Verified Learning Without Regret

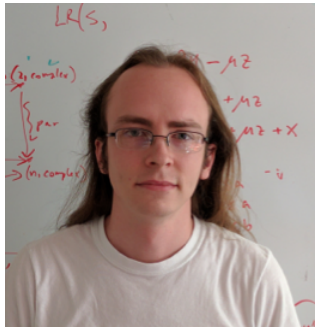
A Mechanized Proof of the Multiplicative Weights Update Algorithm

Gordon Stewart

Assistant Professor, EECS

Ohio University

Sam Merten
(PhD)



Alex Bagnall
(MSc)

Software Is Hard!

In 2014 alone…

- April 2014: HeartBleed OpenSSL bug
  - buffer overread due to missing bounds check
  - 17% of servers running TLS affected
- September 2014: Shellshock
  - Bash – unintended command execution
  - undiscovered for 25 years (!)
- October 2014: POODLE
  - TLS: for interoperability, fall back to SSL 3.0
  - … exposing a padding oracle attack

2000s:

Toyota Unintended Acceleration

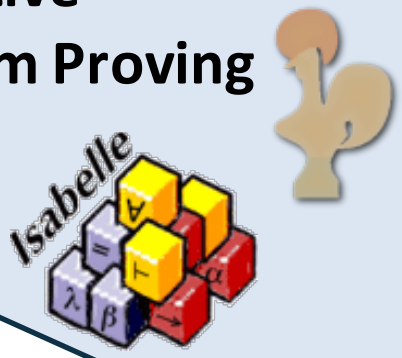
- lives lost…probably due to software
- $1.2b settlement

What Do We Do About It?

Expressivity

HOL
Interactive Theorem Prover

Interactive Theorem Proving



MY WEAPON OF CHOICE

NuSMV

CBMC About CBMC

Software Model Checking

SPARK Toolset

Static Analysis

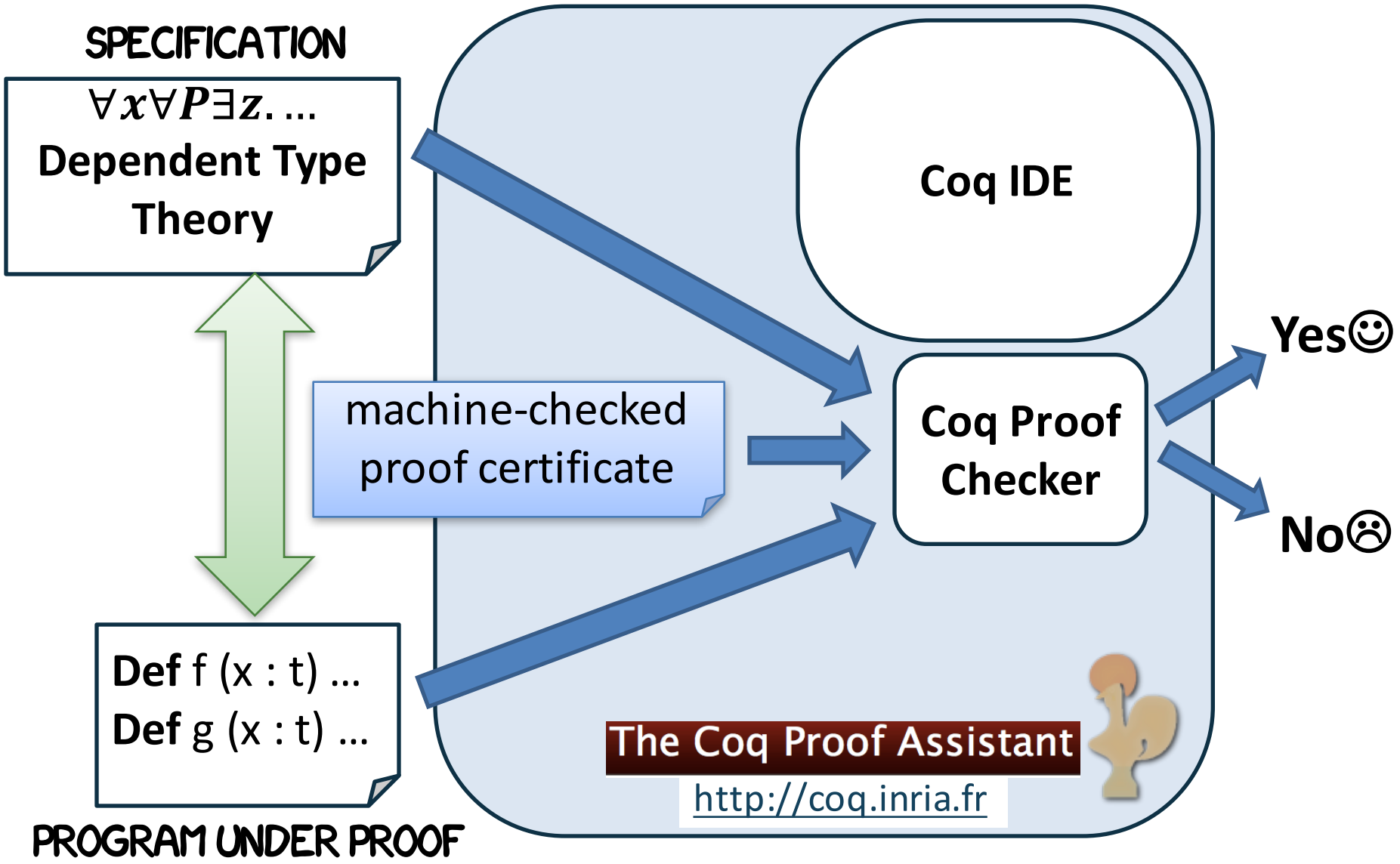
Type Systems

$\Gamma \vdash e_1 + e_2 : \tau$

Astrée

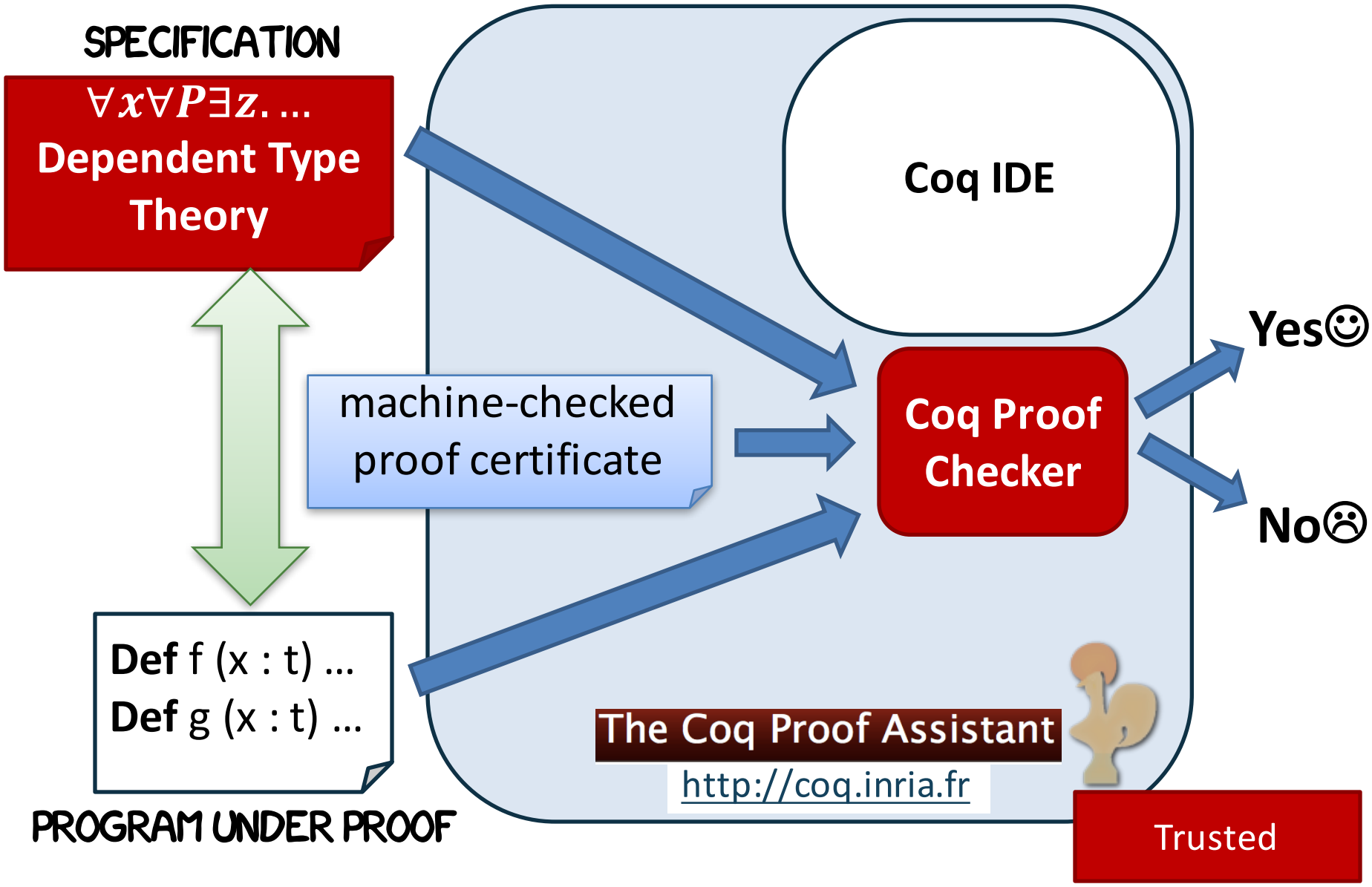
User Interaction

Interactive Theorem Proving



Trusted Computing Base

Which of these pieces do we need to trust?



Theorem Proving In Practice

State Context Goal Retract Undo Next Use Goto Qed Home Find Info Command Prooftree Interrupt Restart Help

```
Hypothesis (HAgt0 : 0 < A).

(** The bound is proved assuming there exist
    real numbers [A] and [B] such that for any state [t],
    [Phi t] is bounded on the left by
    [Cost t / A] and bounded on the right by [B * Cost t]. *)
Hypothesis AB_bound_Phi :
  forall t : sT, Cost t / A <= Phi t <= B * Cost t.

(** Under the conditions stated above, the
    Price of Stability of any potential game is
    at most [A * B]. (For games in which the
    PNE is unique, this bound gives a bound on
    the Price of Anarchy as well.) *)
Lemma PoS_bounded (t0 : sT) (PNE_t0 : PNE t0) :
  PoS t0 <= A * B.
Proof.
  set (tN := Phi_minimizer t0).
  generalize (minimal_Phi_minimizer t0); move/forallP=> HtN.
  case: (andP (AB_bound_Phi tN))=> H3 H4; rewrite /PoS.
  set (tStar := arg_min optimal Cost t0).
  move: (HtN tStar)=> H5.
  case: (andP (AB_bound_Phi tStar))=> H6 H7.
  rewrite ler_pdivr_mulr; last by apply: Cost_pos.
  apply: ler_trans.
```

```
1 subgoal, subgoal 1 (ID 54)

T : game
X0 : Moveable T
X : Potential
Cost_pos : forall t : {ffun 'I_(numplayers T) -> T}, 0 < Cost t
A, B : rty
HAgt0 : 0 < A
AB_bound_Phi : forall t : {ffun 'I_(numplayers T) -> T},
           Cost t / A <= Phi t <= B * Cost t
t0 : {ffun 'I_(numplayers T) -> T}
PNE_t0 : PNE t0
tN := Phi_minimizer t0
  : finfun_of_finType (ordinal_finType (numplayers T)) T
============================
 PoS t0 <= A * B
```

PROOF WINDOW

- current proof state
- including hypotheses & goals

PROOF SCRIPT

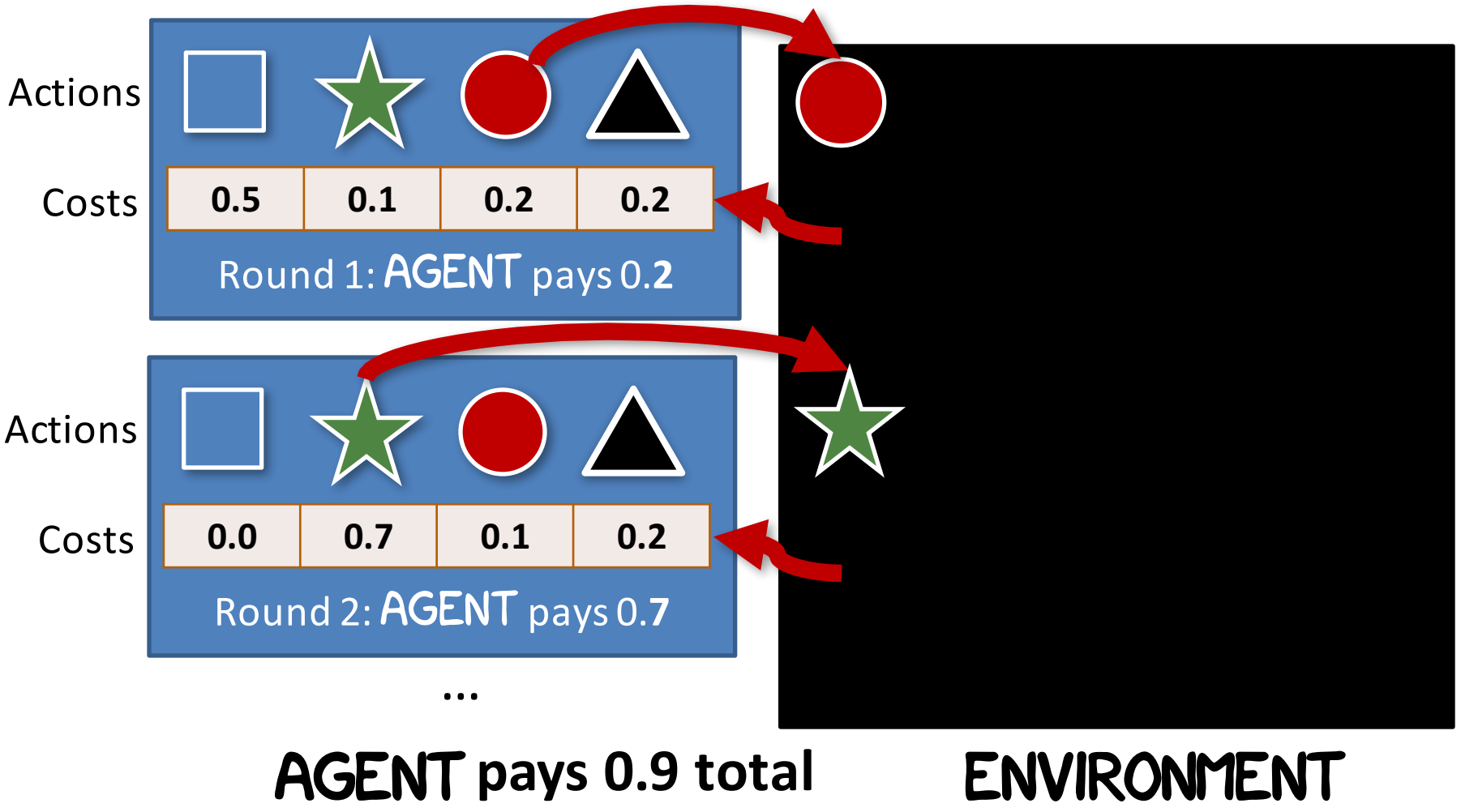
Used to construct independently checkable proof object

MULTIPLICATIVE WEIGHTS UPDATE (MWU)

The Setting

Learning on-line, in uncertain environments

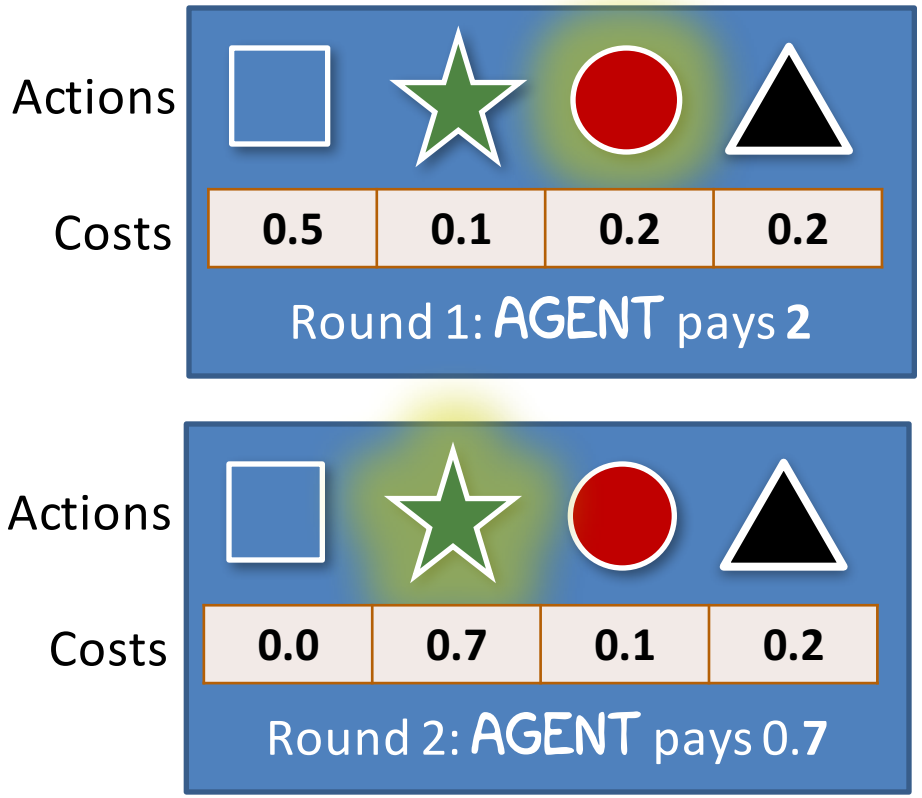
(For the remainder, I'll assume costs in range [0, 1].)



Regret

A learning algorithm is ***bounded regret*** if it has constant expected cost wrt. the best fixed action, as the number of iterations $T \to \infty$.

$$Regret(A) \coloneqq \mathbf{E}[C_{tot}(A)] - \min_a C_{tot}(a)$$



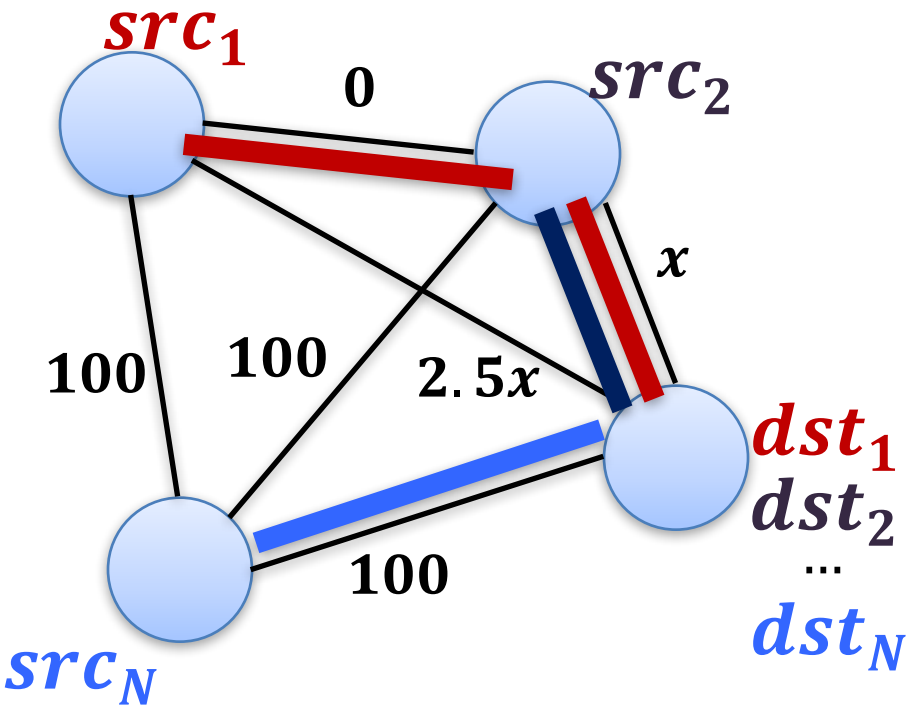
AGENT pays 0.9 total

$C_{tot}($  $) = 0.3$

Regret = 0.9-0.3 = 0.6

Why (Verify) Regret?

Bounded-regret algorithms: natural ***distributed*** execution semantics yielding ***approximate equilibria***



DISTRIBUTED ROUTING GAME

MACHINE-VERIFIED WHOLE-SYSTEM PERFORMANCE GUARANTEES

AGENT 1: Regret At Most ϵ

AGENT 2: Regret At Most ϵ

...

AGENT N: Regret At Most ϵ

ϵ-approximate CCE

APPROXIMATES

optimal configuration

The MWU Algorithm

- Associate to each action $\boldsymbol{a}$ weight $\boldsymbol{w(a)}$
- Choose actions by drawing from the distribution

$$p(a) = \frac{w(a)}{\sum_b w(b)}$$

- Update weights according to the following rule

$$w^{i+1}(a) = w^i(a) * (1 - \epsilon * c^i(a))$$

PARAMETER $\epsilon \in (0, ½]$

MORE -> LESS EXPLORATION

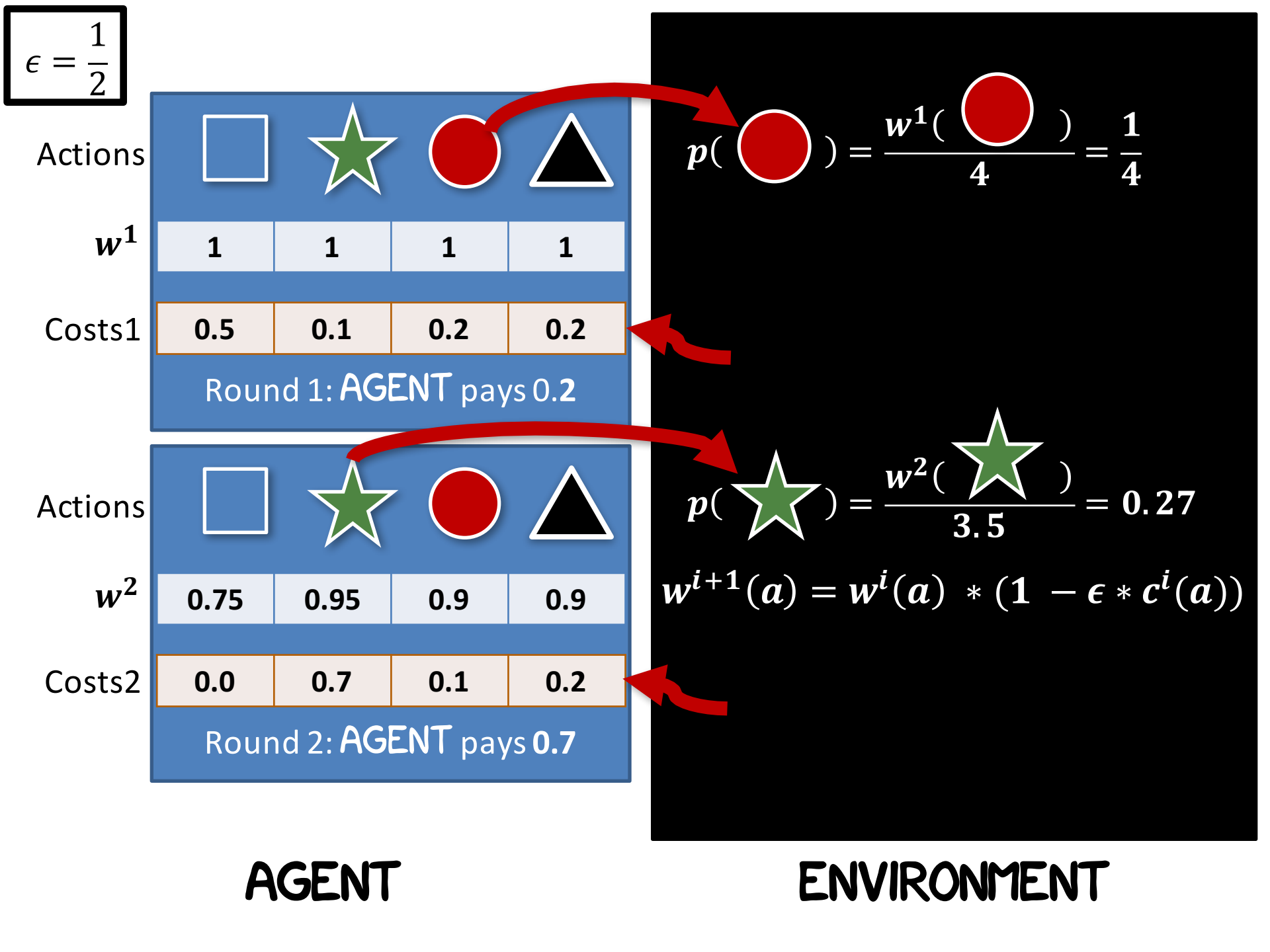
A Rose By Any Other Name…

- **"Combining Expert Advice"**
- **Winnow**
  - an algorithm for learning linear classifiers
  - [Littlestone '88]

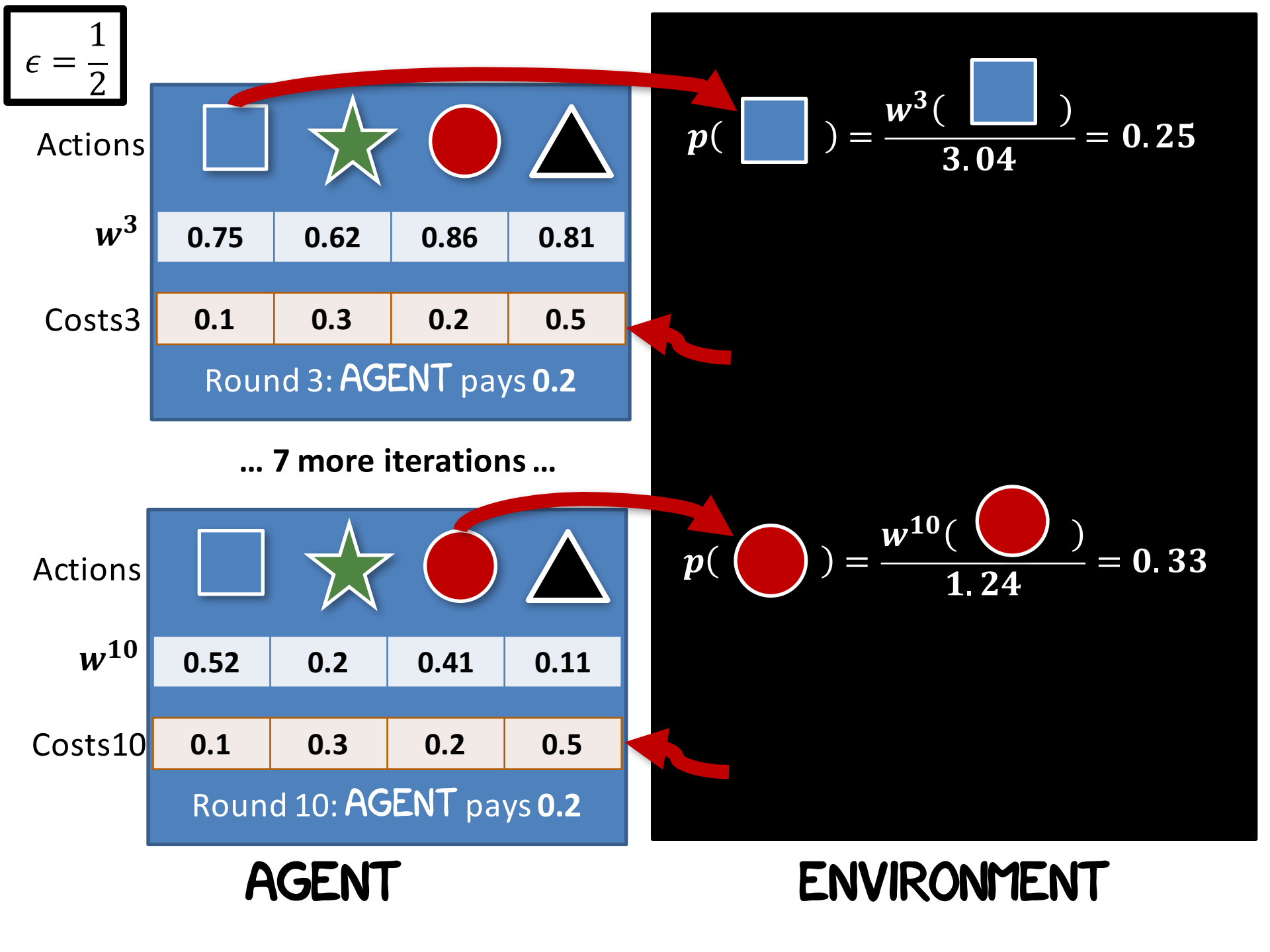
- **Weighted Majority Hedging**
  - Exponential update rule:

$$w^{i+1}(a) = w^{i}(a) * (1 - \epsilon^{c^{i}(a)})$$

- **AdaBoost / Boosting**
  - [Freund and Schapire '97]


$\epsilon = \frac{1}{2}$
Actions
w^1
1
1
1
1
Costs1
0.5
0.1
0.2
0.2
Round 1: AGENT pays 0.2
$p(\ ) = \frac{w^1(\ )}{4} = \frac{1}{4}$
Actions
w^2
0.75
0.95
0.9
0.9
Costs2
0.0
0.7
0.1
0.2
Round 2: AGENT pays 0.7
$p(\ ) = \frac{w^2(\ )}{3.5} = 0.27$
$w^{i+1}(a) = w^i(a) * (1 - \epsilon * c^i(a))$
AGENT
ENVIRONMENT


$\epsilon = \frac{1}{2}$
Actions
w^2
0.75
0.95
0.9
0.9
Costs2
0.0
0.7
0.1
0.2
Round 2: AGENT pays 0.7
$p(\ ) = \frac{w^2(\ )}{3.5} = 0.27$
Actions
w^3
0.75
0.62
0.86
0.81
Costs3
0.1
0.3
0.2
0.5
Round 3: AGENT pays 0.1
$p(\ ) = \frac{w^3(\ )}{3.04} = 0.25$
PENALIZED
AGENT
ENVIRONMENT


$\epsilon = \frac{1}{2}$
Actions
w^3
0.75
0.62
0.86
0.81
Costs3
0.1
0.3
0.2
0.5
Round 3: AGENT pays 0.2
$p(\square) = \frac{w^3(\square)}{3.04} = 0.25$
... 7 more iterations ...
Actions
w^{10}
0.52
0.2
0.41
0.11
Costs10
0.1
0.3
0.2
0.5
Round 10: AGENT pays 0.2
$p(\bigcirc) = \frac{w^{10}(\bigcirc)}{1.24} = 0.33$
AGENT
ENVIRONMENT

MWU Is Bounded Regret

Theorem: MWU is bounded regret.

$$\left(\mathbf{E}[C_{tot}(MWU)] - \min_a C_{tot}(a)\right) / T \leq \epsilon + \frac{\ln |A|}{\epsilon T}$$

EXPECTED TOTAL COST OF MWU

COST OF BEST FIXED ACTION

NUMBER OF STEPS

SIZE OF ACTION SPACE

Proof: Potential function $\Gamma^i = \sum_a w^i(a)$

Corollary:

$\frac{\mathbf{c} * \mathbf{ln}|\boldsymbol{A}|}{\boldsymbol{\epsilon}}$ steps to achieve $\epsilon + \frac{1}{c}$ per-step regret.

PART I

- Theorem Proving
- MWU By Example
- Bounded-Regret Learning & Why
- MWU Is Bounded Regret

PART II

- Formalizing MWU
- Verifying Regret

VERIFIED MWU

MWU Formalized

The Coq Proof Assistant

Core Files

spec	proof	comments	
349	754	31	weights.v
738	932	75	weightslang.v
370	831	69	weightsextract.v
1457	2517	175	total

Auxiliary Files

spec	proof	comments	
269	1062	17	numerics.v
75	3	1	strings.v
110	80	5	dist.v
60	109	11	extrema.v
60	75	1	bigops.v
42	475	28	neps_exp_le.v
616	1804	63	total

TOTAL:
6632 LOC

Theorem: MWU Is Bounded Regret

Formal:

```
Notation astar := (best_action a0 cs).
Notation OPT := (\sum_(c <- cs) c astar).
Notation OPTR := (rat_to_R OPT).
```

… more definitions and notations …

```
Lemma perstep_weights_noregret :
((expCostsR - OPTR) / T <= epsR + ln size_A / (epsR * T))%R.
```

Informal:

$$(\mathbf{E}[C_{tot}(MWU)] - \min_a C_{tot}(a)) \,/\, T \leq \epsilon + \frac{\ln |A|}{\epsilon T}$$

EXPECTED TOTAL COST OF MWU

COST OF BEST FIXED ACTION

NUMBER OF STEPS

SIZE OF ACTION SPACE

A Hierarchy of Refinements

High-Level Functional Specification

```
Definition update_weights (w:weights) (c:costs) : weights :=
  finfun (fun a : A => w a * (1 - eps * c a)).
```

REFINES

MWU DSL

```
Binary Arith. Operations
  b ::= + | - | *
Expressions
  e ::= q
      | eps
      | e b e | …
Commands
  c ::= skip
      | update f | …
```

Operational Semantics

$$\vdash c, \sigma \Rightarrow c', \sigma'$$

REFINES

```
Fixpoint interp (c:com A.t) (s:cstate)
: option cstate := match c with … end.
```

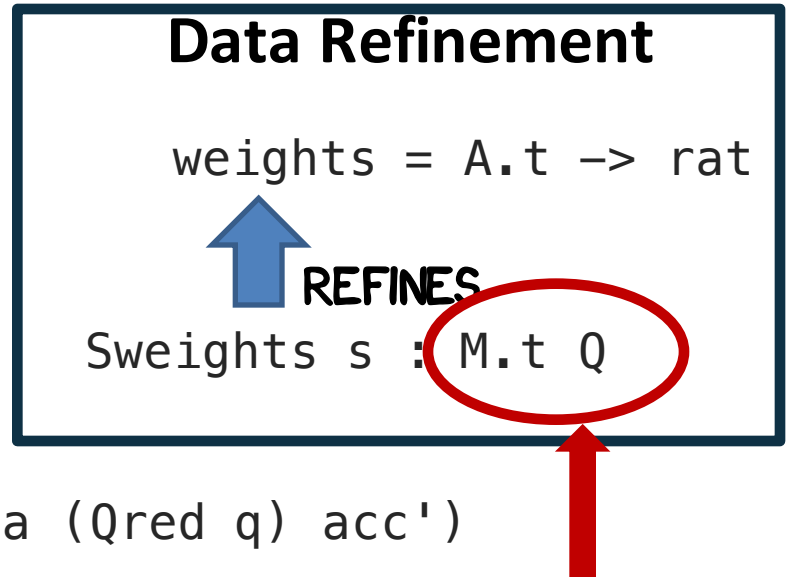
Executable Interpreter

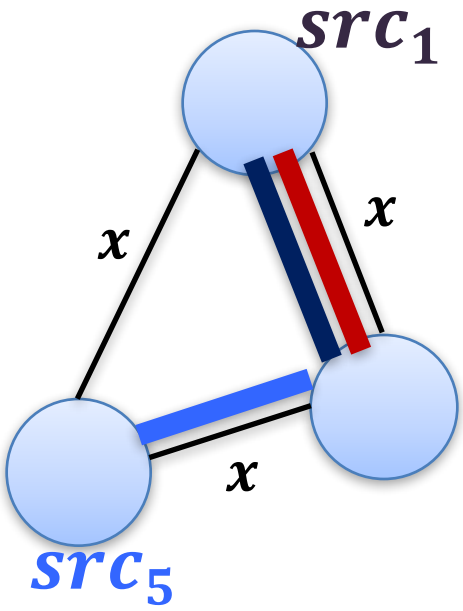
Even moderate-size proof developments (just like moderate-size software developments!) benefit from abstraction

Update Weights

```
Definition update_weights (w:weights) (c:costs) : weights :=
  finfun (fun a : A => w a * (1 - eps * c a)).
```

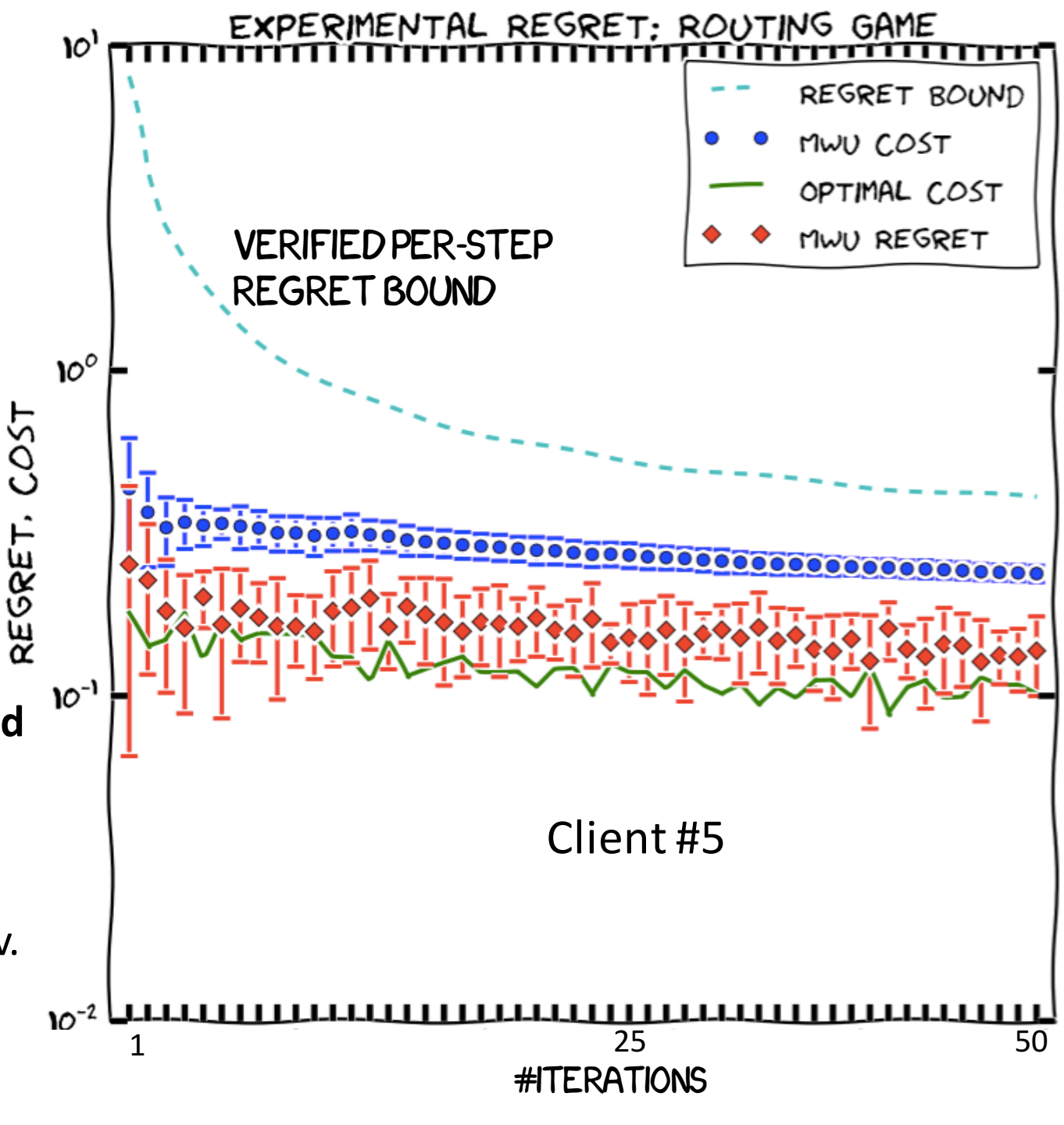

```
Definition update_weights (f : A.t -> expr A.t) (s : cstate)
  : option (M.t Q) :=
  M.fold
  (fun a _ acc =>
     match acc with
      | None => None
      | Some acc' =>
        match evalc (f a) s with
          | None => None
          | Some q =>
            match 0 ?= q with
              | Lt => Some (M.add a (Qred q) acc')
              | _ => None
            end end end)
  (SWeights s)
  (Some (M.empty Q)).
```





Simplified distributed routing game with

- 5 players
- 50 iterations
- mean and std. dev. over 10 trials
- $\epsilon = \frac{1}{4}$



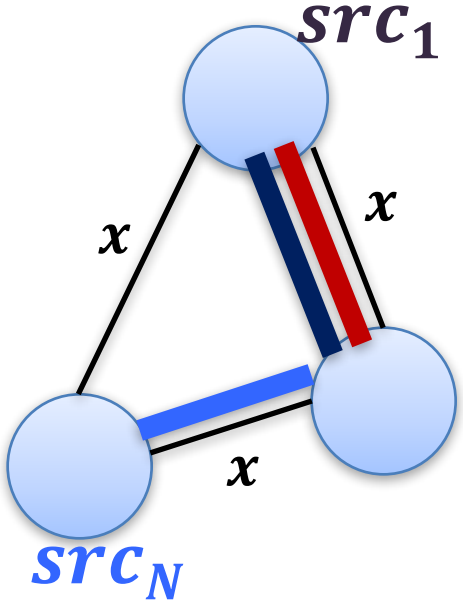
Extensions, Connections

- **Bandit Model**
  - revealing cost of all actions at each step imposes high communication overhead
  - assume, instead, only chosen action's cost is revealed
  - slightly more complex algorithms, slightly worse bounds, but perhaps faster in practice?
- **Linear Programming**
  - Verified MWU as a verified approximate LP solver!
- **AdaBoost** **[Freund & Schapire '97]**
- **[Arora et al., '12]**
  - a treasure trove of additional connections!

Conclusion

Verified Multiplicative Weights Update:

- **Machine-verified implementation** of a simple yet powerful algorithm for "combining expert advice"
- **Proof strategy:** layered program refinements, from executable MWU to high-level specification



Short/Medium Term Plans:

- From bounded regret to whole-system performance guarantees
- with applications to distributed systems (e.g., distributed routing, load balancing, etc.)

Thank You!

QUESTIONS?

References

[Arora et al., '12]: The Multiplicative Weights Update Method: A Meta-Algorithm and Applications. Theory of Computing, Volume 8 (2012), pp. 121–164.

[Freund & Schapire '97]: A Decision-Theoretic Generalization of On-Line Learning and an Application to Boosting. Journal of Comp. and System Sci. 55, 119-139 (1997).

[Littlestone, '88]: Learning quickly when irrelevant attributes abound: A new linear-threshold algorithm. *Machine Learning* 2.4 (1988): pp. 285-318.