The Network as a Language Construct

Tony Garnock-Jones       Sam Tobin-Hochstadt       Matthias Felleisen
Routing Events
Hierarchical Layering
Publish/Subscribe for Actors
Actor Programming Languages
   Erlang/OTP, Scala/Akka, ...

Network Calculus
Actor Calculus
PART I: The Problem
**Functional I/O**

Scaling up **big-bang** from domain-specific to general functional I/O

Apps in a functional I/O style:
- echo server
- multi-user chat
- DNS server
- SSH server

**Distributed Systems**

Implementing **RabbitMQ** and using it to build distributed systems

Investigated other paradigms:
- OO languages
- Network architecture
- CORBA services
- Erlang applications
- Modern Unix services
Ubiquitous Patterns and Problems

Event broadcasting  
Naming service  
Service discovery  
Startup ordering  
Crash/exit signalling  
Conversation management
Ubiquitous Patterns and Problems

Event broadcasting
Naming service

Uniform Linguistic Solution

Startup ordering
Crash/exit signalling
Conversation management
Recipe for Actor Languages

\( f(\alpha, u) = (\overline{a}, u') \)

\( \alpha = \text{event, } a = \text{action} \)
Log producers $\xrightarrow{\text{log messages}}$ Log consumers

$\langle \text{log, [subsystem, severity, data]} \rangle$

Consumers filter by subsystem, severity
Logging: Requirements Scorecard

- Route log entries from producers to consumers
- Consumers filter log messages
- Decouple producers from consumers
- Avoid shared-state explosion
- Discovery of logging service
- Only produce if someone's listening
- Alert when a producer crashes/exports
- Uniform treatment of I/O
PART II: Why Publish/Subscribe? How?
# Logging: Requirements Scorecard

<table>
<thead>
<tr>
<th>Requirement</th>
<th>Actor Status</th>
</tr>
</thead>
<tbody>
<tr>
<td>Route log entries from producers to consumers</td>
<td>✔️ &quot;Router&quot;</td>
</tr>
<tr>
<td>Consumers filter log messages</td>
<td>✔️ &quot;Router&quot;</td>
</tr>
<tr>
<td>Decouple producers from consumers</td>
<td>✔️ &quot;Router&quot;</td>
</tr>
<tr>
<td>Avoid shared-state explosion</td>
<td>✔️ &quot;Router&quot;</td>
</tr>
<tr>
<td>Discovery of logging service</td>
<td></td>
</tr>
<tr>
<td>Only produce if someone's listening</td>
<td></td>
</tr>
<tr>
<td>Alert when a producer crashes/exits</td>
<td></td>
</tr>
<tr>
<td>Uniform treatment of I/O</td>
<td></td>
</tr>
</tbody>
</table>
Route by address

Messages \( m = \langle x, v \rangle \)
\( x \in \text{Addresses} \)

Route by content

Messages \( m = \langle v \rangle \)
\( v = u \mid v, v \)
Patterns \( p = u \mid p, p \mid \ast \)
Interests \( \pi = \langle p \rangle \)
Route by address

\((C_1, \star)\)

Route by content

\((\text{log}, \star)\) or 
\((\text{log}, [\star, \text{error}, \star])\) or 
\((\text{log}, [P_1, \star, \star])\) or 
\(\ldots\)
\[ C \left\{ \begin{array}{c}
A \\
\Sigma
\end{array} \right\} \]

\[ \begin{array}{c}
(f, u) \\
\bar{a} \\
\bar{A}
\end{array} \]

\[ \begin{array}{c}
\bar{\pi} \\
\bar{\alpha}
\end{array} \]

- \( f = \) Base language functions
- \( u = \) Base language values
- \( B = (f, u) \) Behaviors
- \( \Sigma = \bar{a} \triangleleft B \) Actor States
- \( A = \bar{\pi} : \Sigma \) Actors
- \( C = [\bar{\alpha} ; \bar{A}] \) Configurations

- \( \alpha = \langle v \rangle \) Events
- \( a = \alpha \mid A \) Actions
- \( v = u \mid v, v \) Message values
- \( p = u \mid p, p \mid \ast \) Message patterns
- \( \pi = (p) \) Interests
\[ A = \tilde{\pi} : \bar{a} \triangleleft B \quad \text{Actors} \]
\[ A_Q = \tilde{\pi} : \cdot \triangleleft B \quad \text{Quiescent Actors} \]
\[ C = [\bar{\alpha} ; \bar{A}] \quad \text{Configurations} \]
\[ C_Q = [\cdot ; \bar{A}_Q] \quad \text{Quiescent Configurations} \]
Event broadcast

\[
\begin{align*}
A_Q & \xrightarrow{\alpha} A' \\
[\alpha\bar{a}_0; \overline{A_Q}] & \rightarrow [\bar{a}_0; \overline{A'}]
\end{align*}
\]

\[
C \rightarrow C'
\]

\[
f(\alpha | \overline{\pi}, u) = (\bar{a}, u')
\]

\[
\tilde{\pi} : \cdot \triangleleft (f, u) \xrightarrow{\alpha} \tilde{\pi} : \bar{a} \triangleleft (f, u')
\]

Actions interpreted  Event interpreted
Event Broadcast

\[
\begin{align*}
A_Q \xrightarrow{\alpha} A' \\
[\alpha \overline{\alpha}_0; \overline{A_Q}] & \rightarrow [\overline{\alpha}_0; \overline{A'}]
\end{align*}
\]
Event Filtering

\[ \alpha \mid_{\tilde{\pi}} : \alpha \times \tilde{\pi} \rightarrow \alpha \]

\[ v \mid_{p} : v \times p \]

\[ \langle v \rangle \mid_{\tilde{\pi}} = \langle v \rangle, \text{ if } \exists(p) \in \tilde{\pi} \text{ such that } v \mid_{p} \]

\[ \alpha \mid_{\tilde{\pi}} \text{ otherwise undefined} \]
Event Interpretation, $A \xrightarrow{\alpha} A$

\[
\begin{align*}
\tilde{\pi} : \cdot \triangleleft B & \xrightarrow{\alpha} \tilde{\pi} : \cdot \triangleleft B \quad (\alpha \upharpoonright \tilde{\pi} \text{ is undefined}) \\
\tilde{\pi} : \cdot \triangleleft (f, u) & \xrightarrow{\alpha} \tilde{\pi} : \bar{a} \triangleleft (f, u') \quad (\alpha \upharpoonright \tilde{\pi} \text{ is defined})
\end{align*}
\]
Action Interpretation: Spawn

\[ [\tilde{\alpha} ; \overline{A_Q} (\tilde{\pi} : A' \tilde{\alpha} \triangleleft B) \overline{A}] \longrightarrow [\tilde{\alpha} ; \overline{A_Q} (\tilde{\pi} : \tilde{\alpha} \triangleleft B) \overline{A} A'] \]
Action Interpretation: Message send

$$[\bar{\alpha} ; \overline{A_Q}(\bar{\pi} : \langle v \rangle \bar{a} \triangleleft B) \overline{A}] \rightarrow [\bar{\alpha} \langle v \rangle ; \overline{A_Q}(\bar{\pi} : \bar{a} \triangleleft B) \overline{A}]$$
Logging: Requirements Scorecard

Route log entries from producers to consumers  ✔️ pub/sub
Consumers filter log messages  ✔️ pub/sub
Decouple producers from consumers  ✔️ pub/sub
Avoid shared-state explosion  ✔️ pub/sub
Discovery of logging service  ✗ no need!
Only produce if someone's listening  ✗
Alert when a producer crashes/ exits  ✗
Uniform treatment of I/O  ✔️ pub/sub
PART III: Why Routing Events? How?
Logging: Requirements Scorecard

- Route log entries from producers to consumers  
  - Consumers filter log messages  
  - Decouple producers from consumers  
  - Avoid shared-state explosion  
  - Discovery of logging service
- Only produce if someone's listening
- Alert when a producer crashes/ exits
- Uniform treatment of I/O
Shared Conversational Interest

\[
\pi = (p) \mid \langle p \rangle
\]

\[
\langle p \rangle \cap \langle p' \rangle = \emptyset
\]

\[
(q) \cap (q') = \emptyset
\]

\[
\langle p \rangle \cap (q) = \langle p \cap q \rangle
\]

\[
(q) \cap \langle p \rangle = (p \cap q)
\]

Any pattern language will do — if it supports \( \cap \)
What is a Routing Event?

\{\tilde{\pi}\}

From Actor to Network

\tilde{\pi}_{old} \quad \tilde{\pi}_{new}

From Network to Actor

\{\tilde{\pi}_\blacklozenge\}
Routing Events for Service Discovery

Client

\{ \langle srv, \star \rangle \}
Routing Events for Service Discovery
Routing Events for Presence Detection

\[
\langle \text{log, [P1, *, *]} \rangle
\]
Routing Events for Presence Detection

\[
\{\text{log, [P1, error, \star]}\} \cap \{\text{log, [P1, error, \star]}\} = \{\text{log, [P1, error, \star]}\}
\]
Routing Events for Crash Detection

\[\langle \text{log, [P1, \star, \star]} \rangle\] \quad (\text{log, [\star, \text{error, \star}]}\]

cf. Erlang's links/monitors [Armstrong 2003]
Basic Actor Model + Pub/sub + Routing Events

\[ C \models A \models \sum \models (f, u) \models A \models \bar{\alpha} \]

\( f = \text{Base language functions} \)
\( u = \text{Base language values} \)
\( B = (f, u) \quad \text{Behaviors} \)
\( \Sigma = \bar{a} \bowtie B \quad \text{Actor States} \)
\( A = \tilde{\pi} : \Sigma \quad \text{Actors} \)
\( C = [\bar{\alpha} ; \overline{A}] \quad \text{Configurations} \)

\( \alpha = \langle v \rangle \mid \{ \tilde{\pi} \} \quad \text{Events} \)
\( a = \alpha \mid A \quad \text{Actions} \)
\( v = u \mid v, v \quad \text{Message values} \)
\( p = u \mid p, p \mid * \quad \text{Message patterns} \)
\( \pi = (p) \mid \langle p \rangle \quad \text{Interests} \)
Action Interpretation: Routing event

\[ \overline{\alpha} ; \overline{A_Q}(\tilde{\pi} : \{\tilde{\pi}'\} \overline{a} \triangleleft B) \overline{A} \]

\[ \overrightarrow{} \quad [\overline{\alpha}\{\tilde{\pi}.\} ; \overline{A_Q}(\tilde{\pi}' : \overline{a} \triangleleft B) \overline{A}] \]

\[ \tilde{\pi}. = \text{interests}(\overline{A_Q}) \cup \tilde{\pi}' \cup \text{interests}(\overline{A}) \]
Event Filtering

\[ \alpha \mid \bar{\pi} : \alpha \times \bar{\pi} \rightarrow \alpha \]

\[ \nu \mid_p : \nu \times p \]

\[ \langle \nu \rangle \mid \bar{\pi} = \langle \nu \rangle, \text{ if } \exists (p) \in \bar{\pi} \text{ such that } \nu \mid_p \]

\[ \{ \bar{\pi}_1 \} \mid \bar{\pi}_2 = \{ (\pi_{11} \cap \pi_{21}) \cup \ldots \cup (\pi_{11} \cap \pi_{2m}) \cup \\
(\pi_{12} \cap \pi_{21}) \cup \ldots \cup (\pi_{12} \cap \pi_{2m}) \cup \\
\vdots \\
(\pi_{1n} \cap \pi_{21}) \cup \ldots \cup (\pi_{1n} \cap \pi_{2m}) \} \]

\[ \alpha \mid \bar{\pi} \text{ otherwise undefined} \]
Logging: Requirements Scorecard

Route log entries from producers to consumers  ✔ pub/sub
Consumers filter log messages  ✔ pub/sub
Decouple producers from consumers  ✔ pub/sub
Avoid shared-state explosion  ✔ pub/sub
Discovery of logging service  ✔ routing events
Only produce if someone's listening  ✔ routing events
Alert when a producer crashes/exit  ✔ routing events
Uniform treatment of I/O  ❗ not finished!
PART IV: Why Hierarchical Layering? How?
Logging: Requirements Scorecard

- Route log entries from producers to consumers ✓
- Consumers filter log messages ✓
- Decouple producers from consumers ✓
  - Avoid shared-state explosion ✓
  - Discovery of logging service ✓
- Only produce if someone's listening ✓
- Alert when a producer crashes/exports ✓
- Uniform treatment of I/O □ not finished!
Layers make I/O Uniform

`Logging` → `Pager communication`

`C1` → `Pager`

`magic!`
Layers Scope Conversations

\[ \tilde{\pi}_A \rightarrow \tilde{\pi}_B \rightarrow \tilde{\pi}_C \rightarrow \tilde{\pi}_C \rightarrow \tilde{\pi}_D \]
Layers Scope Conversations

\[ \tilde{\pi}_A \quad \tilde{\pi}_B \quad \tilde{\pi}_C \quad \tilde{\pi}_D \]

\[ \tilde{\pi}_{C1} \quad \tilde{\pi}_{C2} \]
Layers Compose
One Layer = One Protocol

Speak $L_3$ ($\&L_2$)

Speak $L_2$ ($\&L_1$)

Speak $L_1$
One Layer = One Protocol

App protocol (&SSH)

App (REPL)

Speak SSH (&TCP)

SSH commands

Speak TCP

TCP

TCP header

Encrypted payload

SSH command

App message
One Layer = One Protocol

Snoops via pub/sub to populate cache!

UDP header
DNS header
Question Answer Answer ...
$f = \text{Base language functions}$

$u = \text{Base language values}$

$B = (f, u) \mid C$ \hspace{1cm} \text{Behaviors}

$\Sigma = \overline{a} \triangleleft B$ \hspace{1cm} \text{Actor States}

$A = \tilde{\pi} : \Sigma$ \hspace{1cm} \text{Actors}

$C = [\overline{\alpha} ; \tilde{\pi}_\circ ; \overline{\cal{A}}]$ \hspace{1cm} \text{Configurations}

$\alpha = m \mid \{\tilde{\pi}\}$ \hspace{1cm} \text{Events}

$a = \alpha \mid A$ \hspace{1cm} \text{Actions}

$m = \langle v \rangle \mid \parallel \pi$ \hspace{1cm} \text{Messages}

$v = u \mid v, v$ \hspace{1cm} \text{Message values}

$p = u \mid p, p \mid \star$ \hspace{1cm} \text{Message patterns}

$\pi = (p) \mid \langle p \rangle \mid \parallel \pi$ \hspace{1cm} \text{Interests}
Event Interpretation, \( A \xrightarrow{\alpha} A \)

\[
\frac{\text{inj} \ (\alpha \mid \pi, C') = C''}{\pi : \llcorner C \xrightarrow{\alpha} \pi : \llcorner C'} \quad (\alpha \mid \pi \text{ is defined})
\]
Event Interpretation: Routing event arrival

\[ \text{inject} : \alpha \times C \rightarrow C \]

\[ \text{inject} (\{\tilde{\pi}\}, [\overline{x} ; \tilde{\pi} \o ; \overline{A}]) = [\overline{x} \{\tilde{\pi} \o\} ; \text{lift}(\tilde{\pi}) ; \overline{A}] \]

\[ \tilde{\pi} \o = \tilde{\pi}_1 \cup \tilde{\pi}_2 \cup \cdots \cup \tilde{\pi}_n \cup \text{lift}(\tilde{\pi}) \]
\[ C \rightarrow C' \]

\[ \bar{a} \triangleleft C \rightarrow \bar{a}' \triangleleft C' \]
Action Interpretation: Routing event (with layering)

\[
\bar{a}_0 \prec [\bar{\alpha} ; \tilde{\pi}_\circ ; \overline{A_Q}(\tilde{\pi} : \{\tilde{\pi}'\} \bar{a} \triangleleft B)\overline{A}]
\]

\[
\rightarrow \bar{a}_0\{\text{drop(\tilde{\pi}.)}\} \prec [\bar{\alpha}\{\tilde{\pi}.\tilde{\pi}_\circ\} ; \tilde{\pi}_\circ ; \overline{A_Q}(\tilde{\pi}' : \bar{a} \triangleleft B)\overline{A}]
\]

\[
\tilde{\pi}. = \text{interests}(\overline{A_Q}) \cup \tilde{\pi}' \cup \text{interests}(\overline{A})
\]
**Logging: Requirements Scorecard**

<table>
<thead>
<tr>
<th>Requirement</th>
<th>Pub/Sub</th>
<th>Routing Events</th>
<th>Layering</th>
</tr>
</thead>
<tbody>
<tr>
<td>Route log entries from producers to consumers</td>
<td>✔️ pub/sub</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Consumers filter log messages</td>
<td>✔️ pub/sub</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Decouple producers from consumers</td>
<td>✔️ pub/sub</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Avoid shared-state explosion</td>
<td>✔️ pub/sub</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Discovery of logging service</td>
<td>✔️ routing events</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Only produce if someone's listening</td>
<td>✔️ routing events</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Alert when a producer crashes/exits</td>
<td>✔️ routing events</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Uniform treatment of I/O</td>
<td>✔️ layering</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

+ great additional benefits from layering
PART V: Conclusions
<table>
<thead>
<tr>
<th>Marketplace</th>
<th>Minimart</th>
<th>JS-Marketplace</th>
</tr>
</thead>
<tbody>
<tr>
<td>Typed Racket</td>
<td>Racket</td>
<td>Javascript</td>
</tr>
<tr>
<td>DNS server (UDP)</td>
<td>Websocket driver</td>
<td></td>
</tr>
<tr>
<td>SSH server (TCP)</td>
<td>Generic msg broker</td>
<td></td>
</tr>
<tr>
<td>Chat server</td>
<td></td>
<td>Websocket driver</td>
</tr>
<tr>
<td>Echo server</td>
<td></td>
<td>DOM driver</td>
</tr>
<tr>
<td></td>
<td></td>
<td>jQuery driver</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Chat + roster</td>
</tr>
<tr>
<td></td>
<td></td>
<td>GUI composition</td>
</tr>
</tbody>
</table>

Details and experience report in the paper!
Thank you!

Actor Programming Language
+ Publish/Subscribe
+ Routing Events
+ Hierarchical Layering

Network Calculus
Actor Calculus (see paper)

Experience reports (see paper)

http://www.ccs.neu.edu/home/tonyg/marketplace/
Network Calculus Summary
\[ C \subseteq A \subseteq \Sigma \]

\[ (f, u) \]

\[ \alpha = m \mid \{\widetilde{\pi}\} \quad \text{Events} \]

\[ a = \alpha \mid A \quad \text{Actions} \]

\[ B = (f, u) \mid C \quad \text{Behaviors} \]

\[ \Sigma = \overline{a} \triangleleft B \quad \text{Actor States} \]

\[ A = \widetilde{\pi} : \Sigma \quad \text{Actors} \]

\[ C = [\overline{\alpha}; \widetilde{\pi}_o; \overline{A}] \quad \text{Configurations} \]

\[ \pi = (p) \mid \langle p \rangle \mid \upharpoonright \pi \quad \text{Subscriptions} \]

\[ m = \langle v \rangle \mid \upharpoonright m \quad \text{Messages} \]

\[ v = u \mid v, v \quad \text{Message values} \]

\[ p = u \mid p, p \mid * \quad \text{Message patterns} \]
\[ A = \tilde{\pi} : \bar{a} \triangleleft B \]

\[ A_Q = \tilde{\pi} : \cdot \triangleleft B \]

\[ A_I = \tilde{\pi} : \cdot \triangleleft B_I \]

\[ C = [\bar{\alpha} ; \tilde{\pi} ; \bar{A}] \]

\[ C_Q = [\cdot ; \tilde{\pi} ; \bar{A}_Q] \]

\[ C_I = [\cdot ; \tilde{\pi} ; \bar{A}_I] \]

\[ B_I = (f, u) \mid C_I \]
\[ \text{interests} : A \rightarrow \tilde{\pi} \]
\[ \text{interests}(\tilde{\pi} : \Sigma) = \tilde{\pi} \]

\[ \text{lift} : \tilde{\pi} \rightarrow \tilde{\pi} \]
\[ \text{lift}(\tilde{\pi}) = \bar{\pi} \]

\[ \text{drop} : \tilde{\pi} \rightarrow \tilde{\pi} \]
\[ \text{drop}(\tilde{\pi}) = \text{drop}'(\pi) \]

\[ \text{drop}'(\pi) = \begin{cases} 
\pi' & \text{if } \pi = \bar{\pi}' \\
\cdot & \text{otherwise}
\end{cases} \]
\( \alpha \mid_{\tilde{\pi}} : \alpha \times \tilde{\pi} \rightarrow \alpha \)

\( v \mid_p : v \times p \)

\( \langle v \rangle \mid_{\tilde{\pi}} = \langle v \rangle, \text{ if } \exists (p) \in \tilde{\pi} \text{ such that } v \mid_p \)

\( \downarrow m \mid_{\tilde{\pi}} = \downarrow m, \text{ if } m \mid_{\text{drop}(\tilde{\pi})} \)

\( \{ \tilde{\pi}_1 \} \mid_{\tilde{\pi}_2} = \{ (\pi_{11} \cap \pi_{21}) \cup \cdots \cup (\pi_{11} \cap \pi_{2m}) \cup (\pi_{12} \cap \pi_{21}) \cup \cdots \cup (\pi_{12} \cap \pi_{2m}) \cup \cdots \cup (\pi_{1n} \cap \pi_{21}) \cup \cdots \cup (\pi_{1n} \cap \pi_{2m}) \} \)

\( \alpha \mid_{\tilde{\pi}} \text{ otherwise undefined} \)
\[ \pi \sqcap \pi : \pi \times \pi \to \pi \]

\[
\langle p \rangle \sqcap \langle p' \rangle = \emptyset \\
(\langle q \rangle \sqcap \langle q' \rangle) = \emptyset \\
\langle p \rangle \sqcap (\langle q \rangle) = \langle p \cap q \rangle \\
(\langle q \rangle \sqcap \langle p \rangle) = (p \cap q)
\]
\[
p \cap p : p \times p \rightarrow p
\]
\[
u \cap u = u
\]
\[
p_{11}, p_{12} \cap p_{21}, p_{22} = (p_{11} \cap p_{21}), (p_{12} \cap p_{22})
\]
\[
p \cap \ast = p
\]
\[
\ast \cap p = p
\]
\[
\overline{u \mathbin{|} u}
\]
\[
\overline{v_1 \mathbin{|} p_1}
\overline{v_2 \mathbin{|} p_2}
\]
\[
\overline{v_1, v_2 \mathbin{|} p_1, p_2}
\]
\[
\overline{v \mathbin{|} \ast}
\]
$$\overline{A_Q} \xrightarrow{\alpha} A'$$

$$\overline{a} \triangleleft [\alpha \overline{\alpha_0} ; \overline{\pi}_0 ; \overline{A_Q}] \longrightarrow \overline{a} \triangleleft [\overline{\alpha}_0 ; \overline{\pi}_0 ; \overline{A'}]$$
\[ \tilde{\pi} : \cdot \triangleleft B \xrightarrow{\alpha} \tilde{\pi} : \cdot \triangleleft B \] (\alpha \mid \tilde{\pi} \text{ is undefined})

\[ f(\alpha \mid \tilde{\pi}, u) = (\overline{a}, u') \]

\[ \tilde{\pi} : \cdot \triangleleft (f, u) \xrightarrow{\alpha} \tilde{\pi} : \overline{a} \triangleleft (f, u') \] (\alpha \mid \tilde{\pi} \text{ is defined})

\[ \text{injective } (\alpha \mid \tilde{\pi}, C) = C'' \]

\[ \tilde{\pi} : \cdot \triangleleft C \xrightarrow{\alpha} \tilde{\pi} : \cdot \triangleleft C' \] (\alpha \mid \tilde{\pi} \text{ is defined})
inject : α × C → C

inject (m, [\bar{\alpha}; \tilde{\pi}_o; \bar{A}]) = [\bar{\alpha}|m; \tilde{\pi}_o; \bar{A}]
\[ \text{inject} : \alpha \times C \rightarrow C \]

\[ \text{inject} (\{\tilde{\pi}\}, [\overline{\alpha} ; \tilde{\pi}_o ; \overline{A}]) = [\overline{\alpha} \{\tilde{\pi}_\bullet\} ; \text{lift}(\tilde{\pi}) ; \overline{A}] \]

\[ \tilde{\pi}_\bullet = \tilde{\pi}_1 \cup \tilde{\pi}_2 \cup \cdots \cup \tilde{\pi}_n \cup \text{lift}(\tilde{\pi}) \]
\[ \bar{a}_0 \triangleleft [\bar{\alpha} ; \tilde{\pi}_\circ ; \overline{A_Q}(\tilde{\pi} : \langle v \rangle \bar{a} \triangleleft B)\overline{A}] \rightarrow \bar{a}_0 \triangleleft [\bar{\alpha} \langle v \rangle ; \tilde{\pi}_\circ ; \overline{A_Q}(\tilde{\pi} : \bar{a} \triangleleft B)\overline{A}] \]
\[ \bar{a}_0 \triangleleft [\bar{\alpha} \ ; \ \tilde{\pi}_\circ \ ; \ \overline{A_Q}(\tilde{\pi} : |m\bar{a} \triangleleft B)\overline{A}] \rightarrow \bar{a}_0 m \triangleleft [\bar{\alpha} \ ; \ \tilde{\pi}_\circ \ ; \ \overline{A_Q}(\tilde{\pi} : \bar{a} \triangleleft B)\overline{A}] \]
\[
\begin{align*}
\tilde{a}_0 & \triangleleft [\tilde{\alpha} ; \tilde{\pi}_\circ ; \overline{A_Q}(\tilde{\pi} : \{\tilde{\pi}'\} \tilde{a} \triangleleft B) \overline{A}] \\
\longrightarrow \quad \tilde{a}_0 \{\text{drop}(\tilde{\pi}_\bullet)\} & \triangleleft [\tilde{\alpha} \{\tilde{\pi}_\bullet \tilde{\pi}_\circ\} ; \tilde{\pi}_\circ ; \overline{A_Q}(\tilde{\pi}' : \tilde{a} \triangleleft B) \overline{A}]
\end{align*}
\]

\[\tilde{\pi}_\bullet = \text{interests}(\overline{A_Q}) \cup \tilde{\pi}' \cup \text{interests}(\overline{A})\]
\[ \bar{a}_0 \triangleleft [\bar{\alpha} \; ; \; \bar{\pi}_o \; ; \; \overline{A_Q}(\bar{\pi} : A' \bar{a} \triangleleft B) \overline{A}] \]
\[ \rightarrow \bar{a}_0\{\text{drop}(\bar{\pi}_\bullet)\} \triangleleft [\bar{\alpha}\{\bar{\pi}_\bullet \bar{\pi}_o\} \; ; \; \bar{\pi}_o \; ; \; \overline{A_Q}(\bar{\pi} : \; \bar{a} \triangleleft B) \overline{A} \; A'] \]

\[ \tilde{\pi}_\bullet = \text{interests}(\overline{A_Q}) \cup \tilde{\pi} \cup \text{interests}(\overline{A}) \cup \text{interests}(A') \]
\[
\begin{align*}
\text{·} \triangleleft B & \rightarrow \bar{a} \triangleleft B' \\
\bar{a}_0 \triangleleft [\cdot; \bar{\pi}_o; \overline{A_I(\bar{\pi} \triangleleft B)} \overline{A_Q}] & \rightarrow \bar{a}_0 \triangleleft [\cdot; \bar{\pi}_o; \overline{A_Q} \overline{A_I(\bar{\pi} \triangleleft \bar{a} \triangleleft B')}]
\end{align*}
\]
\[
\frac{f(\alpha \mid \tilde{\pi}, u) = \text{exception}}{\tilde{\pi} : \cdot \triangleq (f, u) \xrightarrow{\alpha} \tilde{\pi} : \cdot \triangleq (\cdot, \cdot)} (\alpha \mid \tilde{\pi} \text{ is defined})
\]