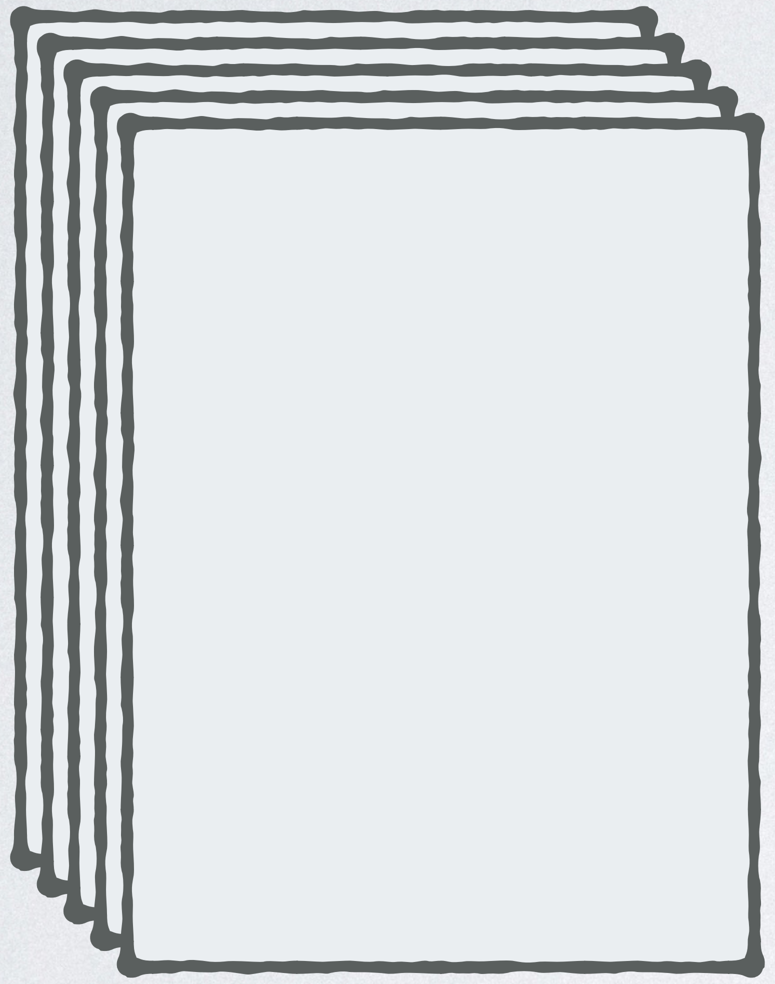
Raising Expectations: Automating Expected Cost Analysis with Types

Di Wang, David M. Kahn, Jan Hoffmann

Carnegie Mellon University

Cost Analysis

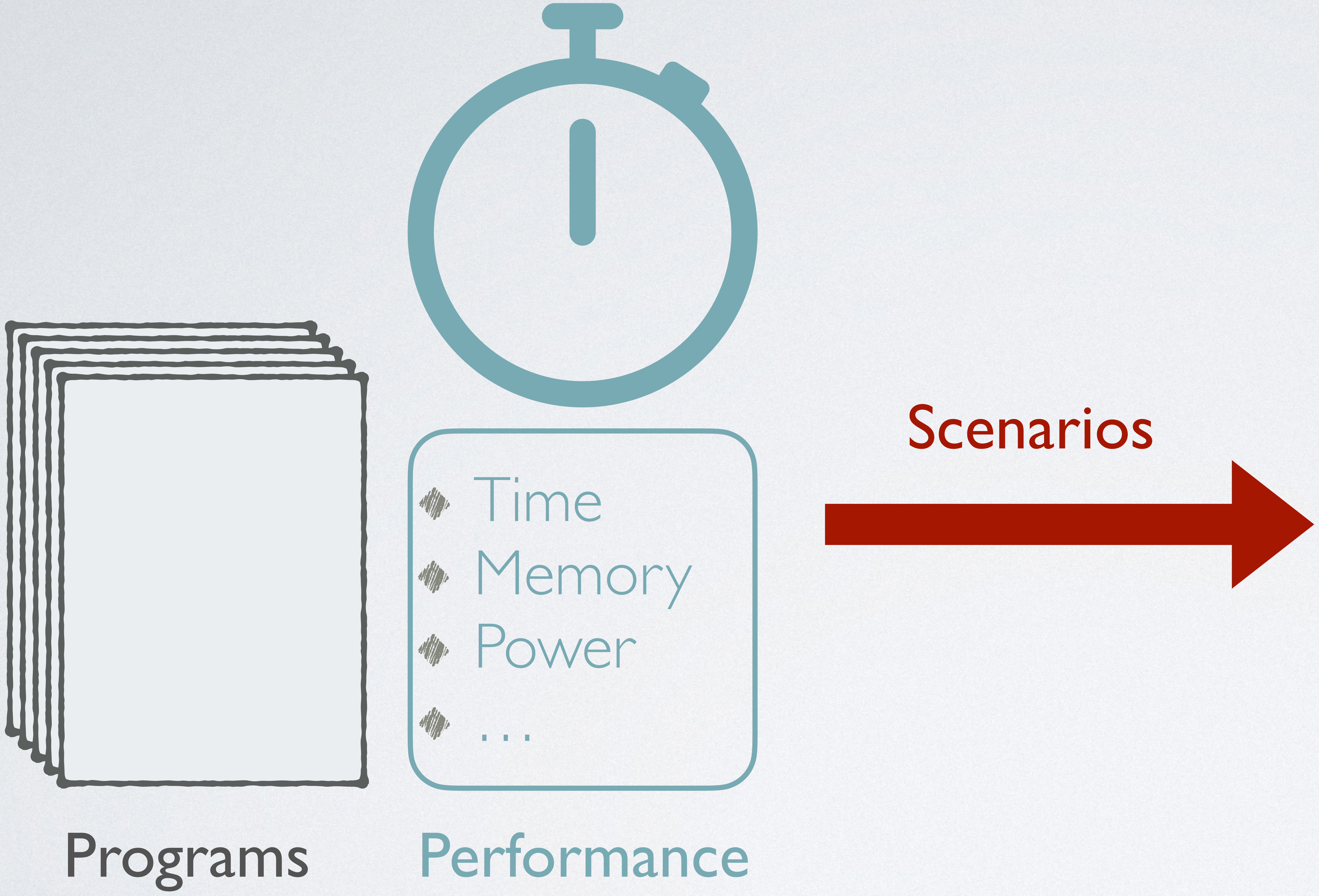


- Time
- Memory
- Power
- ...

Programs

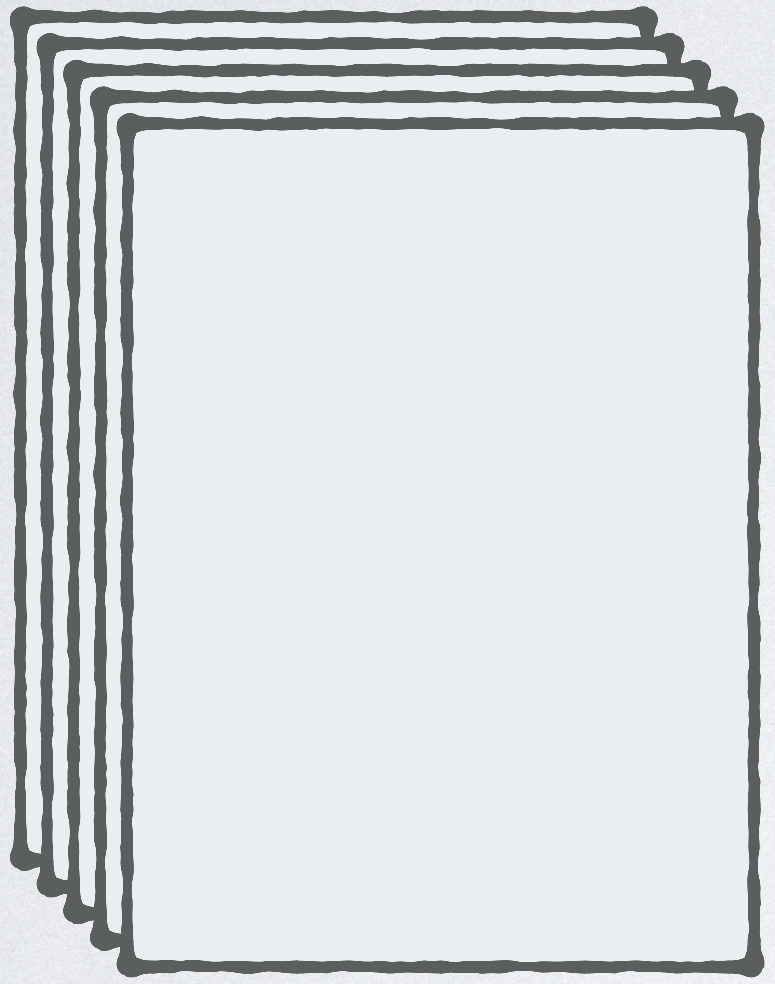
Performance

Cost Analysis



- Identifying bottlenecks

Cost Analysis



- Time
- Memory
- Power
- …

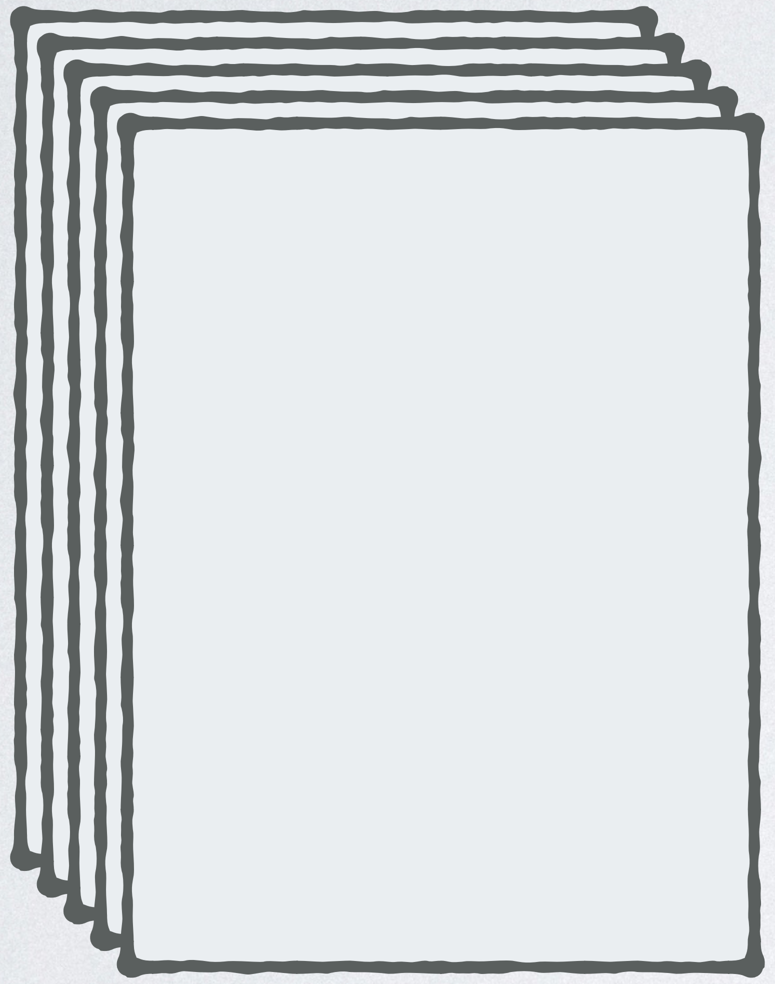
Programs

Performance

Scenarios

- Identifying bottlenecks
- Timing side channels

COST ANALYSIS



- Time
- Memory
- Power
- ...

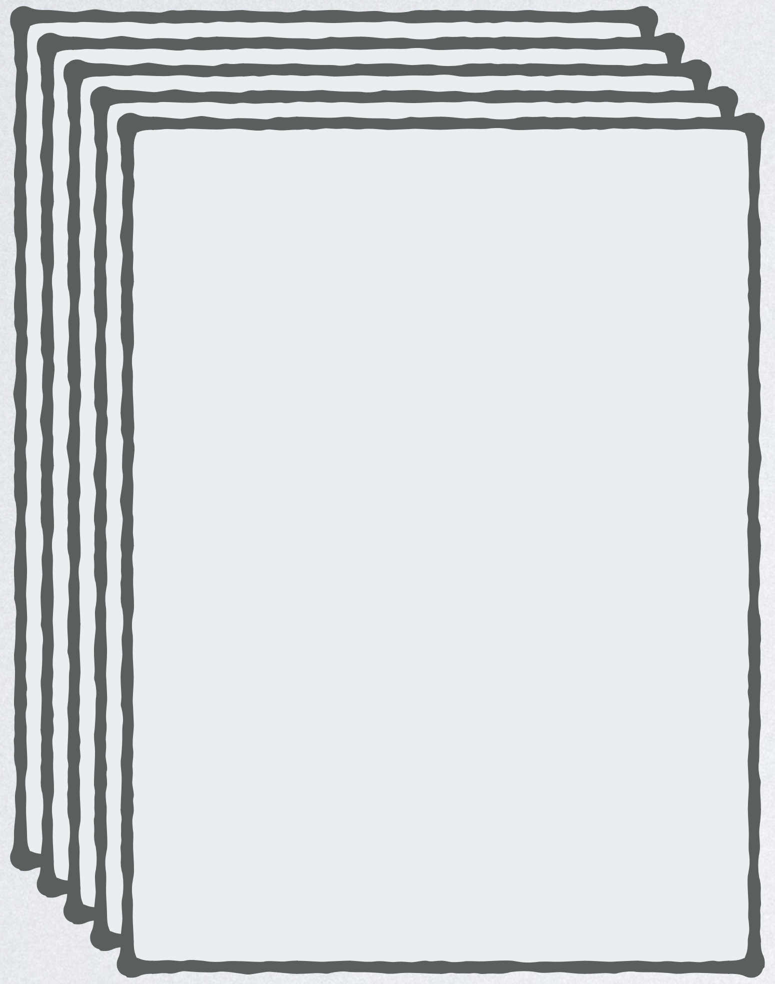
Programs

Performance

Scenarios

- Identifying bottlenecks
- Timing side channels
- Gas usage in blockchains

Cost Analysis



- Time
- Memory
- Power
- ...

Programs

Performance

Scenarios

- Identifying bottlenecks
- Timing side channels
- Gas usage in blockchains
- Carbon footprint

Automated Cost Analysis

Programs

[RaML] J. Hoffmann, A. Das, and S.-C. Weng. 2017. Towards Automatic Resource Bound Analysis for OCaml. In *POPL'17*.

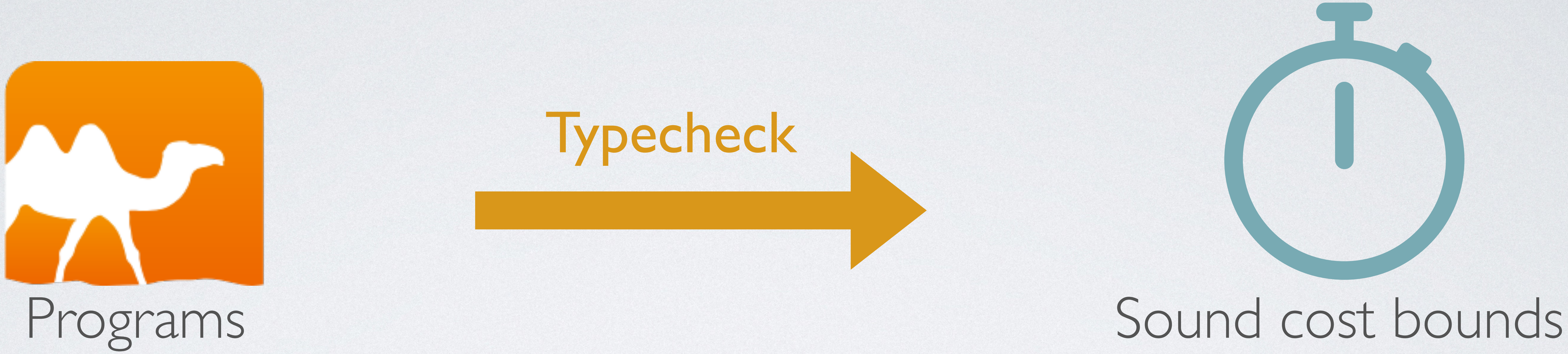
AUTOMATED COST ANALYSIS

Programs

Typecheck

[RaML] J. Hoffmann, A. Das, and S.-C. Weng. 2017. Towards Automatic Resource Bound Analysis for OCaml. In *POPL'17*.

AUTOMATED COST ANALYSIS



[RaML] J. Hoffmann, A. Das, and S.-C. Weng. 2017. Towards Automatic Resource Bound Analysis for OCaml. In *POPL'17*.

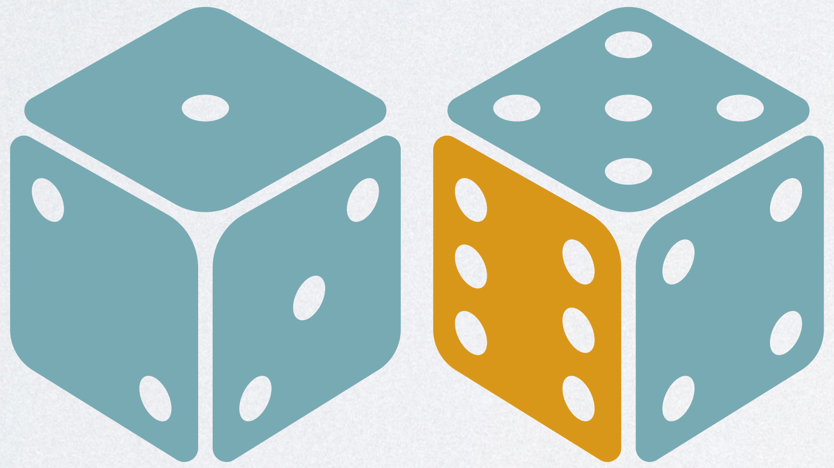
AUTOMATED COST ANALYSIS

[RaML] J. Hoffmann, A. Das, and S.-C. Weng. 2017. Towards Automatic Resource Bound Analysis for OCaml. In *POPL'17*.

AUTOMATED COST ANALYSIS

Programs

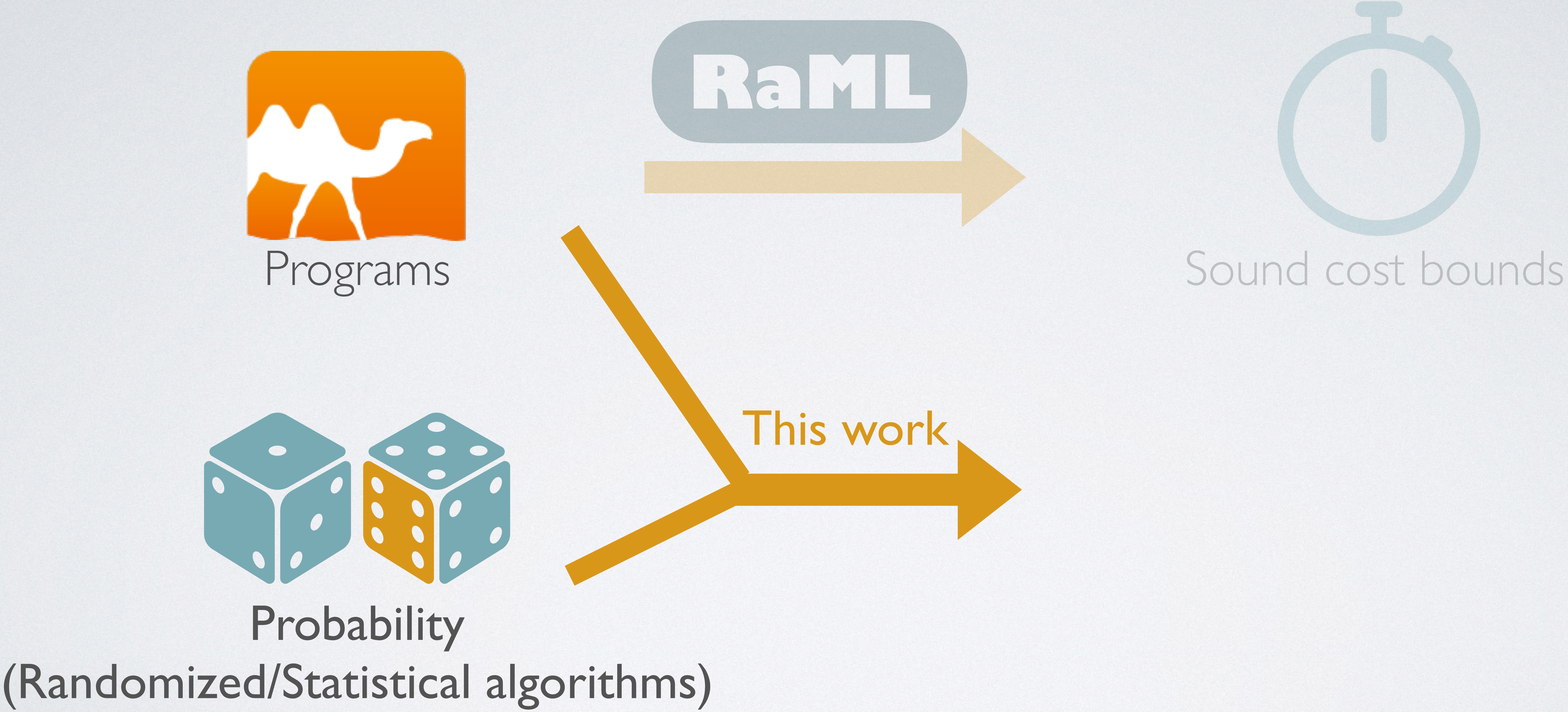
Sound cost bounds



Probability
(Randomized/Statistical algorithms)

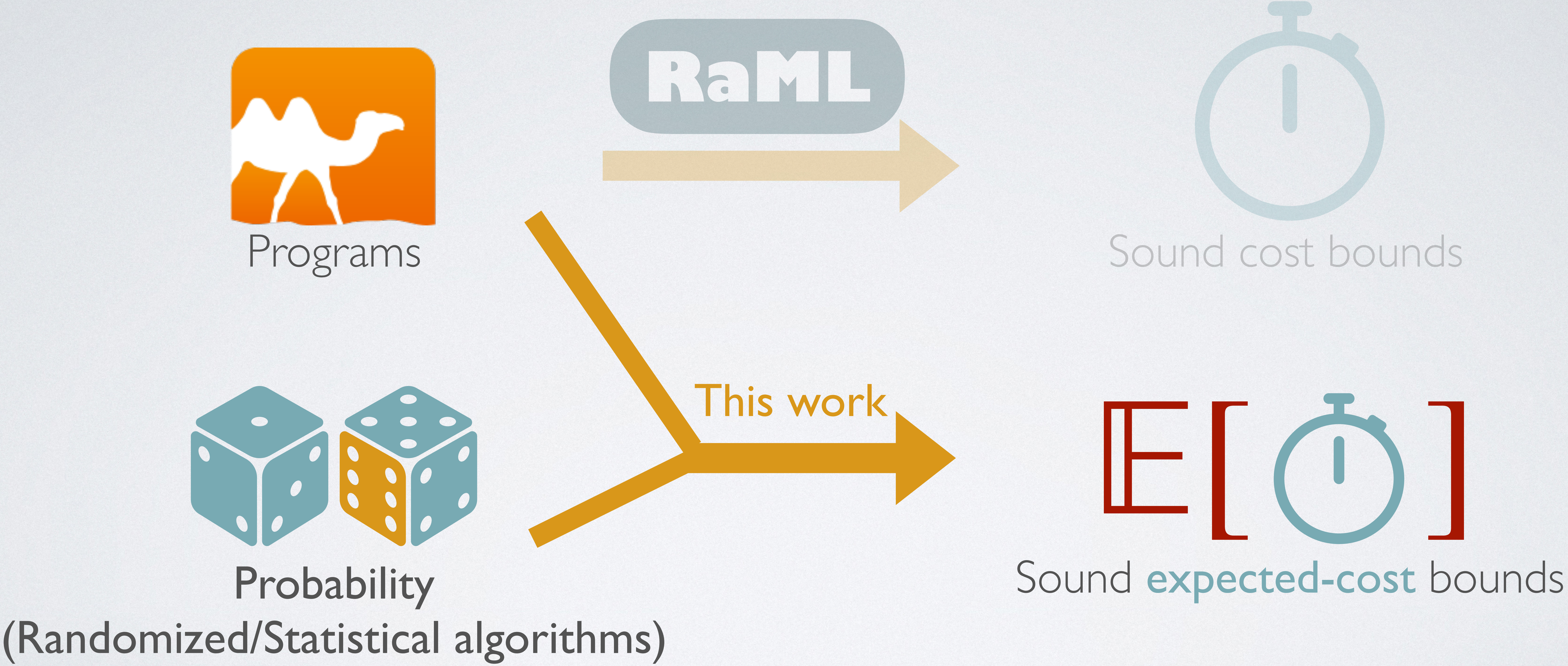
[RaML] J. Hoffmann, A. Das, and S.-C. Weng. 2017. Towards Automatic Resource Bound Analysis for OCaml. In *POPL'17*.

Automated Cost Analysis



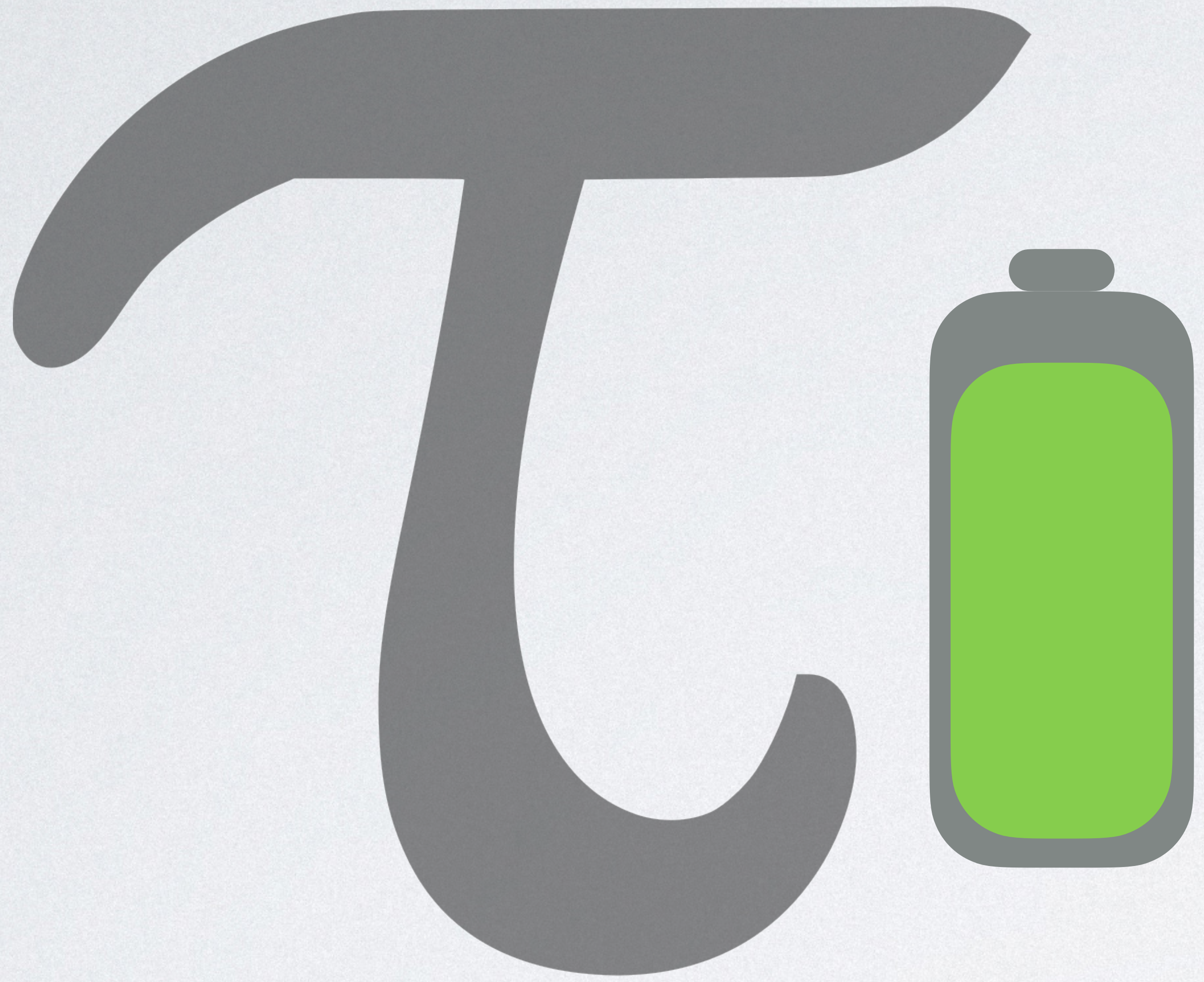
[RaML] J. Hoffmann, A. Das, and S.-C. Weng. 2017. Towards Automatic Resource Bound Analysis for OCaml. In *POPL'17*.

AUTOMATED COST ANALYSIS

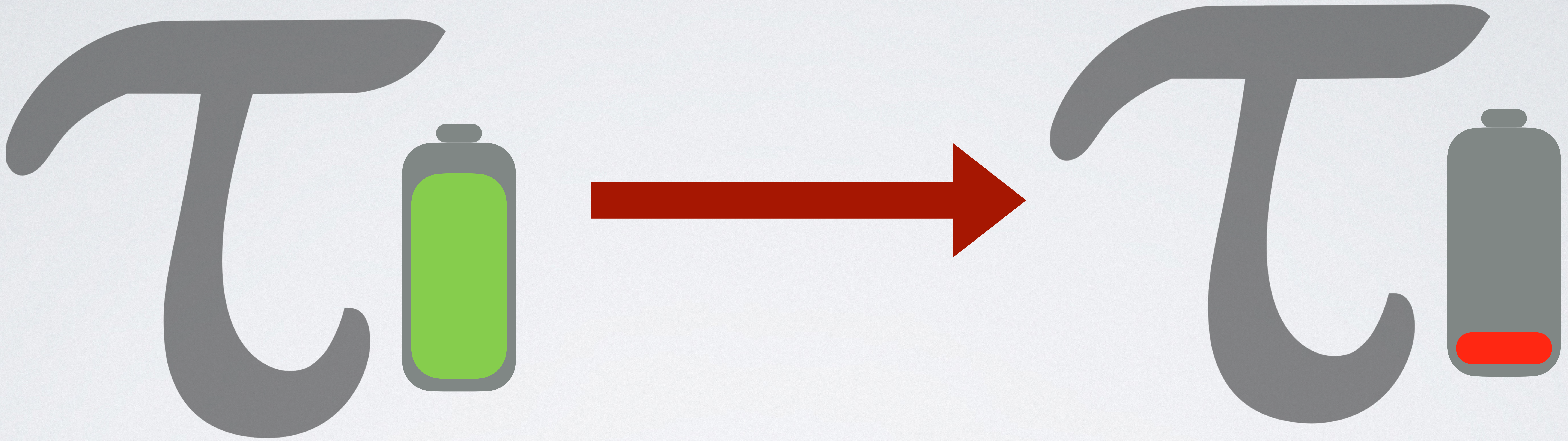


[RaML] J. Hoffmann, A. Das, and S.-C. Weng. 2017. Towards Automatic Resource Bound Analysis for OCaml. In *POPL'17*.

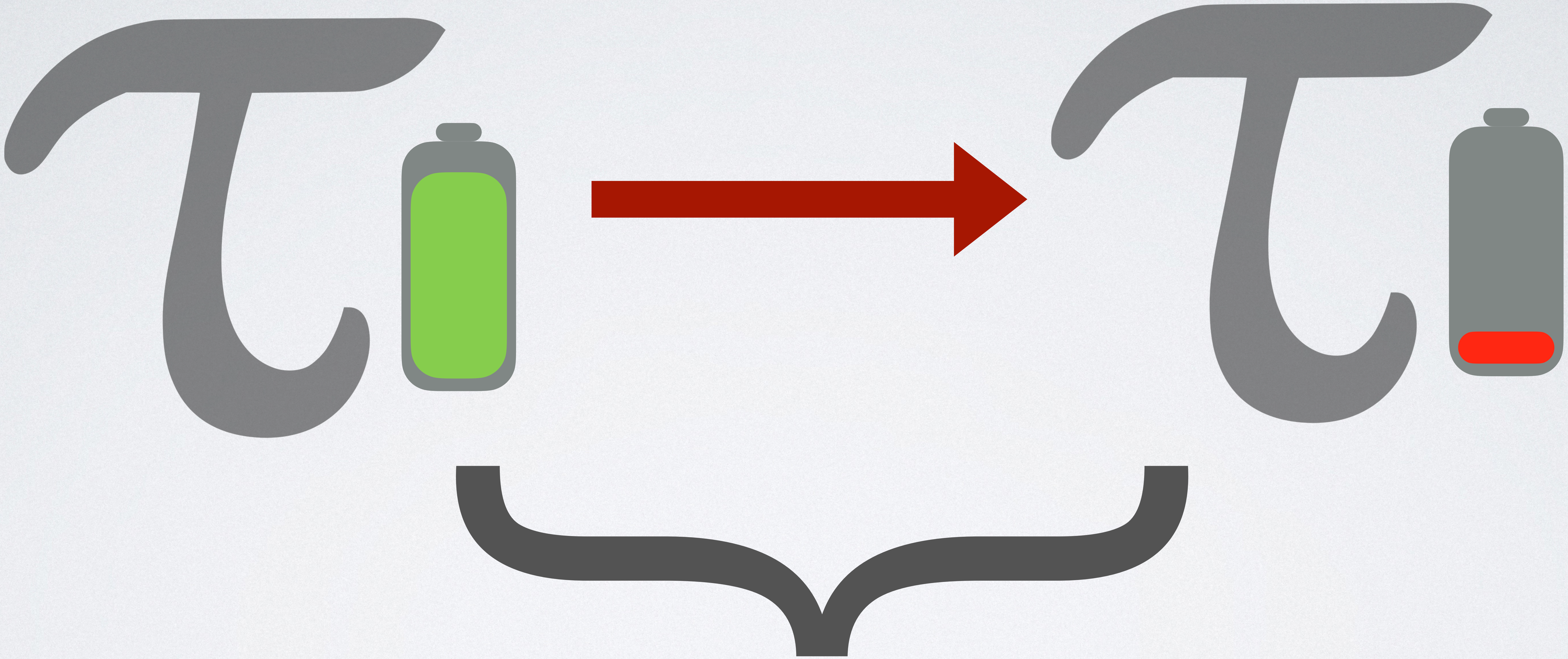
Potential Method



Potential Method



Potential Method



cost

MEM

```
let rec mem x lst =
  match lst with
   | [] -> false
   | h::t ->
     if compare h x = 0
     then true
     else mem x t
```

Mem

```
let rec mem x lst =
  match lst with
  | [] -> false
  | h::t ->
    if compare h x = 0
    then true
```

Mem

```
let rec mem x lst =
  match lst with
  | [] -> false
  | h::t ->
    if compare h x = 0
    then true
    else let () = tick(1.0) in
      mem x t
```

Mem

```
let rec mem x lst =
  match lst with
  | []  -> false
  | h::t ->
    if compare h x = 0
    then true
    else let () = tick(1.0) in
      mem x t
```

x : a   lst : L^1(a)

0 resources

MEM

```
let rec mem x lst =
  match lst with
  | []  -> false
  | h::t ->
    if compare h x = 0
    then true
    else let () = tick(1.0) in
      mem x t
```

x : a lst : L^1(a) mem : <a * L^1(a),0> -> <bool,0>

0 resources

Mem

x : a   lst : L^1(a)   mem : <a * L^1(a),0> -> <bool,0>

0 resources

```
let rec mem x lst =
  match lst with
  | [] -> false
  | h::t ->
    if compare h x = 0
    then true
    else let () = tick(1.0) in
      mem x t
```

MEM

```
let rec mem x lst =
  match lst with
  | [] -> false
  | h::t ->
    if compare h x = 0
    then true
    else let () = tick(1.0) in
      mem x t
```

x : a lst : L¹(a) mem : <a*L¹(a),0> -> <bool,0>

0 resources

MEM

```
let rec mem x lst =
  match lst with
   | [] -> false
   | h::t ->
     if compare h x = 0
     then true
     else let () = tick(1.0) in
       mem x t
```

x : a   lst : L^1(a)

0 resources

mem : <a * L^1(a),0> -> <bool,0>

h : a   t : L^1(a)

1 resource

Mem

```
let rec mem x lst =
  match lst with
   | [] -> false
   | h::t ->
     if compare h x = 0
     then true
     else let () = tick(1.0) in
       mem x t
```

x : a **lst** : L^1(a) **mem** : <a * L^1(a), 0> -> <bool, 0>

0 resources

h : a **t** : L^1(a)

1 resource

1 >= 0 resources

MEM

```
let rec mem x lst =
  match lst with
   | [] -> false
   | h::t ->
     if compare h x = 0
     then true
     else let () = tick(1.0) in
       mem x t
```

x : a   lst : L^1(a)   mem : <a * L^1(a),0> -> <bool,0>

0 resources

h : a   t : L^1(a)

1 resource

1 >= 0 resources

0 resources

Mem

```
let rec mem x lst =          x : a   lst : L^1(a)     mem : <a * L^1(a),0> -> <bool,0>
                               0 resources
  match lst with
  | [] -> false
  | h::t ->                     h : a   t : L^1(a)
    if compare h x = 0            1 resource
    then true                   1 >= 0 resources
    else let () = tick(1.0) in   0 resources
      mem x t                     0 resources
```

Flip Rule

$$\frac{\Gamma \curlyvee (p \times \Gamma_1, (1-p) \times \Gamma_2) \qquad q = p \cdot q_1 + (1-p) \cdot q_2 \qquad \Gamma_1; q_1 \vdash e_1 : A \qquad \Gamma_2; q_2 \vdash e_2 : A}{\Gamma; q \vdash \mathrm{flip}\{e_1; e_2\}(p) : A} \text{ (L:Flip)}$$

Flip Rule

$$\frac{\Gamma \curlyvee (p \times \Gamma_1, (1-p) \times \Gamma_2) \qquad q = p \cdot q_1 + (1-p) \cdot q_2 \qquad \Gamma_1; q_1 \vdash e_1 : A \qquad \Gamma_2; q_2 \vdash e_2 : A}{\Gamma; q \vdash \mathrm{flip}\{e_1; e_2\}(p) : A} \;(\text{L:}\textsc{Flip})$$

Typing Context

FLIP RULE

$$\frac{\Gamma \curlyvee (p \times \Gamma_1, (1-p) \times \Gamma_2) \qquad q = p \cdot q_1 + (1-p) \cdot q_2 \qquad \Gamma_1; q_1 \vdash e_1 : A \qquad \Gamma_2; q_2 \vdash e_2 : A}{\Gamma; q \vdash \mathrm{flip}\{e_1; e_2\}(p) : A} \text{ (L:FLIP)}$$

Typing Context

Resources Present

Flip Rule

$$\frac{\Gamma \curlyvee (p \times \Gamma_1, (1-p) \times \Gamma_2) \qquad q = p \cdot q_1 + (1-p) \cdot q_2 \qquad \Gamma_1; q_1 \vdash e_1 : A \qquad \Gamma_2; q_2 \vdash e_2 : A}{\Gamma; q \vdash \mathrm{flip}\{e_1; e_2\}(p) : A} \text{(L:Flip)}$$

Typing Context

Resources Present

Resource-Annotated Type

FLIP RULE

$$\frac{\Gamma \curlyvee (p \times \Gamma_1, (1-p) \times \Gamma_2) \qquad q = p \cdot q_1 + (1-p) \cdot q_2 \qquad \Gamma_1; q_1 \vdash e_1 : A \qquad \Gamma_2; q_2 \vdash e_2 : A}{\Gamma; q \vdash \mathrm{flip}\{e_1; e_2\}(p) : A} \text{ (L:FLIP)}$$

Typing Context

Resources Present

Coin Flip Probability

Resource-Annotated Type

FLIP RULE

Branches

$$\frac{\Gamma \curlyvee (p \times \Gamma_1, (1-p) \times \Gamma_2) \qquad q = p \cdot q_1 + (1-p) \cdot q_2 \qquad \Gamma_1; q_1 \vdash e_1 : A \qquad \Gamma_2; q_2 \vdash e_2 : A}{\Gamma; q \vdash \text{flip}\{e_1; e_2\}(p) : A} \text{(L:FLIP)}$$

Typing Context

Resource-Annotated Type

Resources Present

Coin Flip Probability

FLIP RULE

Context Weighted Sum

Branches

$$\frac{\Gamma \curlyvee (p \times \Gamma_1, (1-p) \times \Gamma_2) \qquad q = p \cdot q_1 + (1-p) \cdot q_2 \qquad \Gamma_1; q_1 \vdash e_1 : A \qquad \Gamma_2; q_2 \vdash e_2 : A}{\Gamma; q \vdash \mathrm{flip}\{e_1; e_2\}(p) : A}\ (\text{L:FLIP})$$

Typing Context

Resources Present

Coin Flip Probability

Resource-Annotated Type

FLIP RULE

Context Weighted Sum

Branches

Resource Weighted Sum

$$\frac{\Gamma \curlyvee (p \times \Gamma_1, (1-p) \times \Gamma_2) \qquad q = p \cdot q_1 + (1-p) \cdot q_2 \qquad \Gamma_1; q_1 \vdash e_1 : A \qquad \Gamma_2; q_2 \vdash e_2 : A}{\Gamma; q \vdash \mathrm{flip}\{e_1; e_2\}(p) : A} \;(\textsc{L:Flip})$$

Typing Context

Resource-Annotated Type

Resources Present

Coin Flip Probability

FLIP RULE

Context **Weighted Sum**

Branches

Resource **Weighted Sum**

$$\frac{\Gamma \curlyvee (p \times \Gamma_1, (1-p) \times \Gamma_2) \qquad q = p \cdot q_1 + (1-p) \cdot q_2 \qquad \Gamma_1; q_1 \vdash e_1 : A \qquad \Gamma_2; q_2 \vdash e_2 : A}{\Gamma; q \vdash \mathrm{flip}\{e_1; e_2\}(p) : A}\ (\text{L:F}\textsc{lip})$$

Typing Context

Resource-Annotated Type

Resources Present

Coin Flip Probability

Our implementation uses **multivariate** polynomial bounds
that can mention parameterized probabilities

SOUNDNESS

SOUNDNESS

Operational semantics adapted from **Borgström et al. 2016**

SOUNDNESS

Operational semantics adapted from **Borgström et al. 2016**

- If expression ***E*** types as ***A*** under context ***Γ*** with ***Q*** units of initial potential, then the **expected cost** of evaluating ***E*** is bounded by ***Q*** plus the potential in ***Γ***.

SOUNDNESS

Operational semantics adapted from **Borgström et al. 2016**

- If expression ***E*** types as ***A*** under context ***Γ*** with ***Q*** units of initial potential, then the **expected cost** of evaluating ***E*** is bounded by ***Q*** plus the potential in ***Γ***.

- If the cost model counts evaluation steps, then the **existence** of an **expected cost bound** implies **almost-sure termination**.

Bernoulli

```
let rec bernoulli () =
  let () = tick(1.0) in
  match flip(0.5) with
  | H -> ()
  | T -> bernoulli () in
```

Bernoulli

```
let rec bernoulli () =     2 resources
  let () = tick(1.0) in
  match flip(0.5) with
  | H -> ()
  | T -> bernoulli () in
```

Bernoulli

```
let rec bernoulli () =
  let () = tick(1.0) in
  match flip(0.5) with
  | H -> ()
  | T -> bernoulli () in
```

2 resources

bernoulli : <unit,2> -> <bool,0>

Bernoulli

```
let rec bernoulli () =
  let () = tick(1.0) in
  match flip(0.5) with
  | H -> ()
  | T -> bernoulli () in
```

2 resources

1 resource

bernoulli : <unit,2> -> <bool,0>

Bernoulli

```
let rec bernoulli () =        2 resources     bernoulli : <unit,2> -> <bool,0>
  let () = tick(1.0) in         1 resource
  match flip(0.5) with
  | H -> ()                   0 resources
  | T -> bernoulli () in
```

BERNOULLI

```
let rec bernoulli () =        2 resources    bernoulli : <unit,2> -> <bool,0>
  let () = tick(1.0) in          1 resource
  match flip(0.5) with
  | H -> ()                    0 resources
  | T -> bernoulli () in          2 resources
```

Bernoulli

```
let rec bernoulli () =          2 resources     bernoulli : <unit,2> -> <bool,0>
   let () = tick(1.0) in           1 resource
   match flip(0.5) with
   | H -> ()                    0 resources
   | T -> bernoulli () in          2 resources
```

$$(0.5 * 0) + (0.5 * 2) = 1$$

PROBABILISTIC MODELS

```
(* Gambler's ruin *)
let rec gr Alice Bob =
  match Alice with
  | [] -> ()
  | ha::ta ->
    match Bob with
    | [] -> ()
    | hb::tb ->
      let _ = tick 1 in
      match flip 0.5 with
      | H -> gr ta (ha::Bob)
      | T -> gr (hb::Alice) tb
```

```
(* Makes a fair coin from a biased one *)
let rec von_neumann p =
  let _ = tick p*(1-p) in
  match flip p with
  | H ->
    let _ = tick p*(1-p) in
    match flip p with
    | H -> von_neumann p
    | T -> H
  | T ->
    let _ = tick p*(1-p) in
    match flip p with
    | H -> T
    | T -> von_neumann p
```

Probabilistic Models

```
(* Gambler's ruin *)
let rec gr Alice Bob =
  match Alice with
  | [] -> ()
  | ha::ta ->
    match Bob with
    | [] -> ()
    | hb::tb ->
      let _ = tick 1 in
      match flip 0.5 with
      | H -> gr ta (ha::Bob)
      | T -> gr (hb::Alice) tb
```

|Alice| * |Bob|

```
(* Makes a fair coin from a biased one *)
let rec von_neumann p =
  let _ = tick p*(1-p) in
  match flip p with
  | H ->
    let _ = tick p*(1-p) in
    match flip p with
    | H -> von_neumann p
    | T -> H
  | T ->
    let _ = tick p*(1-p) in
    match flip p with
    | H -> T
    | T -> von_neumann p
```

Probabilistic Models

```
(* Gambler's ruin *)
let rec gr Alice Bob =
  match Alice with
  | [] -> ()
  | ha::ta ->
    match Bob with
    | [] -> ()
    | hb::tb ->
      let _ = tick 1 in
      match flip 0.5 with
      | H -> gr ta (ha::Bob)
      | T -> gr (hb::Alice) tb
```

|Alice| * |Bob|

```
(* Makes a fair coin from a biased one *)
let rec von_neumann p =
  let _ = tick p*(1-p) in
  match flip p with
  | H ->
    let _ = tick p*(1-p) in
    match flip p with
    | H -> von_neumann p
    | T -> H
  | T ->
    let _ = tick p*(1-p) in
    match flip p with
    | H -> T
    | T -> von_neumann p
```

1/(p(1-p))

RESULTS TABLE

Program description	Bound	#Constraints	Time (in sec.)
goat (probs=[1/2,3/4])	$(B+1)(2(G+1)-G_B)$	2084	0.15
goat (probs=[2/3,3/4])	$3B+3$	2084	0.14
goat (probs=[1/2,2/3,3/4])	$(B+1)(2(G+1.5)-G_B)$	5336	0.25
goat (probs=[1/2,3/5,2/3,3/4])	$(B+1)(2(G+2.5)-G_B)$	10996	1.95
trade (probs=[3/5,1/3])	$1/15*T^2+1/3*T*P+4/15*T$	157	0.04
trade (probs=[3/5,1])	$1/5*T^2+T*P+4/5*T$	157	0.03
trade (probs=[2/5,1])	$3/10*T^2+T*P+7/10*T$	157	0.03
trade (probs=[2/5,1/3])	$1/10*T^2+1/3*T*P+7/30*T$	157	0.04
probabilistic loop	3/4 probability	61	0.01
bayes sampling	3/5 probability	112	0.01
die simulation from coin	1/6 per die face	5731	0.33
random no-op ***nested*** variant	M^2+M	205	0.03
miner	$15/2*M$	31	0.01
fill and ***consume***	$(1/3+p/6)*M$	633	0.11

AVERAGE CASE ANALYSIS

```
let rec mem x lst =
    let () = tick(1.0) in
    match lst with
    | [] -> false
    | h::t ->
        if compare h x = 0
        then true
        else mem x t
```

vs

```
let rec bernoulli () =
    let () = tick(1.0) in
    match flip(0.5) with
    | H -> ()
    | T -> bernoulli ()
```

flip(0.5)

CONTRIBUTION

- A **type-based** cost analysis for **probabilistic programs**
- Type **soundness** proof with respect to a **probabilistic operational cost semantics**
- **Implementation** that supports multivariate polynomial bounds
- Experiments on **average-case cost estimation** and **sample complexity analysis**