NUS National University of Singapore
School of Computing

APLAS 2018

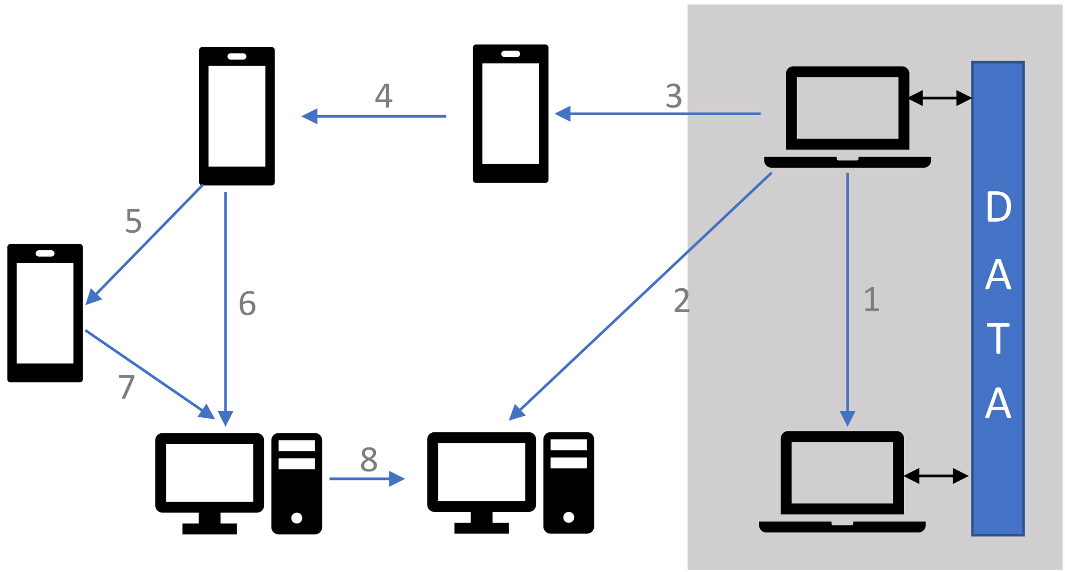
Automated Modular Verification for *Relaxed* Communication Protocols

by Andreea Costea, Wei-Ngan Chin, Shengchao Qin, Florin Craciun

Automated Modular Verification for *Relaxed* Communication Protocols

Why another formalism for describing protocols?

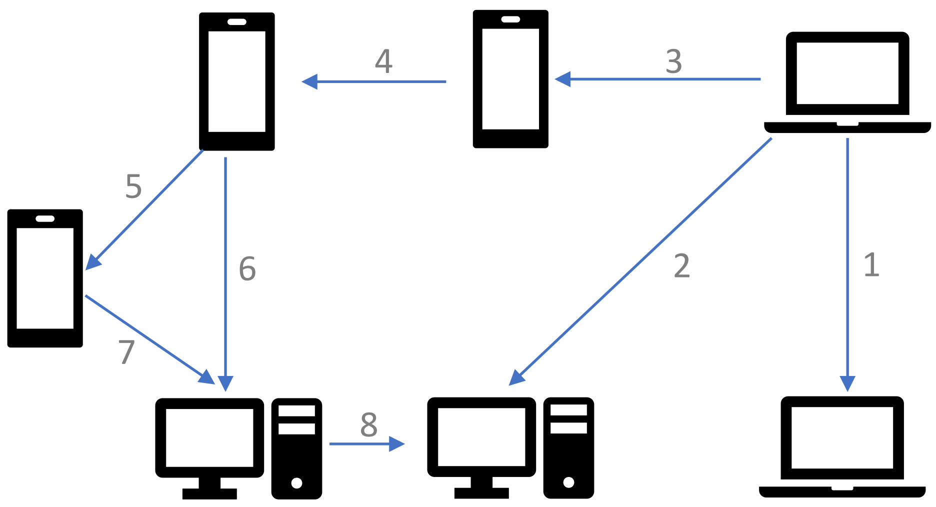
(more than 800 works spawned from [Igarashi & Kobayashi @TCS'04, Honda et al. @POPL'08])



A1: Communicating entities are loosely coupled.

Why another formalism for describing protocols?

(more than 800 works spawned from [Igarashi & Kobayashi @TCS'04, Honda et al. @POPL'08])

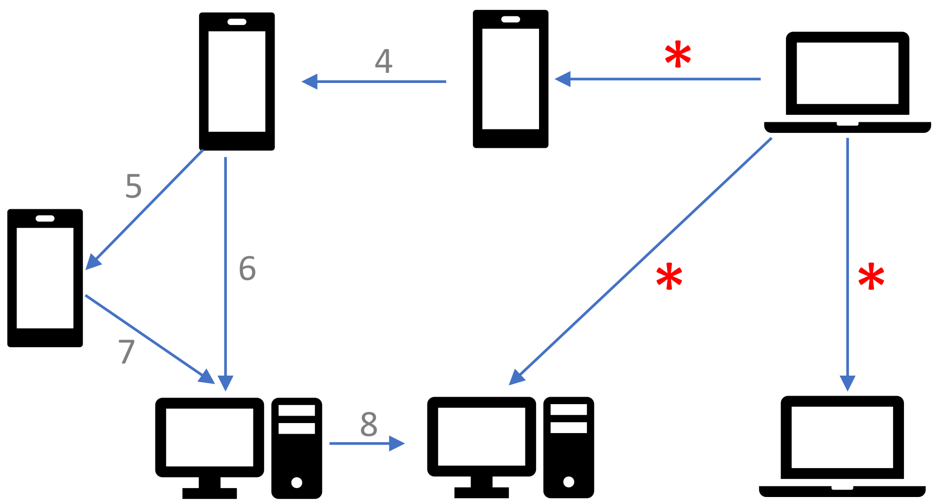


A1: Communicating entities are loosely coupled.

A2: Nondeterminism is generally undesirable.

Why another formalism for describing protocols?

(more than 800 works spawned from [Igarashi & Kobayashi @TCS'04, Honda et al. @POPL'08])



A1: Communicating entities are loosely coupled.

A2: Nondeterminism is generally undesirable.

How?

Session types **+** Separation logic

(friendly – compact syntax) (excellent for resource sharing)

Session logic

✓ *resource-aware protocols**

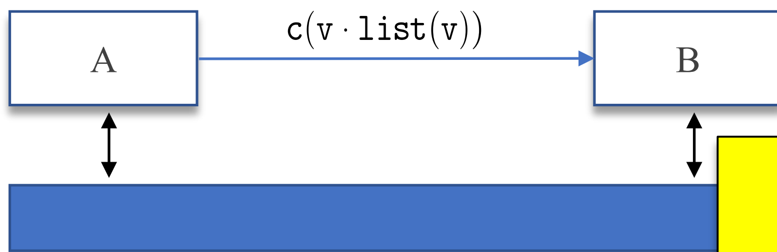
✓ *Reasoning about fine-grained concurrency*

✓ *enables Hoare-style verification guided by protocols*

*[Villard et al. @TACAS’10, Caires & Seco @POPL’13, Pfenning et al. @LICS’18]

A Simple Example

Example 1: resource sharing



Hoare-style verification guided by *resource-aware* communication protocols

$$\mathtt{A} \to \mathtt{B} : \mathtt{c}\langle \mathtt{v} \cdot \mathtt{list}(\mathtt{v}) \rangle$$

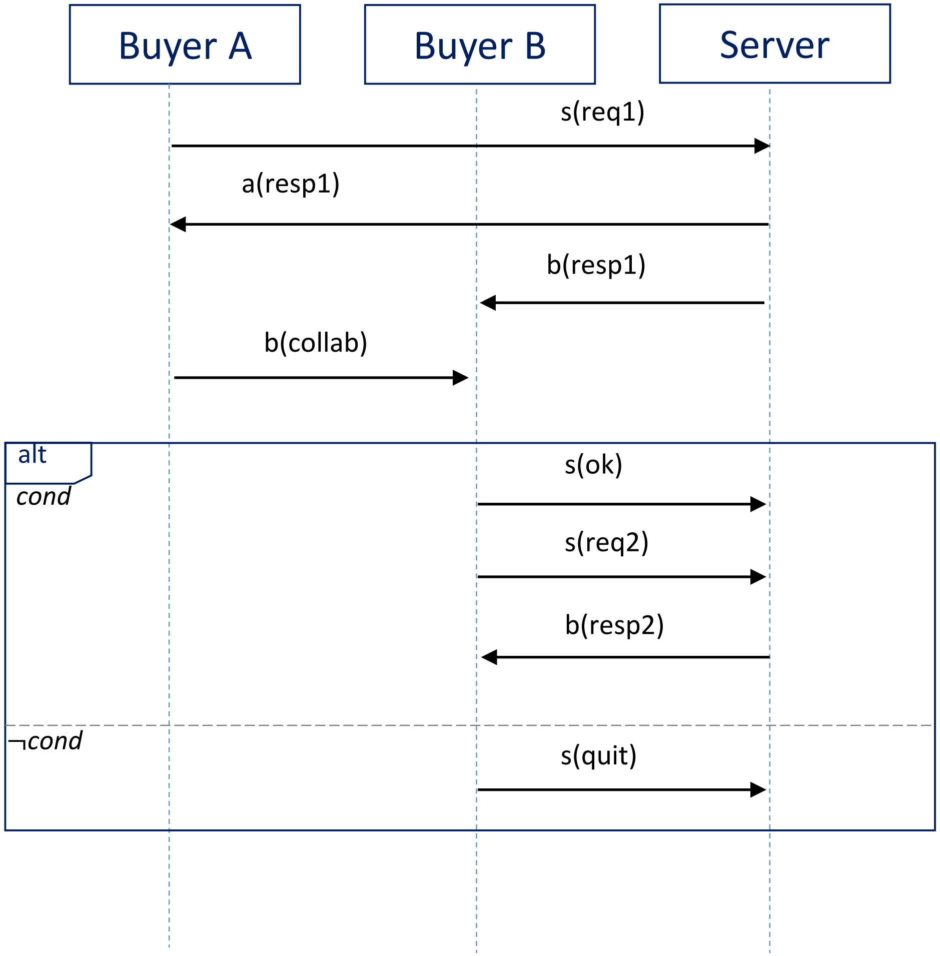
SPECIFY

VERIFY

A	B
…	…
$\{\Delta_A * \mathtt{list}(y)\}$	$\{\Delta_B\}$
`send(c, y);`	`x = receive(c);`
$\{\Delta'_A\}$	$\{\Delta'_B * \mathtt{list}(x)\}$

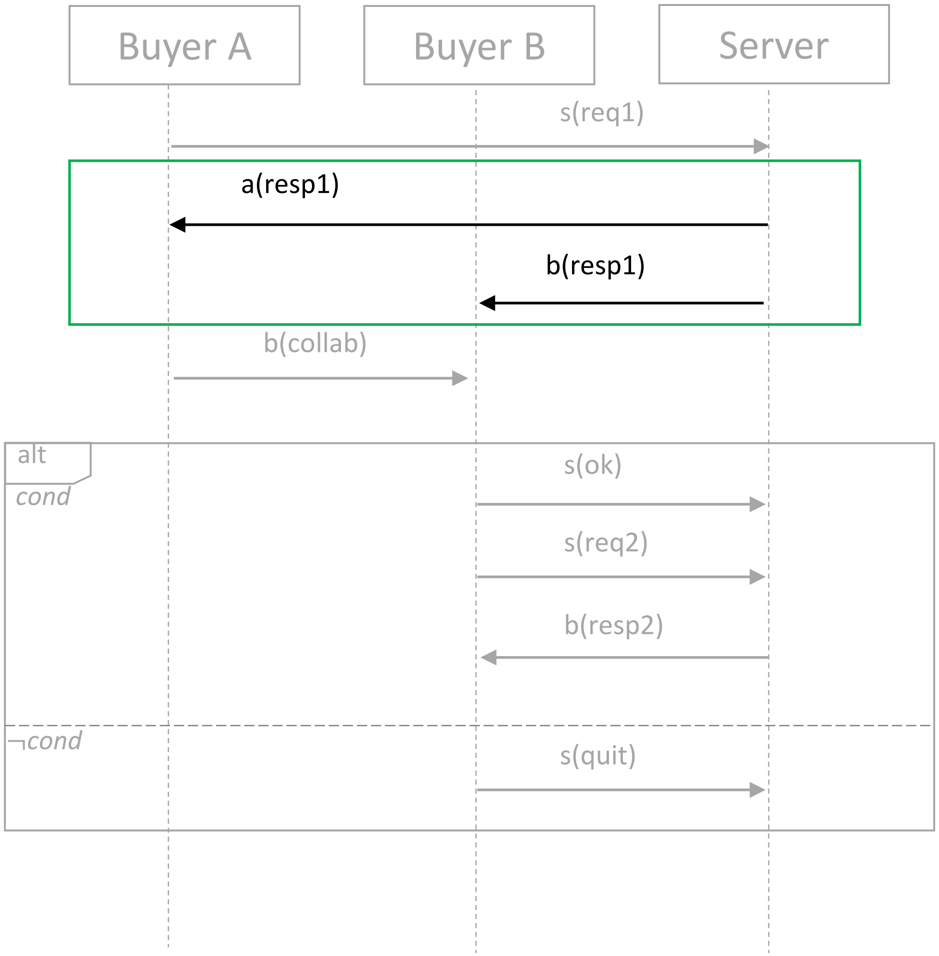
A Telling Example

Example 2



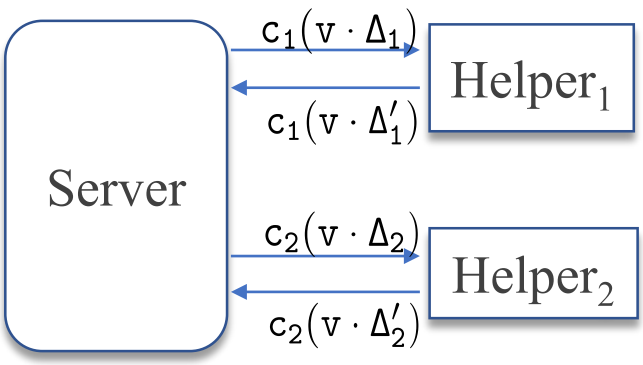
Buyer A
Buyer B
Server
s(req1)
a(resp1)
b(resp1)
b(collab)
alt
cond
s(ok)
s(req2)
b(resp2)
¬cond
s(quit)

Example 2



Buyer A
Buyer B
Server
s(req1)
a(resp1)
b(resp1)
b(collab)
alt
cond
s(ok)
s(req2)
b(resp2)
¬cond
s(quit)

Example 2 – fine-grained concurrency



$$G \triangleq (S \to H_1 : c_1\langle v \cdot \Delta_1 \rangle \,;\, H_1 \to S : c_1\langle v \cdot \Delta_1' \rangle) * (S \to H_2 : c_2\langle v \cdot \Delta_2 \rangle \,;\, H_2 \to S : c_2\langle v \cdot \Delta_2' \rangle).$$

sequence — concurrency — sequence

In classic MPST this protocol FAILS the consistency checks!

Our goal: support more *RELAXED* order of communication!

$$\mathtt{G} \triangleq \big(\mathtt{S}{\rightarrow}\mathtt{H}_1 : \mathtt{c}_1\langle \mathtt{v}\cdot\Delta_1\rangle\,;\,\mathtt{H}_1{\rightarrow}\mathtt{S} : \mathtt{c}_1\langle \mathtt{v}\cdot\Delta_1'\rangle\big) * \big(\mathtt{S}{\rightarrow}\mathtt{H}_2 : \mathtt{c}_2\langle \mathtt{v}\cdot\Delta_2\rangle\,;\,\mathtt{H}_2{\rightarrow}\mathtt{S} : \mathtt{c}_2\langle \mathtt{v}\cdot\Delta_2'\rangle\big).$$

$$\mathtt{G} \triangleq (\mathtt{S} {\rightarrow} \mathtt{H}_1 : \mathtt{c}_1\langle \mathtt{v} \cdot \Delta_1 \rangle \,;\, \mathtt{H}_1 {\rightarrow} \mathtt{S} : \mathtt{c}_1\langle \mathtt{v} \cdot \Delta_1' \rangle) * (\mathtt{S} {\rightarrow} \mathtt{H}_2 : \mathtt{c}_2\langle \mathtt{v} \cdot \Delta_2 \rangle \,;\, \mathtt{H}_2 {\rightarrow} \mathtt{S} : \mathtt{c}_2\langle \mathtt{v} \cdot \Delta_2' \rangle).$$

SPECIFY

VERIFY (server's side)

$$\mathtt{G} \triangleq (\mathtt{S}\to\mathtt{H}_1 : \mathtt{c}_1\langle \mathtt{v}\cdot\Delta_1\rangle \, ; \, \mathtt{H}_1\to\mathtt{S} : \mathtt{c}_1\langle \mathtt{v}\cdot\Delta_1'\rangle) * (\mathtt{S}\to\mathtt{H}_2 : \mathtt{c}_2\langle \mathtt{v}\cdot\Delta_2\rangle \, ; \, \mathtt{H}_2\to\mathtt{S} : \mathtt{c}_2\langle \mathtt{v}\cdot\Delta_2'\rangle).$$

SPECIFY

VERIFY (server's side)

(i)

```
send(c1,fd.vid);
send(c2,fd.aud);
fd.vid = receive(c1);
fd.aud = receive(c2);
```

(ii)

```
send(c1,fd.vid);
fd.vid = receive(c1);
send(c2,fd.aud);
fd.aud = receive(c2);
```

(iii)

```
send(c2,fd.aud);
fd.aud = receive(c2);
send(c1,fd.vid);
fd.vid = receive(c1);
```

RELAXED protocols: support for intra- and inter-process concurrency!

```
(send(c1,fd.vid); fd.vid = receive(c1);)
                ||
 send(c2,fd.aud); fd.aud = receive(c2);)
```

MPST

(v)

```
  (send(c1,fd.vid); || send(c2,fd.aud);)
                    ;
(fd.vid = receive(c1); || fd.aud = receive(c2);
```

MPST

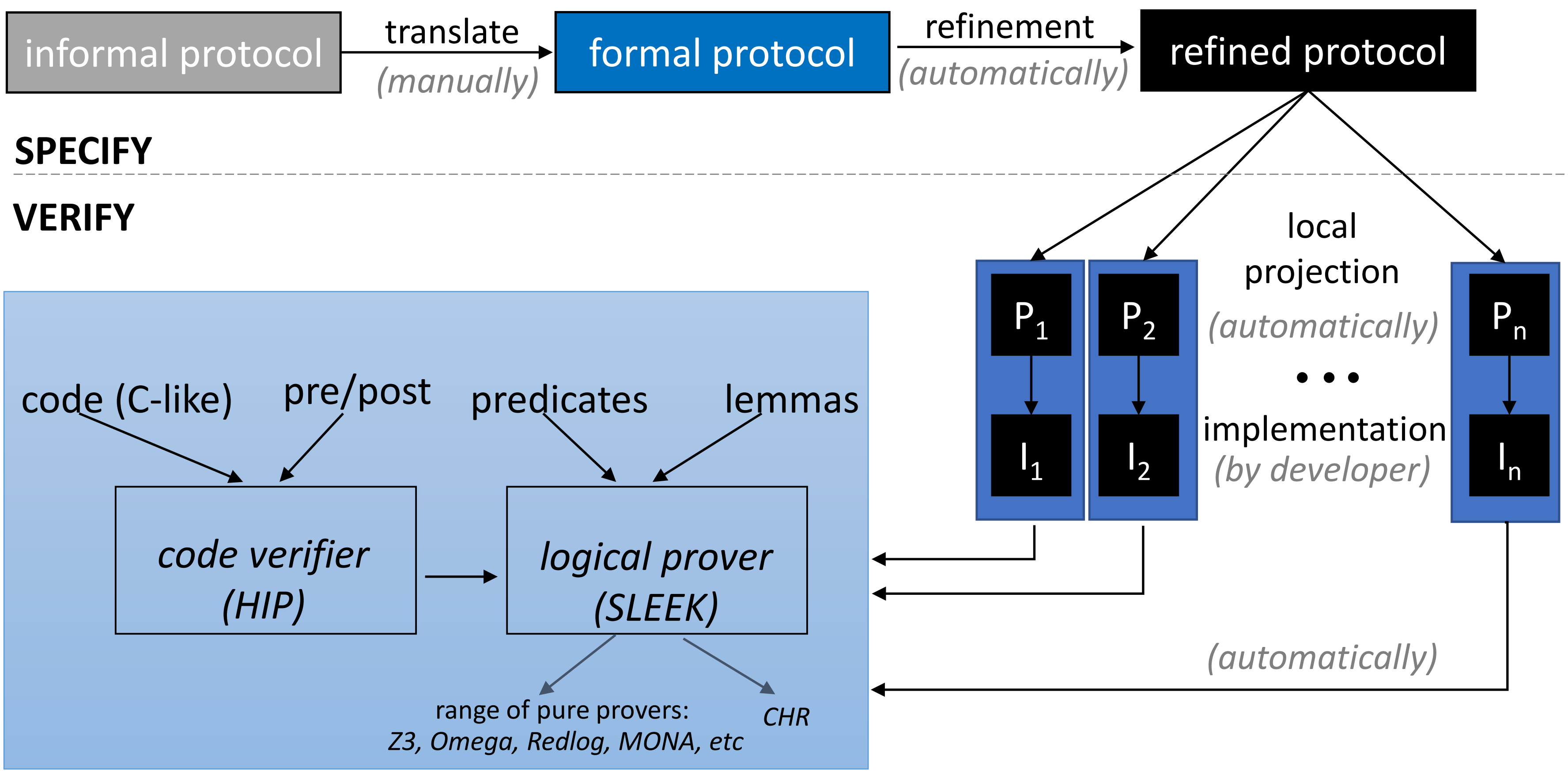
Specification Language

Relaxed Protocols

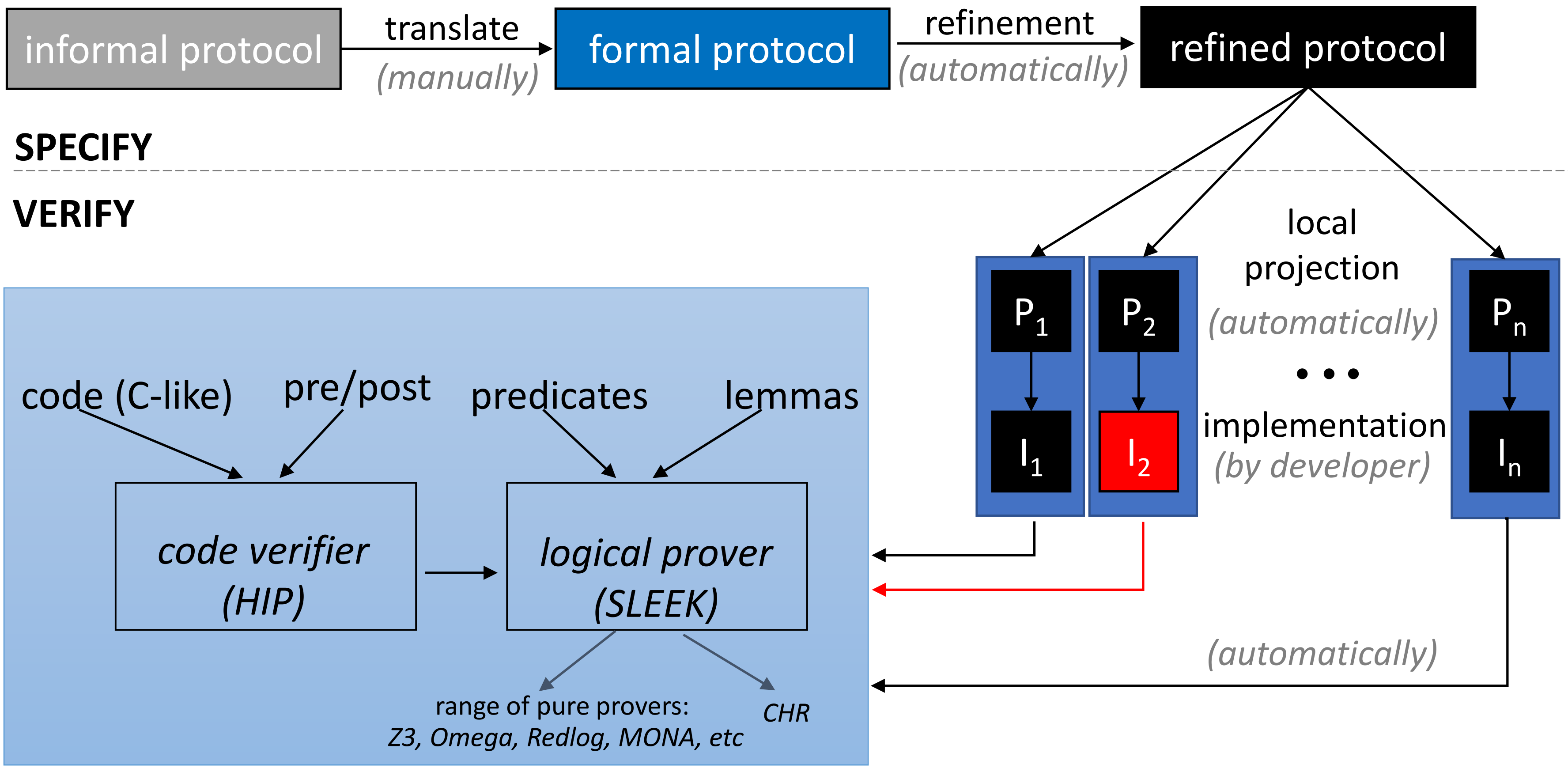
Global protocol	G	$::=$	
Single transmission			$\mathtt{S}{\rightarrow}\mathtt{R} : \mathtt{c}\langle \mathtt{v} \cdot \Delta \rangle$
Concurrency			$\mid G * G$
Choice			$\mid G \vee G$
Sequencing			$\mid G \,;\, G$
Quantification			$\mid \exists\ c^*\ P^*\ v^* \cdot G$
Inaction			$\mid \mathtt{emp}$

$(Parties)\ \mathtt{P}, \mathtt{S}, \mathtt{R} \in \mathcal{R}\mathtt{ole} (Channels)\ \mathtt{c} \in \mathcal{C}\mathtt{han}\ \ (Messages)\ \mathtt{v} \cdot \Delta$

Framework Overview



Framework Overview



Local Projection

per party projection → | per channel projection →

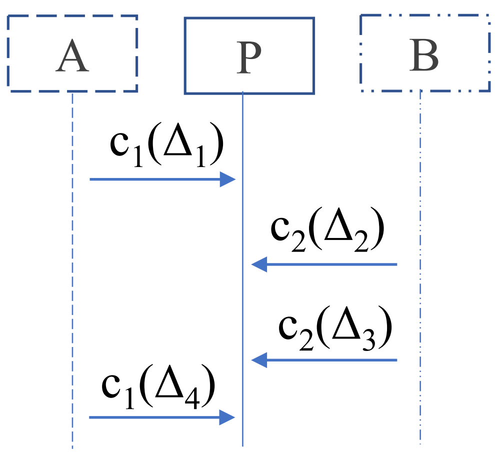
Global protocol	G ::=	Υ ::=	$\mathtt{L}$ ::=
Single transmission	$\mathtt{S}\xrightarrow{i}\mathtt{R} : \mathtt{c}\langle \mathtt{v} \cdot \Delta \rangle$	$\mathtt{c}!\mathtt{v} \cdot \Delta \mid \mathtt{c}?\mathtt{v} \cdot \Delta$	$!\mathtt{v} \cdot \Delta \mid ?\mathtt{v} \cdot \Delta$
Concurrency	$\mid G * G$	$\mid \Upsilon * \Upsilon$	
Choice	$\mid G \vee G$	$\mid \Upsilon \vee \Upsilon$	$\mid \mathtt{L} \vee \mathtt{L}$
Sequencing	$\mid G \,;\, G$	$\mid \Upsilon ; \Upsilon$	$\mid \mathtt{L};\mathtt{L}$
Quantification	$\mid \exists\ \mathtt{c}^*\ \mathtt{P}^*\ \mathtt{v}^* \cdot \mathtt{G}$	$\mid \exists \mathtt{c}^*, \mathtt{v}^* \cdot \Upsilon$	$\mid \exists \mathtt{v}^* \cdot \mathtt{L}$
Inaction	$\mid$ emp	$\mid$ emp	$\mid$ emp

Local Projection

per party projection → per channel projection

Global protocol	G ::=	Υ ::=	L ::=
Single transmission	$\mathtt{S}\xrightarrow{i}\mathtt{R} : \mathtt{c}\langle \mathtt{v}\cdot\Delta\rangle$	$\mathtt{c}!\mathtt{v}\cdot\Delta \mid \mathtt{c}?\mathtt{v}\cdot\Delta$	$!\mathtt{v}\cdot\Delta \mid ?\mathtt{v}\cdot\Delta$
Concurrency	$\mid G * G$	$\mid \Upsilon * \Upsilon$	
Choice	$\mid G \vee G$	$\mid \Upsilon \vee \Upsilon$	$\mid L \vee L$
Sequencing	$\mid G \,;\, G$	$\mid \Upsilon ; \Upsilon$	$\mid L ; L$
Quantification	$\mid \exists\ \mathtt{c}^*\ \mathtt{P}^*\ \mathtt{v}^* \cdot \mathtt{G}$	$\mid \exists \mathtt{c}^*, \mathtt{v}^* \cdot \Upsilon$	$\mid \exists \mathtt{v}^* \cdot L$
Inaction	$\mid$ emp	$\mid$ emp	$\mid$ emp

$$\mathtt{A}\rightarrow\mathtt{P} : \mathtt{c}_1\langle\Delta_1\rangle\ ;\ \mathtt{B}\rightarrow\mathtt{P} : \mathtt{c}_2\langle\Delta_2\rangle\ ;\ \mathtt{B}\rightarrow\mathtt{P} : \mathtt{c}_2\langle\Delta_3\rangle\ ;\ \mathtt{A}\rightarrow\mathtt{P} : \mathtt{c}_1\langle\Delta_4\rangle.$$

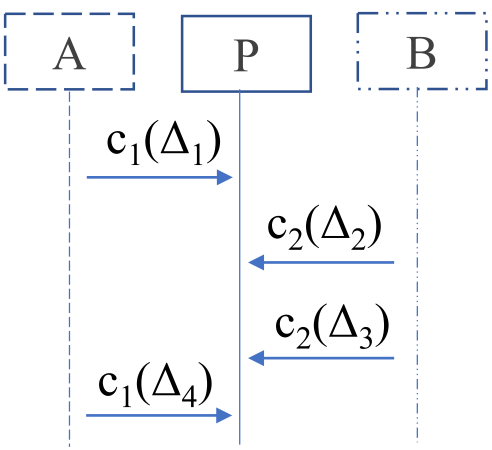


Local Projection

per party projection →

per channel projection →

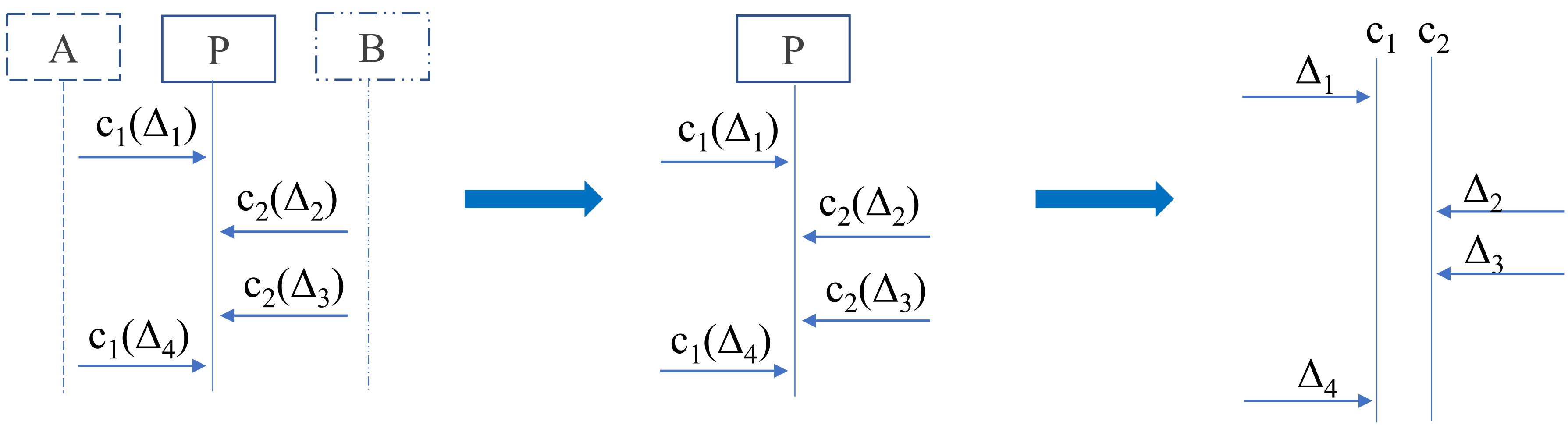
Global protocol	$G ::=$	$\Upsilon ::=$	$L ::=$
Single transmission	$S \xrightarrow{i} R : c\langle v \cdot \Delta \rangle$	$c!v \cdot \Delta \mid c?v \cdot \Delta$	$!v \cdot \Delta \mid ?v \cdot \Delta$
Concurrency	$\mid G * G$	$\mid \Upsilon * \Upsilon$	
Choice	$\mid G \vee G$	$\mid \Upsilon \vee \Upsilon$	$\mid L \vee L$
Sequencing	$\mid G ; G$	$\mid \Upsilon ; \Upsilon$	$\mid L ; L$
Quantification	$\mid \exists\ c^*\ P^*\ v^* \cdot G$	$\mid \exists c^*, v^* \cdot \Upsilon$	$\mid \exists v^* \cdot L$
Inaction	$\mid$ emp	$\mid$ emp	$\mid$ emp



Local Projection

per party projection → per channel projection

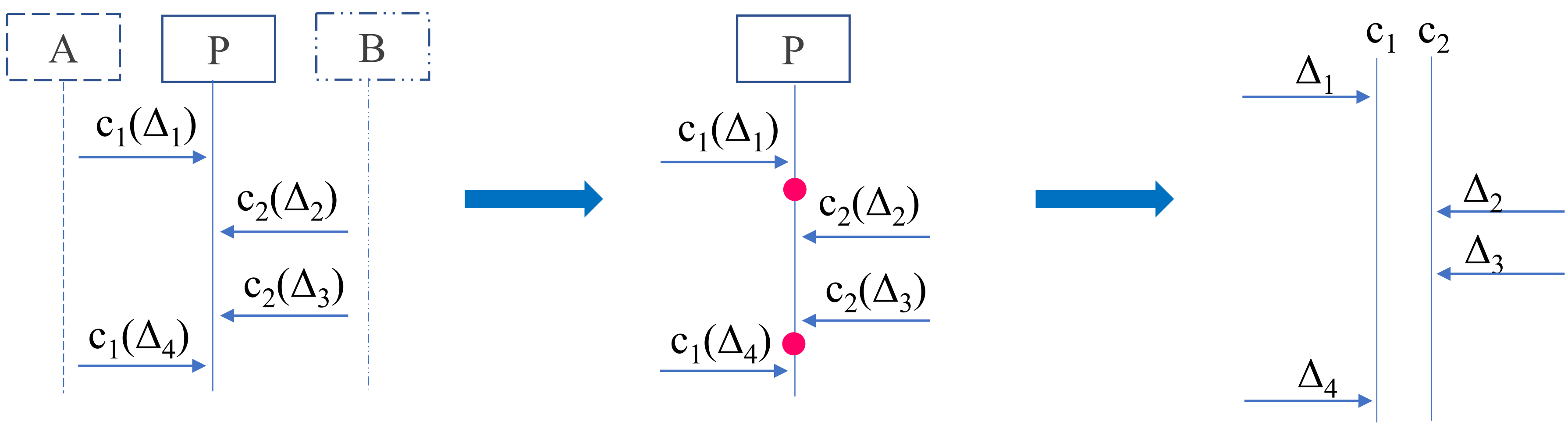
	$G ::=$	$\Upsilon ::=$	$L ::=$
Global protocol			
Single transmission	$S \xrightarrow{i} R : c\langle v \cdot \Delta \rangle$	$c!v \cdot \Delta \mid c?v \cdot \Delta$	$!v \cdot \Delta \mid ?v \cdot \Delta$
Concurrency	$\mid G * G$	$\mid \Upsilon * \Upsilon$	
Choice	$\mid G \vee G$	$\mid \Upsilon \vee \Upsilon$	$\mid L \vee L$
Sequencing	$\mid G ; G$	$\mid \Upsilon ; \Upsilon$	$\mid L ; L$
Quantification	$\mid \exists\ c^*\ P^*\ v^* \cdot G$	$\mid \exists c^*, v^* \cdot \Upsilon$	$\mid \exists v^* \cdot L$
Inaction	$\mid$ emp	$\mid$ emp	$\mid$ emp



Local Projection

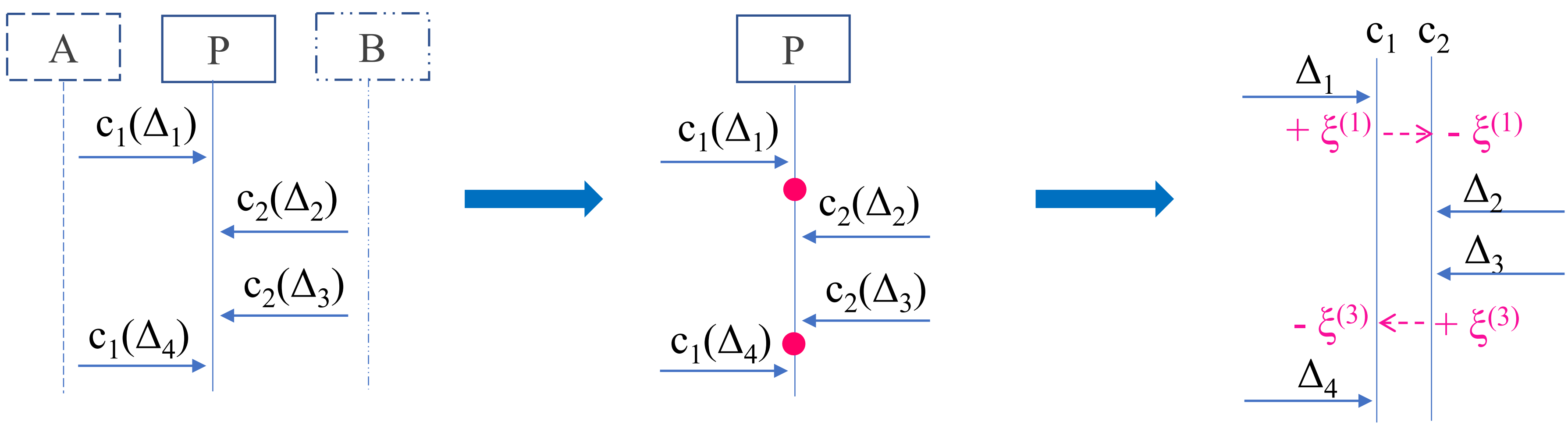
per party projection →   per channel projection →

Global protocol	$G ::=$	$\Upsilon ::=$	$\mathtt{L} ::=$
Single transmission	$\mathtt{S} \xrightarrow{i} \mathtt{R} : \mathtt{c}\langle \mathtt{v} \cdot \Delta \rangle$	$\mathtt{c!v} \cdot \Delta \mid \mathtt{c?v} \cdot \Delta$	$\mathtt{!v} \cdot \Delta \mid \mathtt{?v} \cdot \Delta$
Concurrency	$\mid G * G$	$\mid \Upsilon * \Upsilon$	
Choice	$\mid G \vee G$	$\mid \Upsilon \vee \Upsilon$	$\mid \mathtt{L} \vee \mathtt{L}$
Sequencing	$\mid G ; G$	$\mid \Upsilon ; \Upsilon$	$\mid \mathtt{L};\mathtt{L}$
Quantification	$\mid \exists\ \mathtt{c}^*\ \mathtt{P}^*\ \mathtt{v}^* \cdot \mathtt{G}$	$\mid \exists \mathtt{c}^*, \mathtt{v}^* \cdot \Upsilon$	$\mid \exists \mathtt{v}^* \cdot \mathtt{L}$
Inaction	$\mid$ emp	$\mid$ emp	$\mid$ emp



Local Projection

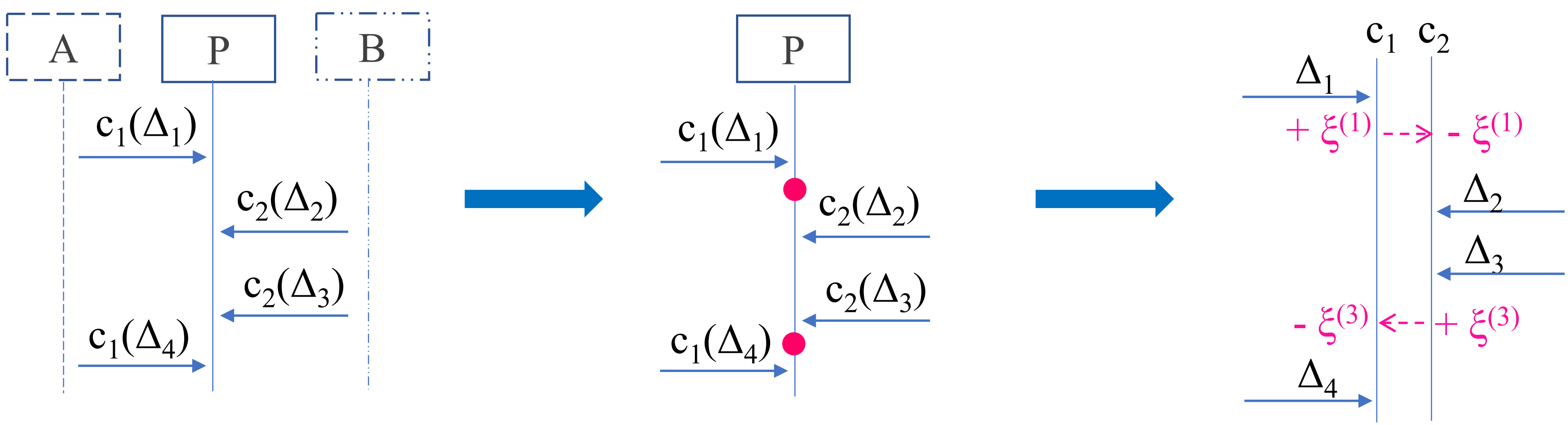
	Global protocol $G ::=$	per party projection → $\Upsilon ::=$	per channel projection → $\mathtt{L} ::=$
Single transmission	$\mathtt{S}\xrightarrow{i}\mathtt{R} : \mathtt{c}\langle \mathtt{v}\cdot\Delta\rangle$	$\mathtt{c!v}\cdot\Delta \mid \mathtt{c?v}\cdot\Delta$	$\mathtt{!v}\cdot\Delta \mid \mathtt{?v}\cdot\Delta$
Concurrency	$\mid G * G$	$\mid \Upsilon * \Upsilon$	
Choice	$\mid G \vee G$	$\mid \Upsilon \vee \Upsilon$	$\mid \mathtt{L} \vee \mathtt{L}$
Sequencing	$\mid G\, ;\, G$	$\mid \Upsilon ; \Upsilon$	$\mid \mathtt{L};\mathtt{L}$
Quantification	$\mid \exists\ \mathtt{c}^*\ \mathtt{P}^*\ \mathtt{v}^*\cdot \mathtt{G}$	$\mid \exists \mathtt{c}^*, \mathtt{v}^* \cdot \Upsilon$	$\mid \exists \mathtt{v}^* \cdot \mathtt{L}$
Inaction	$\mid \mathtt{emp}$	$\mid \mathtt{emp}$	$\mid \mathtt{emp}$



Local Projection

per party projection → per channel projection →

	Global protocol G ::=	Υ ::=	L ::=
Single transmission	$S \xrightarrow{i} R : c\langle v \cdot \Delta \rangle$	$c!v \cdot \Delta \mid c?v \cdot \Delta$	$!v \cdot \Delta \mid ?v \cdot \Delta$
Concurrency	$\mid G * G$	$\mid \Upsilon * \Upsilon$	
Choice	$\mid G \vee G$	$\mid \Upsilon \vee \Upsilon$	$\mid L \vee L$
Sequencing	$\mid G ; G$	$\mid \Upsilon ; \Upsilon$	$\mid L ; L$
Quantification	$\mid \exists\ c^*\ P^*\ v^* \cdot G$	$\mid \exists c^*, v^* \cdot \Upsilon$	$\mid \exists v^* \cdot L$
Inaction	$\mid$ emp	$\mid$ emp	$\mid$ emp
Fence	$\mid \xi(\{P^*\}, c, n)$	$\mid \xi(\{P\}, c, n)$	$\mid \oplus(\xi^{(n)}) \mid \ominus(\xi^{(n)})$



Local Projection

		per party projection →	per channel projection →
Global protocol	G ::=	Υ ::=	$\mathtt{L}$ ::=
Single transmission	$\mathtt{S}\xrightarrow{i}\mathtt{R} : \mathtt{c}\langle \mathtt{v}\cdot\Delta\rangle$	$\mathtt{c!v}\cdot\Delta \mid \mathtt{c?v}\cdot\Delta$	$\mathtt{!v}\cdot\Delta \mid \mathtt{?v}\cdot\Delta$
Concurrency	$\mid G * G$	$\mid \Upsilon * \Upsilon$	
Choice	$\mid G \vee G$	$\mid \Upsilon \vee \Upsilon$	$\mid \mathtt{L}\vee\mathtt{L}$
Sequencing	$\mid G ; G$	$\mid \Upsilon;\Upsilon$	$\mid \mathtt{L};\mathtt{L}$
Quantification	$\mid \exists\ \mathtt{c}^*\ \mathtt{P}^*\ \mathtt{v}^*\cdot \mathtt{G}$	$\mid \exists \mathtt{c}^*, \mathtt{v}^* \cdot \Upsilon$	$\mid \exists \mathtt{v}^* \cdot \mathtt{L}$
Inaction	$\mid$ emp	$\mid$ emp	$\mid$ emp
Fence	$\mid \xi(\{\mathtt{P}^*\}, \mathtt{c}, \mathtt{n})$	$\mid \xi(\{\mathtt{P}\}, \mathtt{c}, \mathtt{n})$	$\mid \oplus(\xi^{(\mathtt{n})}) \mid \ominus(\xi^{(\mathtt{n})})$

VERIFY

HO predicate example:

$\mathcal{C}(\mathtt{c}, \mathtt{P}, \mathtt{L})$ - associates a specification $\mathtt{L}$ to a channel $\mathtt{c}$ which is manipulated by party $\mathtt{P}$.

$$
\begin{array}{lll}
[\mathbf{L}+] & \mathcal{C}(\mathtt{c}_1, \mathtt{P}, \oplus(\xi^{(\mathtt{n})});\mathtt{L}) & \mapsto\ \mathcal{C}(\mathtt{c}_1, \mathtt{P}, \mathtt{L}) \wedge \xi^{(\mathtt{n})}. \\
[\mathbf{L}-] & \mathcal{C}(\mathtt{c}_1, \mathtt{P}, \ominus(\xi^{(\mathtt{n})});\mathtt{L}) \wedge \xi^{(\mathtt{n})} & \mapsto\ \mathcal{C}(\mathtt{c}_1, \mathtt{P}, \mathtt{L}).
\end{array}
$$

Communication Primitives

$$\frac{}{\vdash \{\texttt{true}\}\ \texttt{open()}\ \texttt{with}\ (\texttt{c}, \texttt{P}^*)\ \{\ \texttt{opened}(\texttt{c}, \texttt{P}^*, \texttt{res})\}}\ [\textbf{OPEN}]$$

$$\frac{}{\vdash \{\ \texttt{empty}(\tilde{\texttt{c}})\}\ \texttt{close}(\tilde{\texttt{c}})\ \{\ \texttt{true}\ \}}\ [\textbf{CLOSE}]$$

$$\frac{\texttt{inv} \triangleq \texttt{Peer}(\texttt{P}) \wedge \texttt{opened}(\texttt{c}, \texttt{P}^*, \tilde{\texttt{c}}) \wedge \texttt{P} {\in} \texttt{P}^*}{\vdash \{\mathcal{C}(\texttt{c}, \texttt{P}, !\texttt{v} \cdot \texttt{V}(\texttt{v}); \texttt{L}) {*} \texttt{V}(\texttt{x}) {*} \texttt{inv}\}\ \texttt{send}(\tilde{\texttt{c}}, \texttt{x})\ \{\mathcal{C}(\texttt{c}, \texttt{P}, \texttt{L}) {*} \texttt{inv}\}}\ [\textbf{SEND}]$$

$$\frac{\texttt{inv} \triangleq \texttt{Peer}(\texttt{P}) \wedge \texttt{opened}(\texttt{c}, \texttt{P}^*, \tilde{\texttt{c}}) \wedge \texttt{P} {\in} \texttt{P}^*}{\vdash \{\mathcal{C}(\texttt{c}, \texttt{P}, ?\texttt{v} \cdot \texttt{V}(\texttt{v}); \texttt{L}) {*} \texttt{inv}\}\ \texttt{recv}(\tilde{\texttt{c}})\ \{\mathcal{C}(\texttt{c}, \texttt{P}, \texttt{L}) {*} \texttt{V}(\texttt{res}) {*} \texttt{inv}\}}\ [\textbf{RECV}]$$

Implementation

Mercurius - OCaml prototype for reasoning about relaxed and resource-aware protocols:

http://loris-5.d2.comp.nus.edu.sg/Mercurius

Highly modular – using HO predicates and lemmas to describe the communication model.

Case studies: Simple Calculator, Media Cloud Service, Simple Smart Contract (Rock-Paper-Scissor);
- max 8 seconds of verification time per implementation.

More in the paper:

- main results on communication safety: session fidelity and deadlock freedom.
- usage of logical disjunction instead of internal & external choice.
- protocol composition, recursion, communication delegation.

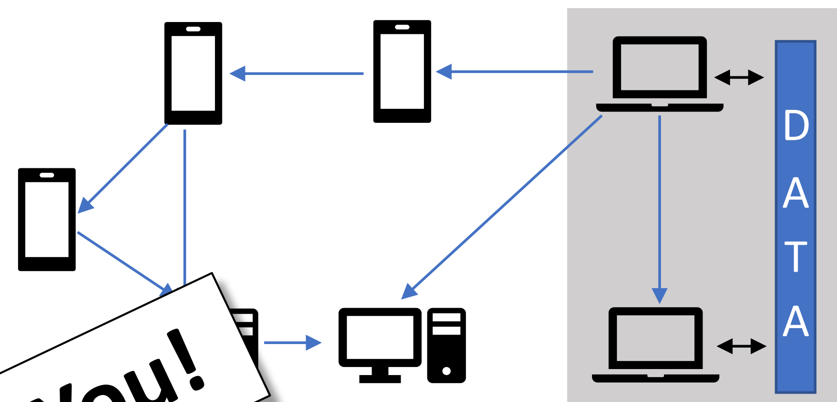
Related Work: MPST, generic types, Disel, IronFleet, Iris, Heap-Hop etc …

On-going work:

- Support for both linear and non-linear protocols.
- Even more fine-grained concurrency.
- Investigating smart-contracts.

We have shown how to support:

- *resource-aware protocols* (with precise verification of distributed programs).



$$G \triangleq (S \rightarrow H_1 : c_1\langle v \cdot \Delta_1\rangle ; H_1 \rightarrow S : c_1\langle v \cdot \Delta_1'\rangle) * (S \rightarrow H_2 : c_2\langle v \cdot \Delta_2\rangle ; \ldots$$

SPECIFY

VERIFY (server's side)

(i)
```
send(c1,fd.vid);
send(c2,fd.aud);
fd.vid = receive(c1);
fd.aud = receive(c2);
```

(ii)
```
send(c1,fd.vid);
fd.vid = receive(c1);
send(c2,fd.aud);
fd.aud = receive(c2);
```

(iii)
```
send(c2,fd.aud);
fd.aud = receive(c2)
send(c1,fd.vid);
fd.vid = receive(c1)
```

(iv)
```
(send(c1,fd.vid); fd.vid = receive(c1);)
                  ||
(send(c2,fd.aud); fd.aud = receive(c2);)
```

MPST

Thank You!

- *relaxed protocols* without sacrificing safety (weaker consistency rules than in the case of MPST).

One ~~ring to rule~~ them all!

protocol to specify

Fence Projection

$$(\xi(\{P^*\}, c, n))\lfloor_P \quad := \begin{cases} \xi(\{P\}, c, n) \text{ if } P \in \{P^*\} \\ \text{emp} \qquad\qquad \text{otherwise} \end{cases}$$

$$(\xi(\{P\}, c_0, n))\lfloor_c \quad := \begin{cases} \oplus\, \xi^{(n)} \text{ if } c = c_0 \\ \ominus\, \xi^{(n)} \text{ if } c \neq c_0 \end{cases}$$

$(G)\lfloor_P$:	$c_1?v \cdot \Delta_1$	;	$\xi(\{P\}, c_1, 1)$	;	$c_2?v \cdot \Delta_2$	;	$\xi(\{P\}, c_2, 2)$	;	$c_2?v \cdot \Delta_3$	;	$\xi(\{P\}, c_2, 3)$	;	$c_1?v \cdot \Delta_4$
$(G)\lfloor_{P,c_1}$:	$?v \cdot \Delta_1$	;	$\oplus\, \xi^{(1)}$	;	emp	;	$\ominus\, \xi^{(2)}$	;	emp	;	$\ominus\, \xi^{(3)}$	;	$?v \cdot \Delta_4$
$(G)\lfloor_{P,c_2}$:	emp	;	$\ominus\, \xi^{(1)}$	;	$?v \cdot \Delta_2$	;	$\oplus\, \xi^{(2)}$	;	$?v \cdot \Delta_3$	;	$\oplus\, \xi^{(3)}$	;	emp