

# On Session Typing, Probabilistic Polynomial Time, and Cryptographic Experiments

Ugo Dal Lago

Giulia Giusti



ALMA MATER STUDIORUM  
UNIVERSITÀ DI BOLOGNA

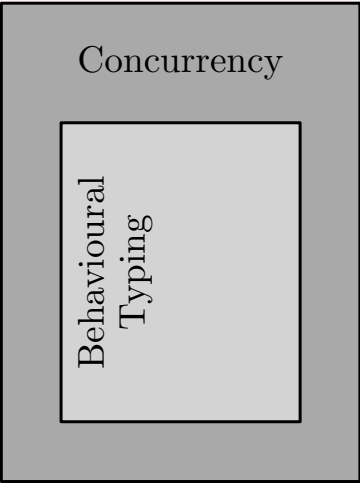


*IFIP WG 2.2, Münster, September 2022*

Part I

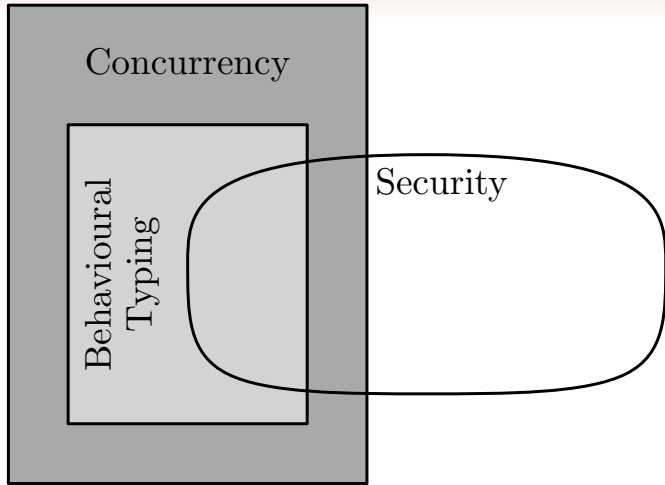
# Cryptography and Concurrency

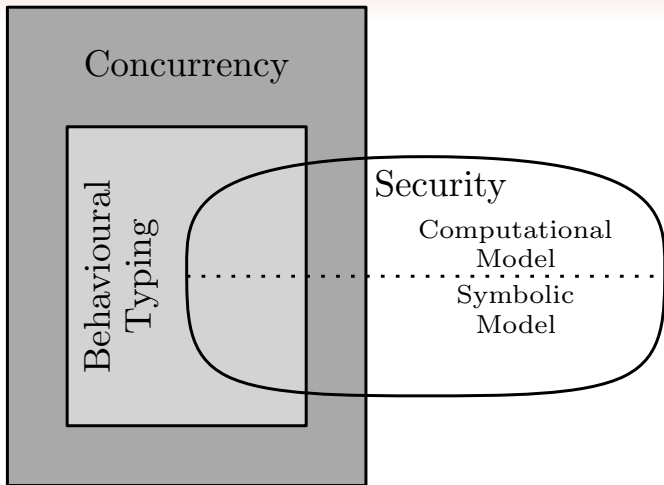
Concurrency

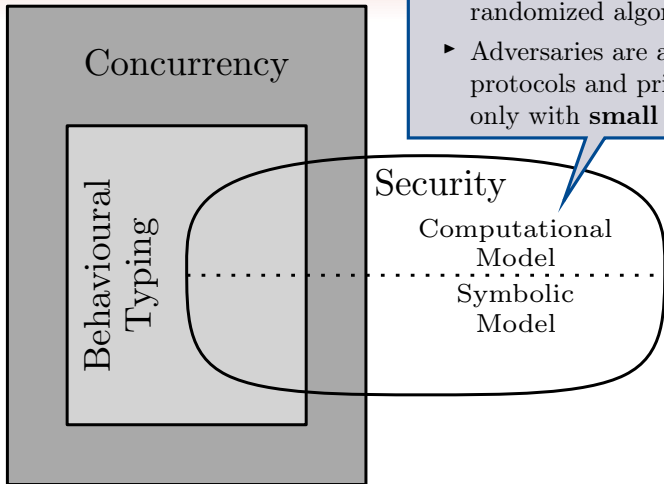


Concurrency

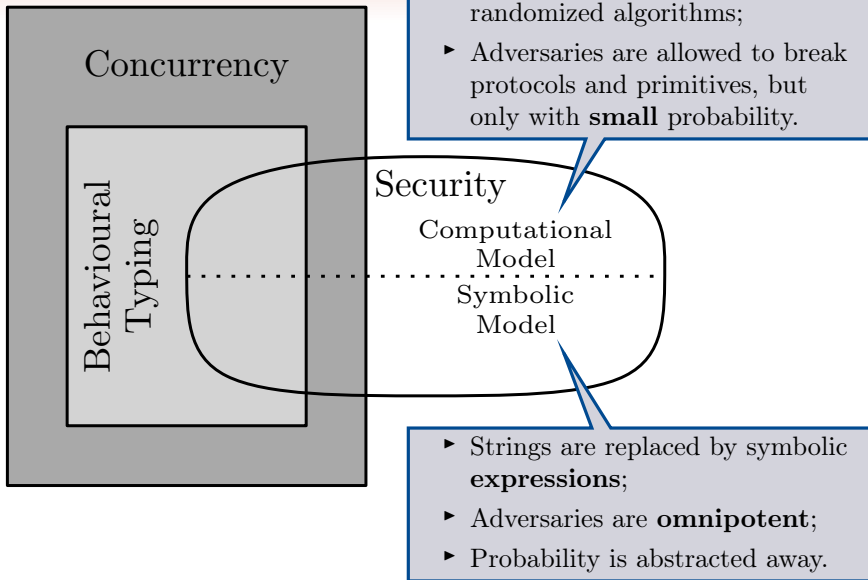
Behavioural  
Typing



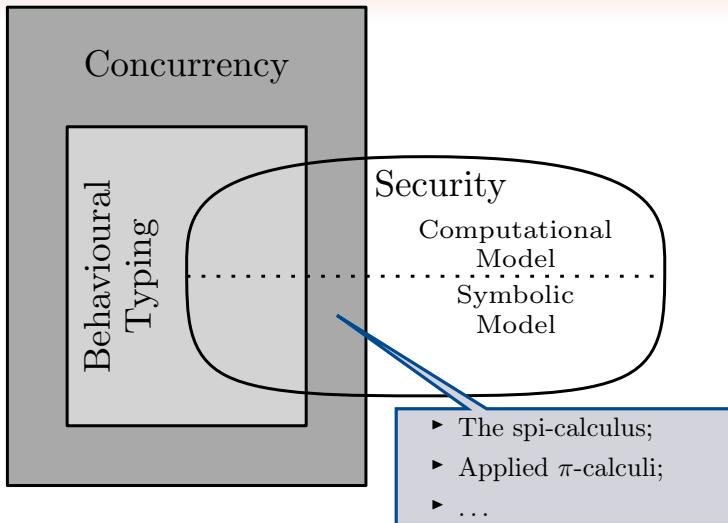


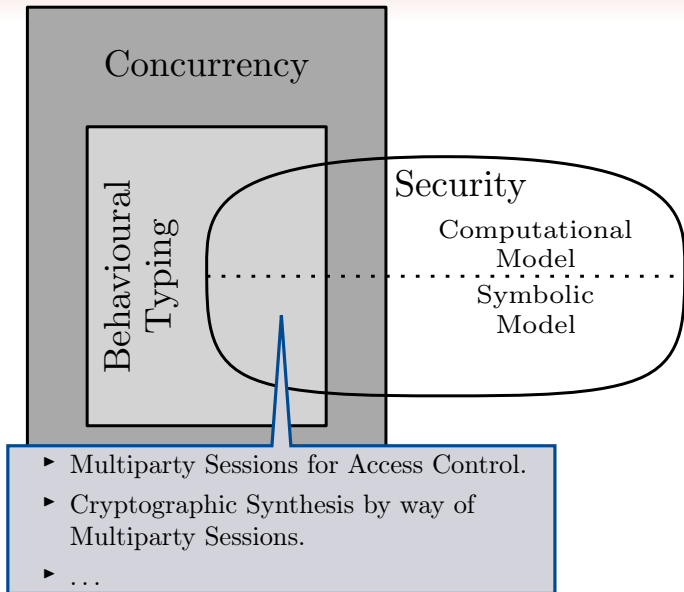


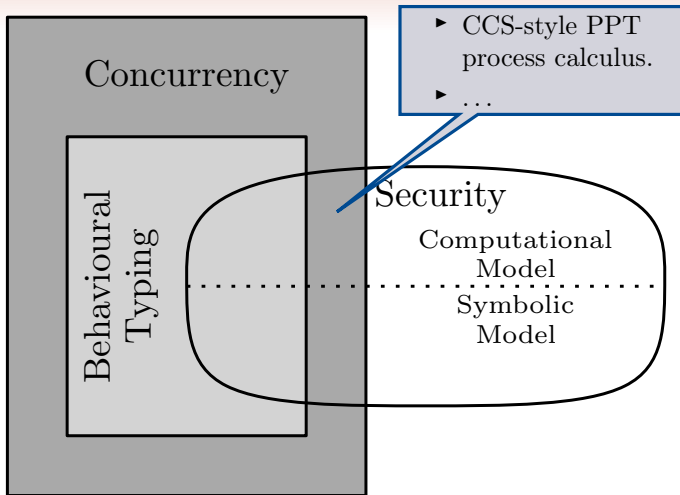
- ▶ Adversaries are **efficient** randomized algorithms;
- ▶ Adversaries are allowed to break protocols and primitives, but only with **small** probability.



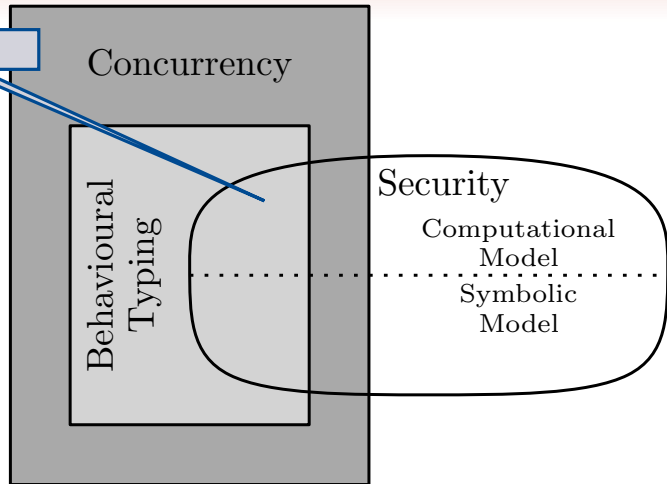








This work!



# Cryptographic Experiments

$\text{PrivK}_{A,\Pi}^{eav}(n) :$

$m_0, m_1 \leftarrow A(1^n)$

$k \leftarrow \text{Gen}(1^n)$

$b \leftarrow \{0, 1\}$

$c \leftarrow \text{Enc}(k, m_b)$

$g \leftarrow A(c)$

**return**  $(b = g)$

# Cryptographic Experiments

Adversary

$\text{PrivK}_{A,\Pi}^{eav}(n) :$

$m_0, m_1 \leftarrow A(1^n)$

$k \leftarrow \text{Gen}(1^n)$

$b \leftarrow \{0, 1\}$

$c \leftarrow \text{Enc}(k, m_b)$

$g \leftarrow A(c)$

**return**  $(b = g)$


# Cryptographic Experiments

$\text{PrivK}_{A,\Pi}^{eav}(n) :$

$m_0, m_1 \leftarrow A(1^n)$

$k \leftarrow \text{Gen}(1^n)$   Key generation

$b \leftarrow \{0, 1\}$

$c \leftarrow \text{Enc}(k, m_b)$   Encryption

$g \leftarrow A(c)$

**return**  $(b = g)$

# Cryptographic Experiments

$\text{PrivK}_{A,\Pi}^{eav}(n) :$

$m_0, m_1 \leftarrow A(1^n)$

$k \leftarrow \text{Gen}(1^n)$

$b \leftarrow \{0, 1\}$

$c \leftarrow \text{Enc}(k, m_b)$

$g \leftarrow A(c)$

**return**  $(b = g)$

$\forall A. \exists \varepsilon. \Pr [\text{PrivK}_{A,\Pi}^{eav}(n) = 1] \leq \frac{1}{2} + \varepsilon(n)$



# Cryptographic Experiments

$\text{PrivK}_{A,\Pi}^{eav}(n) :$

$m_0, m_1 \leftarrow A(1^n)$

$k \leftarrow \text{Gen}(1^n)$

$b \leftarrow \{0, 1\}$

$c \leftarrow \text{Enc}(k, m_b)$

$g \leftarrow A(c)$

**return**  $(b = g)$

$\text{PRIVK}_{\Pi} :$

input  $m_0$  from  $adv$ ;

input  $m_1$  from  $adv$ ;

let  $k = \text{gen}()$  in

let  $b = \text{flipcoin}()$  in

let  $c = \text{enc}(k, m_b)$  in

output  $c$  to  $adv$ ;

input  $g$  from  $adv$ ;

let  $r = \text{eq}(g, b)$  in

output  $r$  to  $exp$ ;

# Cryptographic Experiments

- ▶ An ordinary process term;
- ▶ The adversary is now an external process.

$PRIVK_{\Pi}$  :

$\text{PrivK}_{A,\Pi}^{eav}(n)$  :

$m_0, m_1 \leftarrow A(1^n)$

$k \leftarrow \text{Gen}(1^n)$

$b \leftarrow \{0, 1\}$

$c \leftarrow \text{Enc}(k, m_b)$

$g \leftarrow A(c)$

**return**  $(b = g)$

input  $m_0$  from  $adv$ ;

input  $m_1$  from  $adv$ ;

let  $k = \text{gen}()$  in

let  $b = \text{flipcoin}()$  in

let  $c = \text{enc}(k, m_b)$  in

output  $c$  to  $adv$ ;

input  $g$  from  $adv$ ;

let  $r = \text{eq}(g, b)$  in

output  $r$  to  $exp$ ;

# Security as Equivalence

$$\forall A. \quad (PRIVK_{\Pi_g} \mid A) \sim FLIPCOIN$$

# Security as Equivalence

$$\forall A. \quad (PRIVK_{\Pi_g} \mid A) \sim FLIPCOIN$$

The two involved processes behave  
**approximately the same.**

# Security as Equivalence

$$\Pi_g = (Gen, Enc, Dec)$$

$$Enc(m, k) = m \oplus g(k)$$

$$Dec(c, k) = c \oplus g(k)$$

$$\forall A. \quad (PRIVK_{\Pi_g} \mid A) \sim FLIPCOIN$$

The two involved processes behave  
**approximately the same.**

# Security as Equivalence

$$\forall D. \quad (RAND \mid D) \sim (PRAND_g \mid D)$$

$$\Downarrow$$

$$\forall A. \quad (PRIVK_{\Pi_g} \mid A) \sim FLIPCOIN$$

# Security as Equivalence

Sends a **random**  
string to  $D$ .

Sends a **pseudorandom**  
string obtained through  $g$   
to  $D$ .

$$\forall D. \quad (RAND \mid D) \sim (PRAND_g \mid D)$$

$\Downarrow$

$$\forall A. \quad (PRIVK_{\Pi_g} \mid A) \sim FLIPCOIN$$

**THEOREM 3.18** If  $G$  is a pseudorandom generator, then Construction 3.17 is a fixed-length private-key encryption scheme that has indistinguishable encryptions in the presence of an eavesdropper.

**PROOF** Let  $\Pi$  denote Construction 3.17. We show that  $\Pi$  satisfies Definition 3.8. Namely, we show that for any probabilistic polynomial-time adversary  $\mathcal{A}$  there is a negligible function  $\text{negl}$  such that

$$\Pr[\text{PrivK}_{\mathcal{A},\Pi}^{\text{cpw}}(n) = 1] \leq \frac{1}{2} + \text{negl}(n). \quad (3.2)$$

The intuition is that if  $\Pi$  used a uniform pad in place of the pseudorandom pad  $G(k)$ , then the resulting scheme would be identical to the one-time pad encryption scheme and  $\mathcal{A}$  would be unable to correctly guess which message was encrypted with probability any better than  $1/2$ . Thus, if Equation (3.2) does not hold then  $\mathcal{A}$  must implicitly be distinguishing the output of  $G$  from a random string. We make this explicit by showing a reduction; namely, by showing how to use  $\mathcal{A}$  to construct an efficient distinguisher  $D$ , with the property that  $D$ 's ability to distinguish the output of  $G$  from a uniform string is directly related to  $\mathcal{A}$ 's ability to determine which message was encrypted by  $\Pi$ . Security of  $G$  then implies security of  $\Pi$ .

Let  $\mathcal{A}$  be an arbitrary PPT adversary. We construct a distinguisher  $D$  that takes a string  $w$  as input, and whose goal is to determine whether  $w$  was chosen uniformly (i.e.,  $w$  is a “random string”) or whether  $w$  was generated by choosing a uniform  $k$  and computing  $w := G(k)$  (i.e.,  $w$  is a “pseudorandom string”). We construct  $D$  so that it emulates the eavesdropping experiment for  $\mathcal{A}$ , as described below, and observes whether  $\mathcal{A}$  succeeds or not. If  $\mathcal{A}$  succeeds then  $D$  guesses that  $w$  must be a pseudorandom string, while if  $\mathcal{A}$  does not succeed then  $D$  guesses that  $w$  is a random string. In detail:

**Distinguisher  $D$ :**

$D$  is given as input a string  $w \in \{0,1\}^{\ell(n)}$ . (We assume that  $n$  can be determined from  $\ell(n)$ .)

1. Run  $\mathcal{A}(1^n)$  to obtain a pair of messages  $m_0, m_1 \in \{0,1\}^{\ell(n)}$ .
2. Choose a uniform bit  $b \in \{0,1\}$ . Set  $c := w \oplus m_b$ .
3. Give  $c$  to  $\mathcal{A}$  and obtain output  $b'$ . Output 1 if  $b' = b$ , and output 0 otherwise.

$D$  clearly runs in polynomial time (assuming  $\mathcal{A}$  does).

Before analyzing the behavior of  $D$ , we define a modified encryption scheme  $\tilde{\Pi} = (\text{Gen}, \text{Enc}, \text{Dec})$  that is exactly the one-time pad encryption scheme, except that we now incorporate a security parameter that determines the length of the message to be encrypted. That is,  $\text{Gen}(1^n)$  outputs a uniform key  $k$  of length  $\ell(n)$ , and the encryption of message  $m \in 2^{\ell(n)}$  using key  $k \in \{0,1\}^{\ell(n)}$

is the ciphertext  $e := k \oplus m$ . (Decryption can be performed as usual, but is inessential to what follows.) Perfect secrecy of the one-time pad implies

$$\Pr[\text{PrivK}_{\mathcal{A},\Pi}^{\text{cpw}}(n) = 1] = \frac{1}{2}. \quad (3.3)$$

To analyze the behavior of  $D$ , the main observations are:

1. If  $w$  is chosen uniformly from  $\{0,1\}^{\ell(n)}$ , then the view of  $\mathcal{A}$  when run as a subroutine by  $D$  is distributed identically to the view of  $\mathcal{A}$  in experiment  $\text{PrivK}_{\mathcal{A},\Pi}^{\text{cpw}}(n)$ . This is because when  $\mathcal{A}$  is run as a subroutine by  $D(w)$  in this case,  $\mathcal{A}$  is given a ciphertext  $c = w \oplus m_b$  where  $w \in \{0,1\}^{\ell(n)}$  is uniform. Since  $D$  outputs 1 exactly when  $\mathcal{A}$  succeeds in its eavesdropping experiment, we therefore have (cf. Equation (3.3))

$$\Pr_{w \leftarrow \{0,1\}^{\ell(n)}}[D(w) = 1] = \Pr[\text{PrivK}_{\mathcal{A},\Pi}^{\text{cpw}}(n) = 1] = \frac{1}{2}. \quad (3.4)$$

(The subscript on the first probability just makes explicit that  $w$  is chosen uniformly from  $\{0,1\}^{\ell(n)}$  there.)

2. If  $w$  is instead generated by choosing uniform  $k \in \{0,1\}^n$  and then setting  $w := G(k)$ , the view of  $\mathcal{A}$  when run as a subroutine by  $D$  is distributed identically to the view of  $\mathcal{A}$  in experiment  $\text{PrivK}_{\mathcal{A},\Pi}^{\text{cpw}}(n)$ . This is because  $\mathcal{A}$ , when run as a subroutine by  $D$ , is now given a ciphertext  $c = w \oplus m_b$  where  $w = G(k)$  for a uniform  $k \in \{0,1\}^n$ . Thus,

$$\Pr_{k \leftarrow \{0,1\}^n}[D(G(k)) = 1] = \Pr[\text{PrivK}_{\mathcal{A},\Pi}^{\text{cpw}}(n) = 1]. \quad (3.5)$$

Since  $G$  is a pseudorandom generator (and since  $D$  runs in polynomial time), we know there is a negligible function  $\text{negl}$  such that

$$\left| \Pr_{w \leftarrow \{0,1\}^{\ell(n)}}[D(w) = 1] - \Pr_{k \leftarrow \{0,1\}^n}[D(G(k)) = 1] \right| \leq \text{negl}(n).$$

Using Equations (3.4) and (3.5), we thus see that

$$\left| \frac{1}{2} - \Pr[\text{PrivK}_{\mathcal{A},\Pi}^{\text{cpw}}(n) = 1] \right| \leq \text{negl}(n),$$

which implies  $\Pr[\text{PrivK}_{\mathcal{A},\Pi}^{\text{cpw}}(n) = 1] \leq \frac{1}{2} + \text{negl}(n)$ . Since  $\mathcal{A}$  was an arbitrary PPT adversary, this completes the proof that  $\Pi$  has indistinguishable encryptions in the presence of an eavesdropper. ■

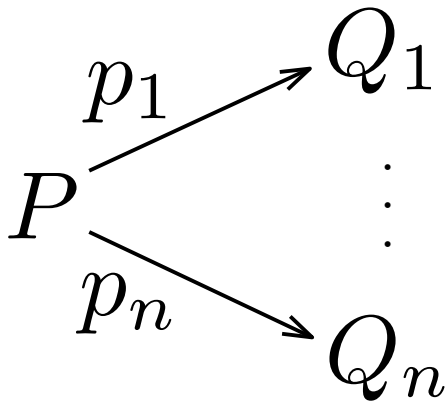
It is easy to get lost in the details of the proof and wonder whether anything has been gained as compared to the one-time pad; after all, the one-time pad also encrypts an  $\ell$ -bit message by XORing it with an  $\ell$ -bit string! The point of the construction, of course, is that the  $\ell$ -bit string  $G(k)$  can be much



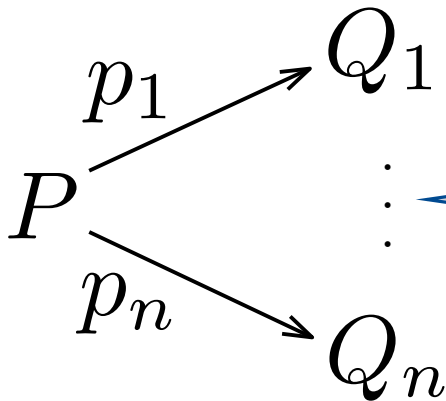
## Part II

# The Three Challenges

# Randomized Evolution



# Randomized Evolution



- The  $Q_i$ s can be very different as for the values they produce
- Their behaviors have the same structure.

# Approximate Equivalence

$$\Pr[\text{Obs}(P) = \text{Obs}(Q)] \geq 1 - \varepsilon(n)$$

where  $\varepsilon$  is a negligible function.

$$P \sim Q$$

- ▶ Process definitions should be parameterized on  $n$ , the **security parameter**.
- ▶ Roughly speaking,  $n$  is the **length of keys**.

# Polynomial Time Bounds

- ▶ For every  $P$  there must be a polynomial  $q$  such that  $m \leq q(k)$  for every  $k$ .
- ▶ Otherwise, e.g., any modern encryption scheme would be insecure.

$$P[n \mapsto k] := P_1 \mapsto P_2 \mapsto \cdots \mapsto P_m$$

## Part III

# System $\pi$ **DIBLL**

## Linear logic propositions as session types

LUÍS CAIRES<sup>†</sup>, FRANK PFENNING<sup>‡</sup> and BERNARDO TONINHO<sup>†‡</sup>

<sup>†</sup>*Faculdade de Ciências e Tecnologia and CITI, Universidade Nova de Lisboa, Lisboa, Portugal*  
Emails: btoninho@gmail.com and luis.caires@fct.unl.pt

<sup>‡</sup>*Computer Science Department, Carnegie Mellon University, Pittsburgh, PA, USA*  
Email: fp@cs.cmu.edu

*Received 14 November 2012; revised 5 March 2013*

Throughout the years, several typing disciplines for the  $\pi$ -calculus have been proposed. Arguably, the most widespread of these typing disciplines consists of session types. Session types describe the input/output behaviour of processes and traditionally provide strong guarantees about this behaviour (i.e. deadlock-freedom and fidelity). While these systems exploit a fundamental notion of linearity, the precise connection between linear logic and session types has not been well understood.

This paper proposes a type system for the  $\pi$ -calculus that corresponds to a standard sequent calculus presentation of intuitionistic linear logic, interpreting linear propositions as session types and thus providing a purely logical account of all key features and properties of session types. We show the deep correspondence between linear logic and session types by exhibiting a tight operational correspondence between cut-elimination steps and process reductions. We

## Fundamental Study

---

# Bounded linear logic: a modular approach to polynomial-time computability

Jean-Yves Girard

*Équipe de logique, UA 753 du CNRS Mathématiques, tour 45–55, Université Paris 7, 2 Place  
Jussieu, 75251 Paris Cedex 05, France*

Andre Scedrov

*Department of Mathematics, University of Pennsylvania, Philadelphia, PA 19104-6395, USA*

BLL



# Processes

These are exactly the process expressions of  $\pi\mathbf{DILL}$ .

$$P, Q ::= 0 \mid P \mid Q \mid (\nu y) P \mid x\langle y \rangle.P \mid x(y).P \mid \\ !x(y).P \mid x.\mathbf{inl}; P \mid x.\mathbf{inr}; P \mid x.\mathbf{case}(P, Q)$$

# Processes

$$\begin{aligned} P, Q ::= & 0 \mid P \mid Q \mid (\nu y) P \mid x\langle y \rangle.P \mid x(y).P \mid \\ & !x(y).P \mid x.\text{inl}; P \mid x.\text{inr}; P \mid x.\text{case}(P, Q) \mid \\ & [x \leftarrow v] \mid x.P \mid \text{let } x = a \text{ in } P \mid \\ & \text{if } v \text{ then } P \text{ else } Q \end{aligned}$$

# Processes

$$\begin{aligned} P, Q ::= & 0 \mid P \mid Q \mid (\nu y) P \mid x\langle y \rangle.P \mid x(y).P \mid \\ & !x(y).P \mid x.\text{inl}; P \mid x.\text{inr}; P \mid x.\text{case}(P, Q) \mid \\ & [x \leftarrow v] \mid x.P \mid \text{let } x = a \text{ in } P \mid \\ & \text{if } v \text{ then } P \text{ else } Q \end{aligned}$$

- ▶ The term  $a$  is of course **not** a process!
- ▶ It's built from first-order function symbols computing random functions and which can be evaluated in probabilistic polynomial time in  $n$ .

# Processes

$$P, Q ::= 0 \mid P \mid Q \mid (\nu y) P \mid x\langle y \rangle.P \mid x(y).P \mid \\ !x(y).P \mid x.\text{inl}; P \mid x.\text{inr}; P \mid x.\text{case}(P, Q) \mid \\ [x \leftarrow v] \mid x.P \mid \text{let } x = a \text{ in } P \mid \\ \text{if } v \text{ then } P \text{ else } Q$$

Input of a  
ground value.

Output of a  
ground value.

# Types

These are (almost) exactly the  
types of  $\pi\mathbf{DILL}$

$A, B ::= 1 \mid A \multimap B \mid !_p A \mid A \otimes B \mid A \oplus B \mid A \& B \mid$   
 $\mathbb{B} \mid \mathbb{S}[p]$

Booleans

Strings of length  $p(n)$

# Typing Judgments

Term variables

Polynomial variables

$$\Gamma; \Delta; \Theta \vdash^{\mathcal{V}} P :: x : A$$

Restricted channels

Unrestricted channels, each associated with a type and a polynomial.

# Some Typing Rules

$$\frac{\Gamma, u_p : A; \Delta, y : A; \Theta \vdash^\nu P :: T}{\Gamma, u_{p+1} : A; \Delta; \Theta \vdash^\nu (\nu y) u\langle y \rangle.P :: T} [T_{copy}]$$

# Some Typing Rules

Unrestricted environments can be aggregated, **summing** up polynomials.

$$\frac{\Gamma_1 \boxplus \Gamma_2 \sqsubseteq \Gamma \quad \begin{array}{l} \Gamma_1; \Delta_1; \Theta \vdash^\nu P :: x : A \\ \Gamma_2; \Delta_2, x : A; \Theta \vdash^\nu Q :: T \end{array}}{\Gamma; \Delta_1, \Delta_2; \Theta \vdash^\nu (\nu x) (P \mid Q) :: T} [T_{cut}]$$



# Some Typing Rules

Terms have to be typable themselves.

$$\frac{\Theta \vdash^{\nu} a : A \quad \Gamma; \Delta; \Theta, x : A \vdash^{\nu} P :: T}{\Gamma; \Delta; \Theta \vdash^{\nu} \text{let } x = a \text{ in } P :: T} [T_{\text{let}}]$$

# Reduction Semantics

$$P \rightarrow \mathcal{D}$$

# Reduction Semantics

$$P \rightarrow \mathcal{D}$$

Processes rewrite to  
**distributions** of processes.

# Reduction Semantics

$$P \rightarrow \mathcal{D}$$

Processes rewrite to  
**distributions** of processes.

$$P \mapsto Q$$

# Reduction Semantics

$$P \rightarrow \mathcal{D}$$

Processes rewrite to  
**distributions** of processes.

$$P \mapsto Q$$

$P \rightarrow \mathcal{D}$  and  
 $Q \in \text{SUPPORT}(\mathcal{D})$ .

# Reduction Semantics

$$P \rightarrow \mathcal{D}$$

Processes rewrite to  
**distributions** of processes.

$$P \mapsto Q$$

$P \rightarrow \mathcal{D}$  and  
 $Q \in \text{SUPPORT}(\mathcal{D})$ .

$$\mathcal{D} \Rightarrow \mathcal{E}$$

# Reduction Semantics

$$P \rightarrow \mathcal{D}$$

Processes rewrite to  
**distributions** of processes.

$$P \mapsto Q$$

$P \rightarrow \mathcal{D}$  and  
 $Q \in \text{SUPPORT}(\mathcal{D})$ .

$$\mathcal{D} \Rightarrow \mathcal{E}$$

The Kleisli Lifting of  $\rightarrow$ .

## Part IV

# Main Technical Results



# Subject Reduction

## Theorem

If  $\Gamma; \Delta \vdash P :: z : C$  and  $P \mapsto R$ , then it holds that  
 $\Gamma; \Delta \vdash R :: z : C$ .

# Subject Reduction

Randomization plays **no role**.

Theorem

If  $\Gamma; \Delta \vdash P :: z : C$  and  $P \mapsto R$ , then it holds that  
 $\Gamma; \Delta \vdash R :: z : C$ .

# Polytime Soundness

## Theorem

For every derivation  $\pi$  typing  $P$ , there is a polynomial  $p_\pi$  such that for every substitution  $\rho$ , if  $P\rho \mapsto^* Q$ , the overall computational cost of the aforementioned reduction is bounded by  $p_\pi(\rho)$ .

# Polytime Soundness

The **weight** of  $\pi$ , defined following the structure of  $P$ .

## Theorem

For every derivation  $\pi$  typing  $P$ , there is a polynomial  $p_\pi$  such that for every substitution  $\rho$ , if  $P\rho \mapsto^* Q$ , the overall computational cost of the aforementioned reduction is bounded by  $p_\pi(\rho)$ .

# Confluence

## Theorem

If  $\Gamma; \Delta \vdash P :: T$  and  $\mathcal{D} \leftarrow P \rightarrow \mathcal{E}$  then either  $\mathcal{D} = \mathcal{E}$  or there exists  $\mathcal{F}$  such that  $\mathcal{D} \Rightarrow \mathcal{F} \Leftarrow \mathcal{E}$ .

Part V

# Back To Cryptography

# Observational Equivalence

## A Notion of Observation

Given a process  $\vdash P :: x : \mathbb{B}$ , the fact that  $P$  outputs  $b$ , written  $P \downarrow_x b$  **does not depend** on any scheduler, thanks to confluence.

# Observational Equivalence

## A Notion of Observation

Given a process  $\vdash P :: x : \mathbb{B}$ , the fact that  $P$  outputs  $b$ , written  $P \downarrow_x b$  **does not depend** on any scheduler, thanks to confluence.

## The Definition

Two processes  $P, Q$  are **observationally equivalent** iff for every closing context  $\mathcal{C}$  there is  $\varepsilon$  negligible such that

$$|\Pr[\mathcal{C}[P]\rho \downarrow_x b] - \Pr[\mathcal{C}[Q]\rho \downarrow_x b]| \leq \varepsilon(\rho)$$



# Observational Equivalence

## A Notion of Contexts

Given a process  $\vdash P :: x : \mathbb{B}$ , the fact that  $P$  **not depend** on any scheduler  $\sigma$  means that  $P$  **does not depend** on any scheduler  $\sigma$ .

A process  $\mathcal{C}$  in which the hole  $[\cdot]$  occurs only once and such that  $\vdash \mathcal{C}[P] :: x : \mathbb{B}$  is said to be a **closing context** for  $P$ .

## The Definition

Two processes  $P, Q$  are **observationally equivalent** iff for every closing context  $\mathcal{C}$  there is  $\varepsilon$  negligible such that

$$|\Pr[\mathcal{C}[P]\rho \downarrow_x b] - \Pr[\mathcal{C}[Q]\rho \downarrow_x b]| \leq \varepsilon(\rho)$$

# The Proof of Security for $\Pi_g$

$$\forall D. \nu out. (PRAND_g \mid D) \sim \nu out. (RAND \mid D)$$

$\Downarrow$

$$\forall A. \nu adv. (PRIVK_{\Pi_g} \mid A) \sim FAIRFLIP$$

# The Proof of Security for $\Pi_g$

$A$

$\downarrow$

$$\nu_{out.}(PRAND_g \mid D_A) \sim \nu_{adv.}(PRIVK_{\Pi_g} \mid A)$$

$$\nu_{out.}(RAND \mid D_A) \sim FAIRFLIP$$

# The Proof of Security for $\Pi_g$

It is built out of  
 $\nu adv.(PRIVK_{\Pi_g} \mid A)$ , by isolating  
the role played by  $g$ .

$A$



$$\nu out.(PRAND_g \mid D_A) \sim \nu adv.(PRIVK_{\Pi_g} \mid A)$$

$$\nu out.(RAND \mid D_A) \sim FAIRFLIP$$

# The Proof of Security for $\Pi_g$

It is built out of  
 $\nu adv.(PRIVK_{\Pi_g} \mid A)$ , by isolating  
the role played by  $g$ .

$A$



Follows from simple equations,  
proved sound for observational  
equivalence.

$$\begin{aligned} \nu out.(PRAND_g \mid D_A) &\sim \nu adv.(PRIVK_{\Pi_g} \mid A) \\ \nu out.(RAND \mid D_A) &\sim FAIRFLIP \end{aligned}$$

# The Proof of Security for $\Pi_g$

It is built out of  $\nu adv.(PRIVK_{\Pi_g} \mid A)$ , by isolating the role played by  $g$ .

$A$



Follows from simple equations, proved sound for observational equivalence.

$$\nu out.(PRAND_g \mid D_A) \sim \nu adv.(PRIVK_{\Pi_g} \mid A)$$

$$\nu out.(RAND \mid D_A) \sim FAIRFLIP$$

Follows easily from the security of the so-called **one-time pad**

# Wrapping Up

## Contributions

- ▶  $\pi$ **DIBLL**, a type system for polynomial time randomized processes.
- ▶ Session types can indeed **faithfully** model cryptographic experiments and proofs.

# Wrapping Up

## Contributions

- ▶  $\pi$ **DIBLL**, a type system for polynomial time randomized processes.
- ▶ Session types can indeed **faithfully** model cryptographic experiments and proofs.

## Future Work

- ▶ Extending the language of processes with a construct for **iteration**.
- ▶ Towards Canetti's **Universal Composability**.



# Wrapping Up

## Contributions

- ▶  $\pi$ **DIBLL**, a type system for polynomial time randomized processes.
- ▶ Session types can indeed **faithfully** model cryptographic experiments and proofs.

## Future Work

- ▶ Extending the language of processes with a construct for **iteration**.
- ▶ Towards Canetti's **Universal Composability**.

Thank you! Questions?

# Typing Rules for Processes - Standard Operators I

$$\frac{\Gamma; \Delta; \Theta \vdash^{\nu} P :: T}{\Gamma; \Delta, x:1; \Theta \vdash^{\nu} P :: T} [T1L] \quad \frac{}{\Gamma; \cdot; \Theta \vdash^{\nu} 0 :: x:1} [T1R]$$

$$\frac{\Gamma; \Delta, y:A, x:B; \Theta \vdash^{\nu} P :: T}{\Gamma; \Delta, x:A \otimes B; \Theta \vdash^{\nu} x(y).P :: T} [T \otimes L]$$

$$\frac{\Gamma_1 \boxplus \Gamma_2 \sqsubseteq \Gamma \quad \Gamma_1; \Delta; \Theta \vdash^{\nu} P :: y:A \quad \Gamma_2; \Delta'; \Theta \vdash^{\nu} Q :: x:B}{\Gamma; \Delta, \Delta'; \Theta \vdash^{\nu} (\nu y) x\langle y \rangle.(P \mid Q) :: x:A \otimes B} [T \otimes R]$$

$$\frac{\Gamma_1 \boxplus \Gamma_2 \sqsubseteq \Gamma \quad \Gamma_1; \Delta; \Theta \vdash^{\nu} P :: y:A \quad \Gamma_2; \Delta', x:B; \Theta \vdash^{\nu} Q :: T}{\Gamma; \Delta, \Delta', x:A \multimap B; \Theta \vdash^{\nu} (\nu y) x\langle y \rangle.(P \mid Q) :: T} [T \multimap L]$$

$$\frac{\Gamma; \Delta, y:A; \Theta \vdash^{\nu} P :: x:B}{\Gamma; \Delta; \Theta \vdash^{\nu} x(y).P :: x:A \multimap B} [T \multimap R]$$

$$\frac{\Gamma_1 \boxplus \Gamma_2 \sqsubseteq \Gamma \quad \Gamma_1; \Delta; \Theta \vdash^{\nu} P :: x:A \quad \Gamma_2; \Delta', x:A; \Theta \vdash^{\nu} Q :: T}{\Gamma; \Delta, \Delta'; \Theta \vdash^{\nu} (\nu x) (P \mid Q) :: T} [T_{cut}]$$

# Typing Rules for Processes - Standard Operators II

$$\frac{p * \Gamma_1 \boxplus \Gamma_2 \sqsubseteq \Gamma \quad \Gamma_1; \cdot; \Theta \vdash^\nu P :: y : A \quad \Gamma_2, u_p : A; \Delta; \Theta \vdash^\nu Q :: T}{\Gamma; \Delta; \Theta \vdash^\nu (\nu u) (!u(y).P \mid Q) :: T} [T_{cut!}]$$

$$\frac{\Gamma, u_p : A; \Delta, y : A; \Theta \vdash^\nu P :: T}{\Gamma, u_{p+1} : A; \Delta; \Theta \vdash^\nu (\nu y) u\langle y \rangle.P :: T} [T_{copy}]$$

$$\frac{\Gamma, u_p : A; \Delta; \Theta \vdash^\nu P :: T}{\Gamma; \Delta, x : !_p A; \Theta \vdash^\nu P\{x/u\} :: T} [T!_p L]$$

$$\frac{\Gamma; \cdot; \Theta \vdash^\nu Q :: y : A}{p * \Gamma; \cdot; \Theta \vdash^\nu !x(y).Q :: x : !_p A} [T!_p R]$$

$$\frac{\Gamma; \Delta, x : A; \Theta \vdash^\nu P :: T \quad \Gamma; \Delta, x : B; \Theta \vdash^\nu Q :: T}{\Gamma; \Delta, x : A \oplus B; \Theta \vdash^\nu x.\text{case}(P, Q) :: T} [T \oplus L]$$

$$\frac{\Gamma; \Delta; \Theta \vdash^\nu P :: x : A}{\Gamma; \Delta; \Theta \vdash^\nu x.\text{inl}; P :: x : A \oplus B} [T \oplus R_1]$$

# Typing Rules for Processes - Standard Operators III

$$\frac{\Gamma; \Delta; \Theta \vdash^{\nu} P :: x : B}{\Gamma; \Delta; \Theta \vdash^{\nu} x.\text{inr}; P :: x : A \oplus B} [T \oplus R_2]$$

$$\frac{\Gamma; \Delta, x : A; \Theta \vdash^{\nu} P :: T}{\Gamma; \Delta, x : A \& B; \Theta \vdash^{\nu} x.\text{inl}; P :: T} [T \& L_1]$$

$$\frac{\Gamma; \Delta, x : B; \Theta \vdash^{\nu} P :: T}{\Gamma; \Delta, x : A \& B; \Theta \vdash^{\nu} x.\text{inr}; P :: T} [T \& L_2]$$

$$\frac{\Gamma; \Delta; \Theta \vdash^{\nu} P :: x : A \quad \Gamma; \Delta; \Theta \vdash^{\nu} Q :: x : B}{\Gamma; \Delta; \Theta \vdash^{\nu} x.\text{case}(P, Q) :: x : A \& B} [T \& R]$$

# Typing Rules for Processes - New Operators

$$\frac{\Gamma; \Delta; \Theta, x : \mathbb{S}[p] \vdash^{\nu} Q :: T}{\Gamma; \Delta, x : \mathbb{S}[p]; \Theta \vdash^{\nu} x.Q :: T} [TSL]$$

$$\frac{\Theta \vdash^{\nu} v : \mathbb{S}[p]}{\Gamma; \Delta; \Theta \vdash^{\nu} [x \leftarrow v] :: x : \mathbb{S}[p]} [TSR]$$

$$\frac{\Gamma; \Delta; \Theta, x : \mathbb{B} \vdash^{\nu} Q :: T}{\Gamma; \Delta, x : \mathbb{B}; \Theta \vdash^{\nu} x.Q :: T} [TBL]$$

$$\frac{\Theta \vdash^{\nu} v : \mathbb{B}}{\Gamma; \Delta; \Theta \vdash^{\nu} [x \leftarrow v] :: x : \mathbb{B}} [TBR]$$

$$\frac{\Theta \vdash^{\nu} a : B \quad \Gamma; \Delta; \Theta, x : B \vdash^{\nu} P :: T}{\Gamma; \Delta; \Theta \vdash^{\nu} \text{let } x = a \text{ in } P :: T} [T_{let}]$$

$$\frac{\Theta \vdash^{\nu} v : \mathbb{B} \quad \Gamma; \Delta; \Theta \vdash^{\nu} P :: x : A \quad \Gamma; \Delta; \Theta \vdash^{\nu} Q :: x : A}{\Gamma; \Delta; \Theta \vdash^{\nu} \text{if } v \text{ then } P \text{ else } Q :: x : A} [T_{if}]$$

# Typing Rules for Terms

$$\begin{array}{c} \frac{\text{vars}(B), \text{vars}(\Theta) \subseteq \mathcal{V}}{\Theta, z : B \vdash^{\mathcal{V}} z : B} [Var] \qquad \frac{|s| \leq p \quad \text{vars}(\Theta), \text{vars}(p) \subseteq \mathcal{V}}{\Theta \vdash^{\mathcal{V}} s : \mathbb{S}[p]} [String] \\[10pt] \frac{\text{vars}(\Theta) \subseteq \mathcal{V}}{\Theta \vdash^{\mathcal{V}} \text{true} : \mathbb{B}} [Bool1] \qquad \frac{\text{vars}(\Theta) \subseteq \mathcal{V}}{\Theta \vdash^{\mathcal{V}} \text{false} : \mathbb{B}} [Bool2] \\[10pt] \frac{\text{typeof}(f) = B_1, \dots, B_m \rightarrow C \quad \Theta \vdash^{\mathcal{V}} v_i : B_i \{n \leftarrow p\} \quad \text{vars}(p) \subseteq \mathcal{V}}{\Theta \vdash^{\mathcal{V}} f_p(v_1, \dots, v_m) : C \{n \leftarrow p\}} [Fun] \end{array}$$

# Reduction Rules

$$\frac{}{x\langle y \rangle.Q \mid x(u).P \rightarrow \{Q \mid P\{y/u\}^1\}}$$

$$\frac{}{x\langle y \rangle.Q \mid !x(u).P \rightarrow \{Q \mid P\{y/u\} \mid !x(u).P^1\}}$$

$$\frac{}{x.\text{inl}; P \mid x.\text{case}(Q, R) \rightarrow \{P \mid Q^1\}}$$

$$\frac{}{x.\text{inr}; P \mid x.\text{case}(Q, R) \rightarrow \{P \mid R^1\}}$$

$$\frac{P \rightarrow \{Q_i^{r_i}\}_{i \in I}}{R \mid P \rightarrow \{R \mid Q_i^{r_i}\}_{i \in I}}$$

$$\frac{P \rightarrow \{Q_i^{r_i}\}_{i \in I}}{(\nu y) P \rightarrow \{(\nu y) Q_i^{r_i}\}_{i \in I}}$$

$$\frac{P \equiv R \quad R \rightarrow \{Q_i^{r_i}\}_{i \in I} \quad Q_i \equiv T_i \text{ for every } i \in I}{P \rightarrow \{T_i^{r_i}\}_{i \in I}}$$

$$\frac{}{[x \leftarrow v] \mid x.Q \rightarrow \{Q\{v/x\}^1\}}$$

$$\frac{a \hookrightarrow \{v_i^{r_i}\}_{i \in I}}{\text{let } x = a \text{ in } P \rightarrow \{P\{v_i/x\}^{r_i}\}_{i \in I}}$$

$$\frac{}{\text{if true then } P \text{ else } Q \rightarrow \{P^1\}}$$

$$\frac{}{\text{if false then } P \text{ else } Q \rightarrow \{Q^1\}}$$