Introduction to protocols

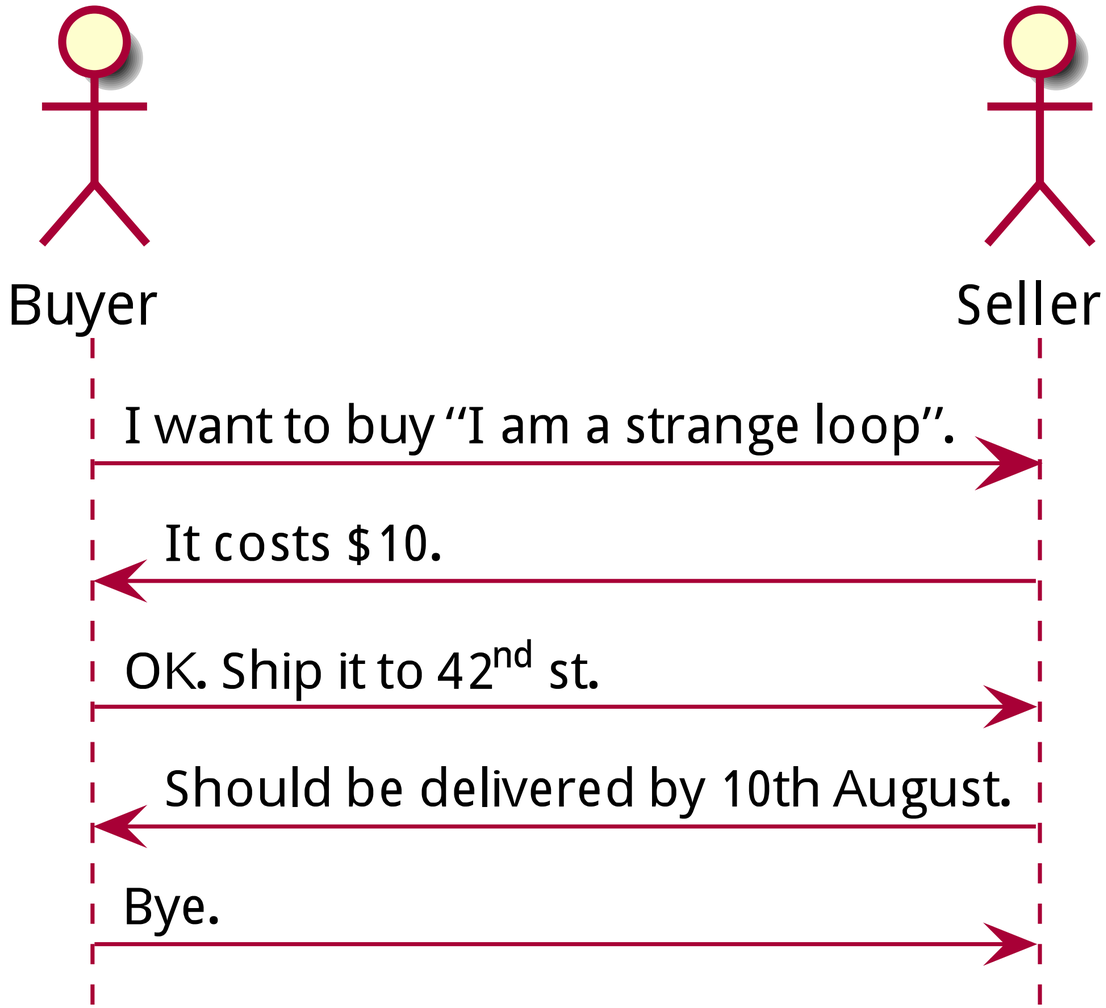
Programming Reactive Systems

Roland Kuhn

What is a protocol?

- protocols are ubiquitous in our society
- successful communication needs the recipient to be ready
- who should say what, and when?

Protocol example: buying a book



Formal description with session types

$$
\begin{aligned}
G = & 1 \longrightarrow 2 : \langle \mathit{string} \rangle . \\
& 2 \longrightarrow 1 : \langle \mathit{int} \rangle . \\
& 1 \longrightarrow 2 : \left\{ \begin{array}{ll} \langle \mathit{string} \rangle : & 2 \longrightarrow 1 : \langle \mathit{date} \rangle . \mathit{end} \\ \mathrm{quit} : & \mathit{end} \end{array} \right\}
\end{aligned}
$$

Formal description with session types

$$
\begin{aligned}
G = & 1 \longrightarrow 2 : \langle \mathit{string} \rangle . \\
& 2 \longrightarrow 1 : \langle \mathit{int} \rangle . \\
& 1 \longrightarrow 2 : \left\{ \begin{array}{ll} \langle \mathit{string} \rangle : & 2 \longrightarrow 1 : \langle \mathit{date} \rangle . \mathit{end} \\ \mathrm{quit} : & \mathit{end} \end{array} \right\}
\end{aligned}
$$

$$
P_{\mathsf{buyer}} = !\mathit{string}; ?\mathit{int}; \oplus\{\textbf{ok: } !\mathit{string}; ?\mathit{date}; \mathit{end}, \textbf{quit: } !\mathrm{quit}; \mathit{end}\}
$$

Formal description with session types

$$
\begin{aligned}
G = & 1 \longrightarrow 2 : \langle \mathit{string} \rangle . \\
& 2 \longrightarrow 1 : \langle \mathit{int} \rangle . \\
& 1 \longrightarrow 2 : \left\{ \begin{array}{ll} \langle \mathit{string} \rangle : & 2 \longrightarrow 1 : \langle \mathit{date} \rangle . \mathit{end} \\ \mathrm{quit} : & \mathit{end} \end{array} \right\}
\end{aligned}
$$

$$P_{\mathsf{buyer}} = !\mathit{string}; ?\mathit{int}; \oplus\{\textbf{ok: } !\mathit{string}; ?\mathit{date}; \mathit{end}, \textbf{quit: } !\mathrm{quit}; \mathit{end}\}$$

$$P_{\mathsf{seller}} = ?\mathit{string}; !\mathit{int}; \&\{\textbf{ok: } ?\mathit{string}; !\mathit{date}; \mathit{end}, \textbf{quit: } ?\mathrm{quit} : \mathit{end}\}$$

Formal description with session types

$$
\begin{aligned}
G = & 1 \longrightarrow 2 : \langle \mathit{string} \rangle . \\
& 2 \longrightarrow 1 : \langle \mathit{int} \rangle . \\
& 1 \longrightarrow 2 : \left\{ \begin{array}{ll} \langle \mathit{string} \rangle : & 2 \longrightarrow 1 : \langle \mathit{date} \rangle . \mathit{end} \\ \mathrm{quit} : & \mathit{end} \end{array} \right\}
\end{aligned}
$$

$$P_{\text{buyer}} = !\mathit{string}; ?\mathit{int}; \oplus\{\textbf{ok: } !\mathit{string}; ?\mathit{date}; \mathit{end}, \textbf{quit: } !\mathrm{quit}; \mathit{end}\}$$

$$P_{\text{seller}} = ?\mathit{string}; !\mathit{int}; \&\{\textbf{ok: } ?\mathit{string}; !\mathit{date}; \mathit{end}, \textbf{quit: } ?\mathrm{quit} : \mathit{end}\}$$

see also *A Gentle Introduction to Multiparty Asynchronous Session Types* by M. Coppo, M. Dezani-Ciancaglini, L. Padovani, N. Yoshida

The type of a channel

- the protocol governs not only when but also what is sent
- the message type is specified for each step

Two options:

The type of a channel

- the protocol governs not only when but also what is sent
- the message type is specified for each step

Two options:

- each participant has one channel per interluctor

The type of a channel

- the protocol governs not only when but also what is sent
- the message type is specified for each step

Two options:

- each participant has one channel per interluctor
- one channel for each message type

The trouble with linear logic

```
val channel = ...
channel.send(msg)
channel.send(msg) // again!
```

Mainstream programming languages only allow the expression of addition of facts, not their removal; examples of exceptions are Clean (1987), Idris (2017), Linear Haskell (2017).

Affine types (references used at most once) start appearing in C++11 and Rust.

Summary

In this video we have seen:

- what constitutes a protocol
- how protocols can be formally specified
- the effect of session types on local channel typing rules