Session Types

Outline

Multiparty Session Types

Syntax

$$T ::= S \quad \text{(session type)} \mid D \quad \text{(data type)}$$

$$\begin{array}{rcll}
S & ::= & \mathtt{end} & \text{(termination)} \\
 & \mid & L.S & \text{(sequence)} \\
 & \mid & \mu\alpha.S & \text{(recursion)} \\
 & \mid & \alpha & \text{(type variable)} \\
 & \mid & (S+S) & \text{(nondeterminism)}
\end{array}$$

$$\begin{array}{rcll}
L & ::= & L|L & \text{(asynchrony)} \\
 & \mid & p \to q(D) & \text{(send-receive)} \\
 & \mid & \mathtt{skip} & \text{(skip)} \\
p, q & \in & \mathbb{Z} \cup \{\infty\}, p \neq q, pq \geq 0 & \text{(index)} \\
D & ::= & \ell|\mathtt{bool}|\mathtt{int}|\mathtt{string}|\ldots & \\
\ell & \in & \mathcal{L} & \text{(label)}
\end{array}$$

Operational Semantics

$$L.S \xrightarrow{\ \mathsf{L}\ } S \qquad \text{(sequence)}$$

$$\mu\alpha.S \xrightarrow{\texttt{skip}} \alpha \xrightarrow{\texttt{skip}} S \qquad \text{(recursion)}$$

$$\begin{tikzcd} & S_1 \\ S_1 + S_2 \arrow[ur, "\texttt{skip}"] \arrow[dr, "\texttt{skip}"'] & \\ & S_2 \end{tikzcd} \qquad \text{(nondeterminism)}$$

Note:

- A type S generates a directed graph (NFA) $\mathrm{gr}(S)$ called a semantic graph, with an initial node and the terminal node `end`. We can rename the nodes for convenience.
- From the graph, we can show commutativity and distributivity of nondeterminism. Hence we can just write $S_1 + S_2 + S_3$ instead of $(S_1 + S_2) + S_3$.
- Conversely, for each directed graph whose edges are defined by the production rules for L (can have multiple terminal nodes), we have a type.

Subtype

We want to encode in our type system

- Determinism is allowed in a nondeterministic context

$$S_1 \preceq S_1 + S_2.$$

- Waiting for more actions to be asynchronously executed is allowed

$$L_1 | L_2 \preceq L_1.$$

The latter can be expressed as a relation between two *edges*.

Trace and Path

A path is a sequence $S_1 \xrightarrow{L_1} S_2 \xrightarrow{L_2} \cdots \xrightarrow{L_n} \texttt{end}$. Given this path, we have a trace $t = L_1.L_2 \ldots L_n$. The length of t is $|t| = n$.

A trace $t = L_1.L_2 \ldots L_n$ is equivalent to the path from t removing a $\texttt{skip}$

$$L_1.L_2 \ldots L_n \equiv L_1 \ldots L_{j-1}.L_j \ldots L_n \text{ if } L_j = \texttt{skip}.$$

Let $t = L_1.L_2 \ldots L_n$ and $t' = L'_1.L'_2 \ldots L'_n$. Define $t \preceq t'$ if $L_i \preceq L'_i$ for all $i \in \{1, \ldots, n\}$.

Let t_1 and t_2 be two traces. Then $t_1 \preceq t_2$ if there exist t'_1 and t'_2 of the same length such that $t_1 \equiv t'_1$, $t_2 \equiv t'_2$ and $t'_1 \preceq t'_2$.

Subtype

Hence we can define subtype relation based on generated graphs.

Let $\mathrm{tr}(S)$ the set of traces of the graph generated by S.

We define $S_1 \preceq S_2$ if for any $t_1 \in \mathrm{tr}(S_1)$, there exists $t_2 \in \mathrm{tr}(S_2)$ such that $t_1 \preceq t_2$.

But the best thing we should do is to derive equational reasoning on types.

- For each type S, there exists a corresponding *regular expression*.
- There have been proof theories on regular expressions (Kleen algebra) and right-linear grammar [1]

[1] Das, Anupam, and Abhishek De. "A proof theory of right-linear (ω-) grammars via cyclic proofs." Proceedings of the 39th Annual ACM/IEEE Symposium on Logic in Computer Science. 2024.

Projection

$$p \to q \downharpoonright_p \quad = \begin{cases} 0 \to \infty, & \text{if } q = 0 \\ 0 \to -q, & \text{if } q < 0 \\ \varnothing, & \text{otherwise} \end{cases} \qquad q \to p \downharpoonright_p \quad = \begin{cases} \infty \to 0, & \text{if } q = 0 \\ -q \to 0, & \text{if } q < 0 \\ \varnothing, & \text{otherwise} \end{cases}$$

$$\begin{aligned} L_1 | L_2 \downharpoonright_p &= L_1 \downharpoonright_p | L_2 \downharpoonright_p \\ \mathtt{skip} \downharpoonright_p &= \mathtt{skip} \\ p \to q(D) \downharpoonright_p &= \begin{cases} p \to q \downharpoonright_p (D), & \text{if } p \to q \downharpoonright_p \neq \varnothing \\ \mathtt{skip}, & \text{otherwise} \end{cases} \end{aligned}$$

From $\mathrm{gr}(S)$, we replace each edge by its projection. The type derived from this graph is the projected type $S \downharpoonright_p$.

Proposition: If $S_1 \preceq S_2$, then $S_1 \downharpoonright_p \preceq S_2 \downharpoonright_p, \forall p \in \mathbb{N}^-$.

Typed Open Automata

Typed Open Automata

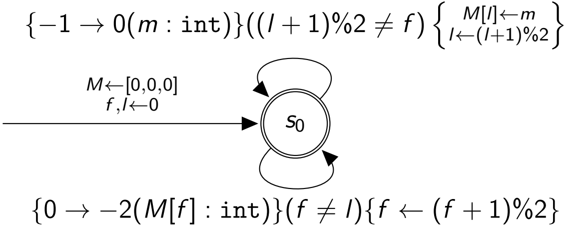
A typed open automaton is a tuple $A = \langle S, s_0, E, V, \phi_0, T \rangle$, where

- S is the set of states
- $s_0 \in S$ is the initial state
- $E \subset S$ is the set of terminal states
- V is the set of variables
- $\psi_0 : V \to \mathcal{P}$ is the initial assignment
- T is the set of transitions. Each $t \in T$ has the form $\dfrac{\beta_j^{j \in J}, g, \psi}{s \xrightarrow{\alpha} s'}$, where
    - $s, s' \in S$ and α is an emitted action.
    - each β_j has the form $p \to q(m : D)$ or $p \to q(\ell)$ such that $p, q \in \mathbb{Z} \cup \infty$, $p \neq q$, $pq \geq 0$ and $\ell \in \mathcal{L}$.
    - g is a predicate over V
    - $\psi : V \to \mathcal{E}_V$ is a reassignment

We can ignore the emitted action and write $s \xrightarrow{\beta_j^{j \in J}, g, \psi} s'$. A pair (s, ϕ), where $s \in S$ and $\phi : V \to \mathcal{P}$ is called a configuration of the automaton.

Example

Consider a producer-consumer communication through a size-2 circular buffer. This is modeled as an automaton A.



Type Generated by an OA

Consider $A = \langle S, s_0, E, V, \phi_0, T \rangle$.

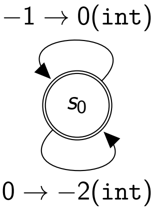
- $[\![p \rightarrow q(m : D)]\!] = p \rightarrow q(D)$
- $[\![p \rightarrow q(\ell)]\!] = p \rightarrow q(\ell)$
- $[\![\beta_1, \ldots, \beta_n]\!] = [\![\beta_1]\!] | \ldots | [\![\beta_n]\!]$

Weak type

The weak type W_A generated by A is derive from the graph, called the weak type graph, such that

- The set of nodes is S, the initial node is s_0, the set of terminal nodes is E
- Each transition $s \xrightarrow{\beta_j^{j \in J}, g, \psi} s'$ has a corresponding edge $s \xrightarrow{[\![\beta_j^{j \in J}]\!]} s'$

Example: The weak type graph for producer-consumer. Let $U = -1 \to 0(\mathtt{int})$ and $V = 0 \to -2(\mathtt{int})$. The type is $W_A = \mu\alpha.(\mathtt{end} + U + V).\alpha$



The DFA generated by the weak type of A

Strong type

The strong type graph G_S has nodes as *configurations*. The initial node is (s_0, ϕ_0). Terminal nodes are (s, ψ) where $s \in E$.

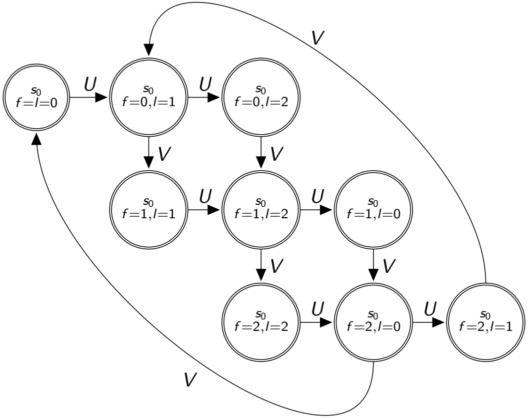
Each $(s, \phi) \xrightarrow{\beta_j^{j \in J}, g, \psi} (s', \phi')$ corresponds to an edge $(s, \phi) \xrightarrow{[\![\beta_j^{j \in J}]\!]} (s', \phi')$.

The graph G_S derives the strong type S_A.

We should be able to show that $S_A \preceq W_A$.

Example: The strong type graph for producer-consumer.

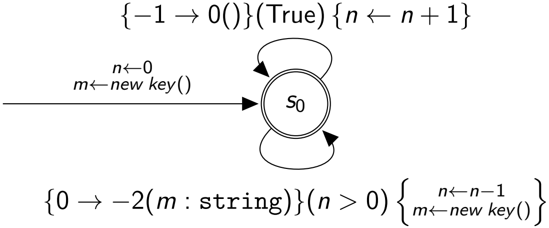
Strong type


s_0
$f=l=0$
s_0
$f=0,l=1$
s_0
$f=0,l=2$
s_0
$f=1,l=1$
s_0
$f=1,l=2$
s_0
$f=1,l=0$
s_0
$f=2,l=2$
s_0
$f=2,l=0$
s_0
$f=2,l=1$
U
V

Strong type

However, the strong type does not always exist i.e. strong type graph generation does not always halt.

Example: Consider a key-generating protocol, where the server request generating a secret key for the client to use later. The number of key consumptions cannot exceed the number of key generation requests. This is modeled as an automaton B.



Relaxed type

A relaxed type graph G_R has nodes of the form (s, P), where P is a predicate over V. *The* initial node is (s_0, P_0) such that $\phi_0 \vdash P_0$. Terminal nodes are (s, P) where $s \in E$.

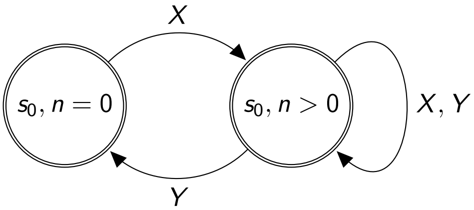
Each $(s, \phi) \xrightarrow{\beta_j^{j \in J}, g, \psi} (s', \phi')$ corresponds to an *edges* $(s, P) \xrightarrow{[\![\beta_j^{j \in J}]\!]} (s', P')$ such that $\phi \vdash P$ and $\phi' \vdash P'$

The graph G_R derives the strong type R_A.

We should be able to show that $S_A \preceq R_A \preceq W_A$.

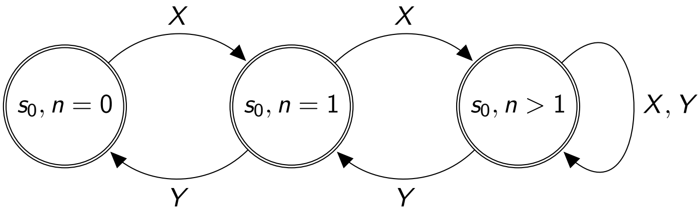
Example: Let $X = -1 \to 0(\texttt{void})$ and $Y = 0 \to -2(\texttt{string})$. We have relaxed type graph for key-generating protocol which derives

$$R_B = \mu\alpha_0.X.\mu\alpha_1.(X.\alpha_1 + Y.\alpha_1 + Y.\alpha_0).$$



Relaxed type

Example: Another relaxed type graph for key-generating protocol. It derives R'_B.



Note that $R'_B \prec R_B$ (strictly) but $R'_B \restriction_{-1} \equiv R_B \restriction_{-1}$.

Directions

Let $\mathcal{S}(A)$ be the set of types generated by A (strongly, weakly and relaxedly). We attempt to prove or disprove that

- $\mathcal{S}(A)$ is totally ordered. In particular, if S_1 and S_2 are generated by an automaton A, then $S_1 \preceq S_2$ if and only if the number of nodes $\mathrm{gr}(S_1)$ is greater than or equal to the number of nodes in $\mathrm{gr}(S_2)$.
- If there exist $S_1, S_2 \in \mathcal{S}(A)$ such that $S_1 \not\equiv S_2$ and $S_1 \downharpoonleft_p \equiv S_2 \downharpoonleft_p$. Then $S \downharpoonleft_p \equiv S_1 \downharpoonleft_p$, for every $S \in \mathcal{S}(A)$.

Note: If the latter is correct, we may be sure about the type of a child without knowing the strong type of the parent.

Composition

Consider an automaton

$$A = \langle\!\langle S_A, s_{0A}, E_A, V_A, \psi_{0A}, T_A \rangle\!\rangle \text{ and } B = \langle\!\langle S_B, s_{0B}, E_B, V_B, \psi_{0B}, T_B \rangle\!\rangle.$$

An automaton B can be safely composed to the child indexed by p of the automaton A if $\inf \mathcal{S}(B) \preceq \inf\{S \downharpoonright_p | S \in \mathcal{S}(A)\}$.

Reindex children of A and B if there is any conflict.

Composition

The composition of B into the internal component indexed by $p < 0$ of A yields an open automaton $A[B/p] := C = \langle\!\langle S_C, s_{0C}, E_C, V_C, \psi_{0C} \rangle\!\rangle$, such that

- $S_C = S_A \times S_B$
- $s_{0C} = (s_{0A}, s_{0B})$
- $E_C = E_A \times E_B$
- $V_C = V_A \uplus V_B$
- $\psi_C = \psi_A \uplus \psi_B$
- $T_C = \ldots$

$$T_C = \left\{ \frac{\beta_{j''}^{j'' \in J''}, g \wedge g', \psi \uplus \psi'}{(s,s') \xrightarrow{\alpha} (t,t')} \middle| \frac{\beta_j^{j \in J}, g, \psi}{s \xrightarrow{\alpha} t} \in T_A \wedge \frac{\beta_{j'}^{j' \in J'}, g', \psi'}{s' \xrightarrow{\alpha'} t'} \in T_B \wedge [\![\beta_{j'}^{j' \in J'}]\!] \preceq [\![\beta_j^{j \in J}]\!] \downharpoonright_p \right\}$$

$$\bigcup \left\{ \frac{\beta_j^{j \in J}, g, \psi}{(s,s') \xrightarrow{\alpha} (t,t')} \middle| s', t' \in S_B \wedge \frac{\beta_j^{j \in J}, g, \psi}{s \xrightarrow{\alpha} t} \in T_A \wedge \left(\nexists \frac{\beta_{j'}^{j' \in J'}, g', \psi'}{s' \xrightarrow{\alpha'} t'} \in T_B, [\![\beta_{j'}^{j' \in J'}]\!] \preceq [\![\beta_j^{j \in J}]\!] \downharpoonright_p \right) \right.$$

$$\bigcup \left\{ \ldots \middle| \frac{\beta_{j'}^{j' \in J'}, g', \psi'}{s' \xrightarrow{\alpha'} t'} \in T_B \wedge \left(\nexists \frac{\beta_j^{j \in J}, g, \psi}{s \xrightarrow{\alpha} t} \in T_A, [\![\beta_{j'}^{j' \in J'}]\!] \preceq [\![\beta_j^{j \in J}]\!] \downharpoonright_p \right) \right\}$$

We have to work more on the last set. Generally speaking, all other communications in B become internal communication in $A[B/p]$.

Plans

- That $\inf \mathcal{S}(A[B/p]) \prec \inf \mathcal{S}(A)$ is not straightforward

Thank you for listening !