A Core Calculus for Equational Proofs of Cryptographic Protocols

Byoungho Son

SVLAB Semniar

2024.11.06

Literature

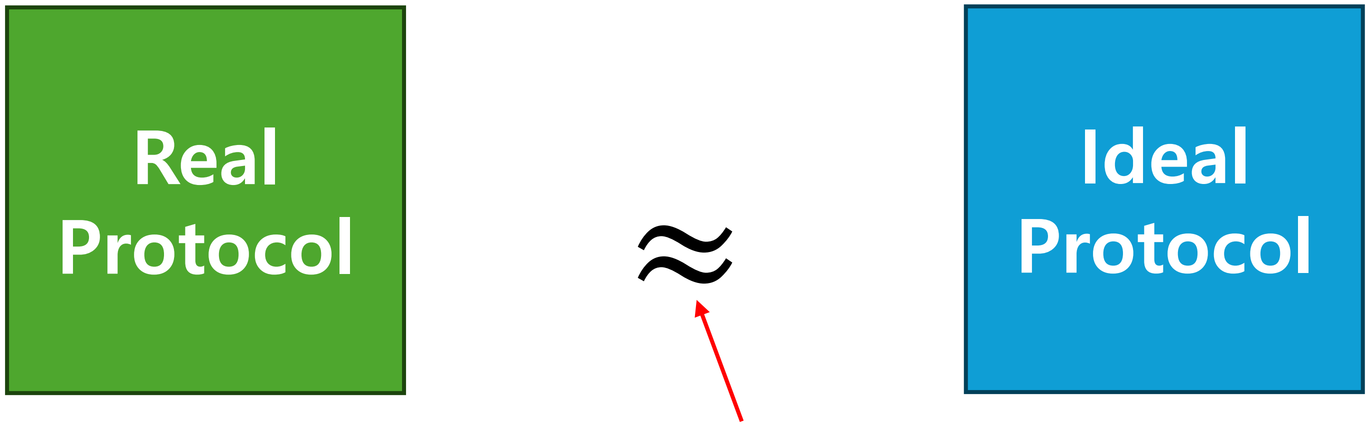
- Cryptographic Protocol Analysis
  - **Symbolic security**
    - Message: term
    - Attack: term rewriting
  - **Computational security** (Our Scope)
    - Message : bitstring
    - Attack: Polynomial-time Probabilistic Turing Machine (PPTM)

Literature

- Cryptographic Protocol Analysis
  - **Symbolic security**
    - Message: term
    - Attack: term rewriting
  - **Computational security** (Our Scope)
    - Message : bitstring
    - Attack: Polynomial-time Probabilistic Turing Machine (PPTM)
- Remark
  - Computational security *subsumes* symbolic security
    - Symbolic security models attackers by specifying what attackers can do
    - Computational security models attackers by specifying what attackers cannot do

Literature

- How to prove computational security?
  - Manual Proof by cryptographers > **Theorem Proving** > Model Checking
  - De facto standard for TP : **Universal Composability (UC)**



Computational Indistinguishability : A Polynomial-time TM cannot distinguish between the two

Universal Composability

- Pros: Composability
- Cons: Scalability

$$\pi = R_1 + H_1 \approx R_1 + H_1' = \cdots = R_k + H_k \approx R_k + H_k' = \text{Ideal}$$

- $=$: *exact* equivalence (bisimulation)
- $\approx$: *approximate* equivalence
  - (computational indistinguishability assumption)

Contribution

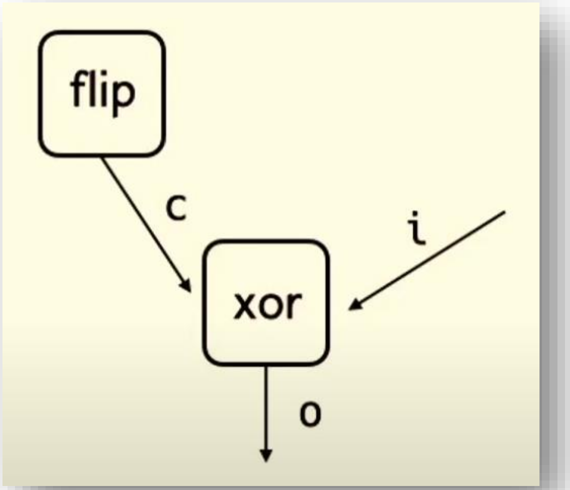
- IPDL (Interactive Probabilistic Dependency Logic)
  - Protocol description language
    - for distributed, interactive message-passing cryptographic protocols
  - Equational logic for $=$ & $\approx$
    - sound w.r.t computational model (i.e. $\vdash P \approx Q$ implies $\models P \approx Q$)
    - does not require explicit bisimulation

- This paper introduces IPDL & mechanizes it in Coq

IPDL : Basic Syntax

Channels	i, o, c		
Reactions	R, S	$::=$	$\text{ret } (e) \mid \text{samp } (D) \mid \text{read } c$
		$\mid$	$\text{if } e \text{ then } R_1 \text{ else } R_2 \mid x : \tau \leftarrow R;\ S$
Protocols	P, Q	$::=$	$o \coloneqq R \mid P \,\|\, Q \mid \text{new } o : \tau \text{ in } P$

```
P[i, o] :=
  new c in
    c ::= samp flip()
    ||
    o ::=
      x <- read c;
      y <- read i;
      ret (x xor y)
```



IPDL : Hello World!

Goal : Prove the Exact Equivalence

```
P[i, o] :=
  new c in
    c ::= samp flip()
    ||
    o ::=
      x <- read c;
      y <- read i;
      ret (x xor y)
```

Real Protocol

=

```
_ <- read i;
samp flip()
```

Ideal Protocol

$$\boxed{\Delta \vdash P = Q : I \longrightarrow O}$$

$$\frac{\Delta \vdash P : I \longrightarrow O}{\Delta \vdash P = P : I \longrightarrow O}\ \textsc{refl}$$

$$\frac{\Delta \vdash P_1 = P_2 : I \longrightarrow O}{\Delta \vdash P_2 = P_1 : I \longrightarrow O}\ \textsc{sym}$$

$$\frac{\Delta \vdash P_1 = P_2 : I \longrightarrow O \qquad \Delta \vdash P_2 = P_3 : I \longrightarrow O}{\Delta \vdash P_1 = P_3 : I \longrightarrow O}\ \textsc{trans}$$

$$\frac{\vdash \theta : \Delta_1 \longrightarrow \Delta_2 \qquad \Delta_1 \vdash P = Q : I \longrightarrow O}{\Delta_2 \vdash \theta^\star(P) = \theta^\star(Q) : \theta^\star(I) \longrightarrow \theta^\star(O)}\ \textsc{embed}$$

$$\frac{(\Delta \vdash P = Q : I \longrightarrow O) \in \mathbb{T}_=}{\Delta \vdash P = Q : I \longrightarrow O}\ \textsc{axiom}$$

$$\frac{o : \tau \in \Delta \qquad \Delta;\, \cdot \vdash R = R' : I \cup \{o\} \longrightarrow \tau}{\Delta \vdash (o \coloneqq R) = (o \coloneqq R') : I \longrightarrow \{o\}}\ \textsc{cong-react}$$

$$\frac{i \notin I, O \qquad \Delta \vdash P = Q : I \longrightarrow O}{\Delta \vdash P = Q : I \cup \{i\} \longrightarrow O}\ \textsc{input-unused}$$

$$\frac{\Delta \vdash P = P' : I \cup O_2 \longrightarrow O_1 \qquad \Delta \vdash Q : I \cup O_1 \longrightarrow O_2}{\Delta \vdash P \mid\mid Q = P' \mid\mid Q : I \longrightarrow O_1 \cup O_2}\ \textsc{cong-comp-left}$$

• • •

IPDL : Hello World!

$$\frac{o : B \qquad \Delta;\ \cdot \vdash R : I \cup \{o\} \to \tau \qquad \Delta;\ x : \tau \vdash S : I \cup \{o\} \to B}{\Delta \vdash (\text{new } c : \tau \text{ in } o \coloneqq x : \tau \leftarrow \text{read } c; S \,||\, c \coloneqq R) = (o \coloneqq x : \tau \leftarrow R; S) : I \to \{o\}} \text{ FOLD-BIND}$$

```
P[i, o] :=
  new c in
    c ::= samp flip()
    ||
    o ::=
      x <- read c;
      y <- read i;
      ret (x xor y)
```

Real Protocol

only sound when c is used linearly!

=

```
P[i, o] :=
    o ::=
      x <- samp flip();
      y <- read i;
      ret (x xor y)
```

IPDL : Hello World!

reactions form a commutative monad

```
x <- samp flip();
y <- read i;
ret (x xor y)
```

=

```
y <- read i;
x <- samp flip();
ret (x xor y)
```

IPDL : Hello World!

axiom for exact equivalence : flip() = flip() xor y

```
y <- read i;
x <- samp flip();
ret (x xor y)
```

=

```
y <- read i;
x <- samp flip();
ret ((x xor y) xor y)
```

IPDL : Hello World!

axiom for exact equivalence :

1) y xor y = 0
2) x xor 0 = x

```
y <- read i;
x <- samp flip();
ret ((x xor y) xor y)
```

=

```
_ <- read i;
samp flip()
```

Ideal Protocol

Computational Security, Intuitively

- How to <u>define</u> the computational security $\mathrm{P} \approx Q : I \rightarrow O$?
- by Security Game b/w adversary & protocol:
  - for each round:
    - adversary gives input
    - protocol returns output
  - adversary guesses P / Q

- If $\mathrm{P} = Q : I \rightarrow O$, Pr(adv. wins) = 0.5, for any adv.
- If $\mathrm{P} \approx Q : I \rightarrow O$, Pr(adv. wins) = 0.5 + ϵ, for polynomial adv.

Modeling Computational Adversaries

Definition 4.5 (Δ-Distinguisher). Given an interpretation $\mathcal{I}$, A $(\mathcal{I}, \Delta, I, O)$-distinguisher $\mathcal{A}$ is a triple of probabilistic algorithms $(\mathcal{A}_{\text{step}}, \mathcal{A}_{\text{out}}, \mathcal{A}_{\text{decide}})$ where:

- $\mathcal{A}_{\text{step}} : \{0,1\}^* \to \{0,1\}^* \times \text{Query}_{\mathcal{I},\Delta,I,O}$ takes input a state s (encoded as a bitstring), and returns a new state and a query;
- $\mathcal{A}_{\text{out}} : \{0,1\}^* \times (o : O) \times (1 + \{0,1\}^{[\![\Delta(o)]\!]^{\mathcal{I}}}) \to \{0,1\}^*$ takes a state s, a channel o, an optional value v for o, and returns a new state; and
- $\mathcal{A}_{\text{decide}} : \{0,1\}^* \to \{0,1\}$ takes a state and returns a single bit.

Distinguishers and Interactions. Let $\mathcal{I}$ be an interpretation for Σ. Then, given channel sets I, O for channel context Δ, we define the set $\text{Query}_{\mathcal{I},\Delta,I,O}$ to be:

$$\text{Query}_{\mathcal{I},\Delta,I,O} := \{\text{Input}(i,v) \mid i \in I, v \in \{0,1\}^{[\![\Delta(i)]\!]^{\mathcal{I}}}\} \cup \{\text{Get}(o), o \in O\} \cup \{\text{Step}\}.$$

Algorithm $\mathcal{A}^k(P^{\mathcal{I}})$:

$s := \epsilon$

For k rounds:

  $(s', q) \leftarrow \mathcal{A}_{\mathsf{step}}(s)$ // update state

  $s := s'$

  If $q = \mathsf{Input}(i, v)$: // give input to P

    $P := P[\mathsf{read}\ i := \mathsf{ret}\ (v)]$

  If $q = \mathsf{Get}(o)$: // get output from P

    If $(o := v) \in P$ for some v : // output may or may not be available

      $s := \mathcal{A}_{\mathsf{out}}(s, o, \mathsf{Some}(v))$ // either way, update the state accordingly

    Else :

      $s := \mathcal{A}_{\mathsf{out}}(s, o, \mathsf{None})$

  $P \leftarrow \eta$, where $P \Downarrow_{\mathcal{I}} \eta$ // evaluate P as much as possible

return $\mathcal{A}_{\mathsf{decide}}(s)$

Fig. 9. Interaction of IPDL program $\Delta \vdash P : I \rightarrow O$ with k-bounded $(\mathcal{I}, \Delta, I, O)$ distinguisher $\mathcal{A}$.

Probabilistic Poly-time Adversary

Definition 4.6 (k-Bounded Distinguisher). A $(\mathcal{I}, \Delta, I, O)$-distinguisher is k-bounded when its algorithms $(\mathcal{A}_{\text{step}}, \mathcal{A}_{\text{out}}, \mathcal{A}_{\text{decide}})$ all run in at most k time steps.

Definition 4.8 (PPT Distinguishers). Let $\{\mathcal{I}_\lambda\}$ be a family of interpretations for Σ, indexed by natural numbers λ. Additionally, let $\{\Delta_\lambda, I_\lambda, O_\lambda\}_\lambda$ be a family of channel contexts Δ_λ and channel sets for Δ_λ. Then a *PPT distinguisher* for $\{\Delta_\lambda, I_\lambda, O_\lambda\}$ is a family $\{\mathcal{A}_\lambda\}_\lambda$ such that $\mathcal{A}_\lambda$ is a $(\mathcal{I}_\lambda, \Delta_\lambda, I_\lambda, O_\lambda)$-distinguisher, along with a polynomial p such that $\mathcal{A}_\lambda$ is $p(\lambda)$-bounded for all λ.

Computational Security, Formally

Definition 4.11 (Approximate Equivalence). Let $\Delta_\lambda \vdash P_\lambda : I_\lambda \to O_\lambda$ and $\Delta_\lambda \vdash Q_\lambda : I_\lambda \to O_\lambda$ be two families of IPDL protocols with identical typing judgments. Then, we say that P_λ and Q_λ are *indistinguishable* under PPT interpretation, written $\mathcal{I}_\lambda; \Delta_\lambda \vDash P_\lambda \approx_\lambda Q_\lambda : I_\lambda \to O_\lambda$, when: $|\Delta_\lambda|$ is bounded by a polynomial in λ; and for any PPT family of program contexts $\{C_\lambda : (\Delta_\lambda \vdash I_\lambda \to O_\lambda) \to (\Delta'_\lambda \vdash I'_\lambda \to O'_\lambda)\}$, and for all PPT families of distinguishers $\{\mathcal{A}_\lambda\}$ for $\{\Delta'_\lambda, I_\lambda, O_\lambda\}$ bounded by $p(\cdot)$, there exists a negligible function ε such that

$$|\Pr[\mathcal{A}_\lambda^{p(\lambda)}(C_\lambda(P_\lambda)^{\mathcal{I}_\lambda})] - \Pr[\mathcal{A}_\lambda^{p(\lambda)}(C_\lambda(Q_\lambda)^{\mathcal{I}_\lambda})]| \leq \varepsilon(\lambda).$$

Recall that a negligible function $\varepsilon : \mathbb{N} \to \mathbb{Q}$ is one that is eventually smaller than the inverse of any polynomial: $\forall K, \exists N, \forall n > N, \varepsilon(n) < \frac{1}{n^K}$

$$\boxed{\Delta \vdash P_1 \approx_\lambda^{(k,l)} P_2 : I \longrightarrow O}$$

$$\frac{\Delta \vdash_{\Sigma,\mathbb{T}} P = Q : I \longrightarrow O}{\Delta \vdash P \approx_\lambda^{(0,0)} Q : I \longrightarrow O}\ \textsc{strict}$$

$$\frac{\Delta \vdash P \approx_\lambda^{(k_1,l_1)} Q : I \longrightarrow O \qquad k_1 \leq k_2 \qquad l_1 \leq l_2}{\Delta \vdash P \approx_\lambda^{(k_2,l_2)} Q : I \longrightarrow O}\ \textsc{subsume}$$

$$\frac{\Delta \vdash P_1 \approx_\lambda^{(k,l)} P_2 : I \longrightarrow O}{\Delta \vdash P_2 \approx_\lambda^{(k,l)} P_1 : I \longrightarrow O}\ \textsc{sym}$$

$$\frac{\Delta \vdash P_1 \approx_\lambda^{(k_1,l_1)} P_2 : I \longrightarrow O \qquad \Delta \vdash P_2 \approx_\lambda^{(k_2,l_2)} P_3 : I \longrightarrow O}{\Delta \vdash P_1 \approx_\lambda^{(k_1+k_2,\max(l_1,l_2))} P_3 : I \longrightarrow O}\ \textsc{trans}$$

$$\frac{\theta : \Delta_1 \longrightarrow \Delta_2 \qquad \Delta_1 \vdash P \approx_\lambda^{(k,l)} Q : I \longrightarrow O}{\Delta_2 \vdash \theta^\star(P) \approx_\lambda^{(k,l)} \theta^\star(Q) : \theta^\star(I) \longrightarrow \theta^\star(O)}\ \textsc{embed}$$

$$\frac{\{\Delta_n \vdash P_n \approx_\lambda Q_n : I_n \longrightarrow O_n\} \in \mathbb{T}_\approx}{\Delta \vdash P_\lambda \approx_\lambda^{(1,0)} Q_\lambda : I_\lambda \longrightarrow O_\lambda}\ \textsc{axiom}$$

$$\frac{i \notin I \cup O \qquad \Delta \vdash P \approx_\lambda^{(k,l)} Q : I \longrightarrow O}{\Delta \vdash P \approx_\lambda^{(k,l)} Q : I \cup \{i\} \longrightarrow O}\ \textsc{input-unused}$$

• • •

Recall : Universal Composability

$$\pi = R_1 + H_1 \overset{<\epsilon}{\approx} R_1 + H_1' = \cdots = R_k + H_k \overset{<\epsilon}{\approx} R_k + H_k' = \text{Ideal}$$

- Using equational logic for $=$ & $\approx$, we deduce $\pi \approx Ideal$.
- Note that each $H_1 \approx H_2$ is an **axiom**, assuming computational indistinguishability defined previously, holds

Recall : Universal Composability

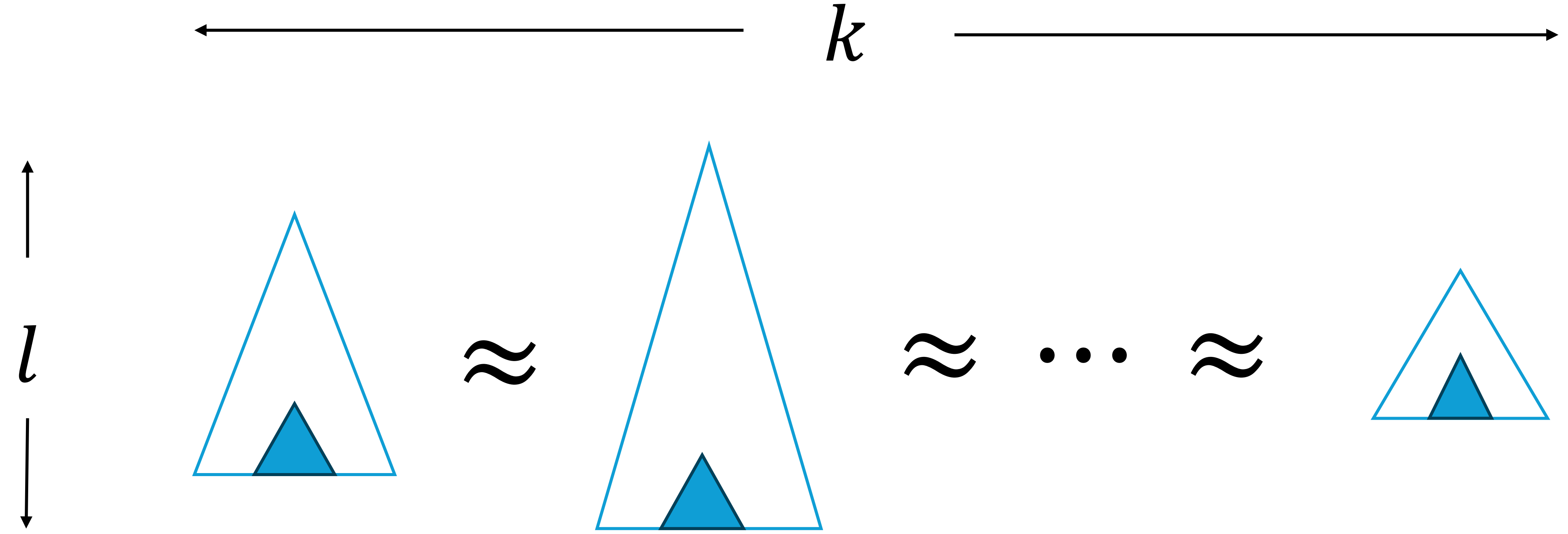
$$\pi = R_1 + H_1 \overset{<\epsilon}{\approx} R_1 + H_1' = \cdots = R_k + H_k \overset{<\epsilon}{\approx} R_k + H_k' = \text{Ideal}$$

- Using equational logic for $=$ & $\approx$, we deduce $\pi \approx Ideal$.
- Note that each $H_1 \approx H_2$ is an **axiom**, assuming computational indistinguishability defined previously, holds

- BUT!
- What if axioms are used **exponentially** many times in the proof?
- Then the equational logic is no longer sound! (why?)

Judgment for Approximate Equivalence

$$\Delta \vdash P_1 \approx_{\lambda}^{(k,l)} P_2 : I \to O$$



focus on how (k, l)' s are updated!

$$\boxed{\Delta \vdash P_1 \approx_\lambda^{(k,l)} P_2 : I \to O}$$

$$\frac{\Delta \vdash_{\Sigma,\mathbb{T}} P = Q : I \to O}{\Delta \vdash P \approx_\lambda^{(0,0)} Q : I \to O}\ \text{STRICT}$$

$$\frac{\Delta \vdash P \approx_\lambda^{(k_1,l_1)} Q : I \to O \qquad k_1 \le k_2 \qquad l_1 \le l_2}{\Delta \vdash P \approx_\lambda^{(k_2,l_2)} Q : I \to O}\ \text{SUBSUME}$$

$$\frac{\Delta \vdash P_1 \approx_\lambda^{(k,l)} P_2 : I \to O}{\Delta \vdash P_2 \approx_\lambda^{(k,l)} P_1 : I \to O}\ \text{SYM}$$

$$\frac{\Delta \vdash P_1 \approx_\lambda^{(k_1,l_1)} P_2 : I \to O \qquad \Delta \vdash P_2 \approx_\lambda^{(k_2,l_2)} P_3 : I \to O}{\Delta \vdash P_1 \approx_\lambda^{(k_1+k_2,\max(l_1,l_2))} P_3 : I \to O}\ \text{TRANS}$$

$$\frac{\theta : \Delta_1 \to \Delta_2 \qquad \Delta_1 \vdash P \approx_\lambda^{(k,l)} Q : I \to O}{\Delta_2 \vdash \theta^\star(P) \approx_\lambda^{(k,l)} \theta^\star(Q) : \theta^\star(I) \to \theta^\star(O)}\ \text{EMBED}$$

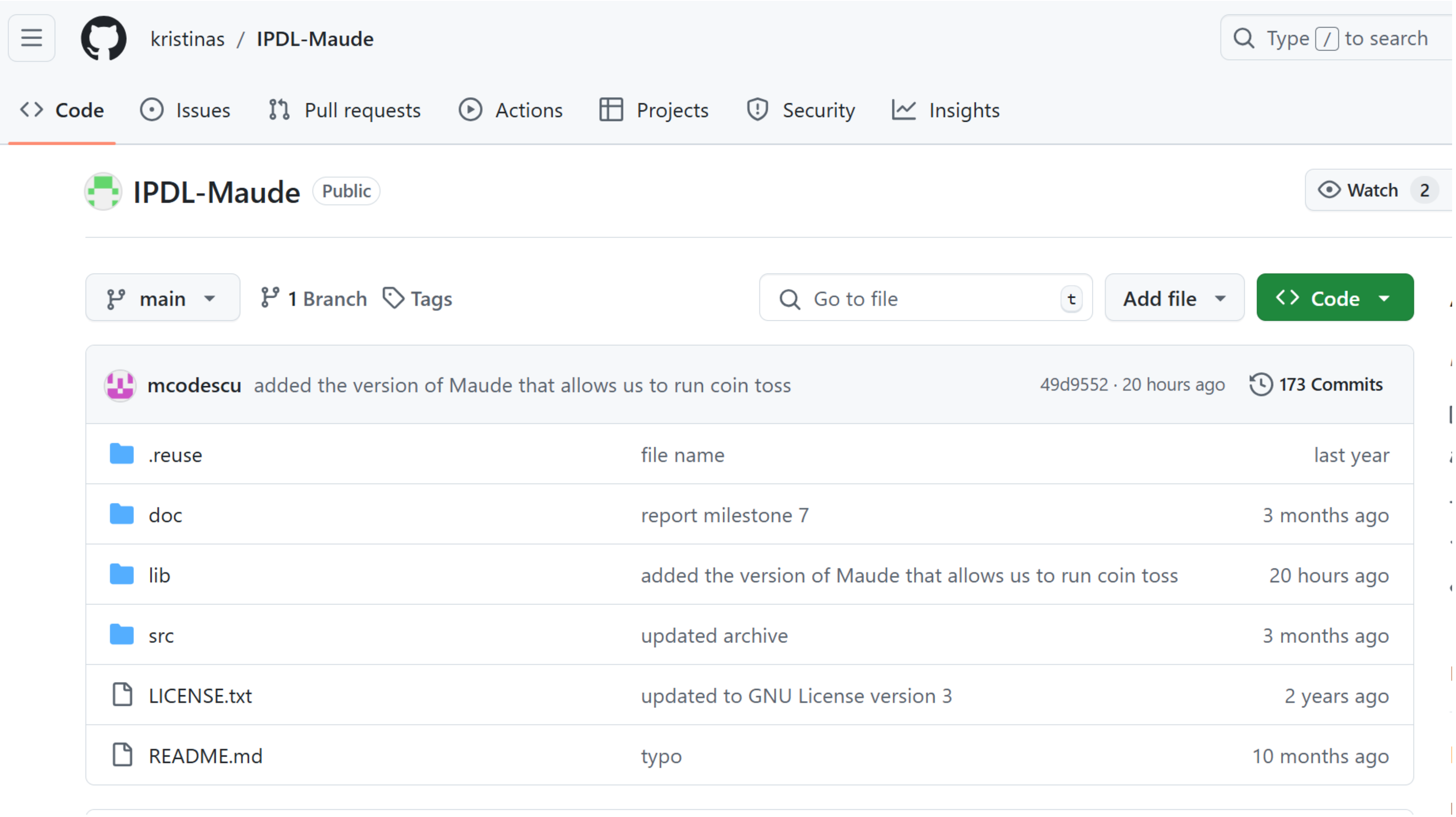
$$\frac{\{\Delta_n \vdash P_n \approx_\lambda Q_n : I_n \to O_n\} \in \mathbb{T}_\approx}{\Delta \vdash P_\lambda \approx_\lambda^{(1,0)} Q_\lambda : I_\lambda \to O_\lambda}\ \text{AXIOM}$$

$$\frac{i \notin I \cup O \qquad \Delta \vdash P \approx_\lambda^{(k,l)} Q : I \to O}{\Delta \vdash P \approx_\lambda^{(k,l)} Q : I \cup \{i\} \to O}\ \text{INPUT-UNUSED}$$

• • •

Soundness of IPDL

THEOREM 4.17 (SOUNDNESS THEOREM FOR THE APPROXIMATE EQUALITY OF IPDL PROTOCOLS). *Let Σ be an* IPDL *signature, and let $\mathbb{T}_=$ and $\mathbb{T}_\approx$ be sound exact and approximate theories with respect to a PPT interpretation $\{\mathcal{I}_\lambda\}$. If $\vdash \{\Delta_\lambda \vdash P_\lambda \approx_\lambda Q_\lambda : I_\lambda \to O_\lambda\}$, then $\mathcal{I}_\lambda; \Delta_\lambda \vDash P_\lambda \approx_\lambda Q_\lambda : I_\lambda \to O_\lambda$.*


kristinas / IPDL-Maude
Type / to search
Code
Issues
Pull requests
Actions
Projects
Security
Insights
IPDL-Maude
Public
Watch 2
main
1 Branch
Tags
Go to file
t
Add file
Code
mcodescu added the version of Maude that allows us to run coin toss
49d9552 · 20 hours ago
173 Commits
.reuse
file name
last year
doc
report milestone 7
3 months ago
lib
added the version of Maude that allows us to run coin toss
20 hours ago
src
updated archive
3 months ago
LICENSE.txt
updated to GNU License version 3
2 years ago
README.md
typo
10 months ago

손병호(컴퓨터공학과)
To: Mihai.Codescu@imar.ro

Tue 3/12/2024 4:24 PM

Dear Mihai Codescu,

Hello, I am Byoungho Son, and I study things related to rewriting logic.

Recently, I found out your project IPDL-Maude on github,

and I took a look at the original paper from POPL 2023.

I was wondering, what was the motivation for using Maude while there is already a Coq implementation of IPDL?

Would there be any advantage of Maude over Coq?

I'd appreciate if you could enlighten me:)

Best,

Byoungho

손병호(컴퓨터공학과)

To: Mihai.Codescu@imar.ro

Tue 3/12/2024 4:24 PM

Dear Mihai Codescu,

Hello, I am Byoungho Son, and I study things related to rewriting logic.

Recently, I found out your project IPDL-Maude on github,

and I too

I was wo

Would th

I'd appre

Best,

Byoungh

Mihai Codescu <mscodescu@gmail.com>

To: 손병호(컴퓨터공학과)

Cc: Kristina Sojakova <sojakova.kristina@gmail.com>

Wed 3/13/2024 3:58 AM

Dear Byoungho,

thank you for your interest.

I have chosen Maude because of my background and previous experience with it, and because I found it appropriate for the task of implementing a term rewriting system. In retrospect, I think it was a good idea. First, it provides a more natural way of specifying the typing and equality rules of IPDL - this is a subjective judgement from someone who isn't a type theorist, but we had this feedback from a cryptographer as well. Second, as you may have seen we are relying heavily on the strategy language for Maude. This allows us to write shorter proofs than in Coq, and we plan to implement a concrete syntax that will hide some Maude technicalities from the user, making the proofs even shorter. Finally, the proofs are not only shorter, but also faster. For a case study that took about 2000 lines of code and about 4 seconds in Coq, we have a Maude variant that takes about 160 lines of code and about 0.3 seconds. I know of a project that moved from Maude to Haskell for performance issues, but our implementation runs pretty fast for case studies of the size that we have formalized so far.

If you have further questions, don't hesitate to contact me.

Best regards,
Mihai