Session Types: From Theory to Practice

Mark Florisson
mark.florisson@cl.cam.ac.uk

April 24, 2015
Motivation

Divide between computation and communication

- **Computation** (formal)
  - abstractions: functions, methods, classes, ...
  - formal specification: types, interfaces, contracts, ...

Mark Florisson | Session Types: From Theory to Practice
Motivation

Divide between computation and communication

- **Computation** (formal)
  - abstractions: functions, methods, classes, ...
  - formal specification: types, interfaces, contracts, ...

- **Communication** (ad-hoc)
  - abstractions: bytes? JSON?
  - informal specification: RFCs (HTTP, POP3, ...)
For communication there are ...

1. No good abstractions, and ...
2. No good specification and reasoning principles
Implementations **mix concerns**:  
1. protocol  
   ▶ network state  
2. (de)serialization  
   ▶ input validation  
3. transporting data (send/recv)  
   ▶ deal with network errors
Implementations **mix concerns**:

1. **protocol**
   - network state
2. **(de)serialization**
   - input validation
3. **transporting data (send/recv)**
   - deal with network errors

... and as a result are buggy, fragile or insecure
Implementations **mix concerns**:

1. protocol
   - network state
2. (de)serialization
   - input validation
3. transporting data (send/recv)
   - deal with network errors

... and as a result are buggy, fragile or insecure
... or don’t follow the informal specification
Example Protocol

POP3

Diagram:
- auth
- end
- cmd
- in1
- in2
- login
- passwd
- emailID
- quit
- err
- retr
- int

Mark Florisson | Session Types: From Theory to Practice
A new language to simplify matters
A language for
- existing protocols (e.g. HTTP, ...)
- ad-hoc protocols, web services, ...

Based on sound theory
- strong safety guarantees

Compatible with existing languages
Language Foundation: Session Types

- Well-studied typing discipline
  - process calculi ($\pi$-calculus)
- Describe allowable interactions
**POP3 Authentication in Session Types**

\[
Auth = \langle \text{quit: } !\text{String. end}, \text{login: } ?\text{Username. } ?\text{Password.} \\
\oplus\langle \text{error: } !\text{ErrAuth. Auth} \\
\text{, ok: } !\text{String. Email} \rangle \rangle
\]
POP3 Example Server

POP3 Authentication in Session Types

\[ Auth = &⟨ \text{quit: } !\text{String}\text{.end} \]
\[ \quad , \text{login: } ?\text{Username. } ?\text{Password.} \]
\[ \quad ⊕⟨ \text{error: } !\text{ErrAuth.} \text{Auth} \]
\[ \quad \quad , \text{ok: } !\text{String.} \text{Email} \⟩\⟩ \]

POP3 Email Retrieval in Session Types

\[ Email = &⟨ \text{retr: } ?\text{Int.} !\text{Message.} \text{Email} \]
\[ \quad , \ldots \]
\[ \quad , \text{quit: end} \⟩\]
Session types in practice:

- Near perfect fit for communication protocols
Session types in practice:

- Near perfect fit for communication protocols
- ... but need to extend theory to match problems in practice
Composability

- modularize protocols into smaller ones
Composability
- modularize protocols into smaller ones

Versioning and subtyping
- backwards compatibility
Extending Theory: Session Types

- Composability
  - modularize protocols into smaller ones
- Versioning and subtyping
  - backwards compatibility
- Multiple parties and hiding
  - e.g. server’s interaction with database is hidden from client
  - no "global" view
Composability
  - modularize protocols into smaller ones

Versioning and subtyping
  - backwards compatibility

Multiple parties and hiding
  - e.g. server’s interaction with database is hidden from client
  - no "global" view

Flexibility
  - optional fields, ....
Composability
- modularize protocols into smaller ones

Versioning and subtyping
- backwards compatibility

Multiple parties and hiding
- e.g. server’s interaction with database is hidden from client
- no "global" view

Flexibility
- optional fields, ....

Concurrent conversations (multiplexing/interleaving)
Generate state machine from protocol description
  - C, Python, Java, Haskell, OCaml, ...

Enforce state machine statically (framework) or dynamically (library)
Strong safety guarantees (automatically enforced)
  - correct interaction pattern (session fidelity)
  - correct message types (safety)
  - no deadlocks (progress)
Separation of concerns (simplicity, efficiency)
- protocol
- serialization
- transport medium
Assemble protocol implementation, e.g.:

- binary serialization over shared memory
- gzipped JSON over HTTP
 Outcome
 Separation of Concerns

- uniform error handling
- optimizations
- isolated testing
Other Applications

- Multi-threading or multi-processing
- Common foreign function interface (FFI) between languages
Conclusion

- Network protocols are complicated
- Session types can help
  - model for communication
  - foundation for a new language
- Marry theory and practice
  - apply theory to practice
  - extend theory to match practice
Thank you for listening! :)

---

Note: The diagram is not included in the plain text representation.
Errors are part of the protocol
Outcome
Separation of Concerns

➤ promise pipelining
➤ binary encoding
➤ ...

Mark Florisson | Session Types: From Theory to Practice
Isolated testing of
- protocol
- serialization
- transport implementations