

# **Interactive and Probabilistic Proof of Mobile Code Safety**

**Yasuyuki Tsukada**  
**NTT Communication Science Laboratories**  
**NTT Corporation**

**September 26, 2001**

**PCP PCC**

**PCP**

Probabilistically Checkable Proof

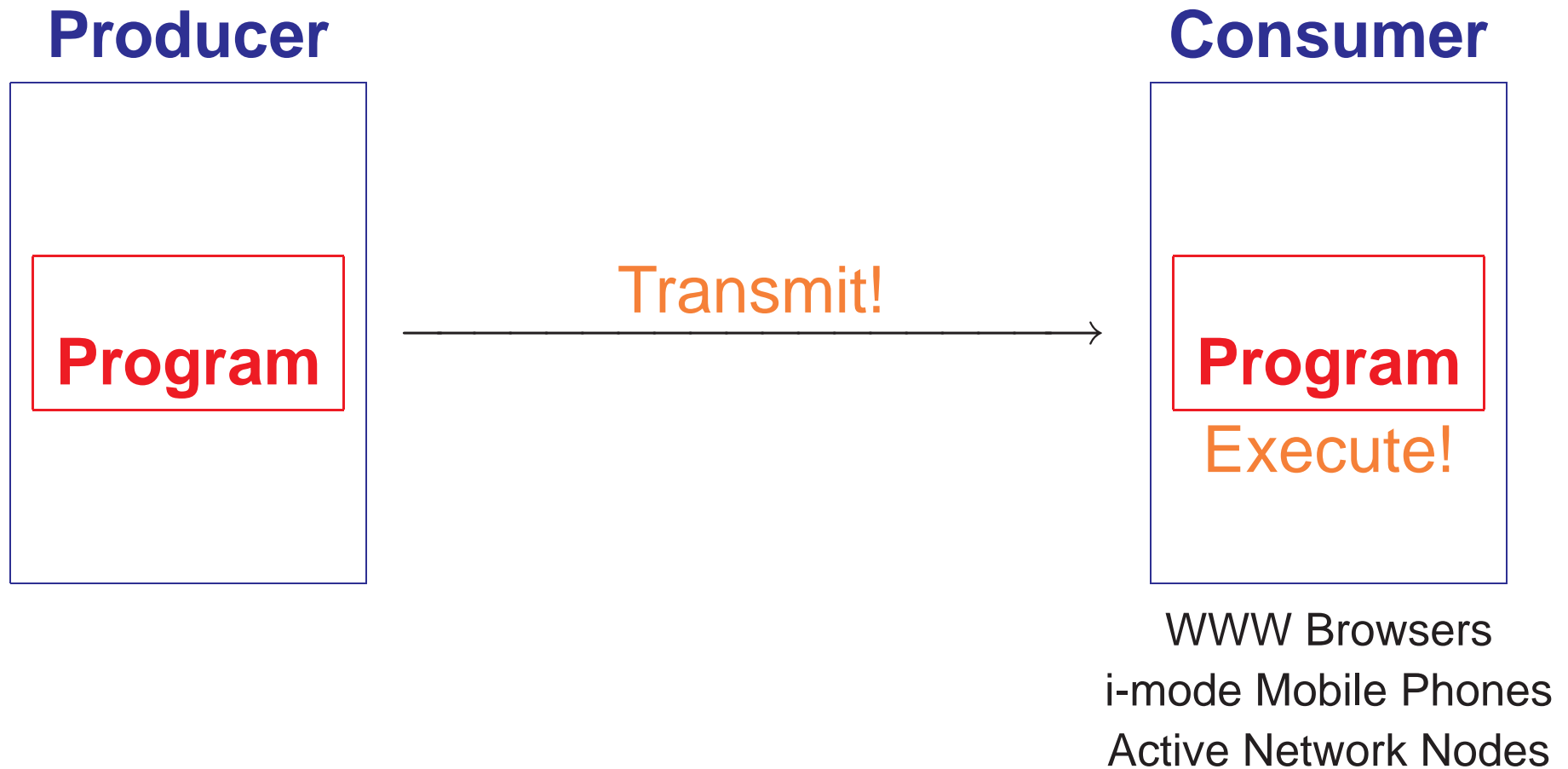
“Volume A”

**PCC**

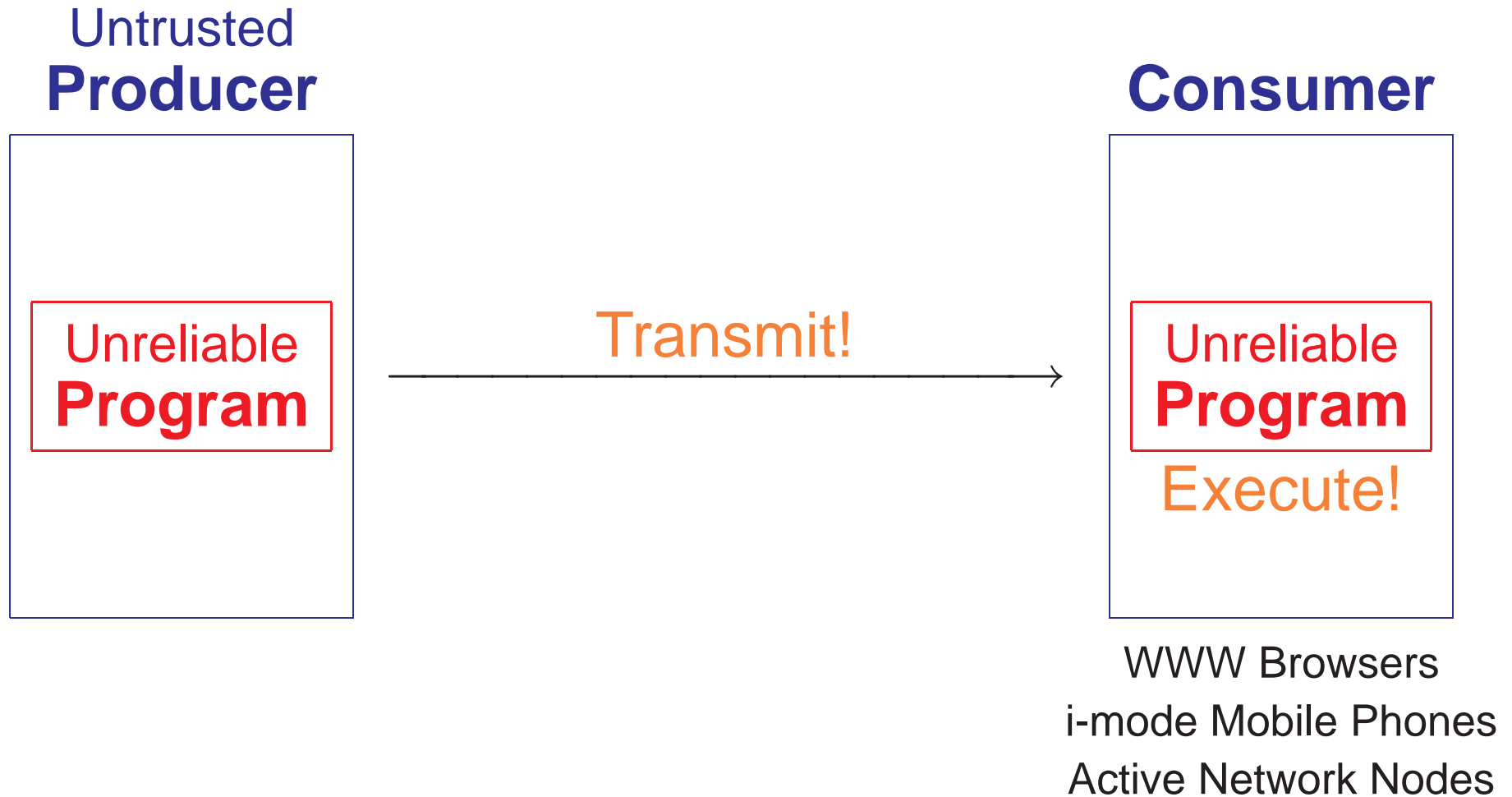
Proof-Carrying Code

“Volume B”

# Extension of Distributed Software Systems

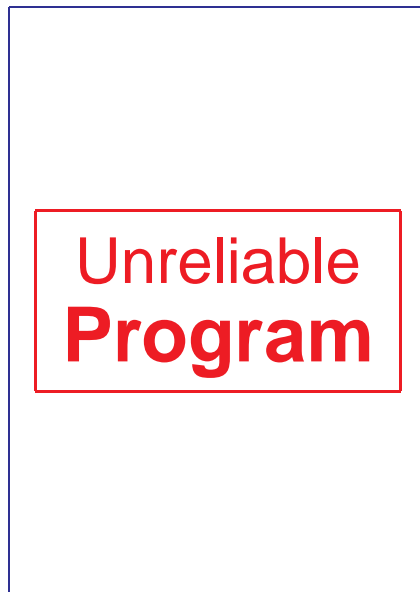


# Extension of Distributed Software Systems



# Safe Extension of Distributed Software Systems

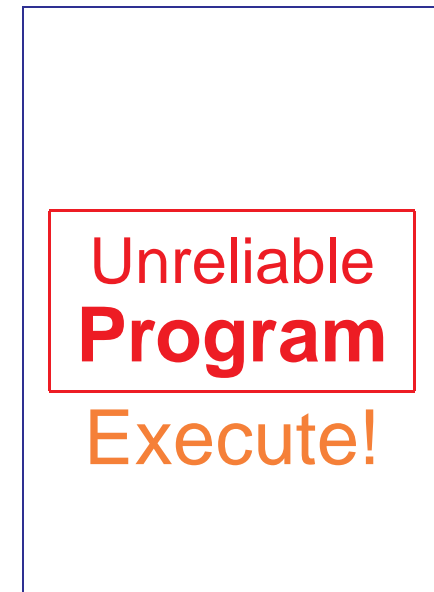
Untrusted  
**Producer**



Transmit!

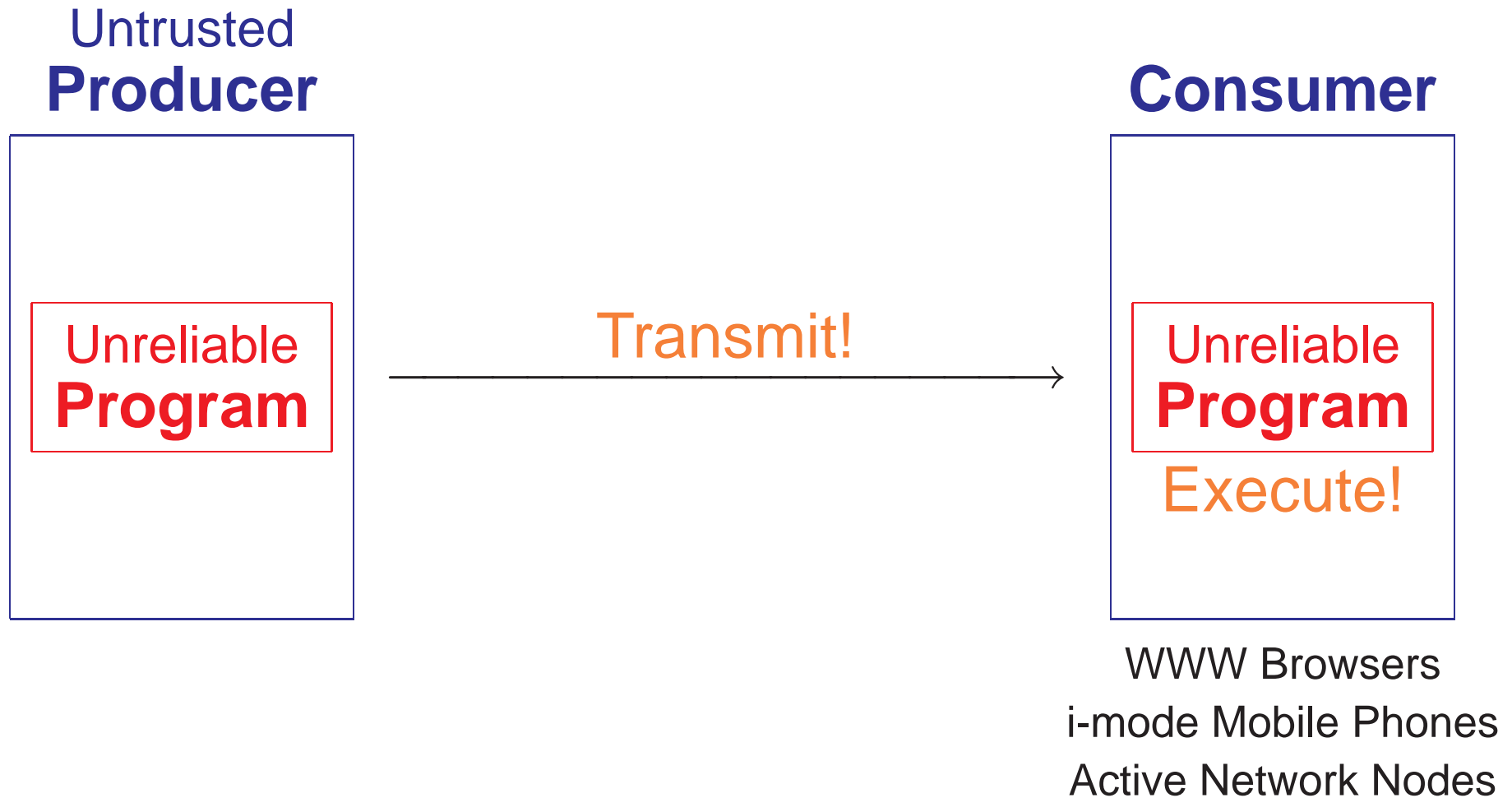


**Consumer**



WWW Browsers  
i-mode Mobile Phones  
Active Network Nodes

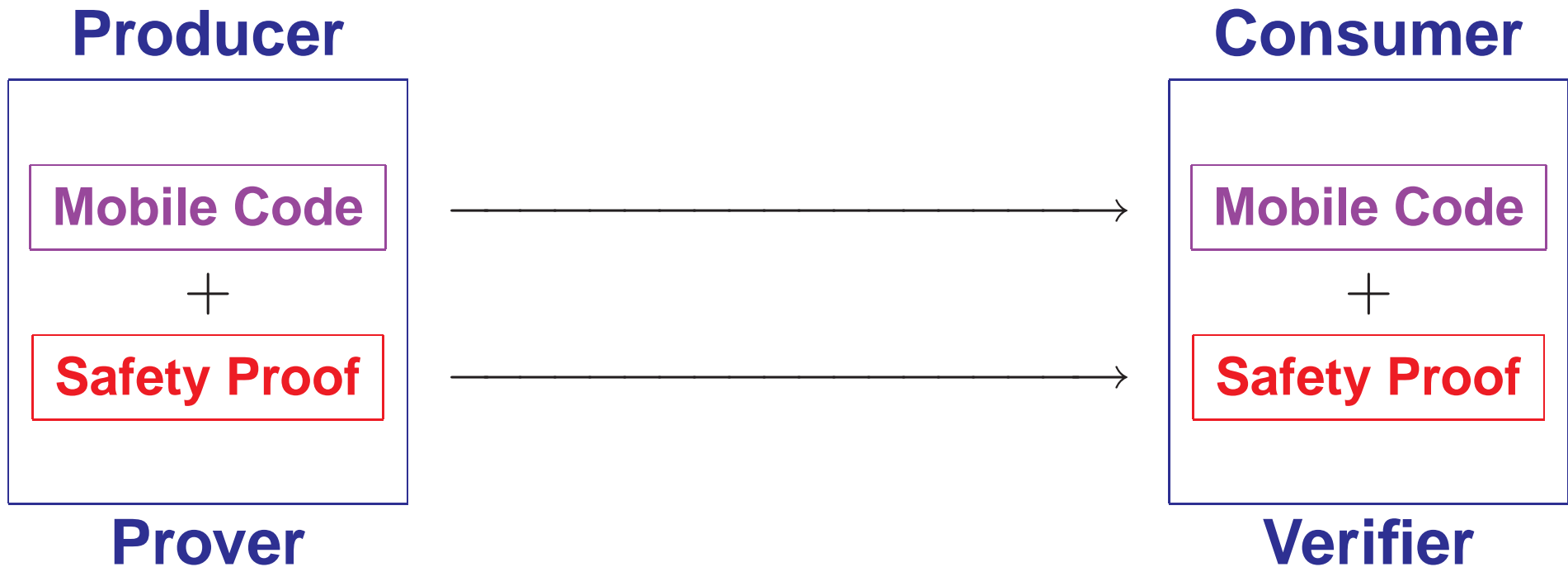
# A Proof-Based Approach to Safe Extension of Distributed Software Systems



# Proof-Carrying Code (PCC)

Mobile Code + Safety Proof

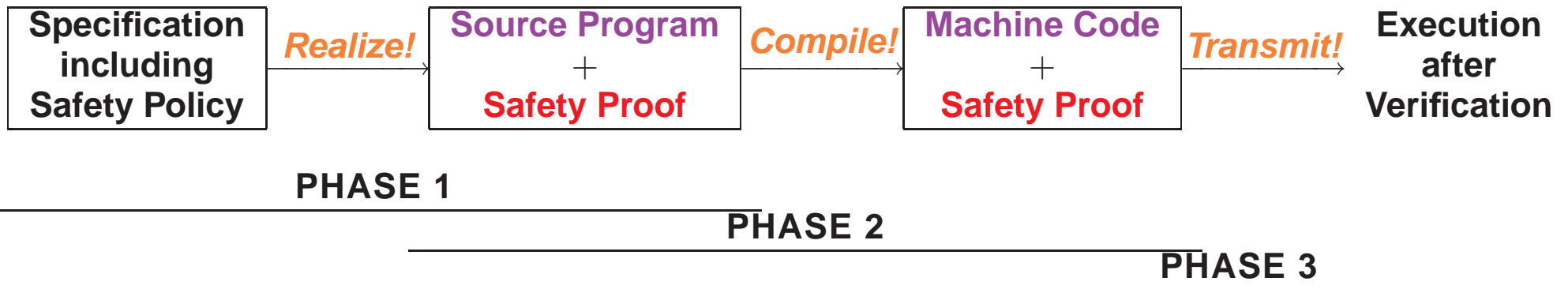
[Necula-Lee 1997]



# PCC: Advantages

- **Higher level of safety** than the code signing mechanism. Semantics of codes vs. Authenticity of code producers.
- **Expressive framework** for a variety of safety properties (owing to the use of first-order predicate logic).
- **Quick execution** without any additional run-time checks (by virtue of the static verification mechanism).
  
- Complex proof-producing procedure can be performed **off-line and only once** for a given code.
- **Certifying compilers** translate “source programs” into “machine codes + safety proofs.”

# Provably Safe Program Distribution



## PCC: Problem

Proofs can be exponentially large.

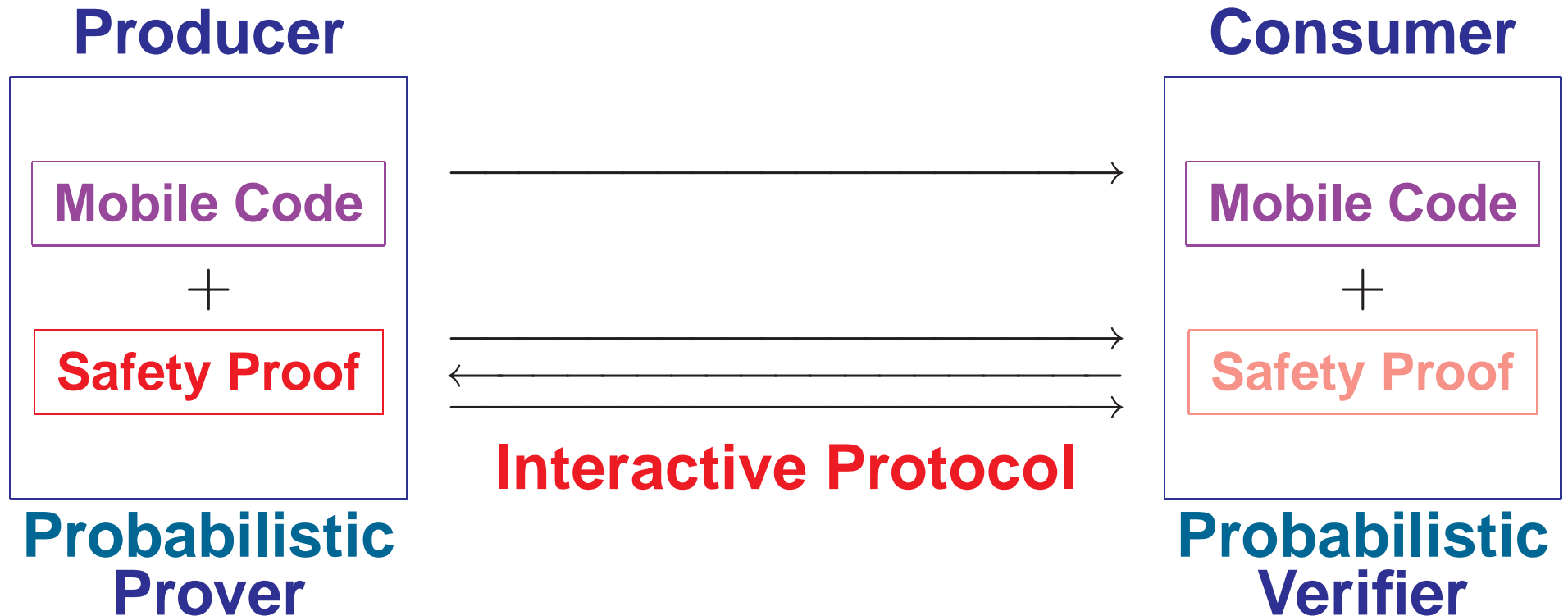
|**Mobile Code**|  $\ll$  |**Safety Proof**|.

If each consumer is allowed to spend only **a reasonable amount of time on verification**, PCC is applicable only to codes that are equipped with **reasonably “short” proofs**.

# Interactive Proof-Carrying Code (iPCC)

Mobile Code + Interactive Protocol  
for Safety Proof

Application of [Goldwasser–Micali–Rackoff 1985]



# iPCC: Advantage

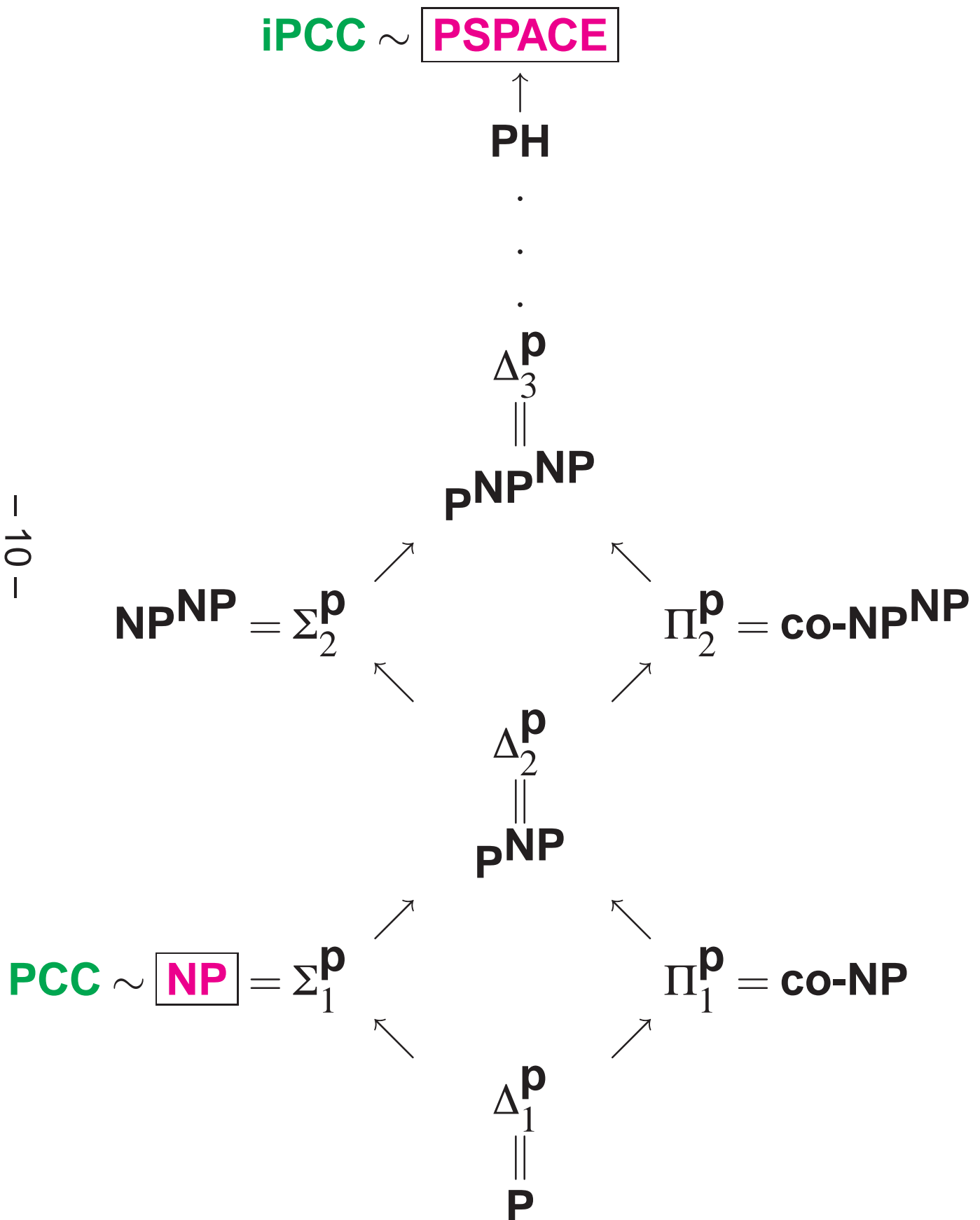
iPCC can prove a greater variety of safety properties that may require longer proofs.

The Class  
**NP**  
of Safety Properties  
Efficiently Provable by  
**PCC**

<

The Class  
**PSPACE**  
of Safety Properties  
Efficiently Provable by  
**iPCC**

# Polynomial-Time Hierarchy



# Abstract Framework for iPCC

Code Producer/Prover

Code Consumer/Verifier

Produce a **code**

$f$

Generate the **verification condition**

$\{Pre\}f\{Post\}$

Find a **safety proof**

$T \vdash p : \{Pre\}f\{Post\}$

Send the **code**

$f$

Receive the **code**

$T \vdash p : \{Pre\}f\{Post\}$

$f$

Prove the existence of a **safety proof**

Verify the existence of the **safety proof**

$f \in Safe$

$\Updownarrow$

$\exists p. T \vdash p : \{Pre\}f\{Post\}?$

$f$

Execute the **code**

## Case Study [Necula-Lee 1997]

Code: Implemented in the  
**Alpha Assembly-Language**

Consumer: **Run-Time System of the  
TIL Compiler for Standard ML**

Safety Requirements: **Memory Safety & Type Safety**

## Example: Code $f$

```

INV Pre           % Precondition
LD  $r_4, 4(r_1)$     % Load constituent
LD  $r_1, 0(r_1)$     % Load constructor
LD  $r_5, 4(r_2)$     % Load constituent
LD  $r_2, 0(r_2)$     % Load constructor
LD  $r_6, 4(r_3)$     % Load constituent
LD  $r_3, 0(r_3)$     % Load constructor
BEQ  $r_1, L_a$       % Is constructor  $inj_l$ ?
La BEQ  $r_2, L_b$   % Is constructor  $inj_l$ ?
Lb BEQ  $r_3, L_c$   % Is constructor  $inj_l$ ?
Lc INV Post      % Postcondition
RET

```

$$\begin{aligned}
 Pre \equiv & \mathbf{r}_m \vdash \mathbf{r}_1 : \mathbf{int} + \mathbf{int} * \mathbf{int} \wedge \\
 & \mathbf{r}_m \vdash \mathbf{r}_2 : \mathbf{int} + \mathbf{int} * \mathbf{int} \wedge \\
 & \mathbf{r}_m \vdash \mathbf{r}_3 : \mathbf{int} + \mathbf{int} * \mathbf{int} \\
 Post \equiv & \mathbf{r}_m \vdash \mathbf{sel}(\mathbf{r}_m, \mathbf{r}_4) + \mathbf{sel}(\mathbf{r}_m, \mathbf{r}_5) + \mathbf{sel}(\mathbf{r}_m, \mathbf{r}_6) : \mathbf{int} \vee \\
 & \mathbf{r}_m \vdash \mathbf{sel}(\mathbf{r}_m, \mathbf{r}_4) + \mathbf{r}_5 : \mathbf{int} \vee \\
 & \mathbf{r}_m \vdash \mathbf{sel}(\mathbf{r}_m, \mathbf{r}_5) + \mathbf{r}_6 : \mathbf{int} \vee \\
 & \mathbf{r}_m \vdash \mathbf{r}_4 + \mathbf{sel}(\mathbf{r}_m, \mathbf{r}_6) : \mathbf{int} \vee \\
 & \mathbf{r}_m \vdash \mathbf{r}_4 + \mathbf{r}_5 + \mathbf{r}_6 : \mathbf{int}
 \end{aligned}$$

## Example: Verification Condition $\{Pre\}f\{Post\}$

$$\begin{aligned}
 & Pre \Rightarrow \mathbf{r}_m \vdash \mathbf{r}_1 + 4 : \mathbf{addr} \wedge \mathbf{r}_m \vdash \mathbf{r}_1 : \mathbf{addr} \wedge \mathbf{r}_m \vdash \mathbf{r}_2 + 4 : \mathbf{addr} \\
 & Pre \Rightarrow \mathbf{r}_m \vdash \mathbf{r}_2 : \mathbf{addr} \wedge \mathbf{r}_m \vdash \mathbf{r}_3 + 4 : \mathbf{addr} \wedge \mathbf{r}_m \vdash \mathbf{r}_3 : \mathbf{addr} \\
 & Pre \wedge \mathbf{sel}(\mathbf{r}_m, \mathbf{r}_1) = 0 \wedge \mathbf{sel}(\mathbf{r}_m, \mathbf{r}_2) = 0 \wedge \mathbf{sel}(\mathbf{r}_m, \mathbf{r}_3) = 0 \Rightarrow Post^* \\
 & Pre \wedge \mathbf{sel}(\mathbf{r}_m, \mathbf{r}_1) = 0 \wedge \mathbf{sel}(\mathbf{r}_m, \mathbf{r}_2) = 0 \wedge \mathbf{sel}(\mathbf{r}_m, \mathbf{r}_3) \neq 0 \Rightarrow Post^* \\
 \bigwedge & Pre \wedge \mathbf{sel}(\mathbf{r}_m, \mathbf{r}_1) = 0 \wedge \mathbf{sel}(\mathbf{r}_m, \mathbf{r}_2) \neq 0 \wedge \mathbf{sel}(\mathbf{r}_m, \mathbf{r}_3) = 0 \Rightarrow Post^* \\
 & Pre \wedge \mathbf{sel}(\mathbf{r}_m, \mathbf{r}_1) = 0 \wedge \mathbf{sel}(\mathbf{r}_m, \mathbf{r}_2) \neq 0 \wedge \mathbf{sel}(\mathbf{r}_m, \mathbf{r}_3) \neq 0 \Rightarrow Post^* \\
 & Pre \wedge \mathbf{sel}(\mathbf{r}_m, \mathbf{r}_1) \neq 0 \wedge \mathbf{sel}(\mathbf{r}_m, \mathbf{r}_2) = 0 \wedge \mathbf{sel}(\mathbf{r}_m, \mathbf{r}_3) = 0 \Rightarrow Post^* \\
 & Pre \wedge \mathbf{sel}(\mathbf{r}_m, \mathbf{r}_1) \neq 0 \wedge \mathbf{sel}(\mathbf{r}_m, \mathbf{r}_2) = 0 \wedge \mathbf{sel}(\mathbf{r}_m, \mathbf{r}_3) \neq 0 \Rightarrow Post^* \\
 & Pre \wedge \mathbf{sel}(\mathbf{r}_m, \mathbf{r}_1) \neq 0 \wedge \mathbf{sel}(\mathbf{r}_m, \mathbf{r}_2) \neq 0 \wedge \mathbf{sel}(\mathbf{r}_m, \mathbf{r}_3) = 0 \Rightarrow Post^* \\
 & Pre \wedge \mathbf{sel}(\mathbf{r}_m, \mathbf{r}_1) \neq 0 \wedge \mathbf{sel}(\mathbf{r}_m, \mathbf{r}_2) \neq 0 \wedge \mathbf{sel}(\mathbf{r}_m, \mathbf{r}_3) \neq 0 \Rightarrow Post^*
 \end{aligned}$$

$$\begin{aligned}
 Post^* \equiv & \mathbf{r}_m \vdash \mathbf{sel}(\mathbf{r}_m, \mathbf{sel}(\mathbf{r}_m, \mathbf{r}_1 + 4)) + \mathbf{sel}(\mathbf{r}_m, \mathbf{sel}(\mathbf{r}_m, \mathbf{r}_2 + 4)) + \mathbf{sel}(\mathbf{r}_m, \mathbf{sel}(\mathbf{r}_m, \mathbf{r}_3 + 4)) : \mathbf{int} \vee \\
 & \mathbf{r}_m \vdash \mathbf{sel}(\mathbf{r}_m, \mathbf{sel}(\mathbf{r}_m, \mathbf{r}_1 + 4)) + \mathbf{sel}(\mathbf{r}_m, \mathbf{r}_2 + 4) : \mathbf{int} \vee \\
 & \mathbf{r}_m \vdash \mathbf{sel}(\mathbf{r}_m, \mathbf{sel}(\mathbf{r}_m, \mathbf{r}_2 + 4)) + \mathbf{sel}(\mathbf{r}_m, \mathbf{r}_3 + 4) : \mathbf{int} \vee \\
 & \mathbf{r}_m \vdash \mathbf{sel}(\mathbf{r}_m, \mathbf{r}_1 + 4) + \mathbf{sel}(\mathbf{r}_m, \mathbf{sel}(\mathbf{r}_m, \mathbf{r}_3 + 4)) : \mathbf{int} \vee \\
 & \mathbf{r}_m \vdash \mathbf{sel}(\mathbf{r}_m, \mathbf{r}_1 + 4) + \mathbf{sel}(\mathbf{r}_m, \mathbf{r}_2 + 4) + \mathbf{sel}(\mathbf{r}_m, \mathbf{r}_3 + 4) : \mathbf{int}
 \end{aligned}$$

- means that if  $f$  is executed on the consumer system starting in a state that satisfies  $Pre$ , then
  - $f$  reads **only from valid memory locations**, and
  - if  $f$  terminates, the final state satisfies  $Post$ .
- generated by the **Floyd-style Verification Condition Generator**



# Example: Formal System $T$

## First-Order Predicate Logic

$$\frac{m \vdash e : \tau_1 * \tau_2}{m \vdash e : \mathbf{addr} \wedge \\ m \vdash e + 4 : \mathbf{addr} \wedge \\ m \vdash \mathbf{sel}(m, e) : \tau_1 \wedge \\ m \vdash \mathbf{sel}(m, e + 4) : \tau_2}$$

$$\frac{m \vdash e : \tau_1 + \tau_2}{m \vdash e : \mathbf{addr} \wedge \\ m \vdash e + 4 : \mathbf{addr} \wedge \\ \mathbf{sel}(m, e) = 0 \Rightarrow m \vdash \mathbf{sel}(m, e + 4) : \tau_1 \wedge \\ \mathbf{sel}(m, e) \neq 0 \Rightarrow m \vdash \mathbf{sel}(m, e + 4) : \tau_2}$$

$$\frac{m \vdash e : \tau \mathbf{list} \quad e \neq 0}{m \vdash e : \mathbf{addr} \wedge \\ m \vdash e + 4 : \mathbf{addr} \wedge \\ m \vdash \mathbf{sel}(m, e) : \tau \wedge \\ m \vdash \mathbf{sel}(m, e + 4) : \tau \mathbf{list}}$$

$$\frac{m \vdash e_1 : \mathbf{int} \quad m \vdash e_2 : \mathbf{int}}{m \vdash e_1 + e_2 : \mathbf{int}} \quad \frac{}{m \vdash 0 : \mathbf{int}}$$

## Example: Safety Property *Safe*

The prover must convince the verifier that

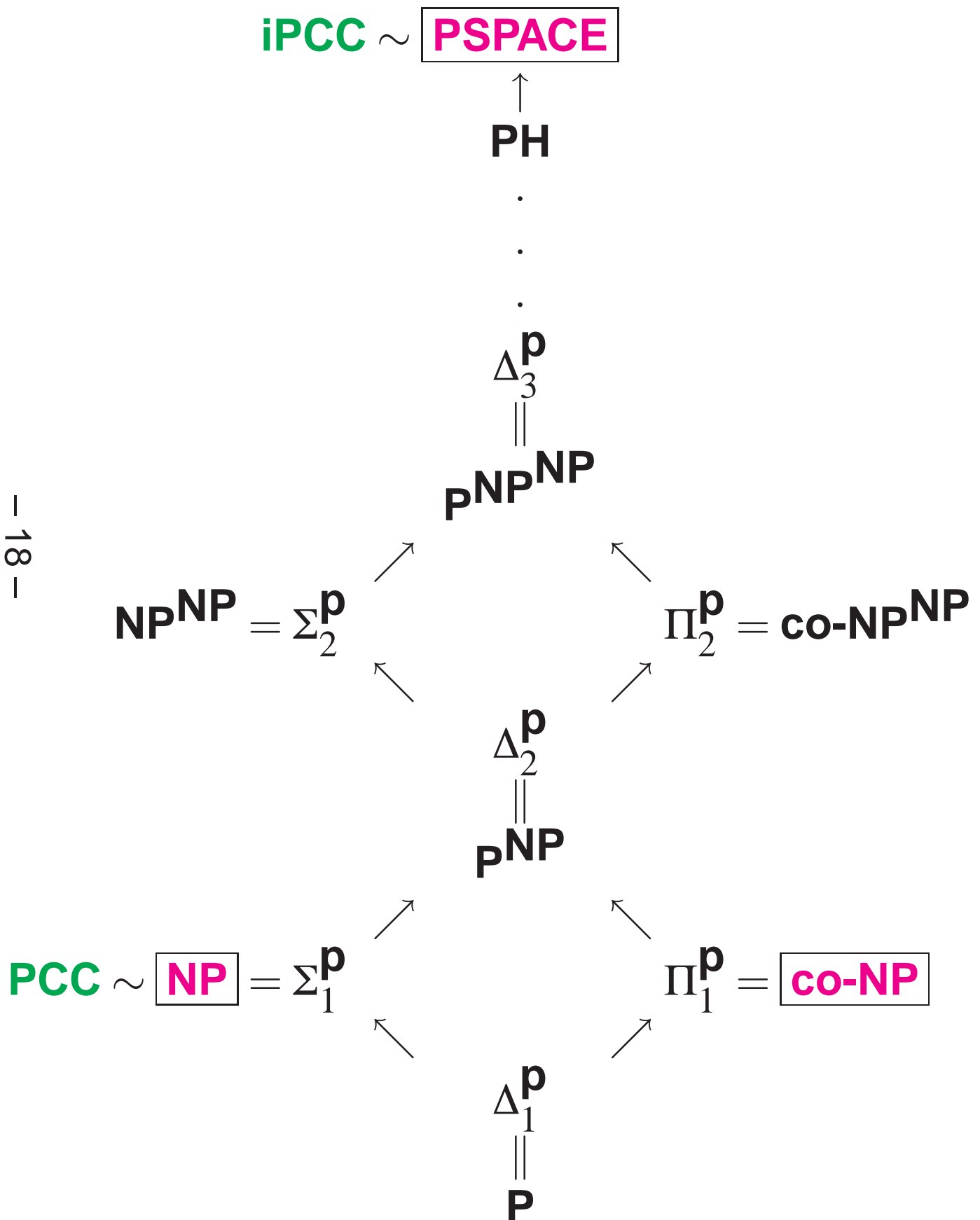
$$f \in \textit{Safe} \Leftrightarrow \exists p. T \vdash p : \boxed{\{Pre\}f\{Post\}}.$$

**Theorem.** *Safe* is co-NP-complete.

(i.e., *Safe* is not simple enough to be efficiently certified by PCC.)

$$\begin{aligned} f \in \textit{Safe} &\Leftrightarrow \exists p. T \vdash p : \boxed{\bigwedge_{\text{path}_{\leq |f|}} \varphi_{\text{path}}} \\ &\Leftrightarrow \forall \text{path}_{\leq |f|} \exists p. T \vdash p : \varphi_{\text{path}} \end{aligned}$$

# Polynomial-Time Hierarchy



# Example: Interactive Proof I

## 1. Encoding to a QBF (Quantified Boolean Formula)

$$\begin{array}{c} f \in \text{Safe} \\ \Downarrow \\ \exists X_1 \exists X_2 \exists X_3. (X_1 \vee X_2 \vee X_3) \wedge (X_1 \vee \neg X_2) \wedge (X_2 \vee \neg X_3) \wedge (\neg X_1 \vee X_3) \wedge (\neg X_1 \vee \neg X_2 \vee \neg X_3) \\ \text{is a false QBF.} \end{array}$$

N.B. *Safe*: co-NP-complete.  
False QBFs: PSPACE-complete.

## 2. Arithmetization

$$\begin{array}{c} \exists X_1 \exists X_2 \exists X_3. (X_1 \vee X_2 \vee X_3) \wedge (X_1 \vee \neg X_2) \wedge (X_2 \vee \neg X_3) \wedge (\neg X_1 \vee X_3) \wedge (\neg X_1 \vee \neg X_2 \vee \neg X_3) \\ \text{is a false QBF} \\ \Downarrow \\ \sum_{X_1=0}^1 \sum_{X_2=0}^1 \sum_{X_3=0}^1 (X_1 + X_2 + X_3)(X_1 + (1 - X_2))(X_2 + (1 - X_3))((1 - X_1) + X_3)((1 - X_1) + (1 - X_2) + (1 - X_3)) \\ = 0. \end{array}$$

## 3. Protocol [Shamir 1990]

# Example: Interactive Proof II

Code Producer/Prover

Code Consumer/Verifier

ROUND 1:

$$A \equiv \sum_{X_1=0}^1 \sum_{X_2=0}^1 \sum_{X_3=0}^1 (X_1+X_2+X_3)(X_1+(1-X_2))(X_2+(1-X_3))((1-X_1)+X_3)((1-X_1)+(1-X_2)+(1-X_3)) = 0?$$

$$A[X_1] \equiv \sum_{X_2=0}^1 \sum_{X_3=0}^1 (X_1+X_2+X_3)(X_1+(1-X_2))(X_2+(1-X_3))((1-X_1)+X_3)((1-X_1)+(1-X_2)+(1-X_3)) = 3X_1^4 - 6X_1^3 - 6X_1^2 + 9X_1$$

$$A = A[0] + A[1] = 0 + 0 = 0 \quad \text{OK!}$$

Randomly chosen number: 2

ROUND 2:

$$A[2] = \sum_{X_2=0}^1 \sum_{X_3=0}^1 (2+X_2+X_3)(3-X_2)(X_2-X_3+1)(X_3-1)(1-X_2-X_3) = -6?$$

$$A[2][X_2] \equiv \sum_{X_3=0}^1 (2+X_2+X_3)(3-X_2)(X_2-X_3+1)(X_3-1)(1-X_2-X_3) = -X_2^4 + X_2^3 + 7X_2^2 - X_2 - 6$$

$$A[2] = A[2][0] + A[2][1] = -6 + 0 = -6 \quad \text{OK!}$$

Randomly chosen number: 4

ROUND 3:

$$A[2][4] = \sum_{X_3=0}^1 (6+X_3)(-1)(5-X_3)(X_3-1)(-3-X_3) = -90?$$

$$A[2][4][X_3] \equiv (6+X_3)(-1)(5-X_3)(X_3-1)(-3-X_3) = -X_3^4 - 3X_3^3 + 31X_3^2 + 63X_3 - 90$$

$$A[2][4] = A[2][4][0] + A[2][4][1] = -90 + 0 = -90 \quad \text{OK!}$$

Convinced!

# Conclusion

- **iPCC:**  
**An interactive and probabilistic extension of PCC.**
- **It solves the explosion problem of proofs with PCC.**

## Future Work I

How can we construct an efficient interactive proof system **for an arbitrary safety property?**

Even for a specific safety property discussed in this talk, the encoding I suggested may not work efficiently.

## Future Work II

Does the interactive protocols constructed have **sufficient performance for the practical problem domain?**

I only discuss the **worst case** computational cost in terms of **P**, **NP**, **co-NP**, **PSPACE**, etc.

## Future Work III

Are there any **interesting practical examples** of codes and proofs that can be transmitted efficiently with iPCC but not with PCC?

- **For memory safety & type safety**

PCC is OK [Necula 2001].

oracle-based PCC + sufficiently many invariants

(iPCC + a small number of invariants)

- **For other advanced safety properties**

iPCC can have an advantage.

$f \in \textit{Safe} \Leftrightarrow \mathcal{K}_f \models \varphi$  (model checking of temporal formulas)

## Future Work IV

PCC/iPCC proofs can contain **too much information...**

What each producer actually wants to show is:

not **the safety proof itself**

but **the “existence” of a valid safety proof.**

(only 1 bit of information!)

**Zero-Knowledge Proof-Carrying Code (zkPCC)**

proves the existence of a valid safety proof

**without revealing any actual information  
about the safety proof itself!**

# Proofs-as-Copyrights Interpretation

$P_1, \dots, P_n$ : competitive agents.

$V$ : a client requesting a **code** and its **proof**.

## PCC:

- Proofs are open and can be used by stealth.
- A cheater  $P_j$ , disguised as  $V$ , might steal the **code** and its **proof** from another  $P_i$  and then sell it to  $V$ .

## zkPCC:

- Only the real producer knows the proof contents.
- The zero-knowledge property guarantees that the **proof** functions as a kind of © of the **code**.