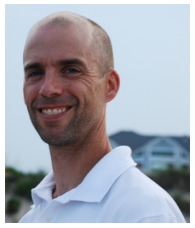
MODELS AND GAMES FOR **QUANTIFYING VULNERABILITY** OF SECRET INFORMATION

Piotr (Peter) Mardziel

University of Maryland, College Park

Stephen Magill
Galois



Michael Hicks
UMD

Mudhakar Srivatsa
IBM TJ Watson

Jonathan Katz
UMD

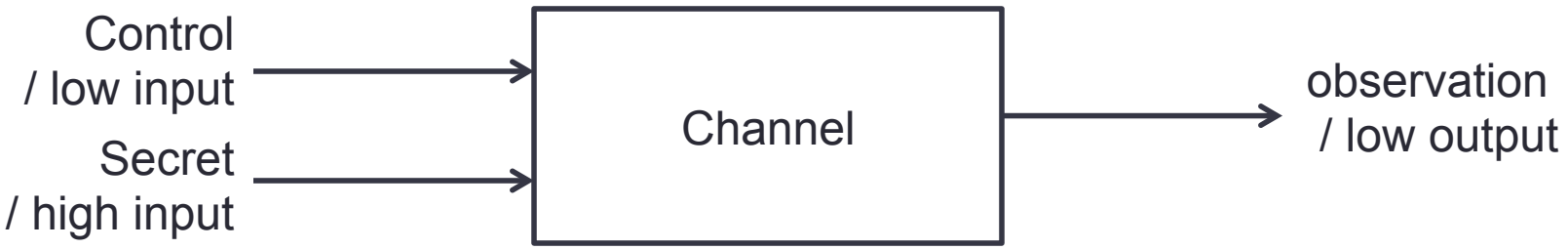
Mário Alvim
UFMG, Brazil

Michael Clarkson
Cornell

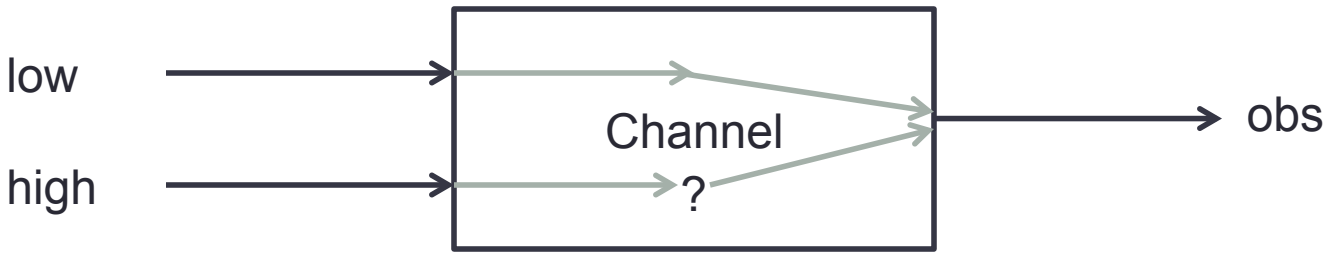
Arman Khouzani
Queen Mary

Carlos Cid
Royal Holloway

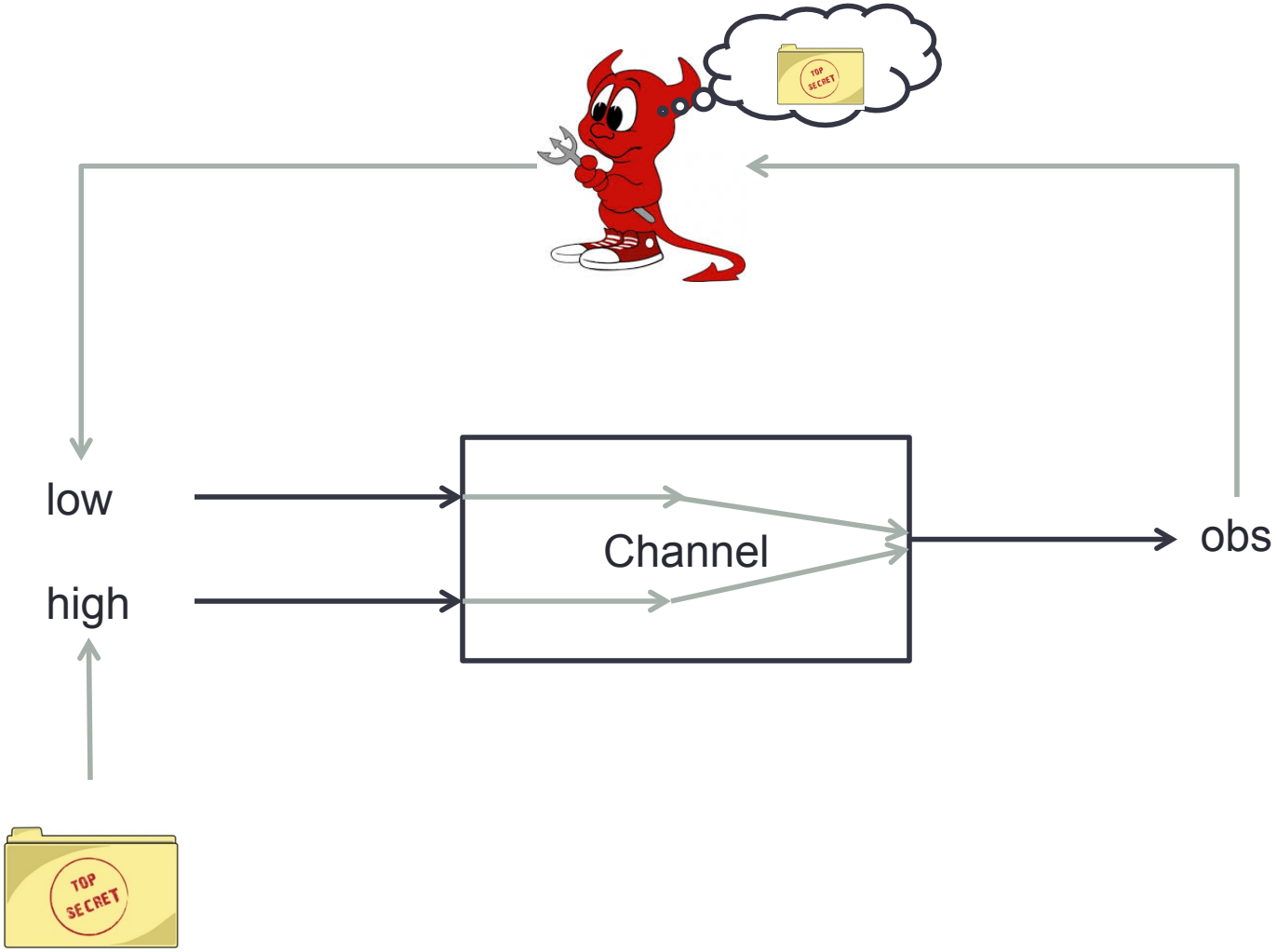
Information flow



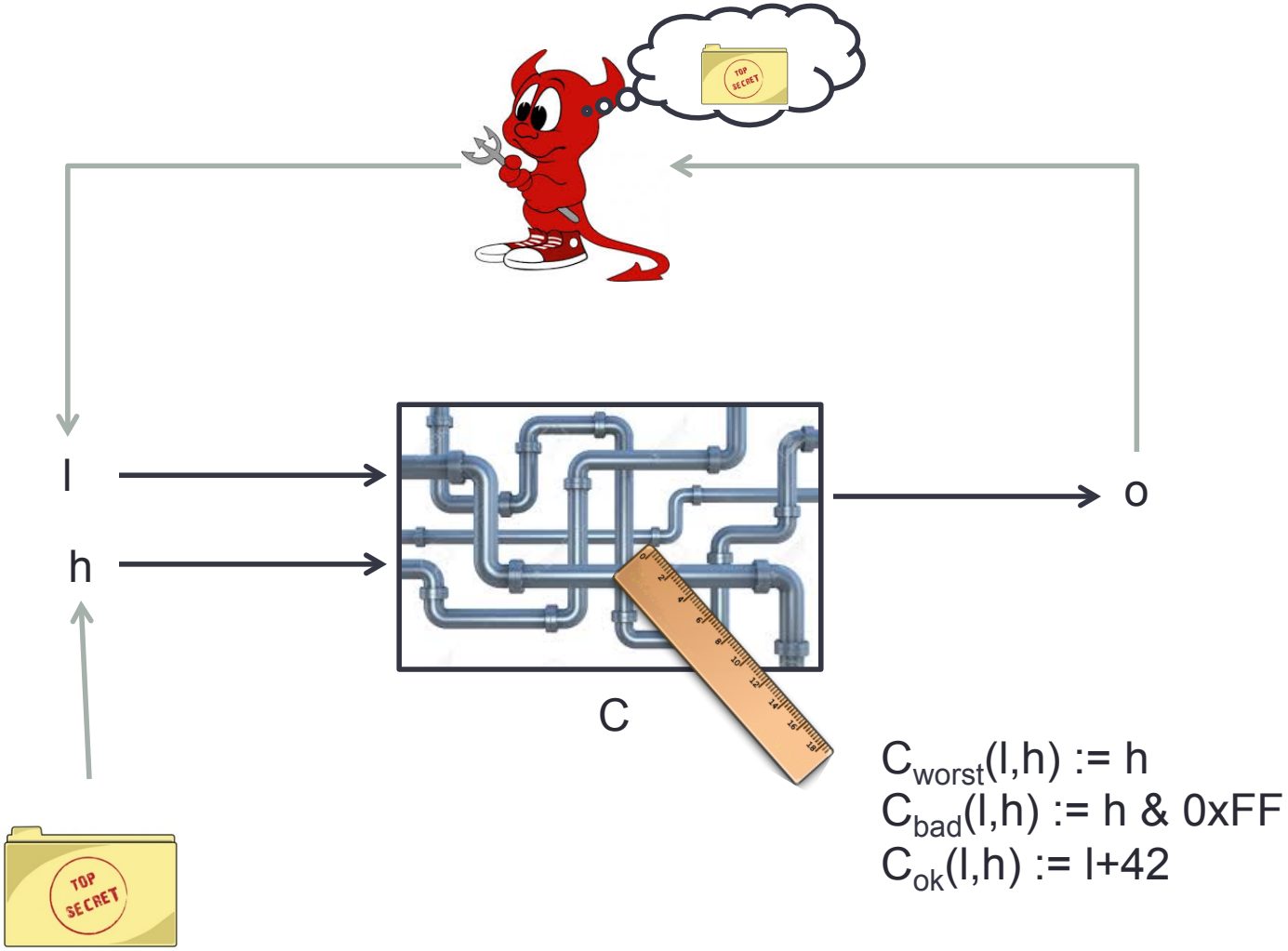
Information flow



Information flow



Quantitative Information flow

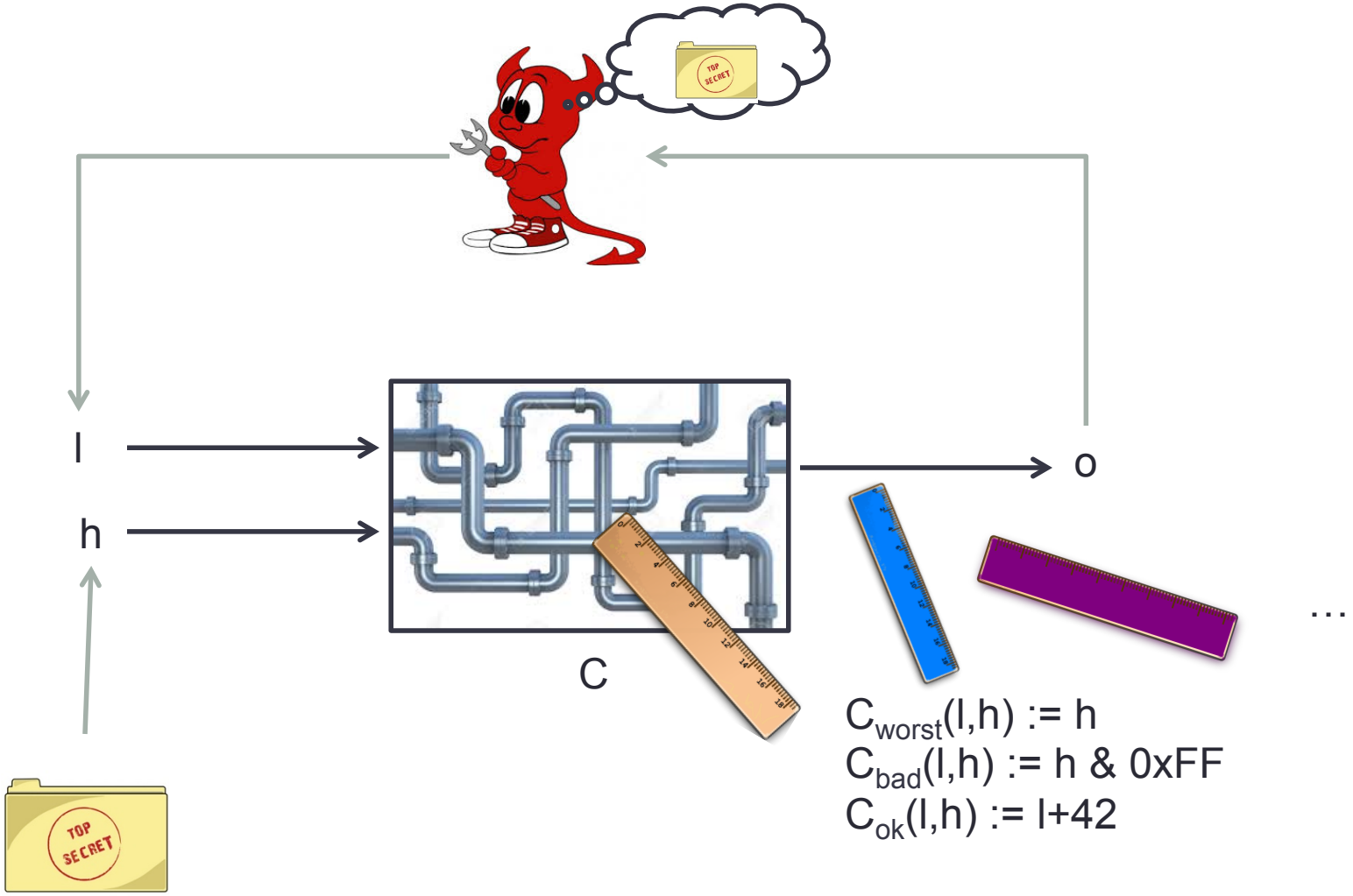


$C_{worst}(l,h) := h$
$C_{bad}(l,h) := h \,\&\, 0xFF$
$C_{ok}(l,h) := l+42$

QIF Examples

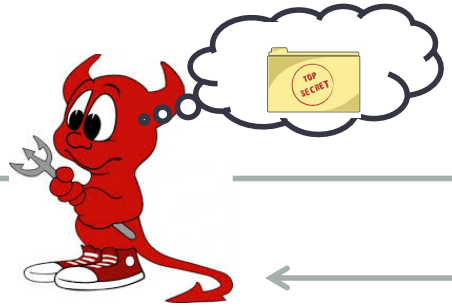
	Example	High	Channel	Low	Observation
	authentication	password	Login interface	Password guess	Login OK/Fail
modeling	ASLR	memory offset	Injection attempt	Offset guess	Payload success/Crash
Information theoretic	Encryption	key	Encryption	Plaintext	Ciphertext
More than input/output	Encryption timing	key	Encryption runtime model	Plaintext	Runtime of encryption

QIF Metrics



$C_{worst}(l,h) := h$
$C_{bad}(l,h) := h \,\&\, 0xFF$
$C_{ok}(l,h) := l+42$

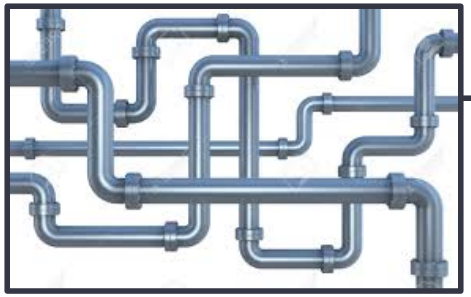
Quantified vulnerability



Exploit e

l

h



o

C

Gain Model

g

h

G(h,e) := 1 if h == e
0 o/w

Formalism

$$H' := \{h \in H : C(h,l) = o\}$$

Exploit e

l

h

o

Gain Model
$G : H \times E \to \mathbb{R}$

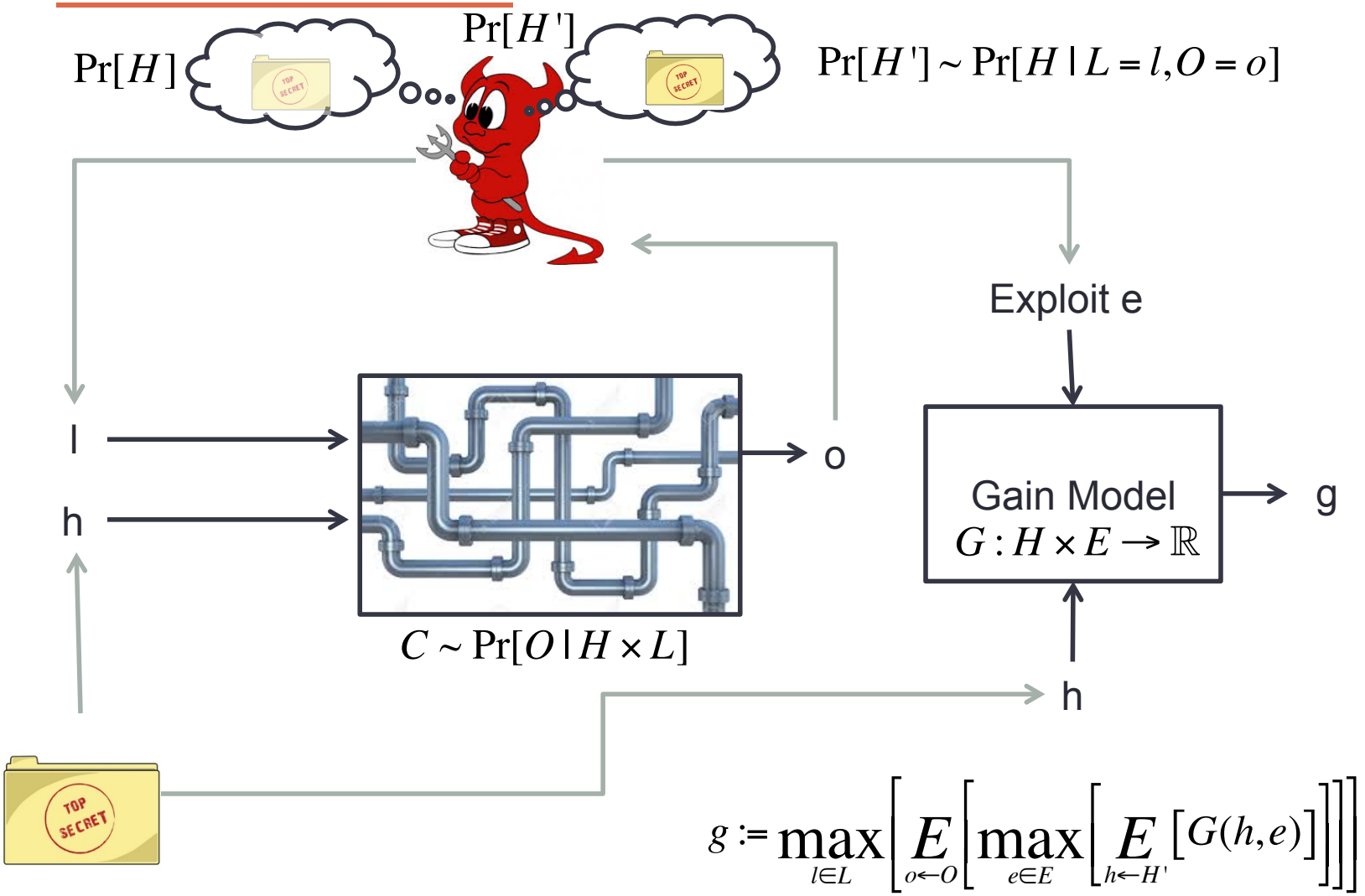
g

$C : H \times L \to O$

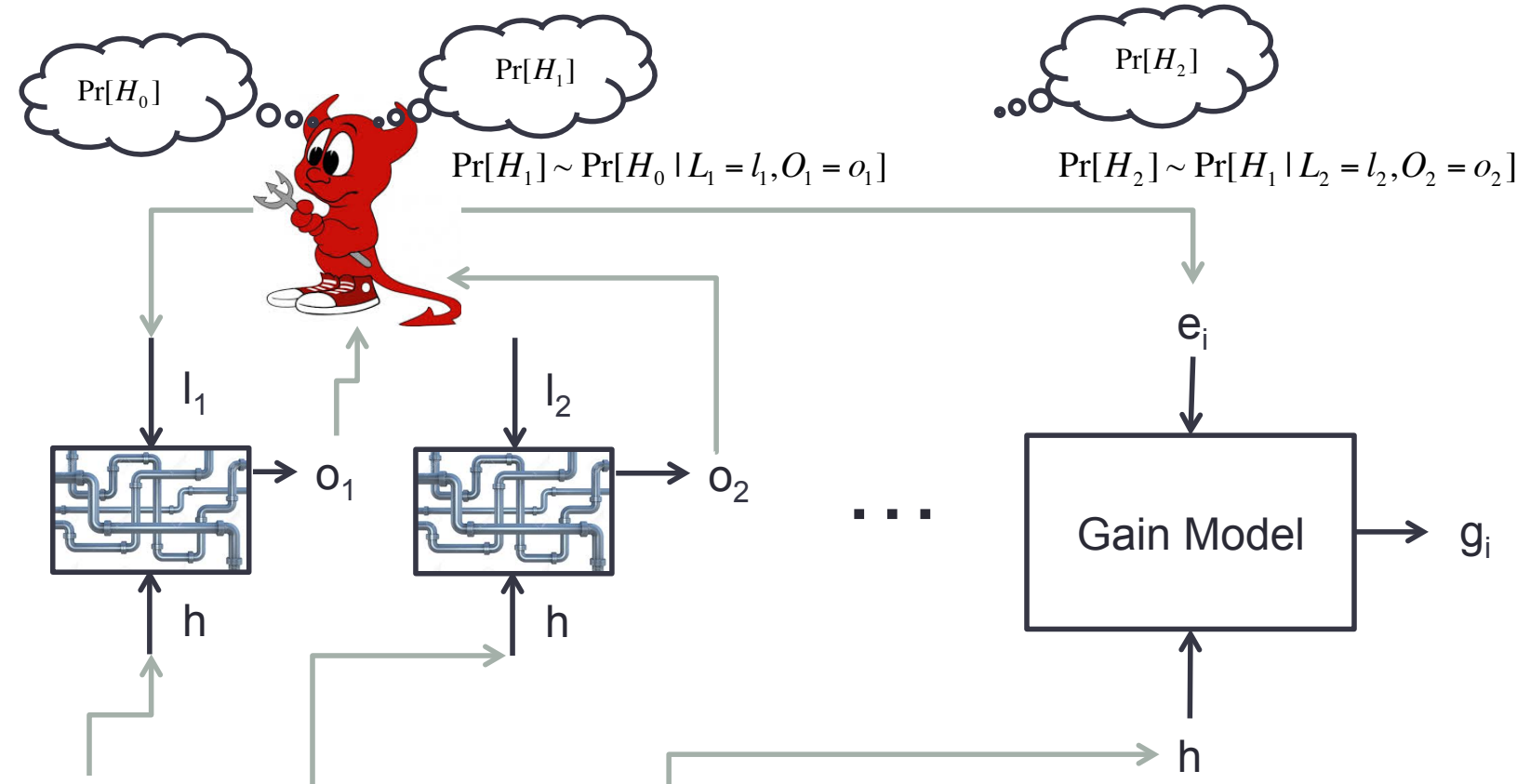
h

$$g := \max_{l \in L}\left[??_{o \in O}\left[\max_{e \in E}\left[??_{h \in H'}[G(h,e)]\right]\right]\right]$$

Probabilistic Formalism



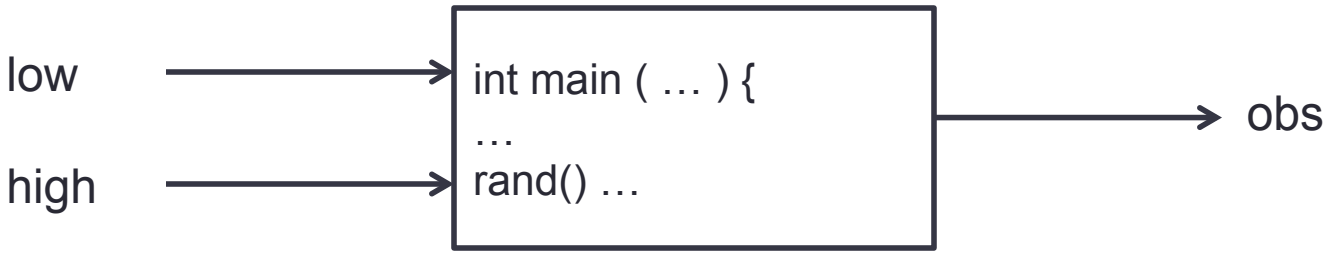
Iteration



$$g := \max_{l_1 \in L} \max_{l_2 \in L} \cdots \left[\underset{o_1 \leftarrow O_1}{E} \underset{o_2 \leftarrow O_2}{E} \cdots \left[\max_{e \in E} \left[\underset{h \leftarrow H_i}{E} [G(h,e)] \right] \right] \right]$$

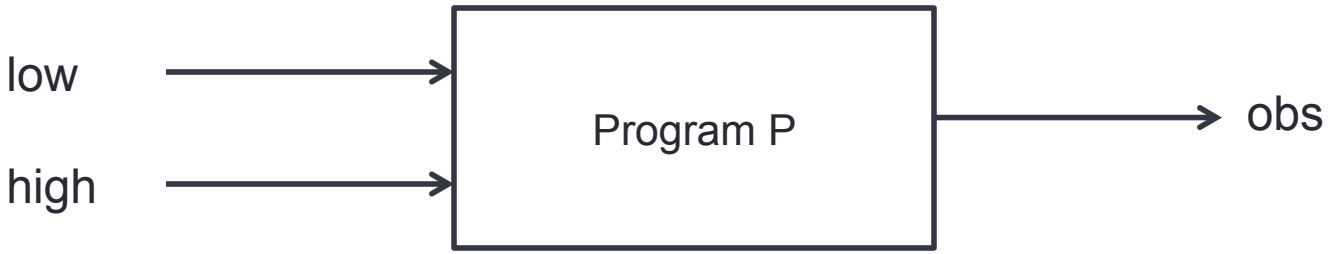
- Basics
- Channels as programs
  - Programs as channels
- Models for …
  - Adaptive adversaries
  - Time-varying secrets
  - Non-zero-sum games
  - Active defenders, equilibrium

Channel → Program



$$C \sim \Pr[O \mid H \times L]$$

Probabilistic programming



$$P : H \times L \rightarrow O$$

$$P \sim \Pr[O \mid H \times L]$$

inference

$$\Pr[H \mid L, O]$$

Program → Channel

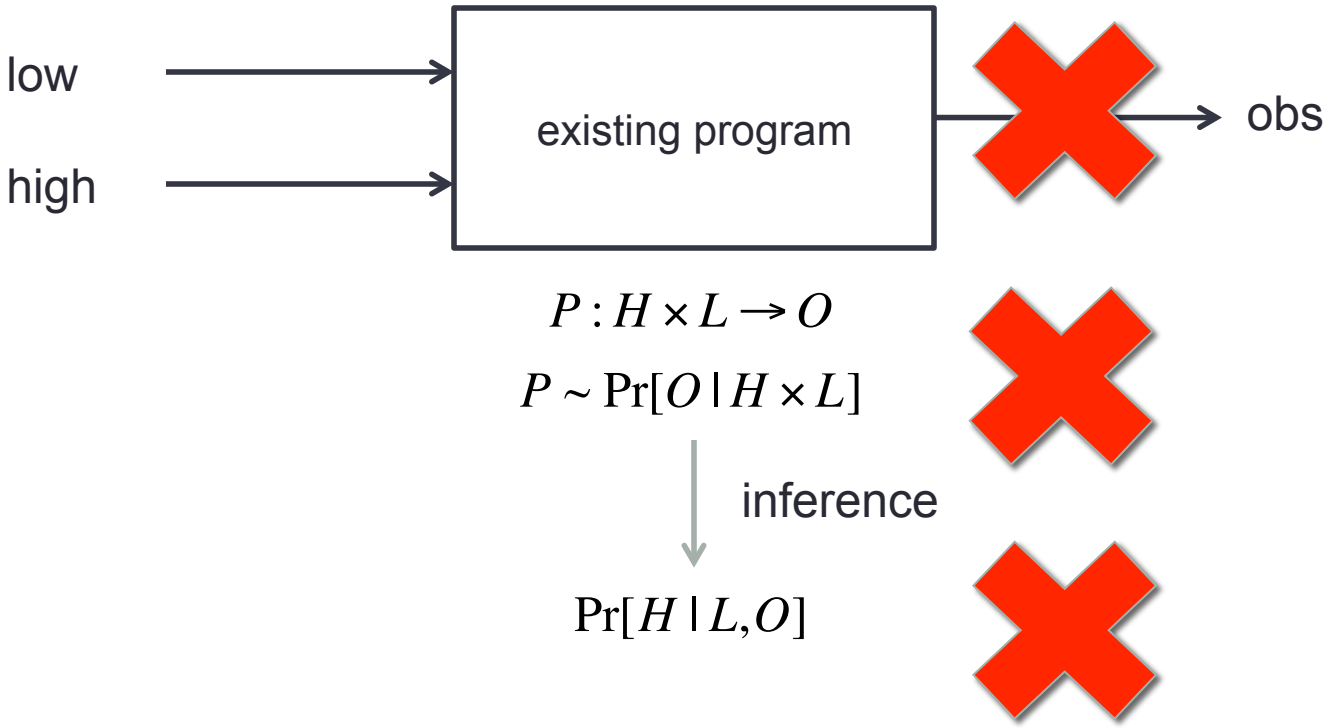
$$P : H \times L \rightarrow O$$

$$P \sim \Pr[O \mid H \times L]$$

inference

$$\Pr[H \mid L, O]$$

“PL”


low
high
existing program
obs
$P : H \times L \to O$
$P \sim \Pr[O \mid H \times L]$
inference
$\Pr[H \mid L, O]$

"PL"

$$P : H \times L \rightarrow O$$

$$P \sim \Pr[O \mid H \times L]$$

approximate inference

$$\Pr[H \mid L, O]$$

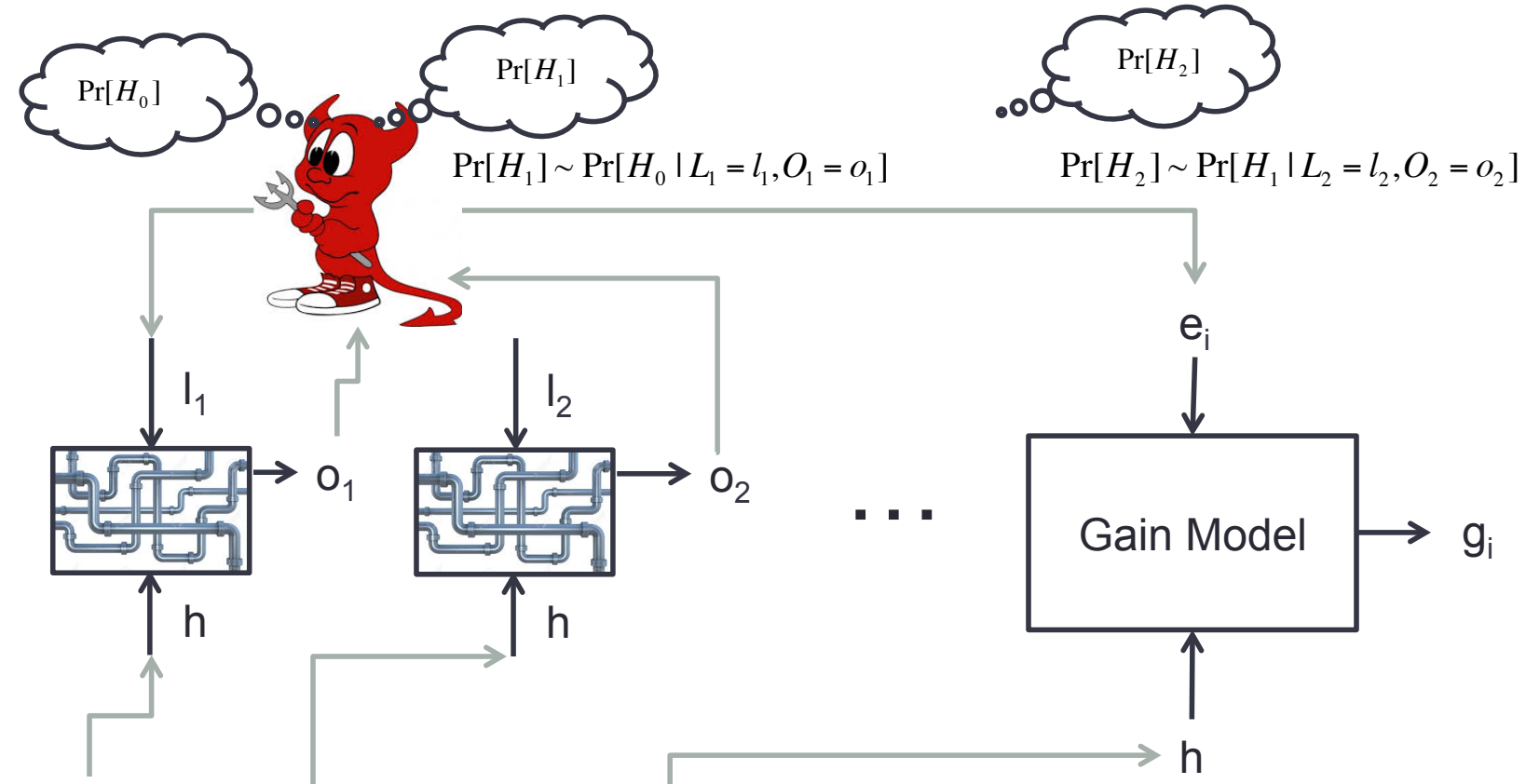
over-approximation of vulnerability

g

- Basics
- Channels as programs
  - Programs as channels
- Models for …
  - Adaptive adversaries
  - Time-varying secrets
  - Non-zero-sum games
  - Active defenders, equilibrium

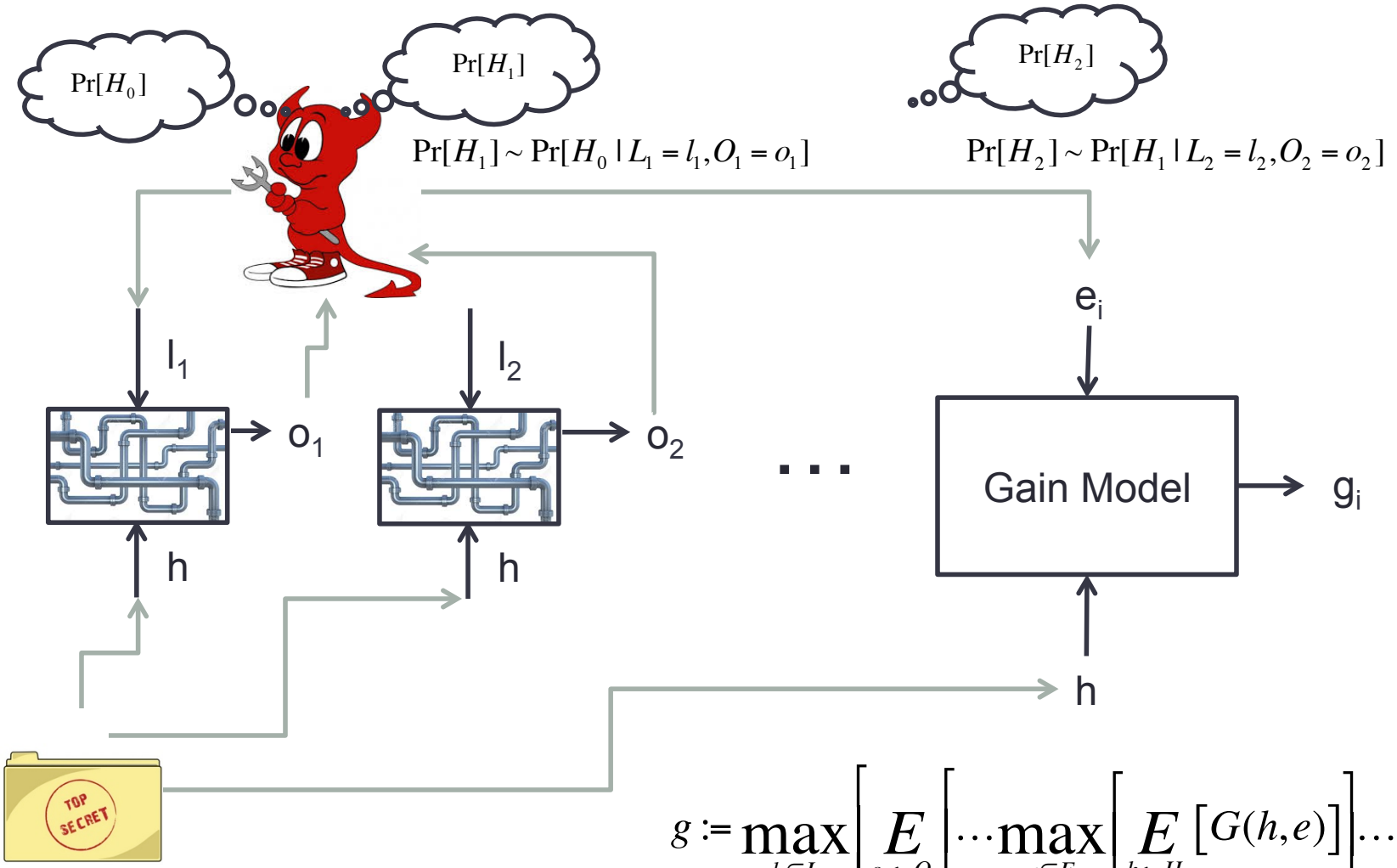
- Basics
- Channels as programs
  - Programs as channels
- Models for …
  - **Adaptive adversaries**
  - Time-varying secrets
  - Non-zero-sum games
  - Active defenders, equilibrium

Adaptive adversary choice



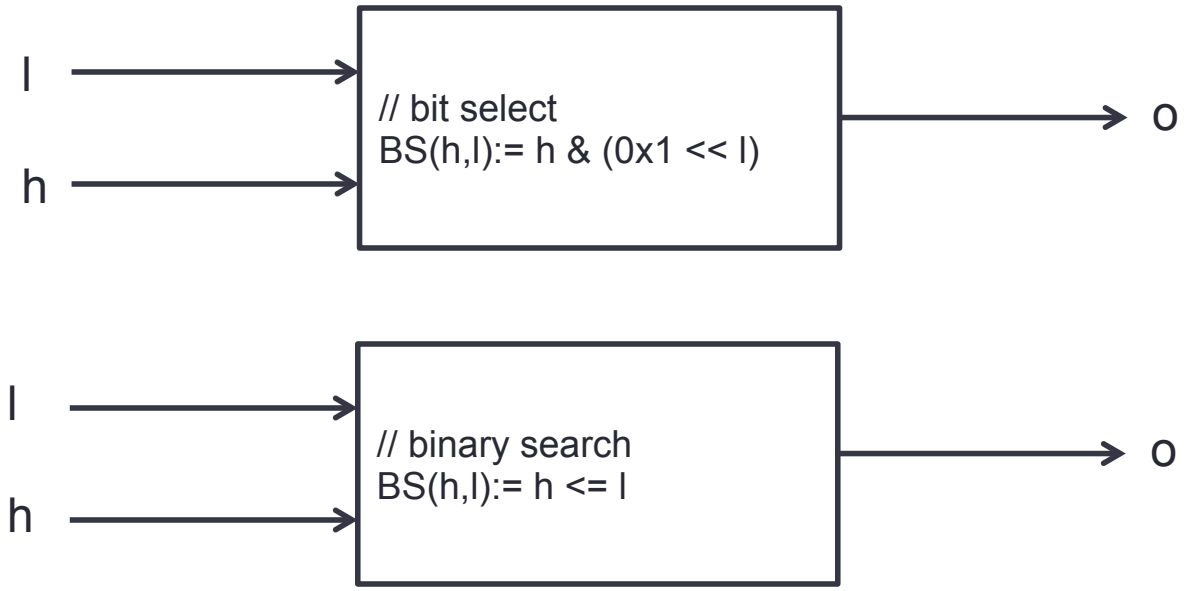
$$g := \max_{l_1 \in L} \max_{l_2 \in L} \cdots \left[\underset{o_1 \leftarrow O_1}{E} \underset{o_2 \leftarrow O_2}{E} \cdots \left[\max_{e \in E} \left[\underset{h \leftarrow H_i}{E} [G(h,e)] \right] \right] \right]$$

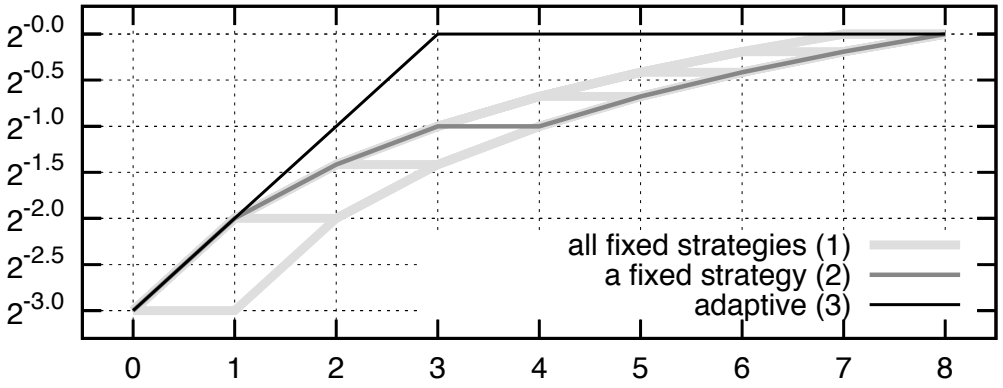
Backward inference



$$g := \max_{l_1 \in L} \left[E_{o_1 \leftarrow O_1} \left[\cdots \max_{e \in E} \left[E_{h \leftarrow H_i} [G(h,e)] \right] \cdots \right] \right]$$

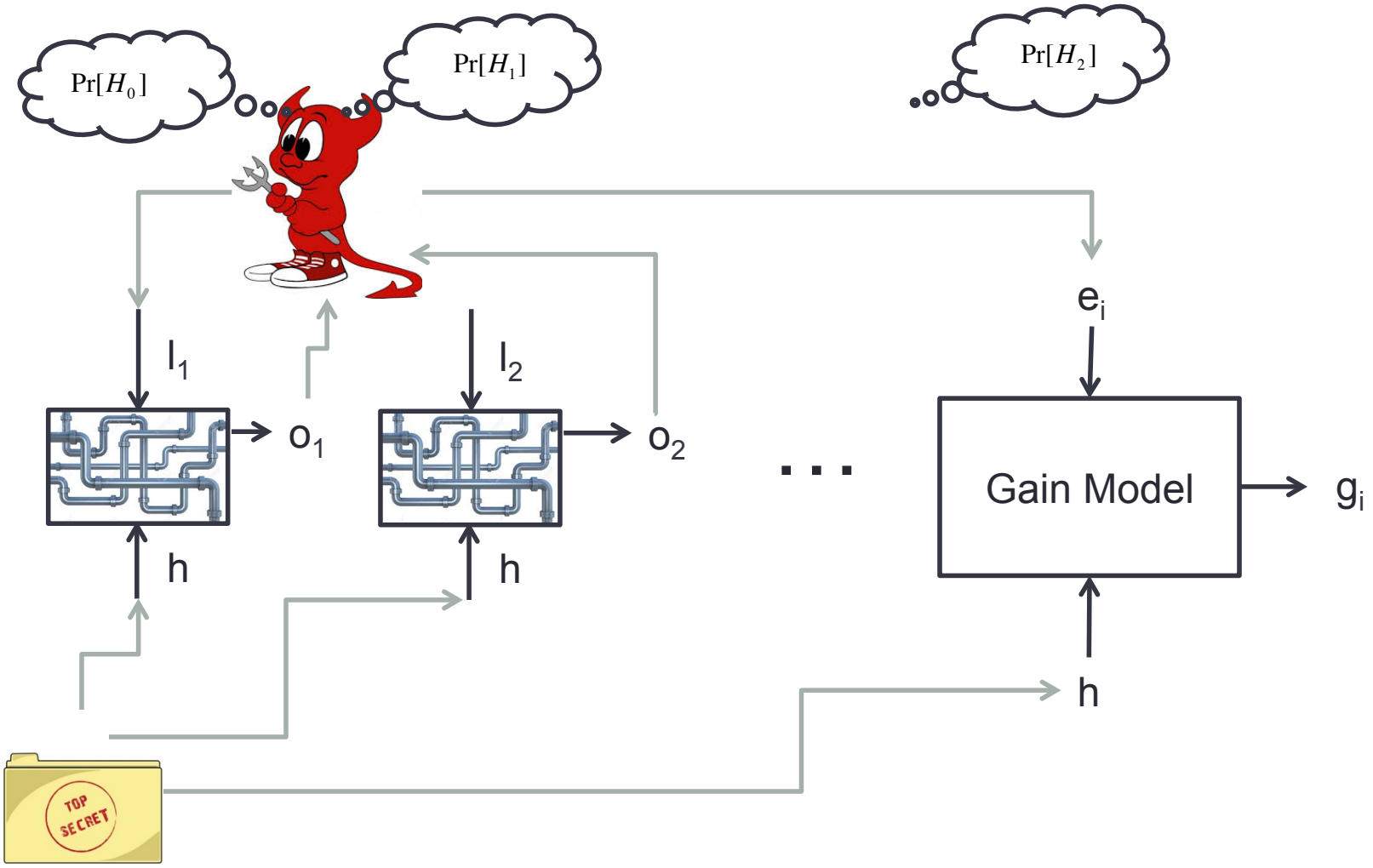
Adaptive power



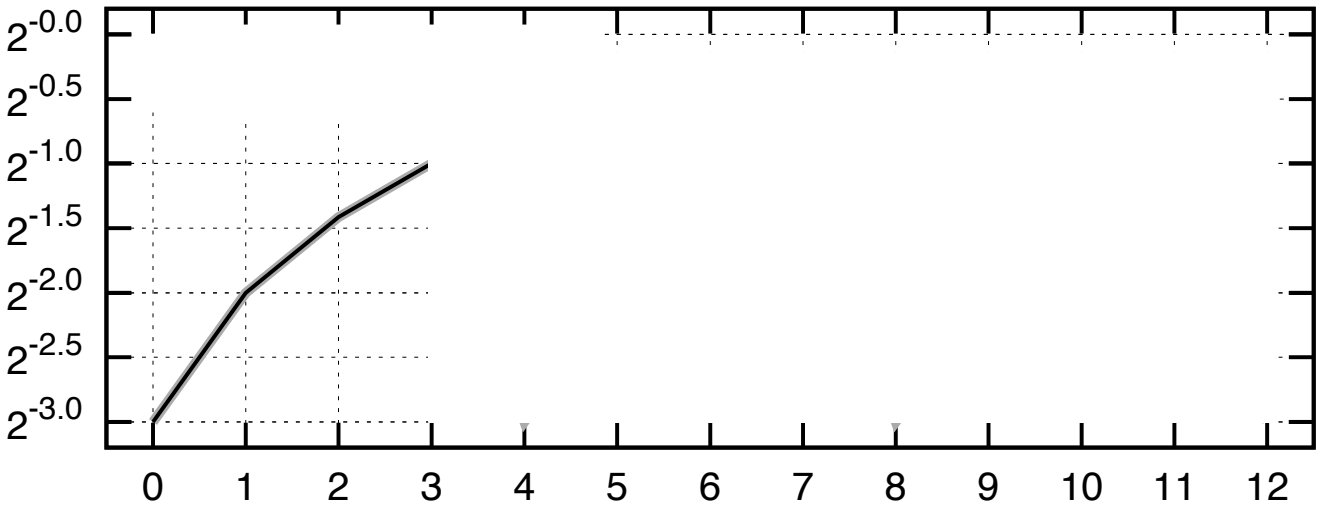


- Basics
- Channels as programs
  - Programs as channels
- Models for …
  - Adaptive adversaries
  - **Time-varying secrets**
  - Non-zero-sum games
  - Active defenders, equilibrium

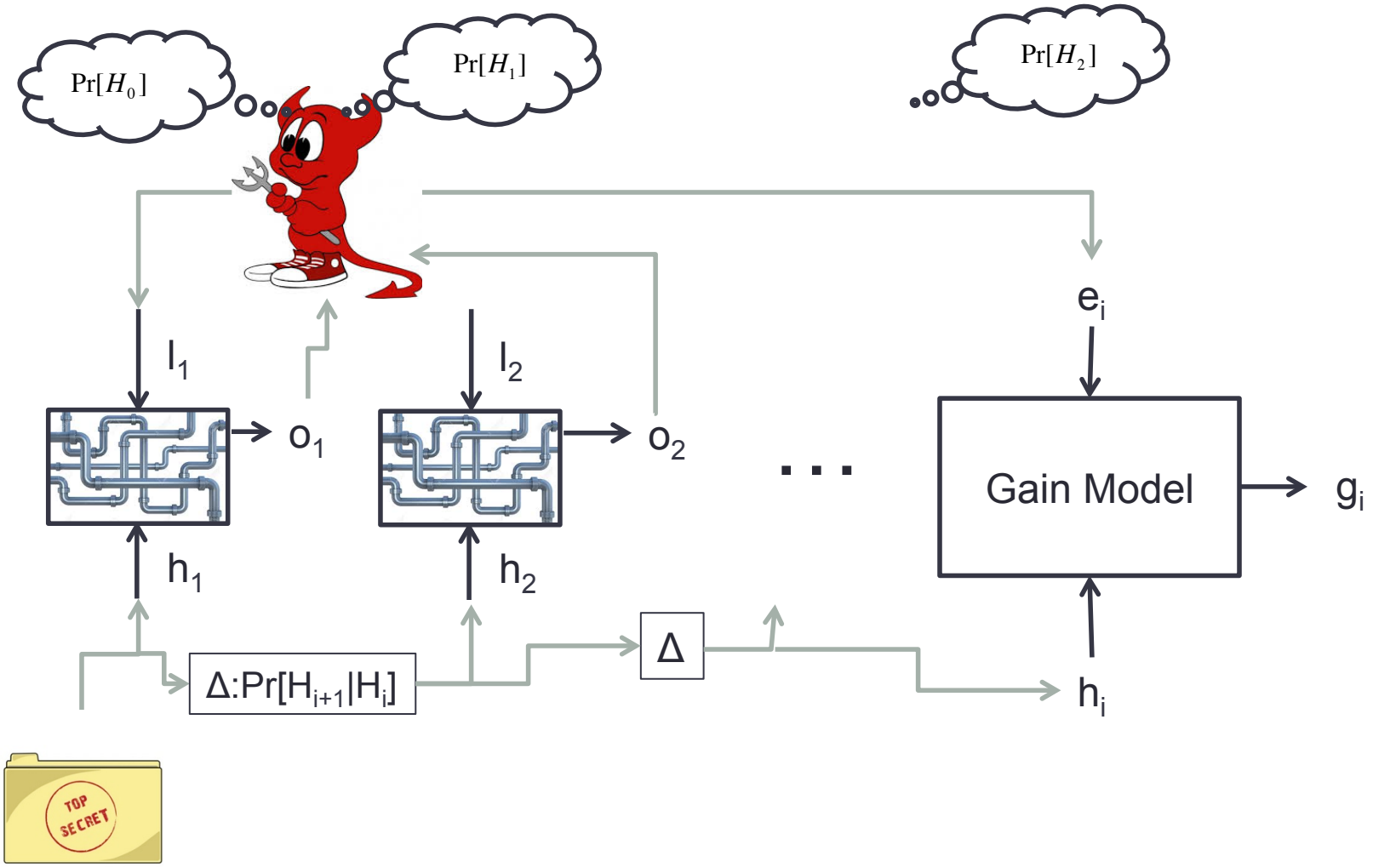
Entropy: non-renewable resource



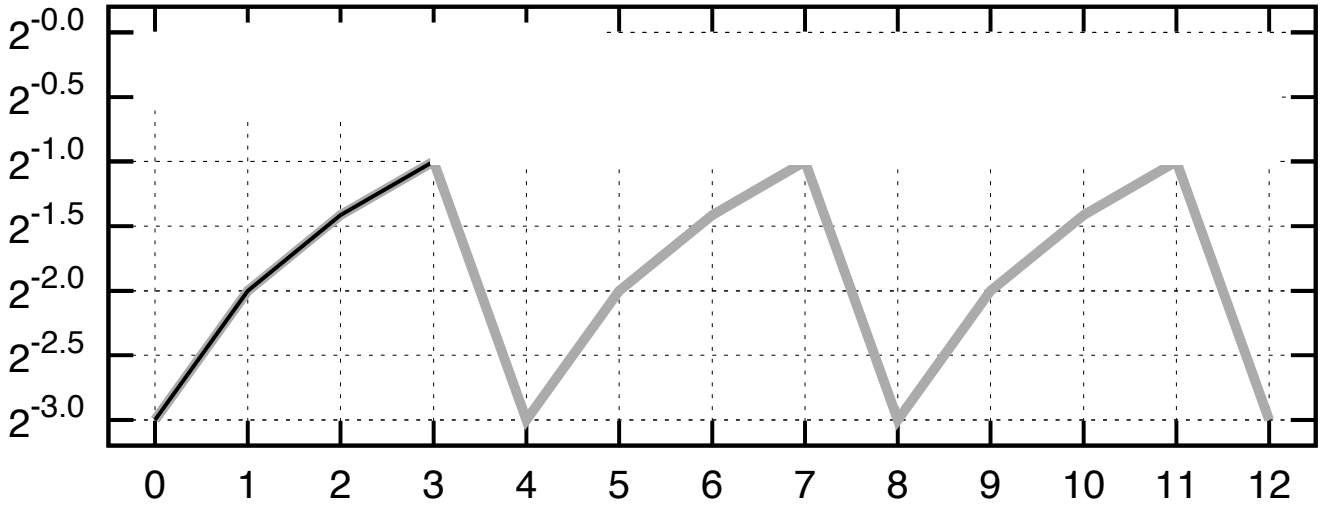
Entropy: non-renewable resource



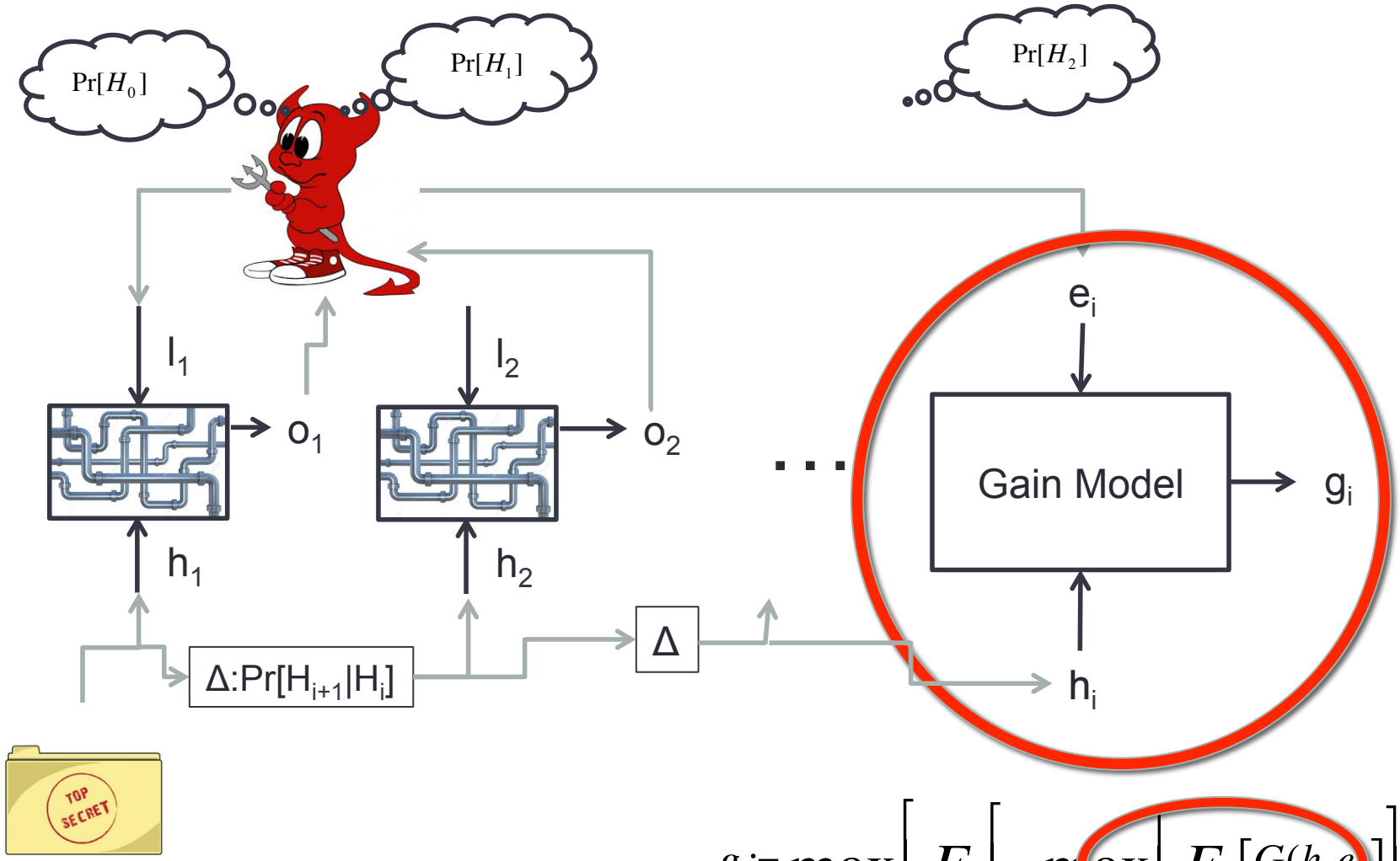
Evolving secrets



Entropy: renewable resource?

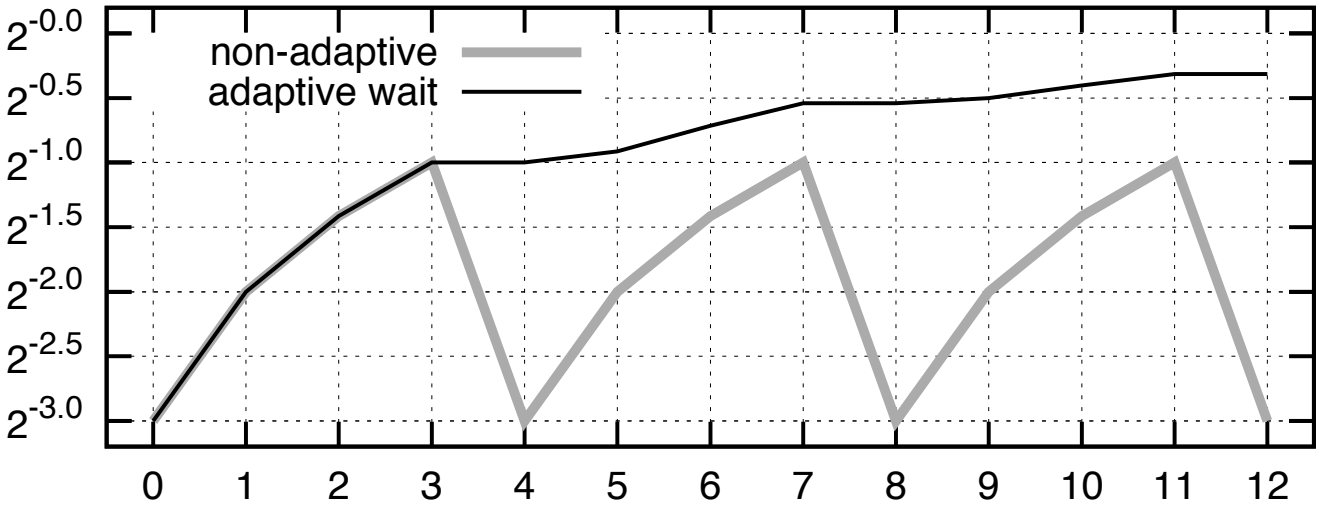


Adaptive exploitation

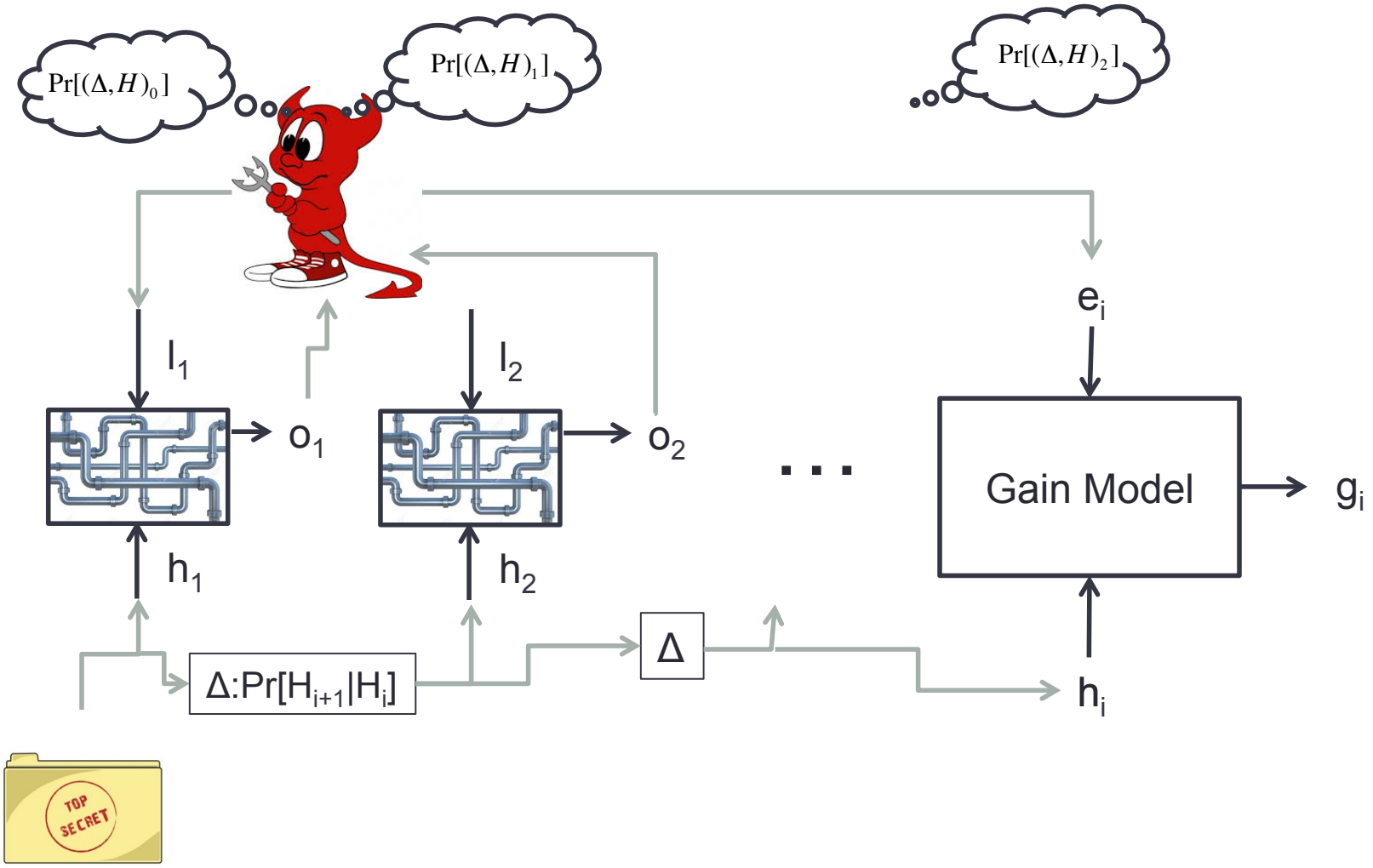


$$g := \max_{l_1 \in L} \left[E_{o_1 \leftarrow O_1} \left[\cdots \max_{e \in E} \left[E_{h \leftarrow H_i} [G(h,e)] \right] \cdots \right] \right]$$

Adaptive exploitation power



Belief in Δ


Pr[(Δ,H)0]
Pr[(Δ,H)1]
Pr[(Δ,H)2]
l1
l2
o1
o2
h1
h2
ei
Gain Model
gi
hi
Δ
Δ:Pr[Hi+1|Hi]
TOP SECRET

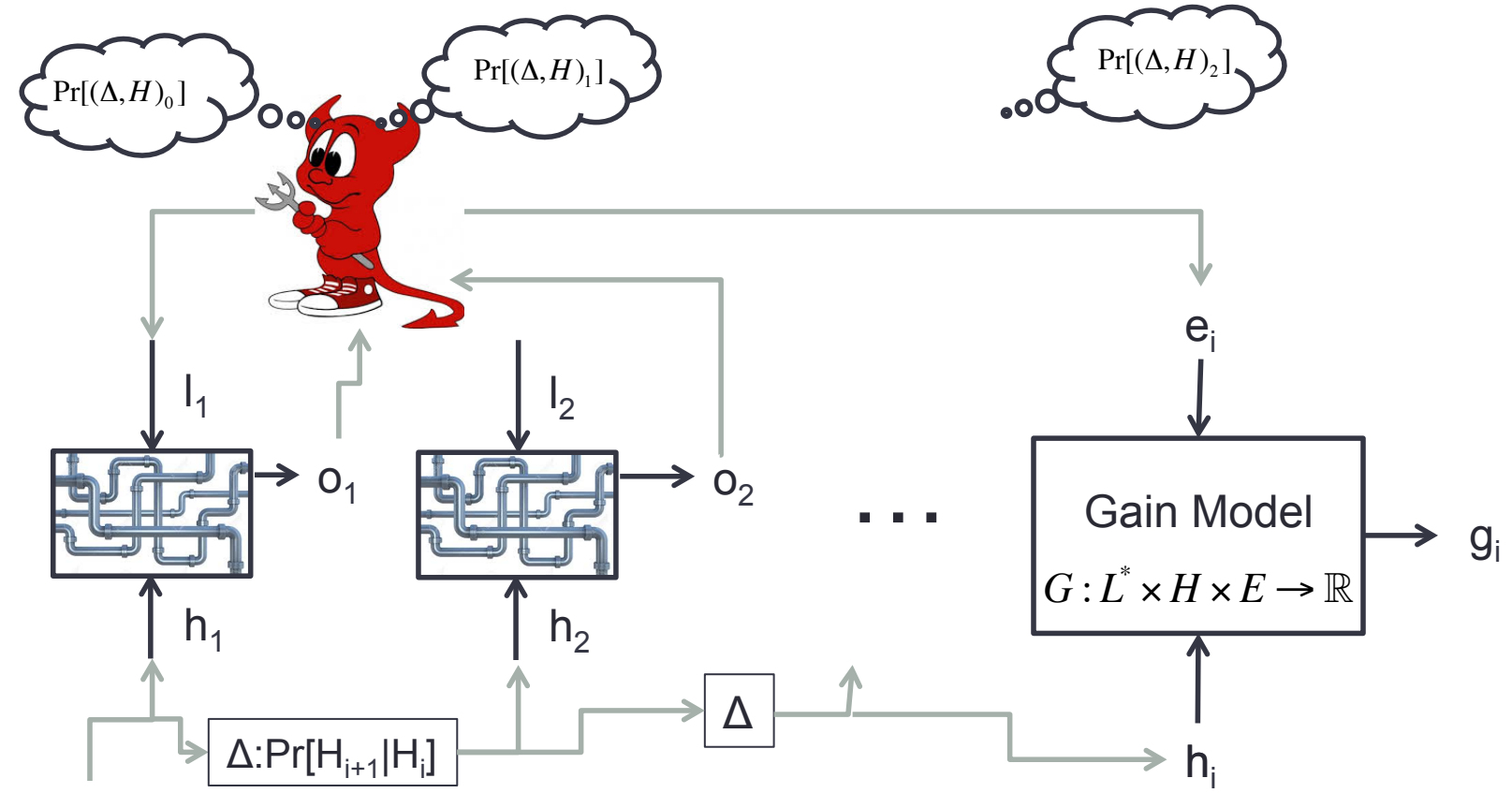
Entropy of H vs. Entropy of Δ

password42 vs.

password42
password43
password44

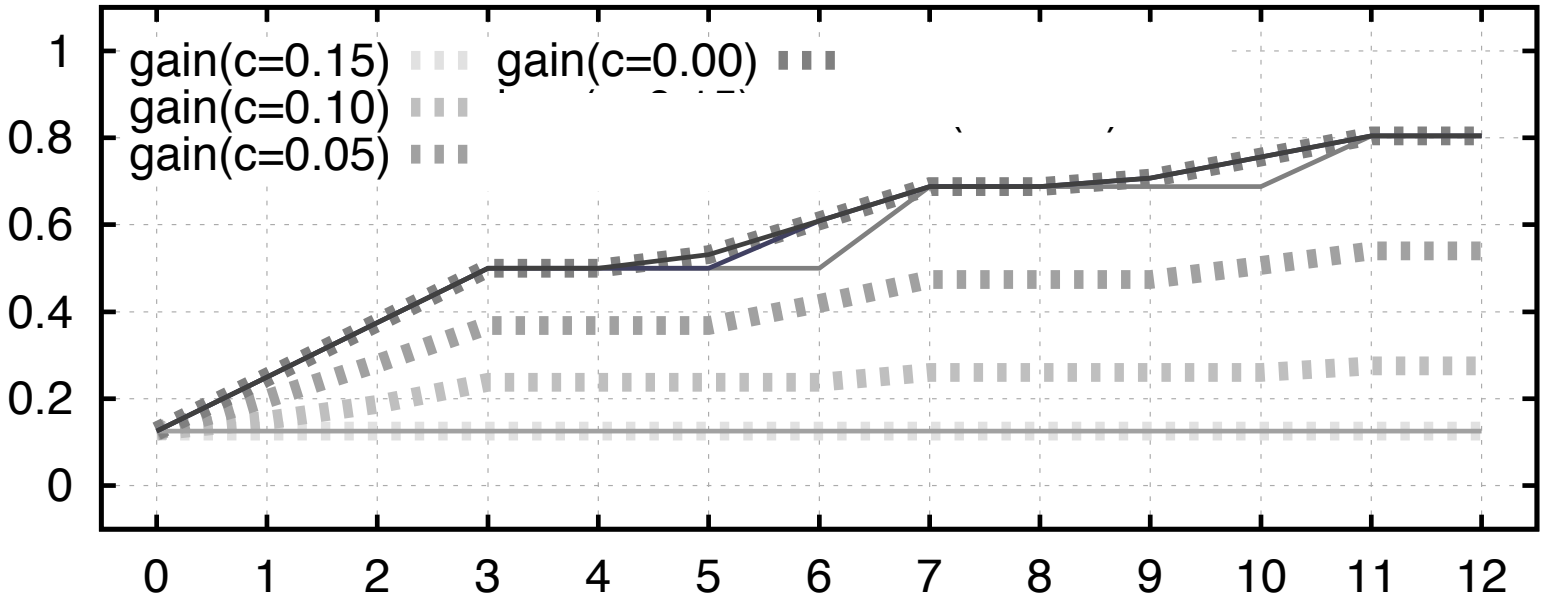
- Basics
- Channels as programs
  - Programs as channels
- Models for …
  - Adaptive adversaries
  - Time-varying secrets
  - **Non-zero-sum games**
  - Active defenders, equilibrium

Generalizing gain model

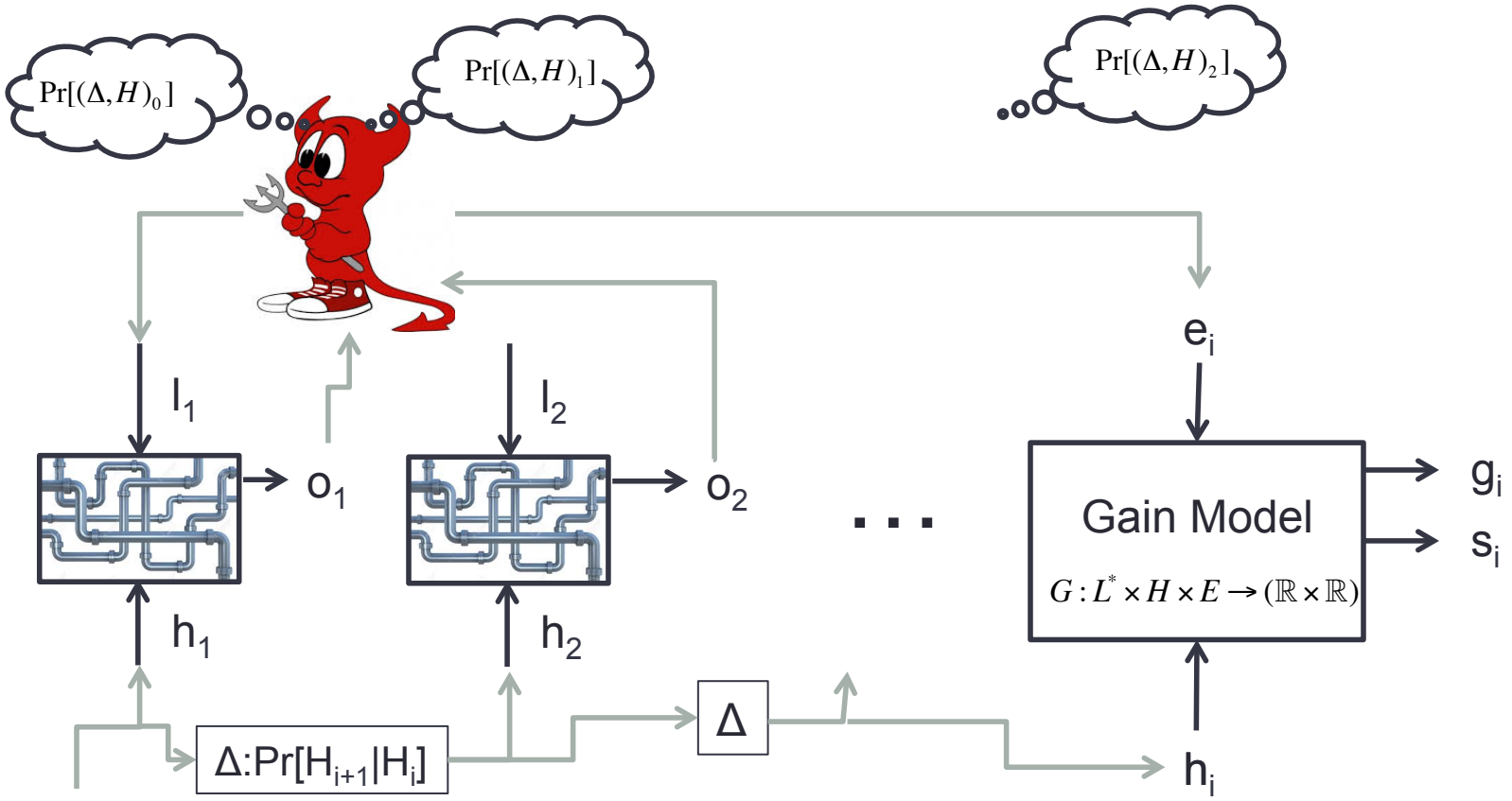


L:={observe, not-observe,…}
G(…):= … - 0.1 * observations

Gain with costly observation

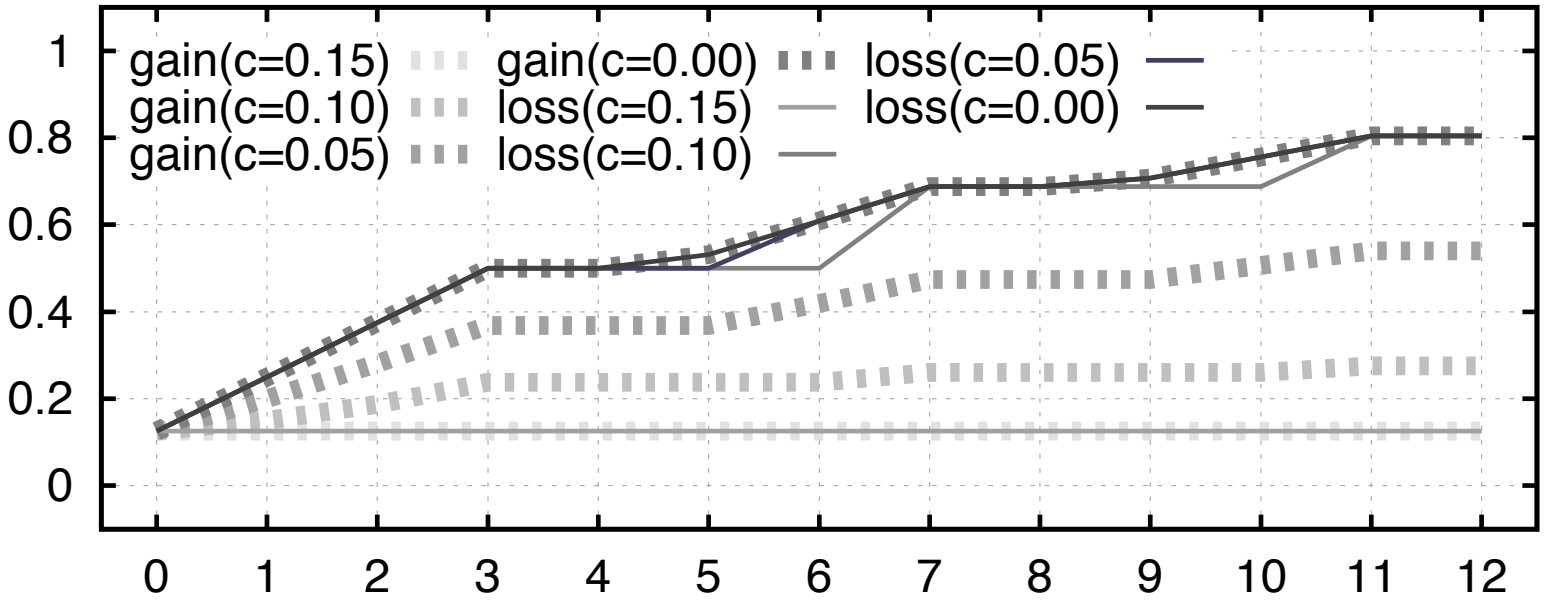


Gain vs. Loss



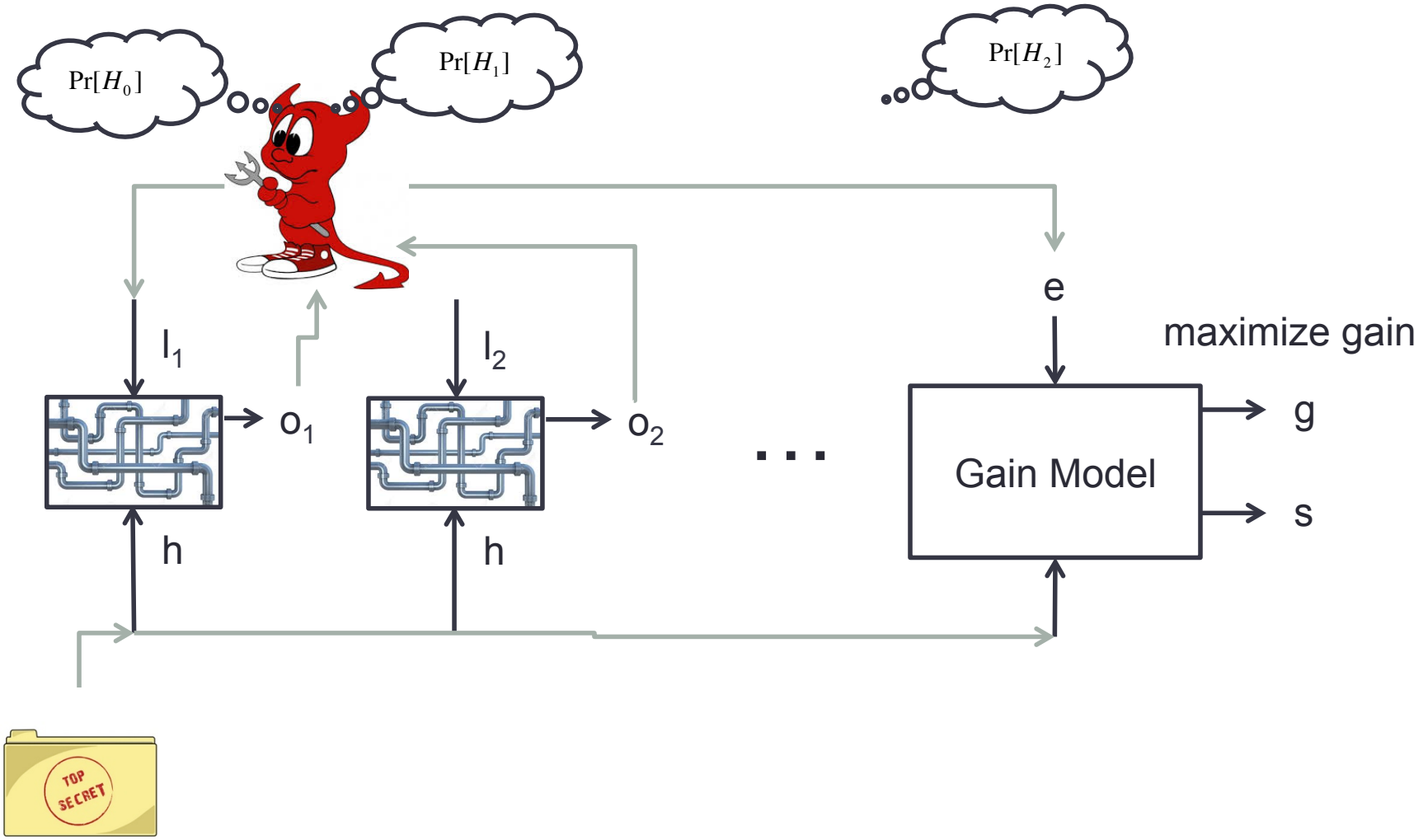
L:={observe, not-observe,…}
G(…):= g = … - 0.1 * observations
s = …

Gain vs. Loss

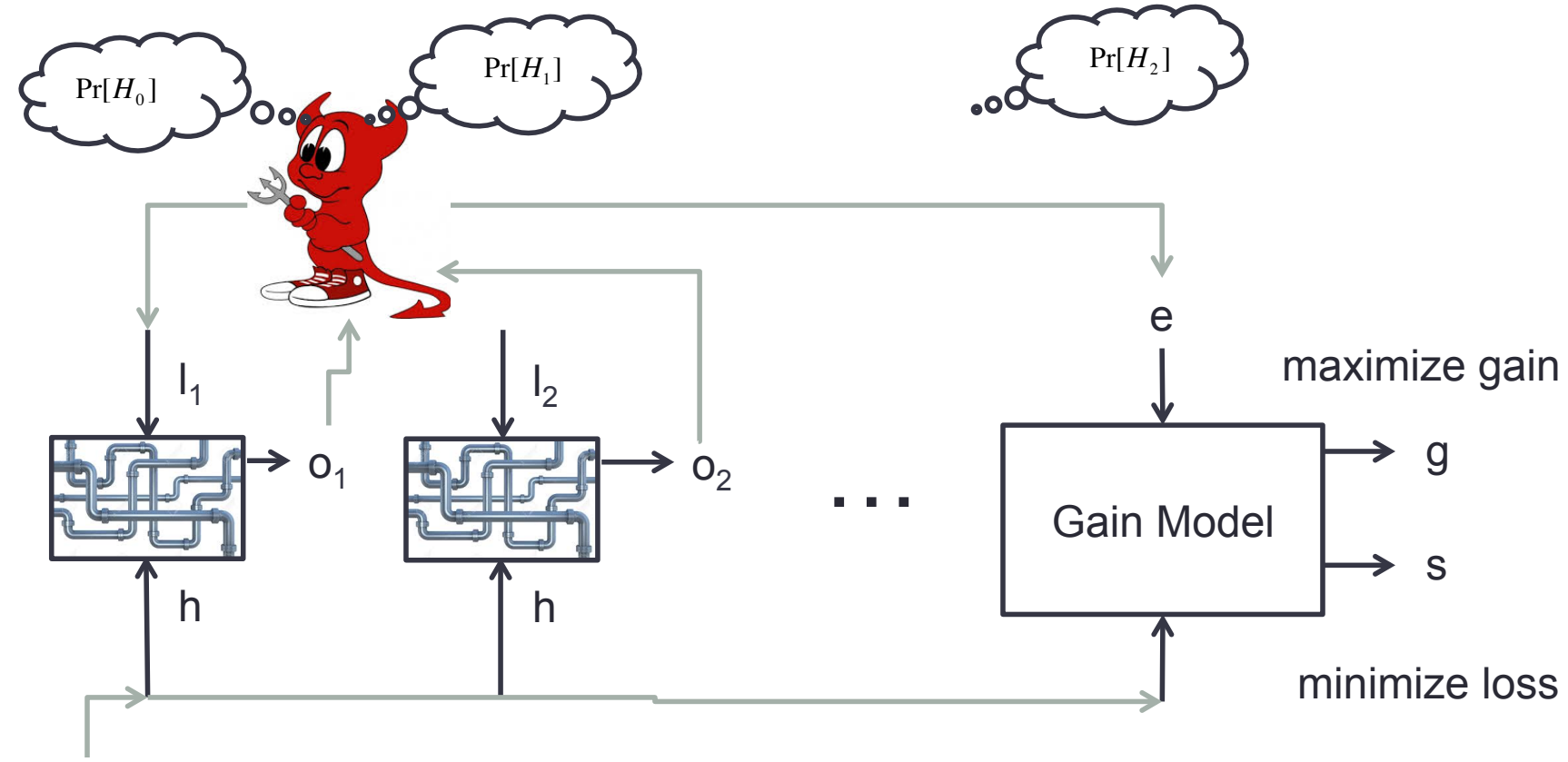

gain(c=0.15)
gain(c=0.10)
gain(c=0.05)
gain(c=0.00)
loss(c=0.15)
loss(c=0.10)
loss(c=0.05)
loss(c=0.00)
1
0.8
0.6
0.4
0.2
0
0
1
2
3
4
5
6
7
8
9
10
11
12

- Basics
- Channels as programs
  - Programs as channels
- Models for …
  - Adaptive adversaries
  - Time-varying secrets
  - Non-zero-sum games
  - **Active defenders, equilibrium**

Passive defender

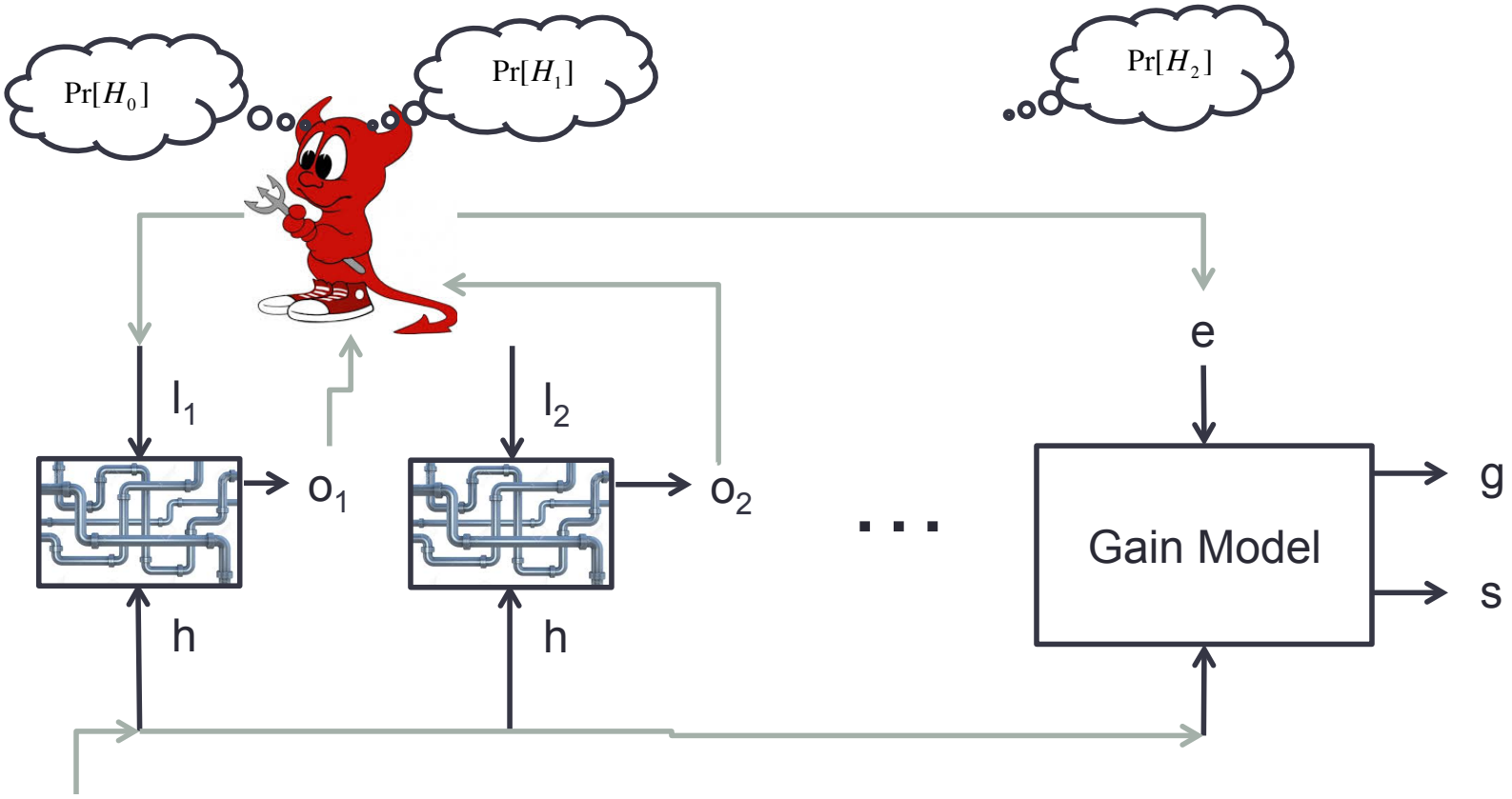


Active defender: simultaneous actions



$$\max_{l_1 \in L} \min_{h \in H} \cdots \quad ??? \quad \min_{h \in H} \max_{l_1 \in L} \cdots$$

Active defender: (Nash) equilibrium



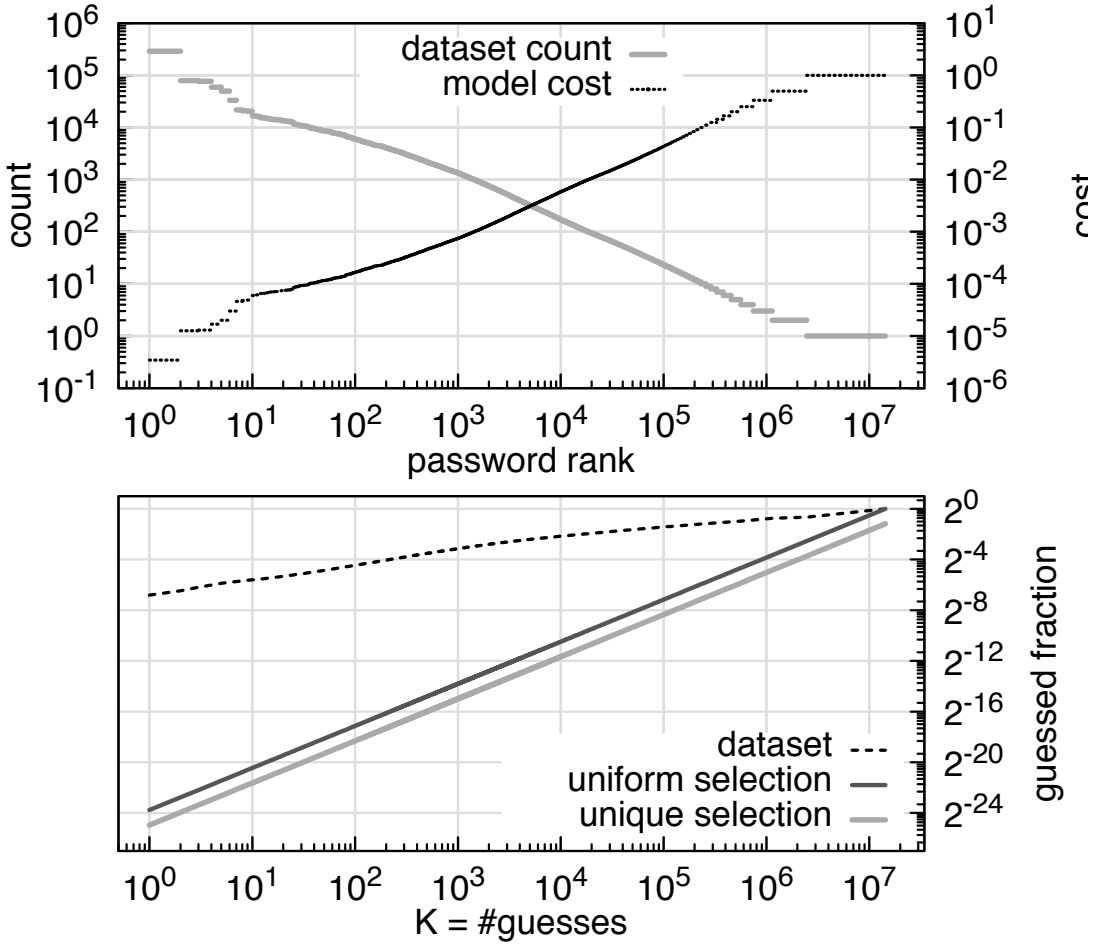
Low strategy γ*
High strategy λ*

Gain (γ*,λ*) ≥ Gain (γ,λ*)
Loss (γ*,λ*) ≤ Loss (γ*,λ)

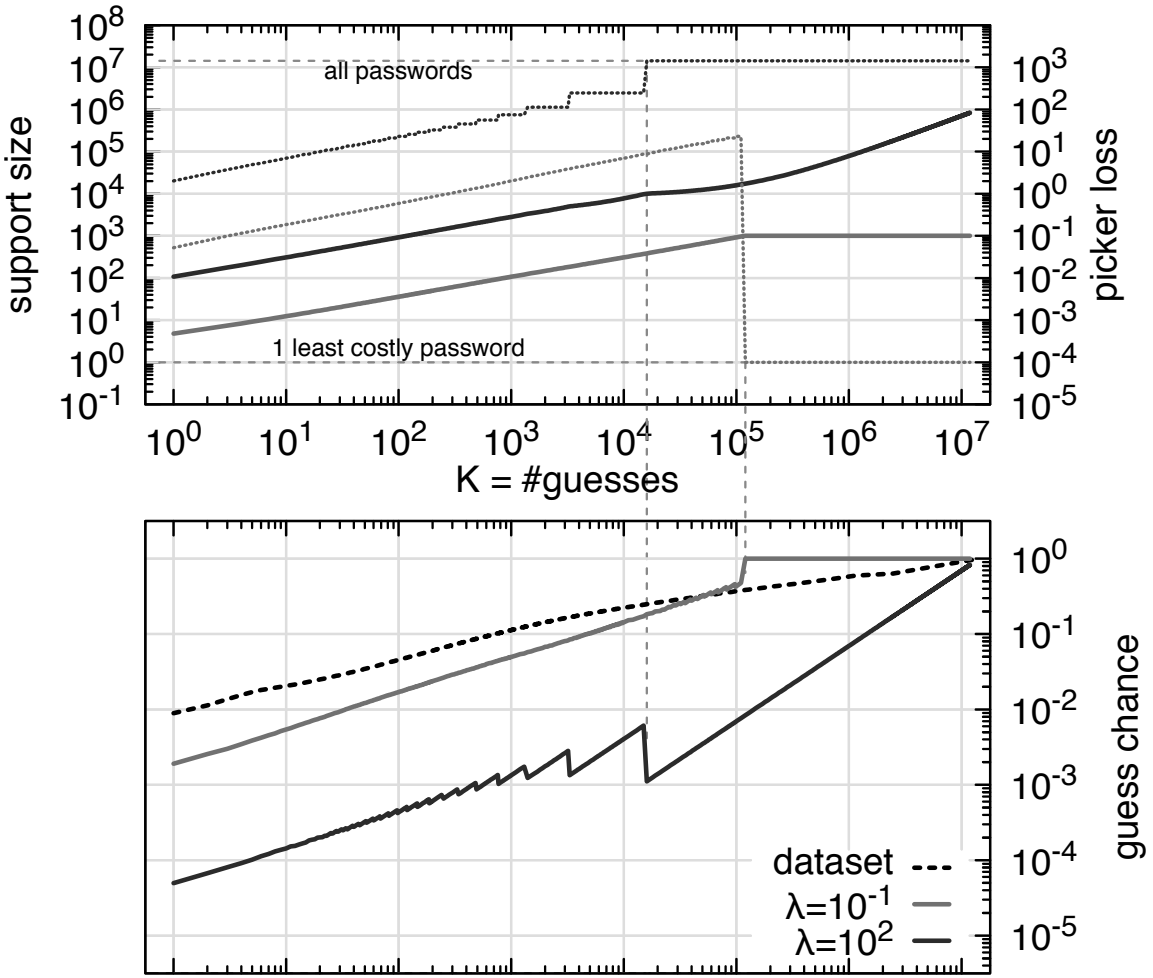
Picking vs. Guessing Passwords / Keys

- Picker picks password (or cryptographic key)
  - **Stronger (longer keys) passwords cost more**
- Guesser tries to guess
  - Online Game: cap of K guesses
  - Offline Game: no cap, but each guess incurs cost σ
- Picker loses λ if their secret is guessed (guesser gains γ)

Password cost model



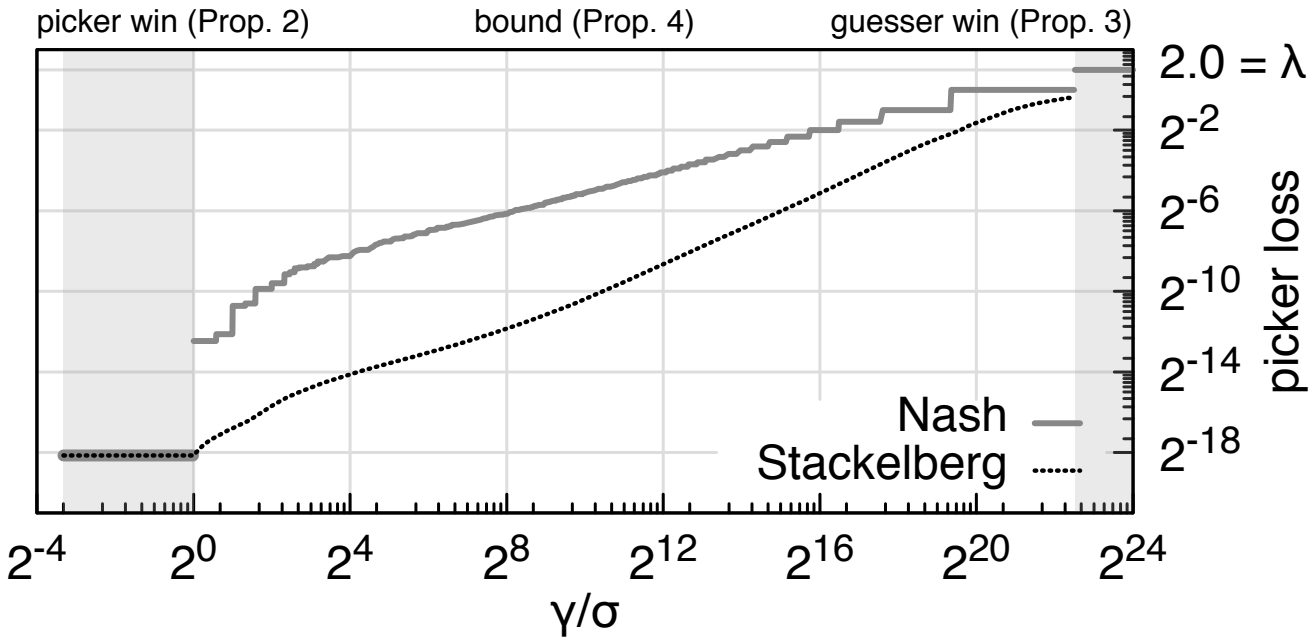
Password equilibria (capped guesses)



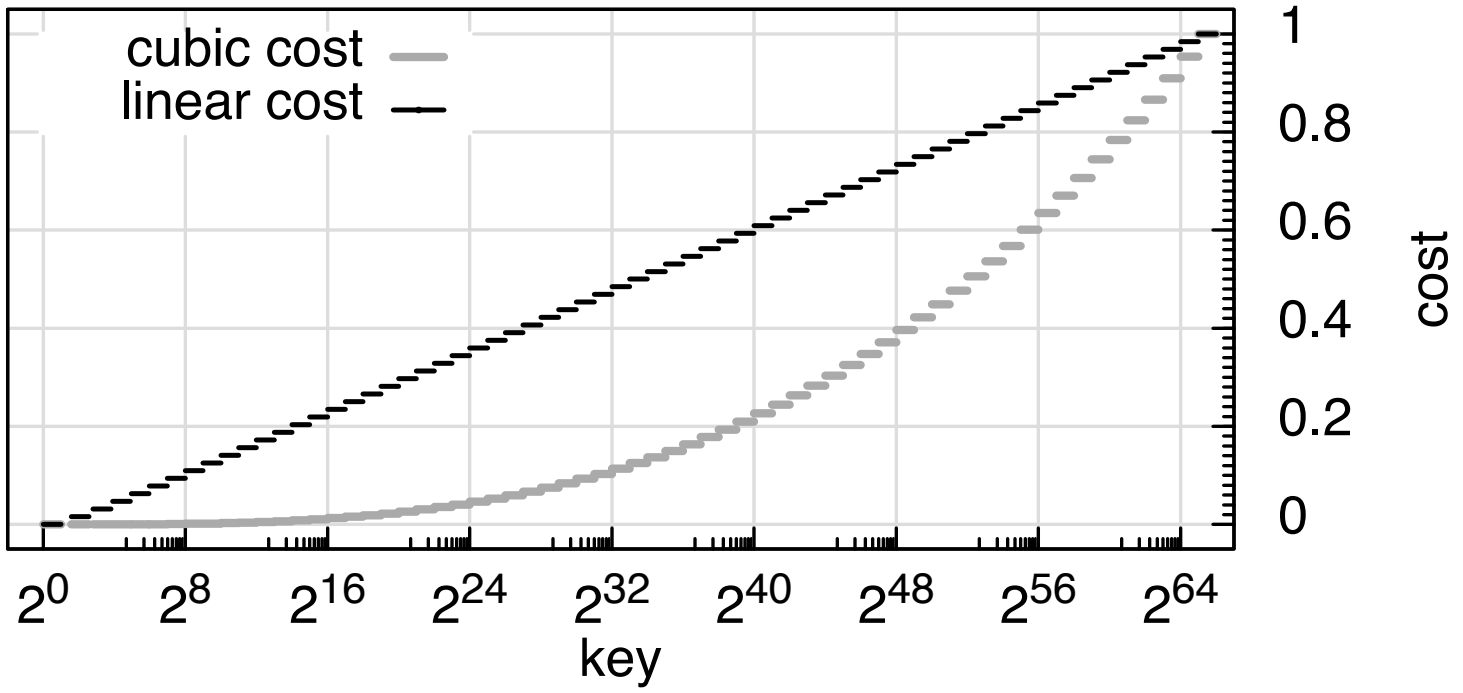
Picker strategy:
Uniformly pick password from J least costly passwords

Guesser strategy:
Guess in range of J least costly passwords, with probability inversely proportional to (picker's) cost.

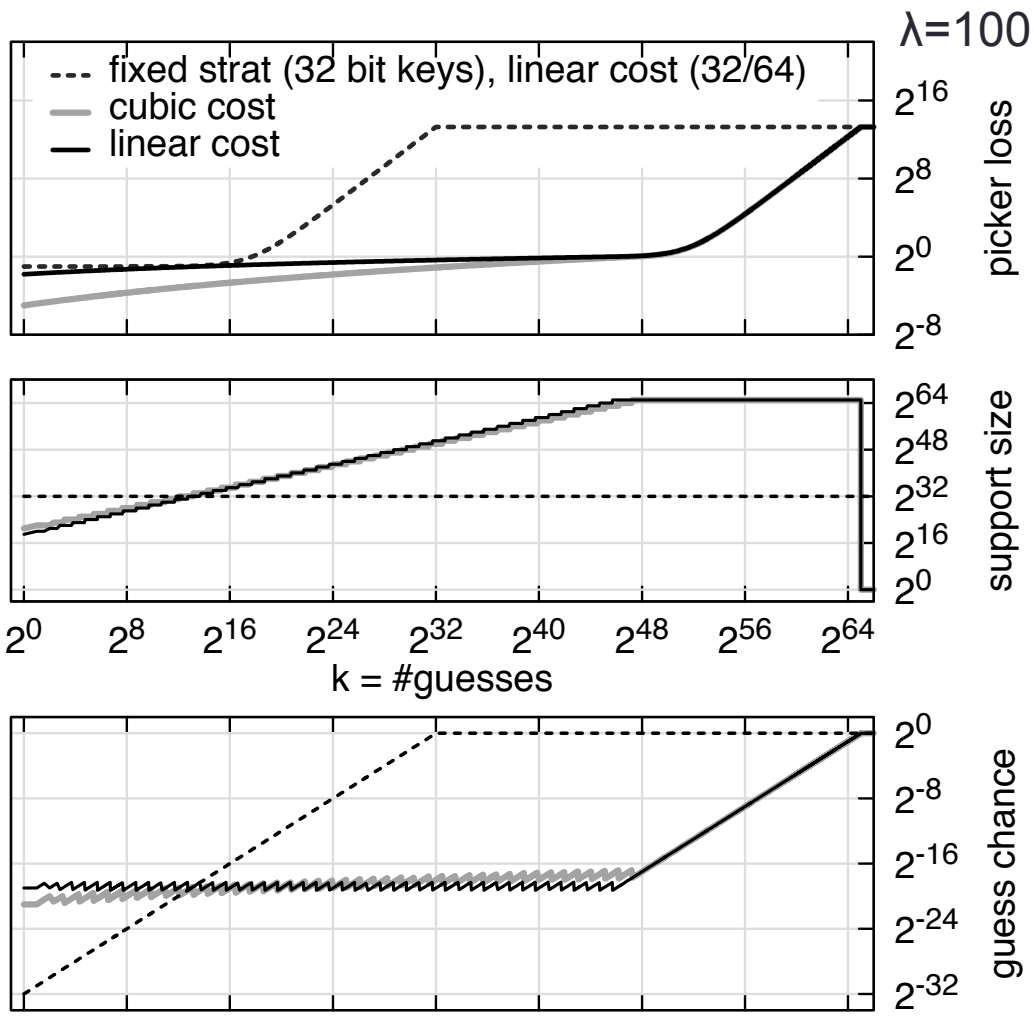
Password (costly guesses)



Key cost model



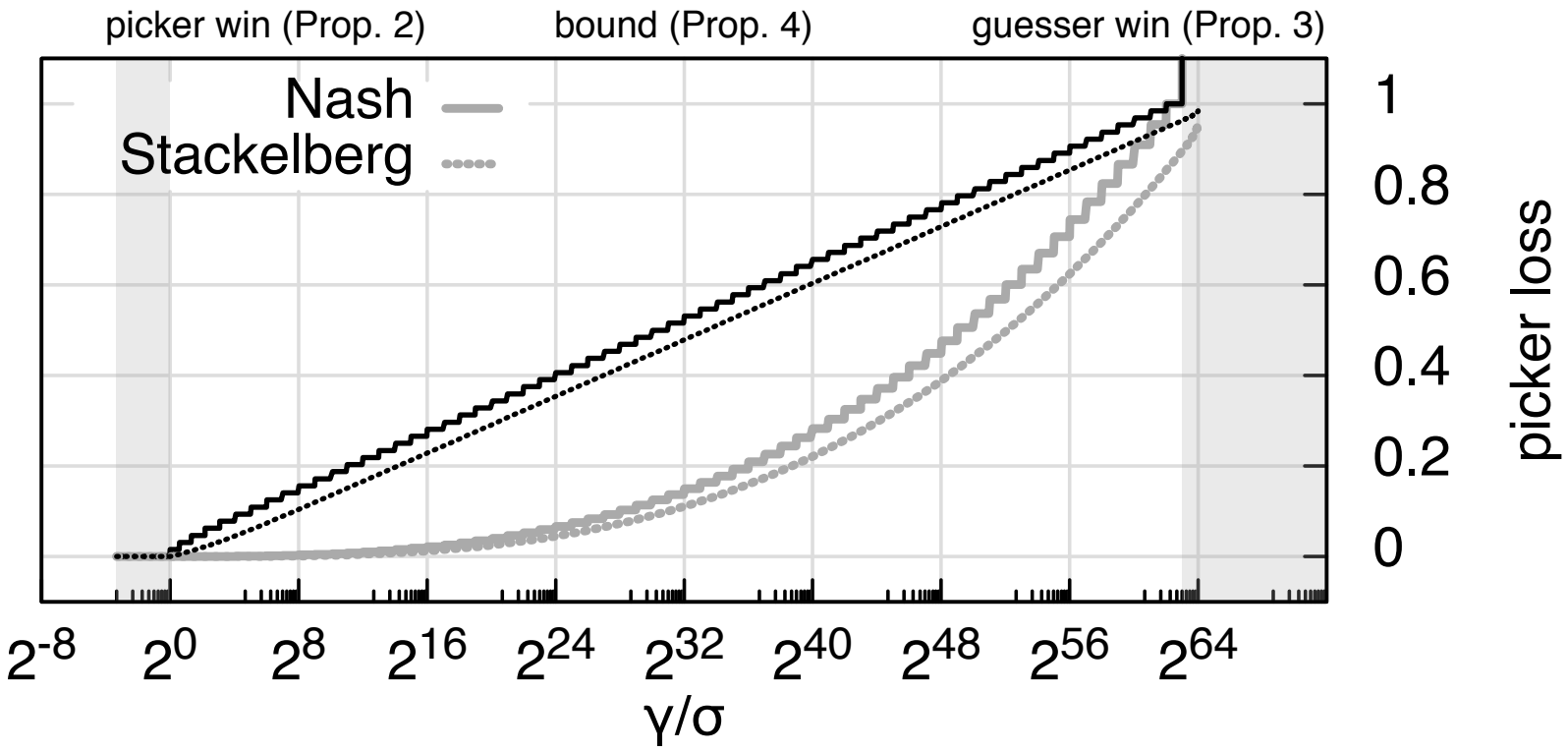
Key equilibria (capped guesses)



Picker strategy:
Uniformly pick keys from J least costly ones.

Guesser strategy:
Guess in range of J least costly keys, with probability inversely proportional to (picker's) cost.

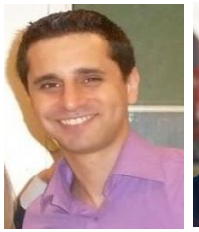
Key (costly guesses)



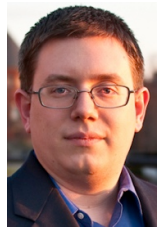
- Basics
- Channels as programs
  - Programs as channels
- Models for …
  - Adaptive adversaries
  - Time-varying secrets
  - Non-zero-sum games
  - Active defenders, equilibrium

- Models for …
  - [S&P14] Adaptive adversaries
  - [S&P14] Time-varying secrets
  - [FCS14] Non-zero-sum games
  - [CSF15]* Active defenders, equilibrium
- [CSF11,PLAS12,JCS13] Mechanism via bounding vulnerability
  - Probabilistic abstract interpretation
  - Simulatable enforcement mechanism
- http://piotr.mardziel.com
- piotrm@gmail.com

Michael Hicks
UMD



Mário Alvim
UFMG, Brazil



Michael Clarkson
Cornell

Mudhakar Srivatsa
IBM TJ Watson

Arman Khouzani
Queen Mary

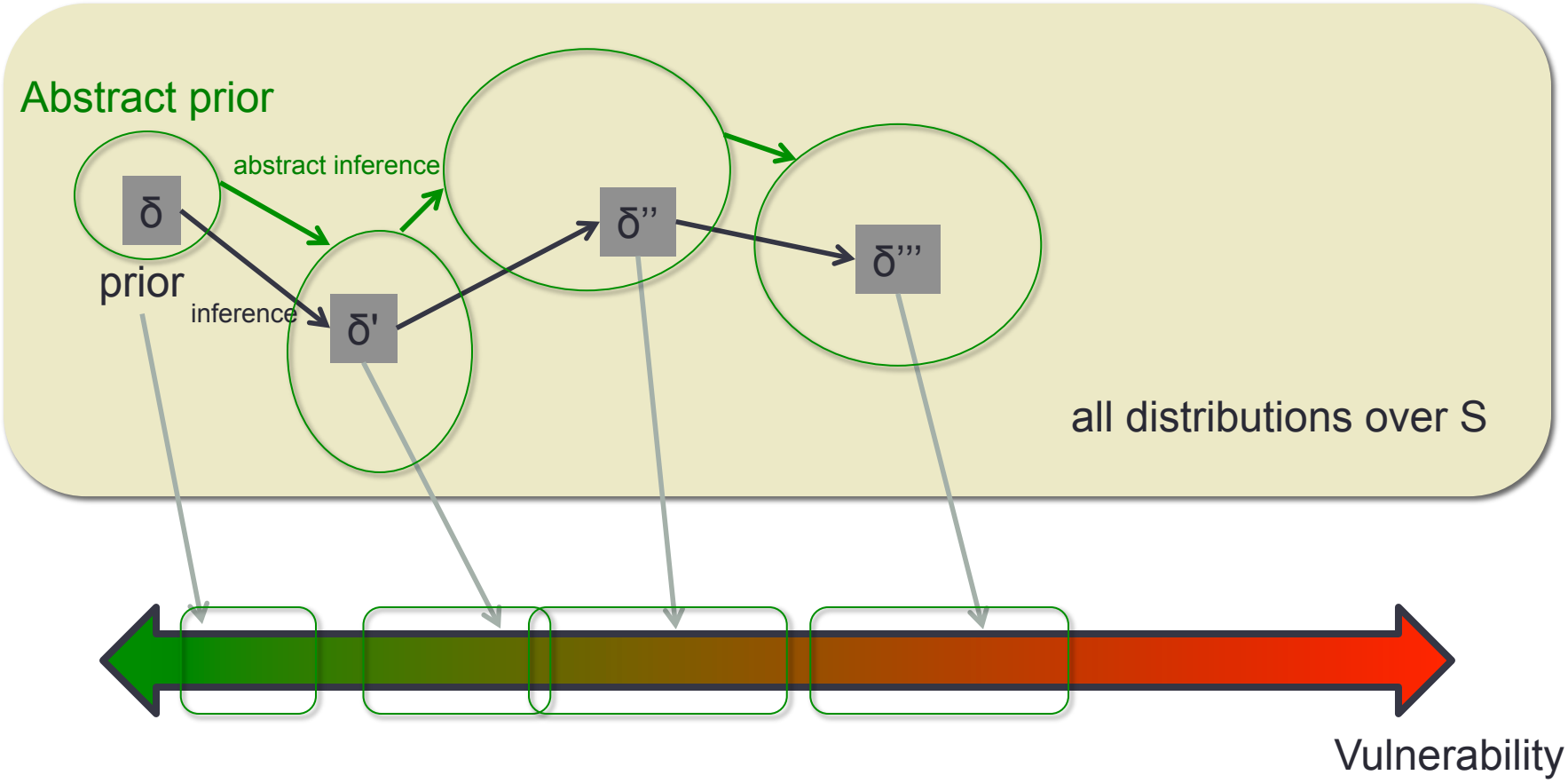
Carlos Cid
Royal Holloway

Stephen Magill
Galois

Michael Hicks
UMD

Mudhakar Srivatsa
IBM TJ Watson

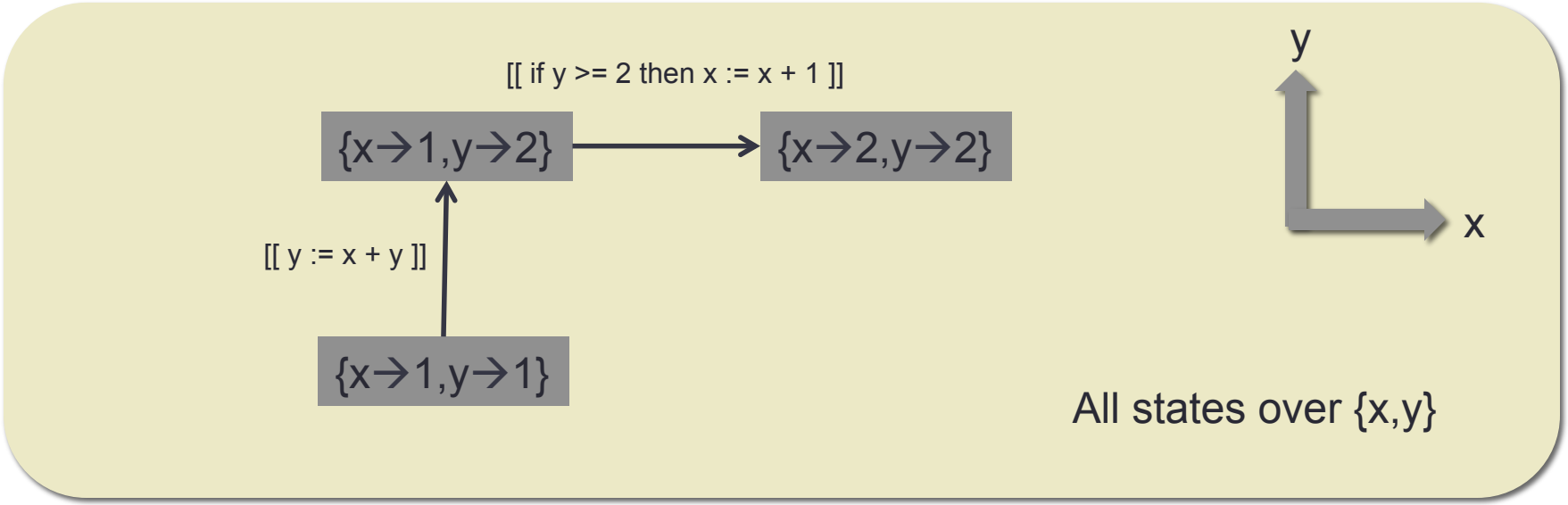
Probabilistic Abstract Interpretation



Concrete Interpretation

(Program) States σ : Variables → Integers

Concrete semantics: [[Stmt]] : States → States



Abstract Interpretation

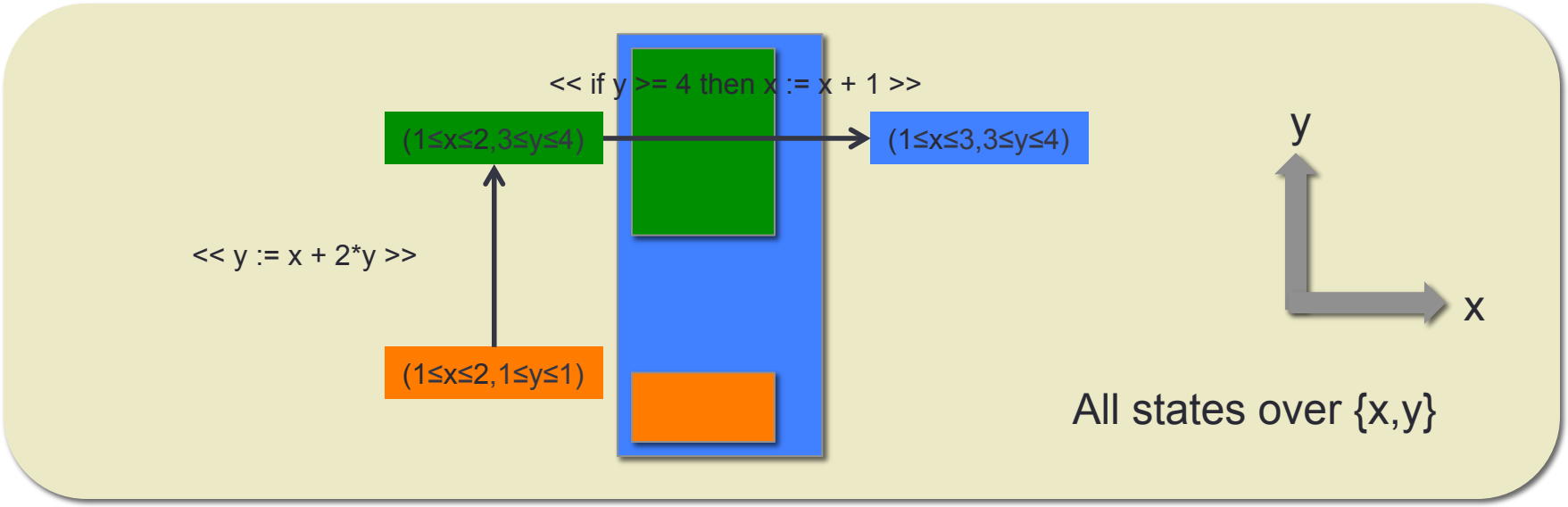
Abstract Program States AbsStates

Concretization: γ(P) := { σ s.t. P(σ) }

Abstract Semantics: << Stmt >> : AbsStates → AbsStates

Example: intervals

- Predicate P is a closed interval on each variable
- γ(1≤x≤2, 1≤y≤1) = all states that assign x between 1 and 2, and y = 1



Abstract Interpretation

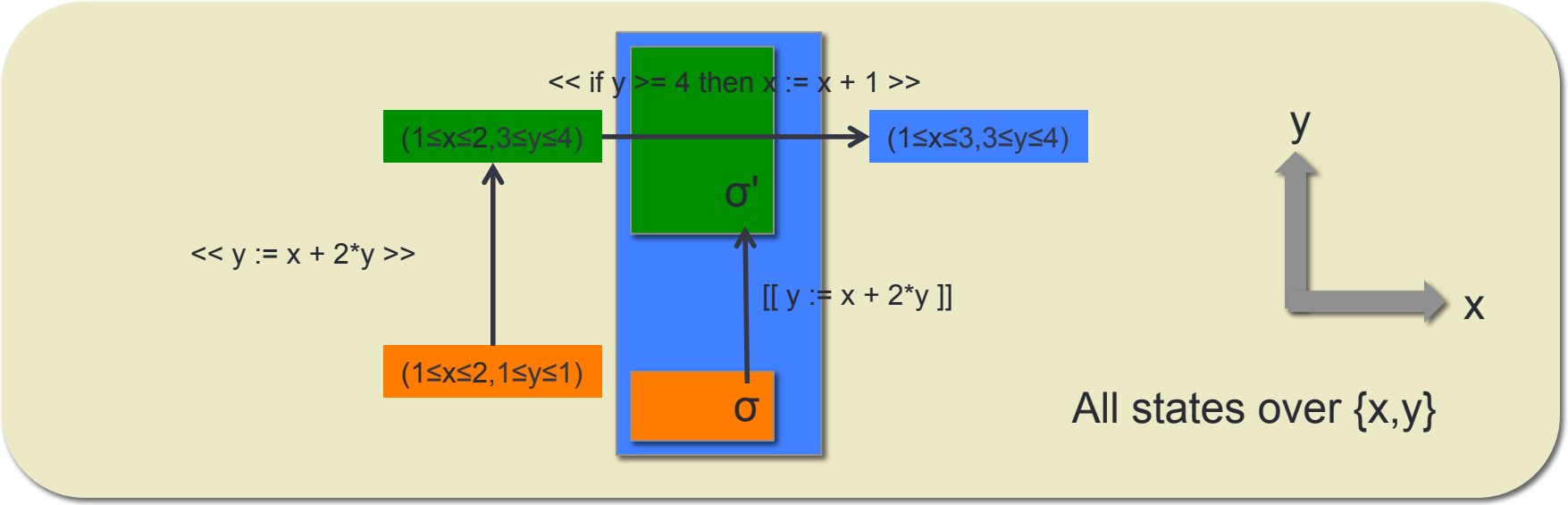
Abstract Program States AbsStates

Concretization: γ(P) := { σ s.t. P(σ) }

Abstract Semantics: << Stmt >> : AbsStates → AbsStates

Example: intervals

- Predicate P is a closed interval on each variable
- γ(1≤x≤2, 1≤y≤1) = all states that assign x between 1 and 2, and y = 1



Probabilistic Interpretation

- Concrete
- Abstraction
  - Abstract semantics

Concrete Probabilistic Semantics

- (sub)distributions δ : States → [0,1]

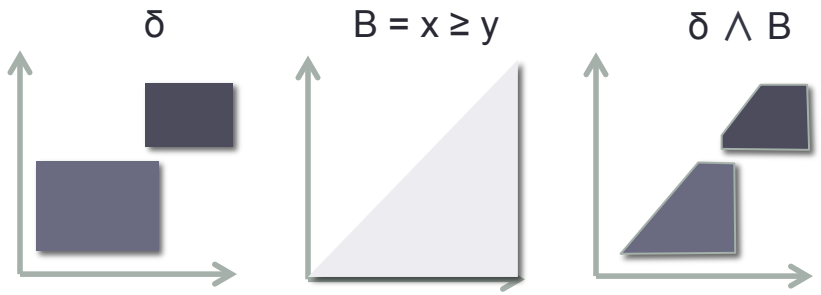
- Semantics
  - $[\![skip]\!]\delta$ $= \delta$
  - $[\![S_1; S_2]\!]\delta$ $= [\![S_2]\!]([\![S_1]\!]\delta)$
  - $[\![\text{if } B \text{ then } S_1 \text{ else } S_2]\!]\delta$ $= [\![S_1]\!](\delta \wedge B) + [\![S_2]\!](\delta \wedge \neg B)$
  - $[\![\text{pif } p \text{ then } S_1 \text{ else } S_2]\!]\delta$ $= [\![S_1]\!](p*\delta) + [\![S_2]\!]((1-p)*\delta)$
  - $[\![x := E]\!]\delta$ $= \delta[x \mapsto E]$
  - $[\![\text{while } B \text{ do } S]\!]$ $= \mathit{lfp}\ (\lambda F.\ \lambda\delta.\ F([\![S]\!](\delta \mid B)) + (\delta \mid \neg B))$

  - $p*\delta$ – scale probabilities by p
    - $p*\delta := \lambda\sigma.\ p*\delta(\sigma)$
  - $\delta \wedge B$ – remove mass inconsistent with B
    - $\delta \wedge B := \lambda\sigma.$ if $[\![B]\!]\sigma$ = true then $\delta(\sigma)$ else 0
  - $\delta_1 + \delta_2$ – combine mass from both
    - $\delta_1 + \delta_2 := \lambda\sigma.\ \delta_1(\sigma) + \delta_2(\sigma)$
  - $\delta[x \mapsto E]$ – transform mass

Subdistribution operations

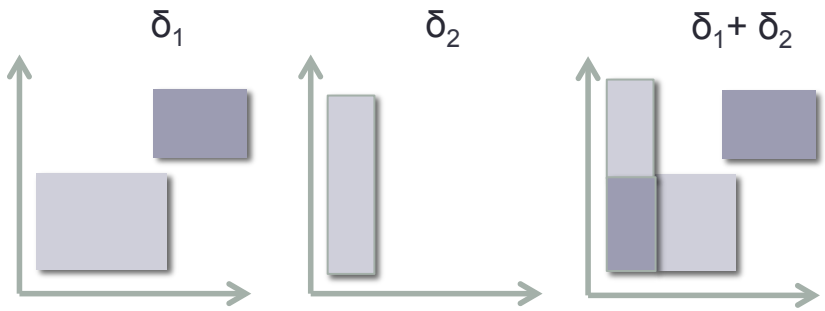
$\delta \wedge B$ – remove mass inconsistent with B

$\delta \wedge B = \lambda\sigma.$ if $[\![B]\!]\sigma$ = true then $\delta(\sigma)$ else 0

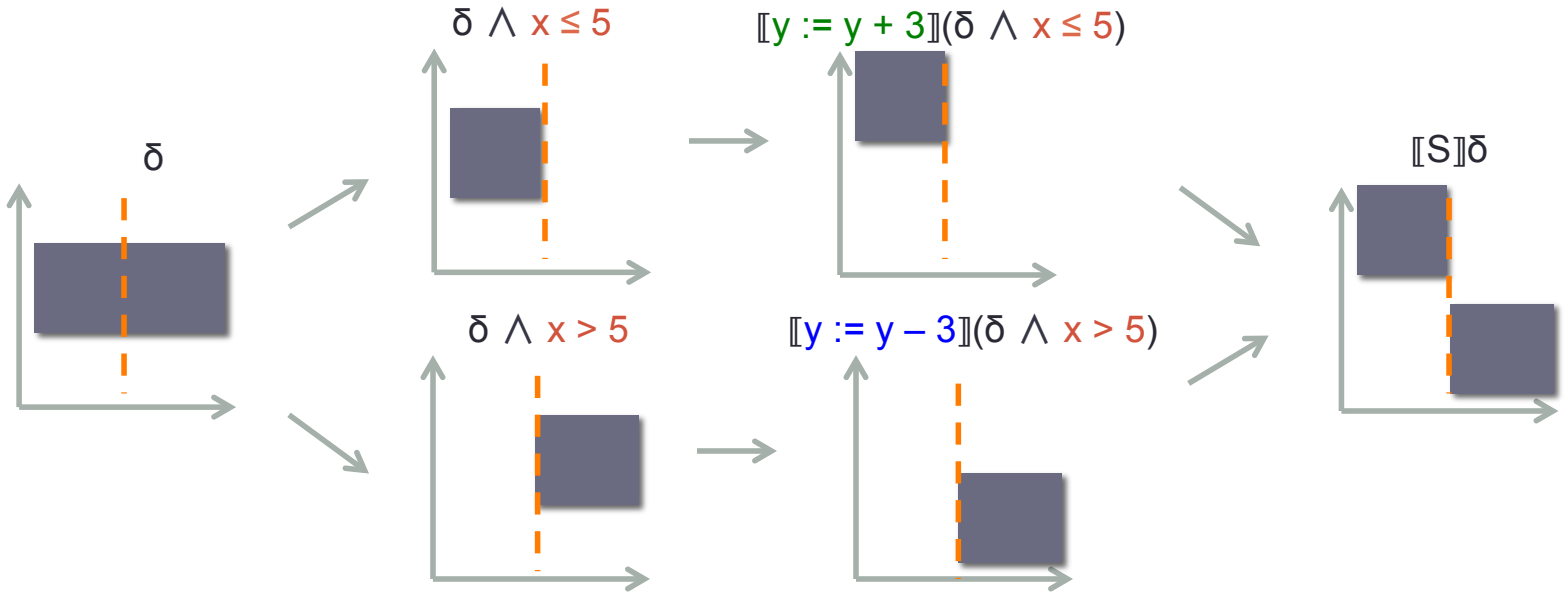


$\delta_1 + \delta_2$ – combine mass from both

$\delta_1 + \delta_2 = \lambda\sigma.\ \delta_1(\sigma) + \delta_2(\sigma)$

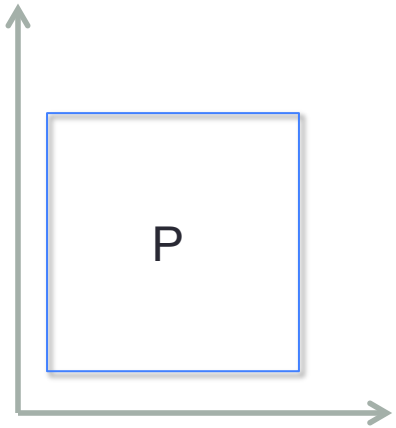


$[\![\text{if } x \le 5 \text{ then } y := y + 3 \text{ else } y := y - 3]\!]\delta = [\![y := y + 3]\!](\delta \wedge x \le 5) + [\![y := y - 3]\!](\delta \wedge x > 5)$



Subdistribution Abstraction

Subdistribution Abstraction: Probabilistic Polyhedra



$V(\delta) = \max_{\sigma} \delta(\sigma)$

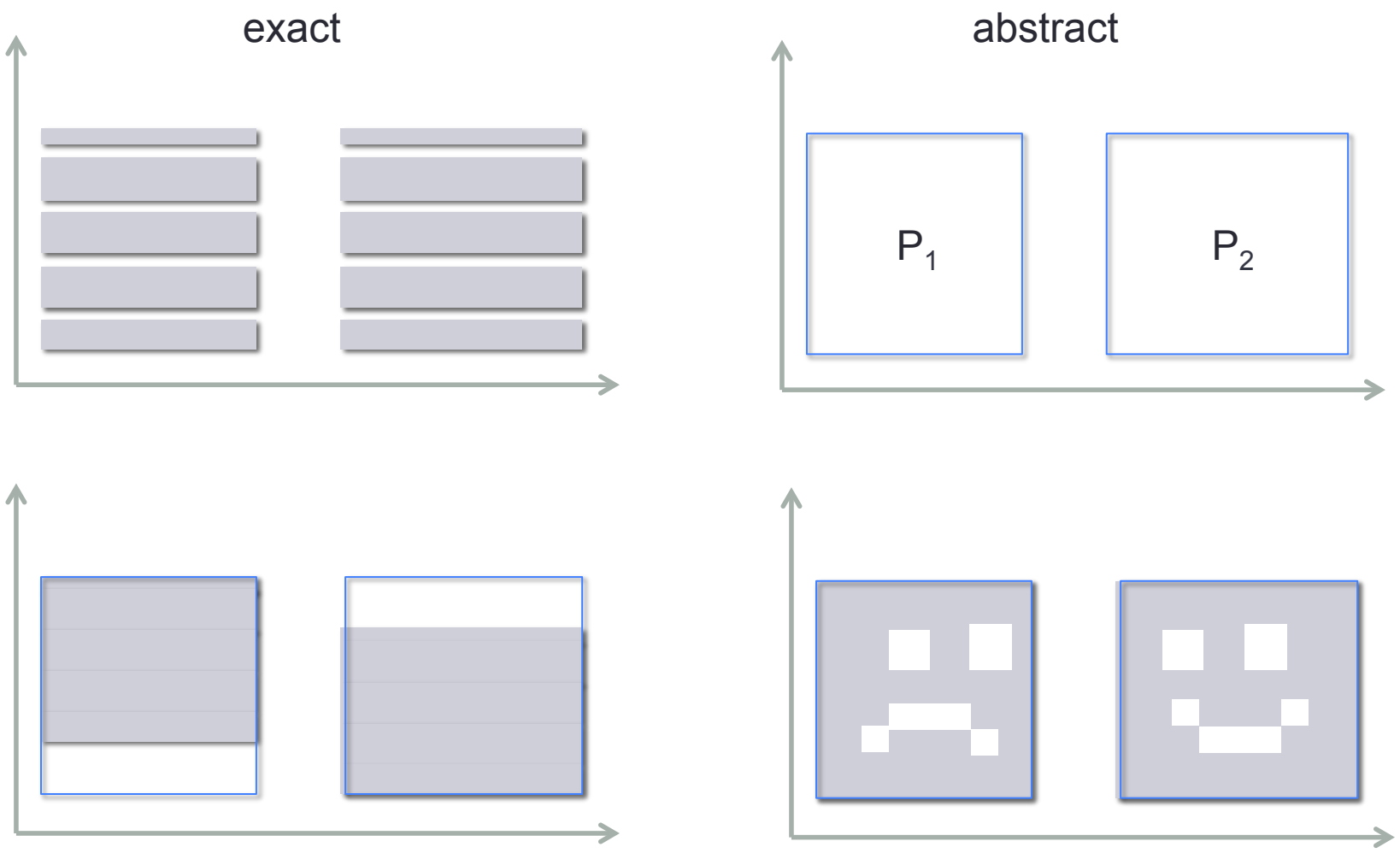
Region of program states (polyhedron)

+ upper bound on probability of each possible state in region
+ upper bound on the number of (possible) states
+ upper bound on the total probability mass (useful)

+ also **lower bounds** on the above

$Pr[A \mid B] = Pr[A \cap B] / \mathbf{Pr[B]}$

Abstraction imprecision



Probabilistic Abstract Interpretation

Define

<<S>> P

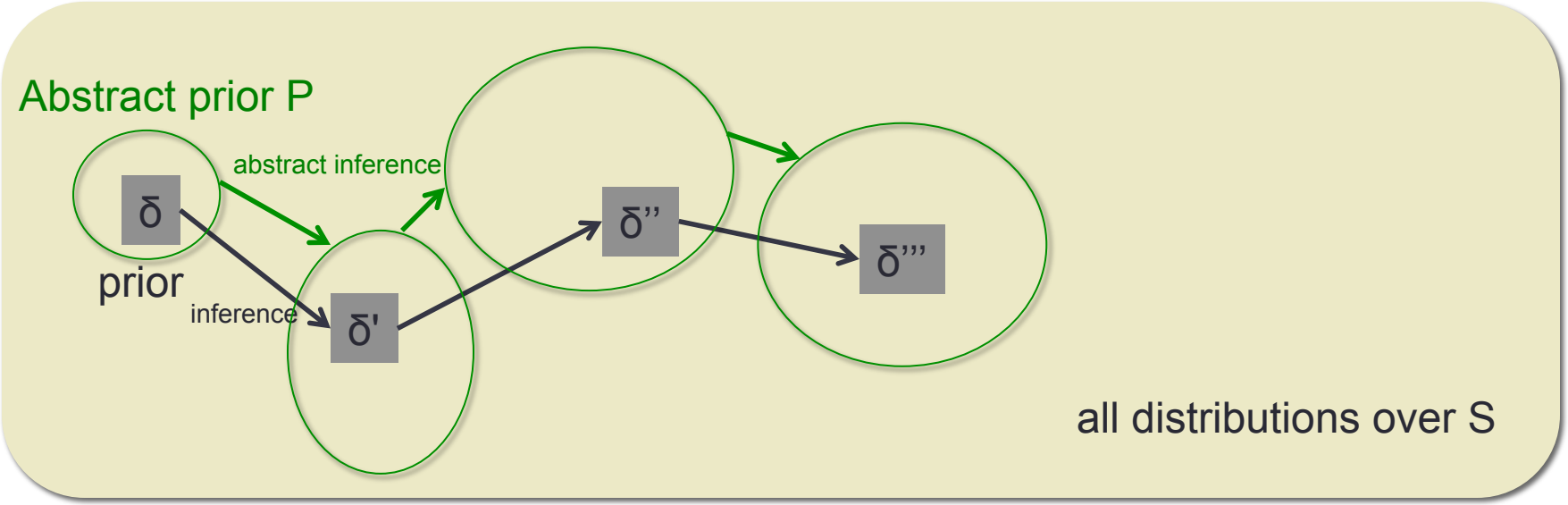
Soundness: if δ ∈ γ(P) then ⟦S⟧δ ∈ γ (<<S>>P)

Abstract versions of subdistribution operations

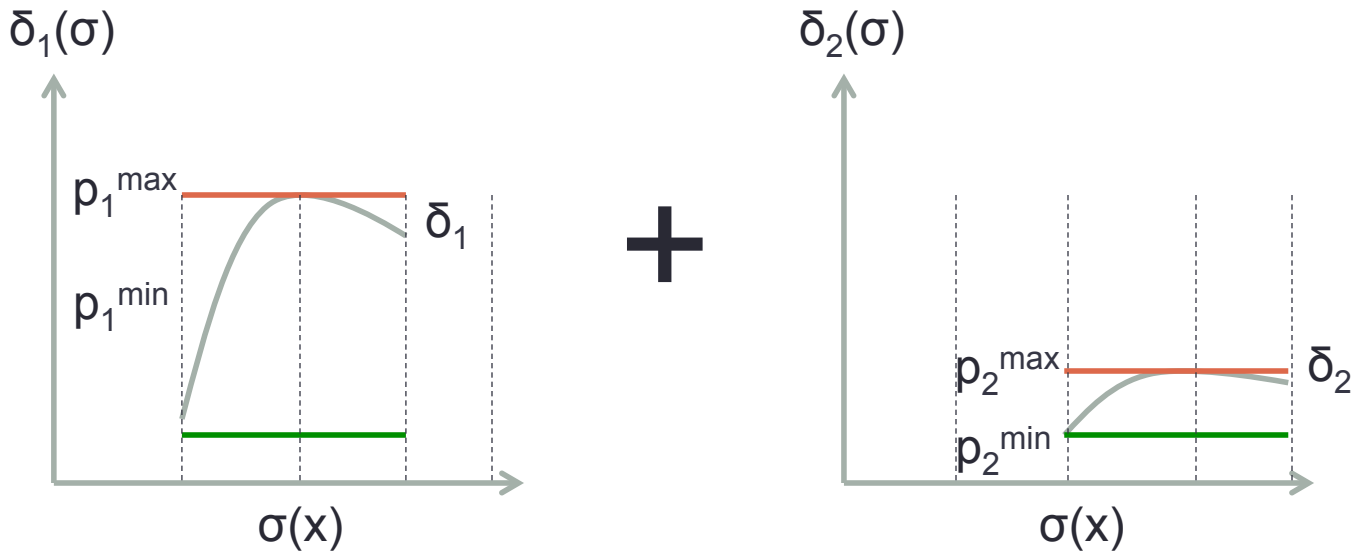
$P_1 + P_2$

$P \wedge B$

p*P

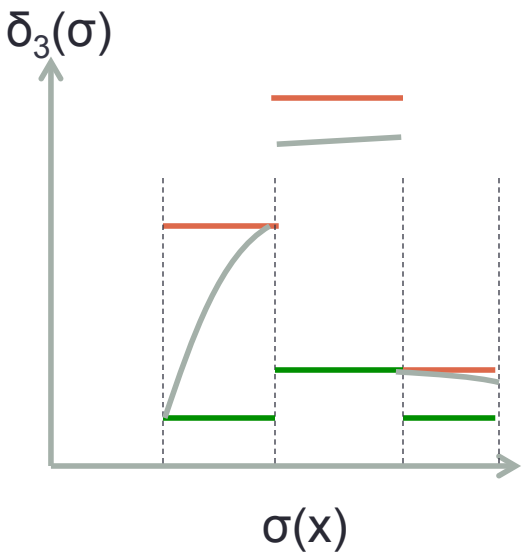


Example abstract operation



$\delta_3 := \delta_1 + \delta_2$

$\{P_3, P_4, P_5\} = \{P_1\} + \{P_2\}$



Conditioning

- Conditioning
  - Concrete $\delta \mid B := \frac{1}{\|\delta \wedge B\|} * (\delta \wedge B)$
  - Abstract: $P \mid B := \frac{1}{\|P \wedge B\|} * (P \wedge B)$

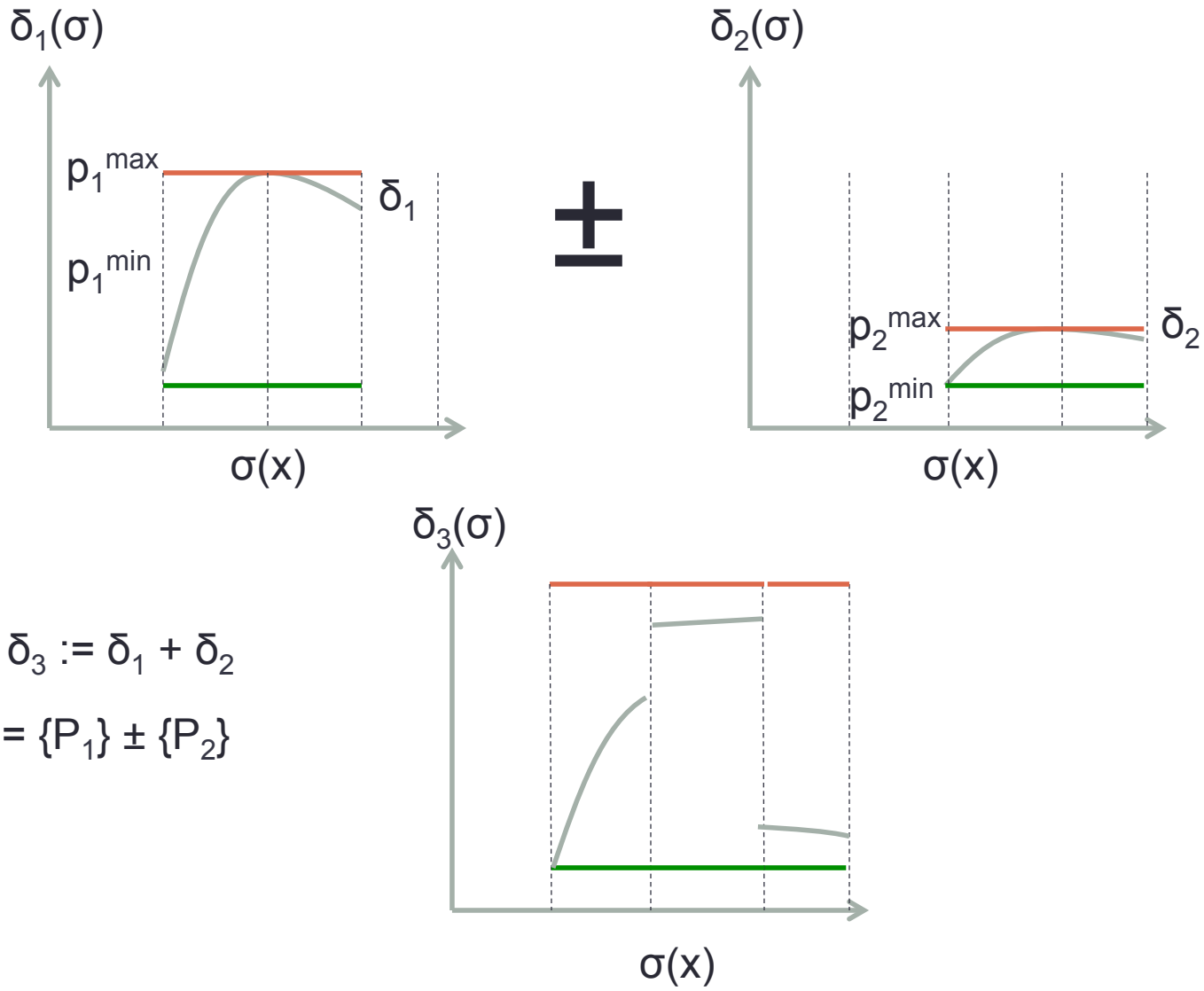
$$\left\|\{P_i\}_i\right\| := \sum_i m_i^{\min}$$

Lower bound on total mass

Simplify representation

- Limit number of probabilistic polyhedra
  - $P_1 \pm P_2$ - merge two probabilistic polyhedra into one
  - Convex hull of regions, various counting arguments

Add and simplify



$\delta_3 := \delta_1 + \delta_2$

$\{P_3\} = \{P_1\} \pm \{P_2\}$

Primitives for operations

- Need to
  - Linear Model Counting: count number of integer points in a convex polyhedra
  - Integer Linear Programming: maximize a linear function over integer points in a polyhedron

Probabilistic Abstract Interpretation