Capable GV
Capabilities for Session Types in GV

Magdalena J. Latifa and Ornela Dardha

TYPES 2022
Session types

- Communication formalism

- Guarantees: communication safety, session fidelity and privacy

- Rely on *linearity*
• Functional calculus with session types
• Correspondence to classical linear logic
• Programs are well behaved
• Practical suitability
• Many extensions have been developed
PQB

• Not currently possible in any of the current GVs

• Can be implemented via simply passing the channel endpoint but then neither P nor Q have the guarantee they are communicating directly with B

• For example, P could pretend to send over channel a and instead send over a new channel and snoop on Q’s details
Channel sharing

- The solution to the PQB problem
- Channel endpoints can be accessed by multiple processes
- Need to enforce linearity for communication safety...
Capabilities

- **Split** channels into *channel endpoints* and their capabilities
- Allows the *endpoints to be unrestricted by keeping capabilities linear*
- Technique originated in region-based memory management
Capable GV

- Incorporates capabilities into the linear setting of GV

- Based predominantly on constructs from PGV rather than other GV extension to take advantage of cyclic structure

- Static elements: terms, types, typing rules

- The rest in progress...
CGV types

\[ T, U : ::= T \times U \mid T + U \mid tr (\rho) \mid [\rho \ (S)] \]
\[ \mid 1 \mid T \ (C) \Rightarrow \ (C') \ U \]

\[ \Gamma ::= \emptyset \mid \Gamma, \ x : T \]
\[ \Delta ::= \emptyset \mid \Delta, \ \rho \]
\[ C ::= \emptyset \mid C \otimes \rho \ (S) \]
CGV types

\[ T, U : ::= T \times U \mid T + U \mid \text{tr} (\rho) \mid [\rho (S)] \mid 1 \mid T (C) \sim (C') U \]

\[ \Gamma : ::= \emptyset \mid \Gamma, x : T \]

\[ \Delta : ::= \emptyset \mid \Delta, \rho \]

\[ C : ::= \emptyset \mid C \otimes \rho (S) \]
CGV types

\[ T, U : ::= T \times U | T + U | tr(\rho) | [\rho(S)] | 1 | T(C) \sim (C')U \]

\[ \Gamma : ::= \emptyset | \Gamma, x : T \]

\[ \Delta : ::= \emptyset | \Delta, \rho \]

\[ C : ::= \emptyset | C \otimes \rho(S) \]
Typing judgements

• $\Gamma ; \Delta ; \vdash V : T$
  under typing environment $\Gamma$ and capability environment $\Delta$, term $V$ is of type $T$

• $\Gamma ; \Delta ; C \vdash M : T \triangleright C'$
  under typing environment $\Gamma$ and capability environment $\Delta$ and with capability set $C$, term $M$ is of type $T$ and produces capability set $C'$
Typing judgements

• \( \Gamma; \Delta; \vdash V : T \)
  under typing environment \( \Gamma \) and capability environment \( \Delta \),
  term \( V \) is of type \( T \)

• \( \Gamma; \Delta; C \vdash M : T \triangleright C' \)
  under typing environment \( \Gamma \) and capability environment \( \Delta \) and with
  capability set \( C \), term \( M \) is of type \( T \) and produces capability set \( C' \)
CGV terms

$$V, W ::= () \mid x \mid \lambda x . M \mid (V, W) \mid \text{inl } V \mid \text{inr } V$$

$$L, M, N ::= VW \mid \text{return } V \mid \text{let } x = M \text{ in } N$$
$$\mid \text{let } (x, y) = V \text{ in } M \mid \text{let } () = V \text{ in } M$$
$$\mid \text{case } L \{ \text{inl } x \mapsto \to M; \text{inr } y \mapsto \to N \}$$
$$\mid \text{new} \mid \text{send } V \mid \text{recv } V \mid \text{close } V$$
$$\mid \text{inact } V \mid \text{act } V \mid \text{spawn } M$$
CGV terms

\[ V, W ::= () \mid x \mid \lambda x . M \mid (V, W) \mid \text{inl } V \mid \text{inr } V \]

\[ L, M, N ::= VW \mid \text{return } V \mid \text{let } x = M \text{ in } N \]
\[ \mid \text{let } (x, y) = V \text{ in } M \mid \text{let } () = V \text{ in } M \]
\[ \mid \text{case } L \{ \text{inl } x \mapsto \rightarrow M; \text{inr } y \mapsto \rightarrow N \} \]
\[ \mid \text{new} \mid \text{send } V \mid \text{recv } V \mid \text{close } V \]
\[ \mid \text{inact } V \mid \text{act } V \mid \text{spawn } M \]
\begin{align*}
\text{\texttt{T-Recv}} \\
\Gamma; \Delta \vdash V : tr(\rho) \\
\hline
\Gamma; \Delta; C \otimes \rho(\texttt{?T.S}) \vdash \texttt{recv} \ V : T \rightarrow C \otimes \rho(S)
\end{align*}

\begin{align*}
\text{\texttt{T-Send}} \\
\Gamma; \Delta \vdash V : T \times tr(\rho) \\
\hline
\Gamma; \Delta; C \otimes \rho(\texttt{!T.S}) \vdash \texttt{send} \ V : 1 \rightarrow C \otimes \rho(S)
\end{align*}
\[ \text{T-Recv} \]

\[
\Gamma; \Delta \vdash V : tr(\rho)
\]

\[
\Gamma; \Delta; C \otimes \rho(\text{?T.S}) \vdash \text{recv} \ V : T \rightarrow C \otimes \rho(S)
\]

\[ \text{T-Send} \]

\[
\Gamma; \Delta \vdash V : T \times tr(\rho)
\]

\[
\Gamma; \Delta; C \otimes \rho(!T.S) \vdash \text{send} \ V : 1 \rightarrow C \otimes \rho(S)
\]
\[ \text{T-Recv} \]
\[
\frac{\Gamma; \Delta \vdash V : tr(\rho)}{\Gamma; \Delta; C \otimes \rho(\mathit{T.S}) \vdash \text{recv } V : T \triangleright C \otimes \rho(S)}
\]

\[ \text{T-Send} \]
\[
\frac{\Gamma; \Delta \vdash V : T \times tr(\rho)}{\Gamma; \Delta; C \otimes \rho(!\mathit{T.S}) \vdash \text{send } V : 1 \triangleright C \otimes \rho(S)}
\]
CGV terms

\[ V, W ::= (\) | x | \lambda x . M | (V, W) | \text{inl } V | \text{inr } V \]

\[ L, M, N ::= V W | \text{return } V | \text{let } x = M \text{ in } N \]
\[ | \text{let } (x, y) = V \text{ in } M | \text{let } () = V \text{ in } M \]
\[ | \text{case } L \{\text{inl } x \mapsto \rightarrow M; \text{inr } y \mapsto \rightarrow N\} \]
\[ | \text{new} | \text{send } V | \text{recv } V | \text{close } V \]
\[ | \text{inact } V | \text{act } V | \text{spawn } M \]
\[
\text{T-Inact} \quad \Gamma; \Delta \vdash V : tr(\rho) \\
\hline
\Gamma; \Delta; C \otimes \rho(S) \vdash \text{inact } V : [\rho(S)] \triangleright C
\]

\[
\text{T-Act} \quad \Gamma; \Delta \vdash V : [\rho(S)] \\
\hline
\Gamma; \Delta; C \vdash \text{act } V : 1 \triangleright C \otimes \rho(S)
\]
\[ T\text{-Inact} \]
\[
\Gamma; \Delta \vdash V : \text{tr}(\rho) \\
\Gamma; \Delta; C \otimes \rho(S) \vdash \text{inact } V : [\rho(S)] \triangleright C
\]

\[ T\text{-Act} \]
\[
\Gamma; \Delta \vdash V : [\rho(S)] \\
\Gamma; \Delta; C \vdash \text{act } V : 1 \triangleright C \otimes \rho(S)
\]
T-Inact

\[ \Gamma; \Delta \vdash V : tr(\rho) \]

\[ \Gamma; \Delta; C \otimes \rho(S) \vdash \text{inact } V : [\rho(S)] \triangleright C \]

T-Act

\[ \Gamma; \Delta \vdash V : [\rho(S)] \]

\[ \Gamma; \Delta; C \vdash \text{act } V : 1 \triangleright C \otimes \rho(S) \]
CGV terms

\[ V, W ::= () \mid x \mid \lambda x \cdot M \mid (V, W) \mid \text{inl } V \mid \text{inr } V \]

\[ L, M, N ::= V W \mid \text{return } V \mid \text{let } x = M \text{ in } N \]
\[ \quad \mid \text{let } (x, y) = V \text{ in } M \mid \text{let } () = V \text{ in } M \]
\[ \quad \mid \text{case } L \{ \text{inl } x \mapsto \rightarrow M; \text{inr } y \mapsto \rightarrow N \} \]
\[ \quad \mid \text{new} \mid \text{send } V \mid \text{recv } V \mid \text{close } V \]
\[ \quad \mid \text{inact } V \mid \text{act } V \mid \text{spawn } M \]
T-LetBind
\[ \Gamma_1; \Delta_1; C \vdash M : T \triangleright C' \quad \Gamma_2, x : T; \Delta_2; C' \vdash N : U \triangleright C'' \]
\[ \Gamma_1 \circ \Gamma_2; \Delta_1, \Delta_2; C \vdash \text{let } x = M \text{ in } N : U \triangleright C'' \]

T-Spawn
\[ \Gamma; \Delta; C_S \vdash M : 1 \triangleright \emptyset \]
\[ \Gamma; \Delta; C \otimes C_S \vdash \text{spawn } M : 1 \triangleright C \]
**T-LetBind**

\[
\Gamma_1; \Delta_1; C \vdash M : T \triangleright C' \quad \Gamma_2, x : T; \Delta_2; C' \vdash N : U \triangleright C''
\]

\[
\Gamma_1 \circ \Gamma_2; \Delta_1, \Delta_2; C \vdash \text{let } x = M \text{ in } N : U \triangleright C''
\]

**T-Spawn**

\[
\Gamma; \Delta; C_s \vdash M : 1 \triangleright \emptyset
\]

\[
\Gamma; \Delta; C \otimes C_s \vdash \text{spawn } M : 1 \triangleright C
\]
T-LetBind
\[ \Gamma_1; \Delta_1; C \vdash M : T \Rightarrow C' \quad \Gamma_2, x : T; \Delta_2; C' \vdash N : U \Rightarrow C'' \]
\[ \Gamma_1 \circ \Gamma_2; \Delta_1, \Delta_2; C \vdash \text{let } x = M \text{ in } N : U \Rightarrow C'' \]

T-Spawn
\[ \Gamma; \Delta; C_S \vdash M : 1 \Rightarrow \emptyset \]
\[ \Gamma; \Delta; C \otimes C_S \vdash \text{spawn } M : 1 \Rightarrow C \]
Conclusion

• CGV allows channel sharing - PQB is typeable
• Linearity of communication is enforced via type-end-effect system
• Operational semantics in the works
• Expecting to preserve subject reduction but introduce deadlocks
• Can restore deadlock-freedom e.g. via priorities

Thank you!