Proof-Carrying Code in a Session-Typed Process Calculus

Frank Pfenning

with Luís Caires and Bernardo Toninho

Department of Computer Science
Carnegie Mellon University

1st International Conference on
Certified Programs and Proofs (CPP)
December 7, 2011

Why do we trust software?

Why do we trust software?

- We don't!

Why do we trust software?

- We don't!
- To the extent that we do, we rely on:

Why do we trust software?

- We don't!
- To the extent that we do, we rely on:
  - Digital signatures (state-of-the-practice)
  - Formal proof (state-of-the-art)

Why do we trust software?

- We don't!
- To the extent that we do, we rely on:
  - Digital signatures (state-of-the-practice)
  - Formal proof (state-of-the-art)
- Can we combine digital signatures and proofs?

Why do we trust software?

- We don't!
- To the extent that we do, we rely on:
  - Digital signatures (state-of-the-practice)
  - Formal proof (state-of-the-art)
- Can we combine digital signatures and proofs?
  - Digital signatures are here to stay
  - Proofs are here to stay

System Modeling and Security Properties

System Modeling and Security Properties

- Communicating processes

System Modeling and Security Properties

- Communicating processes
  - Name-passing (mobile)
  - Value-passing (applied)
  - Proof-passing (proof-carrying)

System Modeling and Security Properties

- Communicating processes
  - Name-passing (mobile)
  - Value-passing (applied)
  - Proof-passing (proof-carrying)
- Reason about process behavior

System Modeling and Security Properties

- Communicating processes
  - Name-passing (mobile)
  - Value-passing (applied)
  - Proof-passing (proof-carrying)
- Reason about process behavior
  - Deadlock-freedom
  - Session fidelity
  - Termination

System Modeling and Security Properties

- Communicating processes
  - Name-passing (mobile)
  - Value-passing (applied)
  - Proof-passing (proof-carrying)
- Reason about process behavior
  - Deadlock-freedom
  - Session fidelity
  - Termination
- Reason about values and proofs

System Modeling and Security Properties

- Communicating processes
  - Name-passing (mobile)
  - Value-passing (applied)
  - Proof-passing (proof-carrying)
- Reason about process behavior
  - Deadlock-freedom
  - Session fidelity
  - Termination
- Reason about values and proofs
  - Types
  - Correctness of proofs
  - Validity of signatures

Approach

Approach

- Communicating processes

Approach

- Communicating processes
  - Value-passing extension of π-calculus

Approach

- Communicating processes
  - Value-passing extension of π-calculus
- Reason about process behavior [CONCUR'10]

Approach

- Communicating processes
  - Value-passing extension of π-calculus
- Reason about process behavior [CONCUR'10]
  - Session types
  - Curry-Howard isomorphism between
    - Intuitionistic linear propositions and session types
    - Sequent proofs and π-calculus processes
    - Proof reduction and process reduction

Approach

- Communicating processes
  - Value-passing extension of π-calculus
- Reason about process behavior [CONCUR'10]
  - Session types
  - Curry-Howard isomorphism between
    - Intuitionistic linear propositions and session types
    - Sequent proofs and π-calculus processes
    - Proof reduction and process reduction
- Reason about values and proofs

Approach

- Communicating processes
  - Value-passing extension of π-calculus
- Reason about process behavior [CONCUR'10]
  - Session types
  - Curry-Howard isomorphism between
    - Intuitionistic linear propositions and session types
    - Sequent proofs and π-calculus processes
    - Proof reduction and process reduction
- Reason about values and proofs
  - Dependent session types [PPDP'11]
  - Terms and proofs from dependent type theory
  - Add proof irrelevance (to avoid sending proofs)
  - Add affirmation (to capture digital signatures)

Outline

Session types: judgment forms

- Judgment $P :: x : A$
  - Process P offers service A along channel x
- Linear sequent

$$\underbrace{x_1{:}A_1, \ldots, x_n{:}A_n}_{\Delta} \Rightarrow P :: x : A$$

P uses $x_i{:}A_i$ and offers $x{:}A$.

- Cut as composition

$$\frac{\Delta \Rightarrow \qquad\qquad A \quad \Delta', \quad A \Rightarrow \qquad\qquad C}{\Delta, \Delta' \Rightarrow \qquad\qquad\qquad\qquad C}\ \text{cut}$$

- Identity as forwarding

$$\frac{}{A \Rightarrow \qquad\qquad\qquad\qquad A}\ \text{id}$$

Session types: judgment forms

- Judgment $P :: x : A$
  - Process P offers service A along channel x
- Linear sequent

$$\underbrace{x_1{:}A_1, \ldots, x_n{:}A_n}_{\Delta} \Rightarrow P :: x : A$$

P uses $x_i{:}A_i$ and offers $x{:}A$.

- Cut as composition

$$\frac{\Delta \Rightarrow \qquad x : A \quad \Delta', x{:}A \Rightarrow \qquad z : C}{\Delta, \Delta' \Rightarrow \qquad z : C}\ \text{cut}$$

- Identity as forwarding

$$\frac{}{A \Rightarrow \qquad A}\ \text{id}$$

Session types: judgment forms

- Judgment $P :: x : A$
  - Process P offers service A along channel x
- Linear sequent

$$\underbrace{x_1{:}A_1, \ldots, x_n{:}A_n}_{\Delta} \Rightarrow P :: x : A$$

P uses $x_i{:}A_i$ and offers $x{:}A$.

- Cut as composition

$$\frac{\Delta \Rightarrow P :: x : A \quad \Delta', x{:}A \Rightarrow Q :: z : C}{\Delta, \Delta' \Rightarrow (\boldsymbol{\nu} x)(P \mid Q) :: z : C}\ \text{cut}$$

- Identity as forwarding

$$\frac{}{A \Rightarrow \qquad\qquad\qquad A}\ \text{id}$$

Session types: judgment forms

- Judgment $P :: x : A$
  - Process P offers service A along channel x
- Linear sequent

$$\underbrace{x_1{:}A_1, \ldots, x_n{:}A_n}_{\Delta} \Rightarrow P :: x : A$$

P uses $x_i{:}A_i$ and offers $x{:}A$.

- Cut as composition

$$\frac{\Delta \Rightarrow P :: x : A \quad \Delta', x{:}A \Rightarrow Q :: z : C}{\Delta, \Delta' \Rightarrow (\boldsymbol{\nu} x)(P \mid Q) :: z : C}\ \text{cut}$$

- Identity as forwarding

$$\frac{}{x{:}A \Rightarrow \qquad\qquad z : A}\ \text{id}$$

Session types: judgment forms

- Judgment $P :: x : A$
  - Process P offers service A along channel x
- Linear sequent

$$\underbrace{x_1{:}A_1, \ldots, x_n{:}A_n}_{\Delta} \Rightarrow P :: x : A$$

P uses $x_i{:}A_i$ and offers $x{:}A$.

- Cut as composition

$$\frac{\Delta \Rightarrow P :: x : A \quad \Delta', x{:}A \Rightarrow Q :: z : C}{\Delta, \Delta' \Rightarrow (\boldsymbol{\nu} x)(P \mid Q) :: z : C}\ \text{cut}$$

- Identity as forwarding

$$\frac{}{x{:}A \Rightarrow [x \leftrightarrow z] :: z : A}\ \text{id}$$

Session types: input ($A \multimap B$)

- $P :: x : A \multimap B$
  - P inputs an A along x and then behaves as B
- Right rule: offer of service

$$\frac{\Delta, \quad A \Rightarrow \qquad\qquad B}{\Delta \Rightarrow \qquad\qquad A \multimap B} \multimap R$$

  - Can reuse x, due to linearity
- Left rule: matching use of service

$$\frac{\Delta \Rightarrow \qquad\qquad A \quad \Delta', \quad B \Rightarrow \qquad\qquad C}{\Delta, \Delta', \quad A \multimap B \Rightarrow \qquad\qquad C} \multimap L$$

  - Can reuse x, due to linearity
  - Channel y must be new

Session types: input ($A \multimap B$)

- $P :: x : A \multimap B$
  - P inputs an A along x and then behaves as B
- Right rule: offer of service

$$\frac{\Delta, y{:}A \Rightarrow \quad x : B}{\Delta \Rightarrow \quad x : A \multimap B} \multimap R$$

  - Can reuse x, due to linearity
- Left rule: matching use of service

$$\frac{\Delta \Rightarrow \quad A \quad \Delta', \quad B \Rightarrow \quad C}{\Delta, \Delta', \quad A \multimap B \Rightarrow \quad C} \multimap L$$

  - Can reuse x, due to linearity
  - Channel y must be new

Session types: input ($A \multimap B$)

- $P :: x : A \multimap B$
  - P inputs an A along x and then behaves as B
- Right rule: offer of service

$$\frac{\Delta, y{:}A \Rightarrow P :: x : B}{\Delta \Rightarrow x(y).P :: x : A \multimap B} \multimap R$$

  - Can reuse x, due to linearity
- Left rule: matching use of service

$$\frac{\Delta \Rightarrow \qquad\qquad A \quad \Delta', \quad B \Rightarrow \qquad\qquad C}{\Delta, \Delta', \quad A \multimap B \Rightarrow \qquad\qquad\qquad\qquad\qquad C} \multimap L$$

  - Can reuse x, due to linearity
  - Channel y must be new

Session types: input ($A \multimap B$)

- $P :: x : A \multimap B$
  - P inputs an A along x and then behaves as B
- Right rule: offer of service

$$\frac{\Delta, y{:}A \Rightarrow P :: x : B}{\Delta \Rightarrow x(y).P :: x : A \multimap B} \multimap R$$

  - Can reuse x, due to linearity
- Left rule: matching use of service

$$\frac{\Delta \Rightarrow \quad y : A \quad \Delta', x{:}B \Rightarrow \quad z : C}{\Delta, \Delta', x{:}A \multimap B \Rightarrow \quad z : C} \multimap L$$

  - Can reuse x, due to linearity
  - Channel y must be new

Session types: input ($A \multimap B$)

- $P :: x : A \multimap B$
  - P inputs an A along x and then behaves as B
- Right rule: offer of service

$$\frac{\Delta, y{:}A \Rightarrow P :: x : B}{\Delta \Rightarrow x(y).P :: x : A \multimap B} \multimap R$$

  - Can reuse x, due to linearity
- Left rule: matching use of service

$$\frac{\Delta \Rightarrow P :: y : A \quad \Delta', x{:}B \Rightarrow Q :: z : C}{\Delta, \Delta', x{:}A \multimap B \Rightarrow (\boldsymbol{\nu} y) x\langle y\rangle.(P \mid Q) :: z : C} \multimap L$$

  - Can reuse x, due to linearity
  - Channel y must be new

Session types: reduction

- Proof reduction

$$\dfrac{\dfrac{\Delta, A \Rightarrow B}{\Delta \Rightarrow A \multimap B}\multimap R \quad \dfrac{\Delta_1 \Rightarrow A \quad \Delta_2, B \Rightarrow C}{\Delta_1, \Delta_2, A \multimap B \Rightarrow C}\multimap L}{\Delta, \Delta_1, \Delta_2 \Rightarrow C}\text{cut}$$

$$\longrightarrow$$

$$\dfrac{\dfrac{\Delta_1 \Rightarrow A \quad \Delta, A \Rightarrow B}{\Delta, \Delta_1 \Rightarrow B}\text{cut} \quad \Delta_2, B \Rightarrow C}{\Delta, \Delta_1, \Delta_2 \Rightarrow C}\text{cut}$$

- Corresponding process reduction

$$\Delta, \Delta_1, \Delta_2 \Rightarrow (\boldsymbol{\nu} x)(x(y).P_1 \mid (\boldsymbol{\nu} w)(x\langle w\rangle.(P_2 \mid Q))) :: z : C$$

$$\longrightarrow$$

$$\Delta, \Delta_1, \Delta_2 \Rightarrow (\boldsymbol{\nu} x)((\boldsymbol{\nu} w)(P_2 \mid P_1\{w/y\}) \mid Q) :: z : C$$

Session types: other connectives

- Linear propositions as session types

$P :: x : A \multimap B$	Input a $y{:}A$ along x and behave as B
$P :: x : A \otimes B$	Output a new $y{:}A$ along x and behave as B
$P :: x : \mathbf{1}$	Terminate session on x
$P :: x : A \,\&\, B$	Offer choice between A and B along x
$P :: x : A \oplus B$	Offer either A or B along x
$P :: x : !A$	Offer A persistently along x

- Sequent proofs as process expressions
- Proof reduction as process reduction

Two small examples

- PDF indexing service, version 1

$$\text{index}_1 : !(\text{file} \multimap \text{file} \otimes \mathbf{1})$$

  Persistently offer to input a file, then output a file and terminate session. Intent: input PDF, output indexed PDF for keyword search.

- Persistent file storage

$$\text{store}_1 : !(\text{file} \multimap !(\text{file} \otimes \mathbf{1}))$$

  Persistently offer to input a file, then output a persistent handle for retrieving this file. Intent: output file is the same as input file.

Outline

Term passing

- Types τ from a (dependent) type theory
- Hypothetical judgment $\underbrace{x_1{:}\tau_1, \ldots, x_k{:}\tau_k}_{\Psi} \vdash M : \tau$
- Some example type constructors

$\Pi x{:}\tau.\sigma$, $\tau \to \sigma$	Functions from τ to σ
$\Sigma x{:}\tau.\sigma$, $\tau \times \sigma$	Pairs of a τ and a σ
nat	Natural numbers

- Integrate into sequent calculus

$$\underbrace{\Psi}_{\text{term variables}} \; ; \; \underbrace{\Gamma}_{\text{persistent channels}} \; ; \; \underbrace{\Delta}_{\text{linear channels}} \Rightarrow P :: \underbrace{x : A}_{\text{linear}}$$

Term passing: input ($\forall y{:}\tau.A$)

- $P :: x : \forall y{:}\tau A$
  - P inputs an $M : \tau$ along x and then behaves as $A\{M/x\}$
- Right rule: offer of service

$$\frac{\Psi, y{:}\tau \,;\, \Gamma \,;\, \Delta \Rightarrow \quad A}{\Psi \,;\, \Gamma \,;\, \Delta \Rightarrow \quad \forall y{:}\tau.A}\ \forall R$$

- Left rule: matching use of service

$$\frac{\Psi \vdash M : \tau \quad \Psi \,;\, \Gamma \,;\, \Delta', \quad A\{M/y\} \Rightarrow \quad C}{\Psi \,;\, \Gamma \,;\, \Delta', \quad \forall y{:}\tau.A \Rightarrow \quad C}\ \forall L$$

- Proof reduction yields

$$\longrightarrow$$

Term passing: input ($\forall y{:}\tau.A$)

- $P :: x : \forall y{:}\tau A$
  - P inputs an $M : \tau$ along x and then behaves as $A\{M/x\}$
- Right rule: offer of service

$$\frac{\Psi, y{:}\tau \; ; \Gamma \; ; \Delta \Rightarrow \qquad x : A}{\Psi \; ; \Gamma \; ; \Delta \Rightarrow \qquad x : \forall y{:}\tau.A} \; \forall R$$

- Left rule: matching use of service

$$\frac{\Psi \vdash M : \tau \quad \Psi \; ; \Gamma \; ; \Delta', \quad A\{M/y\} \Rightarrow \qquad C}{\Psi \; ; \Gamma \; ; \Delta', \quad \forall y{:}\tau.A \Rightarrow \qquad C} \; \forall L$$

- Proof reduction yields

$$\longrightarrow$$

Term passing: input ($\forall y{:}\tau.A$)

- $P :: x : \forall y{:}\tau A$
  - P inputs an $M : \tau$ along x and then behaves as $A\{M/x\}$
- Right rule: offer of service

$$\frac{\Psi, y{:}\tau \,;\, \Gamma \,;\, \Delta \Rightarrow P :: x : A}{\Psi \,;\, \Gamma \,;\, \Delta \Rightarrow x(y).P :: x : \forall y{:}\tau.A}\ \forall R$$

- Left rule: matching use of service

$$\frac{\Psi \vdash M : \tau \quad \Psi \,;\, \Gamma \,;\, \Delta', \quad A\{M/y\} \Rightarrow \qquad C}{\Psi \,;\, \Gamma \,;\, \Delta', \quad \forall y{:}\tau.A \Rightarrow \qquad C}\ \forall L$$

- Proof reduction yields

$$\longrightarrow$$

Term passing: input ($\forall y{:}\tau.A$)

- $P :: x : \forall y{:}\tau A$
  - P inputs an $M : \tau$ along x and then behaves as $A\{M/x\}$
- Right rule: offer of service

$$\frac{\Psi, y{:}\tau \; ; \Gamma \; ; \Delta \Rightarrow P :: x : A}{\Psi \; ; \Gamma \; ; \Delta \Rightarrow x(y).P :: x : \forall y{:}\tau.A} \; \forall R$$

- Left rule: matching use of service

$$\frac{\Psi \vdash M : \tau \quad \Psi \; ; \Gamma \; ; \Delta', x{:}A\{M/y\} \Rightarrow \qquad z : C}{\Psi \; ; \Gamma \; ; \Delta', x{:}\forall y{:}\tau.A \Rightarrow \qquad z : C} \; \forall L$$

- Proof reduction yields

$$\longrightarrow$$

Term passing: input ($\forall y{:}\tau.A$)

- $P :: x : \forall y{:}\tau A$
  - P inputs an $M : \tau$ along x and then behaves as $A\{M/x\}$
- Right rule: offer of service

$$\frac{\Psi, y{:}\tau \,;\, \Gamma \,;\, \Delta \Rightarrow P :: x : A}{\Psi \,;\, \Gamma \,;\, \Delta \Rightarrow x(y).P :: x : \forall y{:}\tau.A}\ \forall R$$

- Left rule: matching use of service

$$\frac{\Psi \vdash M : \tau \quad \Psi \,;\, \Gamma \,;\, \Delta', x{:}A\{M/y\} \Rightarrow Q :: z : C}{\Psi \,;\, \Gamma \,;\, \Delta', x{:}\forall y{:}\tau.A \Rightarrow x\langle M\rangle.Q :: z : C}\ \forall L$$

- Proof reduction yields

$$\longrightarrow$$

Term passing: input ($\forall y{:}\tau.A$)

- $P :: x : \forall y{:}\tau A$
  - P inputs an $M : \tau$ along x and then behaves as $A\{M/x\}$
- Right rule: offer of service

$$\frac{\Psi, y{:}\tau \; ; \Gamma \; ; \Delta \Rightarrow P :: x : A}{\Psi \; ; \Gamma \; ; \Delta \Rightarrow x(y).P :: x : \forall y{:}\tau.A} \; \forall R$$

- Left rule: matching use of service

$$\frac{\Psi \vdash M : \tau \quad \Psi \; ; \Gamma \; ; \Delta', x{:}A\{M/y\} \Rightarrow Q :: z : C}{\Psi \; ; \Gamma \; ; \Delta', x{:}\forall y{:}\tau.A \Rightarrow x\langle M\rangle.Q :: z : C} \; \forall L$$

- Proof reduction yields

$$(\boldsymbol{\nu} x)(x(y).P \mid x\langle M\rangle.Q) \longrightarrow$$

Term passing: input ($\forall y{:}\tau.A$)

- $P :: x : \forall y{:}\tau A$
  - P inputs an $M : \tau$ along x and then behaves as $A\{M/x\}$
- Right rule: offer of service

$$\frac{\Psi, y{:}\tau \; ; \Gamma \; ; \Delta \Rightarrow P :: x : A}{\Psi \; ; \Gamma \; ; \Delta \Rightarrow x(y).P :: x : \forall y{:}\tau.A} \; \forall R$$

- Left rule: matching use of service

$$\frac{\Psi \vdash M : \tau \quad \Psi \; ; \Gamma \; ; \Delta', x{:}A\{M/y\} \Rightarrow Q :: z : C}{\Psi \; ; \Gamma \; ; \Delta', x{:}\forall y{:}\tau.A \Rightarrow x\langle M \rangle.Q :: z : C} \; \forall L$$

- Proof reduction yields

$$(\boldsymbol{\nu} x)(x(y).P \mid x\langle M \rangle.Q) \longrightarrow (\boldsymbol{\nu} x)(P\{M/y\} \mid Q)$$

Term passing: other connectives

- Quantified proposition as dependent session types

$x : \forall y{:}\tau.A$	Input an $M : A$ along x and behave as $A\{M/y\}$
$x : \$\tau \multimap A$	Input an $M : A$ along x and behave as A
$x : \exists y{:}\tau.A$	Output an $M : A$ along x and behave as $A\{M/y\}$
$x : \$\tau \otimes A$	Output an $M : A$ along x and behave as A

- $\$\tau \multimap A$ as shorthand for $\forall y{:}\tau.A$ if y not free in A
- $\$\tau \otimes A$ as shorthand for $\exists y{:}\tau.A$ if y not free in A
- We will omit the '\$' for readability

Examples, carrying proofs

- PDF indexing service

$$\begin{aligned} \mathsf{index}_1 &: !(\mathsf{file} \multimap \mathsf{file} \otimes \mathbf{1}) \\ \mathsf{index}_2 &: !(\forall f{:}\mathsf{file}.\, \mathsf{pdf}(f) \multimap \exists g{:}\mathsf{file}.\, \mathsf{pdf}(g) \otimes \mathbf{1}) \end{aligned}$$

  Persistently offer to input a file f, a proof that f is in PDF format, then output a PDF file g, and a proof that g is in PDF format and terminate the session.

- Persistent file storage

$$\begin{aligned} \mathsf{store}_1 &: !(\mathsf{file} \multimap !(\mathsf{file} \otimes \mathbf{1})) \\ \mathsf{store}_2 &: !(\forall f{:}\mathsf{file}.\, !\exists g{:}\mathsf{file}.\, g \doteq f \otimes \mathbf{1}) \end{aligned}$$

  Persistently offer to input a file, then output a persistent channel for retrieving this file and a proof that the two are equal.

Outline

1. Session types for π-calculus
2. Dependent session types
3. Proof irrelevance
4. Affirmation and digital signatures
5. Conclusion

Proof irrelevance

- In many examples, we want to know that proofs exist, but we do not want to transmit them
  - We can easily check pdf(g) when using the indexing service
  - The proof of $g \doteq f$ (by reflexivity) would not be informative
- Use proof irrelevance in type theory
- $M : [\tau]$ — M is a term of type τ that is computationally irrelevant

Proof irrelevance: rules

- Introduction and elimination

$$\frac{\Psi^{\oplus} \vdash M : \tau}{\Psi \vdash [M] : [\tau]}\ []I \qquad \frac{\Psi \vdash M : [\tau] \quad \Psi, x \div \tau \vdash N : \sigma}{\Psi \vdash \textbf{let}\ [x] = M\ \textbf{in}\ N : \sigma}\ []E$$

- $\Psi^{\oplus}$ promotes hypotheses $x \div \tau$ to $x{:}\tau$
- In examples, may use pattern matching instead of **let**
- By agreement, terms $[M]$ will be erased before transmission
- Typing guarantees this can be done consistently

Examples with proof irrelevance

- Mark proofs as computationally irrelevant
- PDF indexing service

$$\begin{aligned} \mathsf{index}_2 &: !(\forall f{:}\mathsf{file}.\, \mathsf{pdf}(f) \multimap \exists g{:}\mathsf{file}.\, \mathsf{pdf}(g) \otimes \mathbf{1}) \\ \mathsf{index}_3 &: !(\forall f{:}\mathsf{file}.\, [\mathsf{pdf}(f)] \multimap \exists g{:}\mathsf{file}.\, [\mathsf{pdf}(g)] \otimes \mathbf{1}) \end{aligned}$$

- Persistent file storage

$$\begin{aligned} \mathsf{store}_2 &: !(\forall f{:}\mathsf{file}.\, !\exists g{:}\mathsf{file}.\, g \doteq f \otimes \mathbf{1}) \\ \mathsf{store}_3 &: !(\forall f{:}\mathsf{file}.\, !\exists g{:}\mathsf{file}.\, [g \doteq f] \otimes \mathbf{1}) \end{aligned}$$

- After erasure, communication can be optimized further

Examples: affirming the existence of proofs

- In the PDF indexing example, we may want to have some evidence that g and f agree.

$$\mathrm{index}_4 \;:\; !(\forall f{:}\mathrm{file}.\,[\mathrm{pdf}(f)] \multimap \exists g{:}\mathrm{file}.\,[\mathrm{pdf}(g)] \otimes [\mathrm{agree}(g,f)] \otimes \mathbf{1})$$

  agree(g, f) if g and f differ at most in the index
- Since no proof is transmitted, client may require indexer X's explicit affirmation ($=$ digital signature)!
- Similarly, in the persistent file storage example

Outline

1. Session types for π-calculus
2. Dependent session types
3. Proof irrelevance
4. Affirmation and digital signatures
5. Conclusion

Affirmation

- Judgment $M :_K \tau$
  - Principal K affirms property τ due to evidence M.

$$\frac{\Psi \vdash M : \tau}{\Psi \vdash \langle M{:}\tau \rangle_K :_K \tau} \text{ (affirms)}$$

- Internalize judgment as proposition $\Diamond_K \tau$

$$\frac{\Psi \vdash M :_K \tau}{\Psi \vdash M : \Diamond_K \tau} \Diamond I \qquad \frac{\Psi \vdash M : \Diamond_K \tau \quad \Psi, x{:}\tau \vdash N :_K \sigma}{\Psi \vdash \textbf{let } \langle x{:}\tau \rangle_K = M \textbf{ in } N :_K \sigma} \Diamond E$$

- Note same principal K in premises and conclusion of $\Diamond E$
- $\langle M{:}\tau \rangle_K$ can be realized by K's signature on $M{:}\tau$
- Assume some public key infrastructure
- $\Diamond_K$ is a K-indexed family of strong monads

Examples: affirmations

- PDF indexing service, with indexer X

$$\text{index}_5 : !(\forall f{:}\text{file}.\ [\text{pdf}(f)] \multimap \exists g{:}\text{file}.\ [\text{pdf}(g)] \otimes \lozenge_X[\text{agree}(g, f)] \otimes \mathbf{1})$$

- Persistent file storage, with file system Y

$$\text{store}_4 : !(\forall f{:}\text{file}.\ !\exists g{:}\text{file}.\ \lozenge_Y[g \doteq f] \otimes \mathbf{1})$$

- Idiom $\lozenge_K[\tau]$ may transmit
  - $\langle[\,]{:}\tau\rangle_K$, a certificate, digitally signed by K affirming τ
  - Some proof that $[\tau]$ follows from affirmations by K, according to the laws of $\lozenge_K$

Example: a PDF compression service

- A PDF compression service, with compressor C

$$\text{compress} : !(\forall f{:}\text{file}.\, [\text{pdf}(f)] \\ \multimap \exists g{:}\text{file}.\, [\text{pdf}(g)] \otimes \Diamond_C[\text{approx}(g, f)] \otimes \mathbf{1})$$

- A consolidator service: indexing and compression

$$\text{ixc} : !(\forall f{:}\text{file}.\, [\text{pdf}(f)] \\ \multimap \exists g{:}\text{file}.\, [\text{pdf}(g)] \otimes \Diamond_X \Diamond_C[\text{approx}(g, f)] \otimes \mathbf{1})$$

- Have to trust both X and C!

Example: consolidator implementation

- Specification

$$\text{ixc} : !(\forall f{:}\text{file}.\ [\text{pdf}(f)] \multimap \exists g{:}\text{file}.\ [\text{pdf}(g)] \otimes \Diamond_X \Diamond_C [\text{approx}(g, f)] \otimes \mathbf{1})$$

- Implementation

$$\begin{array}{l} \text{consolidator} = \\ !\text{ixc}(a).a(f_1).a([p_1]). \\ \quad (\boldsymbol{\nu} b)index\langle b\rangle.b\langle f_1\rangle.b\langle [p_1]\rangle.b(f_2).b([p_2]).b(q_2). \\ \quad\quad (\boldsymbol{\nu} c)\text{compress}\langle c\rangle.c\langle f_2\rangle.c\langle [p_2]\rangle.c(f_3).c([p_3]).c(q_3). \\ \quad\quad\quad a\langle f_3\rangle.a\langle [p_3]\rangle.a\langle \text{comb}\ q_2\ q_3\rangle.\mathbf{0} \end{array}$$

- Certificate types

$$\begin{array}{rcl} q_2 & : & \Diamond_X [\text{agree}(f_2, f_1)] \\ q_3 & : & \Diamond_C [\text{approx}(f_3, f_2)] \\ \text{comb}\ q_2\ q_3 & : & \Diamond_C \Diamond_X [\text{approx}(f_3, f_1)] \end{array}$$

Certificate combination

- Certificate types

$$\begin{array}{rcl} q_2 & : & \Diamond_X[\text{agree}(f_2, f_1)] \\ q_3 & : & \Diamond_C[\text{approx}(f_3, f_2)] \\ \text{comb}\ q_2\ q_3 & : & \Diamond_C\Diamond_X[\text{approx}(f_3, f_1)] \end{array}$$

- Proof

$$\text{ida} : \text{agree}(f_2, f_1) \to \text{approx}(f_2, f_1)$$
$$\text{tra} : \text{approx}(f_3, f_2) \to \text{approx}(f_2, f_1) \to \text{approx}(f_3, f_1)$$

$$\begin{array}{l} \text{comb}\ q_2\ q_3 = \\ \quad \textbf{let}\ \langle[q_3']{:}[\text{approx}(f_3, f_2)]\rangle_C = q_3\ \textbf{in} \\ \quad\quad \langle\textbf{let}\ \langle[q_2']{:}[\text{agree}(f_2, f_1)]\rangle_X = q_2\ \textbf{in} \\ \quad\quad\quad \langle[\text{tra}\ q_3'\ (\text{ida}\ q_2')]{:}\_\rangle_X{:}\_\rangle_C \end{array}$$

Trust axioms

- Affirmations track aspects of provenance and info. flow
  - "Diamonds are forever"
  - In general, $\nvdash \Diamond_K \tau \to \tau$
  - Need declassification
- Trust axioms
  - For specific types τ and principals K:

$$\mathsf{trust}_{K,\tau} : \Diamond_K \tau \to \tau$$

  - Implementable, in general, by stripping signature
- Omitted proofs $[\tau]$ cannot be recovered, in general

$\nvdash [\tau] \to \tau$	not implementable, in general
$\nvdash \Diamond_K [\tau] \to \tau$	not implementable, in general

Example: mobile code

- For sensitive documents we want to run indexing locally
- Specification

$$\begin{aligned}\mathsf{index}_6 : !(&\Diamond_X(\Pi f{:}\mathsf{file}.\,[\mathsf{pdf}(f)]\\ &\quad\to \Sigma g{:}\mathsf{file}.\,[\mathsf{pdf}(g)] \times [\mathsf{agree}(g, f)]) \otimes \mathbf{1})\end{aligned}$$

- Service persistently offers a function for indexing
- Cannot leak information since only process layer can communicate

Example: electronic commerce

- Signed certificates may have external meaning
- Signed certificates may flow in both directions

$$
\begin{aligned}
\mathrm{index}_7^u : \,!(&\Diamond_u[\mathrm{pay}(u, X, 1)] \\
&\multimap (\forall f{:}\mathrm{file}.\, [\mathrm{pdf}(f)] \\
&\quad \multimap \exists g{:}\mathrm{file}.\, [\mathrm{pdf}(g)] \otimes \Diamond_X[\mathrm{agree}(g, f)] \otimes \mathbf{1}))
\end{aligned}
$$

- Need to make principals more explicit?
- Some experience in proof-carrying authorization

Outline

Summary

- A Curry-Howard isomorphism
  - Linear propositions as session types
    $A \multimap B$ (input), $A \otimes B$ (output), $A \,\&\, B$ (external choice)
    $A \oplus B$ (internal choice), $!A$ (replication)
  - Sequent proofs as π-calculus processes
    with a binary guarded choice and channel forwarding
  - Cut reduction as π-calculus reduction
- Term-passing extension with a type theory
  - $\forall x{:}\tau.A$ (term input), $\exists x{:}\tau.A$ (term output)
- Additional type theory constructs
  - $[\tau]$ for proof irrelevance (not transmitted)
  - $\Diamond_K \tau$ for affirmations (evidenced by digital signatures)

Assessment

- Strong basis in logic and type theory
  - Modular construction and extensibility
  - Integrated computation and reasoning
- Uniform logical integration
  - Proofs (implicit or explicit)
  - Affirmations (implicit or explicit signatures)
- Enable gradual integration of formal proofs in current practice based on digital signatures?

Current and Future Work

- Practical language design and implementation
- Explicit spatial distribution, principals, and authorization (with Jamie Morgenstern)
- Interaction with databases (with João Seco and OutSystems)
- Reasoning about processes (with Jorge Pérez and Henry DeYoung)
  - Observational equivalence via proof theory
  - Towards concurrent type theory

Summary

- A Curry-Howard isomorphism
  - Linear propositions as session types
    $A \multimap B$ (input), $A \otimes B$ (output), $A \,\&\, B$ (external choice)
    $A \oplus B$ (internal choice), $!A$ (replication)
  - Sequent proofs as π-calculus processes
    with a binary guarded choice and channel forwarding
  - Cut reduction as π-calculus reduction
- Term-passing extension with a type theory
  - $\forall x{:}\tau.A$ (term input), $\exists x{:}\tau.A$ (term output)
- Additional type theory constructs
  - $[\tau]$ for proof irrelevance (not transmitted)
  - $\Diamond_K \tau$ for affirmations (evidenced by digital signatures)