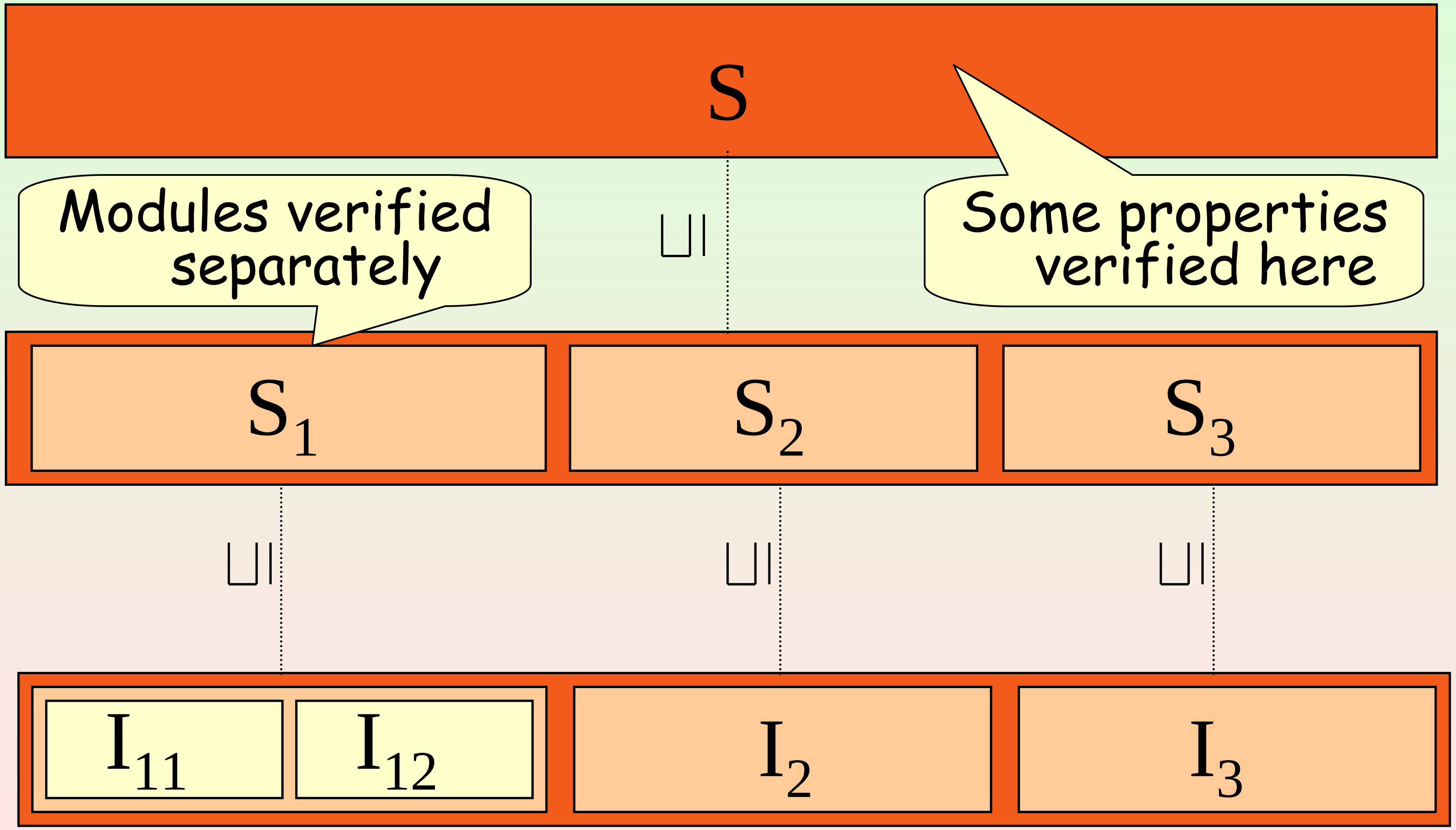
On the Use of Probabilistic Automata for Security Proofs

Roberto Segala
University of Verona

Motivation

- Proofs of cryptographic protocols are hard
  - Especially in the computational model
  - Limited mathematical tools available
    - … or limited willingness to work out the details
- Symbolic methods help
  - But proving soundness requires classical proofs
- Many proofs rely on correspondence between computations of different systems
  - Concurrency theory has a lot to say
- Can we take advantage of concurrency theory
  - … directly in the computational model?

Hierarchical Compositional Verification



Implementation

- Typically some form of behavioral inclusion
  - Traces
    - Ordinary, complete, quiescent, fair
  - Failures
    - Traces followed by actions the system refuses to perform
  - Tests
    - Occurrence of some success event in appropriate contexts

- Nice properties
  - Transitive
  - Compositional
  - Affine with logical implication
    - … when properties are sets of behaviors

- Hard to check
  - Usually Pspace-complete
  - But simulation relations help

Proving Implementation

- Behavioral inclusion
  - Behaviors are full computations
    - Possibly infinite length
  - Properties of complex objects
    - Global reasoning
  - Easy to end up with "proofs by intuition"

- Simulation relations
  - Sound for behavioral inclusion
  - Properties of single computational steps
    - Local reasoning
  - Easier to be rigorous

Nondeterminism and Probability

- Nondeterminism
  - Relative speeds of processes
  - Unknown behavior of users
    - Adversary in DY model
  - Underspecification
  - Abstraction
    - Forget about probabilities

- Probability
  - User behavior may obey probability laws
  - Processes may flip coins
    - Randomized algorithms, protocols
    - Nonces, keys, …

Overview

- Probabilistic Automata
  - Definition, executions, traces
  - Composition, projection
  - Behavioral inclusion
  - Simulation relations
- Task Probabilistic I/O Automata
  - A way to restrict nondeterminism
  - Case study with oblivious transfer
  - Nondeterminism may leak information
  - Reasoning up to negligible errors
- Approximated simulation relations
  - Relate automata that fail with negligible probability with automata that do not fail
  - Case study with agent authentication
- Using Probabilistic Automata for DY-soundness
  - A possibility?

Probabilistic Automata

The Main Idea

- Add probability to Concurrency Theory
  - Nondeterminism should remain
  - Should obtain a conservative extension

- Proposals to tackle the problem
  - Replace points with measures
  - Replace functions with measurable functions

Automata

$$A = (Q\ ,\ q_0\ ,\ E\ ,\ H\ ,\ D)$$

Transition relation
$D \subseteq Q$ ´ $(E \cup H)$ ´ Q

Internal (hidden) actions

External actions: $E \cap H = \emptyset$

Initial state: $q_0 \in Q$

States

Probabilistic Automata

$$PA = (Q, q_0, E, H, D)$$

Transition relation
$D \subseteq Q$ ´ $(E \cup H)$ ´ Disc(Q)

Internal (hidden) actions

External actions: $E \cap H = \emptyset$

Initial state: $q_0 \in Q$

States

Example: Automata

$$A = (Q\ ,\ q_0\ ,\ E\ ,\ H\ ,\ D)$$

$$\begin{tikzcd}
q_0 \arrow[rr, "d"] \arrow[dr, "n"'] & & q_2 \arrow[rr, "choc"] \arrow[dd, "ch"] & & q_4 \\
& q_1 \arrow[ur, "n"] \arrow[dr, "n"'] & & & \\
& & q_3 \arrow[rr, "coffee"'] & & q_5
\end{tikzcd}$$

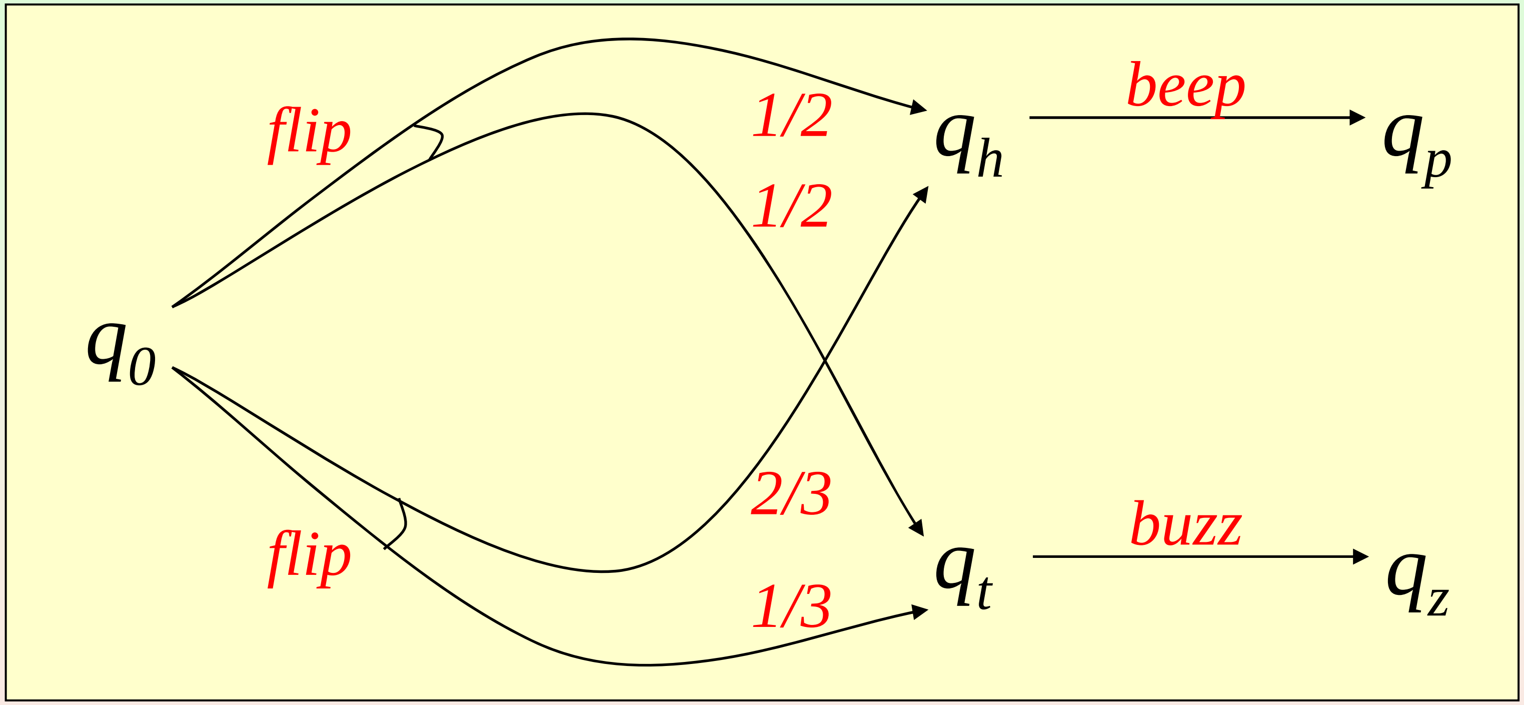
Execution: $q_0\ n\ q_1\ n\ q_2\ ch\ q_3\ coffee\ q_5$

Trace: $n\ n\ coffee$

Example: Probabilistic Automata

$$\begin{tikzcd}
 & q_1 \arrow[r, "flip\ 1/2"] \arrow[ddr, "1/2"] & q_3 \arrow[r, "beep"] & q_5 \\
q_0 \arrow[ur, "fair"] \arrow[dr, "unfair"'] & & & \\
 & q_2 \arrow[r, "flip\ 1/3"'] \arrow[uur, "2/3"] & q_4 &
\end{tikzcd}$$

Example: Probabilistic Automata



Example: Probabilistic Automata

$$\begin{tikzcd}
 & q_1 \arrow[r, "flip\quad 1/2"] \arrow[dr, "1/2"'] & q_3 \arrow[r, "beep"] & q_5 \\
q_0 \arrow[ur, "fair"] \arrow[dr, "unfair"'] & & & \\
 & q_2 \arrow[r, "flip\quad 1/3"'] \arrow[uur, "2/3"] & q_4 &
\end{tikzcd}$$

What is the probability of beeping?

Example: Probabilistic Executions

$$\begin{tikzcd}
q_0 \arrow[r, "fair"] & q_1 \arrow[r, "flip\quad 1/2"] \arrow[dr, "1/2"'] & q_3 \arrow[r, "beep"] & q_5 \\
 & & q_4 &
\end{tikzcd}$$

$\mu(beep) = 1/2$

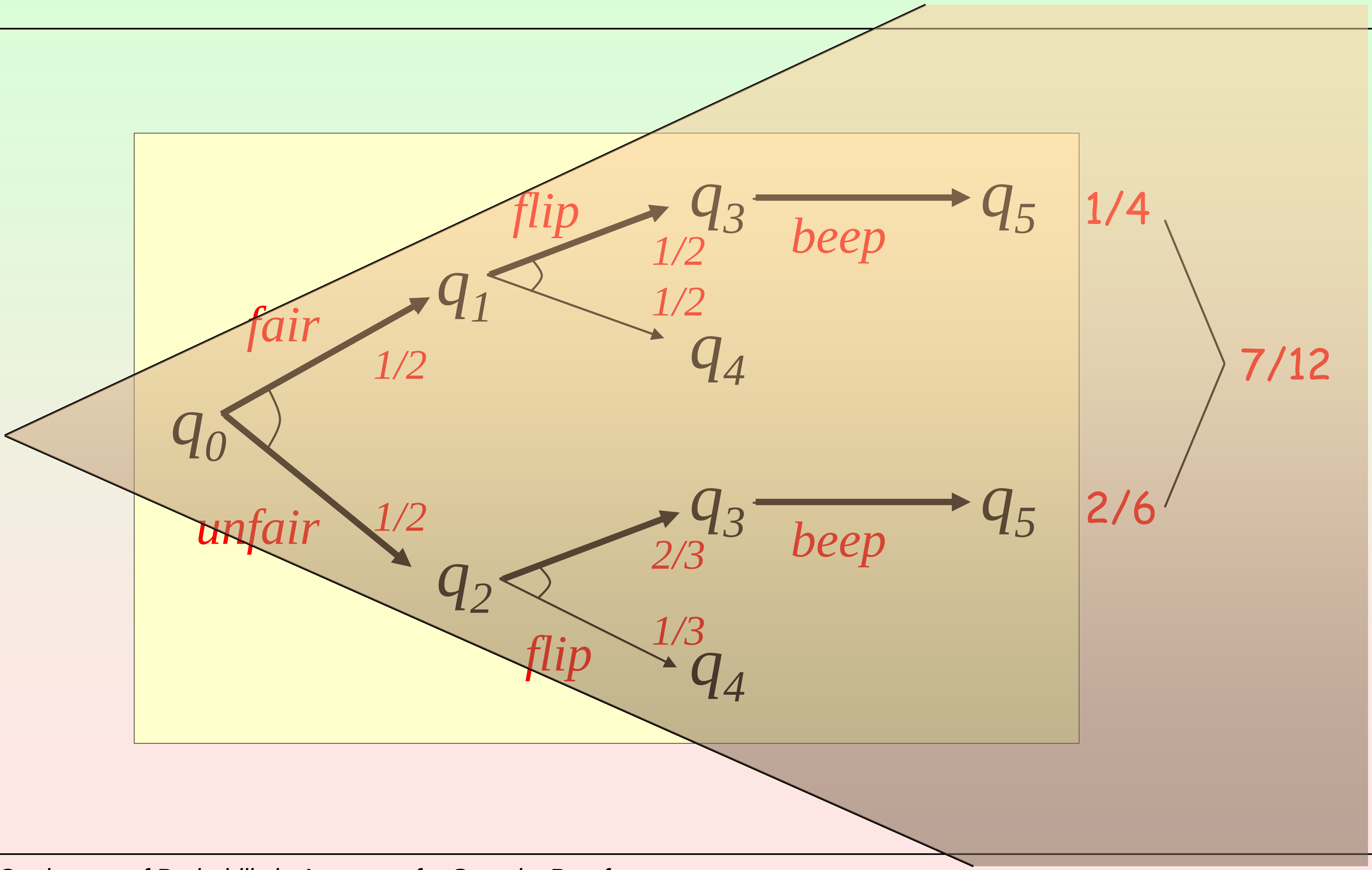
1/2

$$\begin{tikzcd}
 & & q_3 \arrow[r, "beep"] & q_5 \\
q_0 \arrow[r, "unfair"'] & q_2 \arrow[ur, "2/3"'] \arrow[r, "flip\quad 1/3"'] & q_4 &
\end{tikzcd}$$

$\mu(beep) = 2/3$

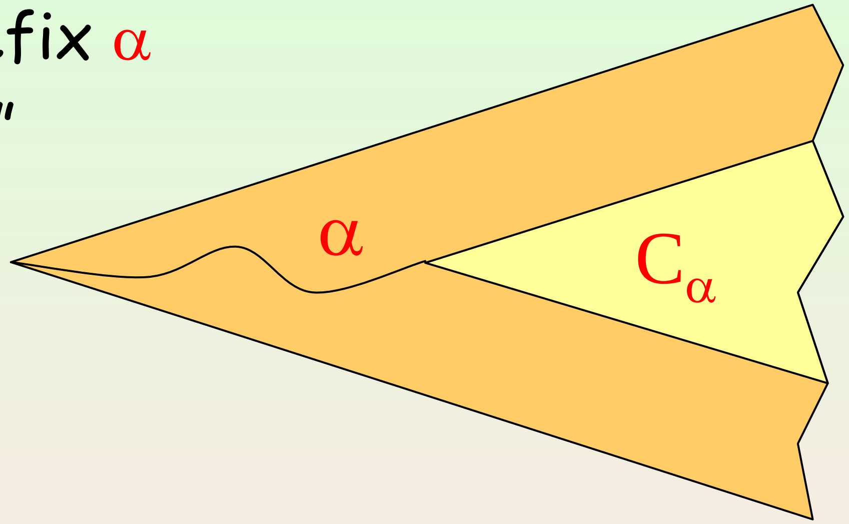
2/3

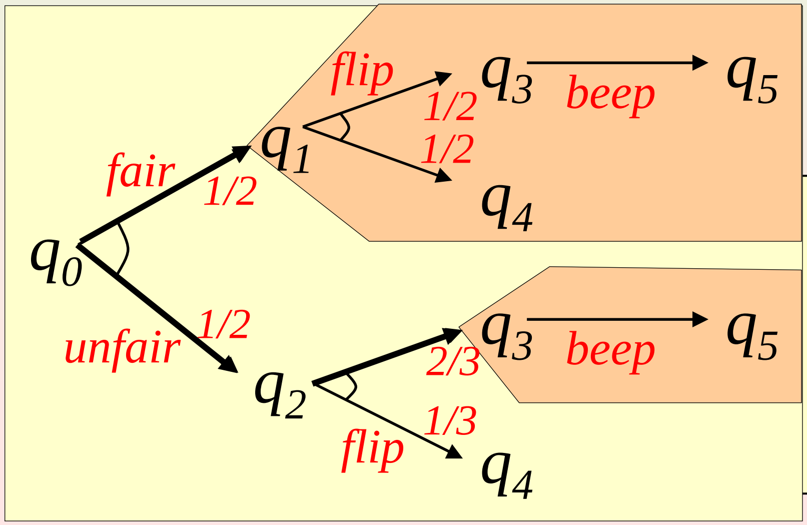
Example: Probabilistic Executions



Cones and Measures

- <u>Cone of α</u>
  - Set of executions with prefix α
  - Represent event "α occurs"
- <u>Measure of a cone</u>
  - Product edges of α





<u>extends uniquely</u>

σ-field generated by cones

Schedulers - Probabilistic Executions

Scheduler

Function $\sigma : exec^*(A) \rightarrow SubDisc(D)$

if $\sigma(\alpha)((q,a,v)) > 0$ then $q = lstate(\alpha)$

Probabilistic execution generated by σ from state r

Measure $\mu_{\sigma,r}$

$\mu_{\sigma,r}(C_s) = 0$ if $r \neq s$

$\mu_{\sigma,r}(C_r) = 1$

$$\mu_{\sigma,r}(C_{\alpha a q}) = \mu_{\sigma,r}(C_\alpha) \cdot \left(\sum_{(s,a,v) \in D} \sigma(\alpha)((s,a,v)) v(q) \right)$$

Summing Up

$$\begin{tikzcd}
\text{Automata} \arrow[d] & & \text{Probabilistic Automata} \arrow[d] \\
\text{Executions} \arrow[d] & \text{schedulers} & \text{Probabilistic Executions (measures over executions)} \arrow[d] \\
\text{Traces} \arrow[d] & \text{trace function} & \text{???} \arrow[d] \\
\text{Trace inclusion} & \text{implementation relation} & \text{???}
\end{tikzcd}$$

Related Models

- Rabin Probabilistic Automata [Rab63]
  - Deterministic Probabilistic Automata
  - Introduced in context of language theory
  - Actions have a different use
- Reactive Systems [LS89, GSST90]
  - Deterministic Probabilistic Automata
- Markov Decision Processes [Bel57]
  - Deterministic Probabilistic Automata
    - Though actions have a completely different use
  - …plus reward functions
- Labeled Concurrent Markov Chains [HJ89]
  - Probabilistic Automata where
    - States are partitioned into deterministic and probabilistic
    - Nondeterministic states enable several ordinary transitions
    - Probabilistic states enable one transition

Parallel Composition

Composition of Probabilistic Automata

$A_1 = (Q_1,q_1,E_1,H_1,D_1)$ $\quad\|\quad$ $A_2 = (Q_2,q_2,E_2,H_2,D_2)$

$$A_1 \,\|\, A_2 \;=\; (Q_1 ´ Q_2\,,\,(q_1,q_2)\,,\,E_1 \cup E_2\,,\,H_1 \cup H_2\,,\,D)$$

$$D = \left\{ (q,a,(s_1,s_2)) \;\middle|\; \begin{array}{l} \text{if } a \in E_i \cup H_i \text{ then } (\pi_i(q)\,,\,a\,,\,s_i\,) \in D_i \\ \text{if } a \notin E_i \cup H_i \text{ then } s_i = \pi_i(q) \end{array} \quad i \in \{1,2\} \right\}$$

$$D = \left\{ (q,a,\mu_1 \times \mu_2) \;\middle|\; \begin{array}{l} \text{if } a \in E_i \cup H_i \text{ then } (\pi_i(q)\,,\,a\,,\,\mu_i\,) \in D_i \\ \text{if } a \notin E_i \cup H_i \text{ then } \mu_i = \delta(\pi_i(q)) \end{array} \quad i \in \{1,2\} \right\}$$

Example: Composition of Automata

$E = \{n, d, choc, coffee\}$

$$\begin{tikzcd}
q_0 \arrow[rr, "d"] \arrow[dr, "n"'] & & q_2 \arrow[r, "choc"] \arrow[dd, "ch"] & q_4 \\
& q_1 \arrow[ur, "n"] \arrow[dr, "n"'] & & \\
& & q_3 \arrow[r, "coffee"'] & q_5
\end{tikzcd}$$

$\|$

$E = \{n, d, choc, coffee\}$

$$\begin{tikzcd}
& & s_2 \\
s_0 \arrow[r, "d"] & s_1 \arrow[ur, "choc"] \arrow[dr, "coffee"'] & \\
& & s_3
\end{tikzcd}$$

$$\begin{tikzcd}
(q_0,s_0) \arrow[r, "d"] & (q_2,s_1) \arrow[r, "choc"] \arrow[d, "ch"] & (q_4,s_2) \\
& (q_3,s_1) \arrow[r, "coffee"] & (q_5,s_3)
\end{tikzcd}$$

Ex. Composition of Probabilistic Automata

$$\begin{tikzcd}
 & s_1 \arrow[r, "fair"] & s_3 \\
s_0 \arrow[ur, "ch", "1/2"'] \arrow[dr, "1/2"] & & \\
 & s_2 \arrow[r, "unfair"'] & s_4
\end{tikzcd}$$

$$\|$$

$$\begin{tikzcd}
 & q_1 \arrow[r, "flip\ 1/2"] \arrow[drr, "1/2"] & q_3 \arrow[r, "beep"] & q_5 \\
q_0 \arrow[ur, "fair"] \arrow[dr, "unfair"'] & & & \\
 & q_2 \arrow[r, "flip\ 1/3"'] \arrow[uur, "2/3"] & q_4 &
\end{tikzcd}$$

$$\begin{tikzcd}
 & (s_1,q_0) \arrow[r, "fair"] & (s_3,q_1) \arrow[r, "1/2"'] \arrow[dr, "flip", "1/2"'] & (s_3,q_3) \arrow[r, "beep"] & (s_3,q_5) \\
(s_0,q_0) \arrow[ur, "ch", "1/2"'] \arrow[ddr, "1/2"] & & & (s_3,q_4) & \\
 & & & (s_4,q_3) \arrow[r, "beep"] & (s_4,q_5) \\
 & (s_2,q_0) \arrow[r, "unfair"] & (s_4,q_2) \arrow[ur, "flip", "2/3"'] \arrow[r, "1/3"] & (s_4,q_4) &
\end{tikzcd}$$

Projections

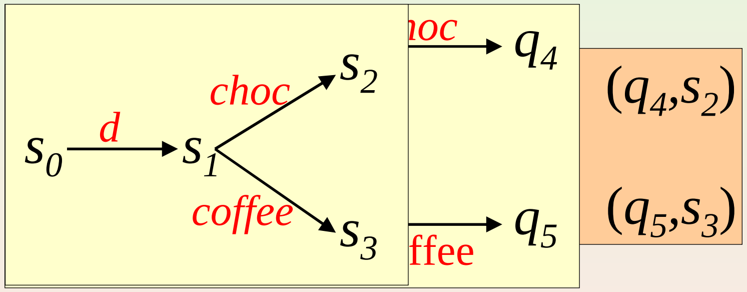
Let α be an execution of $A_1 \parallel A_2$

$\alpha = (q_0,s_0)\ d\ (q_2,s_1)\ ch\ (q_3,s_1)\ coffee\ (q_5,s_3)$

What are the contributions of A_1 and A_2?

$\pi_1(\alpha) \equiv q_0\ d\ q_2\ ch\ q_3\ coffee\ q_5$

$\pi_2(\alpha) \equiv s_0\ d\ s_1\ coffee\ s_3$



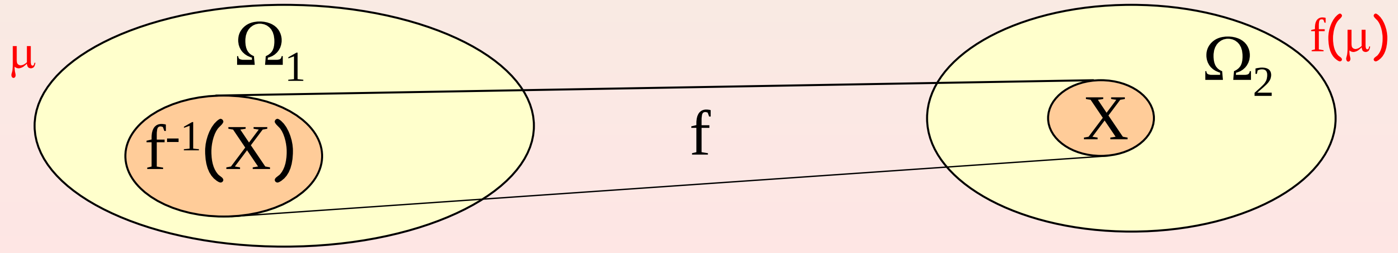
Theorem

$\alpha \in execs(A_1 \| A_2)$ iff $\forall_{i \in \{1,2\}}\ \pi_i(\alpha) \in execs(A_i)$

Measure Theory: Image Measure

- Measurable function from (Ω_1,F_1) to (Ω_2,F_2)
  - Function f from Ω_1 to Ω_2
  - For each element X of F_2, $f^{-1}(X) \in F_1$

- Image measure $f(\mu)$
  - $f(\mu)(X) = \mu(f^{-1}(X))$



Projections

The projection function is measurable

$\pi(\mu)$: image measure under π of μ

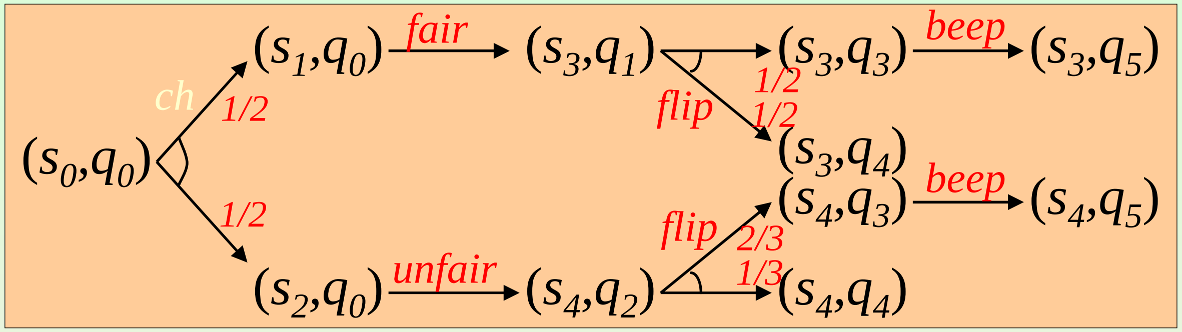
Theorem

If μ is a probabilistic execution of $A_1 \| A_2$

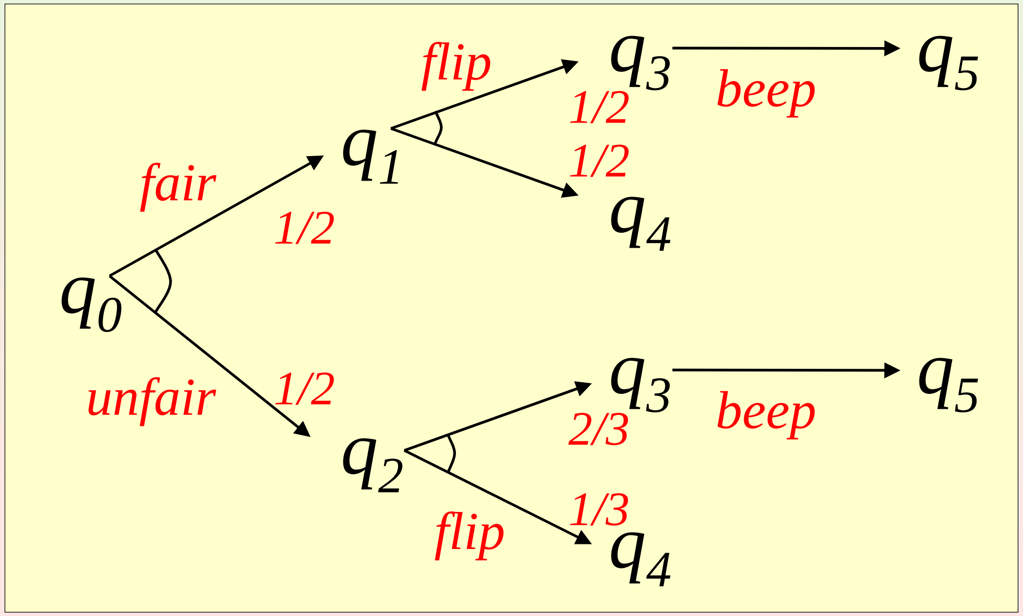
then

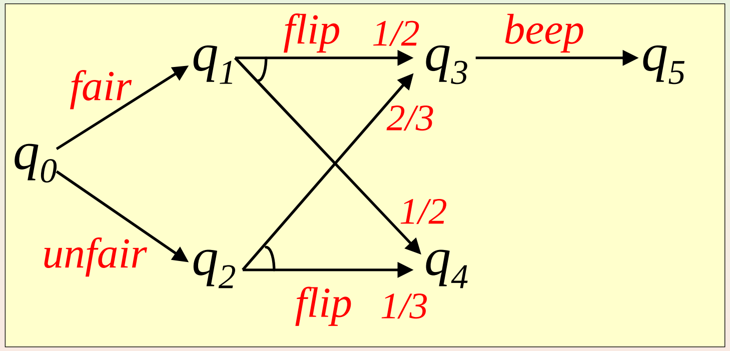
$\pi_i(\mu)$ is a probabilistic execution of A_i

Example: Projection



Projection onto right component





Note that the scheduler is randomized

Use of Projections

- Let $M = MP||CF$
- Suppose that MP satisfies Φ provided that the environment (CF) satisfies Ψ
- Suppose that CF satisfies Ψ with probability p
- Can I conclude that M satisfies Φ with probability p?

$$\frac{MP \models \Psi \Rightarrow \Phi \qquad CF \models [\Psi]_{\geq p}}{M \models [\Phi]_{\geq p}}$$

- This example is taken from a real case study [PLS01]
  - Randomized consensus protocol of Aspnes and Herlihy [AH90]
  - MP is a complex non randomized protocol
  - CF is a relatively simple randomized coin flipper

Formal Argument

Let μ be a probabilistic execution of M.

$$\begin{tikzcd}
 & & \mu \\
 & \boxed{\mu(\pi_2^{-1}(\Psi)) \geq p} \arrow[ddl, "\text{projection}"'] & \\
\pi_1(\mu) & \mathrm{M} & \pi_2(\mu) \\
\boxed{\pi_1(\pi_2^{-1}(\Psi)) \text{ sat. } \Phi} & & \boxed{\pi_2(\mu)(\Psi) \geq p} \arrow[uul, "\text{inverse image}"'] \\
\mathrm{MP} & & \mathrm{CF}
\end{tikzcd}$$

Language Inclusion

Summing Up

$$\begin{tikzcd}
\text{Automata} \arrow[dd] & & \text{Probabilistic Automata} \arrow[dd] \\
& \text{schedulers} & \\
\text{Executions} \arrow[dd] & & \text{Probabilistic Executions (measures over executions)} \arrow[dd] \\
& \text{trace function} & \\
\text{Traces} \arrow[dd] & & \text{Trace distributions (measures over traces)} \arrow[dd] \\
& \text{implementation relation} & \\
\text{Trace inclusion} & & \text{Trace distribution inclusion}
\end{tikzcd}$$

Trace Distributions

The *trace* function is measurable

Trace distribution of μ

$tdist(\mu)$: image measure under *trace* of μ

Trace distribution inclusion preorder

$A_1 \leq_{TD} A_2$ iff $tdists(A_1) \subseteq tdists(A_2)$

Trace Distribution Inclusion is not Compositional

$$\begin{tikzcd}
 & q_0 \arrow[dl, "a"'] \arrow[dr, "a"] & \\
q_1 \arrow[d, "b"'] & & q_2 \arrow[d, "c"] \\
q_3 & & q_4
\end{tikzcd}
\qquad
\begin{tikzcd}
 & s_0 \arrow[d, "a"'] & \\
 & s_1 \arrow[dl, "b"'] \arrow[dr, "c"] & \\
s_2 & & s_3
\end{tikzcd}
\qquad
\begin{tikzcd}
 & c_0 \arrow[dl, "d"'] \arrow[dr] & \\
c_1 \arrow[d, "e"'] & & c_2 \arrow[d, "f"] \\
c_3 & & c_4
\end{tikzcd}$$

$$\begin{tikzcd}
 & & (s_1,c_1) \arrow[r, "e"] & (s_1,c_3) \arrow[r, "b"] & (s_2,c_3) \\
(s_0,c_0) \arrow[r, "a"] & (s_1,c_0) \arrow[ur, "d"] \arrow[dr] & & & \\
 & & (s_1,c_2) \arrow[r, "f"] & (s_1,c_4) \arrow[r, "c"] & (s_3,c_4)
\end{tikzcd}$$

How to Get Compositionality

- Restrict the power of composition
  - Probabilistic reactive modules [AHJ01]
  - Switched probabilistic I/O automata [CLSV04]

- Trace Distribution Precongruence
  - Coarsest precongruence included in preorder
    - That is: close under all contexts
  - Alternative characterizations
    - Principal context [Seg95]
    - Testing [Seg96]
    - Forward simulations [LSV03]

… yet, Proving Language Inclusion is Difficult

- Language inclusion is a global property
  - Need to see the whole result of resolving nondeterminism
- We seek local proof techniques
  - Local arguments are easier
- We use simulation relations

Simulations

Forward Simulations (Automata)

Forward simulation from A_1 to A_2 ($A_1 \leq_F A_2$)
Relation $R \subseteq Q_1 \times Q_2$ such that

$$\begin{tikzcd}
& q_0 \arrow[dl, "a"'] \arrow[dr, "a"] & & & s_0 \arrow[d, "a"] & \\
q_1 \arrow[d, "b"'] \arrow[rrrr, dotted, bend left] & & q_2 \arrow[d, "c"] \arrow[rr, dotted, bend left] & & s_1 \arrow[dl, "b"'] \arrow[dr, "c"] & \\
q_3 & & q_4 & s_3 & & s_4
\end{tikzcd}$$

$$\forall\, q, s, a, q' \;\exists\, s'$$

$$\begin{tikzcd}
s \arrow[r, Rightarrow, "a"] \arrow[d, dash, dotted, "R"'] & s' \arrow[d, dash, dotted, "R"] \\
q \arrow[r, "a"] & q'
\end{tikzcd}$$

Simulation Implies Trace Inclusion

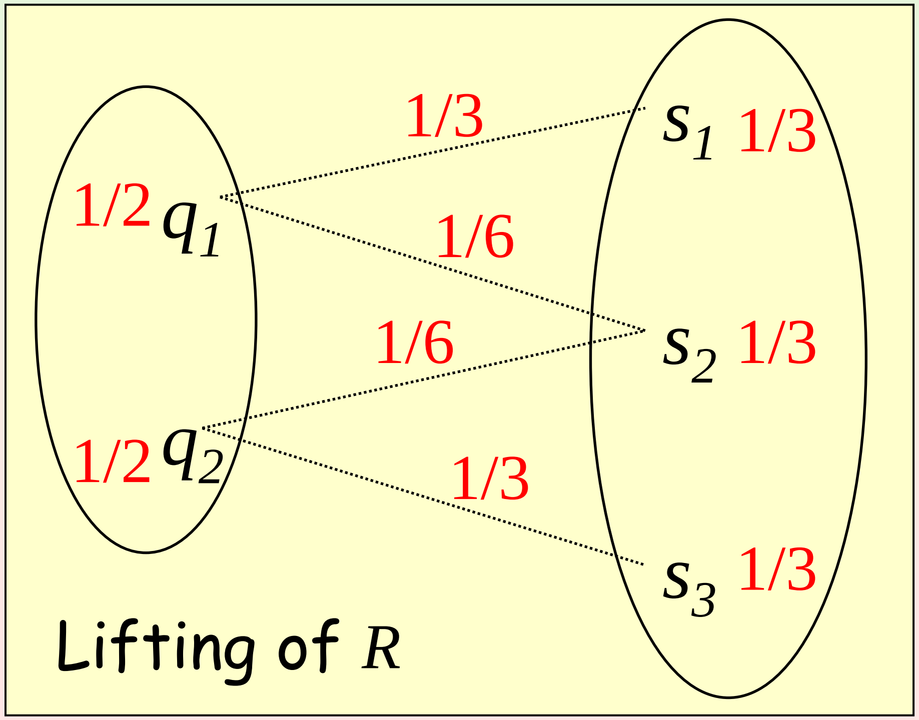
- The step condition can be applied repeatedly

$$\begin{tikzcd}
s \arrow[r, Rightarrow, "a"] \arrow[d, dotted, no head] & s_1 \arrow[r, Rightarrow, "b"] \arrow[d, dotted, no head] & s_2 \arrow[r, Rightarrow, "c"] \arrow[d, dotted, no head] & s_3 \arrow[r, Rightarrow, "d"] \arrow[d, dotted, no head] & s_4 \arrow[r, Rightarrow] \arrow[d, dotted, no head] & \cdots \\
q \arrow[r, "a"] & q_1 \arrow[r, "b"] & q_2 \arrow[r, "c"] & q_3 \arrow[r, "d"] & q_4 \arrow[r] & \cdots
\end{tikzcd}$$

- Thus existence of simulation implies trace inclusion
  - Even more it implies a close correspondence between executions

Forward Simulations

Forward simulation from A_1 to A_2 ($A_1 \leq_F A_2$)
Relation $R \subseteq Q_1 \times Q_2$ such that



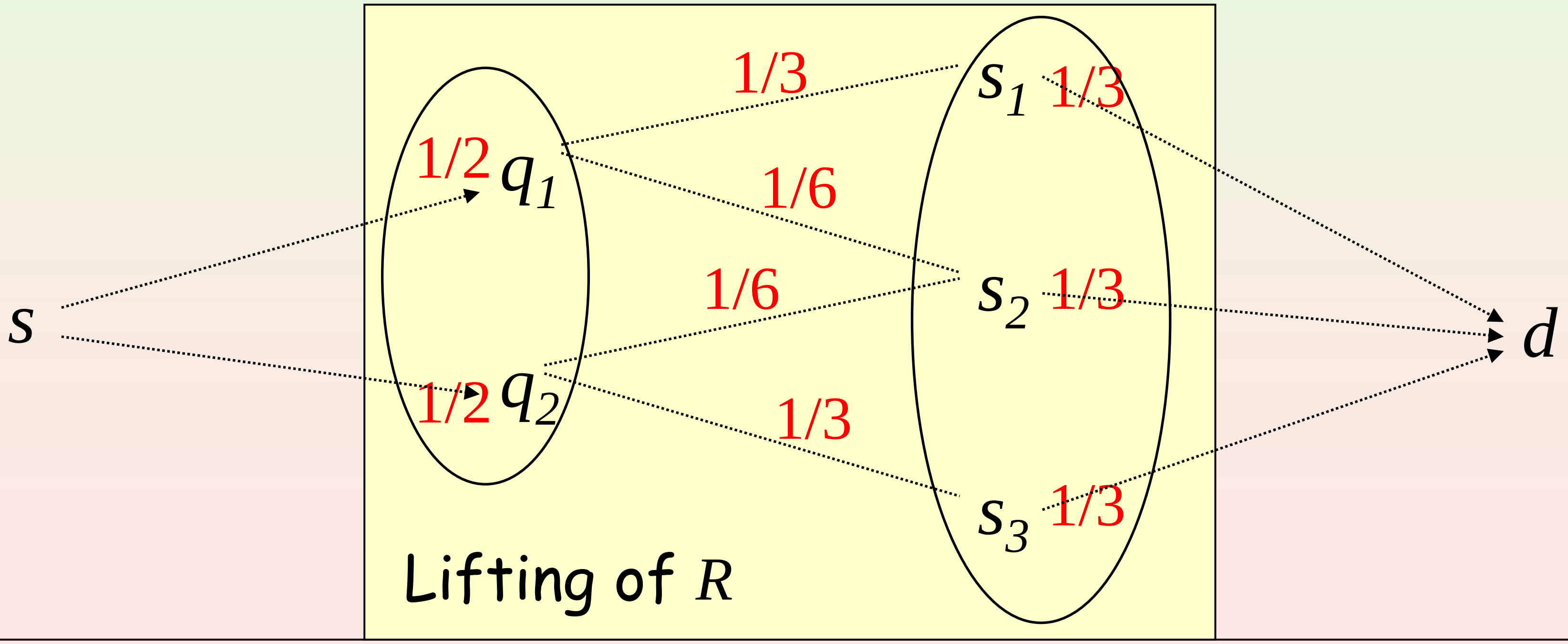
Lifting of R

$\forall\, q, s, a, \mu' \;\exists\, \sigma'$

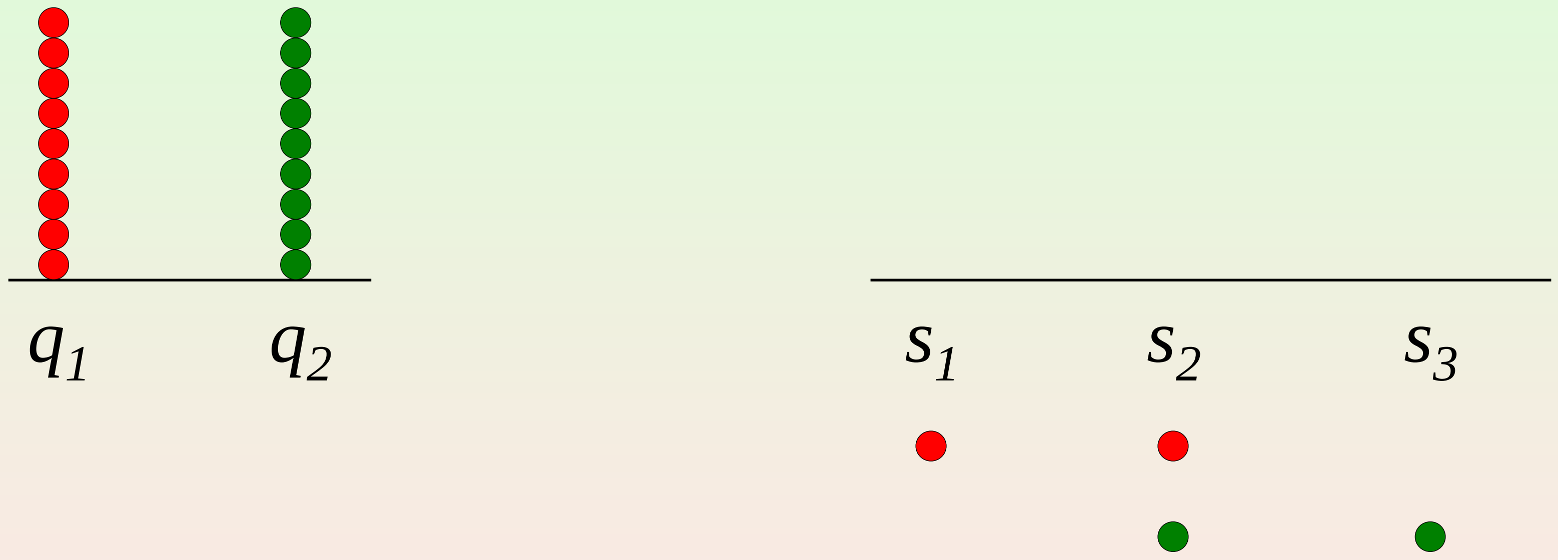
$$\begin{tikzcd}
s \arrow[r, Rightarrow, "a"] \arrow[d, dash, dotted, "R"'] & \sigma' \arrow[d, dash, dotted, "R"] \\
q \arrow[r, "a"] & \mu'
\end{tikzcd}$$

Considerations about Lifting

- It is the solution of a maximum flow problem
- Alternative characterization
  - $\mu_1\ R\ \mu_2$ iff for each upward closed set X
    - $\mu_1(X)\ \mu_2(X)$



Lifting and Transfer of Masses

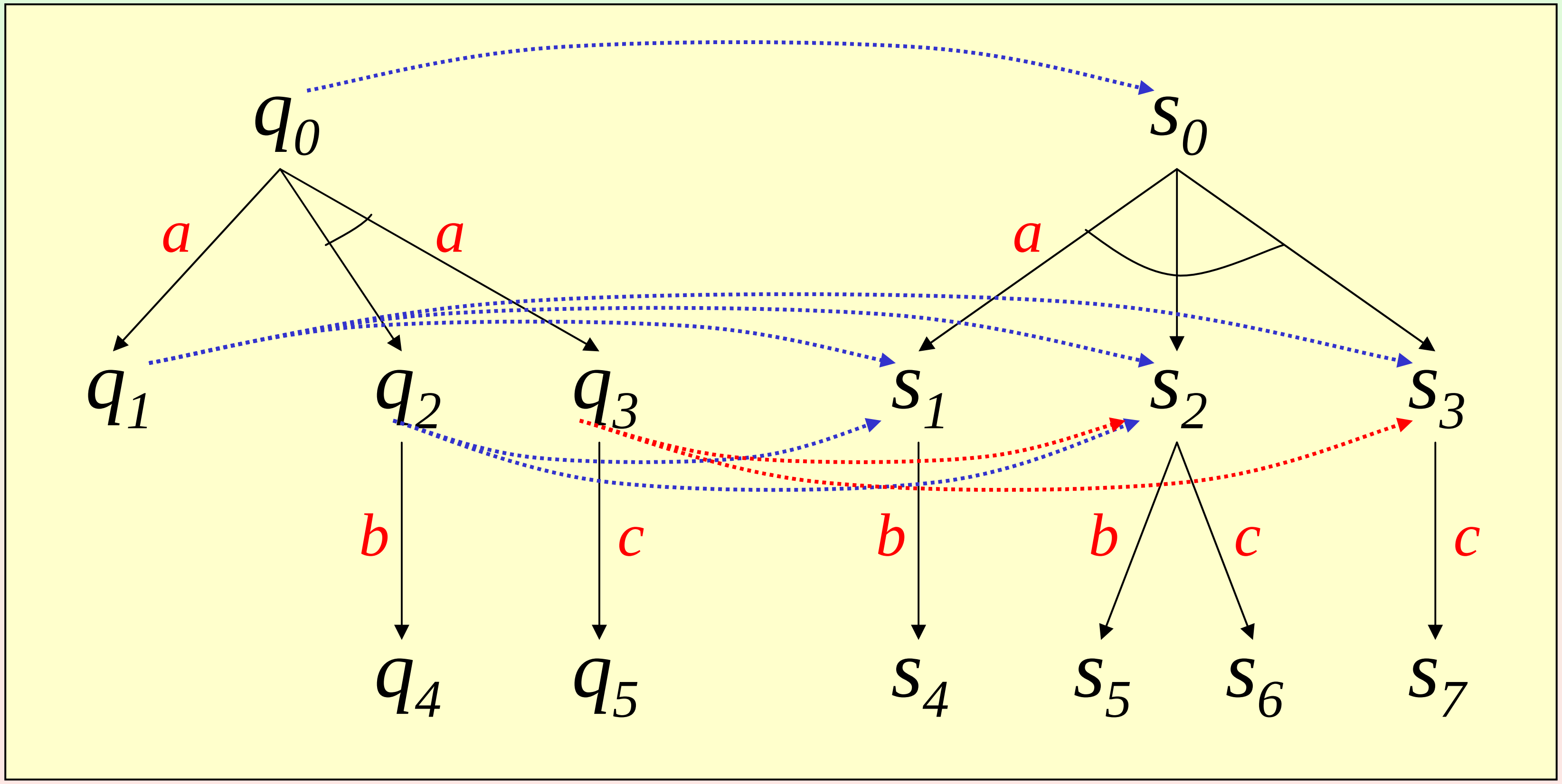


Lifting and joint Measures

μ_1 R μ_2 iff there exists a probability measure w on Q_1 ´ Q_2 such that

- support(w) $\subseteq$ R
  - That is, $w(s_1,s_2)>0$ implies s_1 R s_2
- $w(s_1,Q_2) = \mu_1(s_1)$
  - That is, the left marginal is μ_1
- $w(Q_1,s_2) = \mu_2(s_2)$
  - That is, the right marginal is μ_2

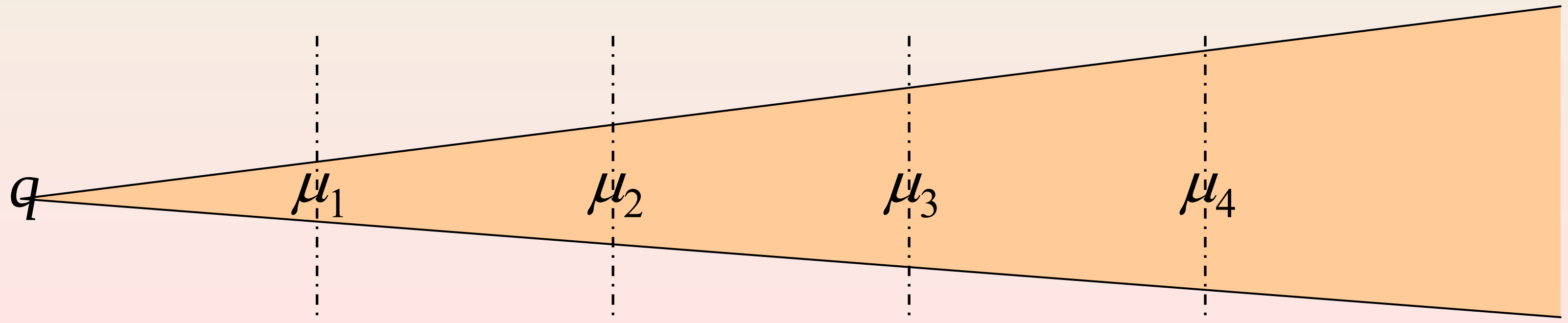
Example: Simulations



Simulation Implies Trace Inclusion

- The step condition can be applied repeatedly

$$\begin{tikzcd}
s \arrow[r, Rightarrow] \arrow[d, dotted, no head] & \rho_1 \arrow[r, Rightarrow] \arrow[d, dotted, no head] & \rho_2 \arrow[r, Rightarrow] \arrow[d, dotted, no head] & \rho_3 \arrow[r, Rightarrow] \arrow[d, dotted, no head] & \rho_4 \arrow[r, Rightarrow] \arrow[d, dotted, no head] & \cdots \\
q \arrow[r] & \mu_1 \arrow[r] & \mu_2 \arrow[r] & \mu_3 \arrow[r] & \mu_4 \arrow[r] & \cdots
\end{tikzcd}$$



Probabilistic I/O Automata

- Probabilistic Automata where
  - External actions partitioned
    - Input actions
    - Output actions
  - Input actions always enabled
- In parallel composition
  - Each action is output of at most one automaton
- Therefore
  - The environment nevel bkocks output actions
  - Language inclusion preserves more properties
  - We know always who controls each action

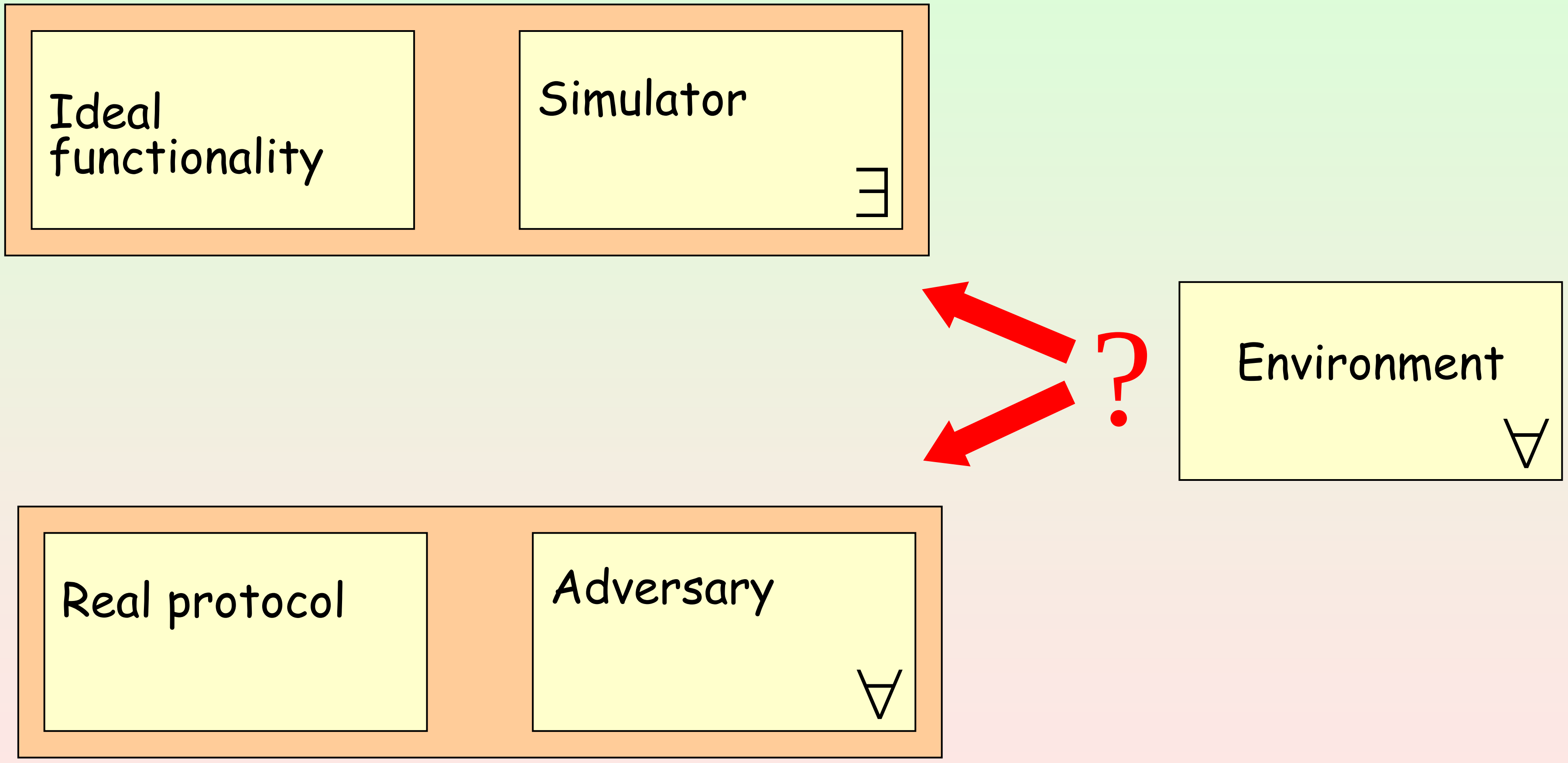
Case Study:

Oblivious Transfer

Even, Goldreich, Lempel 85

Canetti, Cheung, Kaynar,Liskov, Lynch, Pereira, Segala

UC-Framework [Canetti]



Oblivious Transfer

- Ideal functionality
  - Receive
    - input $x \in \{0,1\} \longrightarrow \{0,1\}$ (just to avoid writing x_0, x_1)
    - input $i \in \{0,1\}$
  - Return
    - $x(i)$ (or could be x_i)
- Failure model
  - Either Transmitter or Receiver may be corrupt
  - Adversary sees input of faulty agents
  - Faulty agents send output to adversary
  - Adversary may only forward messages and/or talk to environment
- In practice we have four cases
  - We consider case where no agent is faulty

Automaton for Ideal Functionality No Faulty Agents

Signature

Input

$in(x)_T$ $x \in \{0,1\} \longrightarrow \{0,1\}$

$in(i)_R$ $i \in \{0,1\}$

Output

$out(w)_R$

State

$xval \in \{0,1\} \longrightarrow \{0,1\}$ initially $\perp$

$ival \in \{0,1\} \cup \{\perp\}$ initially $\perp$

Transitions

$in(x)_T$

Effect

If $xval = \perp$ then $xval := x$

$in(i)_R$

Effect

If $ival = \perp$ then $ival := i$

$out(w)_R$

Pre

$xval, ival \neq \perp$

$w = xval(ival)$

Effect

none

wait

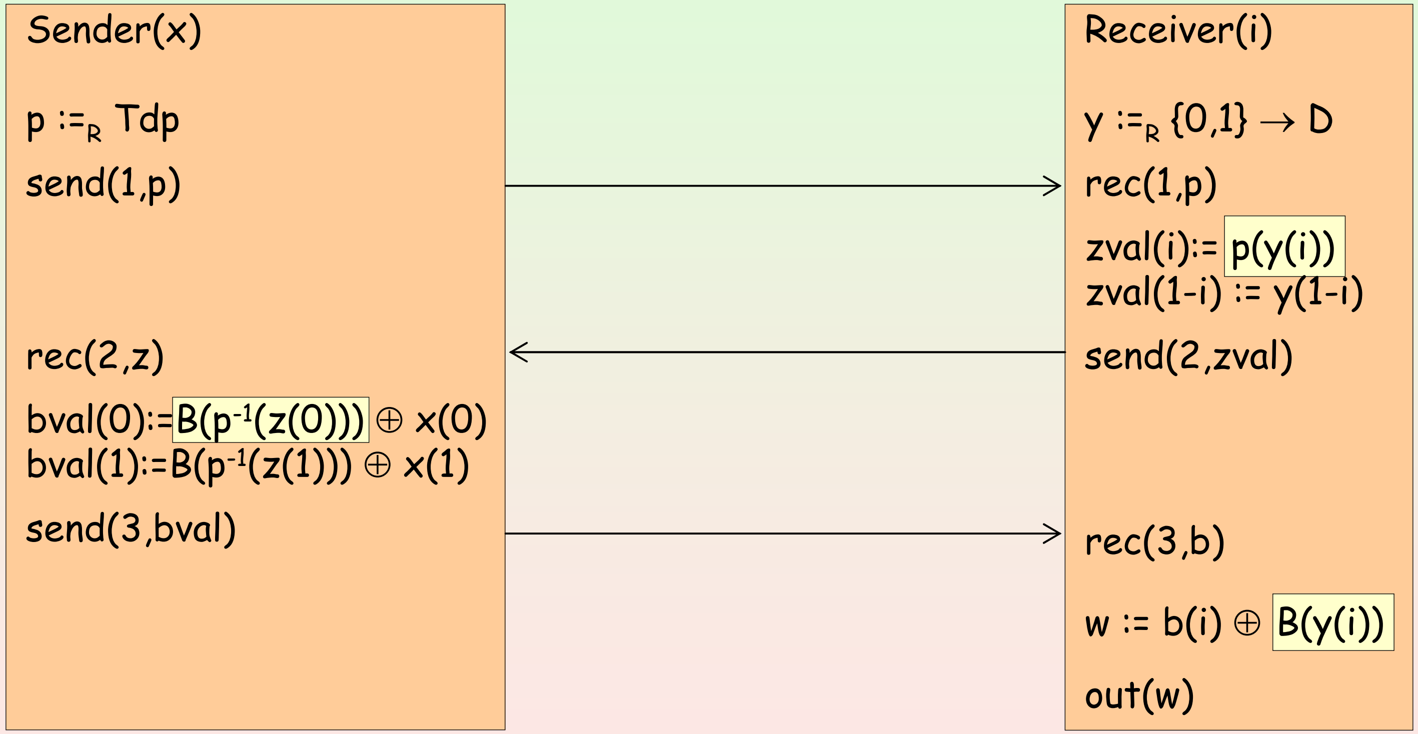
$in(x)_T$, $in(i)_R$

output

$out(x(i))_R$

The Protocol

Tdp:	trap-door permutation
D:	domain of Tdp
B:	hard-core predicate for Tdp

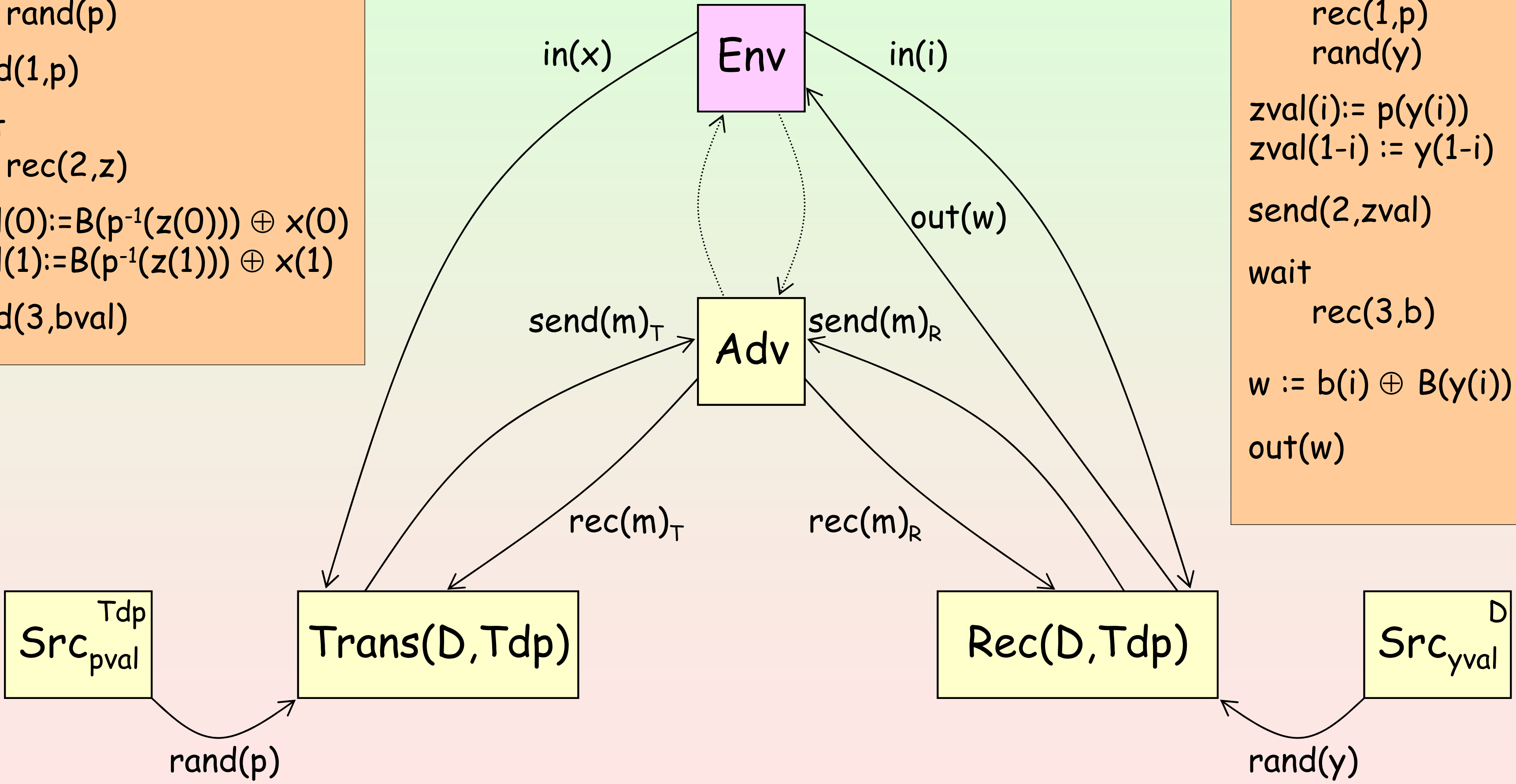


```
Sender(x)                                              Receiver(i)

p :=_R Tdp                                             y :=_R {0,1} → D
send(1,p)          ──────────────────────────────→     rec(1,p)
                                                       zval(i):= p(y(i))
                                                       zval(1-i) := y(1-i)
rec(2,z)           ←──────────────────────────────     send(2,zval)
bval(0):=B(p^-1(z(0))) ⊕ x(0)
bval(1):=B(p^-1(z(1))) ⊕ x(1)
send(3,bval)       ──────────────────────────────→     rec(3,b)
                                                       w := b(i) ⊕ B(y(i))
                                                       out(w)
```

$$\boxed{b(i)} \oplus B(y(i)) = \boxed{B(p^{-1}(z(i)))}\boxed{\oplus x(i)} \oplus B(y(i)) = \boxed{B(y(i))} \oplus x(i) \oplus \boxed{B(y(i))} = x(i)$$

Real Protocol

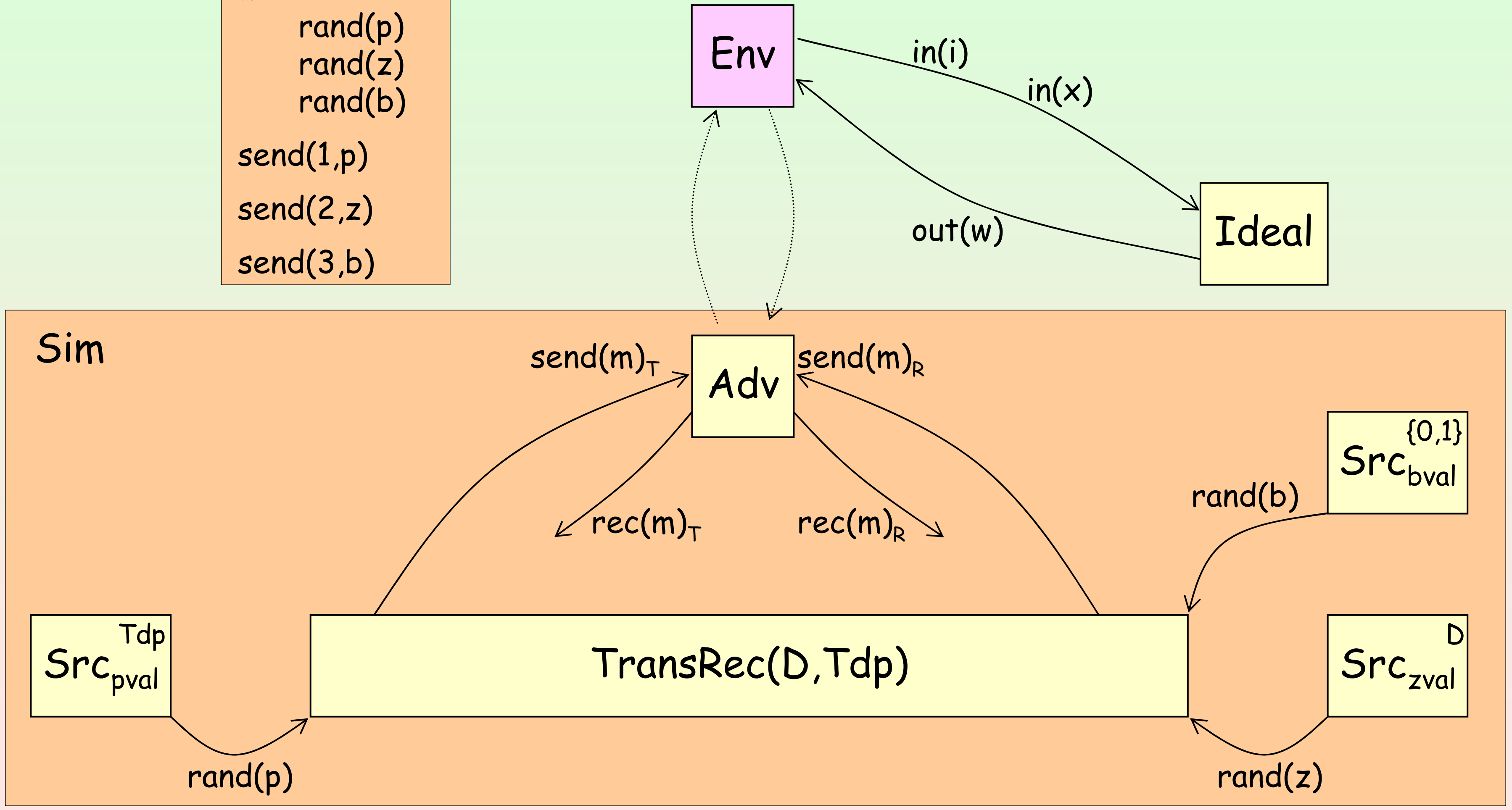
```
wait
    in(x)
    rand(p)
send(1,p)
wait
    rec(2,z)
bval(0):=B(p^{-1}(z(0))) ⊕ x(0)
bval(1):=B(p^{-1}(z(1))) ⊕ x(1)
send(3,bval)
```



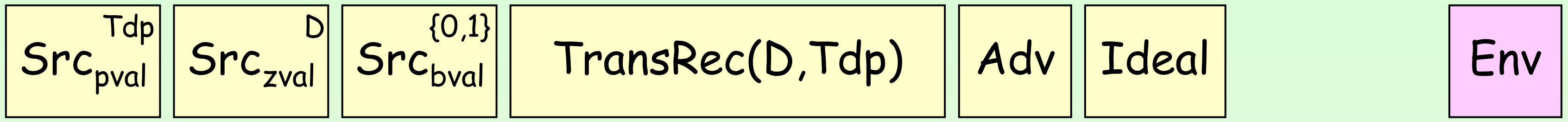
```
wait
    in(i)
    rec(1,p)
    rand(y)
zval(i):= p(y(i))
zval(1-i) := y(1-i)
send(2,zval)
wait
    rec(3,b)
w := b(i) ⊕ B(y(i))
out(w)
```

Ideal Protocol with Simulator

```
wait
      rand(p)
      rand(z)
      rand(b)
send(1,p)
send(2,z)
send(3,b)
```



What we should Prove

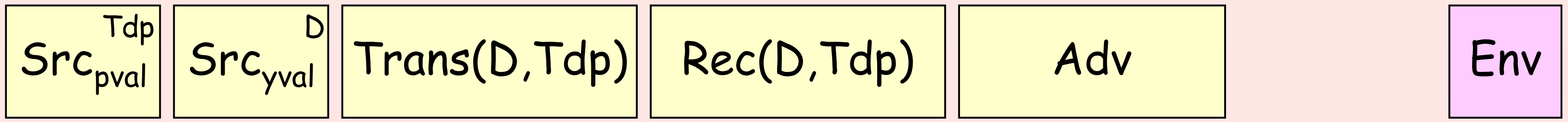

$\mathrm{Src}_{pval}^{Tdp}$
Src_{zval}^{D}
$\mathrm{Src}_{bval}^{\{0,1\}}$
TransRec(D,Tdp)
Adv
Ideal
Env

Objective:

Env should not distinguish real from ideal
Let Env have a special accept action

$\leq_{neg,pt}$

for each PPT environment Env
for each trace distribution of Real | Env
there exists a trace distribution of Ideal | Env
the probabilities of accept differ by a negligible value



Implementation Relation Extends Computational Indistinguishability

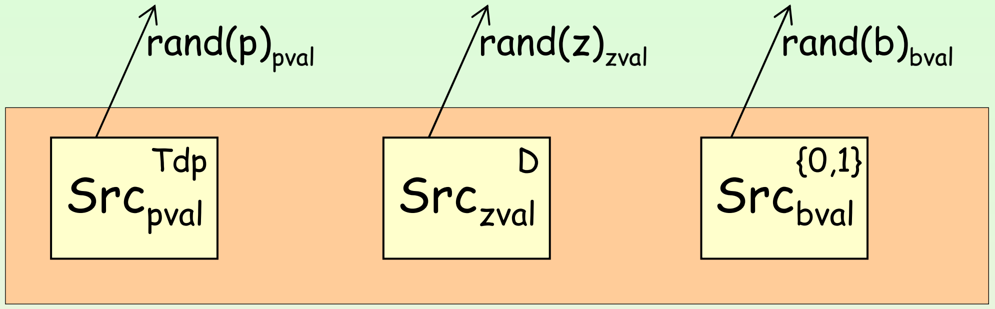
- Families of probabilistic automata
  - Indexed by security parameter k
- Time bounded automata (by some polynomial p)
  - Elements representable with p(k) bits
  - Elements computable in time p(k)
- $\{A_k\} \leq_{neg,pt} \{B_k\}$ iff
  - For each polynomial p, p_1
  - There exists a polynomial p_2
  - There exists a function ε negligible in k $(\forall c \exists \mathbf{k} \forall k > \mathbf{k})$
  - For each Environment $\{E_k\}$
    - p-bounded
    - with special action accept
  - For each trace distribution of $A_k | E_k$ of length at most $p_1(k)$
  - There exists a trace distribution of $B_k | E_k$ of length at most $p_2(k)$
    - Probabilities of accept differ at most by $\varepsilon(k)$ $(\leq k^{-c})$

Hard Core Predicate
Trap-door permutation

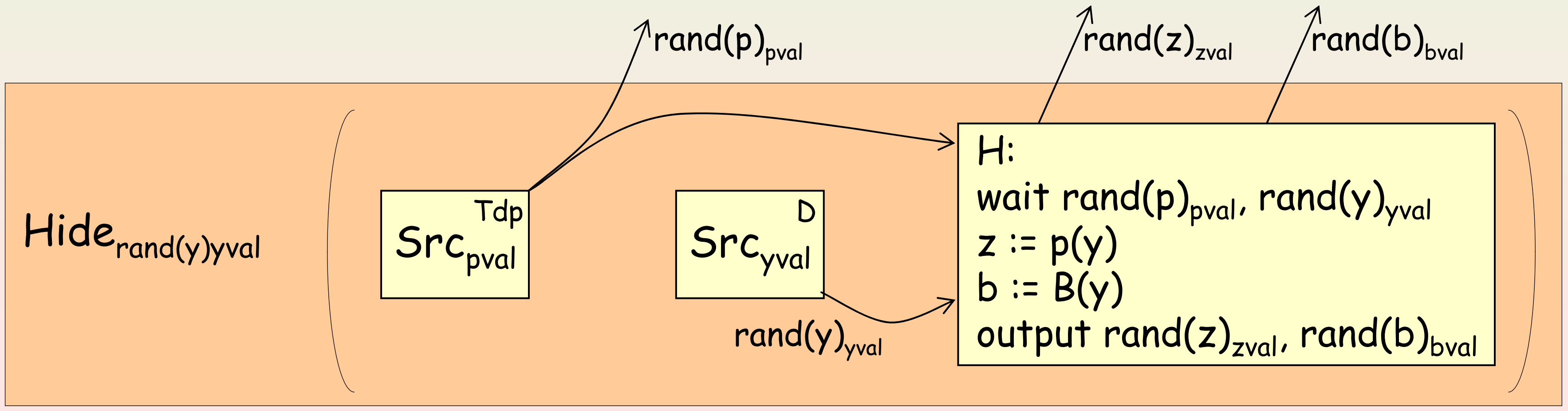
- Domain $D = \{D_k\}$
- Trap-door permutation $Tdp = \{Tdp_k\}$
- Hard-core predicate $B : \{D_k \to \{0,1\}\}$
  - Poly-time computable
  - For each poly-time predicate G there exists negligible ε

$$\left| \Pr\left[\begin{array}{l} f \leftarrow Tdp_k; \\ z \leftarrow D_k \\ b \leftarrow B(f^{-1}(z)); \\ G_k(f,z,b) = 1 \end{array}\right] - \Pr\left[\begin{array}{l} f \leftarrow Tdp_k; \\ z \leftarrow D_k \\ b \leftarrow \{0,1\}; \\ G_k(f,z,b) = 1 \end{array}\right] \right| \leq \varepsilon(k)$$

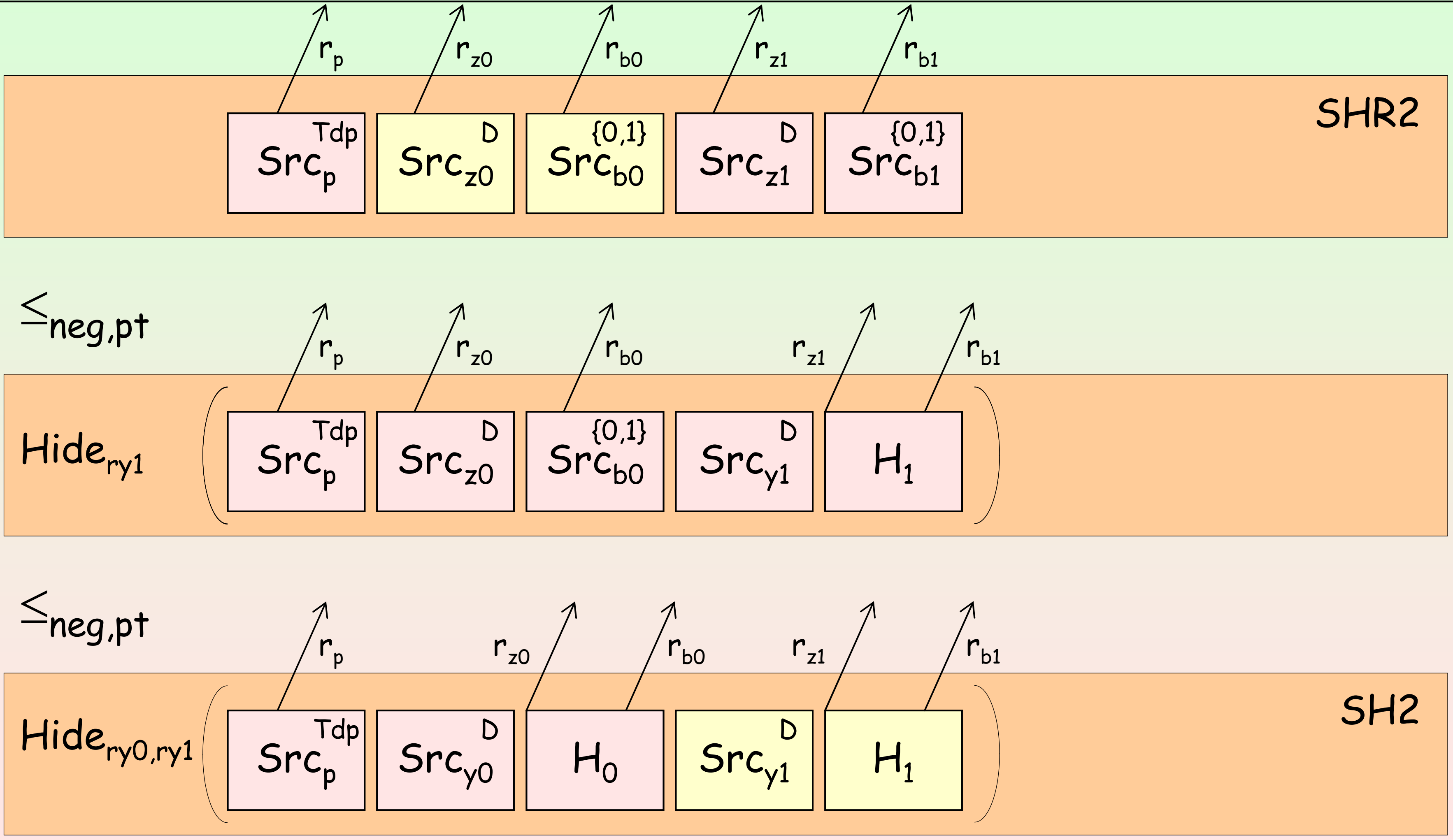
Hard-Core Predicate Definition as Implementation



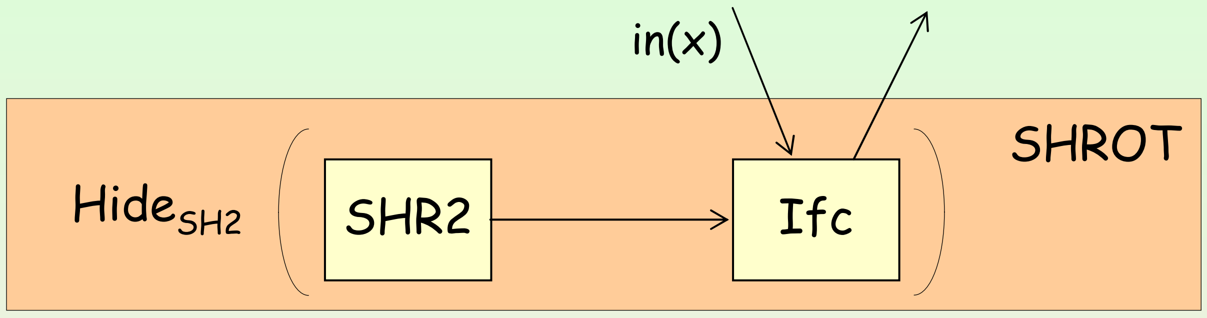
$\leq_{neg,pt}$



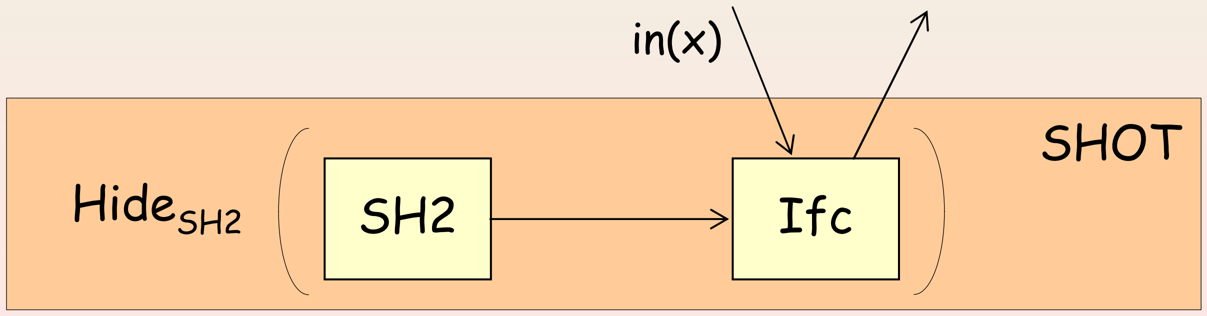
Playing with Hard-Core Predicates



Playing with Hard-Core Predicates



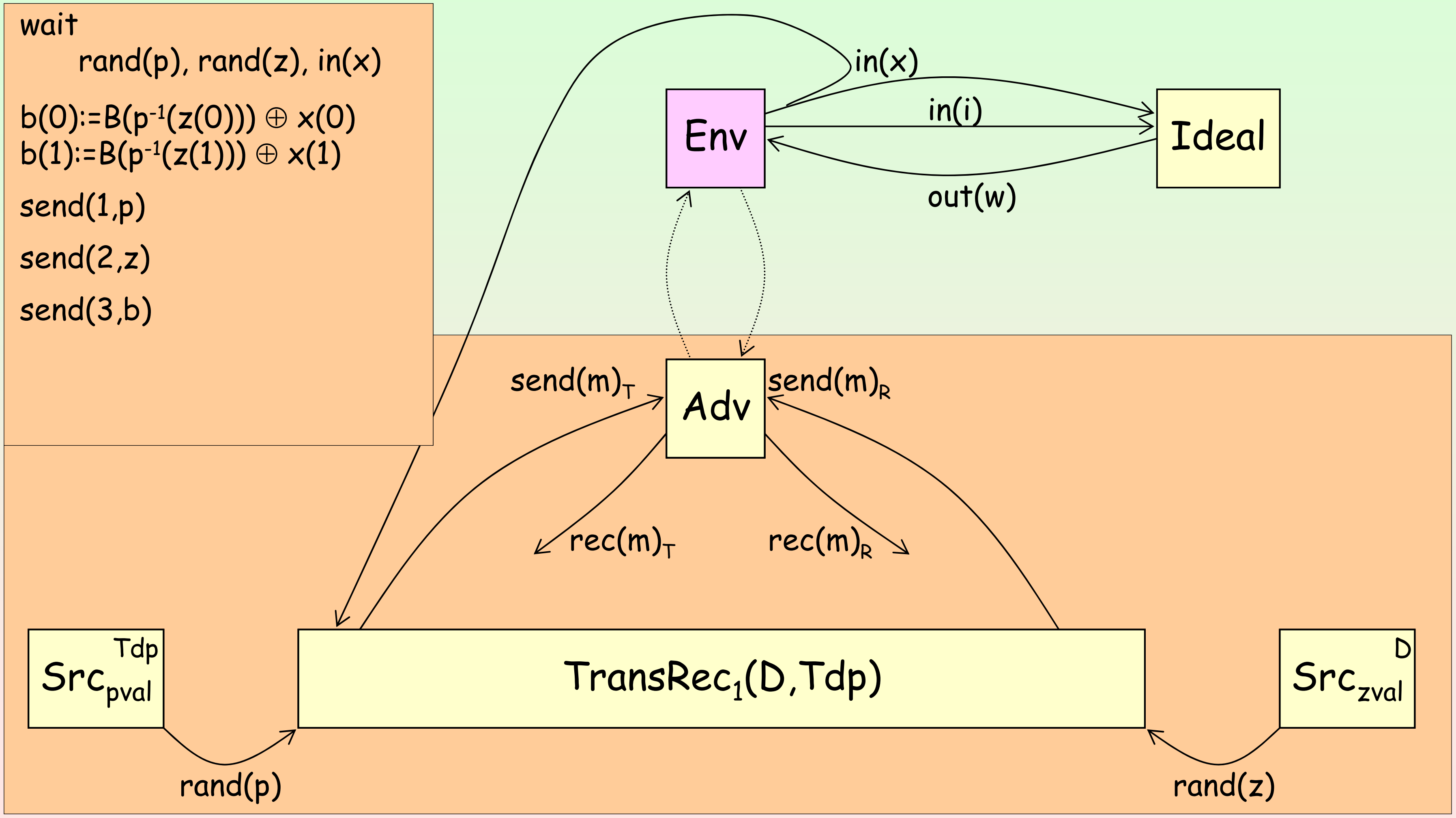
$\leq_{neg,pt}$



```
Ifc:

wait
    in(x)
    $r_p, r_{z0}, r_{b0}, r_{z1}, r_{b1}$

$b(0) := x(0) \oplus b_0$
$b(1) := x(1) \oplus b_1$
$z(0) := z_0$
$z(1) := z_1$

send(1,p)
send(2,z)
send(3,b)
```

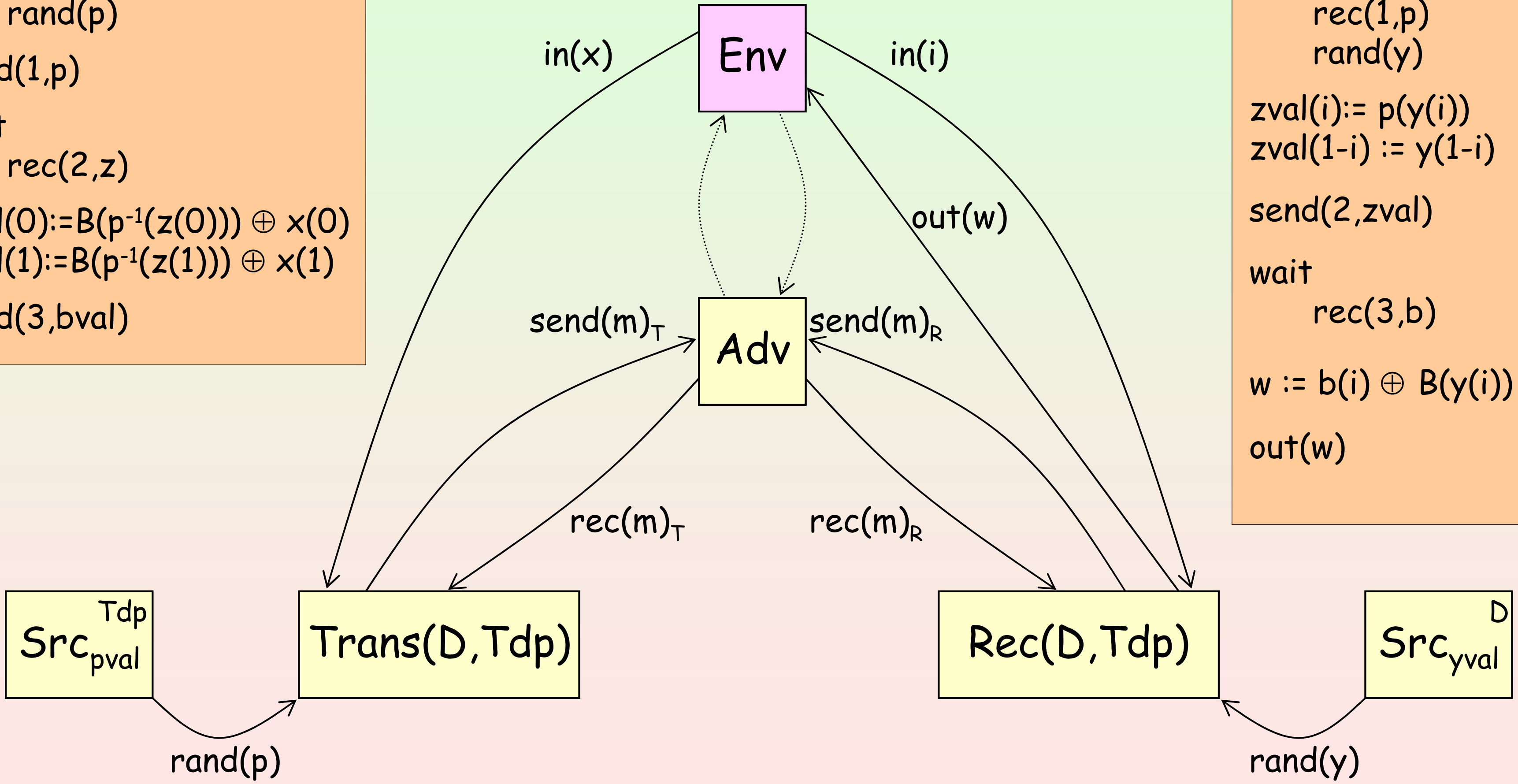
Ideal Protocol with Intermediate Simulator 1

```
wait
    rand(p), rand(z), in(x)
b(0):=B(p^{-1}(z(0))) ⊕ x(0)
b(1):=B(p^{-1}(z(1))) ⊕ x(1)
send(1,p)
send(2,z)
send(3,b)
```



Real Protocol

```
wait
    in(x)
    rand(p)
send(1,p)
wait
    rec(2,z)
bval(0):=B(p^{-1}(z(0))) ⊕ x(0)
bval(1):=B(p^{-1}(z(1))) ⊕ x(1)
send(3,bval)
```



```
wait
    in(i)
    rec(1,p)
    rand(y)
zval(i):= p(y(i))
zval(1-i) := y(1-i)
send(2,zval)
wait
    rec(3,b)
w := b(i) ⊕ B(y(i))
out(w)
```

The Proof

Src^{Tdp}_{pval} Src^{D}_{zval} $Src^{\{0,1\}}_{bval}$ TransRec(D,Tdp) Adv Ideal

```
wait
    rand(p), rand(z),
    rand(c), in(x)
b(0):=c(0) ⊕ x(0)
b(1):=c(1) ⊕ x(1)
send(1,p)
send(2,z)
send(3,b)
```

```
wait
    in(x)
    rand(p)
send(1,p)
wait
    rec(2,z)
bval(0):=B(p^{-1}(z(0))) ⊕ x(0)
bval(1):=B(p^{-1}(z(1))) ⊕ x(1)
send(3,bval)
```

```
wait
    in(i)
    rec(1,p)
    rand(y)
zval(i):= p(y(i))
zval(1-i) := y(1-i)
send(2,zval)
wait
    rec(3,b)
w := b(i) ⊕ B(y(i))
out(w)
```

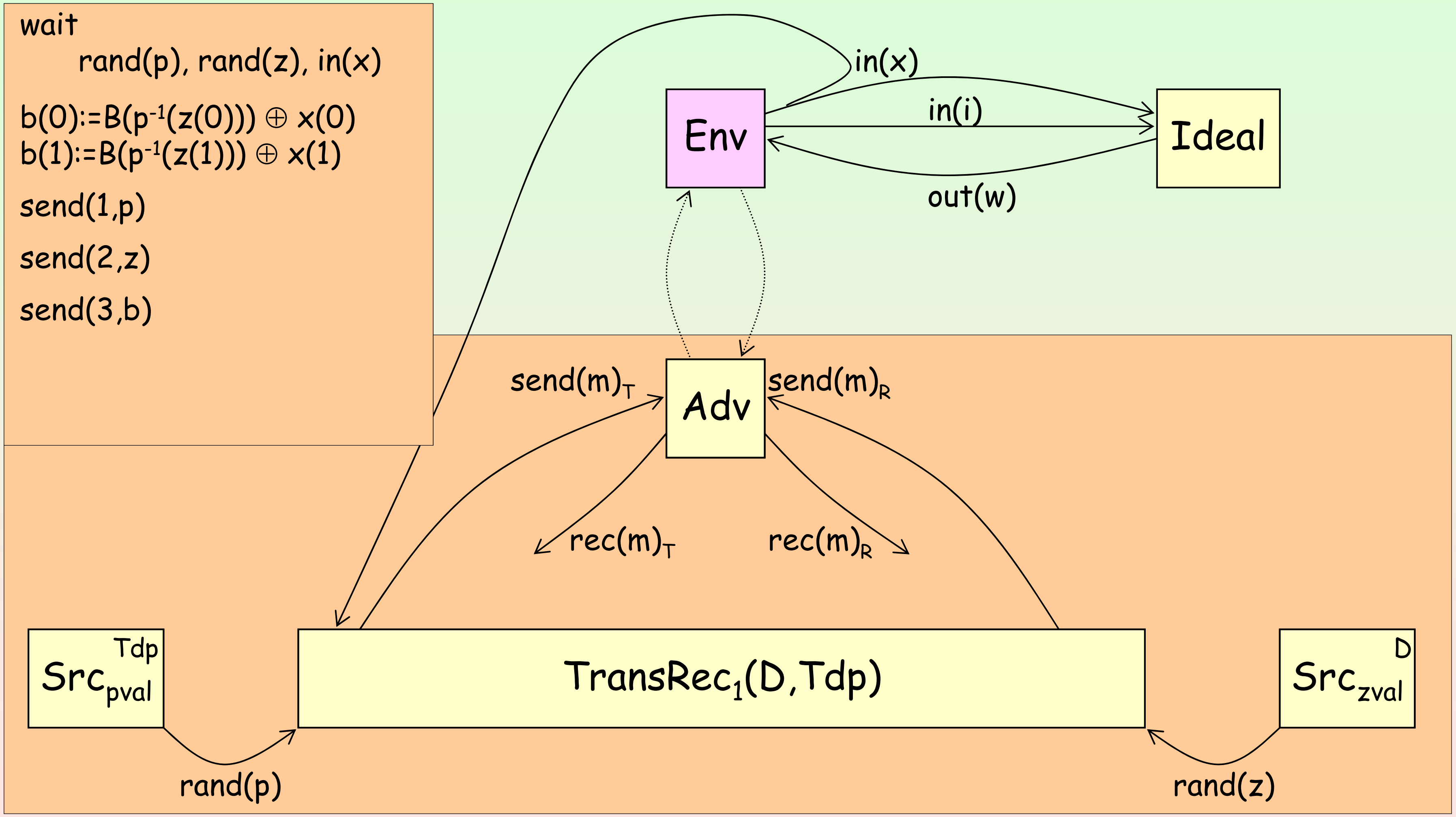
Src^{Tdp}_{pval} Src^{D}_{zval} $TransRec_1(D,Tdp)$ Adv Ideal Env

$\leq_0$

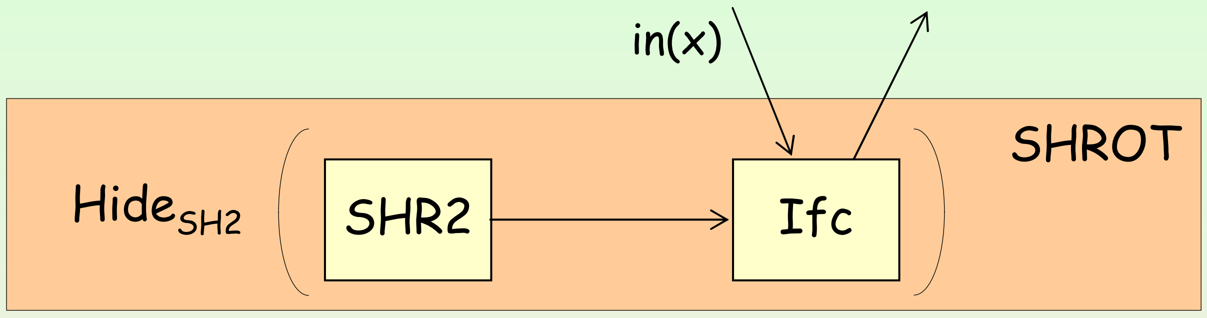
Src^{Tdp}_{pval} Src^{D}_{yval} Trans(D,Tdp) Rec(D,Tdp) Adv Env

Ideal Protocol with Intermediate Simulator 1

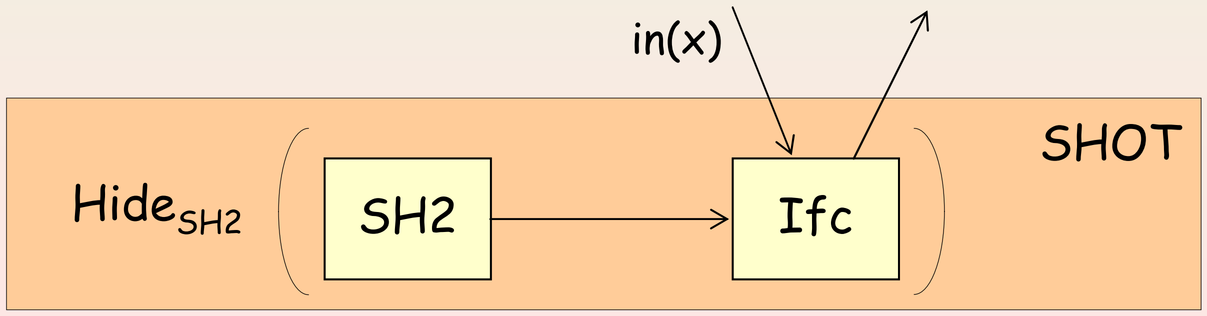
```
wait
    rand(p), rand(z), in(x)
b(0):=B(p^{-1}(z(0))) ⊕ x(0)
b(1):=B(p^{-1}(z(1))) ⊕ x(1)
send(1,p)
send(2,z)
send(3,b)
```



Playing with Hard-Core Predicates

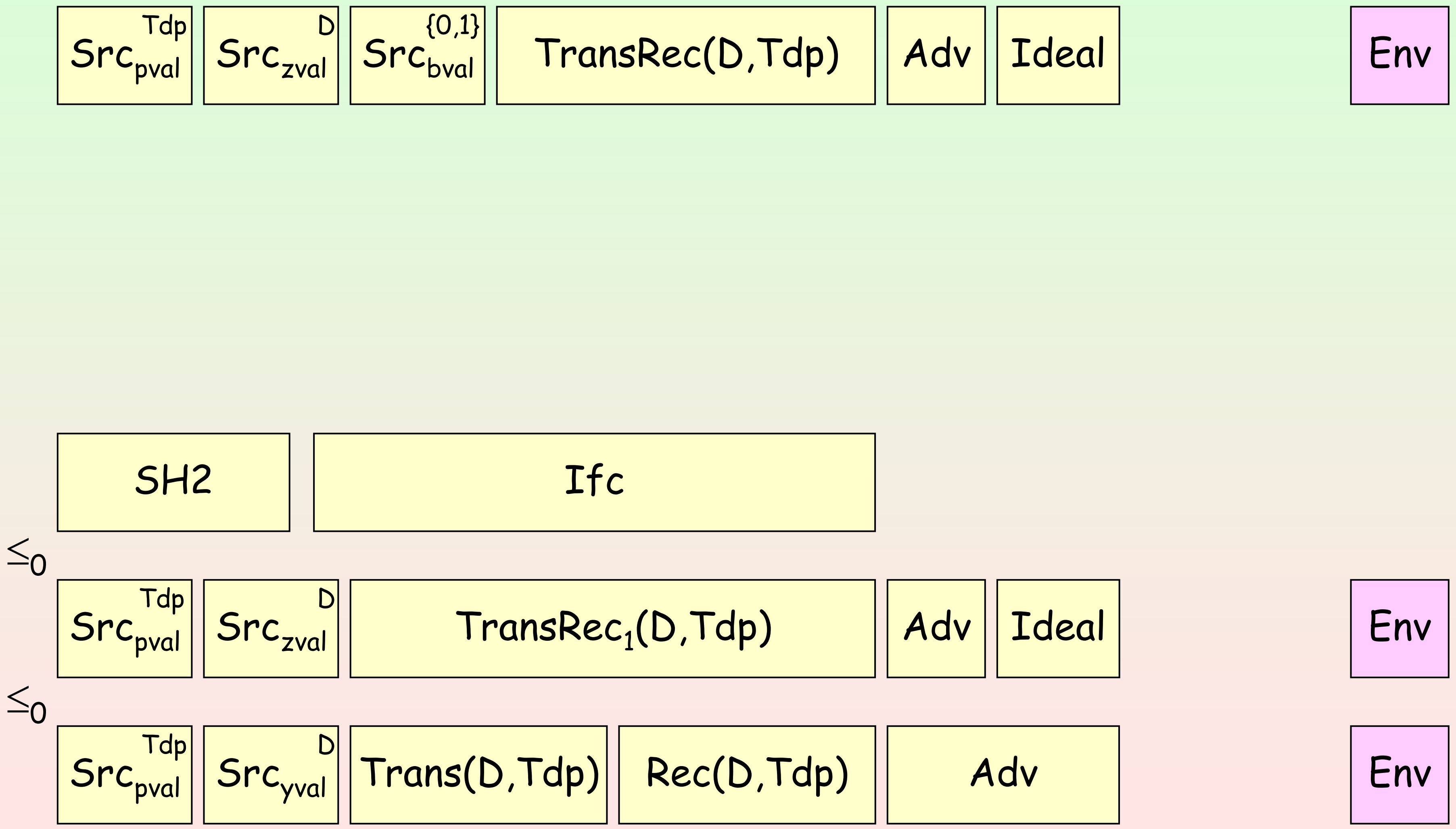


$\leq_{neg,pt}$



```
Ifc:

wait
    in(x)
    r_p, r_z0, r_b0, r_z1, r_b1

b(0) := x(0) ⊕ b_0
b(1) := x(1) ⊕ b_1
z(0) := z_0
z(1) := z_1

send(1,p)
send(2,z)
send(3,b)
```

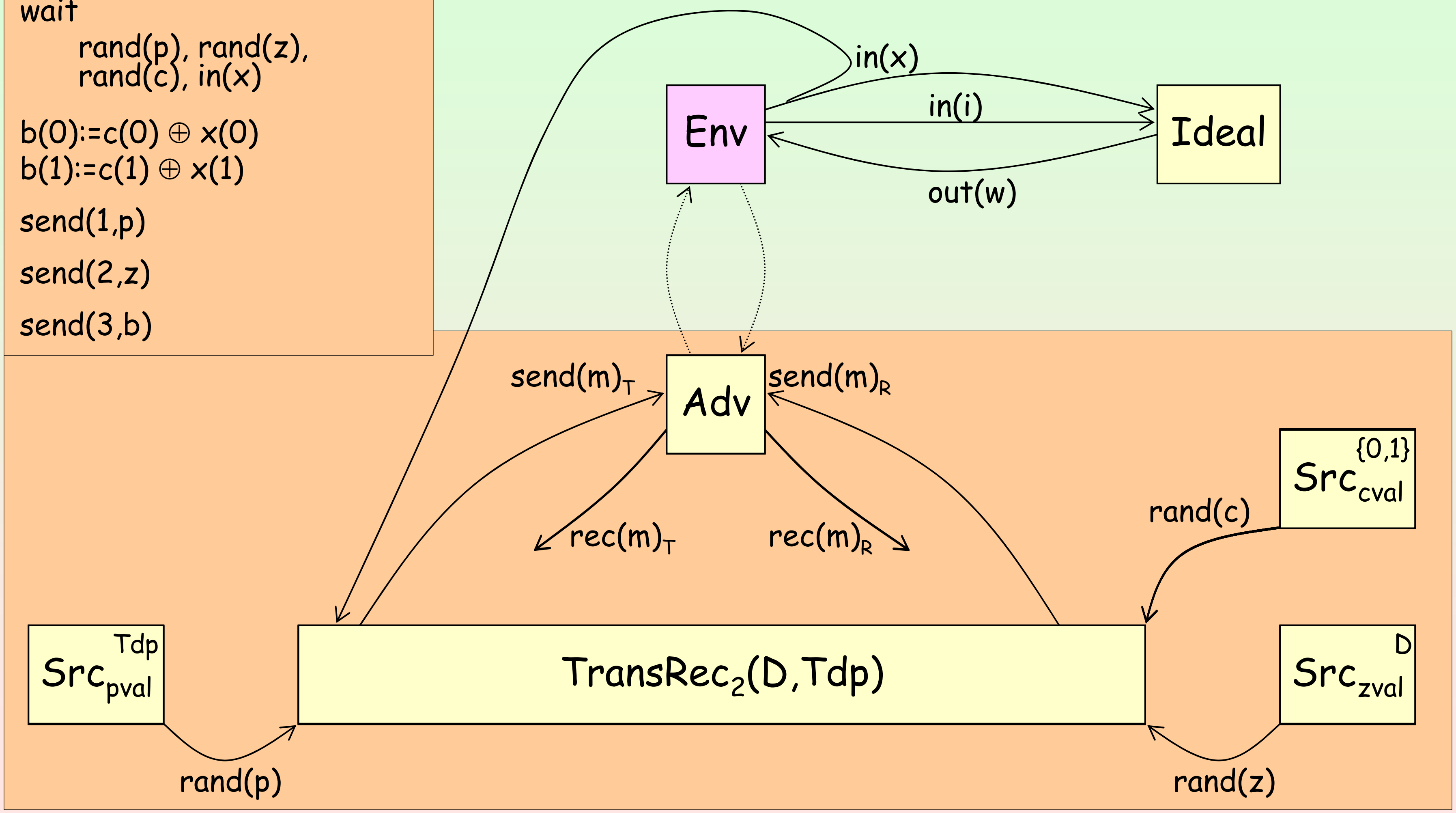
The Proof



$\mathrm{Src}_{pval}^{Tdp}$
Src_{zval}^{D}
$\mathrm{Src}_{bval}^{\{0,1\}}$
TransRec(D,Tdp)
Adv
Ideal
Env
SH2
Ifc
$\leq_0$
$\mathrm{Src}_{pval}^{Tdp}$
Src_{zval}^{D}
$\mathrm{TransRec}_1(D,Tdp)$
Adv
Ideal
Env
$\leq_0$
$\mathrm{Src}_{pval}^{Tdp}$
Src_{yval}^{D}
Trans(D,Tdp)
Rec(D,Tdp)
Adv
Env

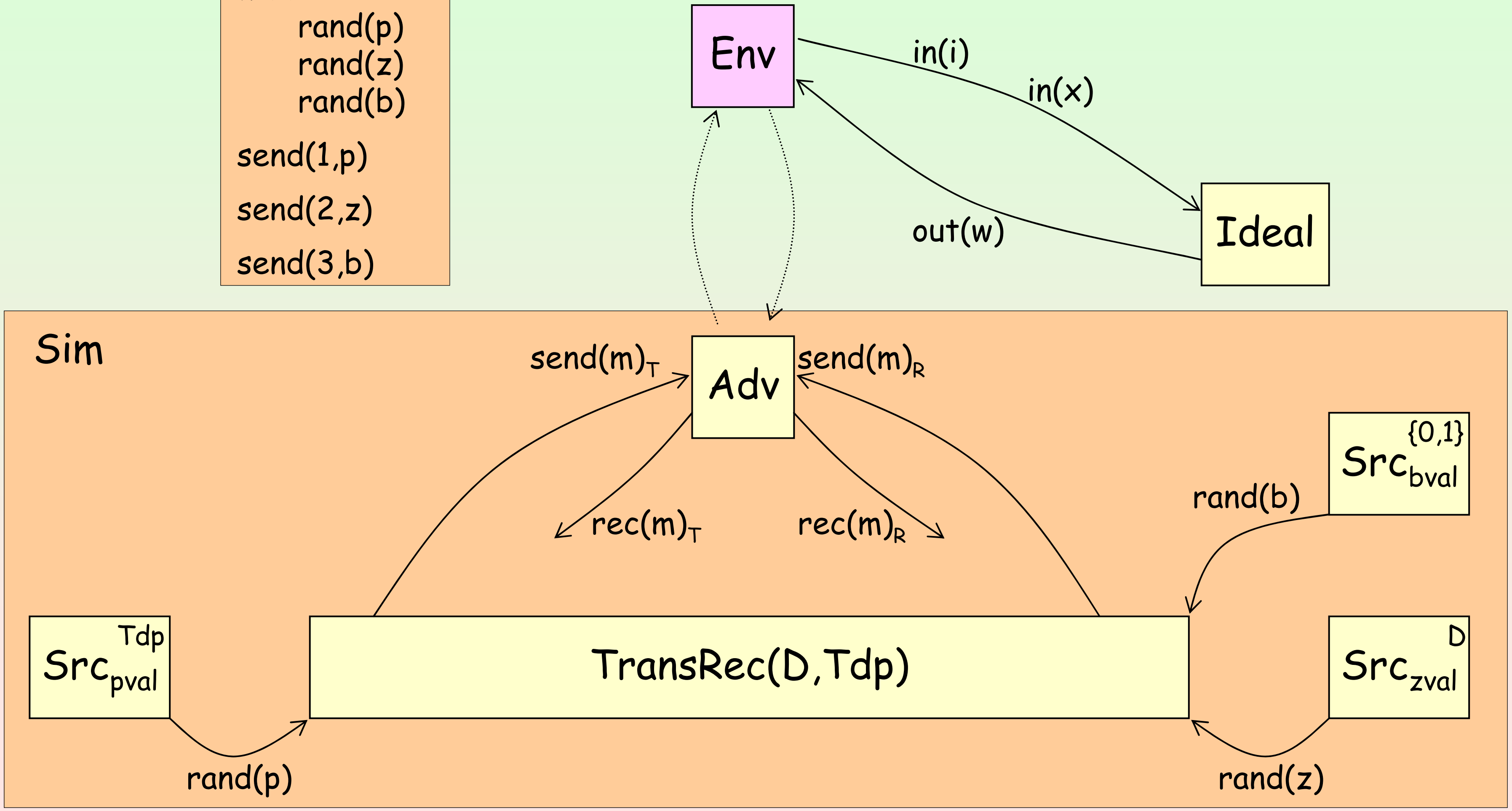
Ideal Protocol with Intermediate Simulator 2

```
wait
    rand(p), rand(z),
    rand(c), in(x)

b(0):=c(0) ⊕ x(0)
b(1):=c(1) ⊕ x(1)

send(1,p)

send(2,z)

send(3,b)
```

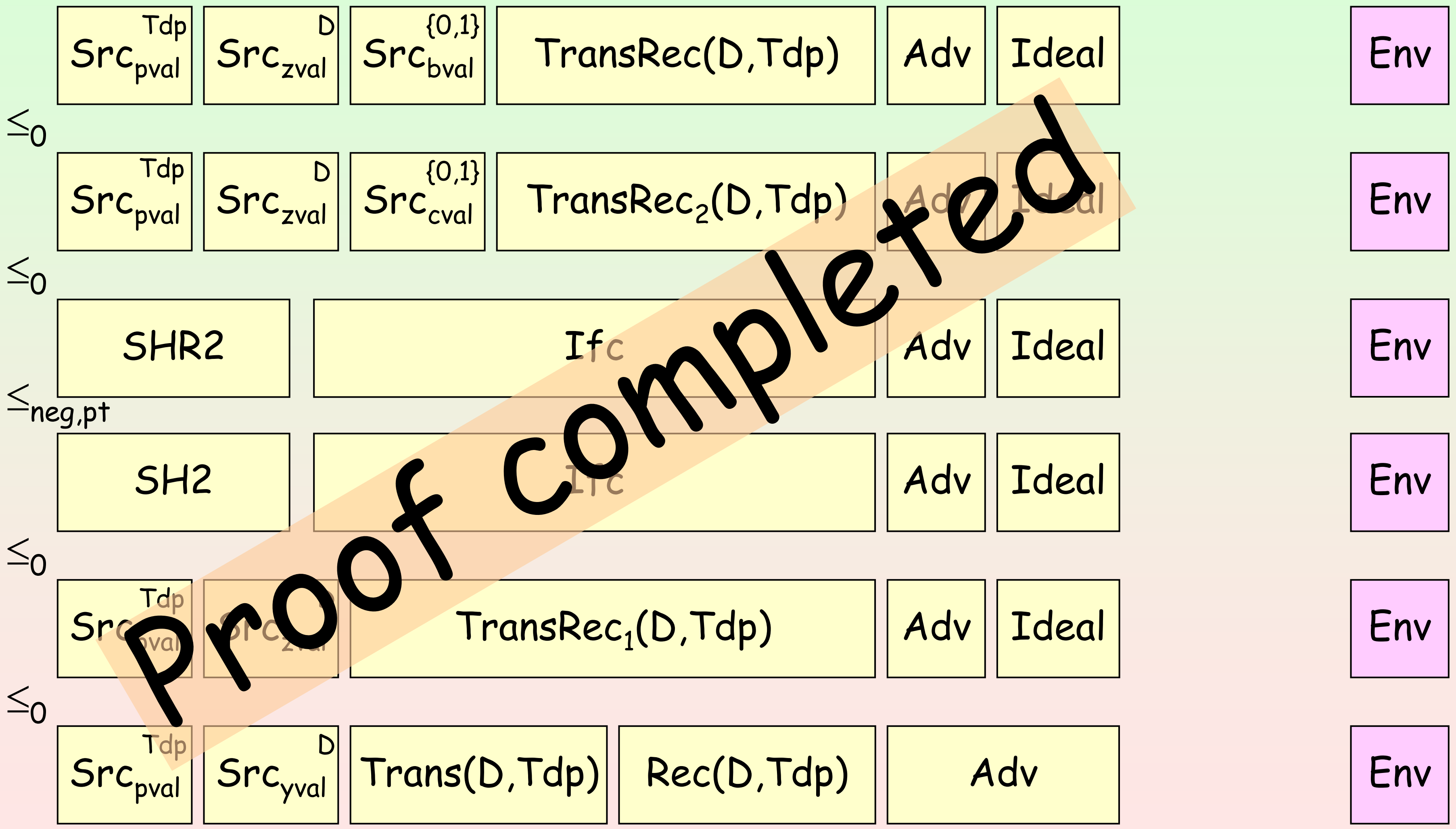


Ideal Protocol with Simulator

```
wait
      rand(p)
      rand(z)
      rand(b)
send(1,p)
send(2,z)
send(3,b)
```



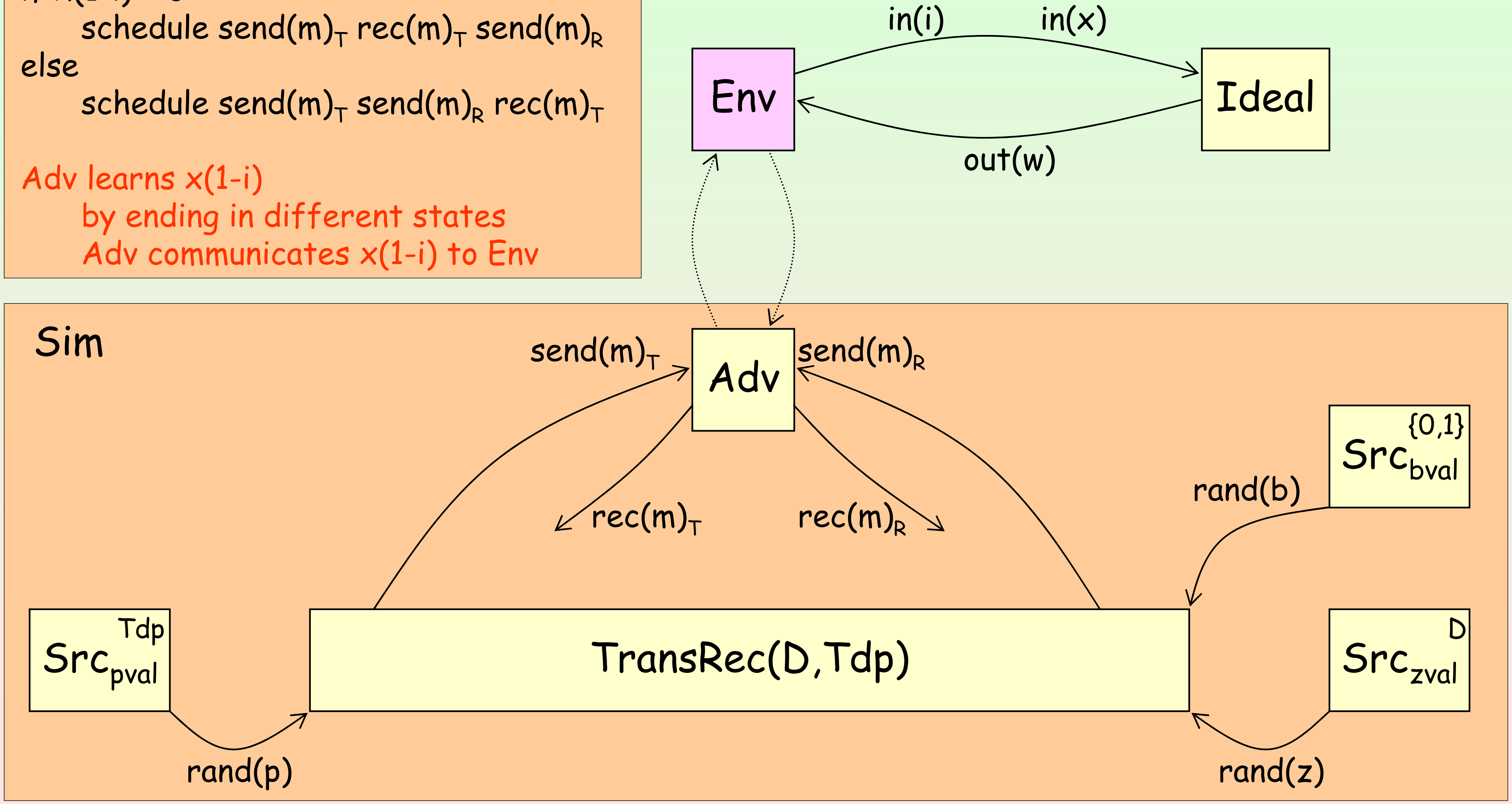
The Proof



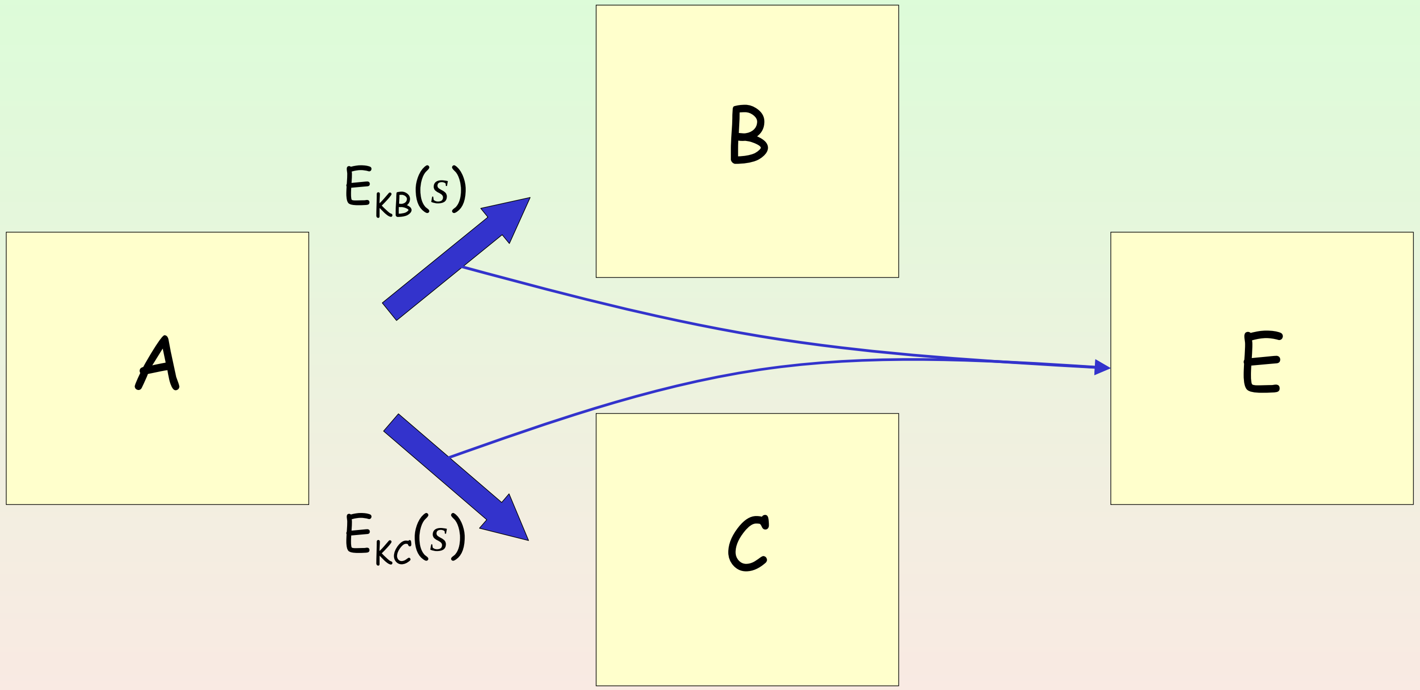
Src^{Tdp}_{pval} Src^{D}_{zval} $\text{Src}^{\{0,1\}}_{bval}$ TransRec(D,Tdp) Adv Ideal Env
$\leq_0$
Src^{Tdp}_{pval} Src^{D}_{zval} $\text{Src}^{\{0,1\}}_{cval}$ TransRec_2(D,Tdp) Adv Ideal Env
$\leq_0$
SHR2 Ifc Adv Ideal Env
$\leq_{neg,pt}$
SH2 Ifc Adv Ideal Env
$\leq_0$
Src^{Tdp}_{pval} Src^{D}_{zval} TransRec_1(D,Tdp) Adv Ideal Env
$\leq_0$
Src^{Tdp}_{pval} Src^{D}_{yval} Trans(D,Tdp) Rec(D,Tdp) Adv Env
Proof completed

Problems with Nondeterminism
Ideal Protocol with Simulator

```
if x(1-i) = 0
    schedule send(m)_T rec(m)_T send(m)_R
else
    schedule send(m)_T send(m)_R rec(m)_T

Adv learns x(1-i)
    by ending in different states
    Adv communicates x(1-i) to Env
```



Problems with Nondeterminism



- Order of messages may reveal one bit of s to E

Approaches to Nondeterminism

- UC framework
  - ITMs have a token passing mechanism
  - No nondeterminism
- Reactive simulatability
  - Again token passing mechanism (implicit)
  - Nondeterminism based on local information only
- Symbolic Dolev-Yao
  - No probability
  - Symbols hide information
- Process Algebras
  - Scheduler sees only enabled action type
- Switched PIOAs
  - Token passing mechanism (explicit)
  - Nondeterminism based on local information only
- Task PIOAs
  - Define equivalence classes of actions
  - Scheduler sees only equivalence classes, not elements
- Careful specifications
  - Avoid dangerous nondeteminism in the specification
  - Is it always possible?

Task PIOAs

- Probabilistic I/O Automata with …
  - Action determinism
    - For each action at most one transition enabled
  - Output and internal actions partitioned into tasks
  - Task determinism
    - For each task at most one transition enabled
- A scheduler is a sequence of tasks
  - Upon scheduling a task from a state q
    - Automaton performs unique transition enabled if it exists
    - Automaton idles if task not enabled
- Essentially scheduling does not depend on secret info

Task PIOAs What???

- Scheduler are oblivious
  - Not quite
  - We can encode the token passing mechanism
  - We could elect an automaton as adversary
- Do simulations continue to work?
  - We have to change the step condition
    - A task should be matched by a task
    - A simulation relates measures over executions
      - Need to know what tasks induced the measure
- Can we do better?
  - We do not know
  - But tasks work better than we expected
  - We can generalize them in many simple ways
  - Yet it would be nice to find something less “oblivious”

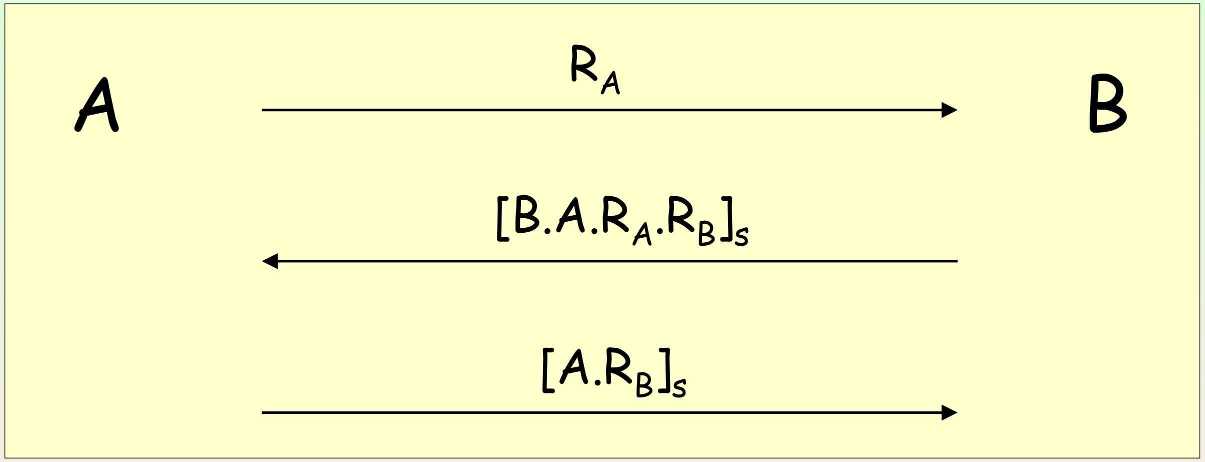
Case Study:

Agent Authentication

Bellare Rogaway 93

Segala, Turrini

Bellare and Rogaway MAP1 Protocol



- Nonces are generated randomly
- The key s is the secret for a Message Authentication Code
  - Specifically, MAC based on pseudo-random functions

Nonces

- Number ONCE
  - Typically drawn randomly

- Claim
  - For each constant c and polynomial p
  - There exists $\mathbf{k}$ such that for each $k \geq \mathbf{k}$
  - If $n_1,n_2,\ldots,n_{p(k)}$ are random nonces from $\{0,1\}^k$
  - Then $\boxed{\Pr[\exists_{i\neq j}\, n_i = n_j] < k^{-c}}$

Message Authentication Code

- Triple (G,A,V)
  - G on input 1^k generates $s \in \{0,1\}^k$
  - For each s and each a
    - Pr[V(s,a,A(s,a))=1]=1

- Forger
  - On input 1^k obtains MAC of strings of its choice
  - Outputs a pair (a,b)
  - Successful if V(s,a,b)=1 and a different from previous queries

- Secure MAC
  - Every feasible forger succeeds with negligible probability

MAP1: Matching Conversations

- Matching conversation between A and B
  - Every message from A to B delivered unchanged
    - Possibly last message lost
    - Response from B returned to A
  - Every message received by A generated by B
    - Messages generated by B delivered to A
    - Possibly last message lost

- Correctness condition
  - Matching conversation implies acceptance
  - Negligible probability of acceptance without matching conversation

MAP1: Correctness Proof

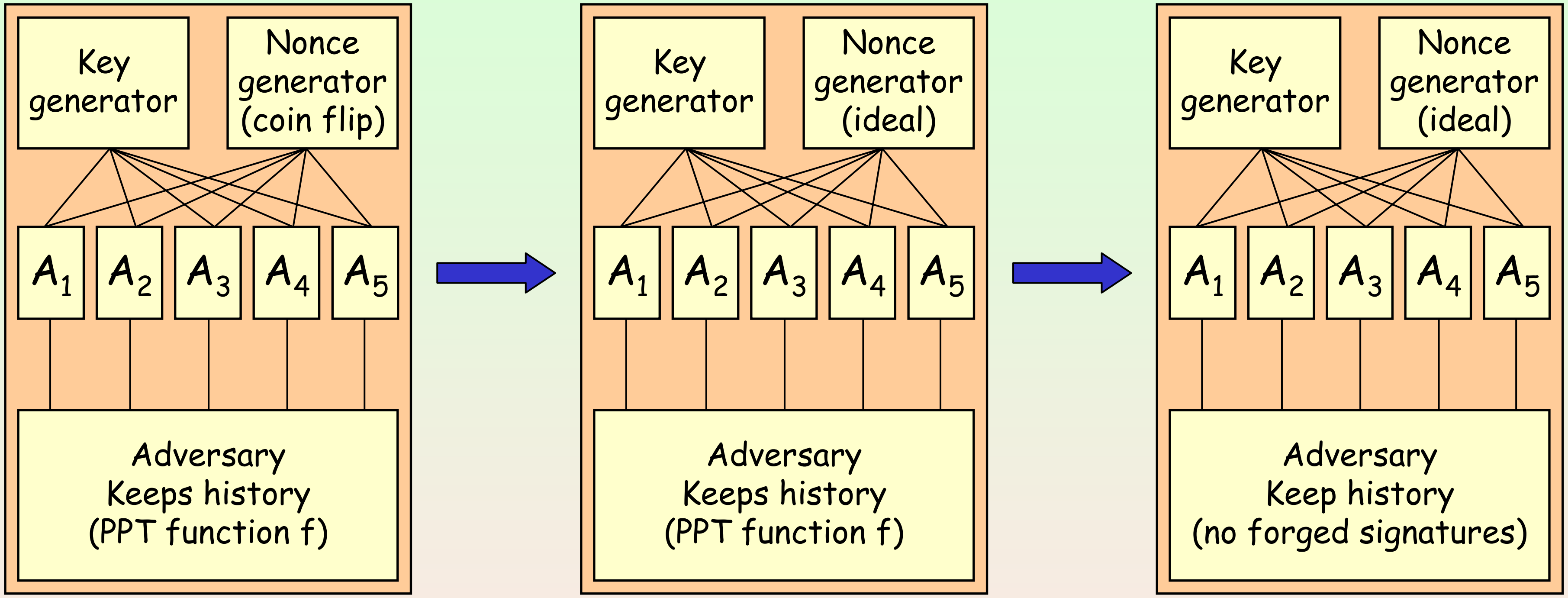
- Let A be a PPT machine that interacts with the agents

- Show that A induces "no-match" with negligible probability
  - Argue that repeated nonces occur with negligible probability
  - Argue that A is an attack against a message authentication code

- Features
  - Relies on underlying pseudo-random functions
  - Proves correctness assuming truly random functions
  - Builds a distinguisher for PRFs if an attack exists

- Criticism
  - The arguments are semi-formal and not immediate
  - Three different concepts intermixed
    - Nonces
    - Message authentication codes
    - Matching conversations

MAP1: Hierarchical Analysis



- Agents indexed by X, Y, t
- Need to find suitable simulations
  - Step conditions lead to local arguments
  - Yet transitions cannot be matched exactly

Nonce Generators

- State
  - $value_{X,Y,t}$ initially $\perp$
  - $FreshNonces$ initially $\{0,1\}^k$
- Transitions
  - Input $NonceRequest_{X,Y,t}$
  - Effect
    - Let $v \in_R \{0,1\}^k$
    - $value_{X,Y,t} = v$
    - $FreshNonces = FreshNonces-\{v\}$
  - Output $NonceResponse_{X,Y,t}(n)$
  - Precondition
    - $n = value_{X,Y,t}$
  - Effect
    - $value_{X,Y,t} = \perp$

Coin flip

Ideal

- Let $v \in_R FreshNonces$

Adversary

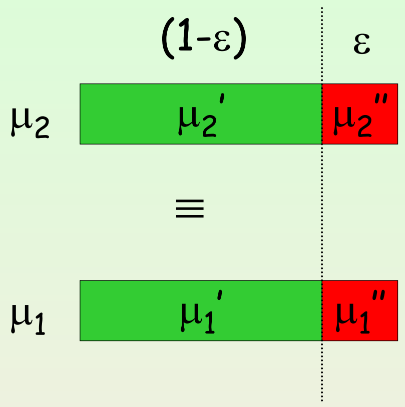
- Keeps a variable *history*
  - Holds all previous messages
- Real adversary
  - Runs a cycle where
    - Computes the next message to send using a PPT function f
    - Sends the message
    - Waits for the answer if expected
- Ideal adversary
  - Highly nondeterministic
  - Stores all input
  - Sends messages that do not contain forged authentications

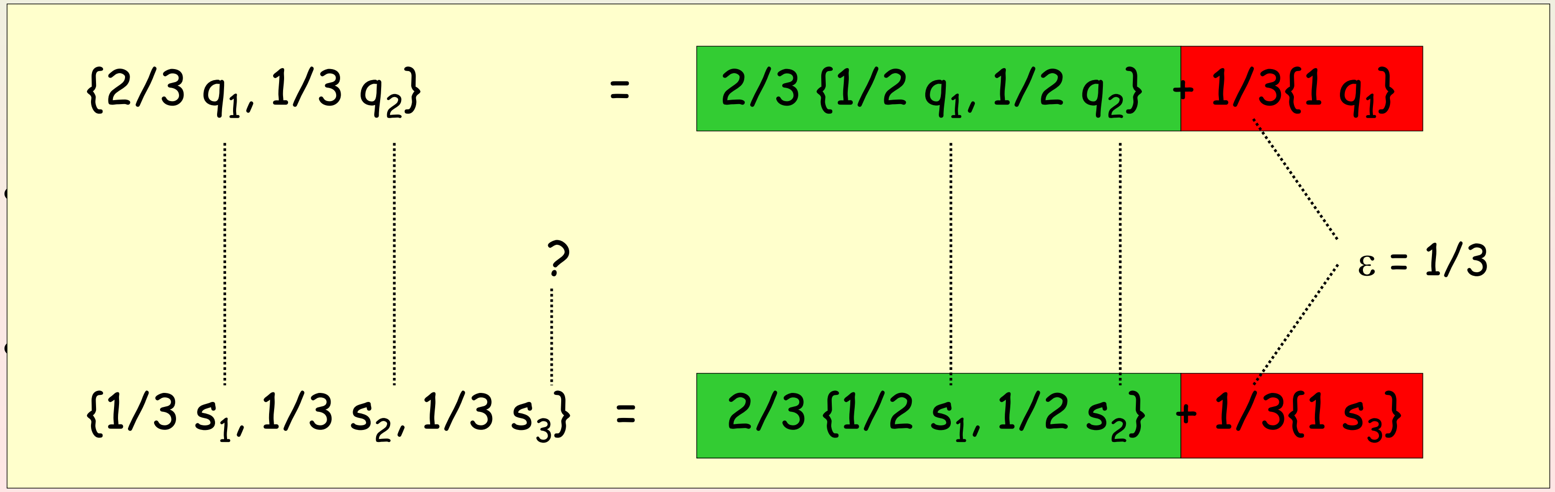
Problems with Simulations

- Problem
  - Consider a transition of the real nonce generator
  - With some probability there is a repeated nonce
  - The ideal nonce generator does not repeat nonces
  - Thus, we cannot match the step
- Solution
  - Match transitions up to some error

Approximate Simulations [ST07]

- Change equivalence on measures
  - $\mu_1 \equiv_\varepsilon \mu_2$ iff
    - $\mu_1 = (1-\varepsilon)\mu_1' + \varepsilon\mu_1''$
    - $\mu_2 = (1-\varepsilon)\mu_2' + \varepsilon\mu_2''$
    - $\mu_1' \equiv \mu_2'$



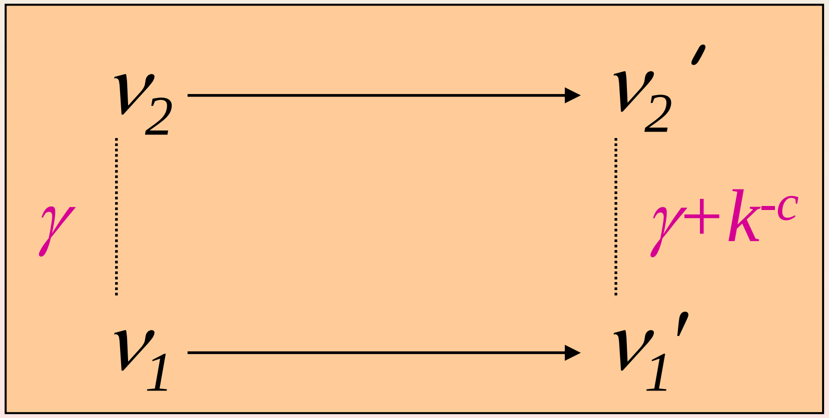


$\{2/3\ q_1, 1/3\ q_2\}$ = $2/3\ \{1/2\ q_1, 1/2\ q_2\} + 1/3\{1\ q_1\}$
?
$\varepsilon = 1/3$
$\{1/3\ s_1, 1/3\ s_2, 1/3\ s_3\}$ = $2/3\ \{1/2\ s_1, 1/2\ s_2\} + 1/3\{1\ s_3\}$

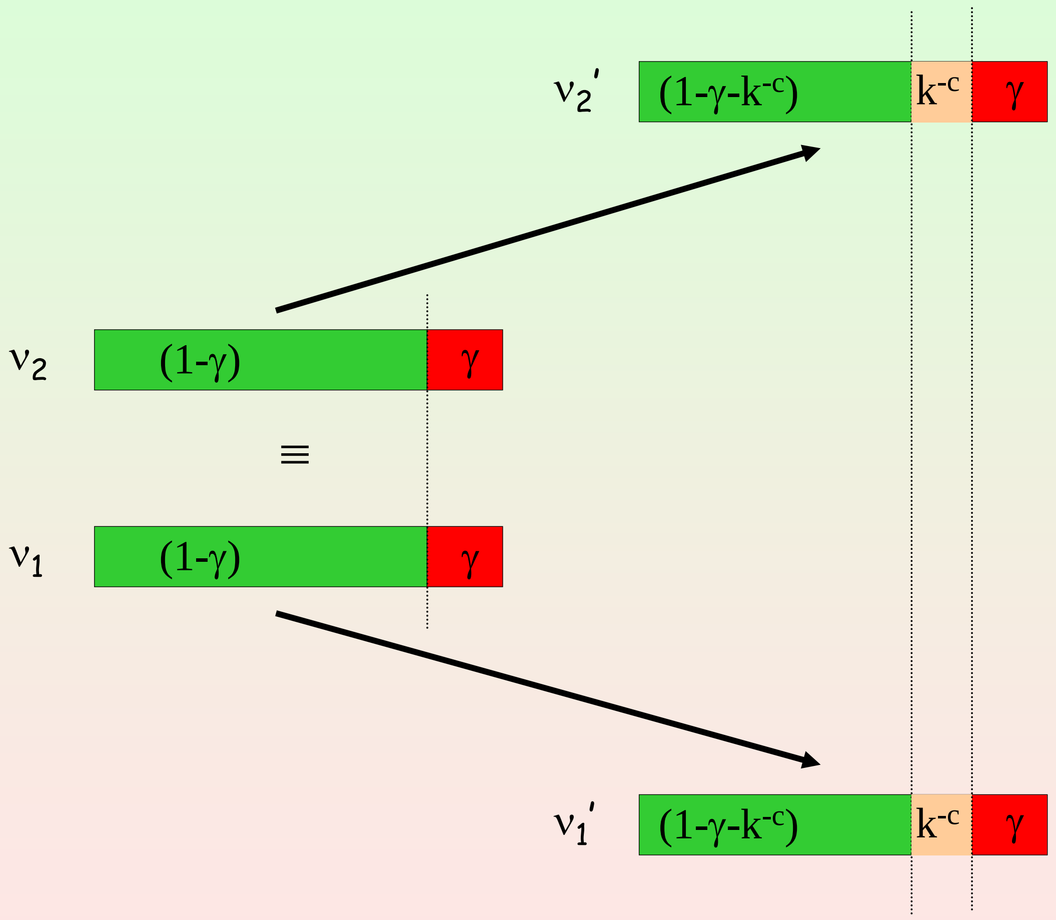
Approximate Simulations

$\{A_k\}\ \{R_k\}\ \{B_k\}$

- For each constant c and polynomial p
- There exists $\mathbf{k}$ such that for each $k \geq \mathbf{k}$
- Whenever
  - ν_1 reached within $p(k)$ steps in A_k
  - $\nu_1\ L(R_k,\gamma)\ \nu_2$
  - $\nu_1 \rightarrow \nu_1'$
- There exists ν_2' such that
  - $\nu_2 \rightarrow \nu_2'$
  - $\nu_1'\ L(R_k,\gamma+k^{-c})\ \nu_2'$


$$\begin{tikzcd}
\nu_2 \arrow[r] \arrow[d, dash, dotted, "\gamma"'] & \nu_2' \arrow[d, dash, dotted, "\gamma+k^{-c}"] \\
\nu_1 \arrow[r] & \nu_1'
\end{tikzcd}$$

Approximate Simulations
Step Condition



Simulation Implies Behavioral Inclusion

- The step condition can be applied repeatedly

$$\begin{tikzcd}
s \arrow[r, Rightarrow] \arrow[d, dotted, no head, "0"] & \rho_1 \arrow[r, Rightarrow] \arrow[d, dotted, no head, "k^{-c}"] & \rho_2 \arrow[r, Rightarrow] \arrow[d, dotted, no head, "2k^{-c}"] & \rho_3 \arrow[r, Rightarrow] \arrow[d, dotted, no head, "3k^{-c}"] & \dots \arrow[r, Rightarrow] \arrow[d, dotted, no head] & \rho_{p(k)} \arrow[d, dotted, no head, "p(k)k^{-c}"] \\
q \arrow[r] & \mu_1 \arrow[r] & \mu_2 \arrow[r] & \mu_3 \arrow[r] & \dots \arrow[r] & \mu_{p(k)}
\end{tikzcd}$$

- Observation
  - $p(k)k^{-c}$ can be smaller than any $k^{-c'}$ by choosing $c=c'+degree(p)$

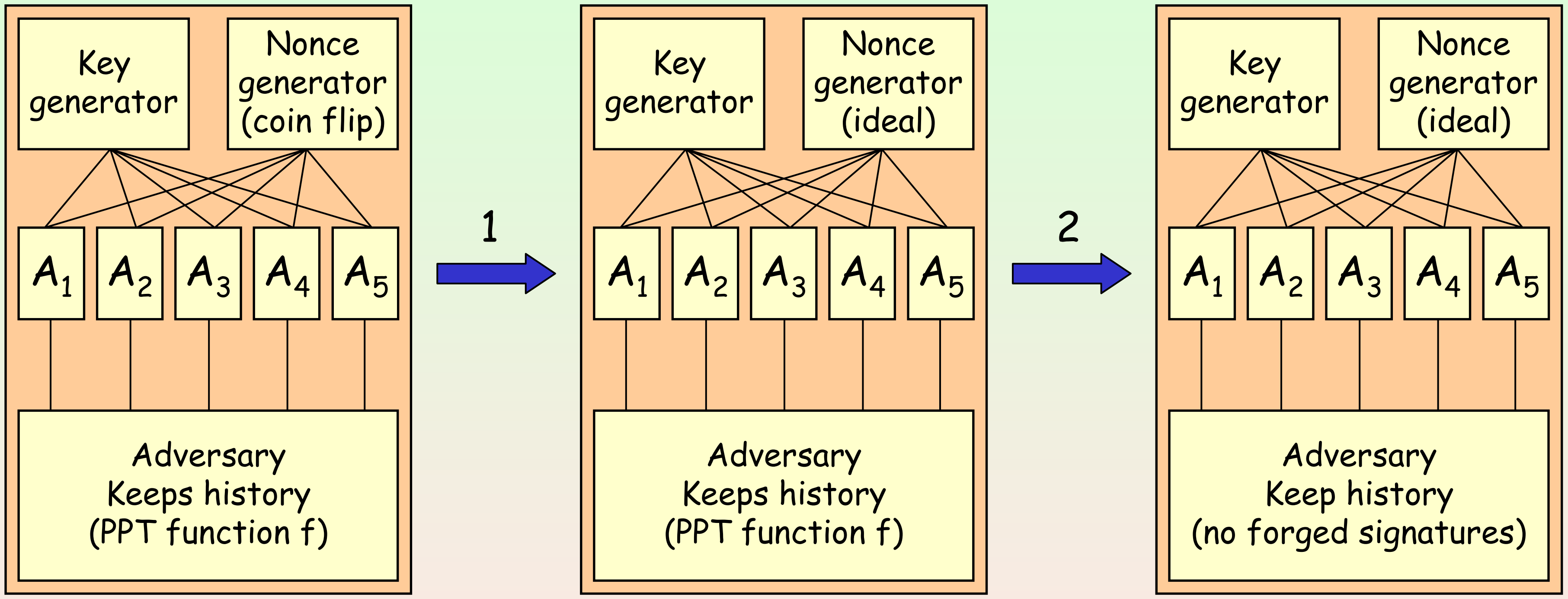
Execution Correspondence under Approximated Simulations

If $\{A_k\}$ $\{R_k\}$ $\{B_k\}$ then

- For each constant c and polynomial p
- There exists $\mathbf{k}$ such that for each $k \geq \mathbf{k}$
- For each scheduler σ_1
  - ν_1 reached within $p(k)$ steps in A_k with σ_1
- There exists σ_2 such that
  - ν_2 reached within $p(k)$ steps in B_k with σ_2
  - ν_1 $L(R_k, p(k)k^{-c})$ ν_2

- Observation
  - $p(k)k^{-c}$ can be smaller than any $k^{-c'}$ by choosing $c=c'+degree(p)$

Example: Approximate Simulations Bellare-Rogaway MAP1 Protocol

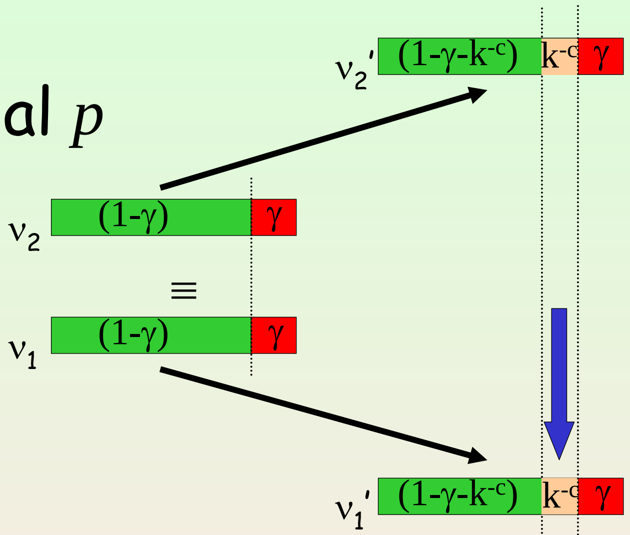


- Negation of the step condition
  - 1: Two random nonces are equal with high probability
  - 2: Function f defines a forger for a signature scheme

Negation of Step Condition

$\{A_k\}\ \{R_k\}\ \{B_k\}$

- There exists constant c and polynomial p
- For each $\mathbf{k}$ there exists $k \geq \mathbf{k}$
- There exists
  - ν_1 reached within $p(k)$ steps in A_k
  - $\nu_1\ L(R_k,\gamma)\ \nu_2$
  - $\nu_1 \rightarrow \nu_1'$
- There is no ν_2' such that
  - $\nu_2 \rightarrow \nu_2'$
  - $\nu_1'\ L(R_k,\gamma+k^{-c})\ \nu_2'$



$$\begin{tikzcd}
\nu_2 \arrow[r, "/" marking] & \nu_2' \\
\nu_1 \arrow[u, dotted, dash, "\gamma"] \arrow[r] & \nu_1' \arrow[u, dotted, dash, "\gamma+k^{-c}"']
\end{tikzcd}$$

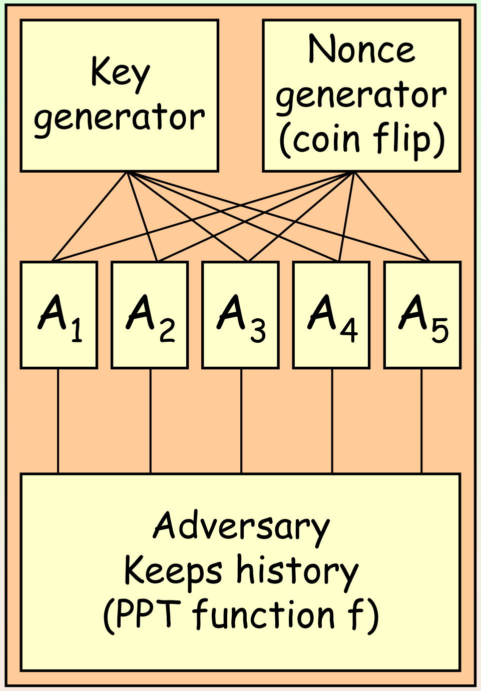
- **Signature forged in ν_1'**
- **Nonce replicated in ν_1'**
  - Probability at least k^{-c}

Nonces

- Number ONCE
  - Typically drawn randomly

- Claim
  - For each constant c and polynomial p
  - There exists $\mathbf{k}$ such that for each $k \geq \mathbf{k}$
  - If $n_1, n_2, \ldots, n_{p(k)}$ are random nonces from $\{0,1\}^k$
  - Then $\boxed{\Pr[\exists_{i \neq j}\, n_i = n_j] < k^{-c}}$

Problems with Nondeterminism MAP1 Protocol [BR93]



- Authentication protocol
  - Symmetric key signature schema
  - Computational Dolev-Yao
  - Adversary queries agents
- Potential problems
  - Let s be the shared key
  - Adversary queries k agents
  - Agent i replies if i^{th} bit of s is 1
  - The adversary knows the shared key
- Solution
  - One query at a time
  - Wait for the answer (agents as oracles)

More About Approximated Simulations

Conditional Automata

- Let A be a probabilistic automaton
- Let B be a set of bad states
- Let $G = Q-B$ be a set of good states

- Let $A|G$ be the same as A except that
  - $D_{A|G} = \{(q,a,\mu|G) : (q,a,\mu)\ D_A \text{ and } \mu(G)>0\}$

Theorem

id_Q is a polynomially accurate simulation from A to $A|G$ iff B is negligible

id_Q is a polynomially accurate simulation from $A|G$ to A iff B is negligible

A Property of Approximated Lifting

Given a relation R from Q_1 to Q_2

Then μ_1 $L(R,\varepsilon)$ μ_2 iff there exists

w: Q_1 ´ $Q_2 \to [0,1]$

- w supported on R
- $w(Q_1,Q_2) = 1-\varepsilon$
- $w(s,Q_2) \leq \mu_1(a)$
- $w(Q_1,s) \leq \mu_2(a)$

Approximated Correspondence

$$\begin{tikzcd}
s \arrow[r] \arrow[d, dash, dotted, "0"] & \rho_1 \arrow[r] \arrow[d, dash, dotted, "k^{-c}"] & \rho_2 \arrow[r] \arrow[d, dash, dotted, "2k^{-c}"] & \rho_3 \arrow[r] \arrow[d, dash, dotted, "3k^{-c}"] & \cdots \arrow[r] \arrow[d, dash, dotted] & \rho_{p(k)} \arrow[d, dash, dotted, "p(k)k^{-c}"] \\
q \arrow[r] & \mu_1 \arrow[r] & \mu_2 \arrow[r] & \mu_3 \arrow[r] & \cdots \arrow[r] & \mu_{p(k)}
\end{tikzcd}$$

This means that …

$$\begin{tikzcd}
s \arrow[r, "a"] \arrow[d, dash, dotted, "R"] & s_1 \arrow[r, "b"] \arrow[d, dash, dotted, "R"] & s_2 \arrow[r, "c"] \arrow[d, dash, dotted, "R"] & s_3 \arrow[r] \arrow[d, dash, dotted, "R"] & \cdots \arrow[r] \arrow[d, dash, dotted, "R"] & s_{p(k)} \arrow[d, dash, dotted, "R"] \arrow[dr, dash, dotted] & \\
q \arrow[r, "a"] & q_1 \arrow[r, "b"] & q_2 \arrow[r, "c"] & q_3 \arrow[r] & \cdots \arrow[r] & q_{p(k)} \arrow[r, dash, dotted] & w(.,.)
\end{tikzcd}$$

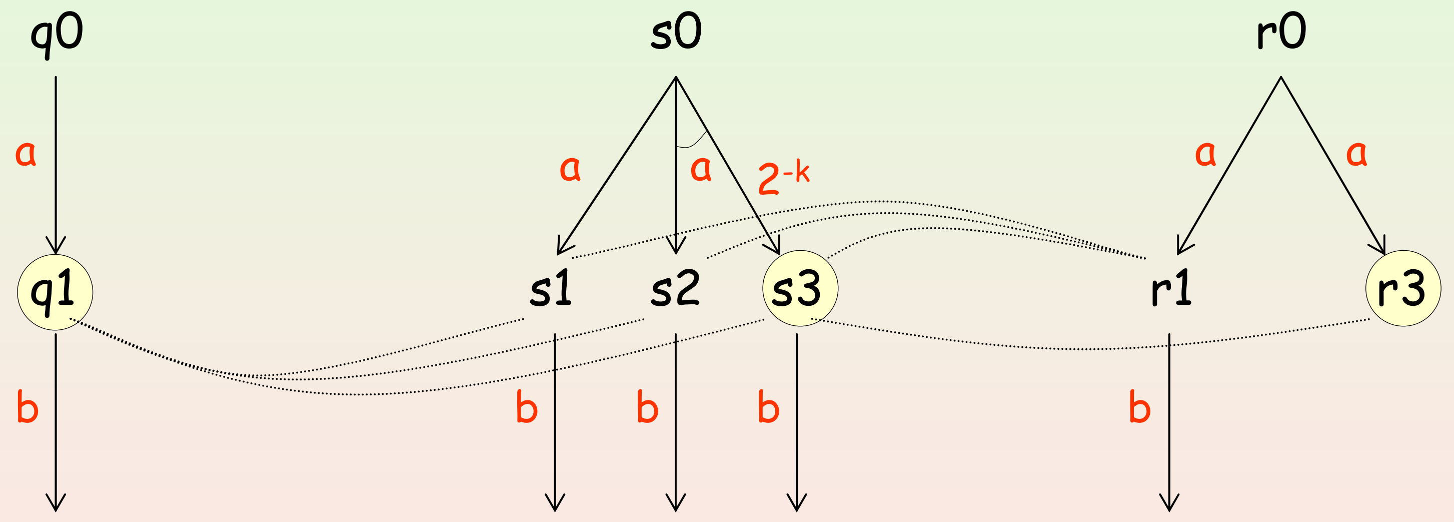
Transitivity

Claim. $\mu\ L(R,\varepsilon)\ \rho$ and $\rho\ L(R',\tau)\ \eta$ imply $\mu\ L(RR',\varepsilon+\tau)\ \eta$

$$\begin{tikzcd}
t \arrow[r] \arrow[d, dotted, dash, "0"] \arrow[dd, dotted, dash, bend left, "0"] & \eta_1 \arrow[r] \arrow[d, dotted, dash, "k^{-c'}"] \arrow[dd, dotted, dash, bend left, "k^{-c}+k^{-c'}"] & \eta_2 \arrow[r] \arrow[d, dotted, dash, "2k^{-c'}"] \arrow[dd, dotted, dash, bend left, "2(k^{-c}+k^{-c'})"] & \eta_3 \arrow[r] \arrow[d, dotted, dash, "3k^{-c'}"] \arrow[dd, dotted, dash, bend left, "3(k^{-c}+k^{-c'})"] & \ldots \arrow[r] \arrow[d, dotted, dash] & \eta_{p(k)} \arrow[d, dotted, dash, "p(k)k^{-c'}"] \arrow[dd, dotted, dash, bend right, "p(k)(k^{-c}+k^{-c'})"'] \\
s \arrow[r] \arrow[d, dotted, dash, "0"] & \rho_1 \arrow[r] \arrow[d, dotted, dash, "k^{-c}"] & \rho_2 \arrow[r] \arrow[d, dotted, dash, "2k^{-c}"] & \rho_3 \arrow[r] \arrow[d, dotted, dash, "3k^{-c}"] & \ldots \arrow[r] \arrow[d, dotted, dash] & \rho_{p(k)} \arrow[d, dotted, dash, "p(k)k^{-c}"] \\
q \arrow[r] & \mu_1 \arrow[r] & \mu_2 \arrow[r] & \mu_3 \arrow[r] & \ldots \arrow[r] & \mu_{p(k)}
\end{tikzcd}$$

Are approximated simulations transitive?

- We do not know
  - … but the result of the previous slide suffices



Are Approximated Simulations Compositional?

No. Need a more refined relation.

$s\ S(R,\varepsilon)\ q$ iff

$\forall\ q, s, a, \mu'\ \exists\ \sigma'$

$$\begin{tikzcd} s \arrow[r, "a"] & \sigma' \arrow[d, dash, dotted, "R"', "\varepsilon"] \\ q \arrow[r, "a"] & \mu' \end{tikzcd}$$

Step condition

For each c there exists **k**
For each k > **k**, each μ_1, μ_2, γ, w

If $\mu_1\ L(R_k,\gamma)\ \mu_2$ via w
then
$\Sigma\ \{w(q_1,q_2) : q_1\ \mathrm{not}(S(R_k,k^{-c}))\ q_2\} < k^{-c}$

Conditional automata continue to work

How About Weak Relations?

- Only one constraint to add
  - Length of matching steps bounded
    - By a constant
    - By a polynomial on length of history

Case Study:

Dolev-Yao Soundness

Cortier Warinschi 04

Segala, Turrini

Protocol Syntax

- Sorts
  - SKey, VKey, EKey, DKey
  - Id, Nonce, Label, Cipertext, Signature, Pair
  - Term: supersort that includes all others
    - Labels should be left out
- Operators
  - $\langle \_,\_ \rangle$: Term $\times$ Term $\rightarrow$ Pair
  - $\{\_\}\_,\_$: EKey $\times$ Term $\times$ Label $\rightarrow$ Cipertext
  - $[\_]\_,\_$: SKey $\times$ Term $\times$ Label $\rightarrow$ Signature
- Variables
  - Sorted variables
  - $X = X.n \cup X.a \cup X.c \cup X.s \cup X.l$
  - $X.a = \{A_1, A_2, \ldots, A_n\}$, n number of protocol participants
  - $X.n = \cup_{A \in X.a} \{X_{A,j} \mid j \in N\}$

Protocol Syntax

- Roles
  - Finite sequence of rules
  - $((\{init\} \times T_\Sigma(X)) \times (T_\Sigma(X) \times \{stop\})^*$
- k-party protocol
  - $\Pi : \{1, \ldots, k\} \rightarrow$ Roles
  - $\Pi(i)$ is the program of player i
- Idea
  - An adversary instantiates protocols and queries parties
  - If role i is ready to execute the pair (l,r)
    and role i is given input m
  - m is parsed according to l
    - Pattern matching, unification
    - Some variables may be bound to new values
  - r is returned as a result

Example: Needham-Schroeder-Lowe

$$A \to B : \{Na,A\}_{ek(B)}$$
$$B \to A : \{Na,Nb,B\}_{ek(A)}$$
$$A \to B : \{Nb\}_{ek(B)}$$

$$\Pi(1) = (init, \qquad \{X_{A1,1},A_1\}_{ek(A2),ag(1)})$$
$$(\{X_{A1,1},X_{A2,1},A_2\}_{ek(A1),L}, \ \{X_{A2,1}\}_{ek(A2),ag(1)})$$

$$\Pi(2) = (\{X_{A1,1},A_1\}_{ek(A2),L1}, \qquad \{X_{A1,1},X_{A2,1},A_2\}_{ek(A1),ag(1)})$$
$$(\{X_{A2,1}\}_{ek(A2),L2}, \qquad stop)$$

Formal Execution Model

- Messages are ground terms from an algebra
  - $T ::= N \mid a \mid ek(a) \mid dk(a) \mid sk(a) \mid vk(a) \mid n(a,j,s) \mid \langle T,T \rangle \mid \{T\}_{ek(a),ag(i)} \mid \{T\}_{ek(a),adv(i)} \mid [T]_{sk(a),ag(i)} \mid [T]_{sk(a),adv(i)}$
- Global state: (SId, f, H)
  - SId: set of session Ids of the form $(n,j,(a_1,\ldots,a_k))$
  - f: associates state (σ,i,p) to each session id
    - Partial function σ associates terms to variables
    - i is the role being executed
    - p is the program counter (next pair to match)
  - H is a set of terms (knowledge of adversary)

Formal Execution Model

- Initially no session ids , H contains nonces of adversary
- Transitions
  - corrupt($a_1,\dots,a_l$)
    - H updated with knowledge of $a_1,\dots,a_l$
  - new(i,($a_1,\dots,a_k$))
    - New session id S created with index s
    - $f(S) = (\sigma, i, 1)$
    - Function σ binds agent variable A_j to a_j
    - Function σ binds nonce variable $X_{Ai,j}$ to $n(a_i,j,s)$
  - send(S,t)
    - Let $f(S)$ be (σ,i,p) and let (l,r) be the pth pair of $\Pi(i)$
    - Match t with l updating σ. Stop if unsuccessful.
    - Compute r and add it to H
    - Update $f(S)$ to $(\sigma,i,p+1)$

Restriction:
t must be DY-deducible from H

Concrete Execution Model

- Agent id's, nonces, messages are bitstrings
- Security parameter ν identifies lengths

- Global state: (SId,g,H)
  - H is the knowledge of the adversary
  - SId: set of session Ids of the form $(n,j,(\eta_1,\ldots,\eta_k))$
  - g: associates state (τ,i,p) to each session id
    - Partial function τ associates bitstrings to variables
    - i is the role being executed
    - p is the program counter (next pair to match)

Concrete Execution Model

- Initially no session ids
- Transitions
  - corrupt($\eta_1,\dots,\eta_l$)
    - H updated with knowledge of $\eta_1,\dots,\eta_l$
    - The necessary missing keys are generated
  - new(i,($\eta_1,\dots,\eta_k$))
    - New session id S created with index s
    - g(S) = (τ,i,1)
    - Function τ binds agent variable A_j to η_j
    - Function τ binds nonce variable $X_{Ai,j}$ to random bitstrings
    - Random coins are flipped for the randomization of encryption and signature
  - send(S,t)
    - Let g(S) be (τ,i,p) and let (l,r) be the p^{th} pair of Π(i)
    - Match t with l updating τ. Stop if unsuccessful.
      - May need to decrypt and verify signatures
    - Compute r and add it to H
      - May need to encrypt and sign
    - Update g(S) to (σ,i,p+1)

Computations of Concrete Model

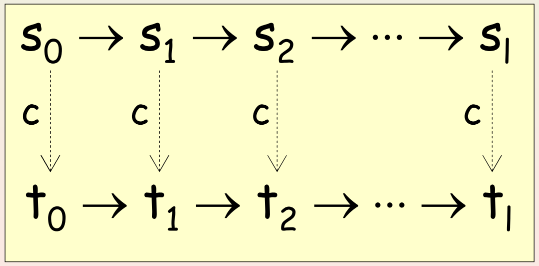
- In the model of [CW04]
  - Choice of transitions by PPT adversary
  - Length of computations bounded by a polynomial
  - Number of needed random bits known in advance
  - Unique computation for each value of the random bits
  - This induces a probability measure on computations
- With Probabilistic Automata
  - Random bits generated within transitions
  - Avoid reasoning about guessing future random bits
    - … though in [CW04] this reasoning is not present

Correspondence Between Computations

- Let c be a mapping from ground terms to bitstrings
- Let s = (SId,f,H) be a state of the formal model
- Let t = (CId,g,H') be a state of the concrete model

- Define $s \equiv_c t$ iff
  - $CId = \{c(S) \mid S \in SId\}$
  - $\forall_{S \in SId}\ g(c(S)) = c(f(S))$
- Where
  - $c(n,i,(a_1,\ldots,a_k)) = (n,i,(c(a_1),\ldots,c(a_k))$
  - $c(s,i,p) = (c(s),i,p)$

- Define $s_0 s_1 \ldots s_l \equiv t_0 t_1 \ldots t_l$ iff
  - $\exists_{c \text{ injective}} \forall_j\ s_j \equiv_c t_j$



$$\begin{tikzcd} s_0 \arrow[r] \arrow[d, dashed, "c"'] & s_1 \arrow[r] \arrow[d, dashed, "c"'] & s_2 \arrow[r] \arrow[d, dashed, "c"'] & \cdots \arrow[r] & s_l \arrow[d, dashed, "c"'] \\ t_0 \arrow[r] & t_1 \arrow[r] & t_2 \arrow[r] & \cdots \arrow[r] & t_l \end{tikzcd}$$

- Concrete model safe iff
  - For each measure μ on concrete executions induced by random coins
  - $\mu(\{a \mid \exists_b\ a \equiv b\})$ is overwhelming

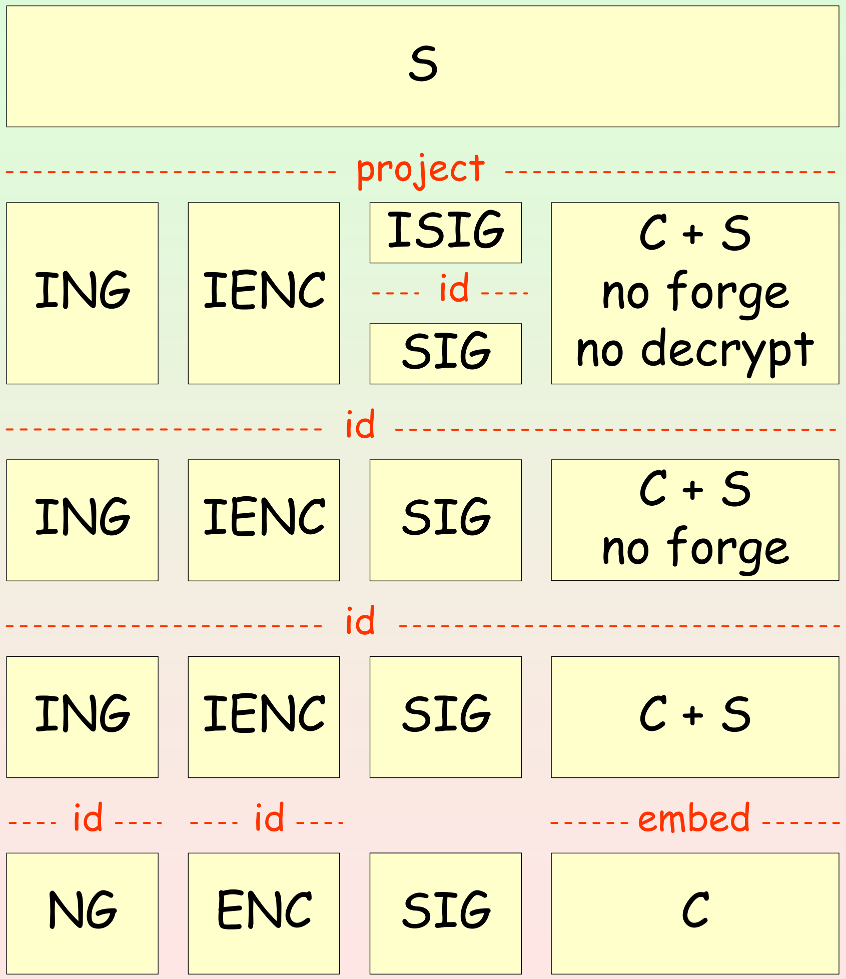
Structure of Original Proof

- Prove properties of DY-non-deducibility
  1. Signature forged, or
  2. Encrypted data used without decrypting
- Fix random coins and get concrete execution α
- Show α is instantiation of some symbolic execution β
  - Follow α building β and mapping bitstrings to abstract terms
    - How do I know the mapping exists?
      - Example: reencrypt a message with a different label and encryptions are the same
  - Let c be the inverse of the mapping above
    - How do I know the mapping is invertible?
      - Example: forward an encrypted message
    - How do I know c is injective?
      - The inverse of a mapping is injective
- Show β follows DY-deducibility with overwhelming probability
  - If not, then either 1 or 2 with non-negligible probability
  - Build attacker to corresponding primitive

Properties of non-DY-Deducibility

- Let S be a set of messages and m a message such that
  - $S \nvdash m$
  - m built out of atoms of elements in S
- Then either
  - There exists subterm $[t]_k$ of m which is not a subterm of terms in S, or
  - There exists a subterm t of m such that
    - all its super-terms in m are not deducible
    - t appears encrypted in S
- Problem
  - A message that contains atoms not in S is not deducible
  - Scenario not included in the cases above

Structure of the Proof with Probabilistic Automata



Now c is injective

ING | SIG | CS are a PPT
Environment for ENC

ING | ENC | CS are a PPT
Environment for SIG

Here we have also function c,
though not injective

Actions chosen by PPT function f
Primitives solved by NG, ENC, SIG

Problems Encountered Concrete Model

- Explicit encoding of
  - Parsing of left expression
  - Computation of right expression
  - Invocations to cryptographic primitives

- What arguments are needed for and computed by …
  - Left parsing
  - Right computation

- Answer
  - The mapping τ

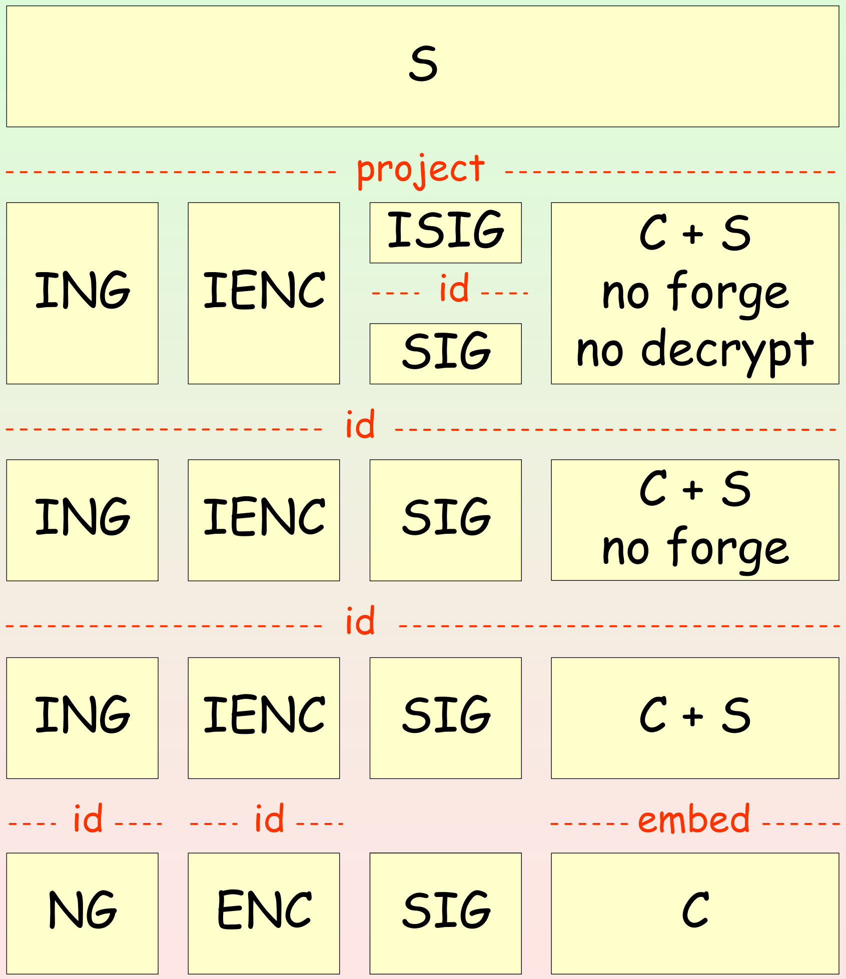
Concrete Model: Some examples

- $(\mathrm{init}, X_{A1,1})\ (\{X_{A2,1}\}_{ek(a1),L}\ ,\ \{X_{A2,1}\}_{ek(a1),ag(1)})\ (X_{A2,2}, \mathrm{stop})$
  - After initialization $\tau(X_{A1,1}) = \eta_1$
  - Upon receiving a bitstring η_2
    - It is decrypted with $dk(a_1)$ and $\tau(X_{A2,1})=\eta_3$
    - What should L be mapped to?
    - Then η_3 is encrypted with $ek(a_1)$ leading to η_4
  - Upon receiving η_5, $\tau(X_{A2,2})=\eta_5$ and terminate

$$\downarrow \qquad\qquad \downarrow$$

- $(\mathrm{init}, X_{A1,1})\ (\{X_{A2,1}\}_{ek(a),L}, \{X_{A2,1}\}_{ek(a),L})\ (X_{A2,2}, \mathrm{stop})$
  - After initialization $\tau(X_{A1,1}) = \eta_1$
  - Upon receiving a bitstring $\boxed{\eta_2}$
    - It is decrypted with $dk(a_1)$ and $\tau(X_{A2,1})=\eta_3$
    - Then η_3 is encrypted with $ek(a_1)$ leading to $\boxed{\eta_4}$
  - Upon receiving η_5, $\tau(X_{A2,2})=\eta_5$ and terminate

Structure of the Proof with Probabilistic Automata



Now c is injective

ING | SIG | CS are a PPT
Environment for ENC

ING | ENC | CS are a PPT
Environment for SIG

Here we have also function c,
though not injective

Actions chosen by PPT function f
Primitives solved by NG, ENC, SIG

Problems Encountered
Definition of C + S

- If the bitstring I receive does not parse what symbolic message should I use?
  - Not said/considered in the original proof

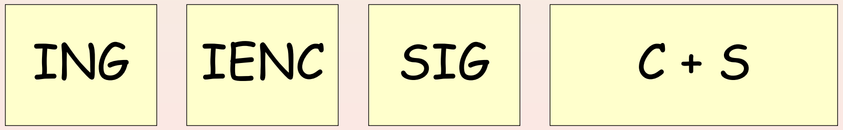
- The bitstring should be kept, though
  - A real system could reuse it later

- Our solution
  - Use a special symbol $\perp$
  - Its meaning is that we are sending junk
  - Function c does not map $\perp$

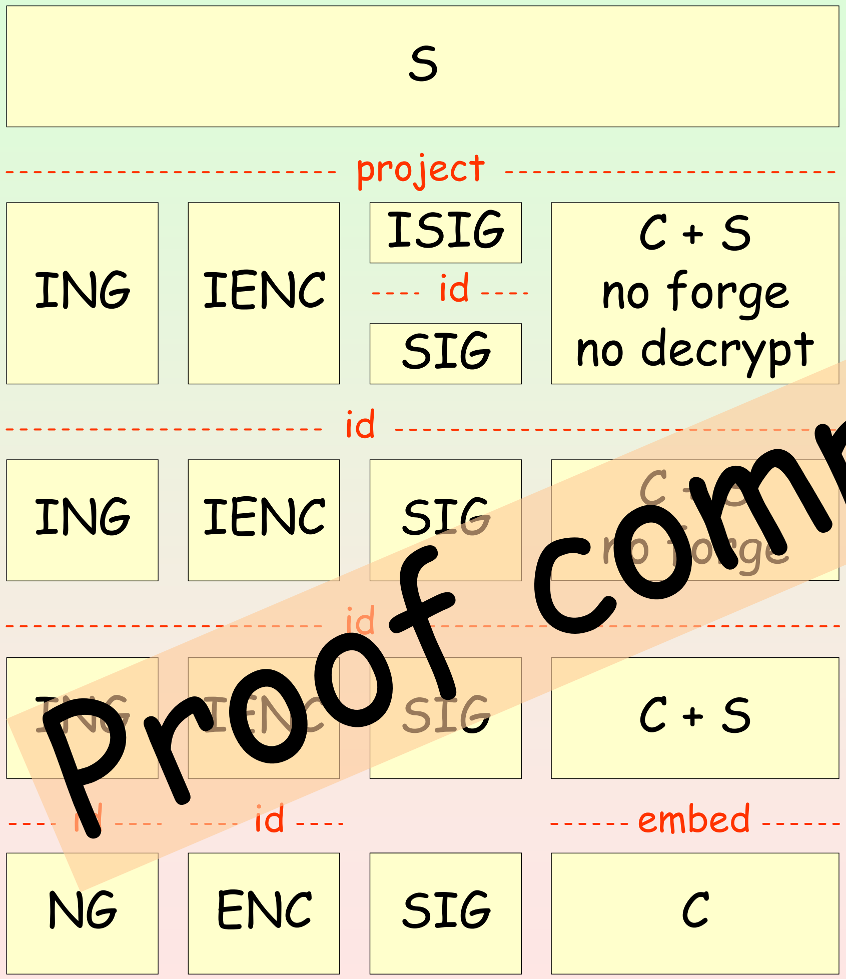
Consequences of our Solution

- All the symbols we use in send actions are build from atomic terms that appear in the history

- The new statement about non-deducibility suffices
  - Do not need to worry about guessing the future



Structure of the Proof with Probabilistic Automata



Now c is injective

ING | SIG | CS are a PPT
Environment for ENC

ING | ENC | CS are a PPT
Environment for SIG

Here we have also function c,
though not injective

Actions chosen by PPT function f
Primitives solved by NG, ENC, SIG

Summing Up …

- What we have seen
  - A theory of Probabilistic Automata
    - Conservative extension of automata
    - Language inclusion
    - Simulation relations
    - Hyerarchical compositional reasoning
  - A notion of task PIOA with restricted schedulers
    - Task equivalence relation on states
    - Action deterministic
    - At most one action for each task
    - A schedule (sequence of tasks) determines a probabilistic execution
  - A notion of approximated language inclusion
    - For each trace distribution of A there exists an indistinguishable trace distribution of B
  - A notion of approximated simulation
    - Works for PAs

Summing Up …

… what we have seen

- Analysis of oblivious transfer in UC framework
  - Task PIOAs as model
  - Hierarchical verification via simulations
  - Crypto-steps via approximated language inclusion
- Analysis of MAP1 protocol
  - PAs as model
  - Approximated simulations as technique
  - Mixture of Dolev-Yao and computational models
  - No restriction of nondeterminism
    - Yet accurate description of objects
- Analysys of DY-soundness
  - PAs as model
  - Approximated simulations, hierarchical compositional analysis
  - Easy to find problems … more difficult to fix them

Several Open Questions

- Connections
  - Approximated simulations with
    - Approximated language inclusion
    - Restricted schedulers
  - Semantics
    - Metrics and exact equivalences
- Properties of definitions
  - Are we transitive?
  - Are there weaker compositional refinements?
- Flexibility on restrictions
  - Task PIOAs are very restrictive
    - … though they work
    - Chatzikokolakis and Palamidessi may help (Concur07)
- Understanding of restrictions
  - Are we restricting too much?
- More case studies
  - Need to understand common points
  - Need to discover missing pieces

A Note about Formal Analysis

- Formal methods are too heavy to use
  - Is it reasonable to apply them all the times?
  - Is it reasonable to use them all the times?
  - Is it reasonable to know them?
  - Are automatic tools everything we need?

- Rarely we can be absolutely rigorous
  - We rather limit the places where to use intuition
  - Formal methods give a lot of sanity checks
  - It is useful to be aware of the formal meaning of what we say
  - It is useful to have theoretical results
    - Some doubts can be eliminated quickly
    - Some bugs may be discovered in a few seconds

Thank You

Convex Combination of Measures

- Let μ_1 and μ_2 be probability measures
- Let p_1 and p_2 be reals in [0,1] such that $p_1+p_2=1$
- Define a new measure $\mu = p_1\mu_1+p_2\mu_2$ as follows
  - $\forall X, \mu(X) = p_1\mu_1(X)+p_2\mu_2(X)$

- Theorem: μ is a proability measure

- Same result extends to countable summation

Weak Transition

$$q \xRightarrow{\quad\quad\quad a \quad\quad\quad} \rho$$

There is a probabilistic execution μ such that

- $\mu(exec^*) = 1$ (it is finite)
- $trace(\mu) = \delta(a)$ (its trace is a)
- $fstate(\mu) = \delta(q)$ (it starts from q)
- $lstate(\mu) = \rho$ (it leads to ρ)

$$q \overset{a}{\Rightarrow} s \quad \text{iff} \qquad \exists\alpha: trace(\alpha)=a,\ fstate(\alpha)=q,\ lstate(\alpha)=s$$