Programming Language Foundations for Distributed Systems

Ankush Das

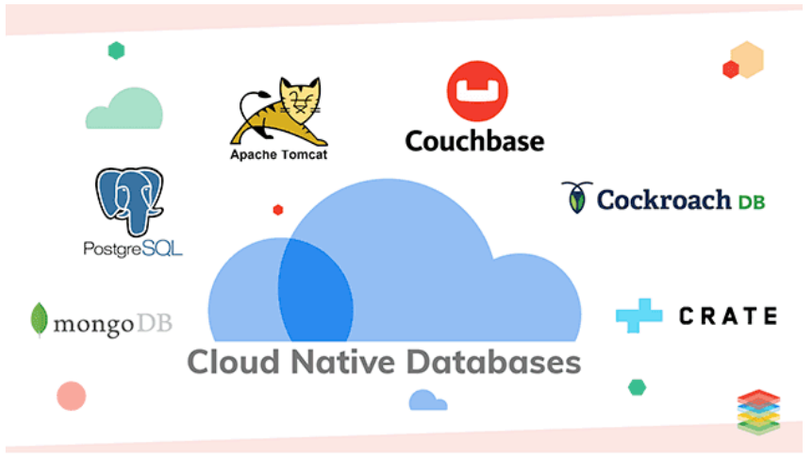
Applied Scientist, Amazon AWS

Imperial College, London

March 6, 2023

We are Living in a Distributed Age!

We are Living in a Distributed Age!



Programming in the Distributed Age is Hard!

Programming in the Distributed Age is Hard!

Software Bugs

DAILY NEWS

Blackout hits New York City and the Northeast in 2003

A software bug known as a race condition

Programming in the Distributed Age is Hard!

Software Bugs

DAILY NEWS

Blackout hits New York City and the Northeast in 2003

A software bug known as a race condition

Performance Defects

The Washington Post
Democracy Dies in Darkness

HealthCare.gov

The System is down at the moment.

We're working to resolve the issue as soon as possible. Please try again later.

Programming in the Distributed Age is Hard!

Software Bugs

DAILY NEWS

Blackout hits New York City and the Northeast in 2003

A software bug known as a race condition

Performance Defects

The Washington Post
Democracy Dies in Darkness

HealthCare.gov

The System is down at the moment.

We're working to resolve the issue as soon as possible. Please try again later.

techradar pro

Microsoft Azure bug left a bunch of cloud databases wide open

By Sead Fadilpašić published April 29, 2022

Domain-Specific Issues

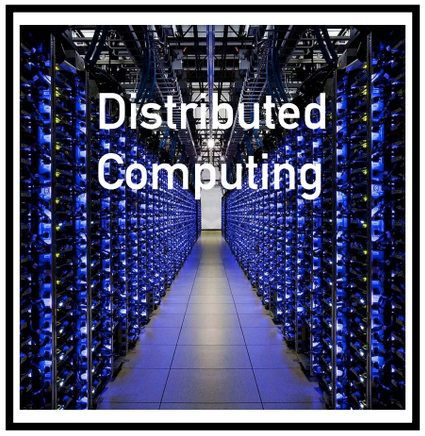
TechRepublic

Report: Software failure caused $1.7 trillion in financial losses in 2017

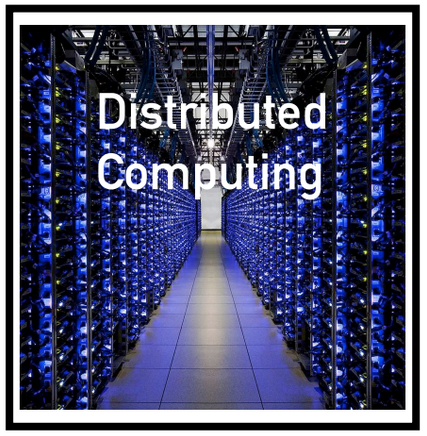
ACM DL DIGITAL LIBRARY

What Bugs Live in the Cloud? A Study of 3000+ Issues in Cloud Systems

How can PL Tools Help?



How can PL Tools Help?



The Software Model Checker
BLAST

CPA✓

CBMC

Model Checkers

Boogie: An Intermediate
Verification Language

Dafny

Z3

VIPER

Verification
Languages

LiquidHaskell

$\Gamma \vdash e : \tau$

Type Systems

The Coq
Proof Assistant

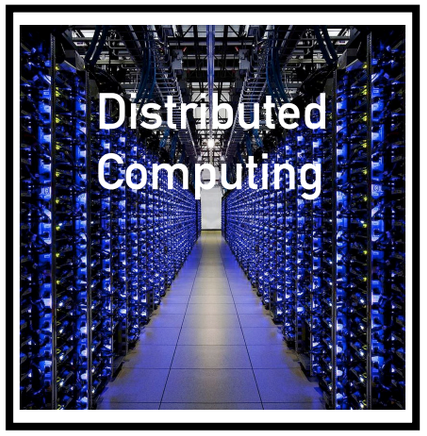
LEAN
THEOREM PROVER

Agda

Isabelle

Proof Assistants

How can PL Tools Help?



Concurrency Reasoning

The Software Model Checker
BLAST

CPA✓

CBMC

Boogie: An Intermediate
Verification Language

Dafny

Z3

VIPER

LiquidHaskell

$\Gamma \vdash e : \tau$

The Coq
Proof Assistant

THEOREM PROVER

Agda

Isabelle

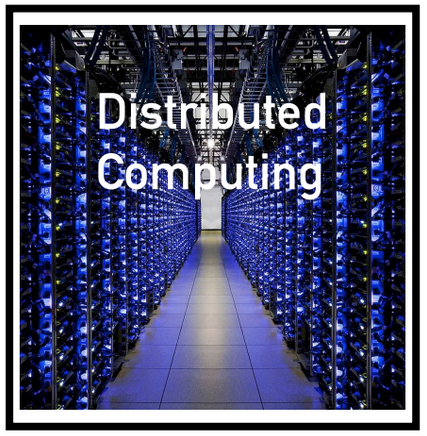
Model Checkers

Verification
Languages

Type Systems

Proof Assistants

How can PL Tools Help?



Concurrency Reasoning

Domain-Specific Support

The Software Model Checker
BLAST

CPA✓

CBMC

Boogie: An Intermediate
Verification Language

Dafny

Z3

VIPER

LiquidHaskell

$\Gamma \vdash e : \tau$

The Coq
Proof Assistant

THEOREM PROVER

Agda

Isabelle

Model Checkers

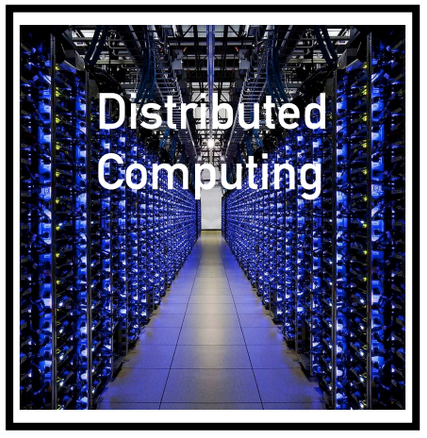
Verification Languages

Type Systems

Proof Assistants

How can PL Tools Help?

Protocol Enforcement & Asset Preservation



Concurrency Reasoning

Domain-Specific Support

The Software Model Checker
BLAST

CPA

CBMC

Boogie: An Intermediate
Verification Language

Dafny

Z3

VIPER

LiquidHaskell

$\Gamma \vdash e : \tau$

The Coq
Proof Assistant

THEOREM PROVER

Agda

Isabelle

Model Checkers

Verification Languages

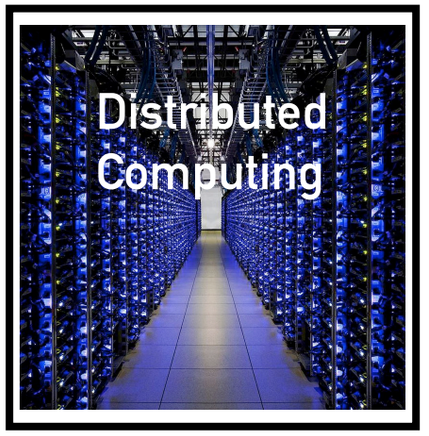
Type Systems

Proof Assistants

How can PL Tools Help?

Protocol Enforcement & Asset Preservation

Probabilistic Reasoning



Concurrency Reasoning

Domain-Specific Support

Model Checkers

The Software Model Checker
BLAST

CPA✓

CBMC

Verification Languages

Boogie: An Intermediate
Verification Language

Dafny

Z3

VIPER

Type Systems

LiquidHaskell

$\Gamma \vdash e : \tau$

Proof Assistants

The Coq
Proof Assistant

THEOREM PROVER

Agda

Isabelle

How can PL Tools Help?

Protocol Enforcement & Asset Preservation

Probabilistic Reasoning

Performance & Verification

Concurrency Reasoning

Domain-Specific Support

The Software Model Checker
BLAST

CPA✓

CBMC

Model Checkers

Boogie: An Intermediate
Verification Language

Dafny

Z3

VIPER

Verification Languages

LiquidHaskell

$\Gamma \vdash e : \tau$

Type Systems

The Coq
Proof Assistant

LEAN
THEOREM PROVER

Agda

Proof Assistants

My Research Focus

Protocol Enforcement & Asset Preservation

Probabilistic Reasoning

Performance & Verification

Concurrency Reasoning

Domain-Specific Support

The Software Model Checker
BLAST

CPA✓

CBMC

Model Checkers

Boogie: An Intermediate
Verification Language

Dafny

Z3

VIPER

Verification Languages

LiquidHaskell

$\Gamma \vdash e : \tau$

Type Systems

The Coq
Proof Assistant

LEAN
THEOREM PROVER

Agda

Isabelle

Proof Assistants

My Research Focus

Protocol Enforcement & Asset Preservation

Probabilistic Reasoning

Performance & Verification

Concurrency Reasoning

Domain-Specific Support

LiquidHaskell

$\Gamma \vdash e : \tau$

Type Systems

My Research Focus

Protocol Enforcement & Asset Preservation

Probabilistic Reasoning

Performance & Verification

Concurrency Reasoning ✓

Domain-Specific Support

Session Types

Type system for concurrent message-passing programs

My Research Focus

Protocol Enforcement & Asset Preservation

Probabilistic Reasoning

Performance & Verification

Concurrency Reasoning ✓

Domain-Specific Support ✓

Session Types

Type system for concurrent message-passing programs

My Research Focus

Protocol Enforcement & Asset Preservation

Probabilistic Reasoning

Performance & Verification

Session Types

Type system for concurrent message-passing programs

My Research Focus

Protocol Enforcement & Asset Preservation

Probabilistic Reasoning

Performance & Verification

Nomos

Resource-Aware Session Types

[LICS '18, SAS '20, CSF '21]

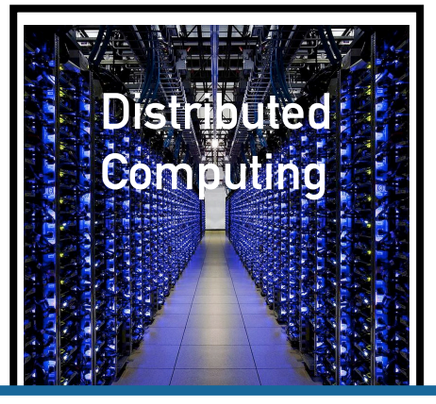
Session Types

Type system for concurrent message-passing programs

My Research Focus

Protocol Enforcement & Asset Preservation

Probabilistic Reasoning



Performance & Verification

Nomos

Resource-Aware Session Types

[LICS '18, SAS '20, CSF '21]

NomosPro

Probabilistic Session Types

[POPL '23]

Session Types

Type system for concurrent message-passing programs

My Research Focus

Protocol Enforcement & Asset Preservation

Probabilistic Reasoning

Performance & Verification

Nomos

Resource-Aware Session Types

[LICS '18, SAS '20, CSF '21]

NomosPro

Probabilistic Session Types

[POPL '23]

Rast

Refinement Session Types

[ICFP '18, FSCD '20, CONCUR '20]

Session Types

Type system for concurrent message-passing programs

My Research Focus

Protocol Enforcement & Asset Preservation

Probabilistic Reasoning

Performance & Verification

Nomos

Resource-Aware Session Types

[LICS '18, SAS '20, CSF '21]

NomosPro

Probabilistic Session Types

[POPL '23]

Rast

Refinement Session Types

[ICFP '18, FSCD '20, CONCUR '20]

Session Types

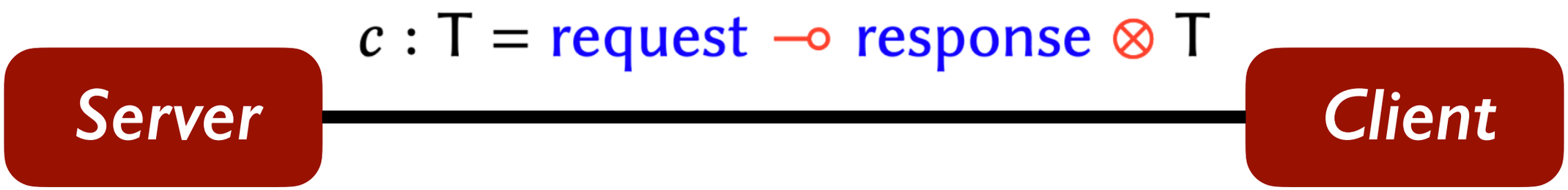
Type system for concurrent message-passing programs

What are Session Types?

- Session types are a system for concurrent message-passing programs
- Communication via typed bi-directional channels
- Session type governs the type and direction of message exchanges

What are Session Types?

- Session types are a system for concurrent message-passing programs
- Communication via typed bi-directional channels
- Session type governs the type and direction of message exchanges

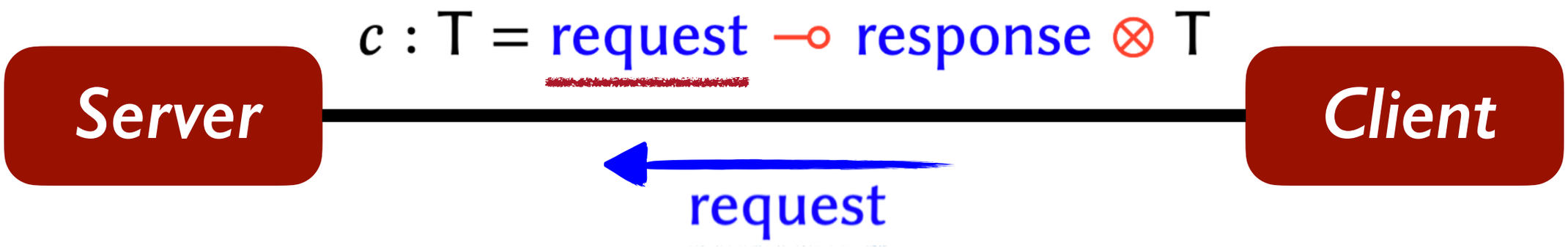


What are Session Types?

- Session types are a system for concurrent message-passing programs
- Communication via typed bi-directional channels
- Session type governs the type and direction of message exchanges

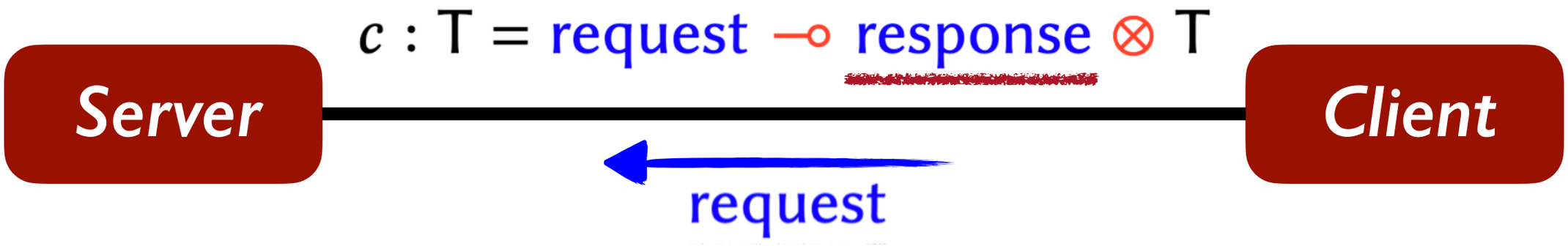
What are Session Types?

- Session types are a system for concurrent message-passing programs
- Communication via typed bi-directional channels
- Session type governs the type and direction of message exchanges



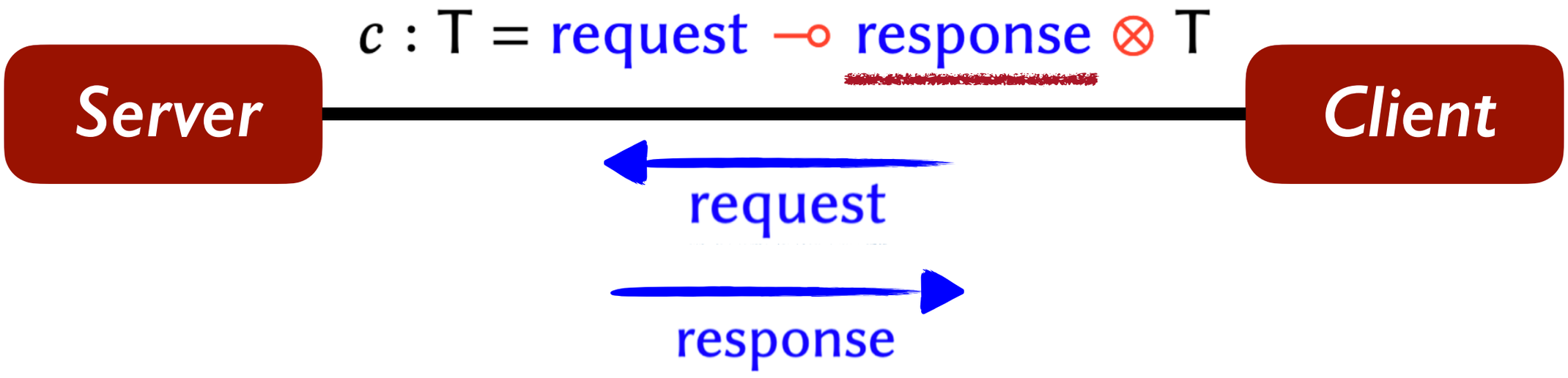
What are Session Types?

- Session types are a system for concurrent message-passing programs
- Communication via typed bi-directional channels
- Session type governs the type and direction of message exchanges



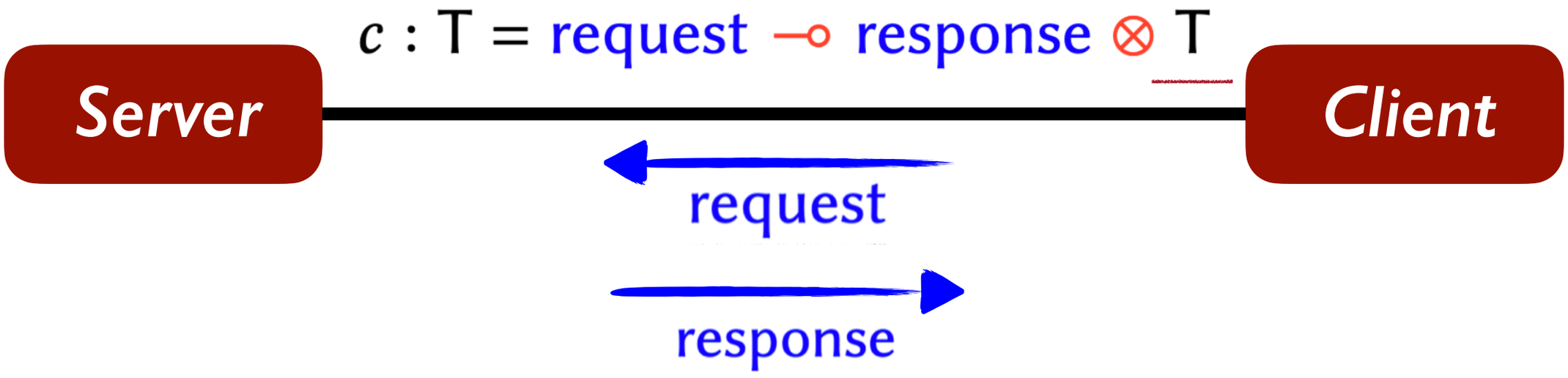
What are Session Types?

- Session types are a system for concurrent message-passing programs
- Communication via typed bi-directional channels
- Session type governs the type and direction of message exchanges



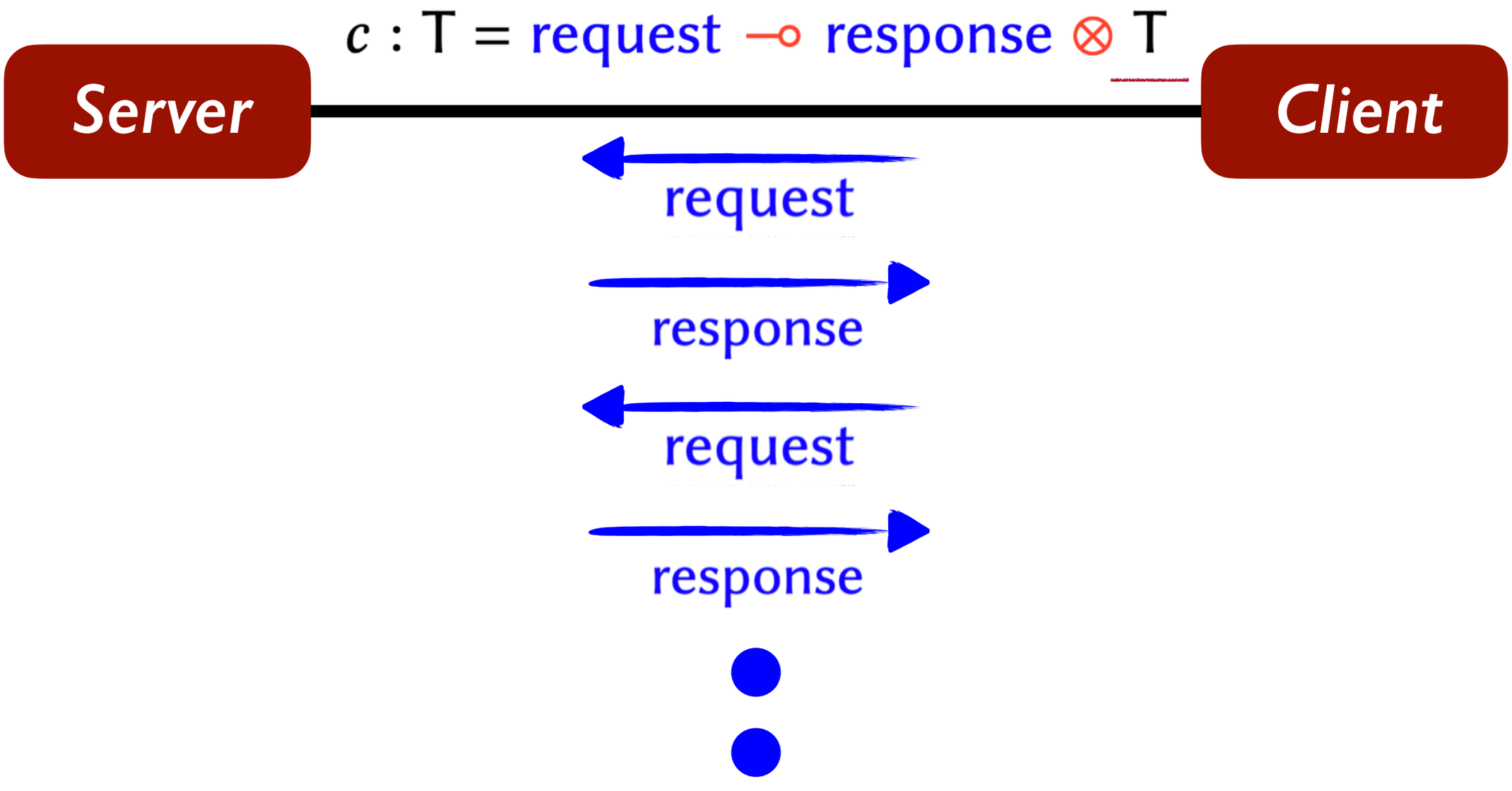
What are Session Types?

- Session types are a system for concurrent message-passing programs
- Communication via typed bi-directional channels
- Session type governs the type and direction of message exchanges

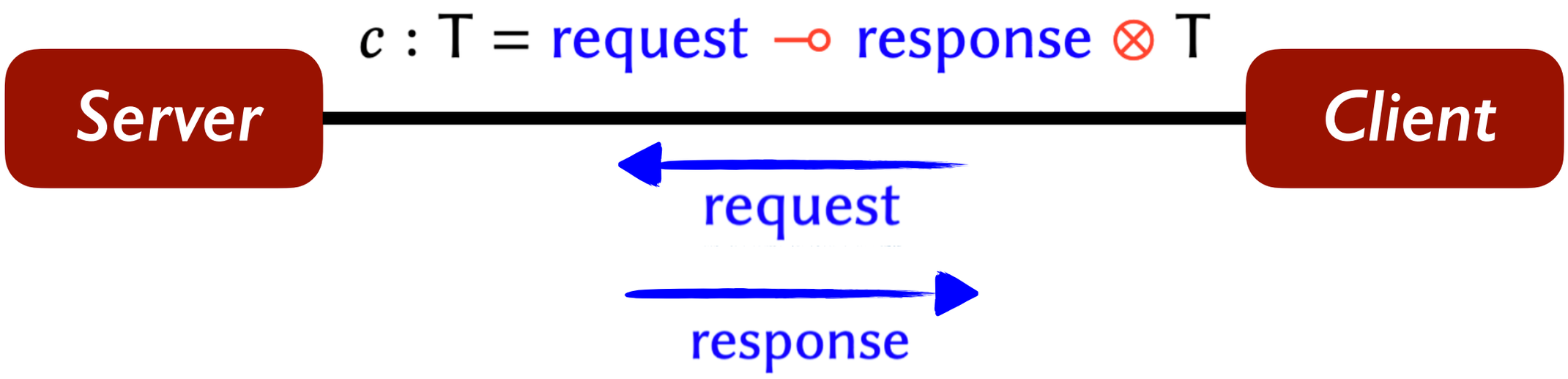


What are Session Types?

- Session types are a system for concurrent message-passing programs
- Communication via typed bi-directional channels
- Session type governs the type and direction of message exchanges

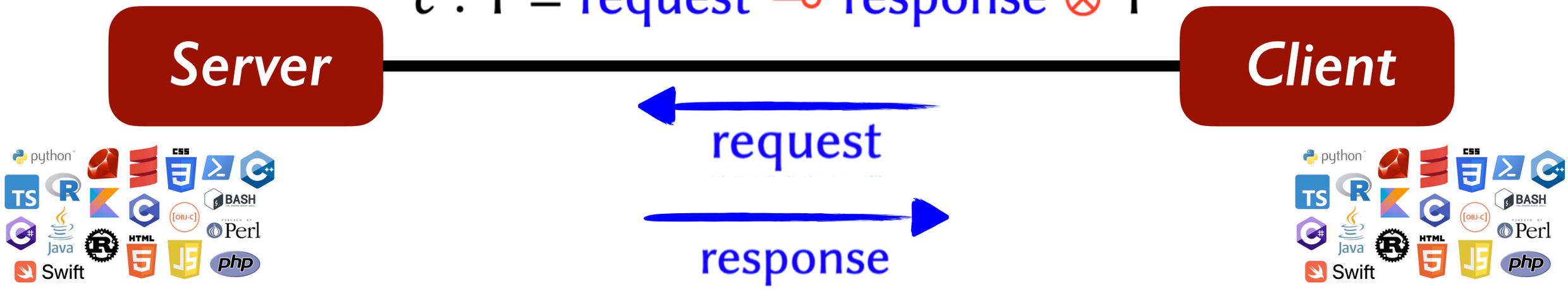


How do Session Types Help?



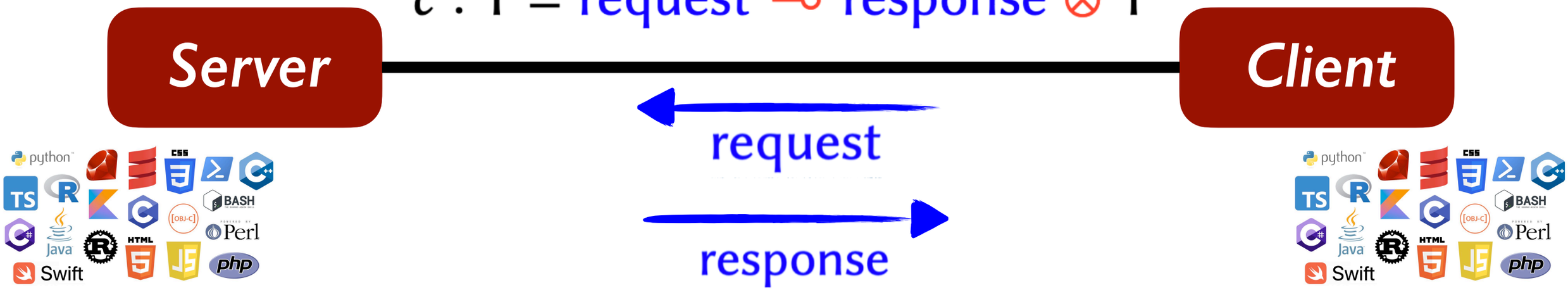
How do Session Types Help?

$$c : \mathsf{T} = \text{request} \multimap \text{response} \otimes \mathsf{T}$$



How do Session Types Help?

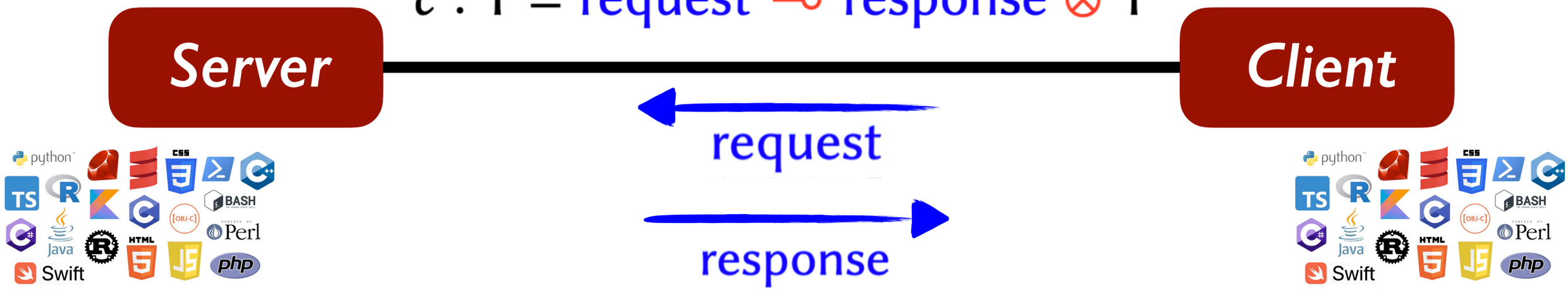
$$c : \mathsf{T} = \text{request} \multimap \text{response} \otimes \mathsf{T}$$



✓ *Communication Safety*

How do Session Types Help?

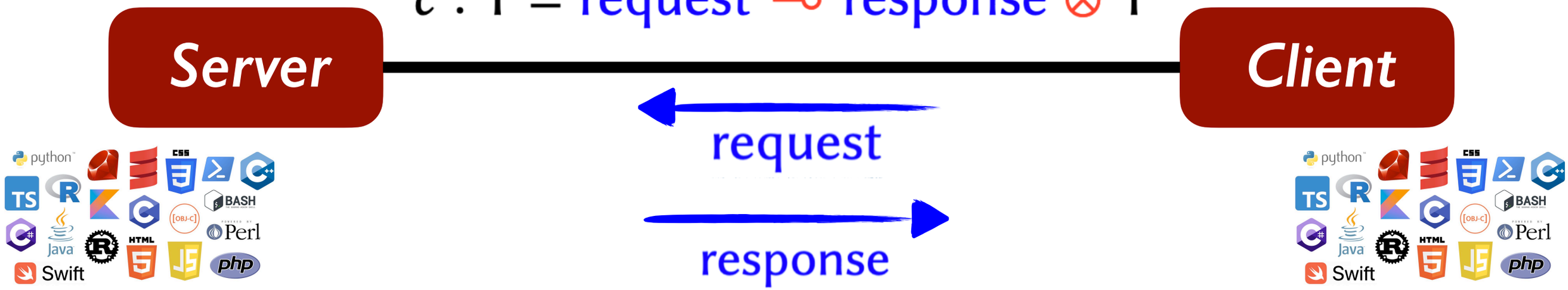
$$c : \mathsf{T} = \text{request} \multimap \text{response} \otimes \mathsf{T}$$



- ✓ *Communication Safety*
- ✓ *Type acts as Specification*

How do Session Types Help?

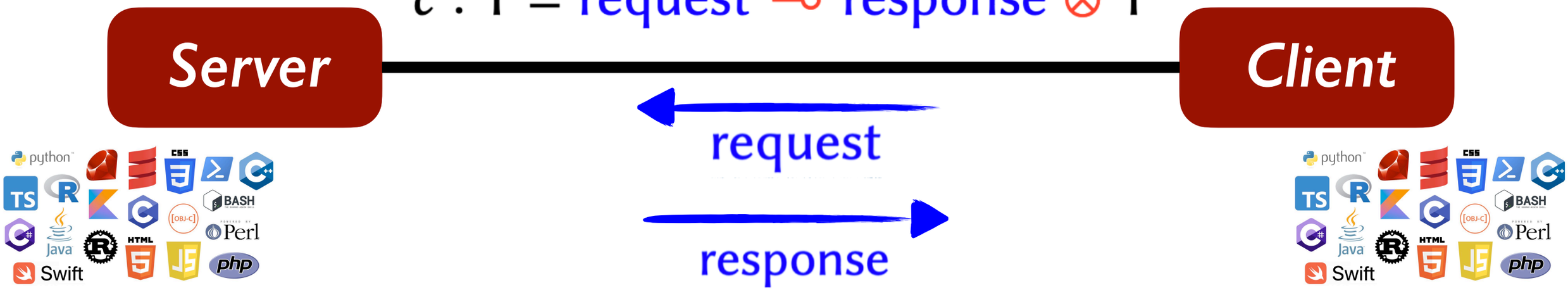
$$c : \mathsf{T} = \text{request} \multimap \text{response} \otimes \mathsf{T}$$



- ✓ *Communication Safety*
- ✓ *Type acts as Specification*
- ✓ *Compositionality & Modularity*

How do Session Types Help?

$$c : \mathsf{T} = \text{request} \multimap \text{response} \otimes \mathsf{T}$$



- *Communication Safety*
- *Type acts as Specification*
- *Compositionality & Modularity*
- *Deadlock & Livelock Freedom!*

Talk Outline

Protocol Enforcement & Asset Preservation

Probabilistic Reasoning

Performance & Verification

Nomos

Resource-Aware Session Types

[LICS '18, SAS '20, CSF '21]

NomosPro

Probabilistic Session Types

[POPL '23]

Rast

Refinement Session Types

[ICFP '18, FSCD '20, CONCUR '20]

Session Types

Type system for concurrent message-passing programs

Talk Outline

Protocol Enforcement & Asset Preservation

Probabilistic Reasoning

Performance & Verification

Nomos

Resource-Aware Session Types

[LICS '18, SAS '20, CSF '21]

NomosPro

Probabilistic Session Types

[POPL '23]

Rast

Refinement Session Types

[ICFP '18, FSCD '20, CONCUR '20]

Session Types

Type system for concurrent message-passing programs

Smart Contracts

Programs that implement a digital (legal or financial) transaction

Smart Contracts

Programs that implement a digital (legal or financial) transaction

Online Transactions

Smart Contracts

Programs that implement a digital (legal or financial) transaction

Online Transactions

Betterment
wealthfront
robinhood
Vanguard®

Wealth Management Applications

Smart Contracts

Programs that implement a digital (legal or financial) transaction

Online Transactions

recently popularized in the form of

wealthfront
robinhood
Betterment
Vanguard®

Wealth Management Applications

Blockchains and Cryptocurrencies

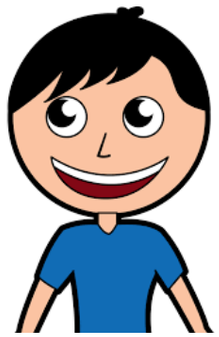
Example: Auction Contract

status: running

Example: Auction Contract

Bidder 1

Bid 1

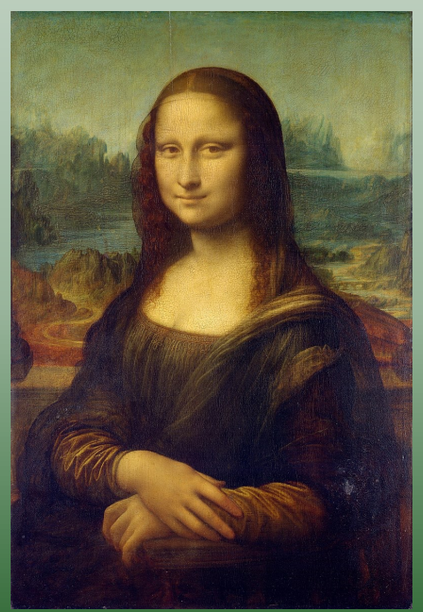


Bidder 2

Bid 2

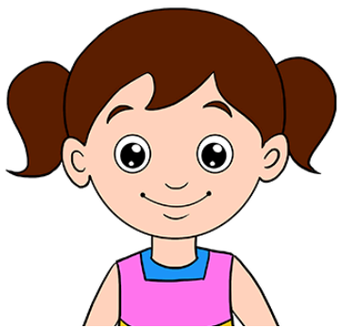
Bidder 3

Bid 3

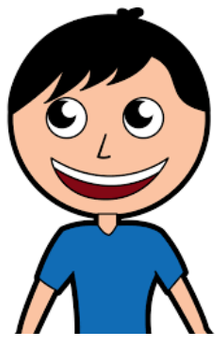


status: running

Example: Auction Contract



Bidder 1



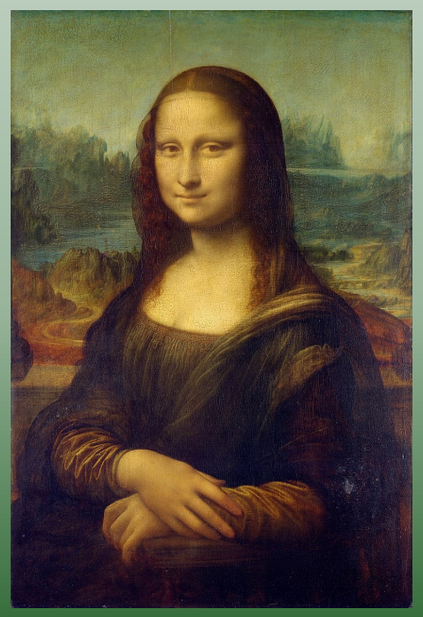
Bidder 2

Bidder 3

Bid 1

Bid 2

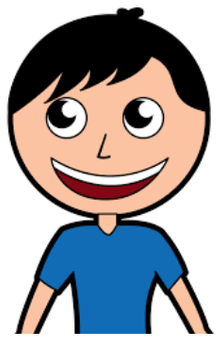
Bid 3



status: running

Example: Auction Contract

Bidder 1



Bidder 2

Bidder 3

Bid 1

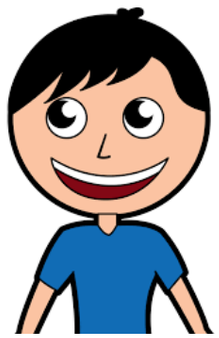
Bid 2

Bid 3

status: ended

Example: Auction Contract

Bidder 1



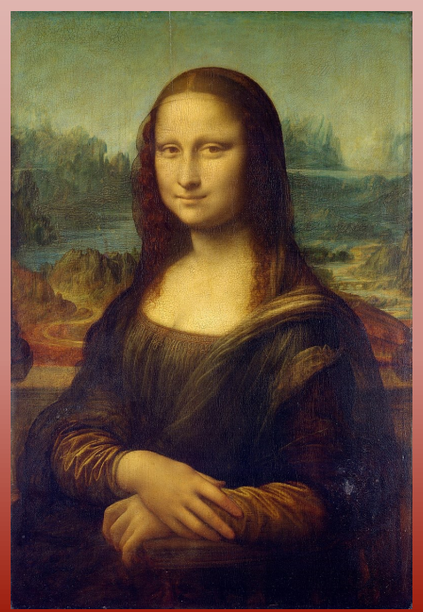
Bidder 2

Bidder 3

Bid 1

Bid 2

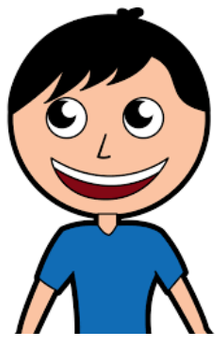
Bid 3 ✓



status: ended

Example: Auction Contract

Bidder 1



Bidder 2

Bidder 3

Bid 1

Bid 2

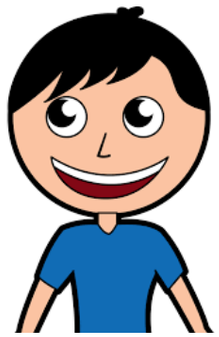
Bid 3 ✓

status: ended

Example: Auction Contract

Bidder 1

Bid 1



Bidder 2

Bid 2

Bidder 3

Bid 3 ✓

status: ended

Auction in Solidity


```
function bid() public payable {
    uint bid = msg.value;                  // bidder's bid
    address bidder = msg.sender;           // bidder's address
    pendingReturns[bidder] = bid;          // store in local dictionary
    if (bid > highestBid) {                // update highest bidder & bid
        highestBidder = bidder;
        highestBid = bid;
    }
}

function collect() public {
    // highest bidder gets Mona Lisa
    require (msg.sender != highestBidder);
    // return bid to bidder
    uint amount = pendingReturns[msg.sender];
    msg.sender.send(amount)
}
```

Spot the Bug!


```
function collect() public {
  // highest bidder gets Mona Lisa
  require (msg.sender != highestBidder);
  // return bid to bidder
  uint amount = pendingReturns[msg.sender];
  msg.sender.send(amount)
}
```

Spot the Bug!


```
function collect() public {
  // highest bidder gets Mona Lisa
  require (msg.sender != highestBidder);
  // return bid to bidder
  uint amount = pendingReturns[msg.sender];
  msg.sender.send(amount)
}
```

Hint: function can be called at *any time* by a bidder

Spot the Bug!


```
function collect() public {
  // highest bidder gets Mona Lisa
  require (msg.sender != highestBidder);
  // return bid to bidder
  uint amount = pendingReturns[msg.sender];
  msg.sender.send(amount)
}
```

What happens if collect is called when auction is running?

Hint: function can be called at *any time* by a bidder

Spot the Bug!


```
function collect() public {
  // highest bidder gets Mona Lisa
  require (msg.sender != highestBidder);
  // return bid to bidder
  uint amount = pendingReturns[msg.sender];
  msg.sender.send(amount)
}
```

What happens if collect is called when auction is running?

Hint: function can be called at *any time* by a bidder

add clause: **require (status == ended) ;**

Spot the Bug!


```
function collect() public {
  // highest bidder gets Mona Lisa
  require (msg.sender != highestBidder);
  // return bid to bidder
  uint amount = pendingReturns[msg.sender];
  msg.sender.send(amount)
}
```

Protocol Violation!

What happens if collect is called when auction is running?

Hint: function can be called at *any time* by a bidder

add clause: require (status == ended) ;

Spot the Bug!


```
function collect() public {
  // highest bidder gets Mona Lisa
  require (msg.sender != highestBidder);
  require (status == ended);
  // return bid to bidder
  uint amount = pendingReturns[msg.sender];
  msg.sender.send(amount)
}
```

Spot the Bug!


```
function collect() public {
  // highest bidder gets Mona Lisa
  require (msg.sender != highestBidder);
  require (status == ended);
  // return bid to bidder
  uint amount = pendingReturns[msg.sender];
  msg.sender.send(amount)
}
```

Hint: function can be called *multiple times* by a bidder

Spot the Bug!


```
function collect() public {
  // highest bidder gets Mona Lisa
  require (msg.sender != highestBidder);
  require (status == ended);
  // return bid to bidder
  uint amount = pendingReturns[msg.sender];
  msg.sender.send(amount)
}
```

What happens if collect is called twice?

Hint: function can be called *multiple times* by a bidder

Spot the Bug!


```
function collect() public {
  // highest bidder gets Mona Lisa
  require (msg.sender != highestBidder);
  require (status == ended);
  // return bid to bidder
  uint amount = pendingReturns[msg.sender];
  msg.sender.send(amount)
}
```

What happens if collect is called twice?

Hint: function can be called *multiple times* by a bidder

set: **pendingReturns[msg.sender] = 0 ;**

Spot the Bug!


```
function collect() public {
  // highest bidder gets Mona Lisa
  require (msg.sender != highestBidder);
  require (status == ended);
  // return bid to bidder
  uint amount = pendingReturns[msg.sender];
  msg.sender.send(amount)
}
```

Asset is Duplicated!

What happens if collect is called twice?

Hint: function can be called *multiple times* by a bidder

set: pendingReturns[msg.sender] = 0 ;

What is the Transaction Fee?


```
function collect() public {
  // highest bidder gets Mona Lisa
  require (msg.sender != highestBidder);
  require (status == ended);
  // return bid to bidder
  uint amount = pendingReturns[msg.sender];
  pendingReturns[msg.sender] = 0;
  msg.sender.send(amount)
}
```

What is the Transaction Fee?


```
function collect() public {
  // highest bidder gets Mona Lisa
  require (msg.sender != highestBidder);
  require (status == ended);
  // return bid to bidder
  uint amount = pendingReturns[msg.sender];
  pendingReturns[msg.sender] = 0;
  msg.sender.send(amount)
}
```

- *Transaction fee is proportional to execution cost*
- *Need to predict cost statically, fees is paid upfront*

What is the Transaction Fee?


```
function collect() public {
  // highest bidder gets Mona Lisa
  require (msg.sender != highestBidder);
  require (status == ended);
  // return bid to bidder
  uint amount = pendingReturns[msg.sender];
  pendingReturns[msg.sender] = 0;
  msg.sender.send(amount)
}
```

Need Automatic Execution Cost Prediction!

- *Transaction fee is proportional to execution cost*
- *Need to predict cost statically, fees is paid upfront*

The Nomos Language

Smart Contracts

Resource-Aware Session Types

Challenges:

1. *contract protocols are violated*
2. *asset duplication/deletion*
3. *automatic inference of execution cost*

Solutions:

1. *session types enforce protocols*
2. *linear types track assets*
3. *resource-aware types infer execution cost*

The Nomos Language

Smart Contracts

Challenges:

1. *contract protocols are violated*
2. *asset duplication/deletion*
3. *automatic inference of execution cost*

Resource-Aware Session Types

Solutions:

1. *session types enforce protocols*
2. *linear types track assets*
3. *resource-aware types infer execution cost*

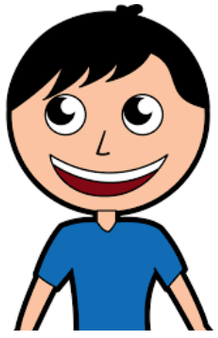
Protocol Enforcement with Types

Protocol Enforcement with Types

running phase

Protocol Enforcement with Types

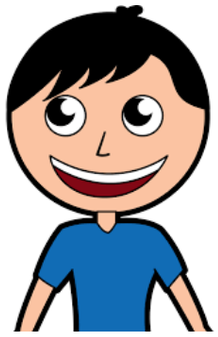
running phase



bid

Protocol Enforcement with Types

running phase



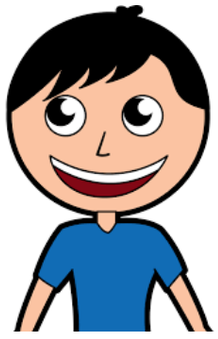
bid

collect

compilation error

Protocol Enforcement with Types

running phase

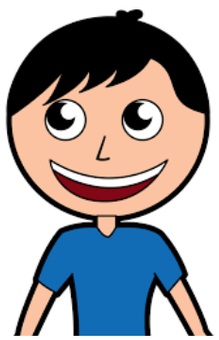


bid

collect

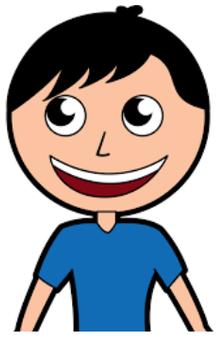
compilation error

ended phase



Protocol Enforcement with Types

running phase

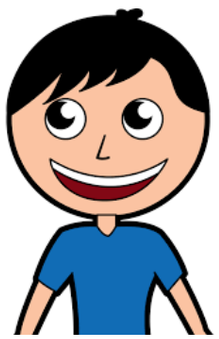


bid

collect

compilation error

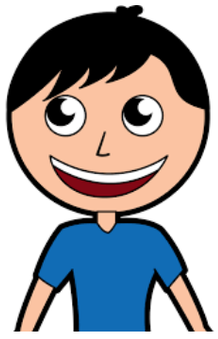
ended phase



collect

Protocol Enforcement with Types

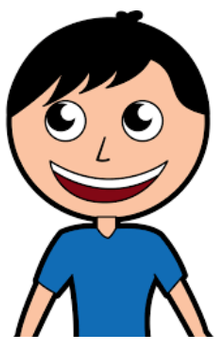
running phase



bid ✓

collect ✗ *compilation error*

ended phase



collect ✓

bid ✗ *compilation error*

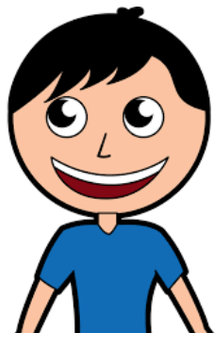
Auction as a Session Type

Auction as a Session Type

$$\begin{aligned} \mathit{type}\ \mathsf{auction} = \oplus\{&\mathbf{running} : \&\{\mathbf{bid} : \mathsf{money} \multimap \mathsf{auction}\}, \\ &\mathbf{ended} : \&\{\mathbf{collect} : \oplus\{\mathbf{won} : \mathsf{monalisa} \otimes \mathsf{auction}, \\ &\qquad\qquad \mathbf{lost} : \mathsf{money} \otimes \mathsf{auction}\}\}\} \end{aligned}$$

Auction as a Session Type

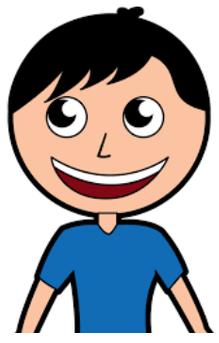
running phase



$$\mathit{type}\ \mathsf{auction} = \oplus\{\mathbf{running} : \&\{\mathbf{bid} : \mathsf{money} \multimap \mathsf{auction}\},$$
$$\mathbf{ended} : \&\{\mathbf{collect} : \oplus\{\mathbf{won} : \mathsf{monalisa} \otimes \mathsf{auction},$$
$$\mathbf{lost} : \mathsf{money} \otimes \mathsf{auction}\}\}\}$$

Auction as a Session Type

running phase

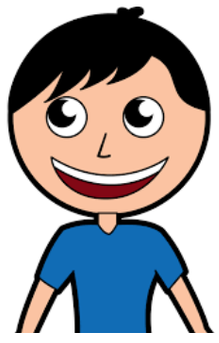


running

$$\mathit{type}\ \mathsf{auction} = \oplus\{\mathbf{running} : \&\{\mathbf{bid} : \mathsf{money} \multimap \mathsf{auction}\},$$
$$\mathbf{ended} : \&\{\mathbf{collect} : \oplus\{\mathbf{won} : \mathsf{monalisa} \otimes \mathsf{auction},$$
$$\mathbf{lost} : \mathsf{money} \otimes \mathsf{auction}\}\}\}$$

Auction as a Session Type

running phase

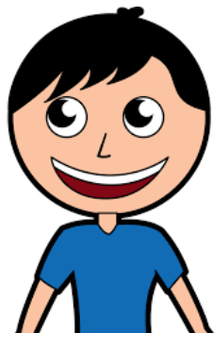


bid

$$\mathit{type}\ \mathsf{auction} = \oplus\{\mathbf{running} : \&\{\mathbf{bid} : \mathsf{money} \multimap \mathsf{auction}\},$$
$$\mathbf{ended} : \&\{\mathbf{collect} : \oplus\{\mathbf{won} : \mathsf{monalisa} \otimes \mathsf{auction},$$
$$\mathbf{lost} : \mathsf{money} \otimes \mathsf{auction}\}\}\}$$

Auction as a Session Type

running phase

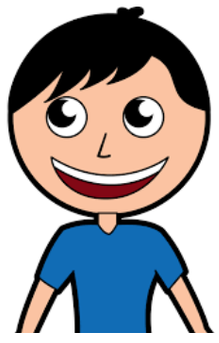


money

$$\mathit{type}\ \mathsf{auction} = \oplus\{\mathbf{running} : \&\{\mathbf{bid} : \mathsf{money} \multimap \mathsf{auction}\},$$
$$\mathbf{ended} : \&\{\mathbf{collect} : \oplus\{\mathbf{won} : \mathsf{monalisa} \otimes \mathsf{auction},$$
$$\mathbf{lost} : \mathsf{money} \otimes \mathsf{auction}\}\}\}$$

Auction as a Session Type

running phase

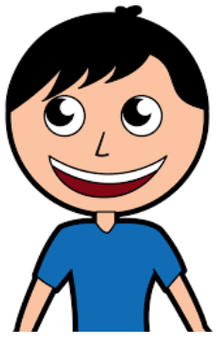


$$\mathit{type}\ \mathsf{auction} = \oplus\{\mathbf{running} : \&\{\mathbf{bid} : \mathsf{money} \multimap \mathsf{auction}\},$$
$$\mathbf{ended} : \&\{\mathbf{collect} : \oplus\{\mathbf{won} : \mathsf{monalisa} \otimes \mathsf{auction},$$
$$\mathbf{lost} : \mathsf{money} \otimes \mathsf{auction}\}\}\}$$

Auction as a Session Type

running phase

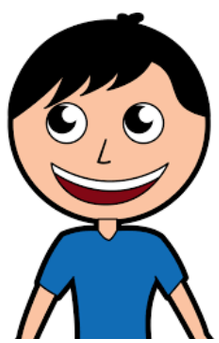
ended phase



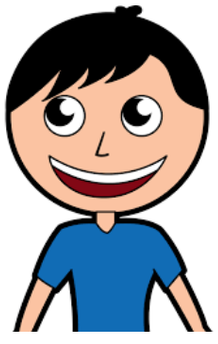
$$type\ \text{auction} = \oplus\{\mathbf{running} : \&\{\mathbf{bid} : \text{money} \multimap \text{auction}\},$$
$$\mathbf{ended} : \&\{\mathbf{collect} : \oplus\{\mathbf{won} : \text{monalisa} \otimes \text{auction},$$
$$\mathbf{lost} : \text{money} \otimes \text{auction}\}\}\}$$

Auction as a Session Type

running phase



ended phase



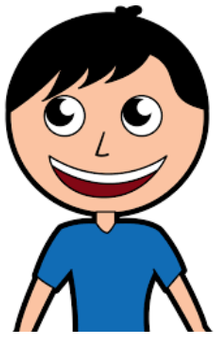
ended

$$type\ \text{auction} = \oplus\{\mathbf{running} : \&\{\mathbf{bid} : \text{money} \multimap \text{auction}\},$$
$$\mathbf{ended} : \&\{\mathbf{collect} : \oplus\{\mathbf{won} : \text{monalisa} \otimes \text{auction},$$
$$\mathbf{lost} : \text{money} \otimes \text{auction}\}\}\}$$

Auction as a Session Type

running phase

ended phase

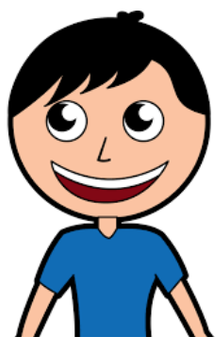


collect

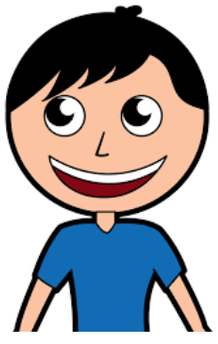
$$\mathit{type}\ \mathsf{auction} = \oplus\{\mathbf{running} : \&\{\mathbf{bid} : \mathsf{money} \multimap \mathsf{auction}\},$$
$$\mathbf{ended} : \&\{\underline{\mathbf{collect}} : \oplus\{\mathbf{won} : \mathsf{monalisa} \otimes \mathsf{auction},$$
$$\mathbf{lost} : \mathsf{money} \otimes \mathsf{auction}\}\}\}$$

Auction as a Session Type

running phase



ended phase

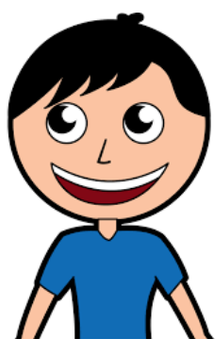


won

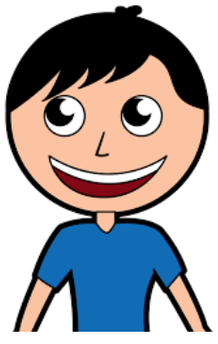
$$\begin{aligned} \mathit{type}\ \mathsf{auction} = \oplus\{&\mathbf{running} : \&\{\mathbf{bid} : \mathsf{money} \multimap \mathsf{auction}\}, \\ &\mathbf{ended} : \&\{\mathbf{collect} : \oplus\{\mathbf{won} : \mathsf{monalisa} \otimes \mathsf{auction}, \\ &\qquad\qquad \mathbf{lost} : \mathsf{money} \otimes \mathsf{auction}\}\}\} \end{aligned}$$

Auction as a Session Type

running phase



ended phase

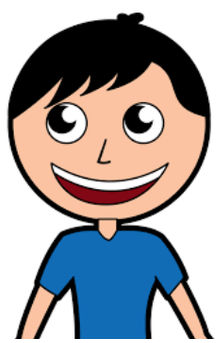


monalisa

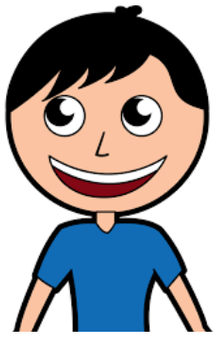
$$\mathit{type}\ \mathsf{auction} = \oplus\{\mathbf{running} : \&\{\mathbf{bid} : \mathsf{money} \multimap \mathsf{auction}\},$$
$$\mathbf{ended} : \&\{\mathbf{collect} : \oplus\{\mathbf{won} : \mathsf{monalisa} \otimes \mathsf{auction},$$
$$\mathbf{lost} : \mathsf{money} \otimes \mathsf{auction}\}\}\}$$

Auction as a Session Type

running phase



ended phase

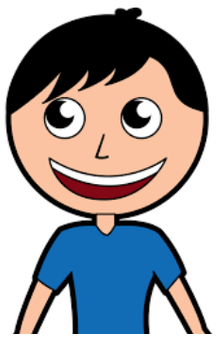


lost

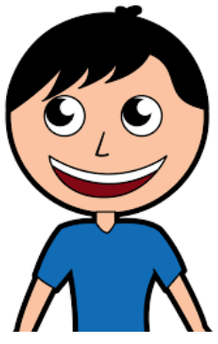
$$\textit{type}\ \text{auction} = \oplus\{\mathbf{running} : \&\{\mathbf{bid} : \text{money} \multimap \text{auction}\},$$
$$\mathbf{ended} : \&\{\mathbf{collect} : \oplus\{\mathbf{won} : \text{monalisa} \otimes \text{auction},$$
$$\mathbf{lost} : \text{money} \otimes \text{auction}\}\}\}$$

Auction as a Session Type

running phase



ended phase



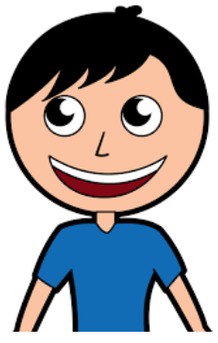
money

$$type\ \text{auction} = \oplus\{\mathbf{running} : \&\{\mathbf{bid} : \text{money} \multimap \text{auction}\},$$
$$\mathbf{ended} : \&\{\mathbf{collect} : \oplus\{\mathbf{won} : \text{monalisa} \otimes \text{auction},$$
$$\mathbf{lost} : \text{money} \otimes \text{auction}\}\}\}$$

Auction as a Session Type

running phase

ended phase



$$type\ \text{auction} = \oplus\{\mathbf{running} : \&\{\mathbf{bid} : \text{money} \multimap \text{auction}\},$$
$$\mathbf{ended} : \&\{\mathbf{collect} : \oplus\{\mathbf{won} : \text{monalisa} \otimes \text{auction},$$
$$\mathbf{lost} : \text{money} \otimes \text{auction}\}\}\}$$

Asset Preservation

Asset Preservation

<u>Key Idea:</u> Session types in Nomos are derived from linear logic (which is the logic of assets!)

Asset Preservation

Key Idea: *Session types in Nomos are derived from linear logic (which is the logic of assets!)*

$$\textit{type}\ \text{auction} = \oplus\{\mathbf{running} : \&\{\mathbf{bid} : \text{money} \multimap \text{auction}\},$$
$$\mathbf{ended} : \&\{\mathbf{collect} : \oplus\{\mathbf{won} : \text{monalisa} \otimes \text{auction},$$
$$\mathbf{lost} : \text{money} \otimes \text{auction}\}\}\}$$

Asset Preservation

Key Idea: *Session types in Nomos are derived from linear logic (which is the logic of assets!)*

$$\textit{type}\ \text{auction} = \oplus\{\mathbf{running} : \&\{\mathbf{bid} : \text{money} \multimap \text{auction}\},$$
$$\mathbf{ended} : \&\{\mathbf{collect} : \oplus\{\mathbf{won} : \text{monalisa} \otimes \text{auction},$$
$$\mathbf{lost} : \text{money} \otimes \text{auction}\}\}\}$$

Nomos statically guarantees:

- *money and monalisa are assets, cannot be duplicated or discarded, only transferred between processes*
- *no additional machinery needed!*

Get Asset Preservation for Free!

Key Idea: *Session types in Nomos are derived from linear logic (which is the logic of assets!)*

$$\textit{type}\ \text{auction} = \oplus\{\mathbf{running} : \&\{\mathbf{bid} : \text{money} \multimap \text{auction}\},$$
$$\mathbf{ended} : \&\{\mathbf{collect} : \oplus\{\mathbf{won} : \text{monalisa} \otimes \text{auction},$$
$$\mathbf{lost} : \text{money} \otimes \text{auction}\}\}\}$$

Nomos statically guarantees:

- *money and monalisa are assets, cannot be duplicated or discarded, only transferred between processes*
- *no additional machinery needed!*

Expressing Transaction Fee

cost of bidding = 2, cost of collecting = 3

Expressing Transaction Fee

cost of bidding = 2, cost of collecting = 3

<u>Key Idea:</u> Nomos enhances session types with potential (or fee) that pays for execution cost

Expressing Transaction Fee

cost of bidding = 2, cost of collecting = 3

<u>Key Idea:</u> Nomos enhances session types with potential (or fee) that pays for execution cost

$$\begin{aligned} \mathit{type}\ \mathsf{auction} = {} & \triangleleft^{3} \oplus \{\mathbf{running} : \&\{\mathbf{bid} : \mathsf{money} \multimap \triangleright^{1}\mathsf{auction}\}, \\ & \mathbf{ended} : \&\{\mathbf{collect} : \oplus\{\mathbf{won} : \mathsf{monalisa} \otimes \triangleright^{0}\mathsf{auction}, \\ & \qquad \mathbf{lost} : \mathsf{money} \otimes \triangleright^{0}\mathsf{auction}\}\}\} \end{aligned}$$

Expressing Transaction Fee

cost of bidding = 2, cost of collecting = 3

Key Idea: *Nomos enhances session types with potential (or fee) that pays for execution cost*

$$\textit{type}\ \text{auction} = \triangleleft^{3} \oplus \{\mathbf{running} : \&\{\mathbf{bid} : \text{money} \multimap \triangleright^{1}\text{auction}\},$$
$$\mathbf{ended} : \&\{\mathbf{collect} : \oplus\{\mathbf{won} : \text{monalisa} \otimes \triangleright^{0}\text{auction},$$
$$\mathbf{lost} : \text{money} \otimes \triangleright^{0}\text{auction}\}\}\}$$

user pays 3 units as fees at start of execution

Expressing Transaction Fee

cost of bidding = 2, cost of collecting = 3

Key Idea: *Nomos enhances session types with potential (or fee) that pays for execution cost*

$$\textit{type}\ \text{auction} = \triangleleft^{3} \oplus \{\mathbf{running} : \&\{\mathbf{bid} : \text{money} \multimap \triangleright^{1}\text{auction}\},$$
$$\mathbf{ended} : \&\{\mathbf{collect} : \oplus\{\mathbf{won} : \text{monalisa} \otimes \triangleright^{0}\text{auction},$$
$$\mathbf{lost} : \text{money} \otimes \triangleright^{0}\text{auction}\}\}\}$$

user pays 3 units as fees at start of execution

return leftover fee back to user in the end

Cost Computation is Automatic!

Cost Computation is Automatic!

$$\mathit{type}\ \mathsf{auction} = \triangleleft^* \oplus \{\mathbf{running} : \&\{\mathbf{bid} : \mathsf{money} \multimap \triangleright^* \mathsf{auction}\},$$
$$\mathbf{ended} : \&\{\mathbf{collect} : \oplus\{\mathbf{won} : \mathsf{monalisa} \otimes \triangleright^* \mathsf{auction},$$
$$\mathbf{lost} : \mathsf{money} \otimes \triangleright^* \mathsf{auction}\}\}\}$$

Cost Computation is Automatic!

$$\mathit{type}\ \text{auction} = \triangleleft^* \oplus \{\mathbf{running} : \&\{\mathbf{bid} : \text{money} \multimap \triangleright^*\text{auction}\},$$
$$\mathbf{ended} : \&\{\mathbf{collect} : \oplus\{\mathbf{won} : \text{monalisa} \otimes \triangleright^*\text{auction},$$
$$\mathbf{lost} : \text{money} \otimes \triangleright^*\text{auction}\}\}\}$$

Linear constraints shipped to LP solver

Constraints are solved and substituted back into the program

Cost Computation is Automatic!

$$\mathit{type}\ \text{auction} = \triangleleft^{*} \oplus \{\mathbf{running} : \&\{\mathbf{bid} : \text{money} \multimap \triangleright^{*}\text{auction}\},$$
$$\mathbf{ended} : \&\{\mathbf{collect} : \oplus\{\mathbf{won} : \text{monalisa} \otimes \triangleright^{*}\text{auction},$$
$$\mathbf{lost} : \text{money} \otimes \triangleright^{*}\text{auction}\}\}\}$$

Linear constraints shipped to LP solver

Constraints are solved and substituted back into the program

$$\mathit{type}\ \text{auction} = \triangleleft^{3} \oplus \{\mathbf{running} : \&\{\mathbf{bid} : \text{money} \multimap \triangleright^{1}\text{auction}\},$$
$$\mathbf{ended} : \&\{\mathbf{collect} : \oplus\{\mathbf{won} : \text{monalisa} \otimes \triangleright^{0}\text{auction},$$
$$\mathbf{lost} : \text{money} \otimes \triangleright^{0}\text{auction}\}\}\}$$

Cost Computation is Hard!

Cost Computation is Hard!

Cost Computation is Hard!

Cost Computation is Hard!

How do you compute the execution cost of distributed computation?

Cost Computation is Hard!

How do you compute the execution cost of distributed computation?

No static data structures

Cost Computation is Hard!

How do you compute the execution cost of distributed computation?

No static data structures

Cost may depend on number of processes, buffer sizes, etc.

Cost Computation is Hard!

How do you compute the execution cost of distributed computation?

No static data structures

Cost may depend on number of processes, buffer sizes, etc.

Inter-process dependencies

Cost Computation is Hard!

How do you compute the execution cost of distributed computation?

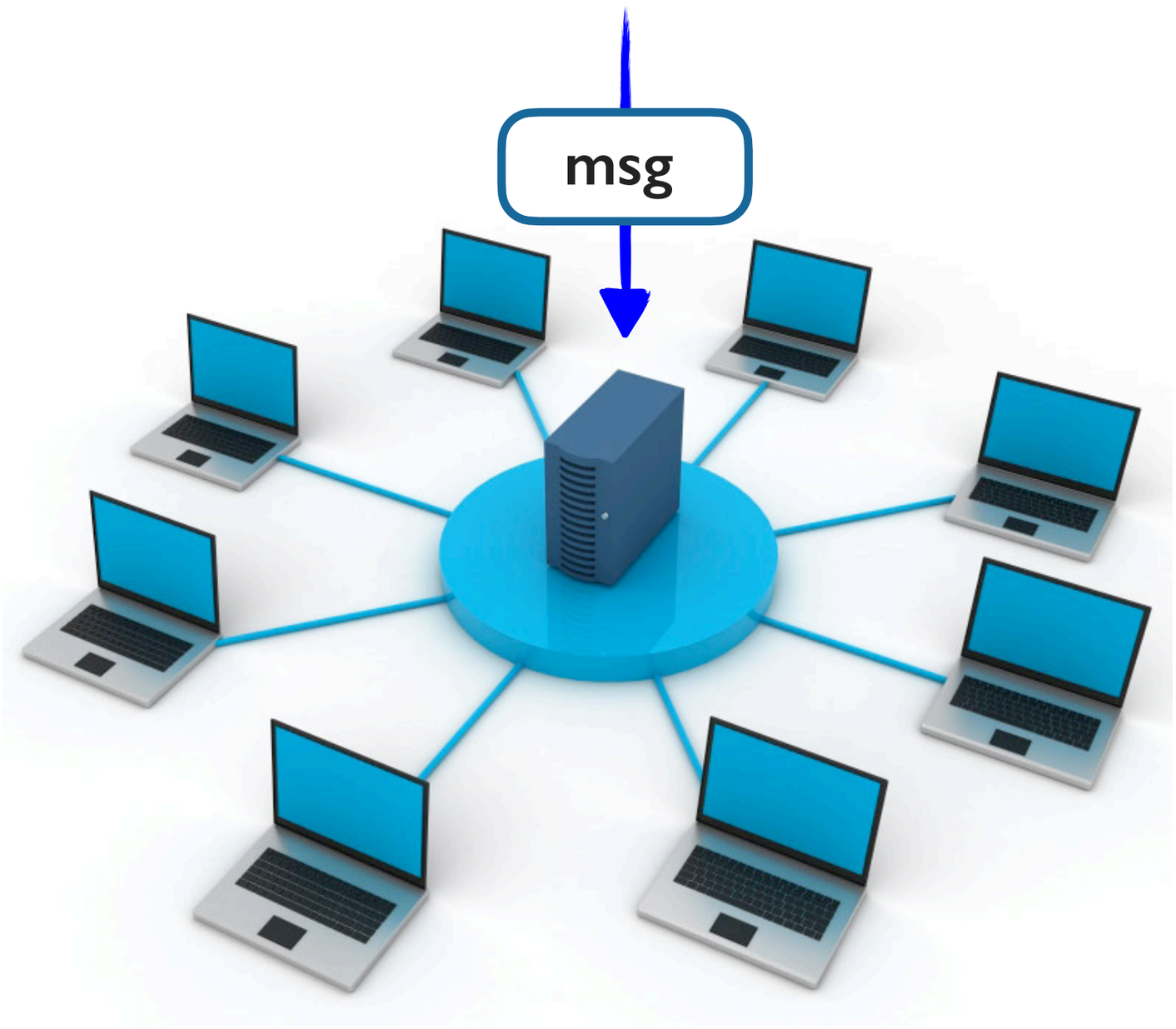
What should cost be function of?

No static data structures

Cost may depend on number of processes, buffer sizes, etc.

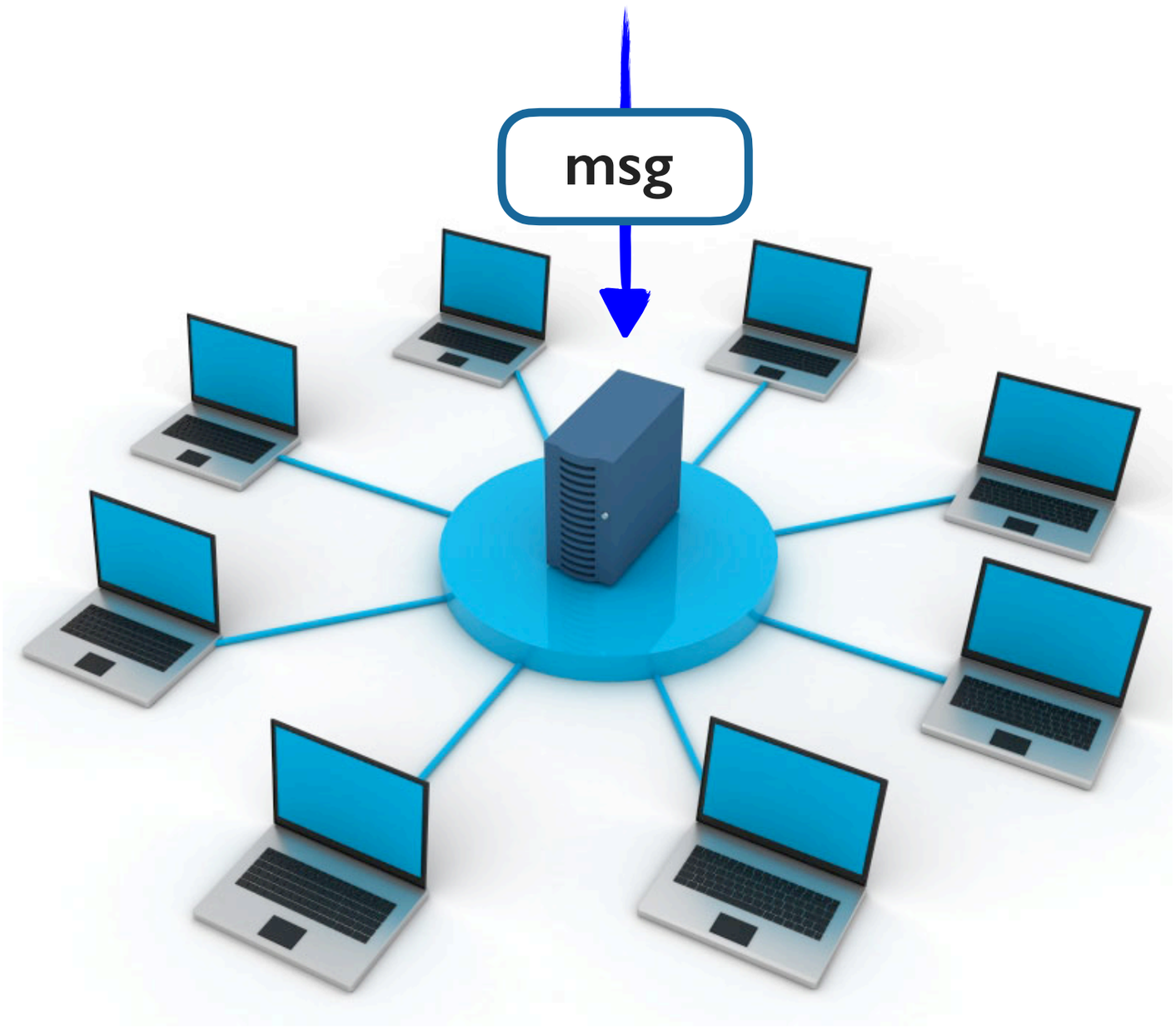
Inter-process dependencies

Resource-Aware Session Types



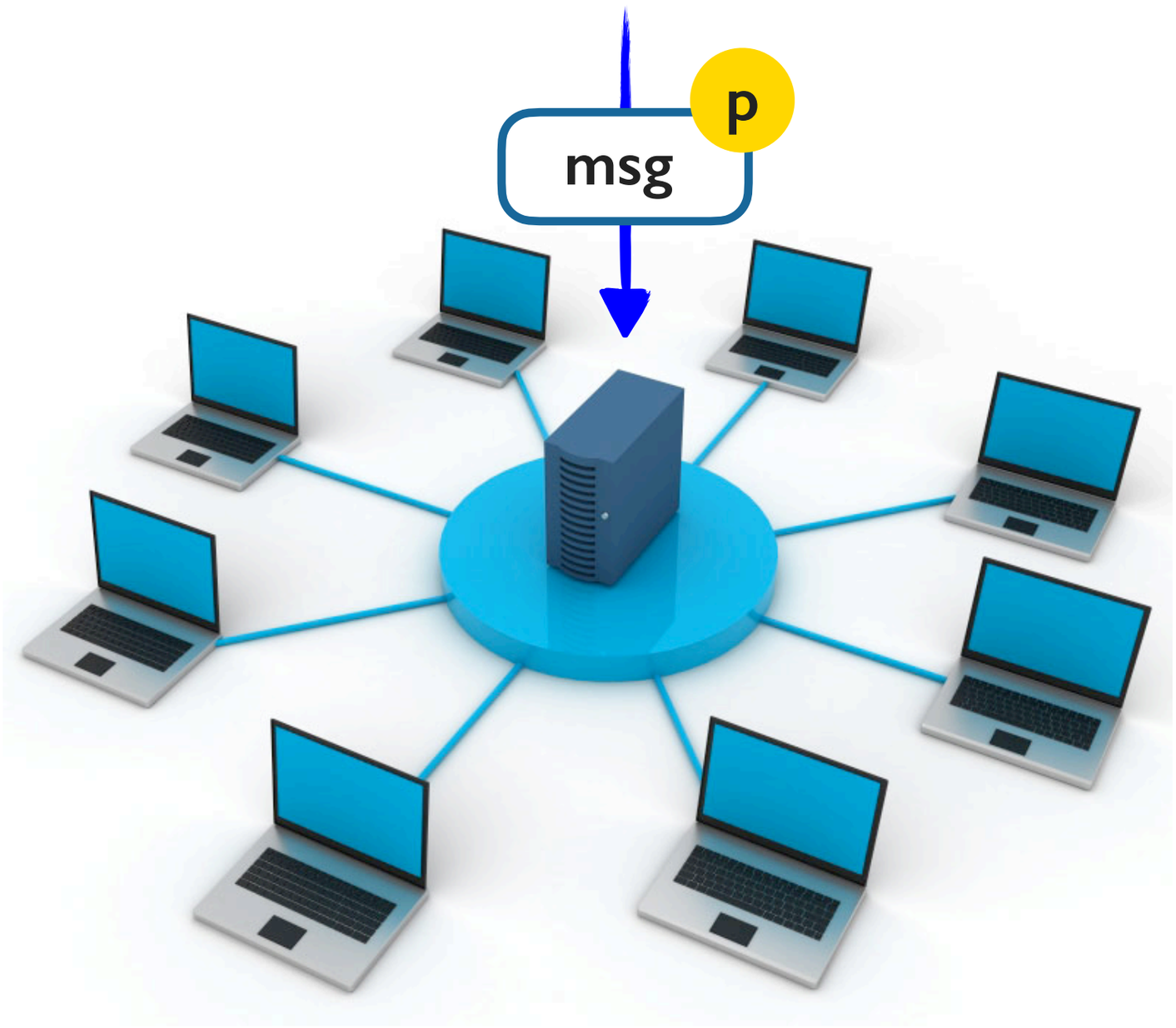
Resource-Aware Session Types

<u>Key Idea:</u> Messages carry potential & processes consume potential



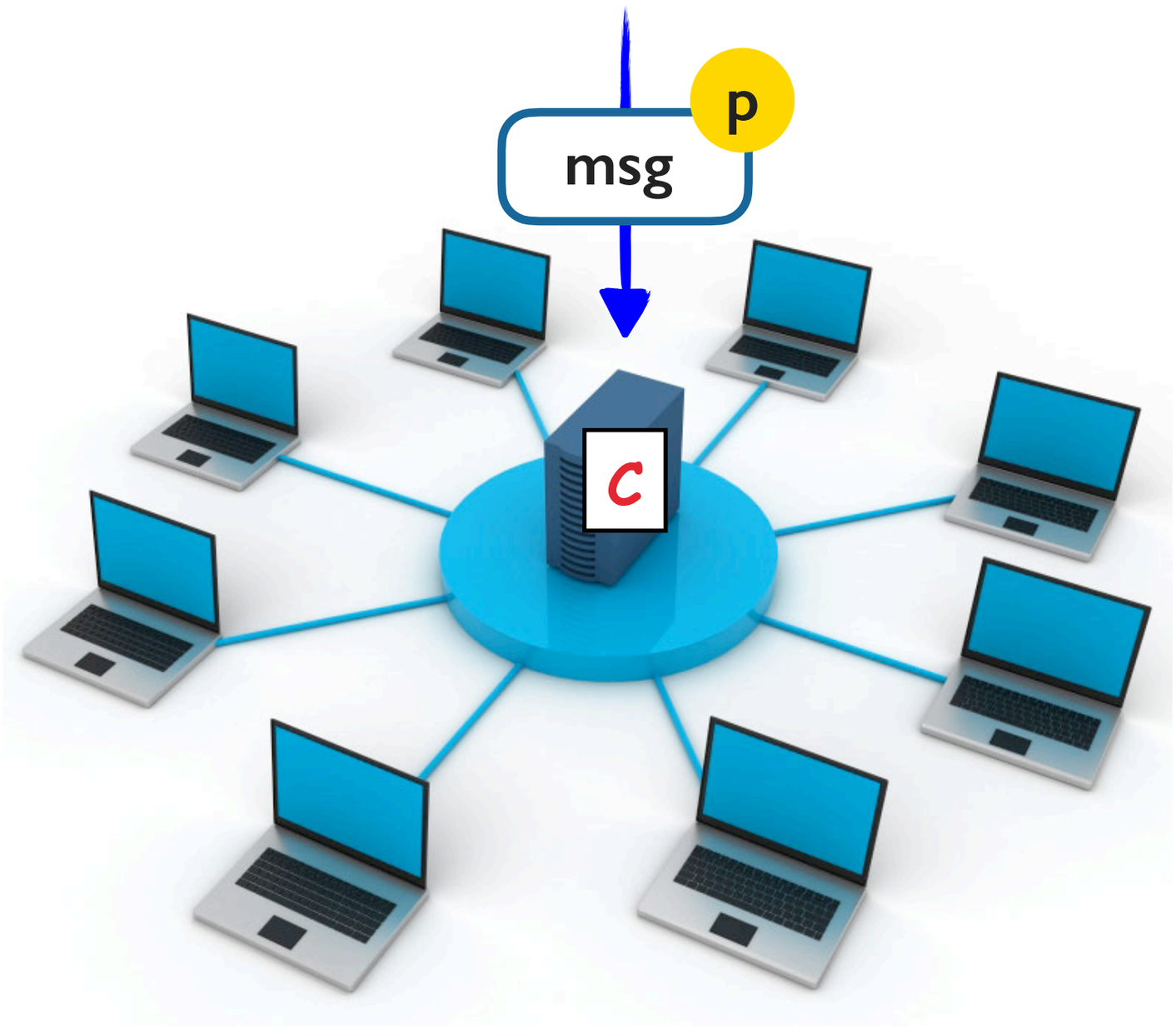
Resource-Aware Session Types

<u>Key Idea:</u> Messages carry potential & processes consume potential



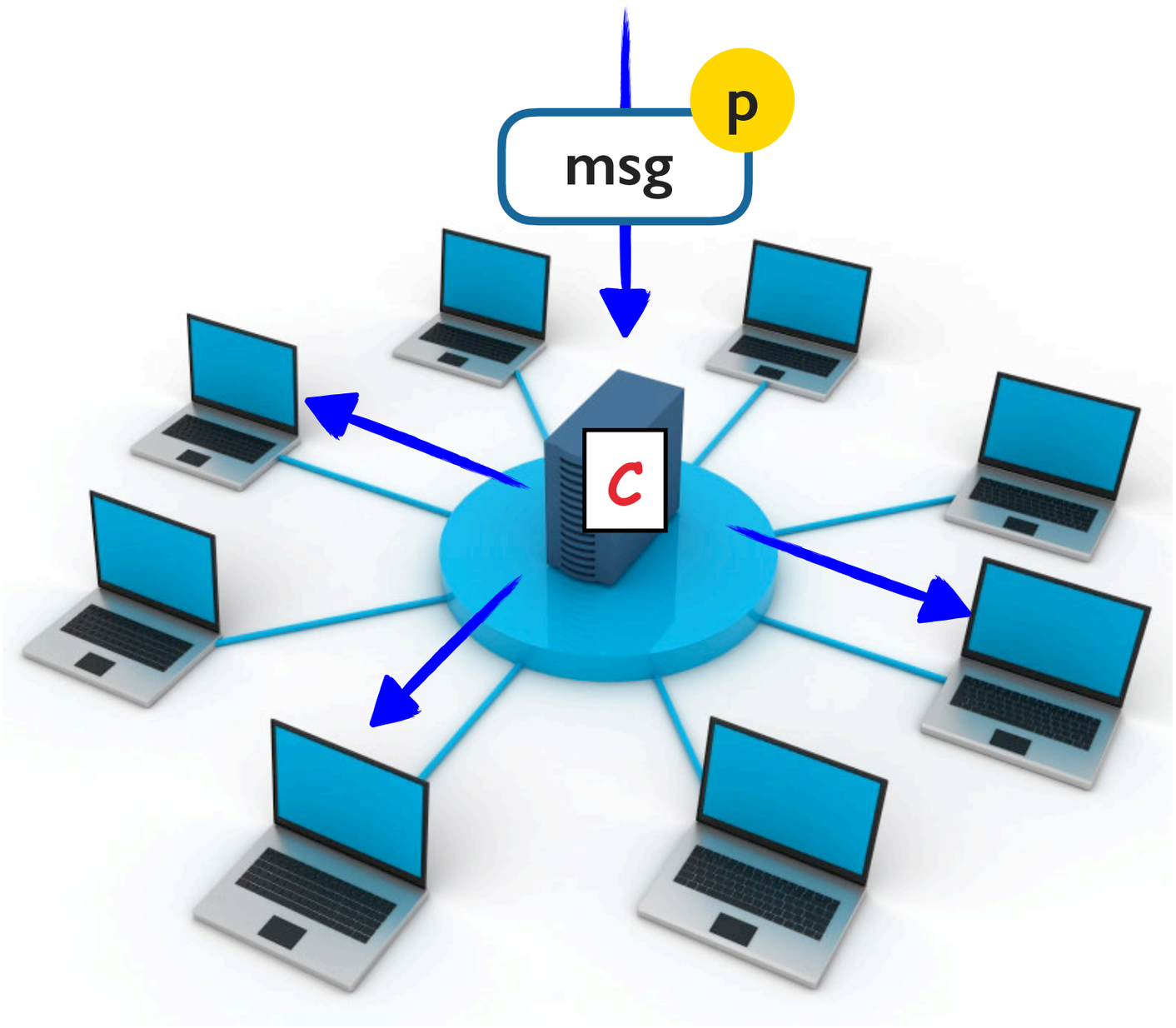
Resource-Aware Session Types

<u>Key Idea:</u> Messages carry potential & processes consume potential



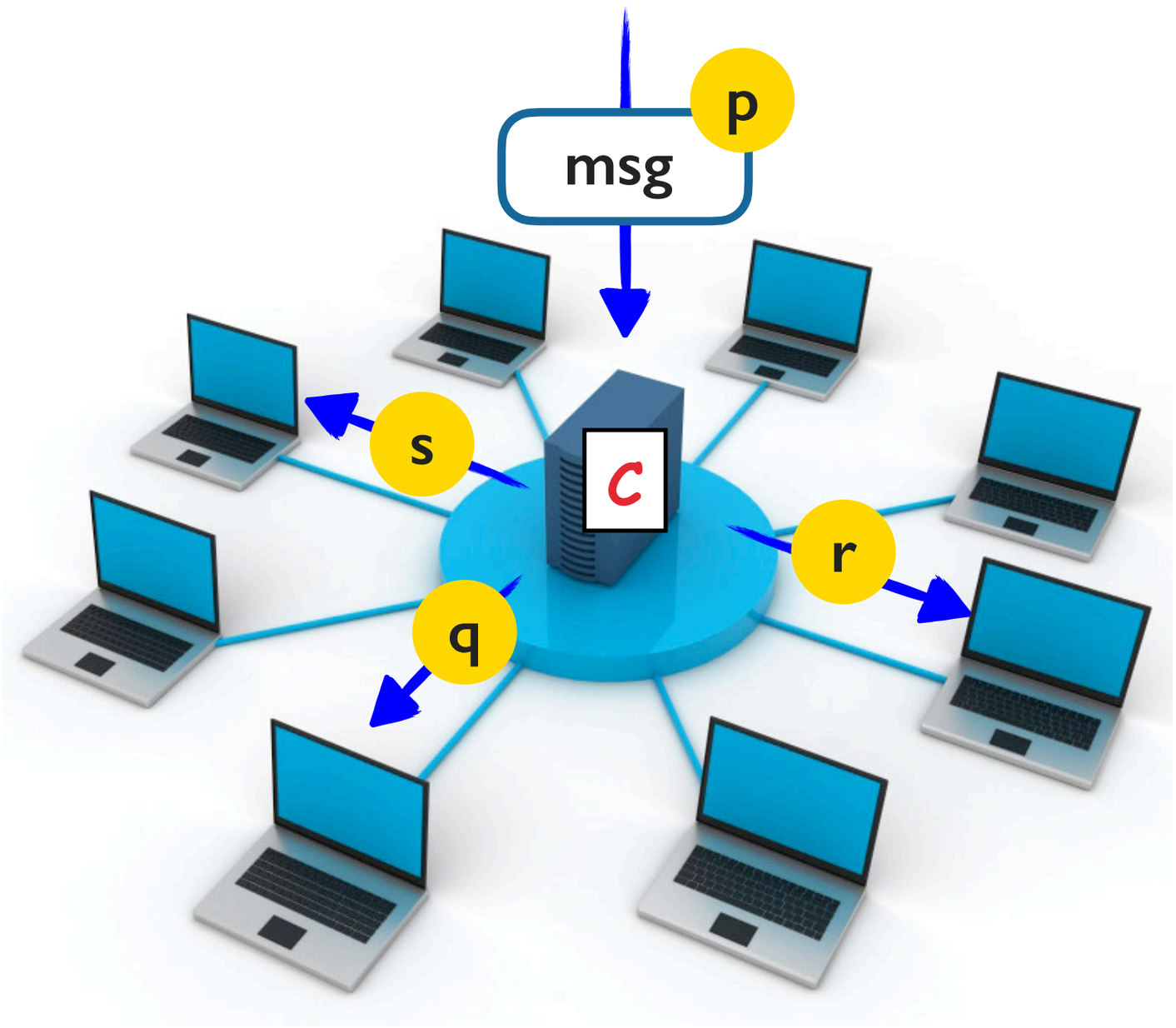
Resource-Aware Session Types

<u>Key Idea:</u> Messages carry potential & processes consume potential



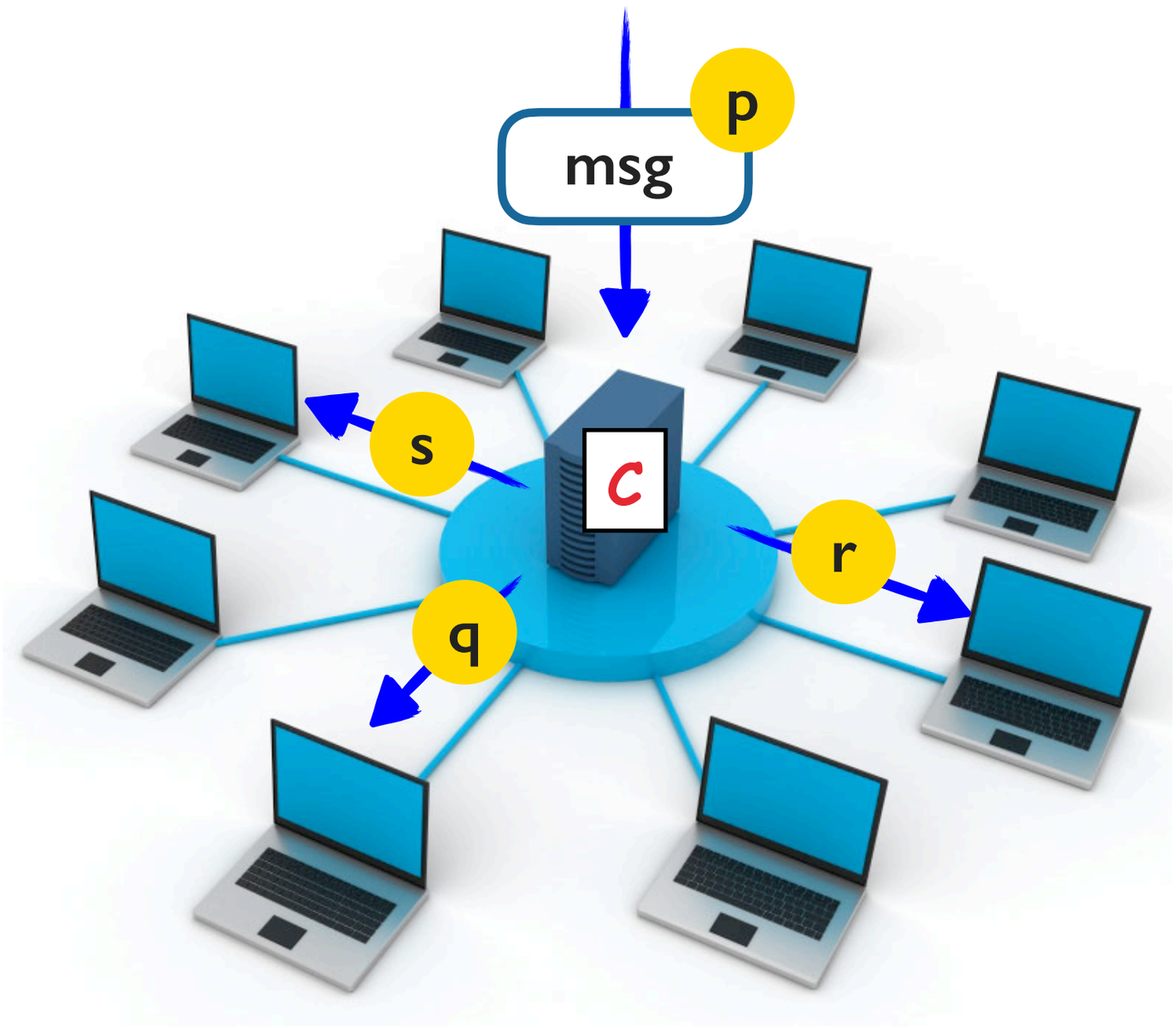
Resource-Aware Session Types

<u>Key Idea:</u> Messages carry potential & processes consume potential



Resource-Aware Session Types

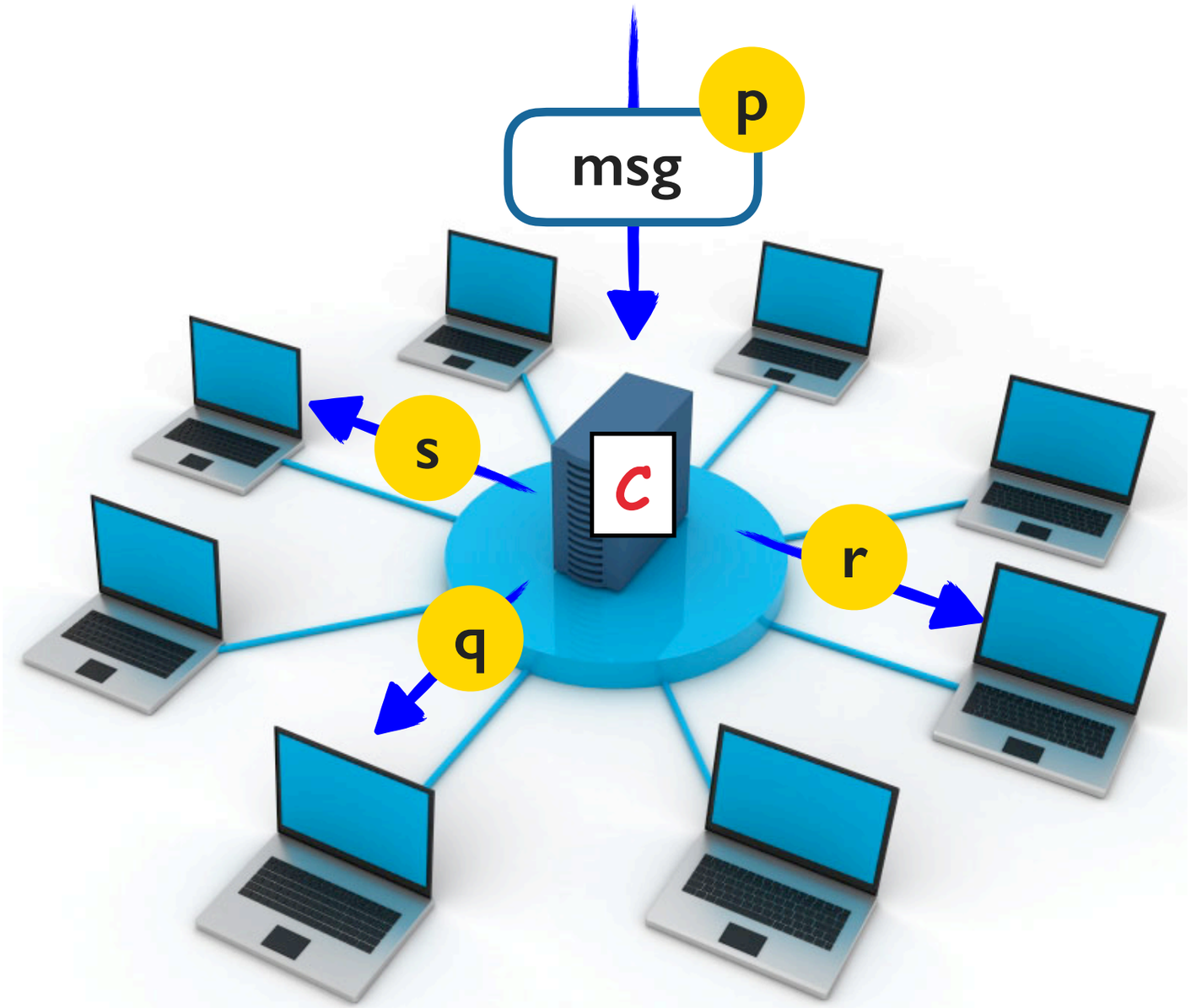
<u>Key Idea:</u> Messages carry potential & processes consume potential



$$p = q + r + s + C$$

Resource-Aware Session Types

<u>Key Idea:</u> Messages carry potential & processes consume potential

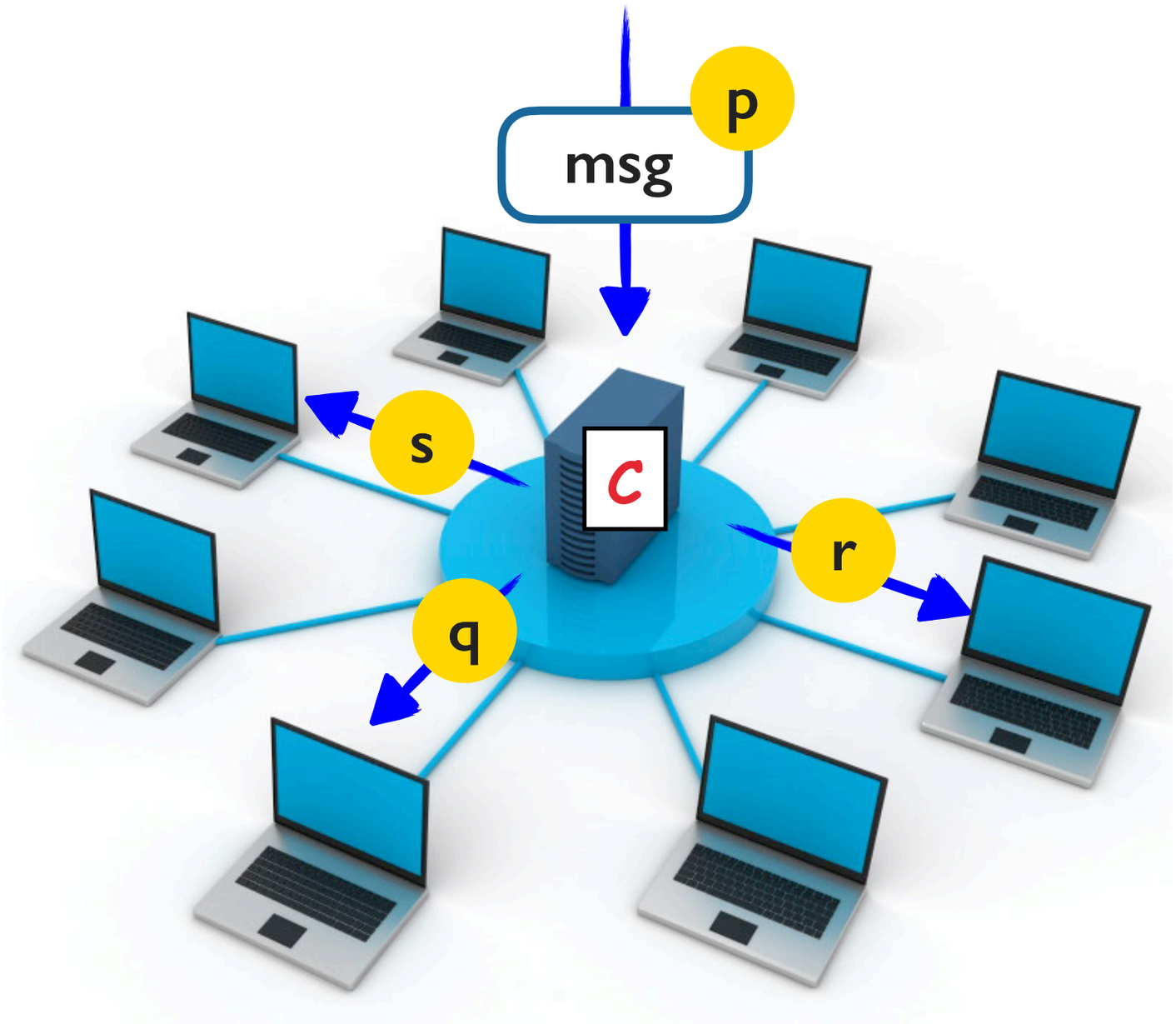


$$p = q + r + s + C$$

$$p, q, r, s \geq 0$$

Resource-Aware Session Types

<u>Key Idea:</u> Messages carry potential & processes consume potential



$$p = q + r + s + C$$

$$p, q, r, s \geq 0$$

$$\bullet \quad \bullet \quad \bullet$$

Talk Outline

Protocol Enforcement & Asset Preservation

Probabilistic Reasoning

Performance & Verification

Nomos

Resource-Aware Session Types

[LICS '18, SAS '20, CSF '21]

NomosPro

Probabilistic Session Types

[POPL '23]

Rast

Refinement Session Types

[ICFP '18, FSCD '20, CONCUR '20]

Session Types

Type system for concurrent message-passing programs

Talk Outline

Protocol Enforcement & Asset Preservation

Probabilistic Reasoning

Performance & Verification

Nomos

Resource-Aware Session Types

[LICS '18, SAS '20, CSF '21]

NomosPro

Probabilistic Session Types

[POPL '23]

Rast

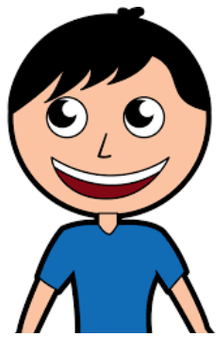
Refinement Session Types

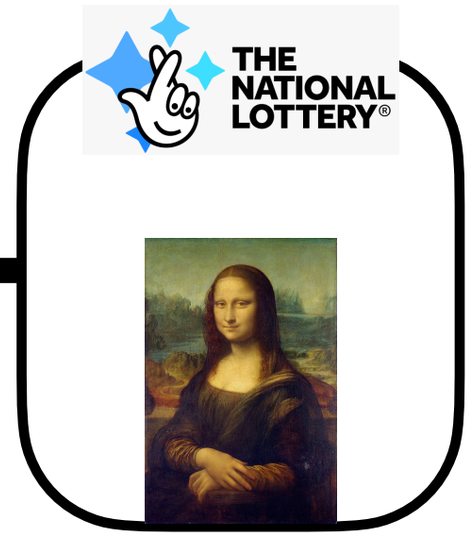
[ICFP '18, FSCD '20, CONCUR '20]

Session Types

Type system for concurrent message-passing programs

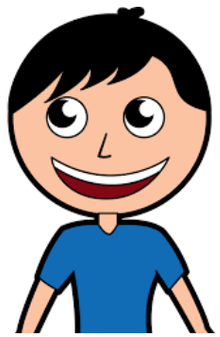
Probabilistic Smart Contracts

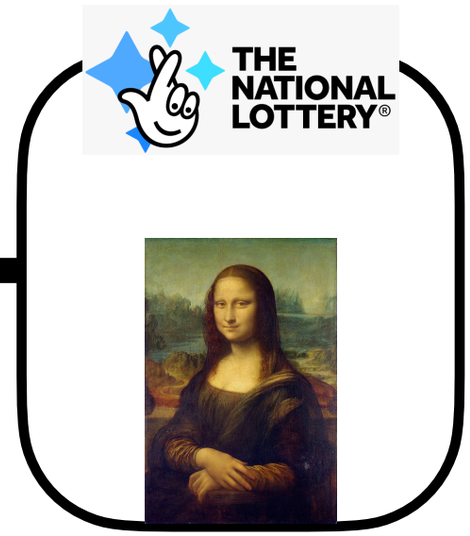




Probabilistic Smart Contracts

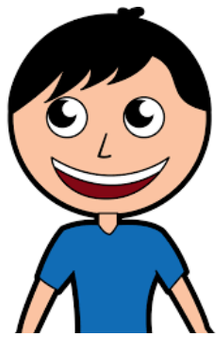
$$\textit{type}\ \text{lottery} = \oplus\{\mathbf{running} : \&\{\mathbf{buy} : \text{money} \multimap \text{lottery}\},$$
$$\mathbf{ended} : \&\{\mathbf{collect} : \oplus\{\mathbf{won} : \text{monalisa} \otimes \text{lottery},$$
$$\mathbf{lost} : \text{lottery}\}\}\}$$





Probabilistic Smart Contracts

$$\textit{type}\ \text{lottery} = \oplus\{\mathbf{running} : \&\{\mathbf{buy} : \text{money} \multimap \text{lottery}\},$$
$$\mathbf{ended} : \&\{\mathbf{collect} : \oplus\{\mathbf{won} : \text{monalisa} \otimes \text{lottery},$$
$$\mathbf{lost} : \text{lottery}\}\}\}$$



What if ‘won’ is sent to nobody?

Probabilistic Smart Contracts

$$\textit{type}\ \text{lottery} = \oplus\{\textbf{running} : \&\{\textbf{buy} : \text{money} \multimap \text{lottery}\},$$
$$\textbf{ended} : \&\{\textbf{collect} : \oplus\{\textbf{won} : \text{monalisa} \otimes \text{lottery},$$
$$\textbf{lost} : \text{lottery}\}\}\}$$

Session Type Does Not Capture Probabilistic Behavior

What if 'won' is sent to nobody?

Probabilistic Session Types

Probabilistic Session Types

<u>Key Idea:</u> NomosPro enhance session types with probabilities to capture prob. distribution of messages

Probabilistic Session Types

Key Idea: *NomosPro enhance session types with probabilities to capture prob. distribution of messages*

$$\begin{aligned} \textit{type}\ \text{lottery} = \oplus\{&\mathbf{running} : \&\{\mathbf{buy} : \text{money} \multimap \text{lottery}\}, \\ &\mathbf{ended} : \&\{\mathbf{collect} : \oplus\{\mathbf{won}^{1/3} : \text{monalisa} \otimes \text{lottery}, \\ &\qquad\qquad \mathbf{lost}^{2/3} : \text{lottery}\}\}\} \end{aligned}$$

Probabilistic Session Types

Key Idea: *NomosPro enhance session types with probabilities to capture prob. distribution of messages*

$$\begin{aligned} \textit{type}\ \text{lottery} = \oplus\{&\mathbf{running} : \&\{\mathbf{buy} : \text{money} \multimap \text{lottery}\}, \\ &\mathbf{ended} : \&\{\mathbf{collect} : \oplus\{\mathbf{won}^{1/3} : \text{monalisa} \otimes \text{lottery}, \\ &\qquad\qquad \mathbf{lost}^{2/3} : \text{lottery}\}\}\} \end{aligned}$$

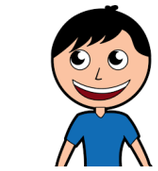
Probabilistic Session Types

Key Idea: *NomosPro enhance session types with probabilities to capture prob. distribution of messages*

$$\textit{type}\ \text{lottery} = \oplus\{\textbf{running} : \&\{\textbf{buy} : \text{money} \multimap \text{lottery}\},$$
$$\textbf{ended} : \&\{\textbf{collect} : \oplus\{\textbf{won}^{1/3} : \text{monalisa} \otimes \text{lottery},$$
$$\textbf{lost}^{2/3} : \text{lottery}\}\}\}$$

Probabilistic Session Types

Key Idea: *NomosPro enhance session types with probabilities to capture prob. distribution of messages*



lost

lost

lost

✗ *compilation error*

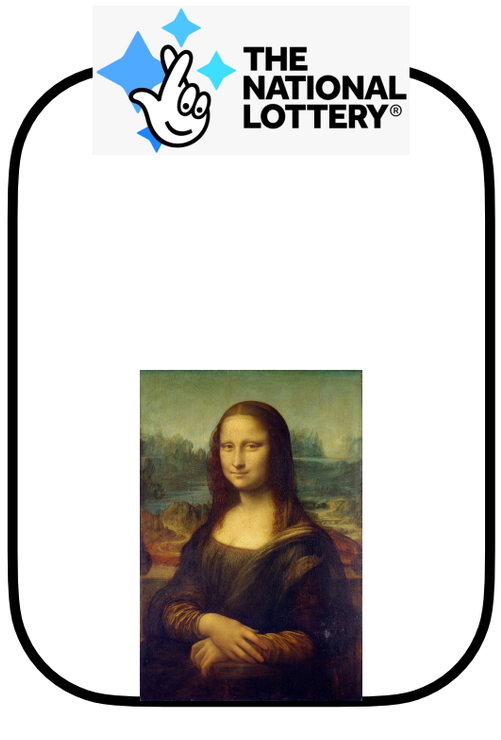
$$\textit{type}\ \text{lottery} = \oplus\{\textbf{running} : \&\{\textbf{buy} : \text{money} \multimap \text{lottery}\},$$
$$\textbf{ended} : \&\{\textbf{collect} : \oplus\{\textbf{won}^{1/3} : \text{monalisa} \otimes \text{lottery},$$
$$\textbf{lost}^{2/3} : \text{lottery}\}\}\}$$

How to Track Probabilities?

$$\textit{type}\ \text{lottery} = \oplus\{\mathbf{running} : \&\{\mathbf{buy} : \text{money} \multimap \text{lottery}\},$$
$$\mathbf{ended} : \&\{\mathbf{collect} : \oplus\{\mathbf{won}^{1/3} : \text{monalisa} \otimes \text{lottery},$$
$$\mathbf{lost}^{2/3} : \text{lottery}\}\}\}$$

How to Track Probabilities?


```
...
let u = uniform(3) ;
case u (
   | 1 ⇒ send lost ; ...

   | 2 ⇒ send lost ; ...

   | 3 ⇒ send lost ; ...
)
...
```



$$\textit{type}\ \text{lottery} = \oplus\{\textbf{running} : \&\{\textbf{buy} : \text{money} \multimap \text{lottery}\},$$
$$\textbf{ended} : \&\{\textbf{collect} : \oplus\{\textbf{won}^{1/3} : \text{monalisa} \otimes \text{lottery},$$
$$\textbf{lost}^{2/3} : \text{lottery}\}\}\}$$

How to Track Probabilities?


```
...
let u = uniform(3) ;
case u (
   | 1 ⇒ send lost ; ...
   | 2 ⇒ send lost ; ...
   | 3 ⇒ send lost ; ...
)
...
```

prob. = 1/3

prob. = 1/3

prob. = 1/3

$$\textit{type}\ \text{lottery} = \oplus\{\textbf{running} : \&\{\textbf{buy} : \text{money} \multimap \text{lottery}\},$$
$$\textbf{ended} : \&\{\textbf{collect} : \oplus\{\textbf{won}^{1/3} : \text{monalisa} \otimes \text{lottery},$$
$$\textbf{lost}^{2/3} : \text{lottery}\}\}\}$$

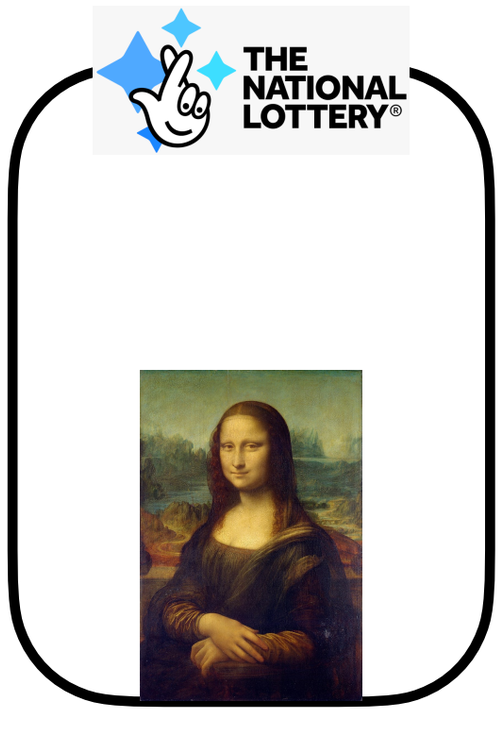
How to Track Probabilities?


```
...
let u = uniform(3) ;
case u (
   | 1 ⇒ send lost ; ...
   | 2 ⇒ send lost ; ...
   | 3 ⇒ send lost ; ...
)
...
```

prob. = 1/3

prob. = 1/3

prob. = 1/3



won [prob. = 0]
lost [prob. = 1]

$$\textit{type}\ \text{lottery} = \oplus\{\textbf{running} : \&\{\textbf{buy} : \text{money} \multimap \text{lottery}\},$$
$$\textbf{ended} : \&\{\textbf{collect} : \oplus\{\textbf{won}^{1/3} : \text{monalisa} \otimes \text{lottery},$$
$$\textbf{lost}^{2/3} : \text{lottery}\}\}\}$$

How to Track Probabilities?


```
...
let u = uniform(3) ;
case u (
   | 1 ⇒ send lost ; ...
   | 2 ⇒ send lost ; ...
   | 3 ⇒ send lost ; ...
)
...
```

prob. = 1/3

prob. = 1/3

prob. = 1/3

won [prob. = 0]
lost [prob. = 1]

✖ *compilation error*

$$\textit{type}\ \text{lottery} = \oplus\{\mathbf{running} : \&\{\mathbf{buy} : \text{money} \multimap \text{lottery}\},$$
$$\mathbf{ended} : \&\{\mathbf{collect} : \oplus\{\mathbf{won}^{1/3} : \text{monalisa} \otimes \text{lottery},$$
$$\mathbf{lost}^{2/3} : \text{lottery}\}\}\}$$

Correct Lottery


```
...
let u = uniform(3) ;
case u (
   | 1 ⇒ send won ; ...

   | 2 ⇒ send lost ; ...

   | 3 ⇒ send lost ; ...
)
...
```

$$\textit{type}\ \text{lottery} = \oplus\{\textbf{running} : \&\{\textbf{buy} : \text{money} \multimap \text{lottery}\},$$
$$\textbf{ended} : \&\{\textbf{collect} : \oplus\{\textbf{won}^{1/3} : \text{monalisa} \otimes \text{lottery},$$
$$\textbf{lost}^{2/3} : \text{lottery}\}\}\}$$

Correct Lottery


```
...
let u = uniform(3) ;
case u (
   | 1 ⇒ send won ; ...
   | 2 ⇒ send lost ; ...
   | 3 ⇒ send lost ; ...
)
...
```

prob. = 1/3

prob. = 1/3

prob. = 1/3

$$\textit{type}\ \text{lottery} = \oplus\{\textbf{running} : \&\{\textbf{buy} : \text{money} \multimap \text{lottery}\},$$
$$\textbf{ended} : \&\{\textbf{collect} : \oplus\{\textbf{won}^{1/3} : \text{monalisa} \otimes \text{lottery},$$
$$\textbf{lost}^{2/3} : \text{lottery}\}\}\}$$

Correct Lottery


```
...
let u = uniform(3) ;
case u (
   | 1 ⇒ send won ; ...
   | 2 ⇒ send lost ; ...
   | 3 ⇒ send lost ; ...
)
...
```

prob. = 1/3

prob. = 1/3

prob. = 1/3

won [prob. = 1/3]
lost [prob. = 2/3]

$$\textit{type}\ \text{lottery} = \oplus\{\textbf{running} : \&\{\textbf{buy} : \text{money} \multimap \text{lottery}\},$$
$$\textbf{ended} : \&\{\textbf{collect} : \oplus\{\textbf{won}^{1/3} : \text{monalisa} \otimes \text{lottery},$$
$$\textbf{lost}^{2/3} : \text{lottery}\}\}\}$$

Correct Lottery


```
...
let u = uniform(3) ;
case u (
   | 1 ⇒ send won ; ...
   | 2 ⇒ send lost ; ...
   | 3 ⇒ send lost ; ...
)
...
```

prob. = 1/3

prob. = 1/3

prob. = 1/3

won [prob. = 1/3]
lost [prob. = 2/3]

✓ *compilation successful*

$$\textit{type}\ \text{lottery} = \oplus\{\textbf{running} : \&\{\textbf{buy} : \text{money} \multimap \text{lottery}\},$$
$$\textbf{ended} : \&\{\textbf{collect} : \oplus\{\textbf{won}^{1/3} : \text{monalisa} \otimes \text{lottery},$$
$$\textbf{lost}^{2/3} : \text{lottery}\}\}\}$$

Probability Inference is Automatic!

$$\textit{type}\ \text{lottery} = \oplus\{\mathbf{running} : \&\{\mathbf{buy} : \text{money} \multimap \text{lottery}\},$$
$$\mathbf{ended} : \&\{\mathbf{collect} : \oplus\{\mathbf{won}^{*} : \text{monalisa} \otimes \text{lottery},$$
$$\mathbf{lost}^{*} : \text{lottery}\}\}\}$$

Probability Inference is Automatic!

$$\textit{type}\ \text{lottery} = \oplus\{\textbf{running} : \&\{\textbf{buy} : \text{money} \multimap \text{lottery}\},$$
$$\textbf{ended} : \&\{\textbf{collect} : \oplus\{\textbf{won}^{*} : \text{monalisa} \otimes \text{lottery},$$
$$\textbf{lost}^{*} : \text{lottery}\}\}\}$$

Linear constraints shipped to LP solver


```
...
let u = uniform(3) ;
case u (
  | 1 ⇒ send won ; ...

  | 2 ⇒ send lost ; ...

  | 3 ⇒ send lost ; ...
)
...
```

Probability Inference is Automatic!

$$\textit{type}\ \text{lottery} = \oplus\{\textbf{running} : \&\{\textbf{buy} : \text{money} \multimap \text{lottery}\},$$
$$\textbf{ended} : \&\{\textbf{collect} : \oplus\{\textbf{won}^{*} : \text{monalisa} \otimes \text{lottery},$$
$$\textbf{lost}^{*} : \text{lottery}\}\}\}$$

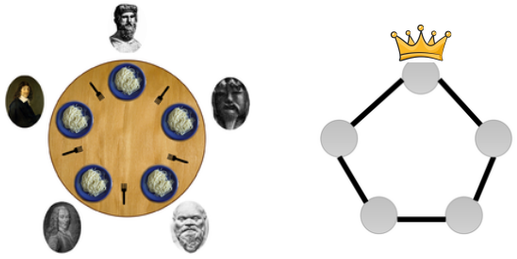
Linear constraints
shipped to LP solver

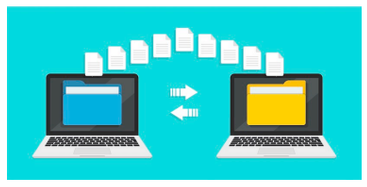

```
...
let u = uniform(3) ;
case u (
  | 1 ⇒ send won ; ...

  | 2 ⇒ send lost ; ...

  | 3 ⇒ send lost ; ...
)
...
```

$$\textit{type}\ \text{lottery} = \oplus\{\textbf{running} : \&\{\textbf{buy} : \text{money} \multimap \text{lottery}\},$$
$$\textbf{ended} : \&\{\textbf{collect} : \oplus\{\textbf{won}^{1/3} : \text{monalisa} \otimes \text{lottery},$$
$$\textbf{lost}^{2/3} : \text{lottery}\}\}\}$$

Applications of NomosPro

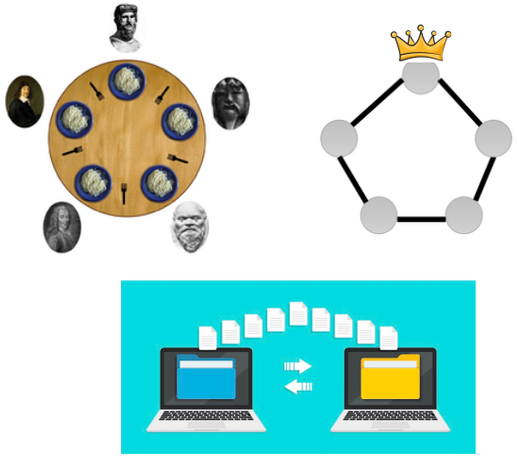
Applications of NomosPro



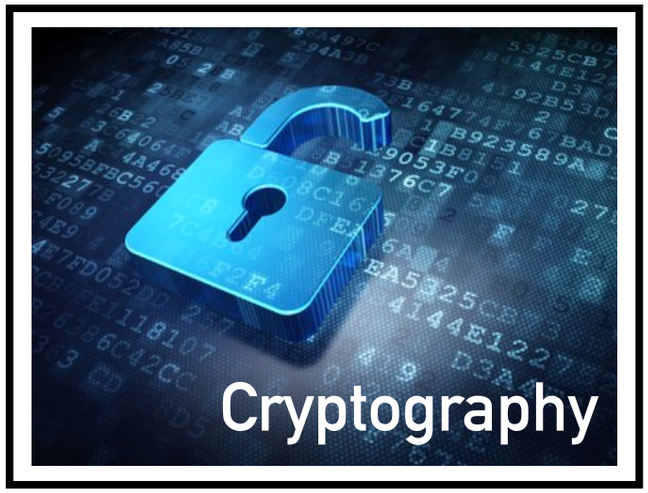


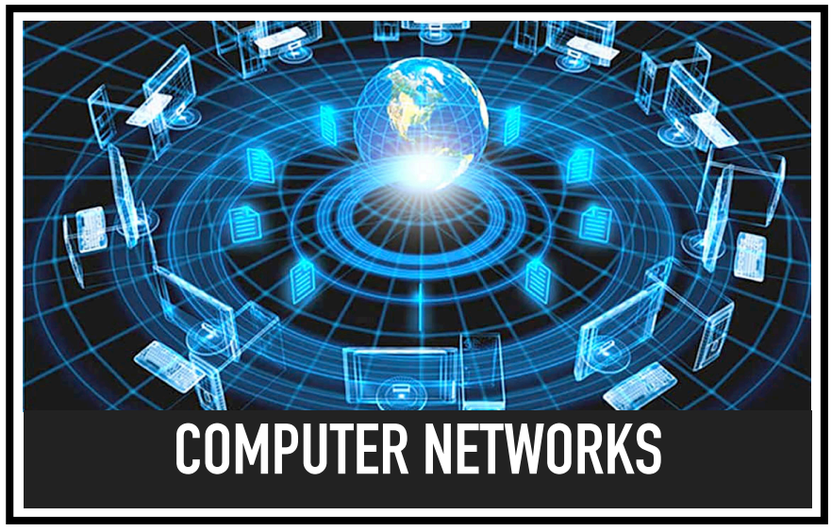
Randomized Algorithms

Applications of NomosPro

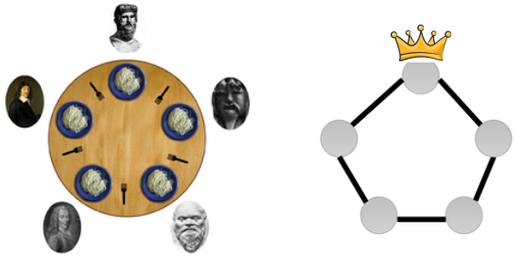


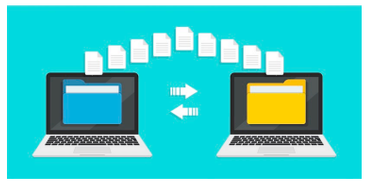
Randomized Algorithms



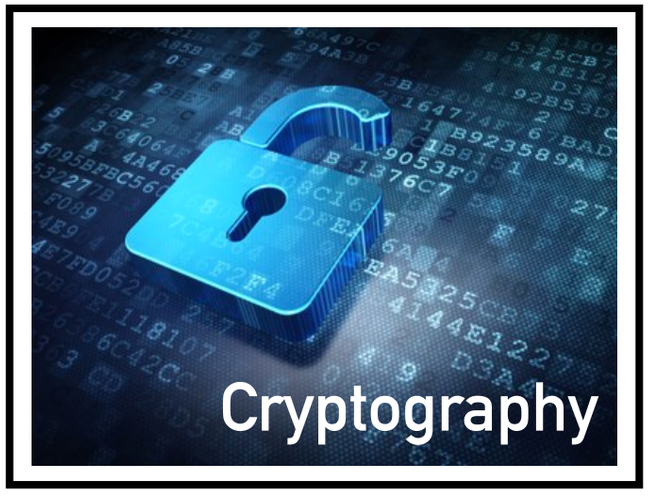


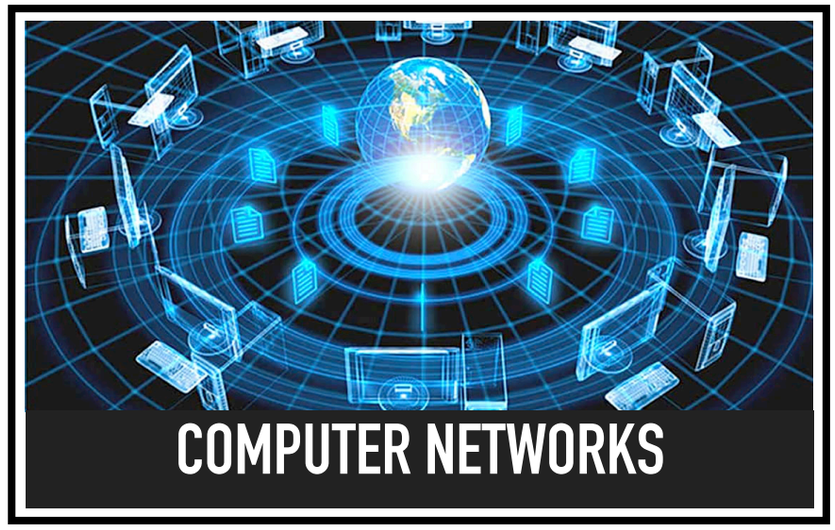
Applications of NomosPro

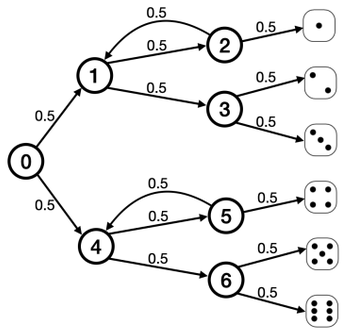


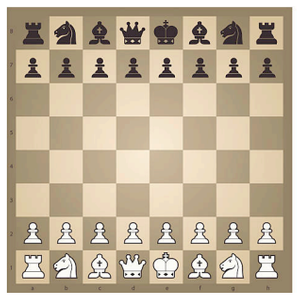


Randomized Algorithms

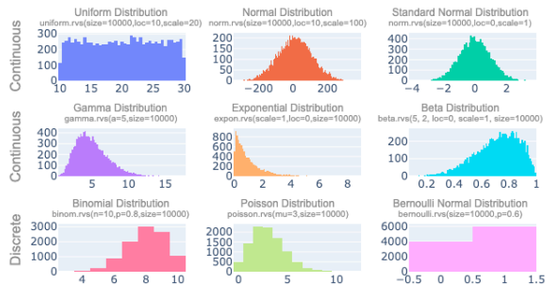






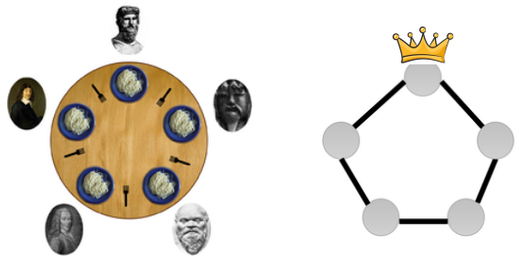


Markov Chains

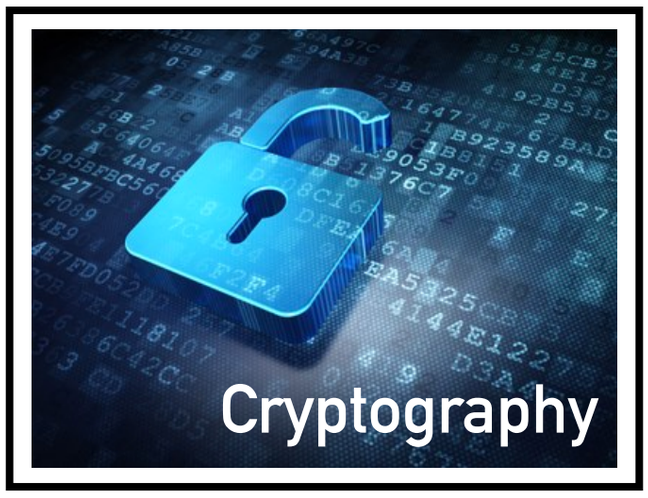


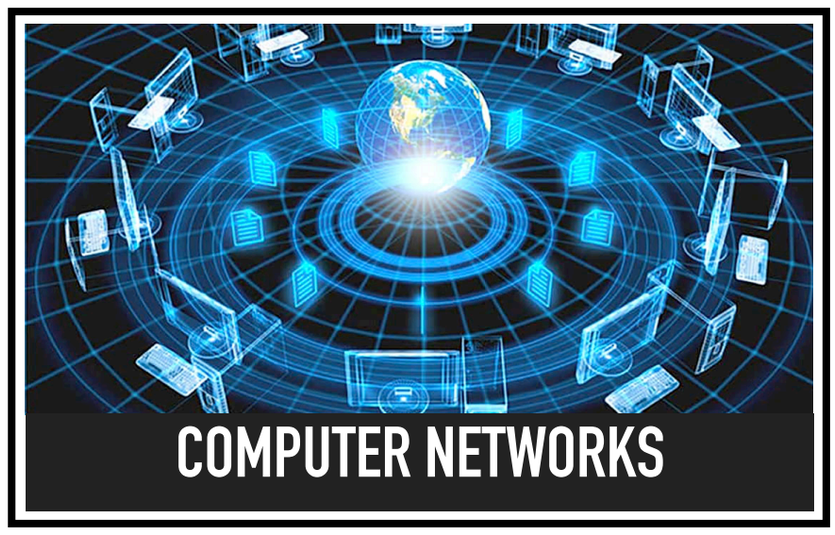
Modeling Unknown Distributions

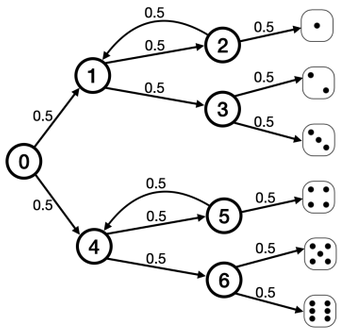
Applications of NomosPro

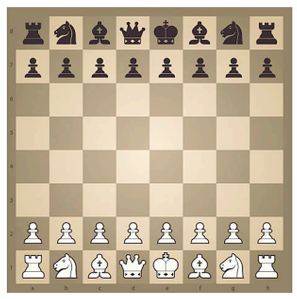


Randomized Algorithms

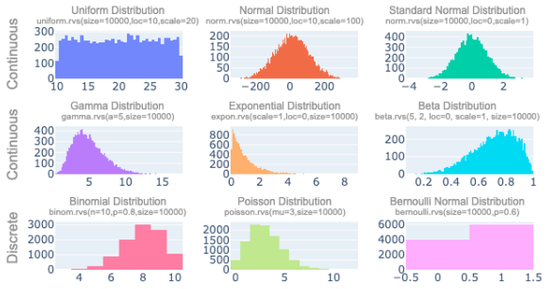








Markov Chains



Modeling Unknown Distributions

Talk Outline

Protocol Enforcement & Asset Preservation

Probabilistic Reasoning

Performance & Verification

Nomos

Resource-Aware Session Types

[LICS '18, SAS '20, CSF '21]

NomosPro

Probabilistic Session Types

[POPL '23]

Rast

Refinement Session Types

[ICFP '18, FSCD '20, CONCUR '20]

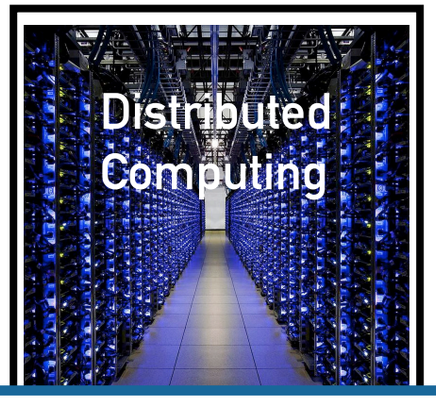
Session Types

Type system for concurrent message-passing programs

Talk Outline

Protocol Enforcement & Asset Preservation

Probabilistic Reasoning



Performance & Verification

Nomos

Resource-Aware Session Types

[LICS '18, SAS '20, CSF '21]

NomosPro

Probabilistic Session Types

[POPL '23]

Rast

Refinement Session Types

[ICFP '18, FSCD '20, CONCUR '20]

Session Types

Type system for concurrent message-passing programs

Who Wins the Auction?


```
function bid() public payable {
    uint bid = msg.value;
    address bidder = msg.sender;
    pendingReturns[bidder] = bid;
    if (bid > highestBid) {
        highestBidder = bidder;
        highestBid = bid;
    }
}
```

$$\begin{aligned} \mathit{type}\ \text{auction} = \oplus\{\mathbf{running} : \&\{\mathbf{bid} : \text{money} \multimap \text{auction}\}, \\ \mathbf{ended} : \&\{\mathbf{collect} : \oplus\{\mathbf{won} : \text{monalisa} \otimes \text{auction}, \\ \mathbf{lost} : \text{money} \otimes \text{auction}\}\}\} \end{aligned}$$

Who Wins the Auction?


```
function bid() public payable {
    uint bid = msg.value;
    address bidder = msg.sender;
    pendingReturns[bidder] = bid;
    if (bid > highestBid) {
        highestBidder = bidder;
        highestBid = bid;
    }
}
```

$$\mathit{type}\ \text{auction} = \oplus\{\mathbf{running} : \&\{\mathbf{bid} : \text{money} \multimap \text{auction}\},$$
$$\mathbf{ended} : \&\{\mathbf{collect} : \oplus\{\mathbf{won} : \text{monalisa} \otimes \text{auction},$$
$$\mathbf{lost} : \text{money} \otimes \text{auction}\}\}\}$$

Who Wins the Auction?


```
function bid() public payable {
    uint bid = msg.value;
    address bidder = msg.sender;
    pendingReturns[bidder] = bid;
    if (bid > highestBid) {
        highestBidder = bidder;
        highestBid = bid;
    }
}
```

What if bid < highestBid*?*

$$\mathit{type}\ \text{auction} = \oplus\{\mathbf{running} : \&\{\mathbf{bid} : \text{money} \multimap \text{auction}\},$$
$$\mathbf{ended} : \&\{\mathbf{collect} : \oplus\{\mathbf{won} : \text{monalisa} \otimes \text{auction},$$
$$\mathbf{lost} : \text{money} \otimes \text{auction}\}\}\}$$

Who Wins the Auction?


```
function bid() public payable {
    uint bid = msg.value;
    address bidder = msg.sender;
    pendingReturns[bidder] = bid;
    if (bid > highestBid) {
        highestBidder = bidder;
        highestBid = bid;
    }
}
```


What if bid < highestBid?

Session Type Does Not Encode Logical Constraints

$$type\ \text{auction} = \oplus\{\mathbf{running} : \&\{\mathbf{bid} : \text{money} \multimap \text{auction}\},$$
$$\mathbf{ended} : \&\{\mathbf{collect} : \oplus\{\mathbf{won} : \text{monalisa} \otimes \text{auction},$$
$$\mathbf{lost} : \text{money} \otimes \text{auction}\}\}\}$$

Tracking Highest Bid in Type

Tracking Highest Bid in Type

<u>Key Idea:</u> Session types in Rast are indexed with integers capturing arithmetic properties

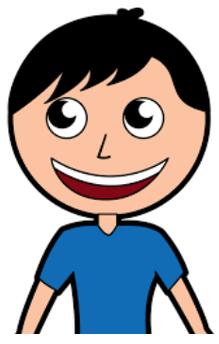
Tracking Highest Bid in Type

<u>Key Idea:</u> Session types in Rast are indexed with integers capturing arithmetic properties

$$\textit{type}\ \text{auction}[\textit{max}] = \oplus\{\textbf{running} : \&\{\textbf{bid} : \text{money}[m \mid m > \textit{max}] \multimap \text{auction}[m]\},$$
$$\textbf{ended} : \&\{\textbf{collect} : \oplus\{\textbf{won} : \text{monalisa} \otimes \text{auction}[\textit{max}],$$
$$\textbf{lost} : \text{money}[m \mid m < \textit{max}] \otimes \text{auction}[\textit{max}]\}\}\}$$

Tracking Highest Bid in Type

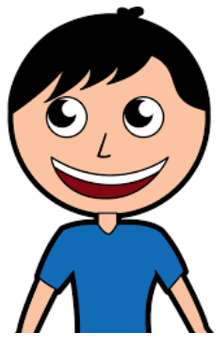
running phase



$$\textit{type}\ \text{auction}[\textit{max}] = \oplus\{\textbf{running} : \&\{\textbf{bid} : \text{money}[m \mid m > \textit{max}] \multimap \text{auction}[m]\},$$
$$\textbf{ended} : \&\{\textbf{collect} : \oplus\{\textbf{won} : \text{monalisa} \otimes \text{auction}[\textit{max}],$$
$$\textbf{lost} : \text{money}[m \mid m < \textit{max}] \otimes \text{auction}[\textit{max}]\}\}\}$$

Tracking Highest Bid in Type

running phase

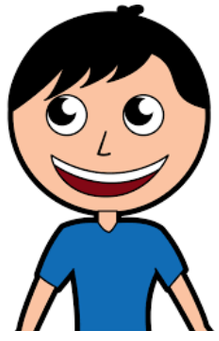


[*max*]

$$\mathit{type}\ \text{auction}[\mathit{max}] = \oplus\{\textbf{running} : \&\{\textbf{bid} : \text{money}[m \mid m > \mathit{max}] \multimap \text{auction}[m]\},$$
$$\textbf{ended} : \&\{\textbf{collect} : \oplus\{\textbf{won} : \text{monalisa} \otimes \text{auction}[\mathit{max}],$$
$$\textbf{lost} : \text{money}[m \mid m < \mathit{max}] \otimes \text{auction}[\mathit{max}]\}\}\}$$

Tracking Highest Bid in Type

running phase



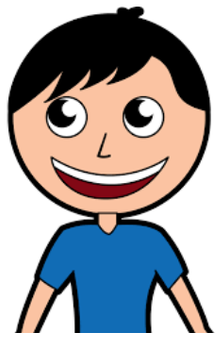
money $[m \mid m > max]$

$[max]$

$$type\ \text{auction}[max] = \oplus\{\textbf{running} : \&\{\textbf{bid} : \text{money}[m \mid m > max] \multimap \text{auction}[m]\},$$
$$\textbf{ended} : \&\{\textbf{collect} : \oplus\{\textbf{won} : \text{monalisa} \otimes \text{auction}[max],$$
$$\textbf{lost} : \text{money}[m \mid m < max] \otimes \text{auction}[max]\}\}\}$$

Tracking Highest Bid in Type

running phase

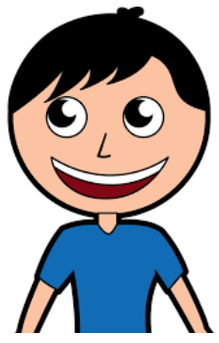


[*max*]

$$\textit{type}\ \text{auction}[\textit{max}] = \oplus\{\textbf{running} : \&\{\textbf{bid} : \text{money}[m \mid m > \textit{max}] \multimap \text{auction}[m]\},$$
$$\textbf{ended} : \&\{\textbf{collect} : \oplus\{\textbf{won} : \text{monalisa} \otimes \text{auction}[\textit{max}],$$
$$\textbf{lost} : \text{money}[m \mid m < \textit{max}] \otimes \text{auction}[\textit{max}]\}\}\}$$

Tracking Highest Bid in Type

running phase



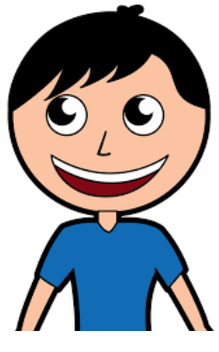
~~[*max*]~~

[*m*]

$$\textit{type}\ \text{auction}[max] = \oplus\{\textbf{running} : \&\{\textbf{bid} : \text{money}[m \mid m > max] \multimap \text{auction}[m]\},$$
$$\textbf{ended} : \&\{\textbf{collect} : \oplus\{\textbf{won} : \text{monalisa} \otimes \text{auction}[max],$$
$$\textbf{lost} : \text{money}[m \mid m < max] \otimes \text{auction}[max]\}\}\}$$

Tracking Highest Bid in Type

running phase



~~[*max*]~~

[*m*]

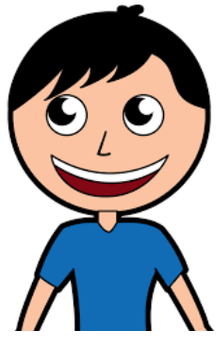
ended phase

[*max*]

$$\textit{type}\ \text{auction}[\textit{max}] = \oplus\{\textbf{running} : \&\{\textbf{bid} : \text{money}[m \mid m > \textit{max}] \multimap \text{auction}[m]\},$$
$$\textbf{ended} : \&\{\textbf{collect} : \oplus\{\textbf{won} : \text{monalisa} \otimes \text{auction}[\textit{max}],$$
$$\textbf{lost} : \text{money}[m \mid m < \textit{max}] \otimes \text{auction}[\textit{max}]\}\}\}$$

Tracking Highest Bid in Type

running phase



~~[*max*]~~

[*m*]

ended phase

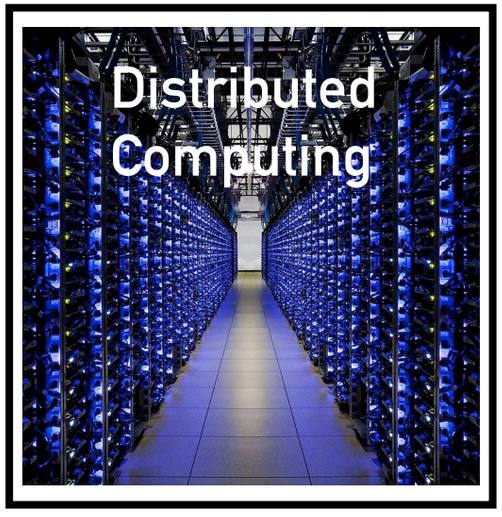
money [*m* | *m* < *max*]

[*max*]

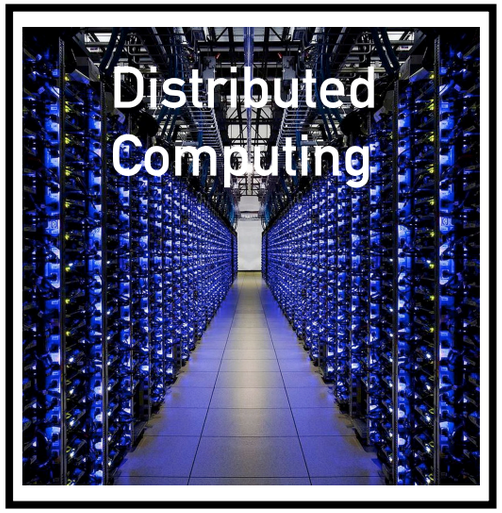
$$\textit{type}\ \text{auction}[\textit{max}] = \oplus\{\textbf{running} : \&\{\textbf{bid} : \text{money}[m \mid m > \textit{max}] \multimap \text{auction}[m]\},$$
$$\textbf{ended} : \&\{\textbf{collect} : \oplus\{\textbf{won} : \text{monalisa} \otimes \text{auction}[\textit{max}],$$
$$\textbf{lost} : \text{money}[m \mid m < \textit{max}] \otimes \text{auction}[\textit{max}]\}\}\}$$

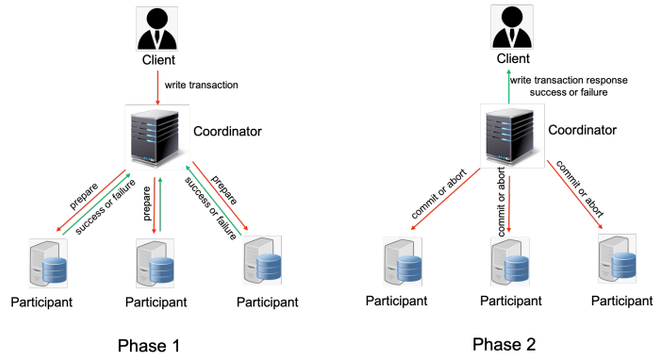
Applications of Rast

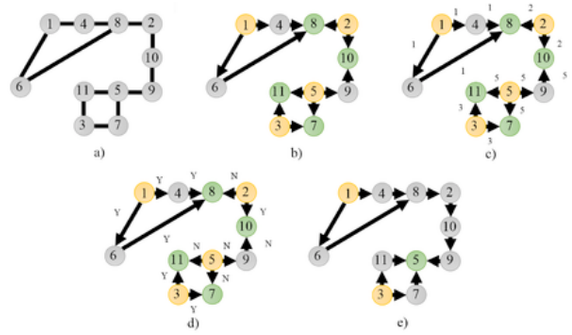
Applications of Rast



Applications of Rast

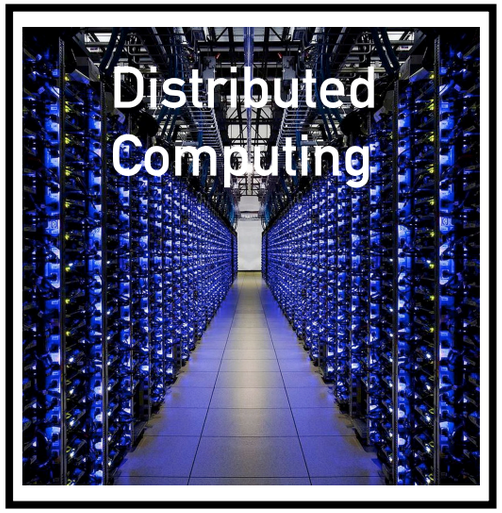


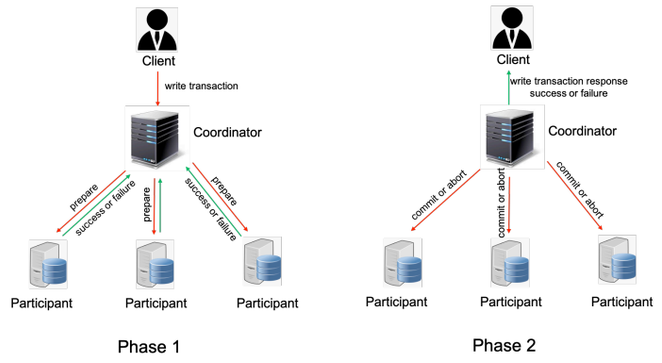


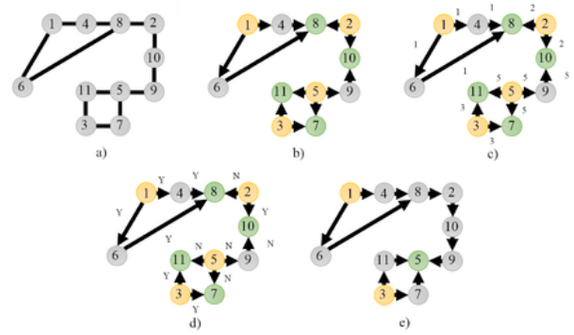


Lightweight Verification of Distributed Algorithms

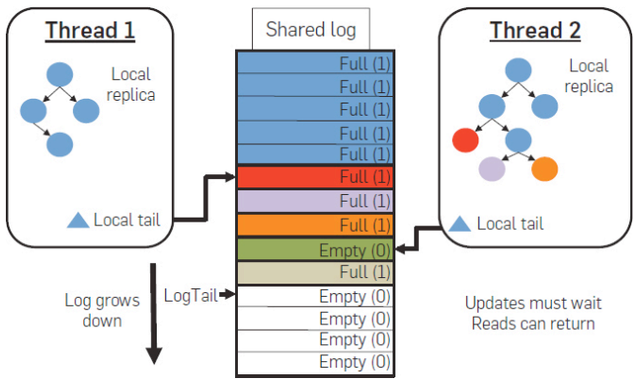
Applications of Rast

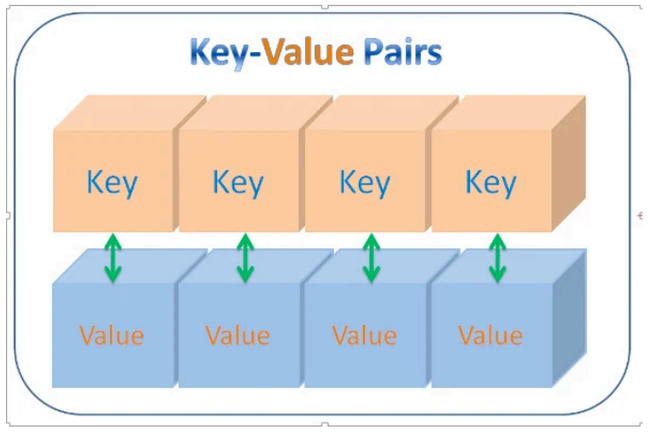


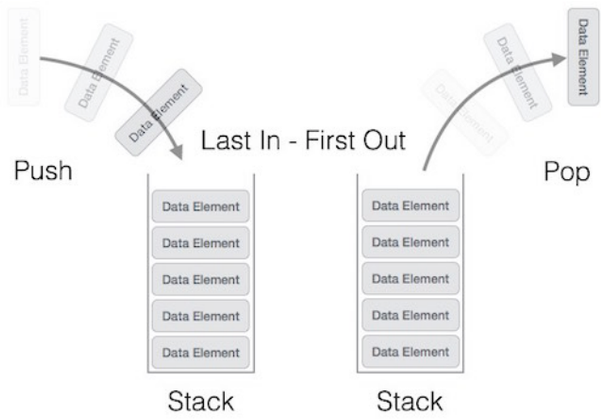




Lightweight Verification of Distributed Algorithms





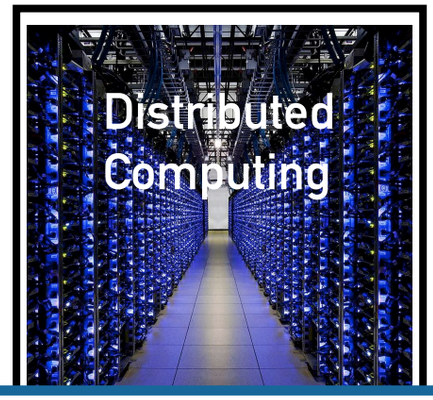


Analysis of Concurrent Data Structures

Key Takeaways

Protocol Enforcement & Asset Preservation

Probabilistic Reasoning



Performance & Verification

Nomos

Resource-Aware Session Types

[LICS '18, SAS '20, CSF '21]

NomosPro

Probabilistic Session Types

[POPL '23]

Rast

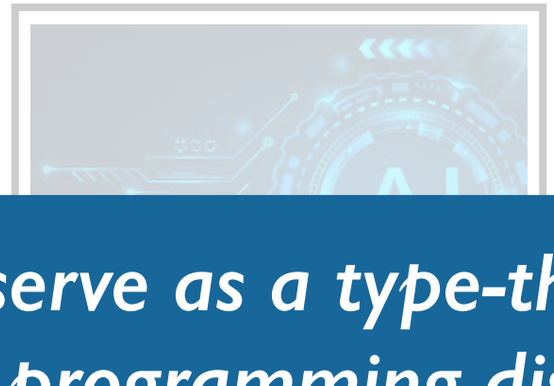
Refinement Session Types

[ICFP '18, FSCD '20, CONCUR '20]

Session Types

Type system for concurrent message-passing programs

Key Takeaways



- *Session Types serve as a type-theoretic foundation for programming distributed systems*
- *Domain-specific abstractions provide domain-specific guarantees*
- *Guarantees are provided by the type itself, which acts as a specification*
- *No need to analyze code once it has compiled!*

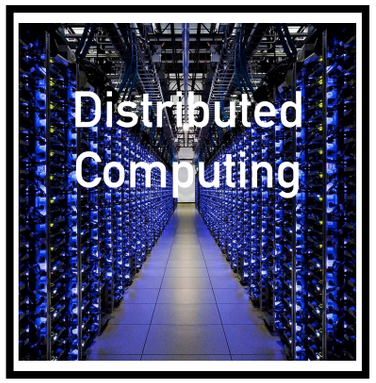
Session Types

Type system for concurrent message-passing programs

Future Research

Adapting PL Techniques to the Distributed Age

Future Research Directions



Nomos

Resource-Aware Session Types

NomosPro

Probabilistic Session Types

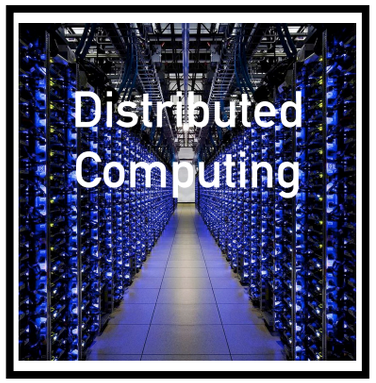
Rast

Refinement Session Types

Session Types

Type system for concurrent message-passing programs

Future Research Directions



More Application Domains …

Nomos

Resource-Aware Session Types

NomosPro

Probabilistic Session Types

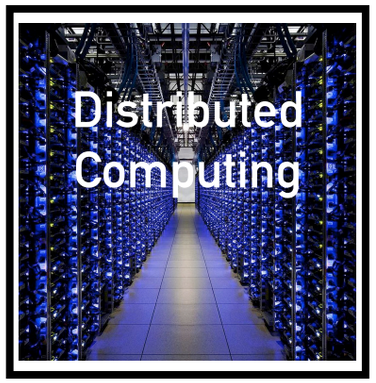
Rast

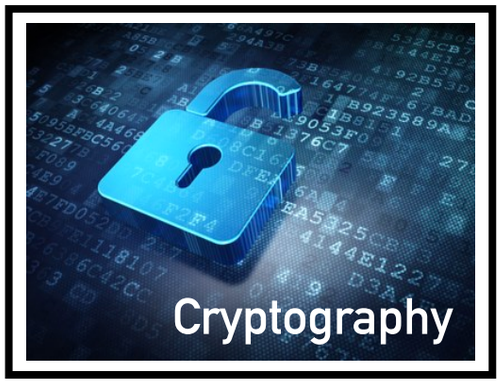
Refinement Session Types

Session Types

Type system for concurrent message-passing programs

Future Research Directions





More Application Domains …

Nomos

Resource-Aware Session Types

NomosPro

Probabilistic Session Types

Rast

Refinement Session Types

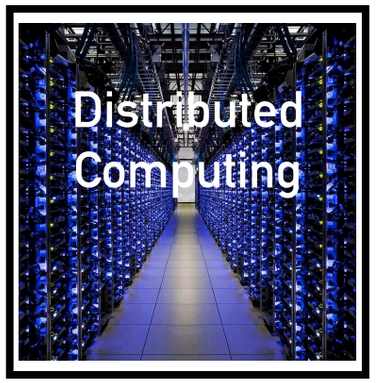
NomosUC

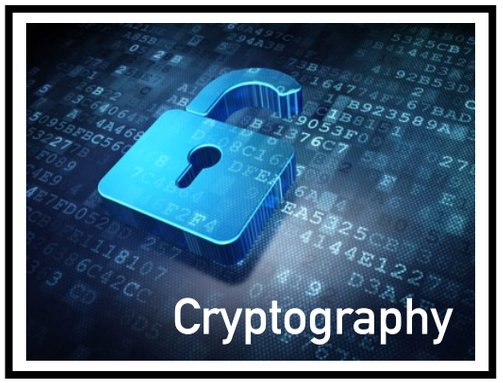
Import Session Types

Session Types

Type system for concurrent message-passing programs

Future Research Directions





More Application Domains …

Nomos

Resource-Aware Session Types

NomosPro

Probabilistic Session Types

Rast

Refinement Session Types

NomosUC

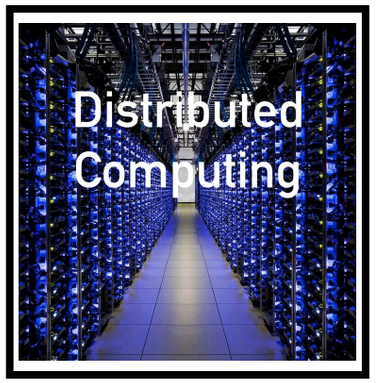
Import Session Types

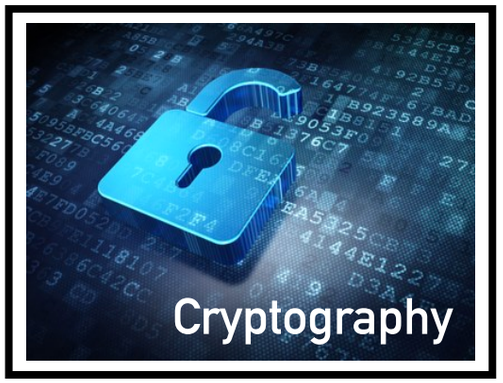
Session Types

Type system for concurrent message-passing programs

Dependent Session Types

Future Research Directions





More Application Domains …

Nomos

Resource-Aware Session Types

NomosPro

Probabilistic Session Types

Rast

Refinement Session Types

NomosUC

Import Session Types

Session Types

Type system for concurrent message-passing programs

Dependent Session Types

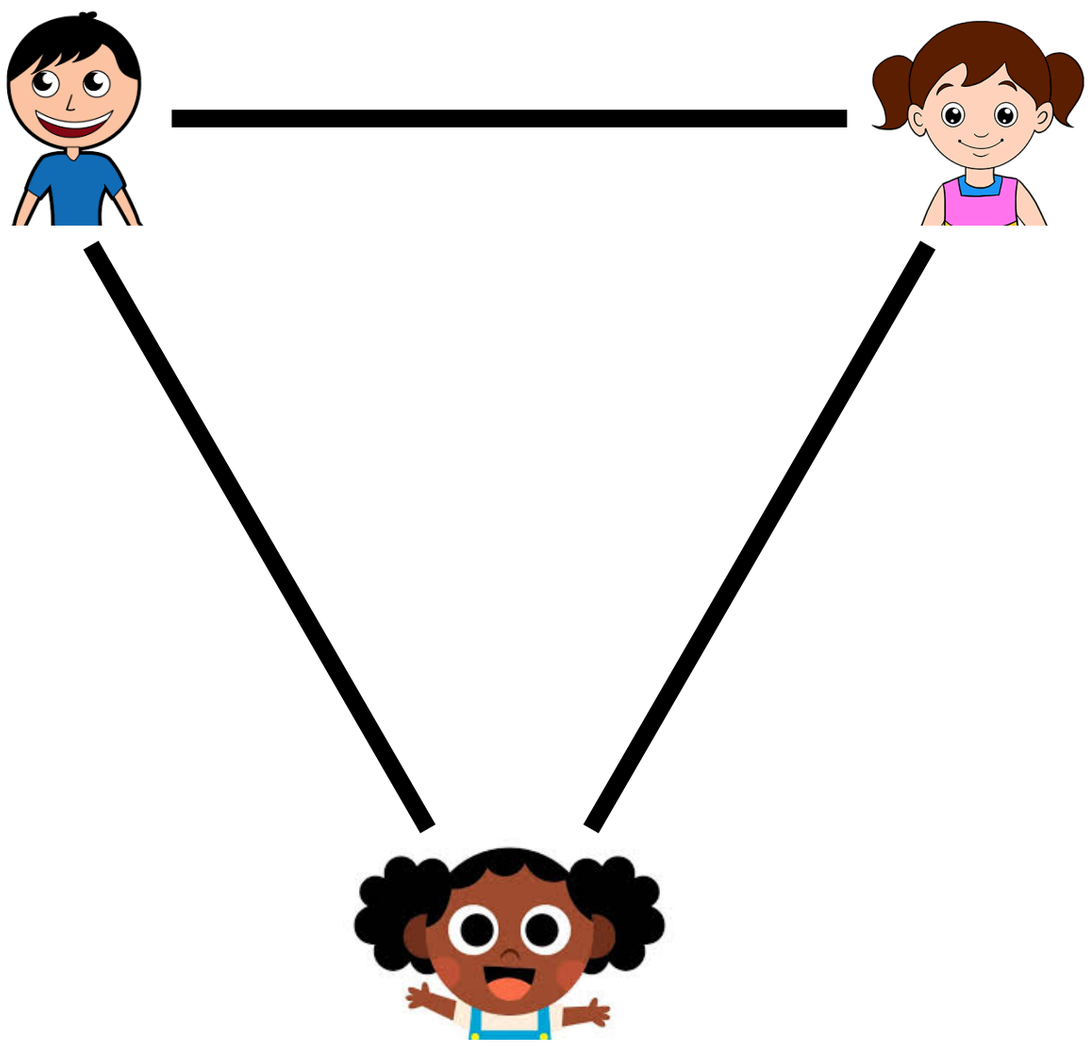
Encoding Failures, Timeouts

Cryptographic Protocols

SSL/TLS

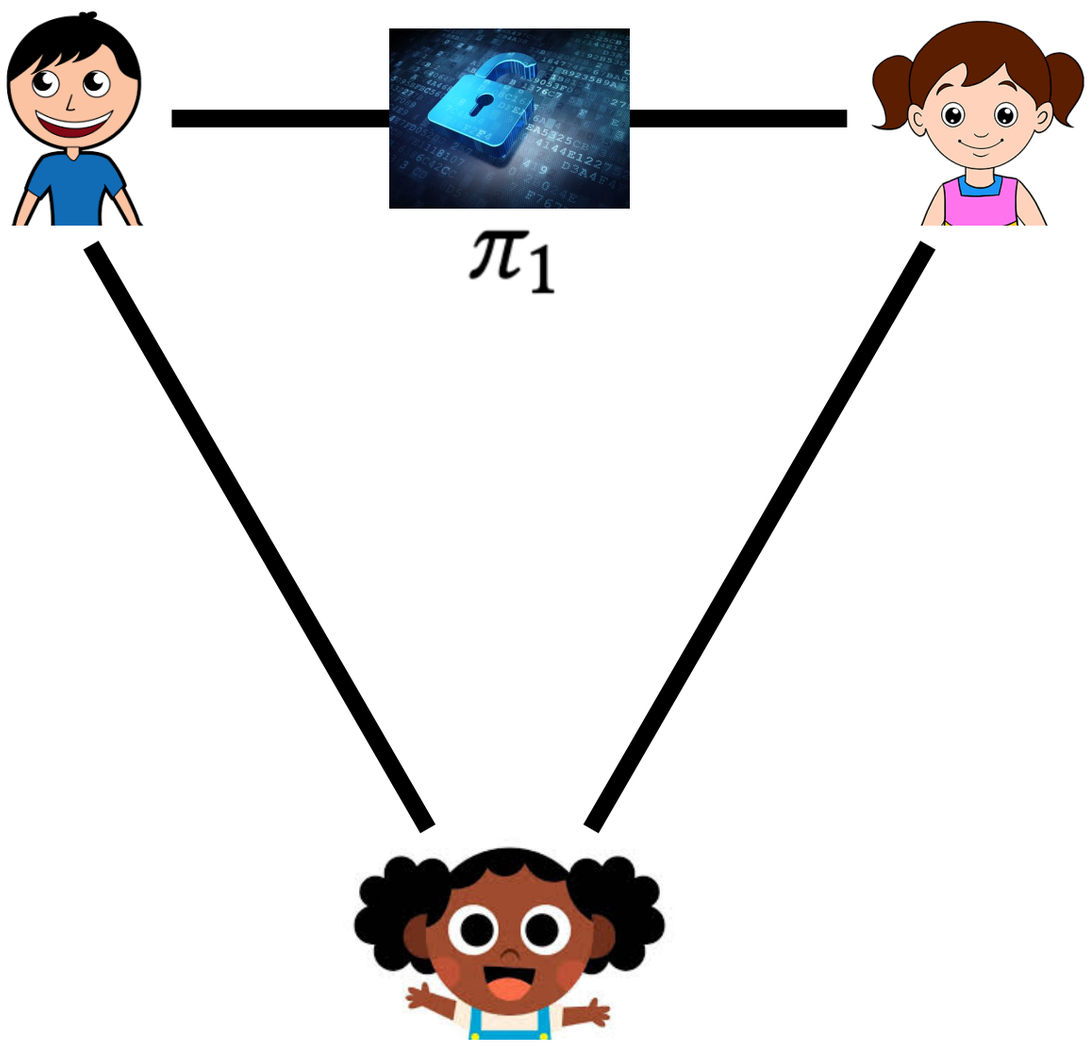
Cryptographic Protocols

SSL/TLS



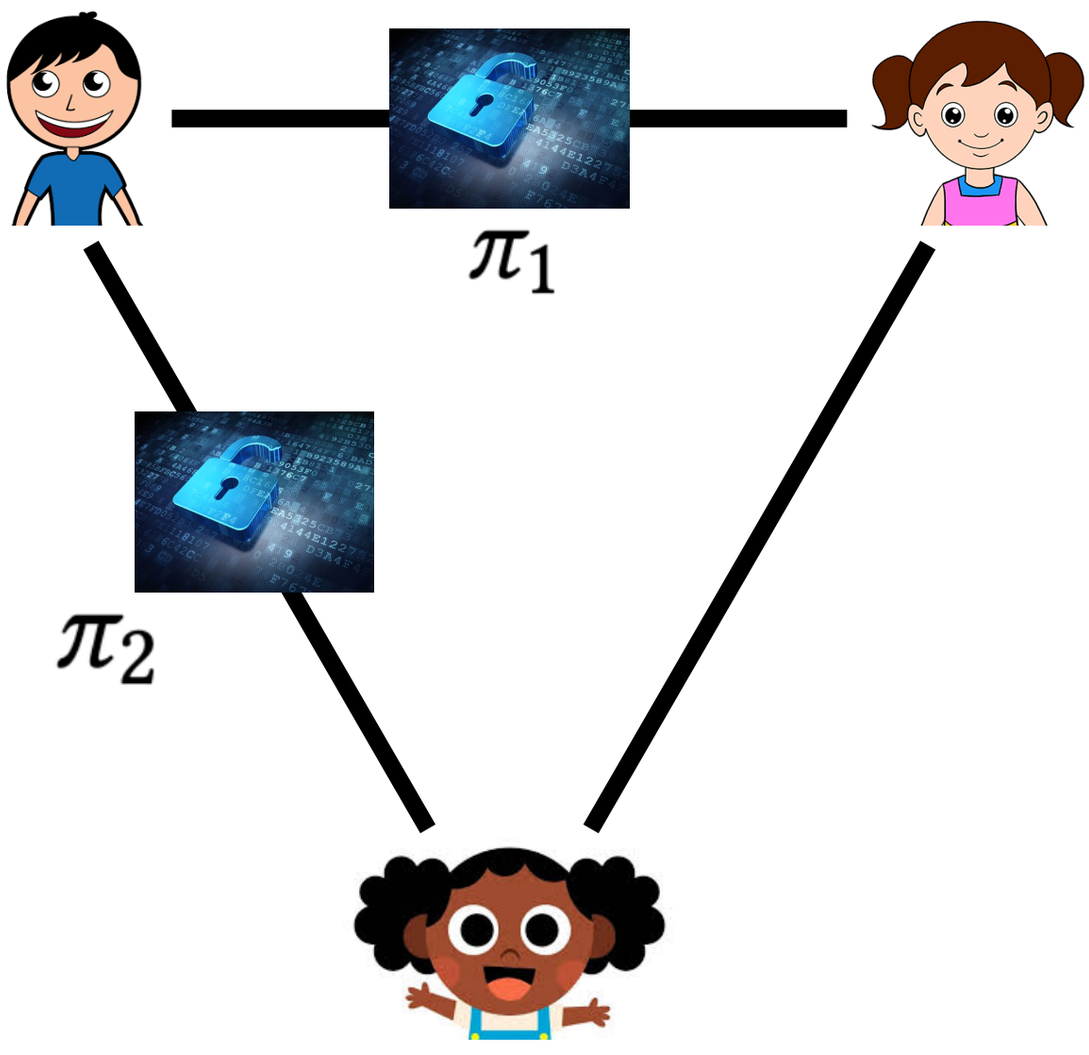
Cryptographic Protocols

SSL/TLS



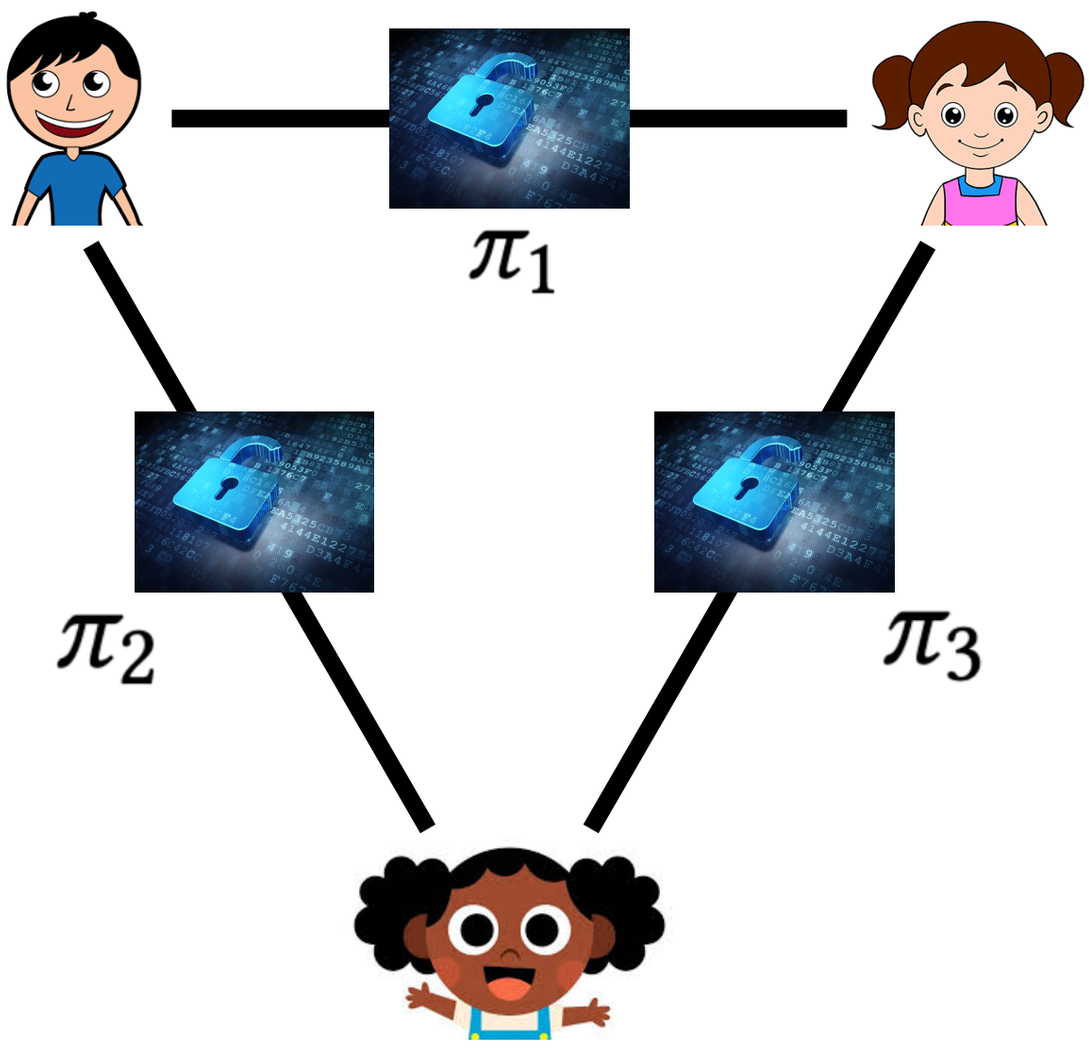
Cryptographic Protocols

SSL/TLS



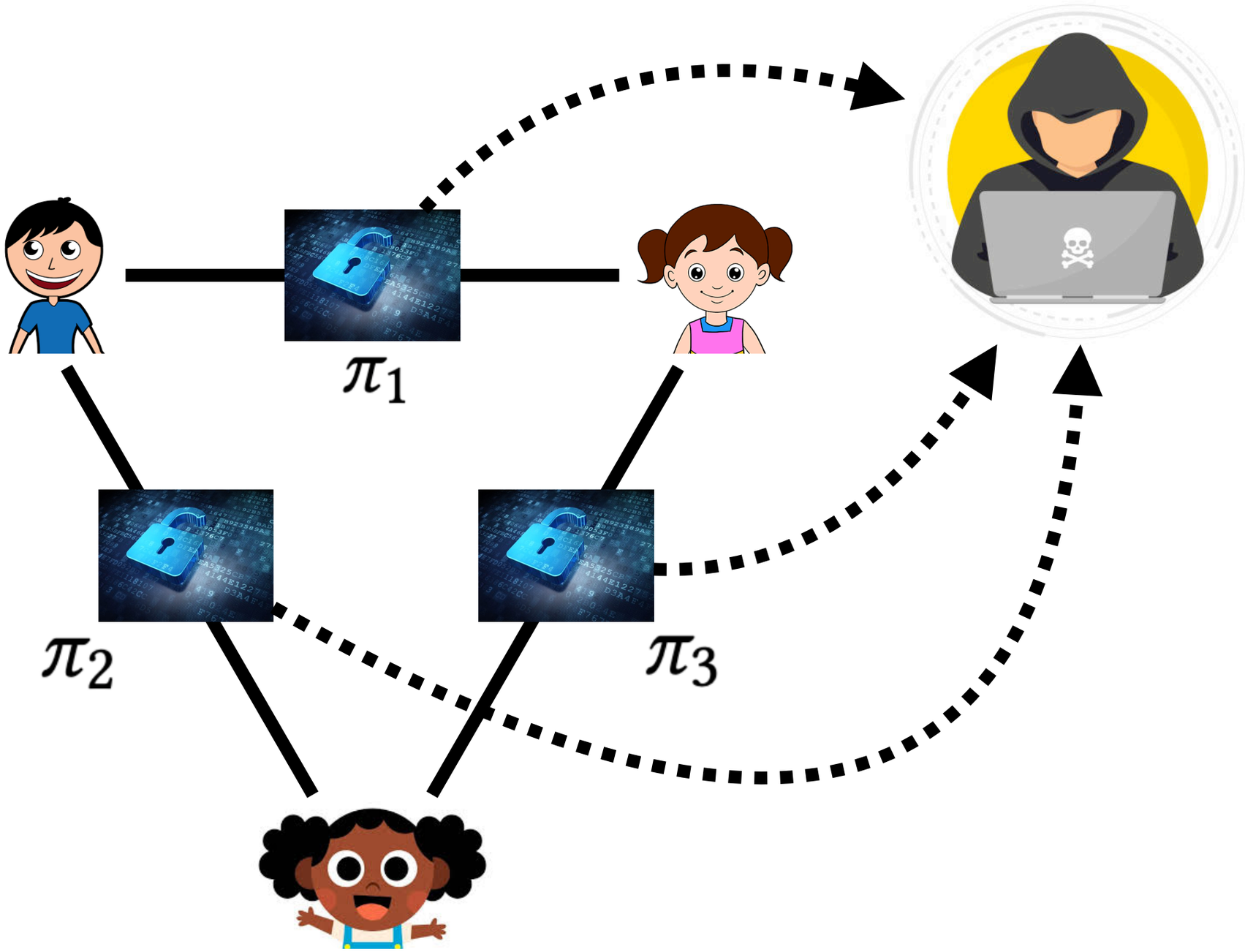
Cryptographic Protocols

SSL/TLS



Cryptographic Protocols

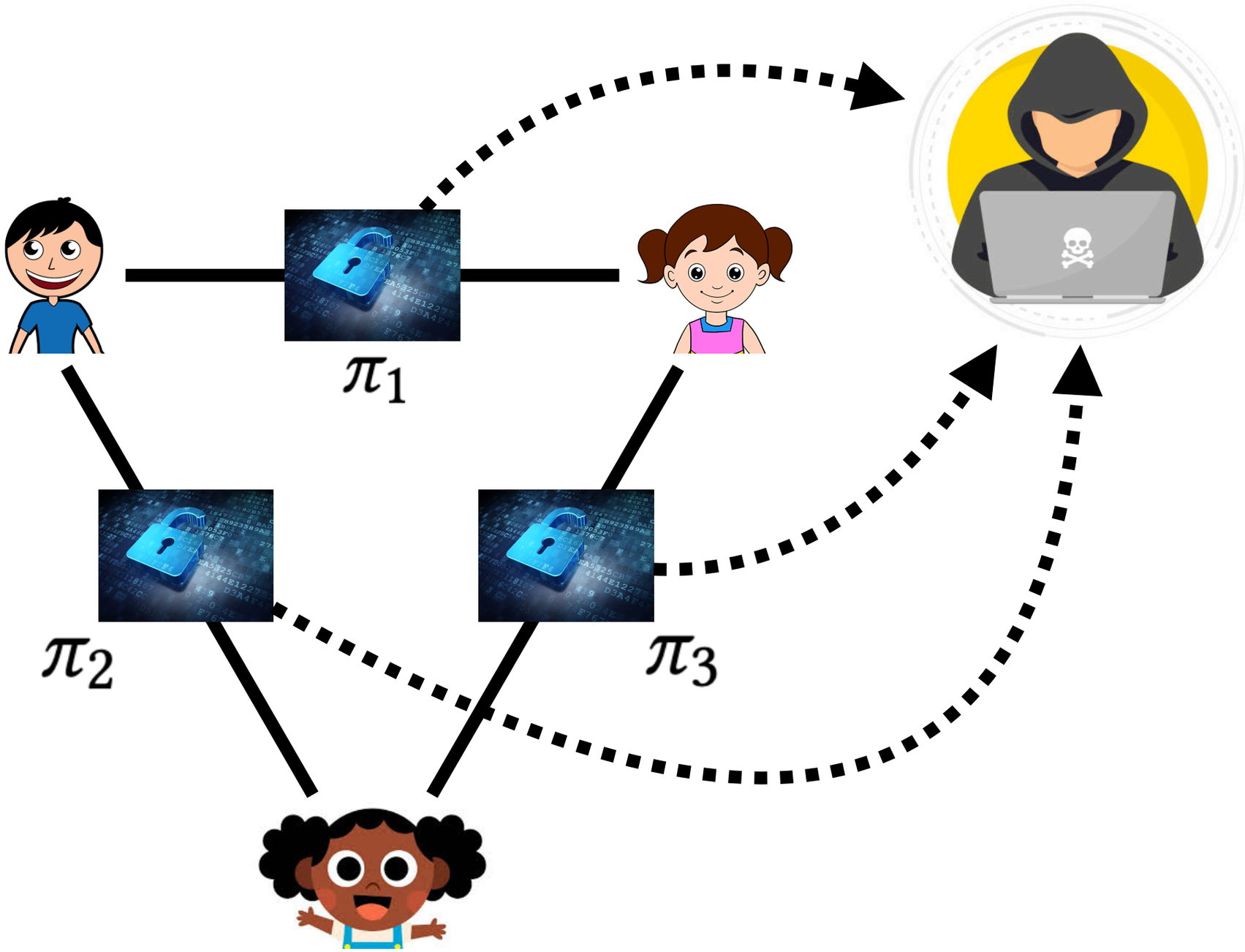
SSL/TLS



Cryptographic Protocols

are also distributed protocols!

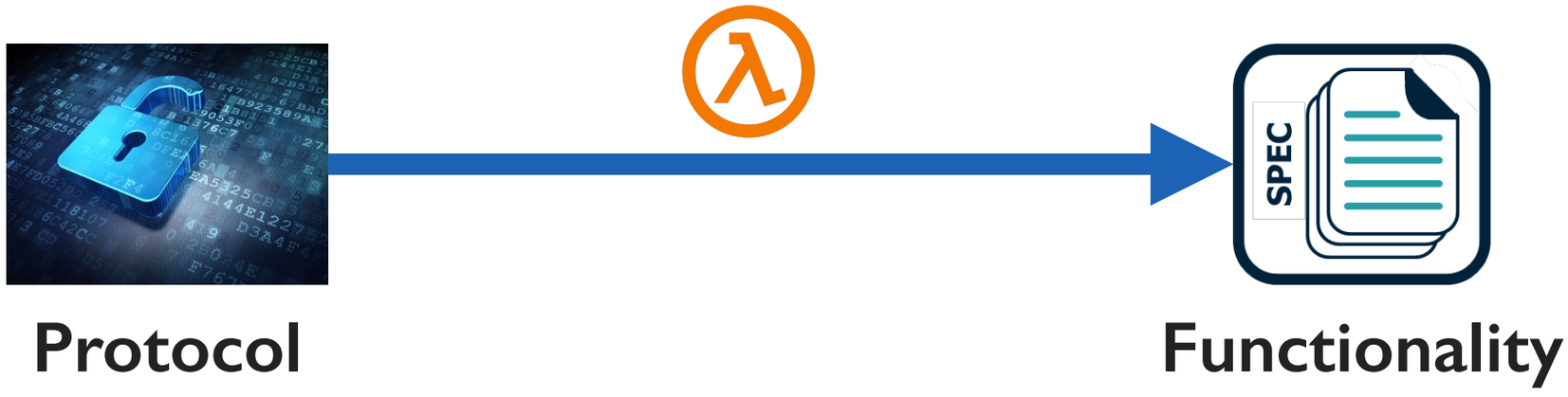
SSL/TLS



Universal Composability (UC)

Protocol

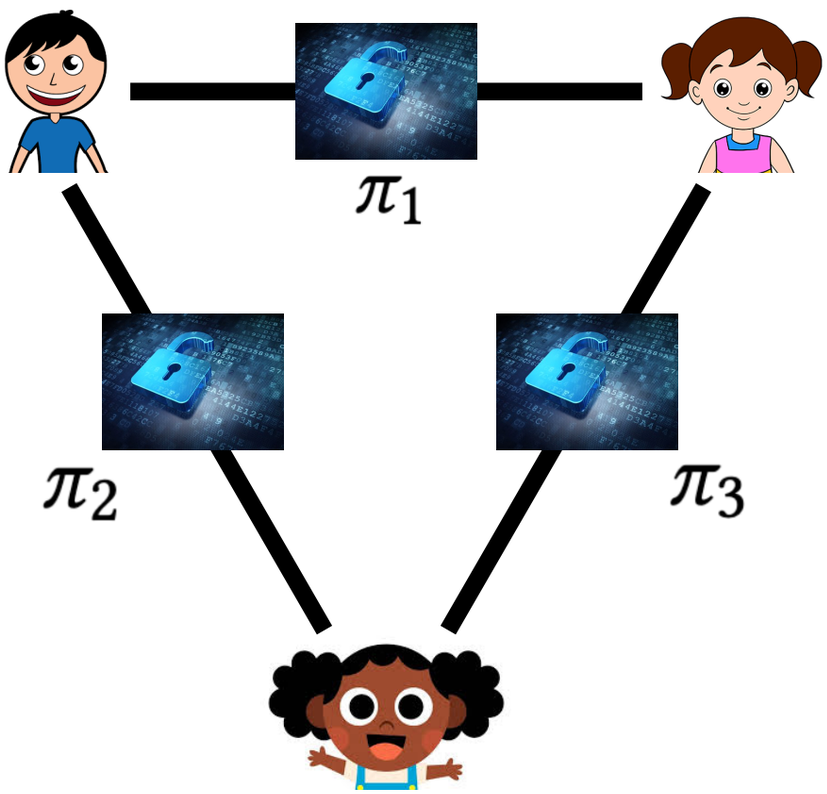
Universal Composability (UC)



Universal Composability (UC)

Protocol

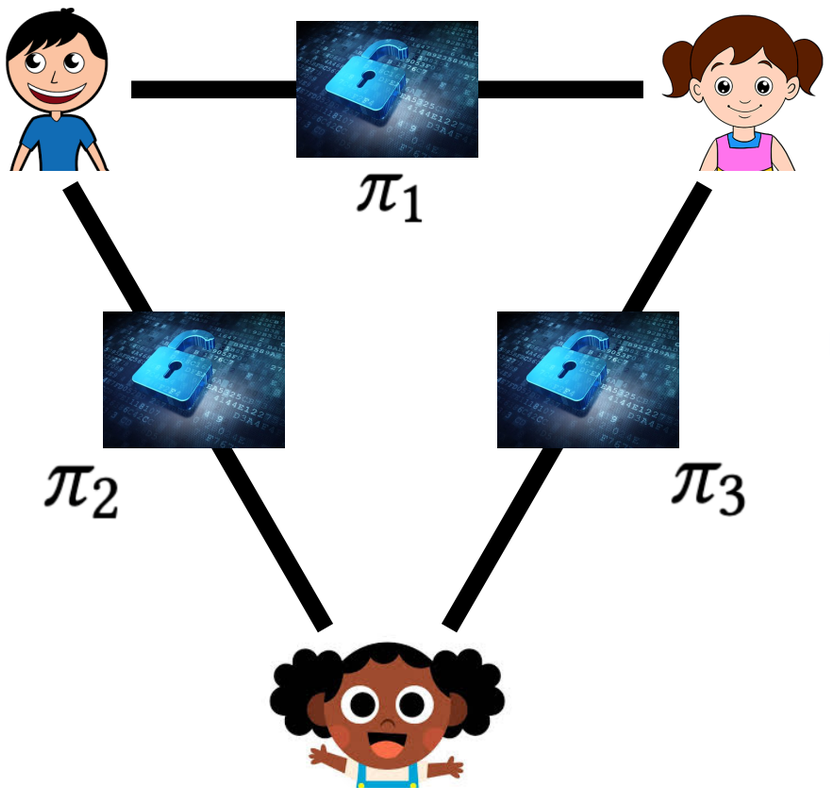
Functionality

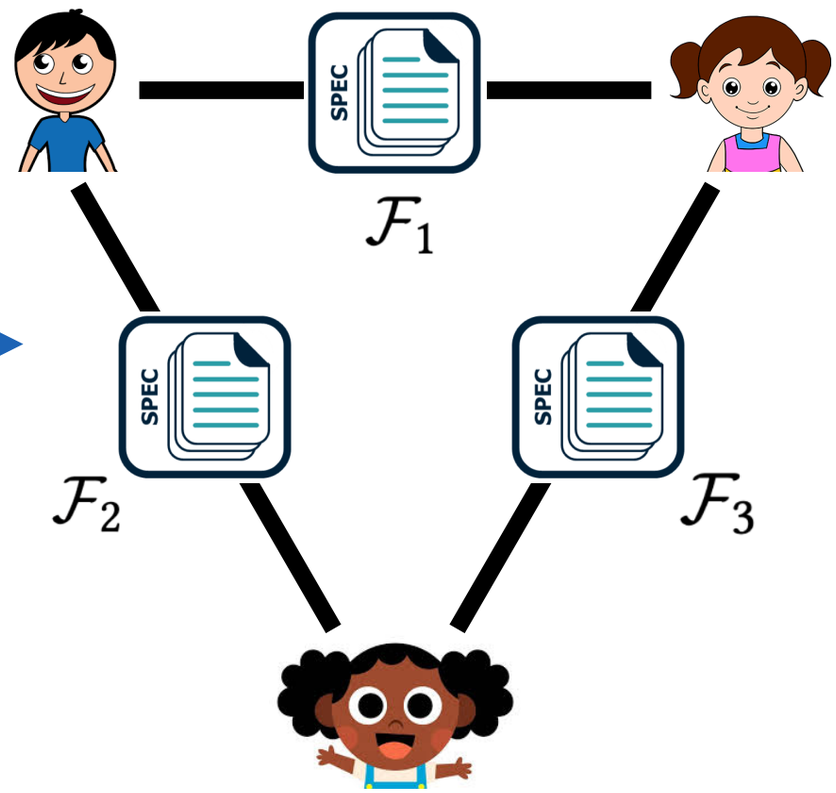


Universal Composability (UC)

Protocol

Functionality

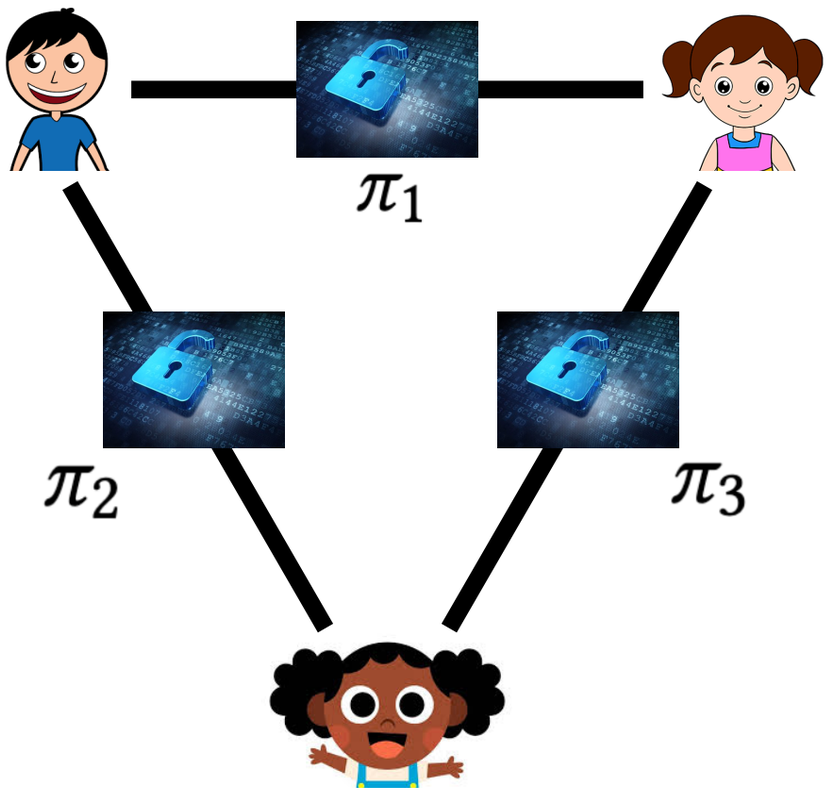


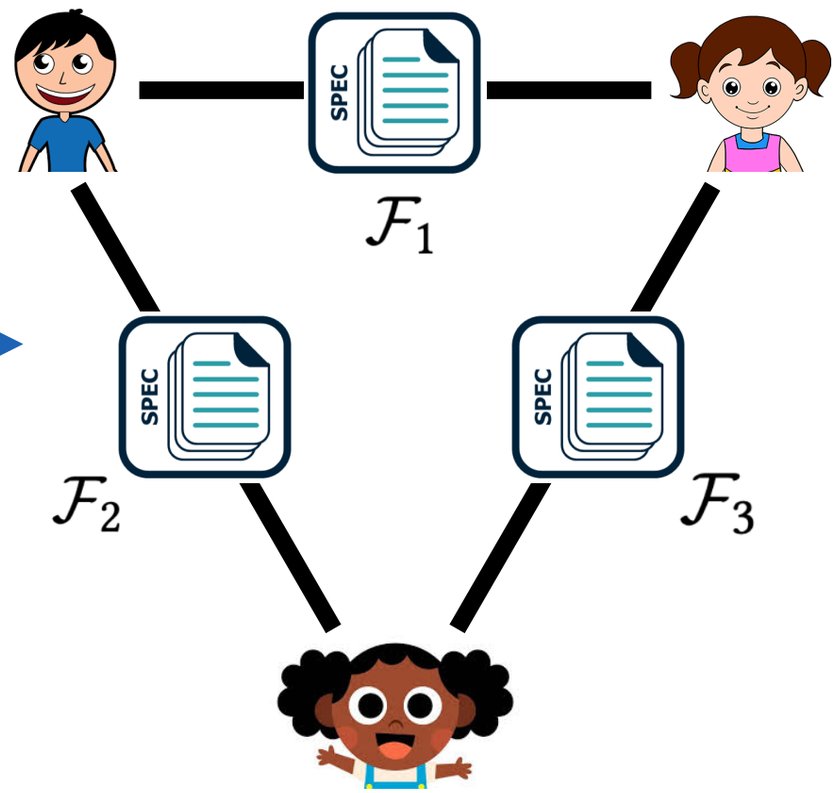


Universal Composability (UC)

Protocol

Functionality





How to ensure resource-bounded computation?

Industrial Research Impacts

Industrial Research Impacts

Program Verifier for
Device Drivers
[CAV '15, ATVA '17]

ships as part of
Static Driver Verifier

Industrial Research Impacts

Program Verifier for Device Drivers
[CAV '15, ATVA '17]

ships as part of Static Driver Verifier

Dust Model Checker for Distributed Protocols

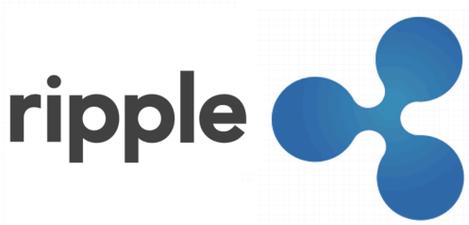
Industrial Research Impacts

Program Verifier for Device Drivers
[CAV '15, ATVA '17]

ships as part of Static Driver Verifier

Dust Model Checker for Distributed Protocols

Smart Contract Analysis Techniques
[SAS' 20, CSF '21]



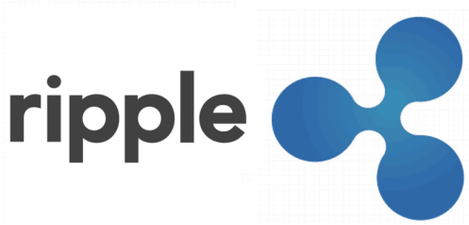
Industrial Research Impacts

Program Verifier for Device Drivers
[CAV '15, ATVA '17]

ships as part of Static Driver Verifier

Dust Model Checker for Distributed Protocols

Smart Contract Analysis Techniques
[SAS' 20, CSF '21]



The Dust Project

Dust

Model Checker for Concurrent Rust Programs

The Dust Project

Concurrency Reasoning

Domain-Specific Support

Dust

Model Checker for
Concurrent Rust Programs

The Dust Project

Distributed Databases

Cloud Storage

Network Protocols

Concurrency Reasoning

Domain-Specific Support

Dust

Model Checker for Concurrent Rust Programs

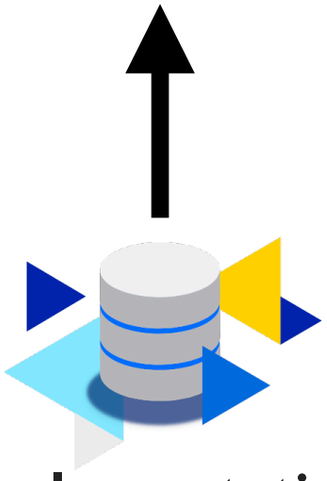
The Dust Model Checker

The Dust Model Checker

Implementation

The Dust Model Checker

Model



Implementation

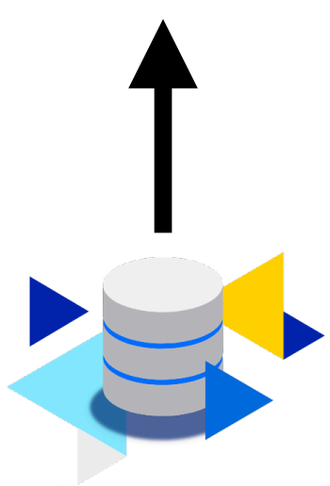
The Dust Model Checker

<u>Key Idea:</u> Model distributed protocols as concurrency execution graphs

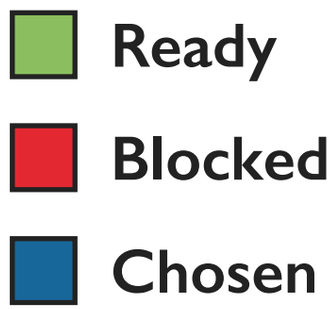
The Dust Model Checker

<u>Key Idea:</u> Model distributed protocols as concurrency execution graphs

Model



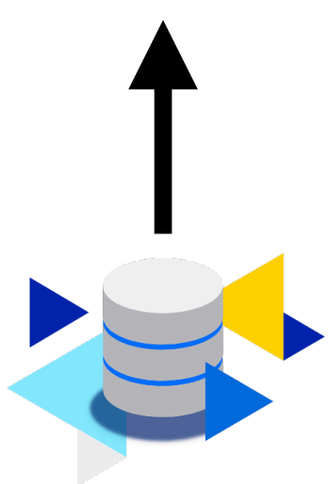
Implementation



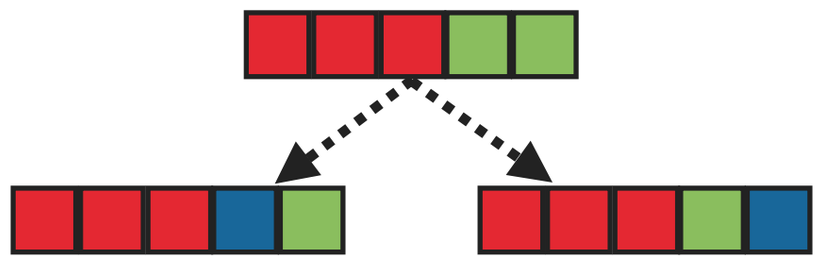
The Dust Model Checker

<u>Key Idea:</u> Model distributed protocols as concurrency execution graphs

Model



Implementation



Ready

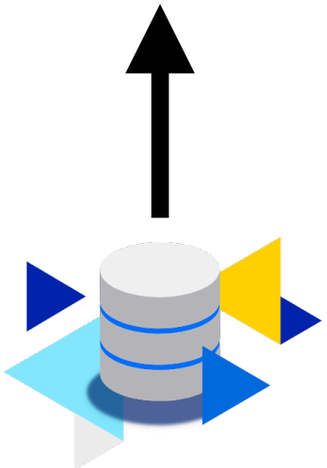
Blocked

Chosen

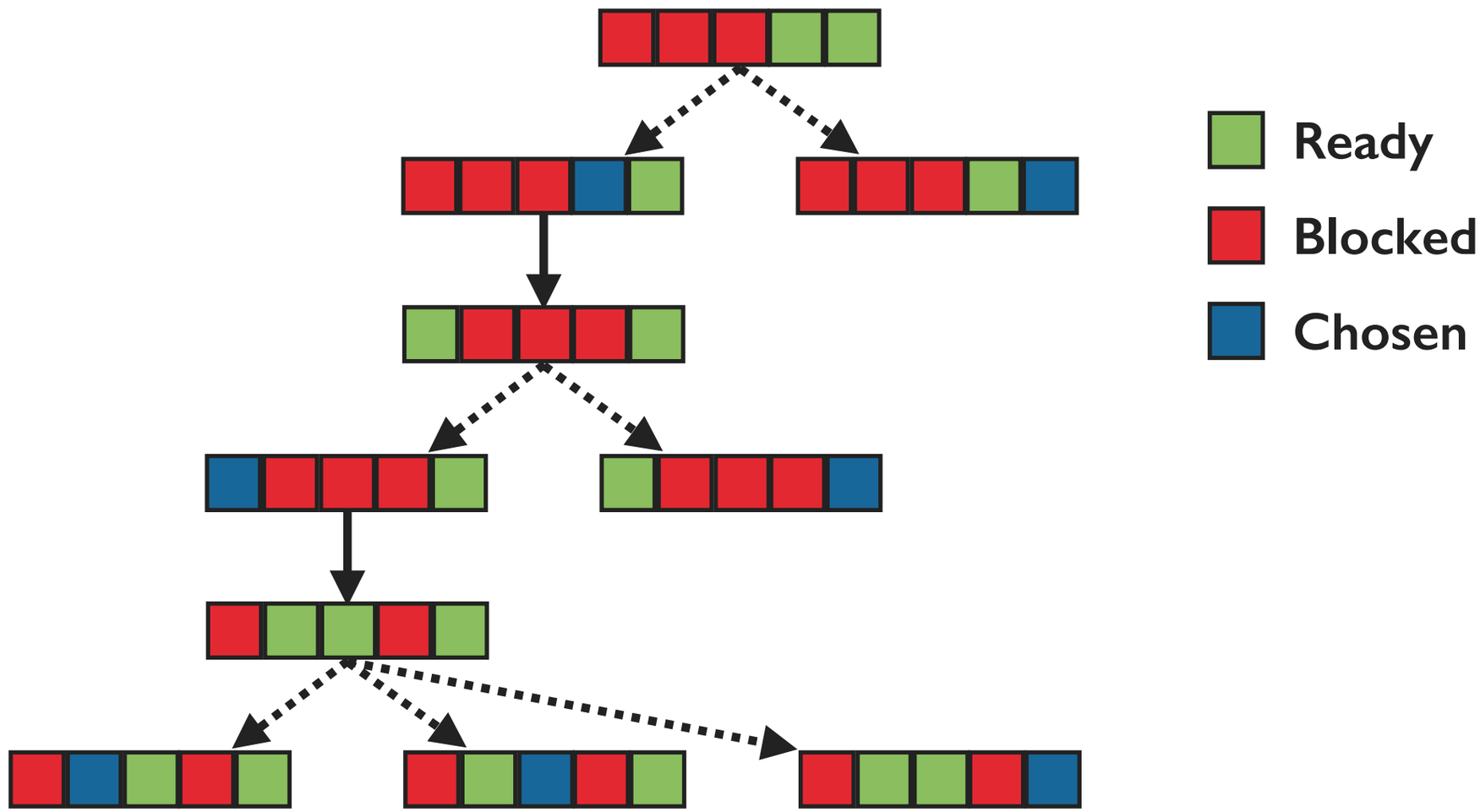
The Dust Model Checker

<u>Key Idea:</u> Model distributed protocols as concurrency execution graphs

Model



Implementation



Ready

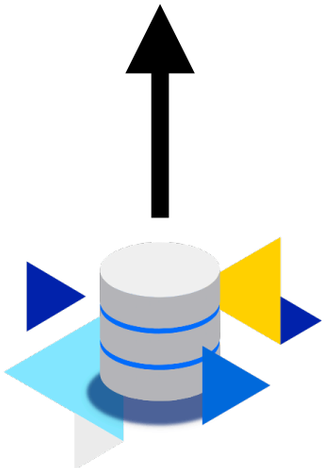
Blocked

Chosen

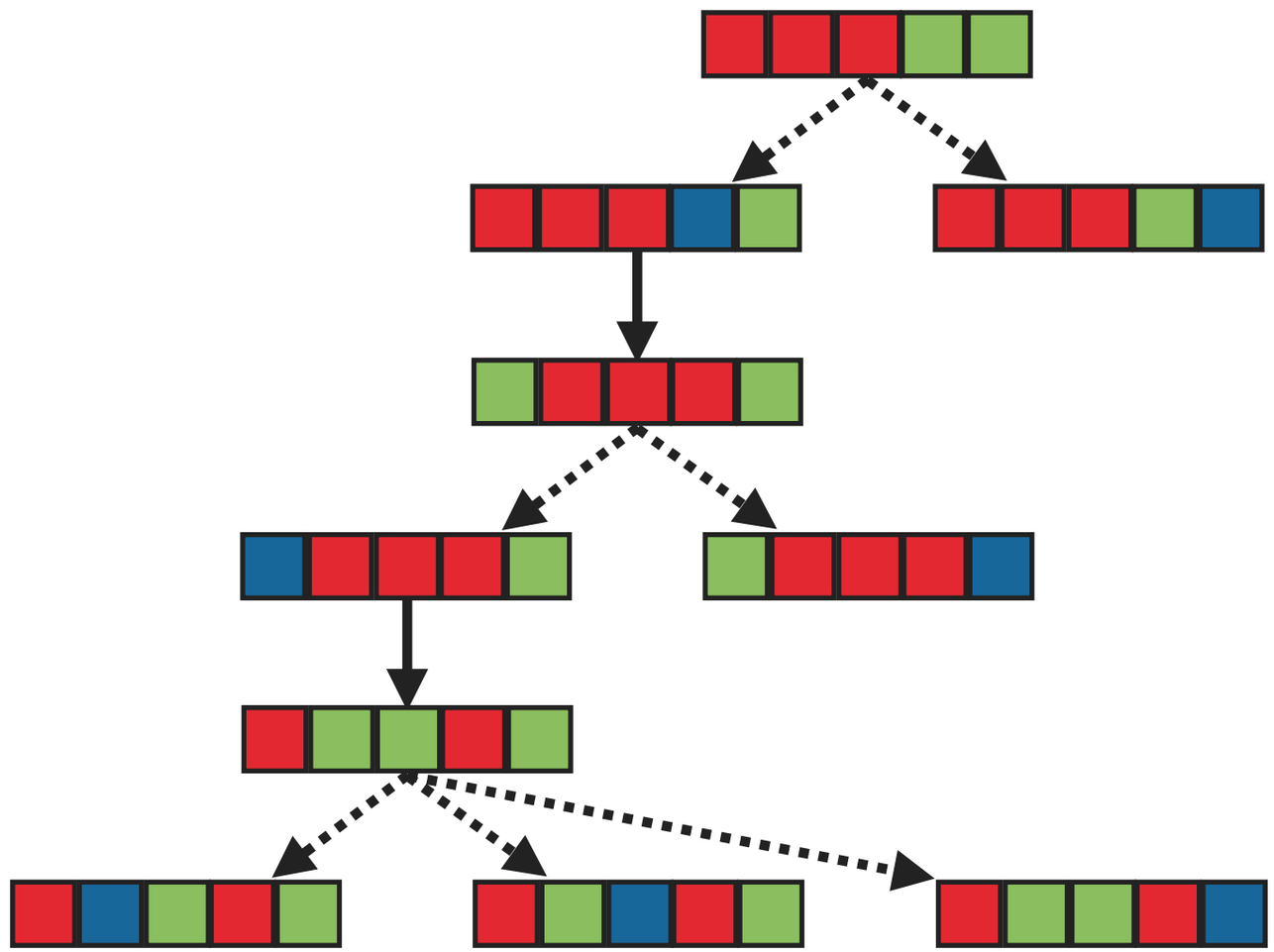
The Dust Model Checker

<u>Key Idea:</u> Model distributed protocols as concurrency execution graphs

Model



Implementation



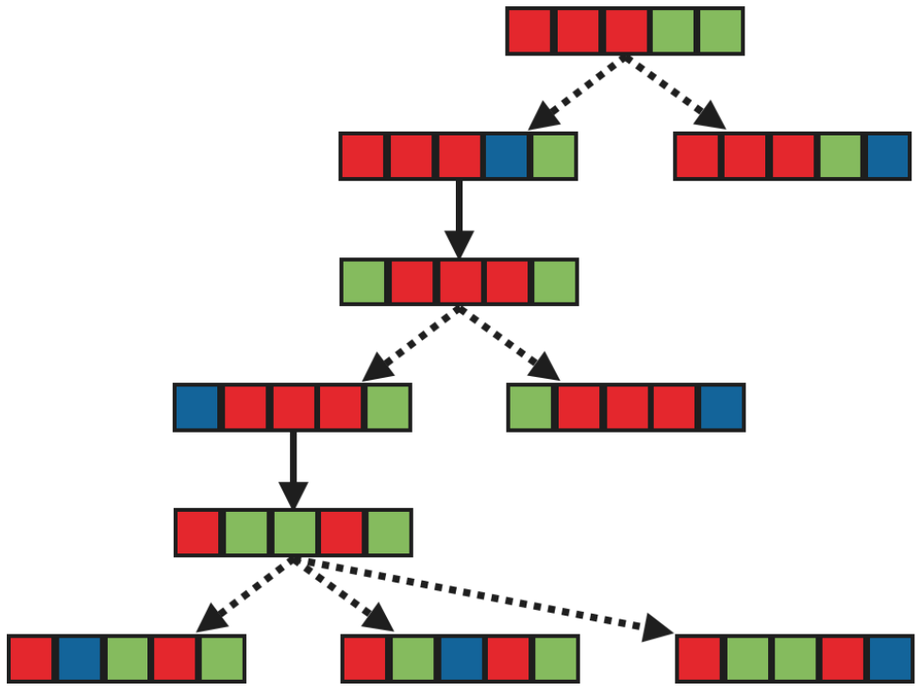
Ready

Blocked

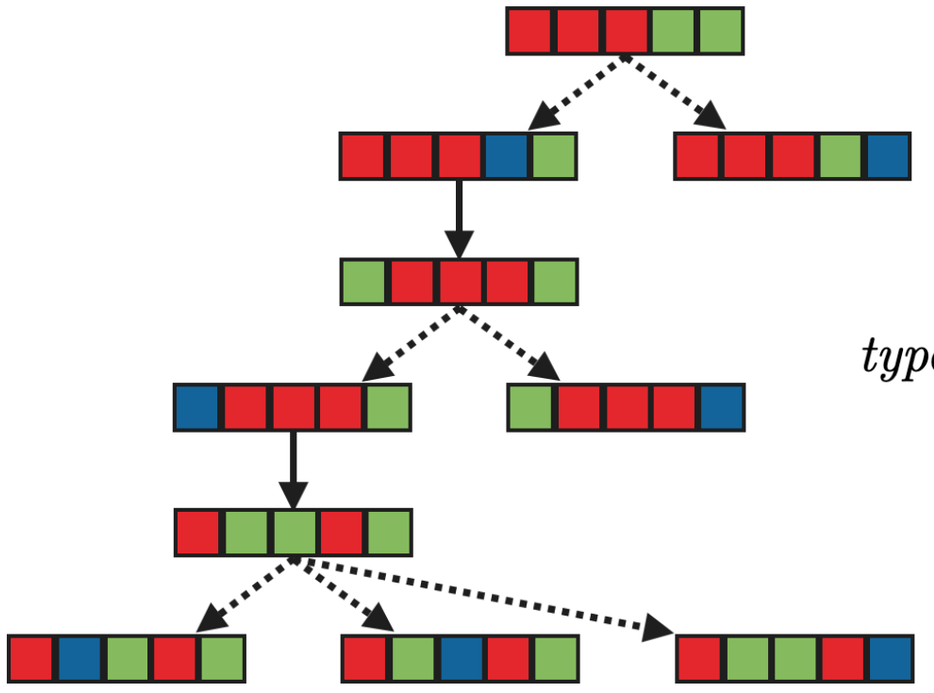
Chosen

State Space Explosion

Session Types Can Help!

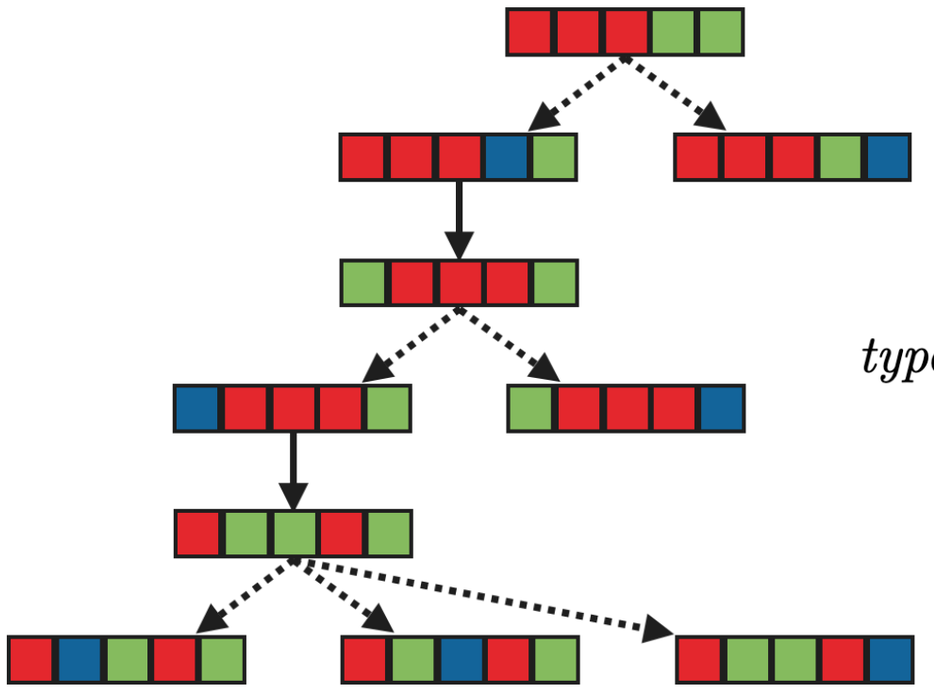


Session Types Can Help!



$$\textit{type}\ \text{auction} = \oplus\{\mathbf{running} : \&\{\mathbf{bid} : \text{money} \multimap \text{auction}\},$$
$$\mathbf{ended} : \&\{\mathbf{collect} : \oplus\{\mathbf{won} : \text{monalisa} \otimes \text{auction},$$
$$\mathbf{lost} : \text{money} \otimes \text{auction}\}\}\}$$

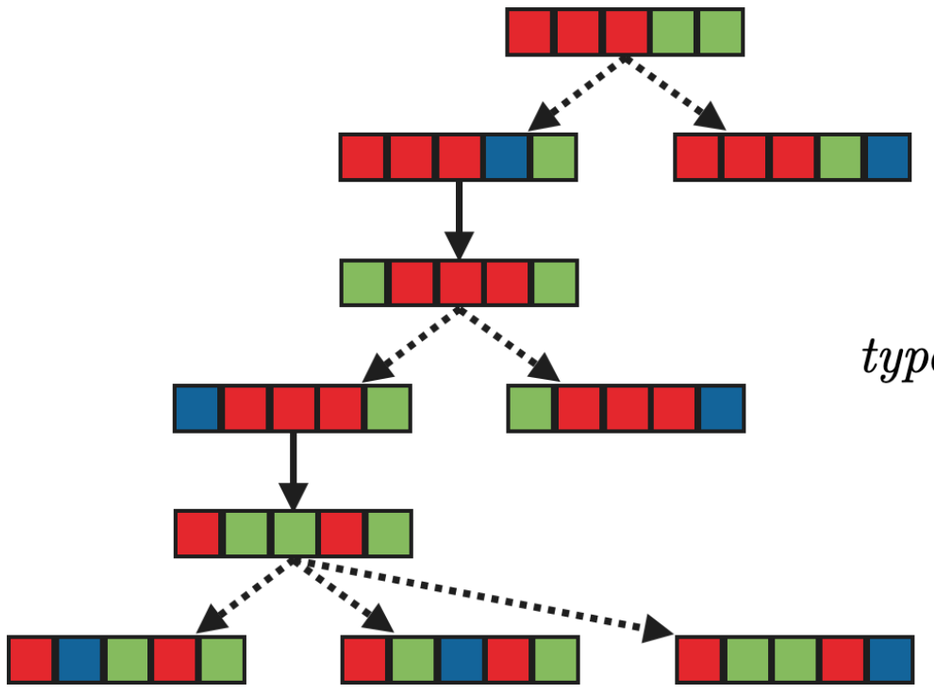
Session Types Can Help!



$$\textit{type}\ \text{auction} = \oplus\{\mathbf{running} : \&\{\mathbf{bid} : \text{money} \multimap \text{auction}\},$$
$$\mathbf{ended} : \&\{\mathbf{collect} : \oplus\{\mathbf{won} : \text{monalisa} \otimes \text{auction},$$
$$\mathbf{lost} : \text{money} \otimes \text{auction}\}\}\}$$

- **Session Types capture communication protocol *with failures!***
- **Many execution traces can be proved *infeasible* via session types**
- ***Eliminate* infeasible traces from execution graph**

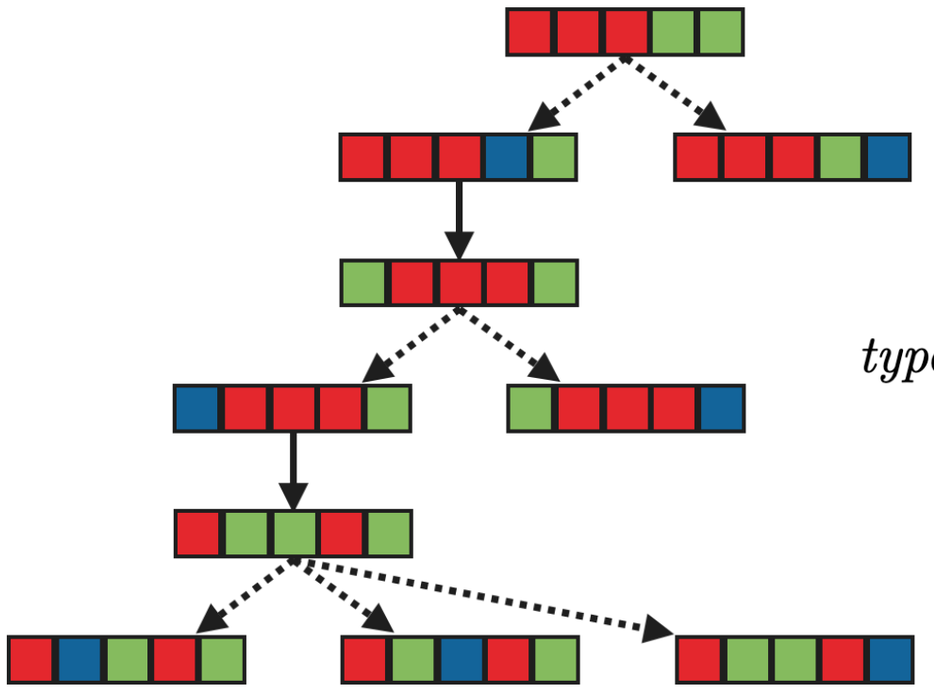
Session Types Can Help!



$$\textit{type}\ \text{auction} = \oplus\{\mathbf{running} : \&\{\mathbf{bid} : \text{money} \multimap \text{auction}\},$$
$$\mathbf{ended} : \&\{\mathbf{collect} : \oplus\{\mathbf{won} : \text{monalisa} \otimes \text{auction},$$
$$\mathbf{lost} : \text{money} \otimes \text{auction}\}\}\}$$

- **Session Types capture communication protocol *with failures!***
- **Many execution traces can be proved *infeasible* via session types**
- ***Eliminate* infeasible traces from execution graph**

Session Types Can Help!

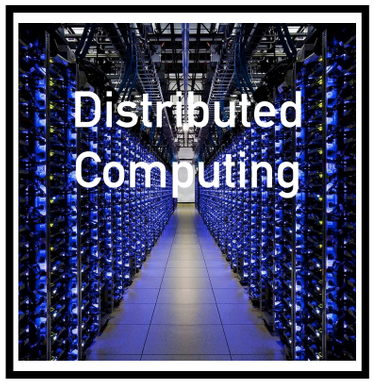


$$\textit{type}\ \text{auction} = \oplus\{\mathbf{running} : \&\{\mathbf{bid} : \text{money} \multimap \text{auction}\},$$
$$\mathbf{ended} : \&\{\mathbf{collect} : \oplus\{\mathbf{won} : \text{monalisa} \otimes \text{auction},$$
$$\mathbf{lost} : \text{money} \otimes \text{auction}\}\}\}$$

- **Session Types capture communication protocol *with failures!***
- **Many execution traces can be proved *infeasible* via session types**
- ***Eliminate* infeasible traces from execution graph**

Das et. al. | Parallel Complexity Analysis with Temporal Session Types | ICFP 2018
Bocchi et. al. | Asynchronous Timed Session Types | ESOP 2019
Brun and Dardha | MAGπ: Types for Failure-Prone Communication
Barwell et. Al. | Generalized Multiparty Session Types with Crash-Stop Failures | CONCUR 2022

Conclusion



Nomos

Resource-Aware Session Types

[LICS '18, SAS '20, CSF '21]

NomosPro

Probabilistic Session Types

[POPL '23]

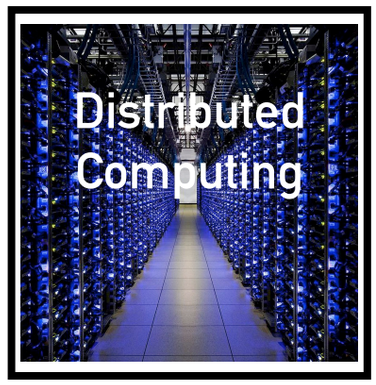
Rast

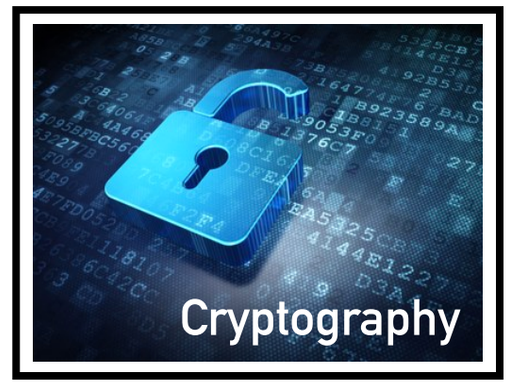
Refinement Session Types

[ICFP '18, FSCD '20, CONCUR '20]

Session Types

Conclusion





Nomos

Resource-Aware Session Types

[LICS '18, SAS '20, CSF '21]

NomosPro

Probabilistic Session Types

[POPL '23]

Rast

Refinement Session Types

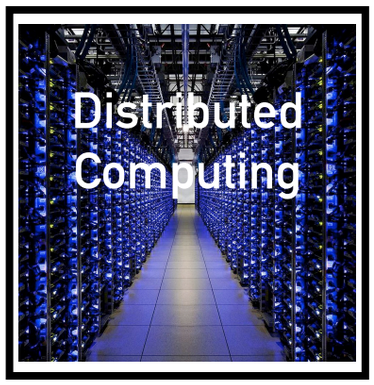
[ICFP '18, FSCD '20, CONCUR '20]

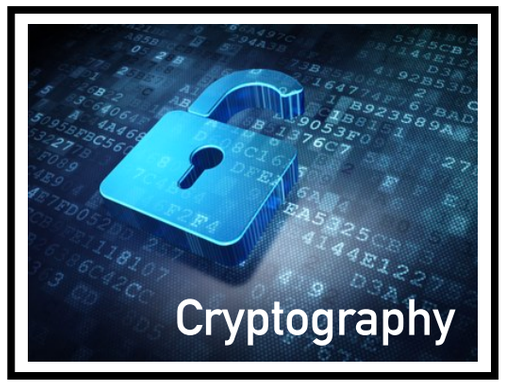
NomosUC

Import Session Types

Session Types

Conclusion





Nomos

Resource-Aware Session Types

[LICS '18, SAS '20, CSF '21]

NomosPro

Probabilistic Session Types

[POPL '23]

Rast

Refinement Session Types

[ICFP '18, FSCD '20, CONCUR '20]

NomosUC

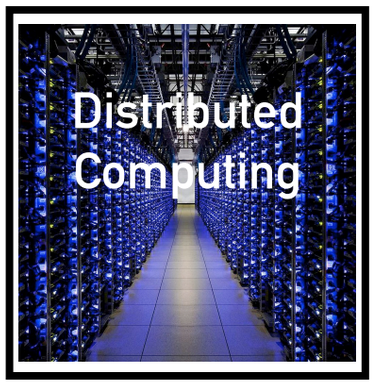
Import Session Types

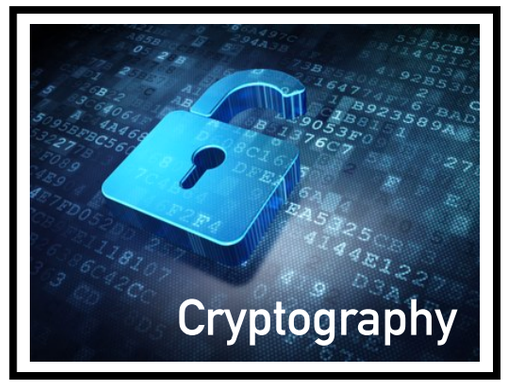
Session Types

Dependent Session Types

Encoding Failures, Timeouts

Conclusion





Nomos
Resource-Aware Session Types
[LICS '18, SAS '20, CSF '21]

NomosPro
Probabilistic Session Types
[POPL '23]

Rast
Refinement Session Types
[ICFP '18, FSCD '20, CONCUR '20]

NomosUC
Import Session Types

Session Types

Dependent Session Types

Encoding Failures, Timeouts