Task-PIOA を用いた
忘却転送の UC 安全性

参考文献

- Ran Canetti, Ling Cheung, Dilsun Kirli Kaynar, Moses Liskov, Nancy A. Lynch, Olivier Pereira, and Roberto Segala. Using Task-Structured Probabilistic I/O Automata to Analyze an Oblivious Transfer Protocol. Technical report, MIT CSAIL-TR-2007-011, 2007
- 米山一樹, 太田和夫: タスク構造確率I/Oオートマトンを用いた安全性証明

Task-PIOA

- Task-Structured Probabilistic I/O Automata
- 入出力を持つ確率状態遷移系
- 並行合成が可能。
  - T1||T2
- タスクとは、出力アクションおよび内部アクションの同値類。
  - 各状態において、個々のタスクの中では、高々一つのアクションのみが実行可能。

忘却転送

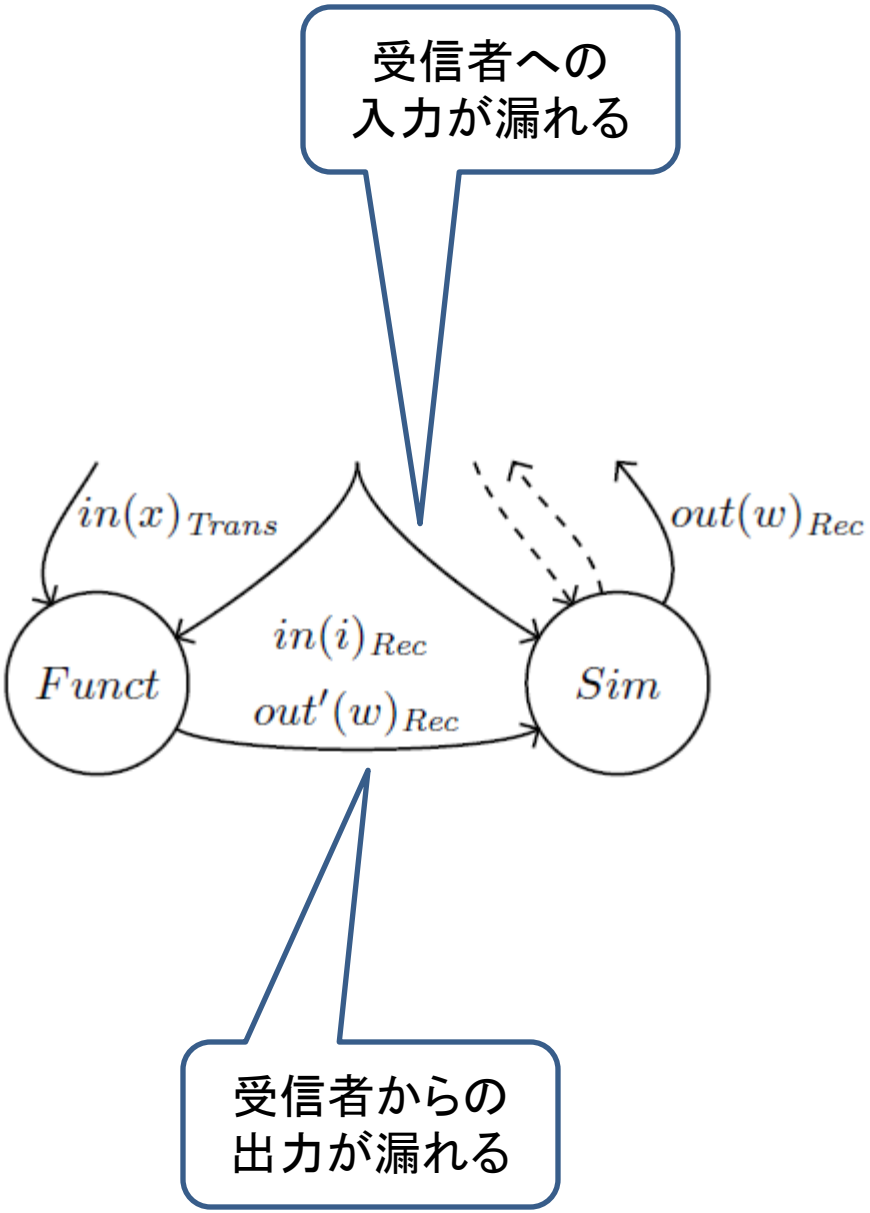
- 送信者は二つのビット x(0) と x(1) を持つ。
- 受信者はビット i を持つ。
- 受信者は x(i) だけを受け取る。
- 受信者は x(1-i) の値を知らない。
- 送信者は i の値を知らない。

攻撃者

- メッセージの盗聴は行うが、改竄は行わないとする。
- 送信者と受信者を支配（corrupt）するかもしれない。
  - ただし、動的にはcorruptしない。
- 以下では、i および x(i) の値を知ることができるが、x(1-i) の値を知ることができない、という状況を考える。
  - 受信者を支配している。

理想機能Func

- $in(x)_{Trans}$
  - $x \in \{0,1\} \rightarrow \{0,1\}$
  - 結果: もし xval=⊥ ならば xval := x
- $in(i)_{Rec}$
  - $i \in \{0,1\}$
  - 結果:もし ival=⊥ ならば ival := i
- $out'(w)_{Rec}$
  - 事前条件: w=xval(ival)
  - 結果: なし


受信者への
入力が漏れる
$in(x)_{Trans}$
$out(w)_{Rec}$
$in(i)_{Rec}$
$out'(w)_{Rec}$
$Funct$
Sim
受信者からの
出力が漏れる

送信者Trans(D,Tdpp)

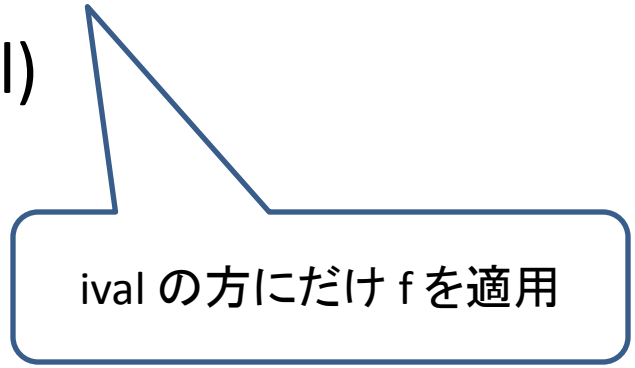
- $in(x)_{Trans}$
  - $x \in \{0,1\} \rightarrow \{0,1\}$
  - 結果: もし xval=⊥ ならば xval := x
- $rand(p)_{pval}$
  - $p \in$ Tdpp: 置換とその逆置換の組
  - 結果:もし pval=⊥ ならば pval := p
- $send(1,f)_{Trans}$
  - 事前条件: pval≠⊥, f=pval.funct (置換の方)
  - 結果: なし

受信者Rec(D,Tdp)

- $in(i)_{Rec}$
  - $i \in \{0,1\}$
  - 結果: もし ival=⊥ ならば ival := i
- $rand(y)_{yval}$
  - $y \in \{0,1\} \rightarrow D$: Dは置換の領域
  - 結果: もし yval=⊥ ならば yval := y
- $receive(1,f)_{Rec}$
  - 結果: もし fval=⊥ ならば fval := f

受信者(つづき)

- $\text{fix-zval}_{\text{Rec}}$
  - 事前条件: ival,fval,yval≠⊥, zval=⊥
  - 結果:　zval(ival) := fval(yval(ival))
    zval(1-ival) := yval(1-ival)
- $\text{send(2,z)}_{\text{Rec}}$
  - 事前条件: z=zval
  - 結果: なし



送信者(続き)

- receive(2,z)$_{Trans}$
  - z∈{0,1}→D: Dは置換の領域
  - 結果: もし zval=⊥ ならば zval := z
- fix-bval$_{Trans}$
  - 事前条件: xval,pval,zval≠⊥, bval=⊥
  - 結果: for i∈{0,1} do

    bval(i) := B(pval.inv(zval(i)))⊕xval(i)
- send(3,b)$_{Trans}$
  - 事前条件: b=bval
  - 結果: なし

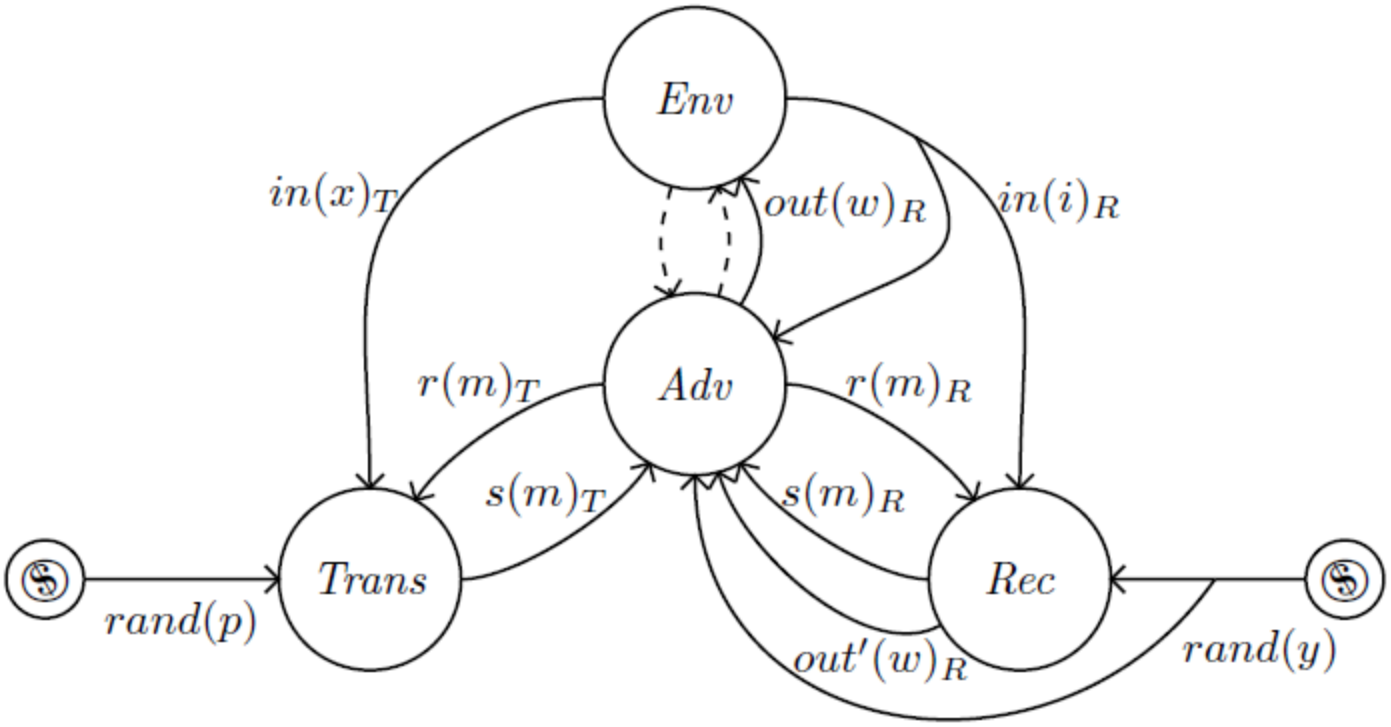
ハードコア述語: この場合、zval(i) から B(pval.inv(zval(i)) を推測することはできない。

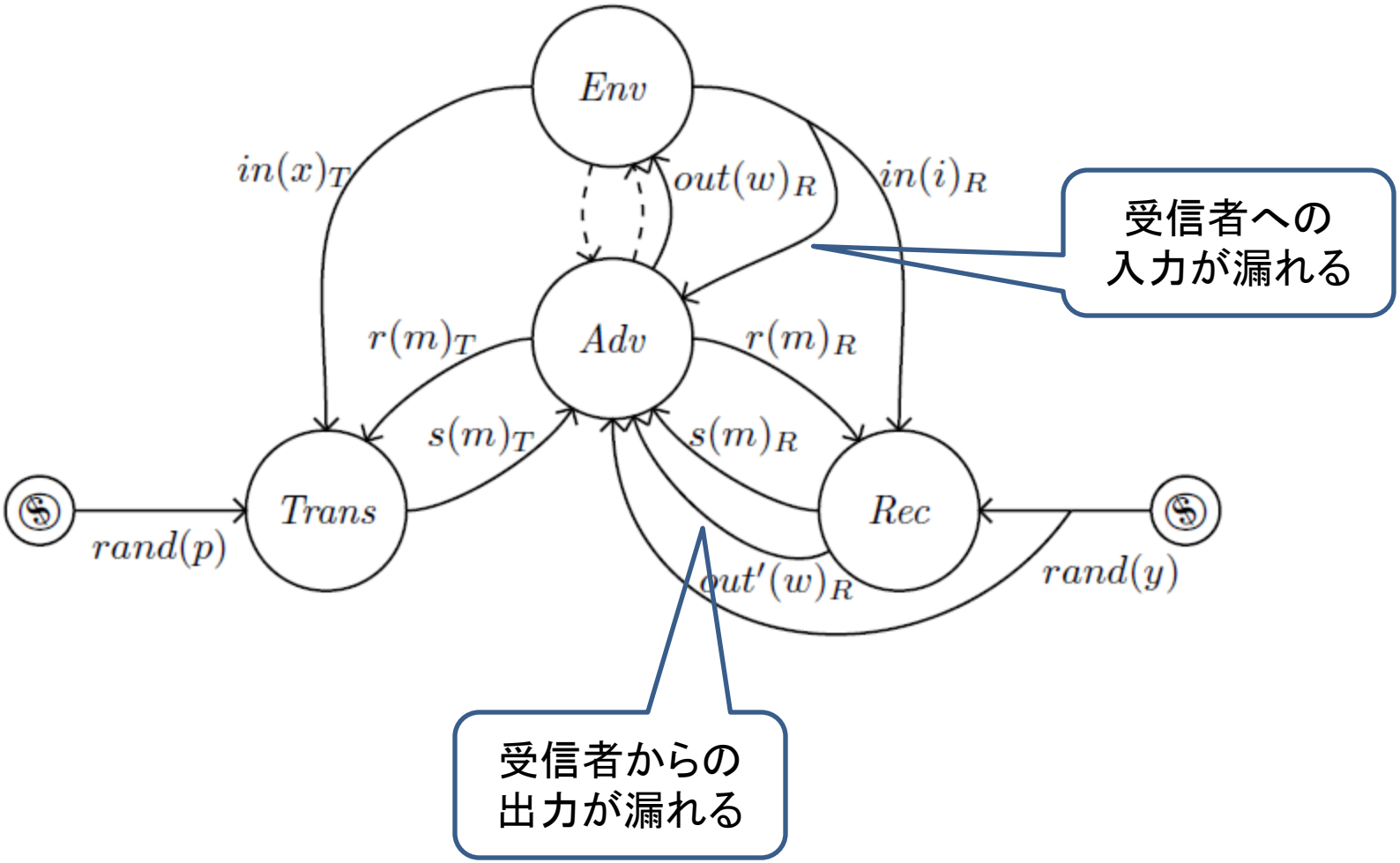
受信者（つづき）

- $\mathrm{receive}(3,b)_{Rec}$
  - 結果: もし ival,yval ≠ ⊥, wval= ⊥ ならば

    $wval := b(ival) \oplus B(yval(ival))$
- $\mathrm{out}'(w)_{Rec}$
  - 事前条件: w=wval
  - 結果: なし

yval(ival) = pval.inv(zval(ival))


Env
Adv
Trans
Rec
$in(x)_T$
$out(w)_R$
$in(i)_R$
$r(m)_T$
$r(m)_R$
$s(m)_T$
$s(m)_R$
$rand(p)$
$out'(w)_R$
$rand(y)$


Env
Adv
Trans
Rec
$in(x)_T$
$out(w)_R$
$in(i)_R$
$r(m)_T$
$r(m)_R$
$s(m)_T$
$s(m)_R$
$rand(p)$
$out'(w)_R$
$rand(y)$
受信者への
入力が漏れる
受信者からの
出力が漏れる

実現可能性

- 現実世界

  RS = Trans(D,Tdpp)||Rec(D,Tdp)||Adv
- 理想世界

  IS = Funct||Sim
- 任意の Adv に対して Sim が存在して、

  $RS \leqq_{neg,pt} IS$

  を示す。

$\leqq_{neg,pt}$

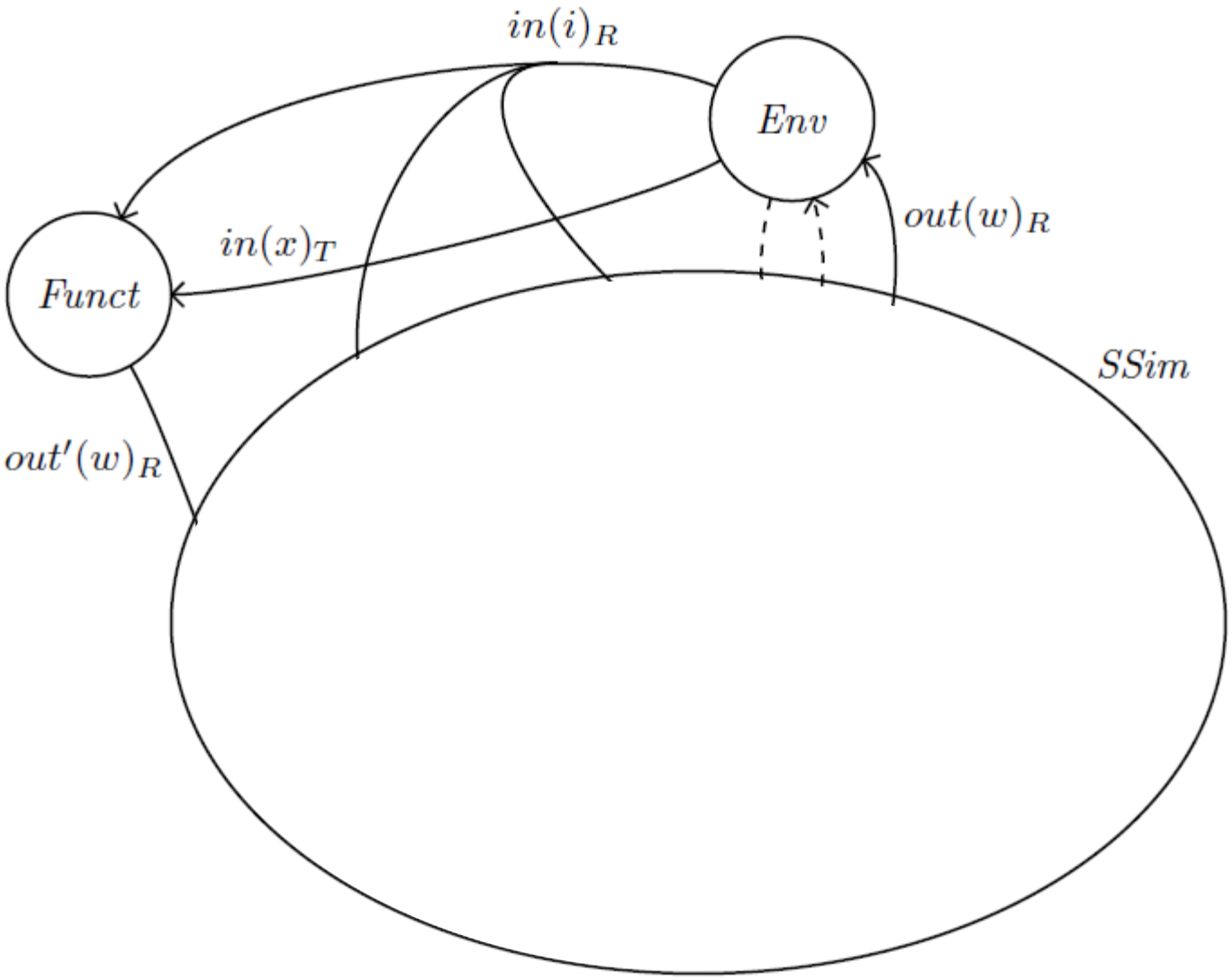
- T1 $\leqq_{neg,pt}$ T2 とは、任意の多項式時間環境 E のもとで、T1||E の任意のタスクスケジューラ ρ1 に対して、T2||E のあるタスクスケジューラ ρ2 が存在して、

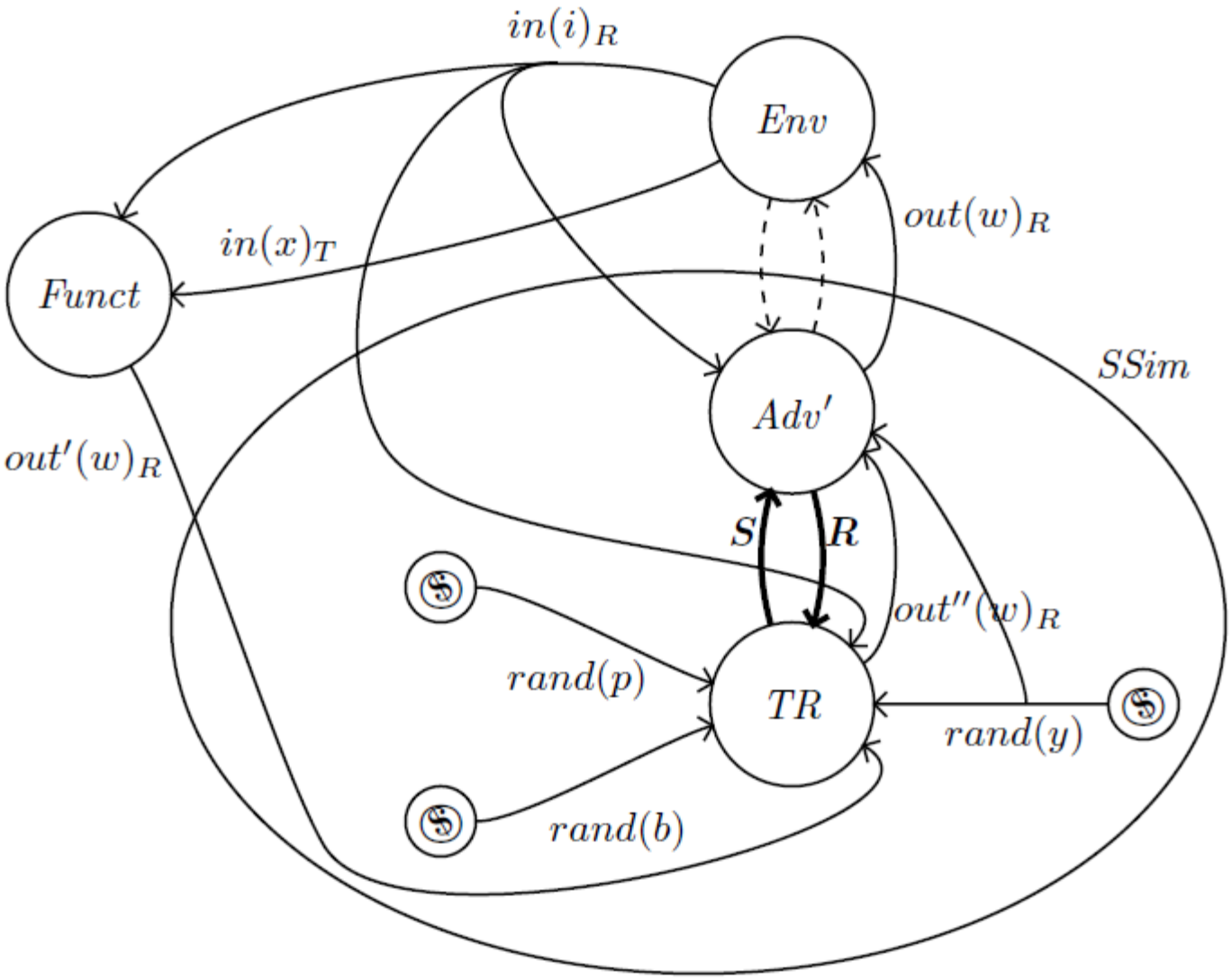
  $|Paccept(T1||E, \rho1) - Paccept(T1||E, \rho2)|$

  が negligible である。
- タスクスケジューラは、非決定的選択を指定。
- Paccept は、環境 E が accept を出力する確率。

$\leqq_0$

- T1 $\leqq_0$ T2 とは、任意の環境 E に対して、tdist(T1||E) $\subseteq$ tdist(T2||E) であること。
- tdist(T1||E) とは、T1||E を実行したときのトレースの確率分布の集合。
  - 非決定選択ごとに、確率分布が得られる。


$in(i)_R$
Env
$out(w)_R$
$in(x)_T$
$Funct$
$SSim$
$out'(w)_R$


$in(i)_R$
Env
$out(w)_R$
$in(x)_T$
Funct
SSim
Adv′
$out'(w)_R$
S
R
$out''(w)_R$
$rand(p)$
TR
$rand(y)$
$rand(b)$

TR(D,Tdpp)

- Sim = Adv||TR(D,Tdpp)
- TR(D,Tdpp) は、Trans(D,Tdpp) と Rec(D,Tdp) を合成し、若干の修正を加えたもの。
- 攻撃者は、i および x(i) の値を知ることができるが、x(1-i) の値を知ることができない。
  - TR(D,Tdpp) は ival および xval(ival) の値を用いることができる。

TR(D,Tdpp)

- receive(2,z)$_{Trans}$
  - z∈{0,1}→D: Dは置換の領域
  - 結果: もし zval=⊥ ならば zval := z
- fix-bval$_{Trans}$
  - 事前条件: xval,pval,zval≠⊥, bval=⊥
  - 結果: bval(ival) := B(yval(ival))⊕xval(ival)
    bval(1-ival) := ランダムビット
- send(3,b)$_{Trans}$
  - 事前条件: b=bval
  - 結果: なし

Int2

- receive(2,z)$_{Trans}$
  - z∈{0,1}→D: Dは置換の領域
  - 結果: もし zval=⊥ ならば zval := z
- fix-bval$_{Trans}$
  - 事前条件: xval,pval,zval≠⊥, bval=⊥
  - 結果: bval(ival) := B(yval(ival))⊕xval(ival)
    bval(1-ival) := ランダムビット⊕xval(1-ival)
- send(3,b)$_{Trans}$
  - 事前条件: b=bval
  - 結果: なし

Int1

- receive(2,z)$_{Trans}$
  - z∈{0,1}→D: Dは置換の領域
  - 結果: もし zval=⊥ ならば zval := z
- fix-bval$_{Trans}$
  - 事前条件: xval,pval,zval≠⊥, bval=⊥
  - 結果: bval(ival) := B(yval(ival))⊕xval(ival)
    bval(1-ival) :=
    B(pval.inv(zval(1-ival)))⊕xval(1-ival)
- send(3,b)$_{Trans}$
  - 事前条件: b=bval
  - 結果: なし

証明の概要

- 中間システム Int1 と Int2 に対して、以下を示す。
  - RS $\leqq_0$ Int1
  - Int1 $\leqq_{neg,pt}$ Int2
  - Int2 $\leqq_0$ IS
- 推移性
- Int1 $\leqq_{neg,pt}$ Int2
  - ハードコア述語の性質
  - 結合性