brussel, 12 Nov. 2013
Motivation

Generalizing session types
- Session types
- Aspectual session types

Schemas for workflows
- Managing workflow adaptations
- Workflow adaptation schemas

Conclusion
1. Interaction contracts for the Cloud

- Interactions
  - Clients/servers, service compositions, ...
  - Existing support: languages/libraries, orchestration

- Our goal
  - Declarative and formal multi-level, cross-site protocols
  - Effective implementation support
  - Support legacy applications

OAuth 2.0 CSRF attacks

M. Südholt (Ascola team)
Motivation: intermittent inconsistent states in complex interacting systems
- Mobile, ambient devices with limited connectivity
- Intermittent property violation in service compositions

Eventuation properties
- Enforce properties after inconsistent situation
- Identify inconsistency?
- Pass info across inconsistent phase?

Common examples
- Eventual consistency
- Accountability in service compositions
- Error handling
Eventual consistency

- Handling data(base) replication in large distributed systems

Applications
- Managing intermittent connectivity
- Code versioning systems
  - Independency of ordering of change history
  - Git and subversion are not eventually consistent
  - Darcs is

Hot topic in language design
- Ex.: recent notions of revision histories, Cloud types
Accountable service compositions

- After-the-fact verification of security, privacy, economic properties, etc.
- Frequently requires anticipated information gathering
- Ex.: missing id information
  1. initial service injects data with id (signature, etc.)
  2. intermediate service strips id for privacy reasons
  3. final service requires id for audit
- Frequently defined using declarative obligation specs.
  - Eventuation properties over choreographies as operational intermediate form
Error handling

- Frequently errors occur silently
  - Inconsistent phase from occurrence to observable effects
- EP: enforce well-defined state after error occurrence
  - Enable or improve handling by shortening inconsistency
- Interest (to us): errors and security/accountability issues
What’s next?

**Overall project**
- Define EPs declaratively
- Provide effective implementation support

**First steps**
- Def.: generalization of session types
- Impl.: multi-level, cross-site accountability properties
2. Session types

Multiparty protocols

Expressive interaction structures

Fig. 3. Examples of Global Types
1. \( G_1 = \text{def } x_0 = \text{Alice }! \text{Bob : Msghnati;x1} \)
   \( x_1 = \text{end in} x_0 \)
2. \( G_2 = \text{def } x_0 = x_1 + x_2 \)
   \( x_1 \ldots \text{exclusive messages: as a consequence, Bob is in a deadlock, waiting for both Book and Film to arrive from Alice.} \)
Characteristics

- Recent advances in expressivity
  - Binary sessions (1990s)
  - Multiparty sessions [Honda, Yoshida, Carbone; POPL’08]
  - Roles [Daniélou, Yoshida; POPL’11]
  - Generalized merge/fork structures [Daniélou, Yoshida; ESOP’12]

- Properties
  - Strongly typed
  - Projection: automatic "transformation" to correct implementation
    - Global types: specification
    - Local types: implementation
  - Absence of deadlocks
Aspectual session types

- Limitations of existing session types
  - Strong restrictions on race conditions
    - Interesting protocols cannot be expressed
  - No support for modular definition
    - New functionality: extensive rewrites
- Both hinder enrichment of existing types
- Aspectual session types
  - Extend session types modularly
  - Allow uniform behavior in parallel threads
Ex.: simple trade session

- 3 participants: seller (S), broker (B), client (C)
- Broker indicates sale to S and purchase actions to C
Ex.: add negotiation (modular extension)

- **Negotiation**
  - Offers from the broker to the client
  - Counteroffers by the client

- **Modular extension**
  - Choice operator $+$

![Diagram showing session types and interactions]

\[
S \to B: \text{Item} \\
\text{proceed} \\
+C \to B: \text{Counter} \\
+B \to C: \text{Offer}
\]
Ex.: add authentication (race conditions)

- **Authentication**
  - Add authentication server A
  - Verify credentials before a purchase

- **Modular extension**
  - Disjunction of triggers
    - \( (B \rightarrow S:* + B \rightarrow C:* ) \)

- **Problem:** inserts race condition in branches of \( \mid \) of original session
Session types technically (ESOP’12)

- **Main conditions**
  - Linearity: parallel activities are triggered by different messages
  - Local choice: any choice can be resolved by one local process
  - Active senders: no different senders from same state

- **Multiparty session automata**
  - Subclass of communicating automata
Our technical contributions

- Aspectual linearity
  - Admit same thread-neutral functionality in parallel threads
  - Relax linearity condition
- Extension of multisession automata to aspectual sessions

Weaving and projections commute
3. Managing workflow adaptations

- EPs over complex workflows
- Need for multi-level and cross-site contracts
Ex.: OAuth 2.0

- Framework for resource access authorization
  - Used by Facebook, Google, Microsoft, SAP, etc.
- Provider yields access tokens to third-party clients on behalf of user
Secure OAuth

- OAuth 2.0 CSRF exploit: attacker abuses existing authentication
  - Remedy: add session-specific state
- State management needs multi-level contracts
  - Saving: implementation level
  - State generation, test: interceptor level
  - State transfer: service level
Workflow adaptation schemas

**schema** OAuthStateIntroduction

**instantiate schema** UpServiceRequests

< pat↓ TC_c@GenState → DispUA( argv) s

pat↓ AcceptGrantTC(ac,st, argv) s → TC_c@Grant?,

act CarryState >

- Patterns for complex-interactions
  - Multi-level (indices)
  - Cross-site (agent, →)

- Generic and instantiated schemas
  - Small DSL (ex.: UpServiceRequests)

- Implementation on top of Apache CXF
Expressive and executable typed formalisms for explicit protocols

Many Cloud/Web applications need multi-level, cross-site protocols
  - Structured expressive protocol transformations?
  - Suitable protocol formalism?

Eventuation properties for accountability as our major target
  - Remedy lack of information
  - Track errors with
References


- SAdapt: Apache CXF-based implementation of workflow adaptation patterns
  http://a4cloud.gforge.inria.fr/doku.php?id=start:advservcomp