Two Approximate-Programmability Birds, One Statistical-Inference Stone

Adrian Sampson
University of Washington

APPROX 2014

sampa

31

EnerJ, the Language of Good-Enough Computing (spectrum.ieee.org)

submitted 8 months ago by jms_nh

23 comments share save hide give gold report instapaper

. . .

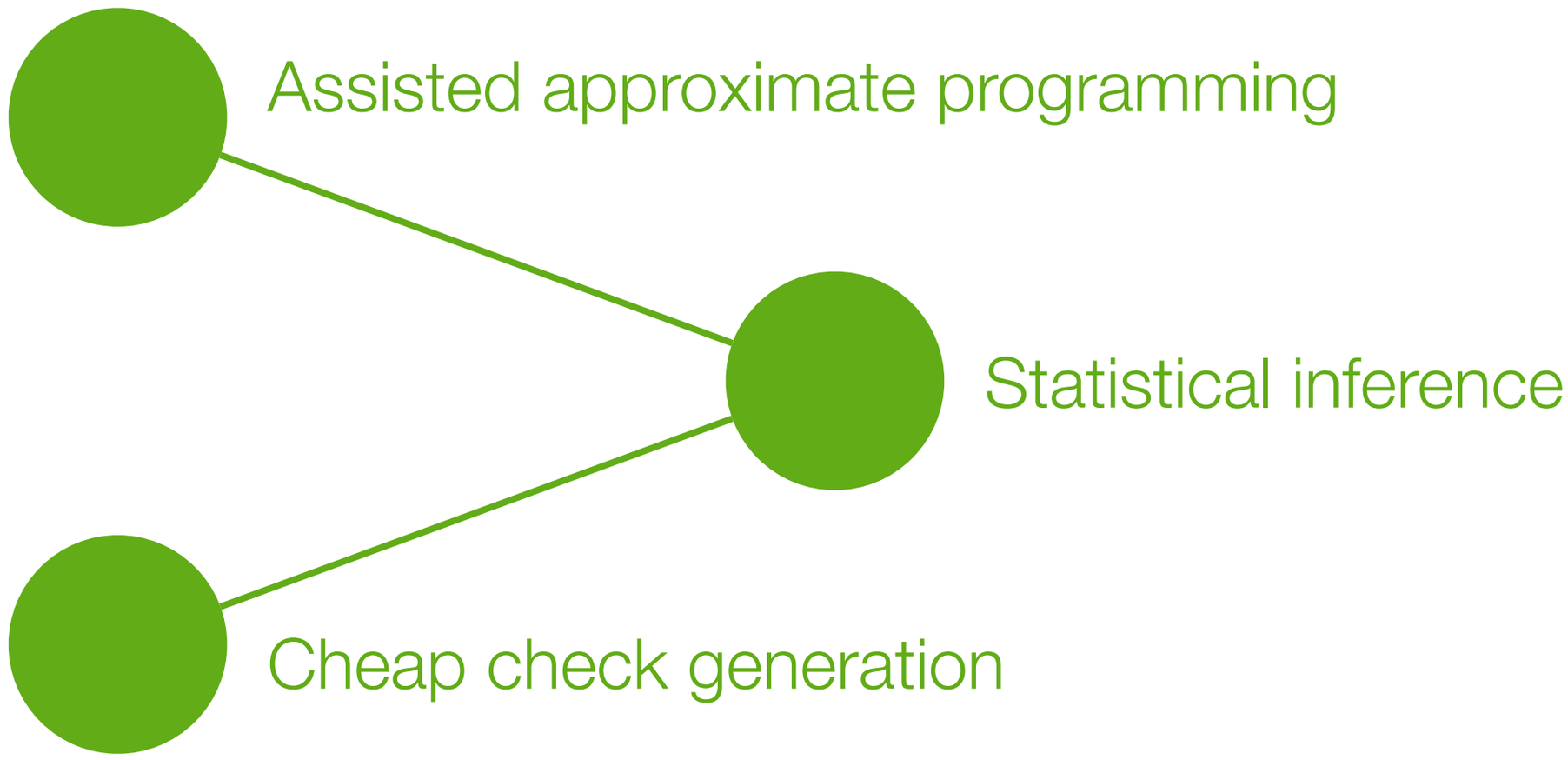
[–] **jtra** 3 points 8 months ago

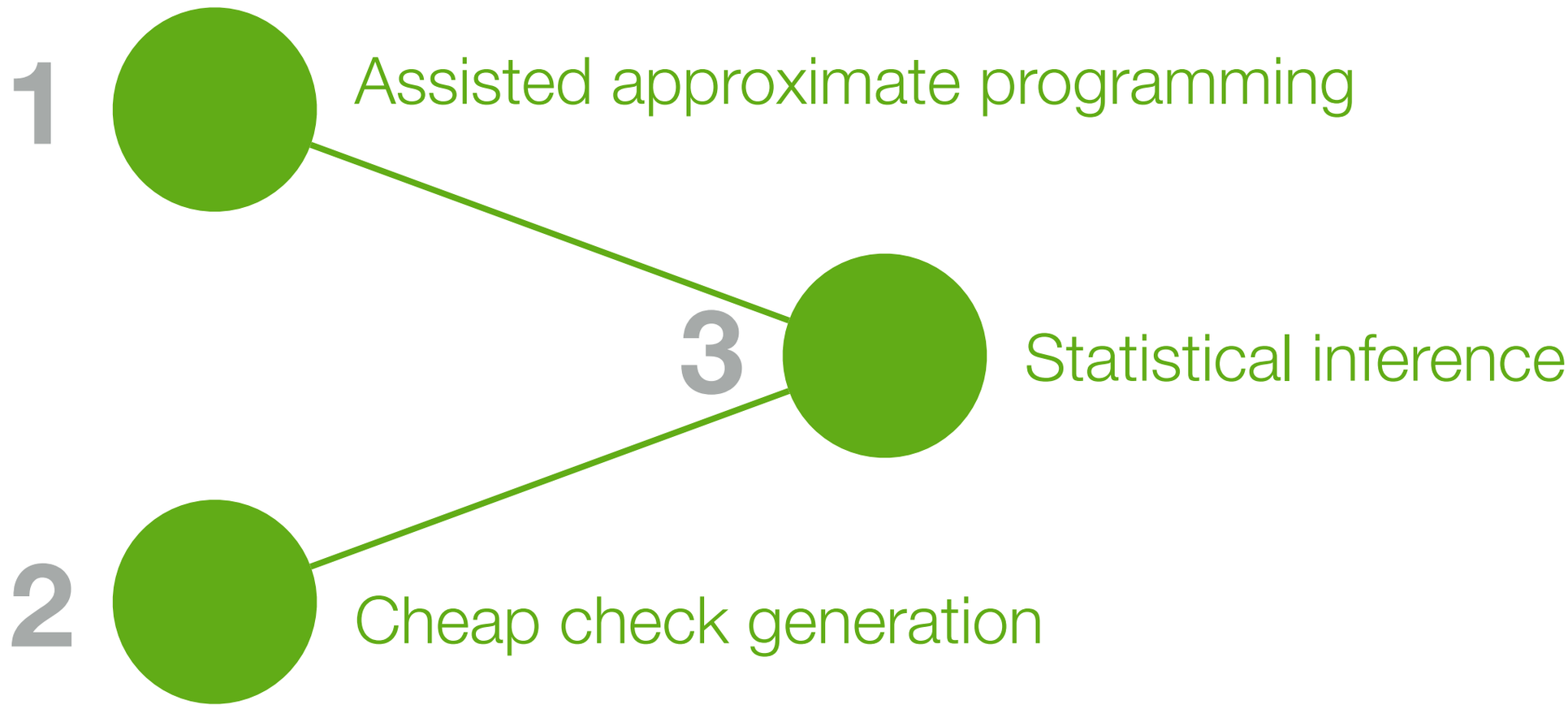
Good luck debugging that...

permalink save give gold

[–] **MorePudding** 10 points 8 months ago

Oh god, this sounds awful .. and it has enough potential to actually r
shoving it down our throats by force (i.e. people recalculating things
standard/"unofficial" hardware that tries to work around the limitatio
that).


Assisted approximate programming
Statistical inference
Cheap check generation


1
Assisted approximate programming
3
Statistical inference
2
Cheap check generation

4 Next steps

Expressing quality

original program

relaxed version

input

$$\Pr[d(f(x), f'(x)) \leq b] \geq p$$

distance metric

bounding parameters

Assisted approximate programming

$$f \longrightarrow f'$$

Manual approximate programming

$$f \longrightarrow f'$$

```
int p = 5;
int a = 7;
for (int x = 0..) {
   a += func(2);
   int z;
   z = p * 2;
   p += 4;
}
a /= 9;
func2(p);
a += func(2);
int y;
z = p * 22 + z;
p += 10;
```

```
int p = 5;
@Approx int a = 7;
for (int x = 0..) {
   a += func(2);
   @Approx int z;
   z = p * 2;
   p += 4;
}
a /= 9;
func2(p);
a += func(2);
@Approx int y;
z = p * 22 + z;
p += 10;
```

Assisted approximate programming

$$f \longrightarrow f'$$

$$+$$

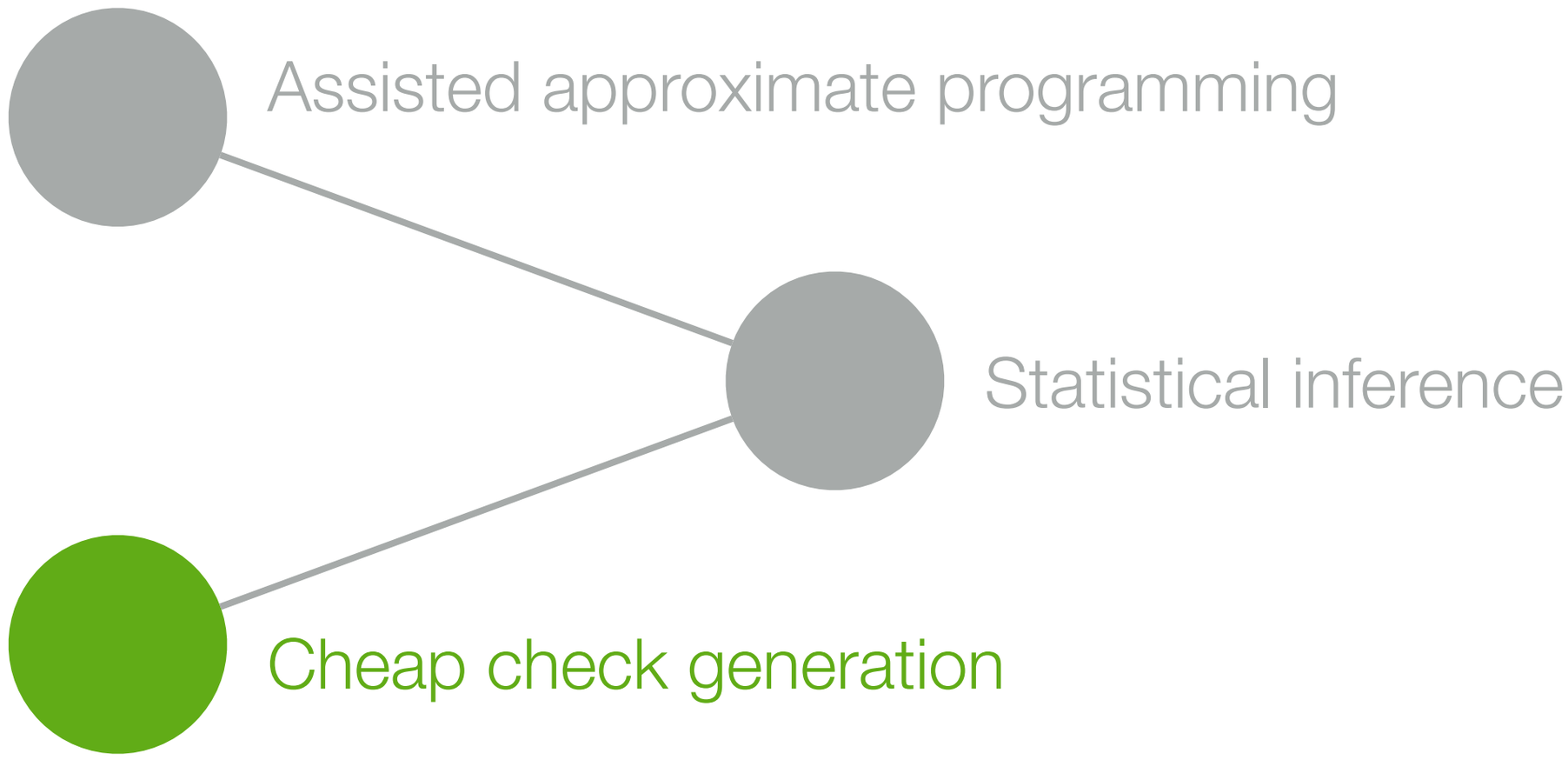
$$\Pr[d(f(x), f'(x)) \leq b] \geq p$$

ExpAX [Esmaeilzadeh+]

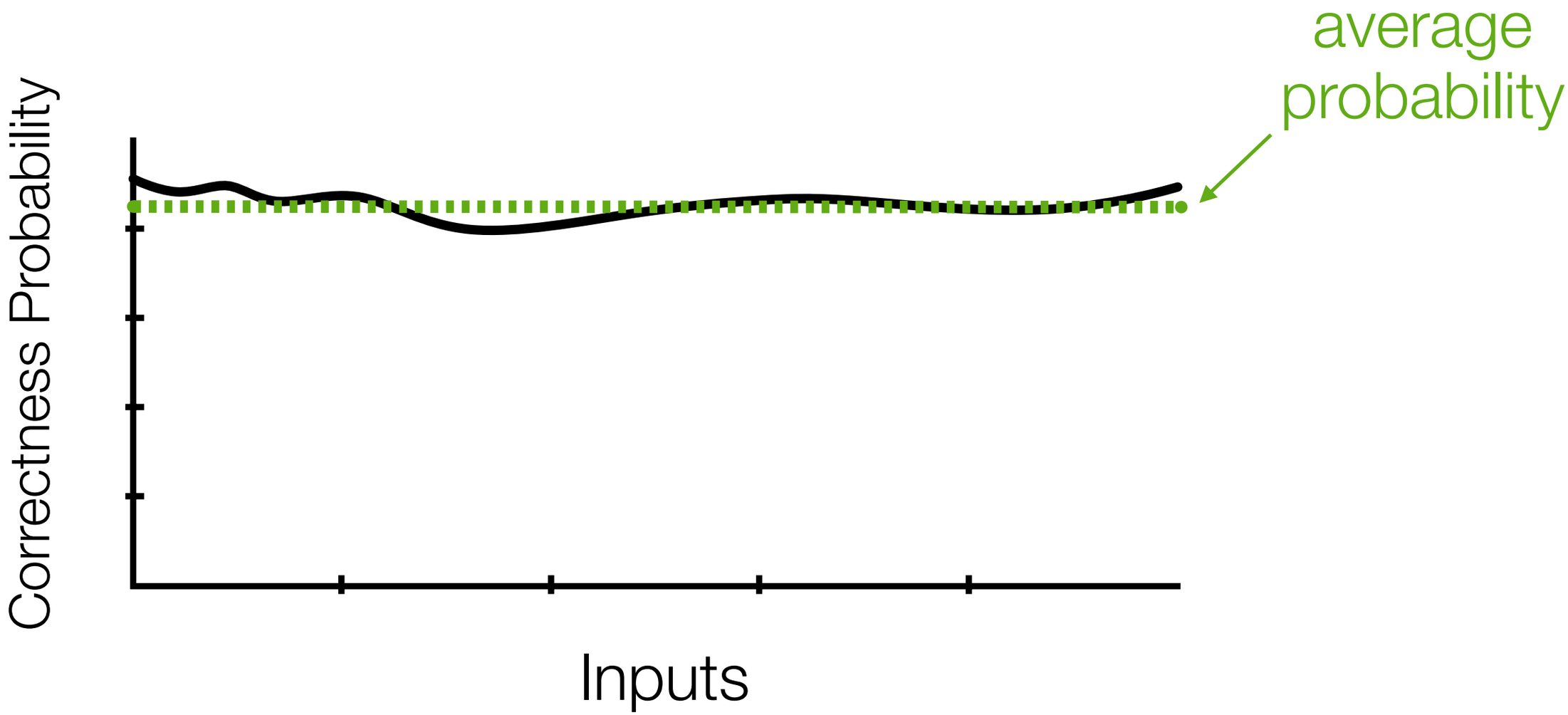
Syndy [Misailovic and Rinard, WACAS]

Optimization in Rely [Misailovic+]

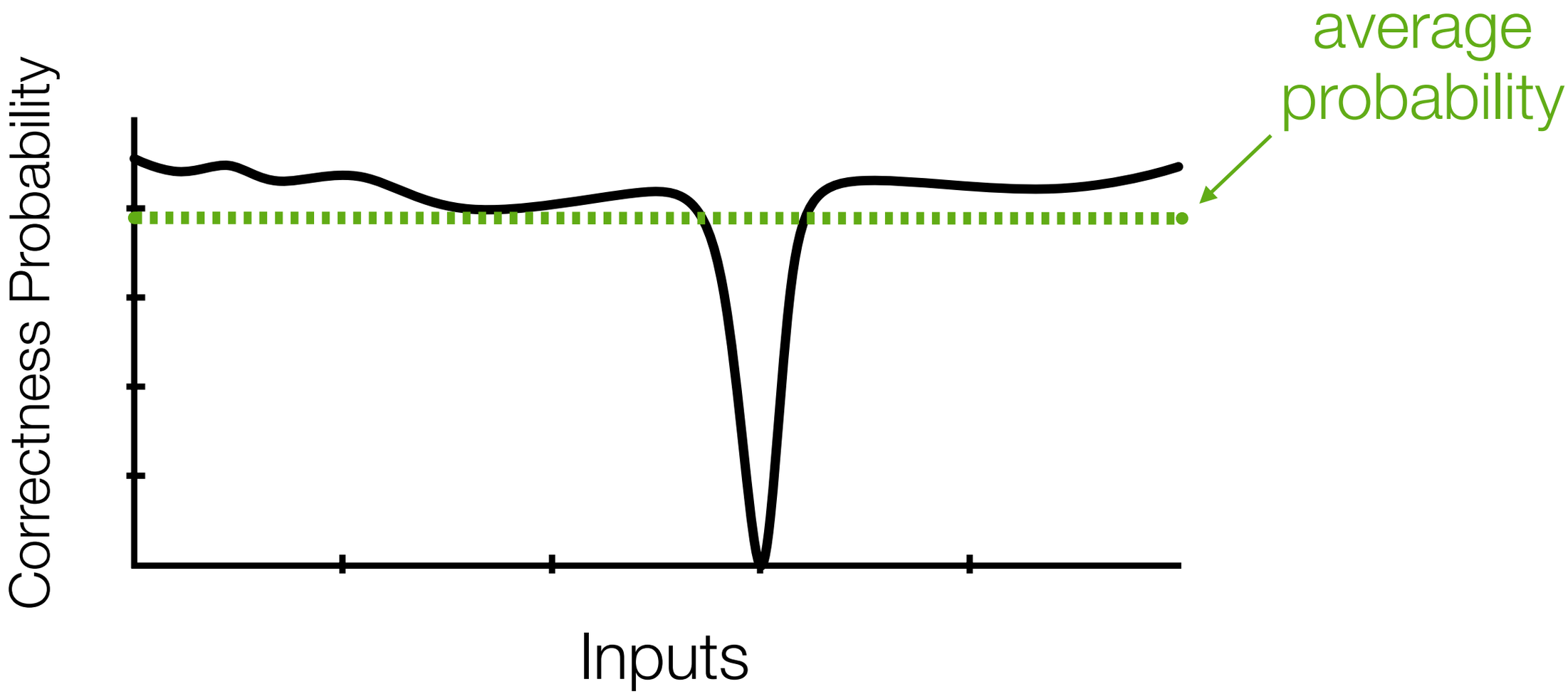
⋮


Assisted approximate programming
Statistical inference
Cheap check generation

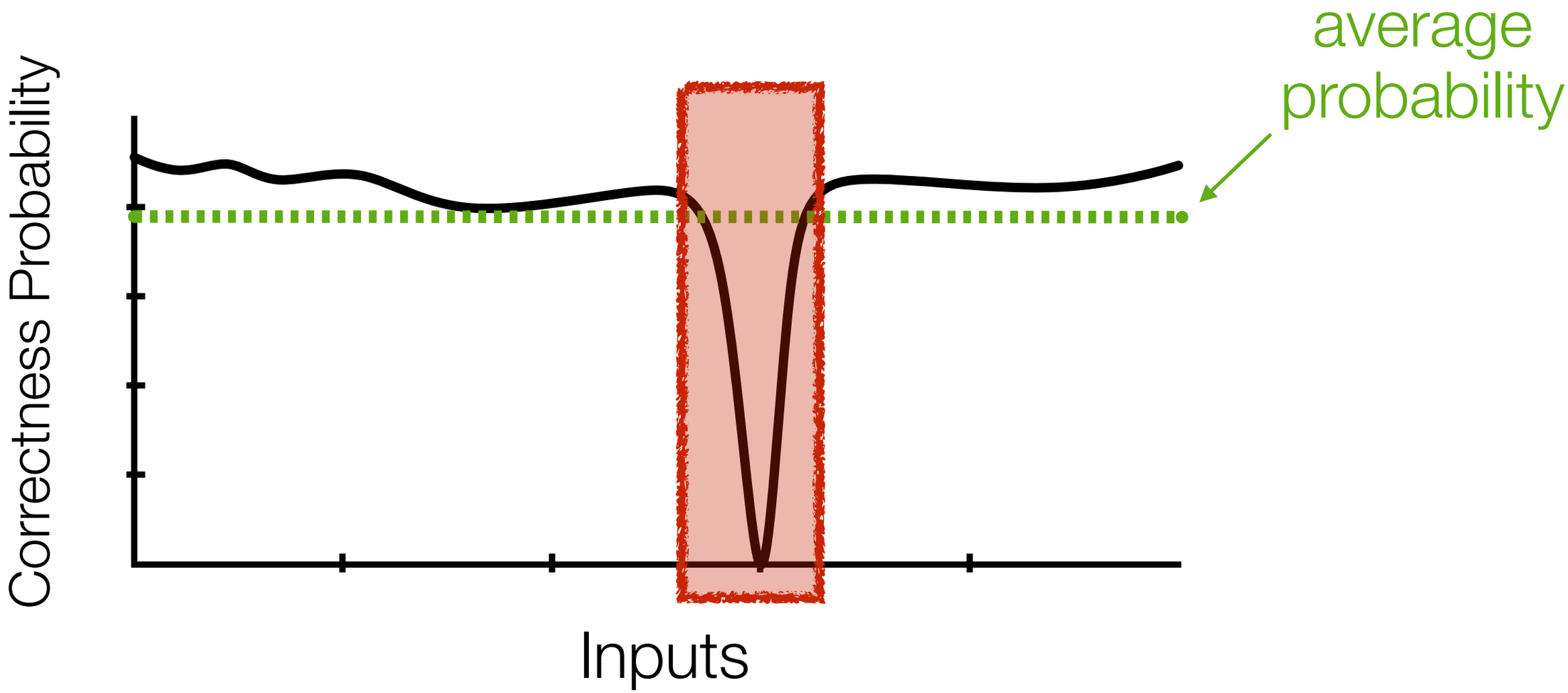
Quality:
the fantasy



Quality:
the reality



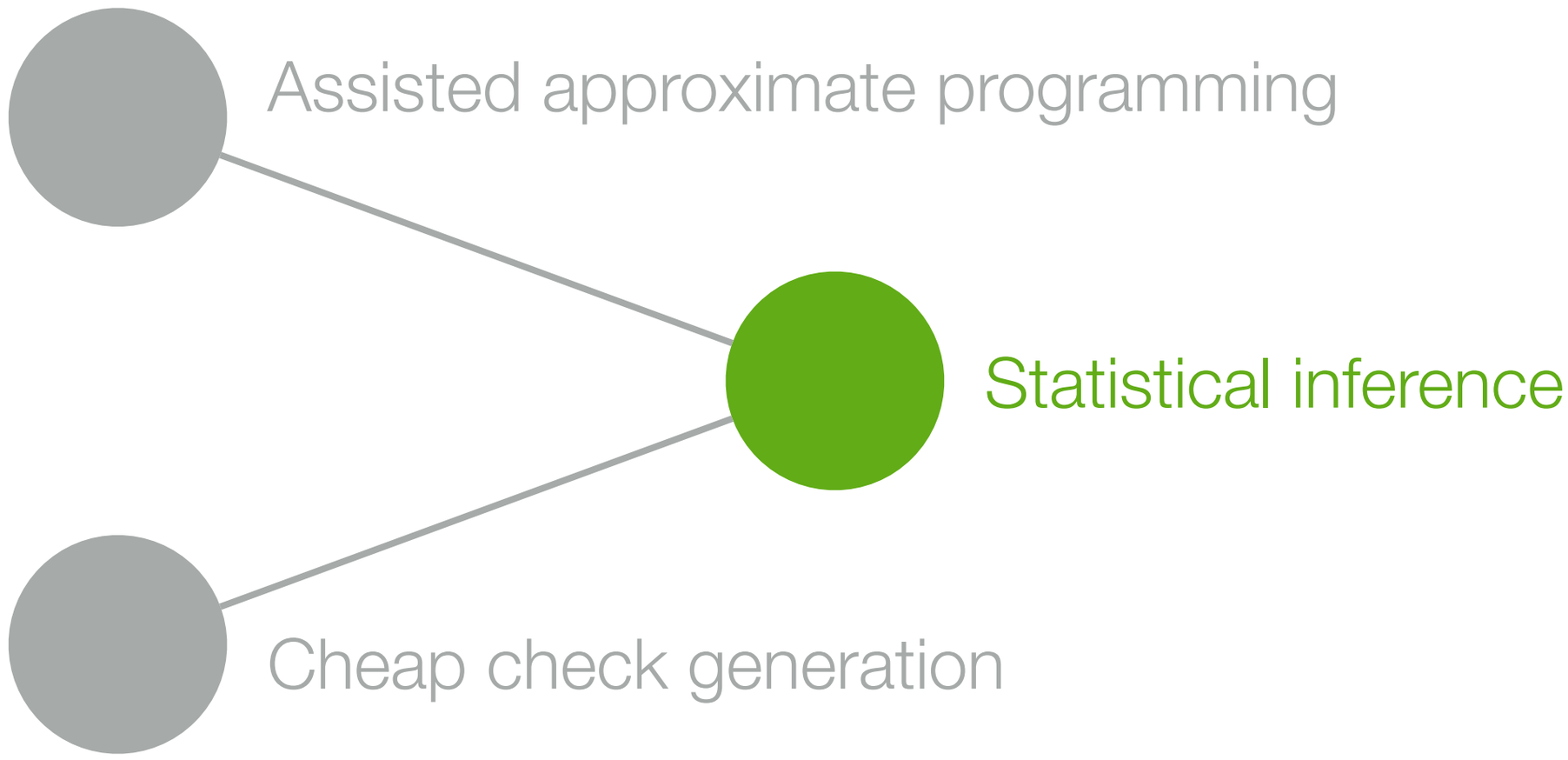
Cheap checks
to fall back to precise execution



Cheap checks

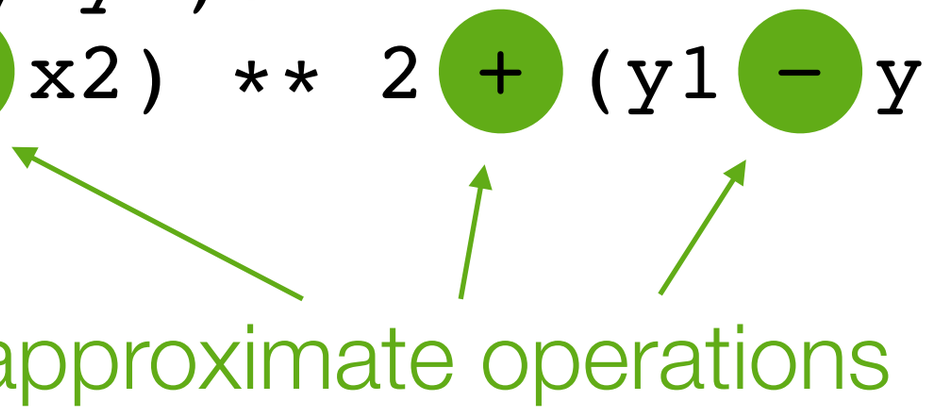
$f \quad f'$ $\Pr[d(f(x), f'(x)) \leq b] \geq p$

x s.t. $\Pr[d(f(x), f'(x)) \leq b] \geq p$


Assisted approximate programming
Statistical inference
Cheap check generation

Approximate program

```
def dist(x1, y1, x2, y2):
  return sqrt((x1 - x2) ** 2 + (y1 - y2) ** 2)
```



Approximate program → probabilistic program

```
def dist(x1, y1, x2, y2):
    return sqrt((x1 - x2 + error()) ** 2
            + (y1 - y2 + error()) ** 2)
            + error()
```

Assisted approximate programming as statistical inference

```
def dist(x1, y1, x2, y2):
  return sqrt((x1 - x2 + error(?)) ** 2
          + (y1 - y2 + error(?)) ** 2)
          + error(?)
```

s.t. $\Pr[d(f(x), f'(x)) \leq b] \geq p$

Cheap check generation as statistical inference

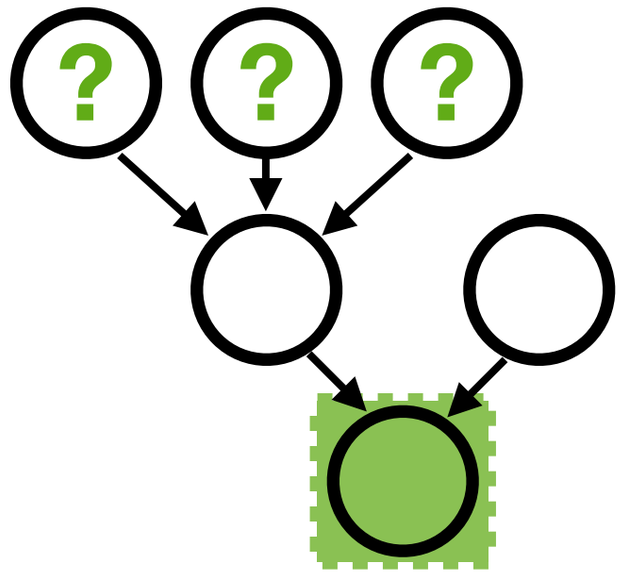
```
x1 = dist(?)
y1 = dist(?)
x2 = dist(?)
y2 = dist(?)

def dist(x1, y1, x2, y2):
  return sqrt((x1 - x2 + error()) ** 2
          + (y1 - y2 + error()) ** 2)
          + error()
```

s.t. $\Pr[d(f(x), f'(x)) \leq b] \geq p$

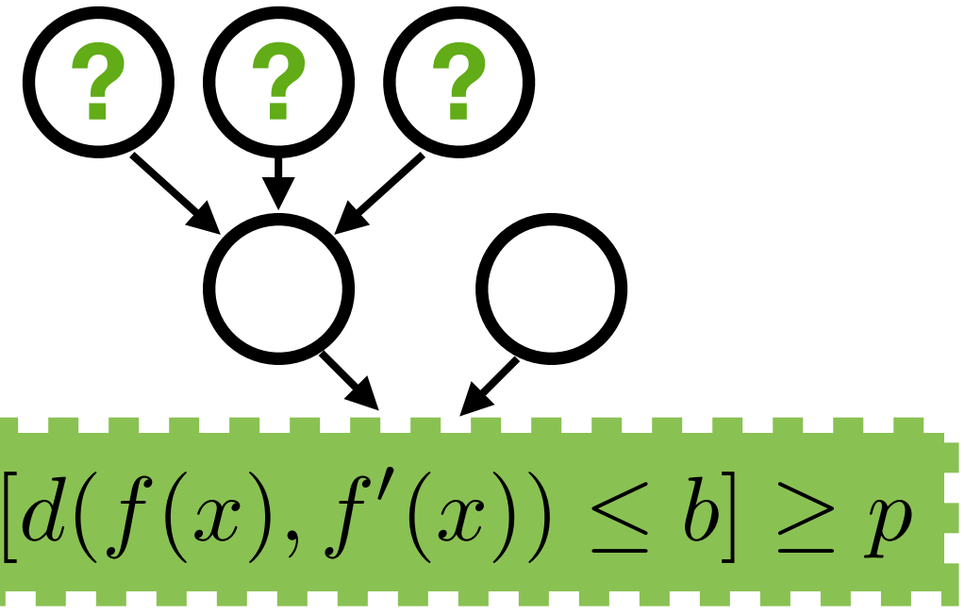
First steps

First steps:
translate to a probability distribution

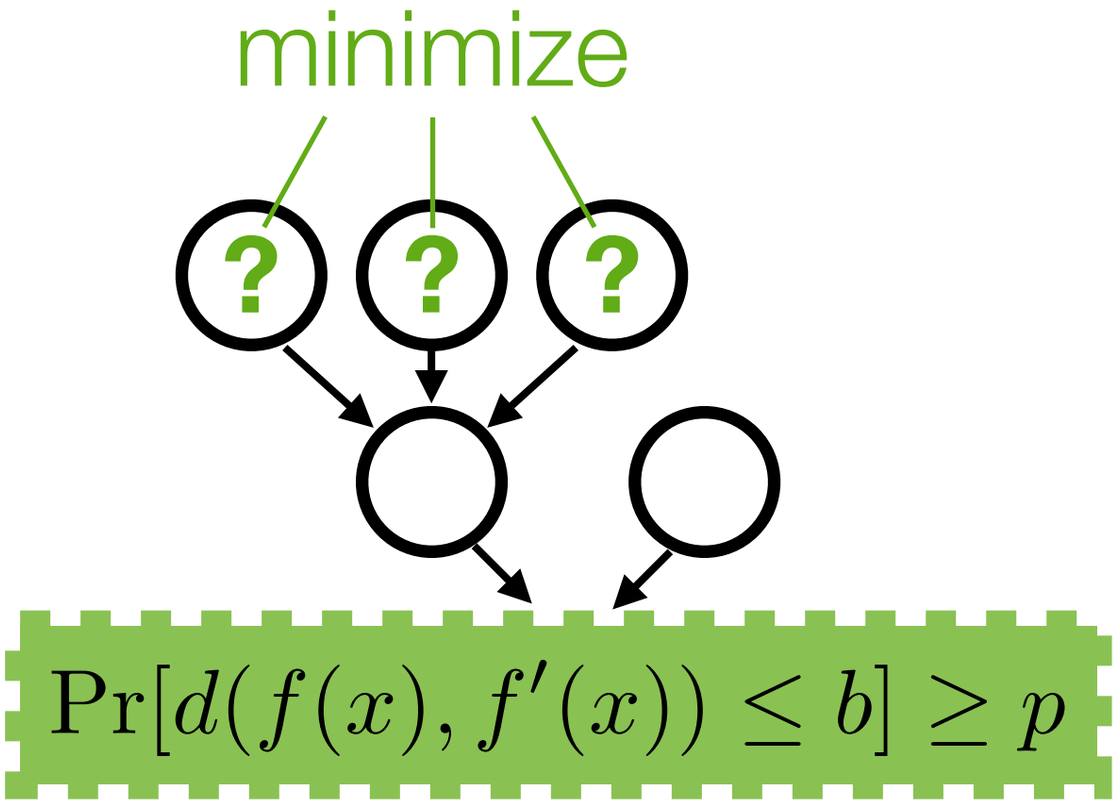
```
int p = 5;
int a = 7;
for (int x = 0..) {
   a += func(2);
   int z;
   z = p * 2;
   p += 4;
}
a /= 9;
func2(p);
a += func(2);
int y;
z = p * 22 + z;
p += 10;
```



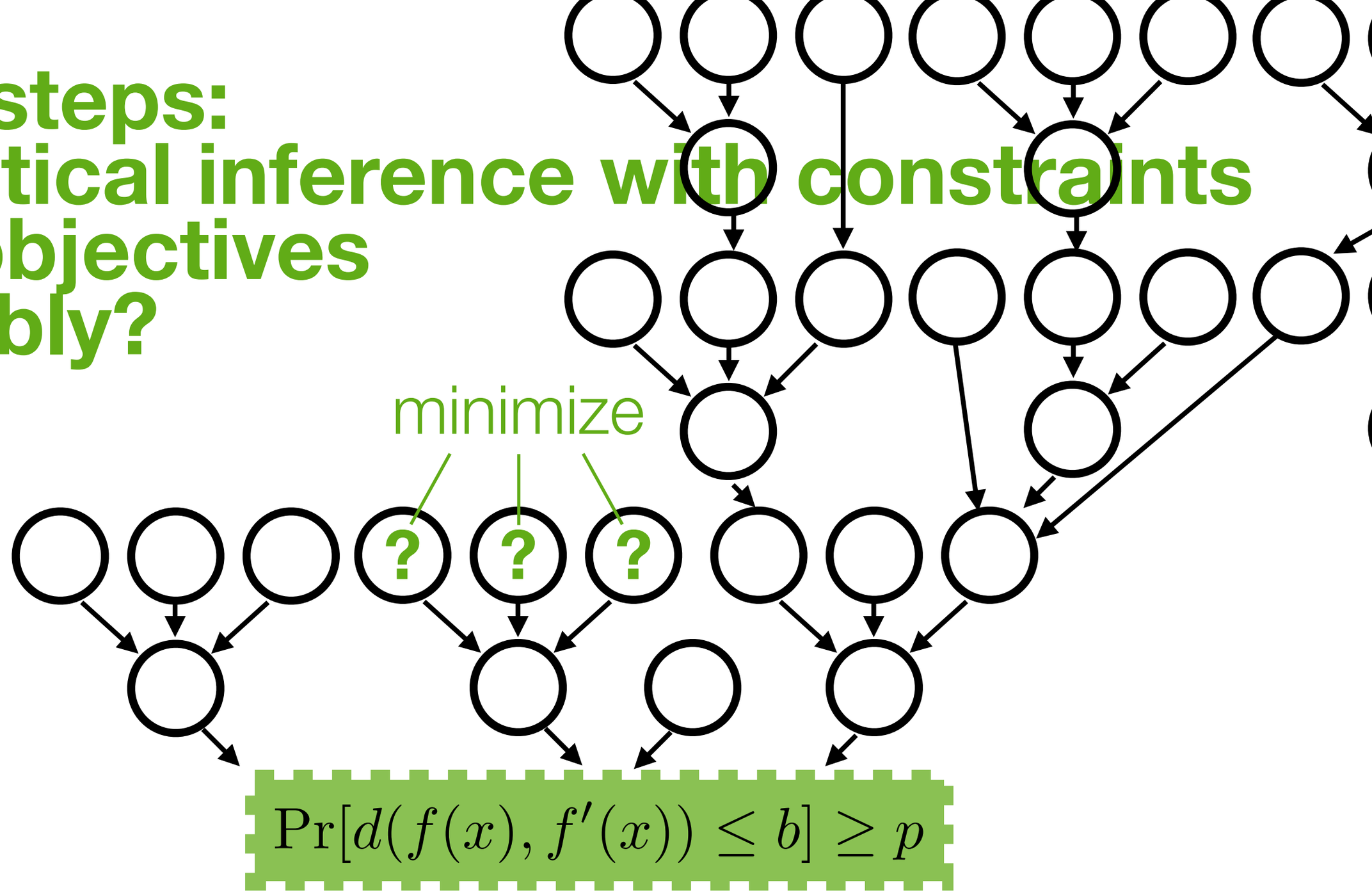
First steps:
statistical inference with constraints?



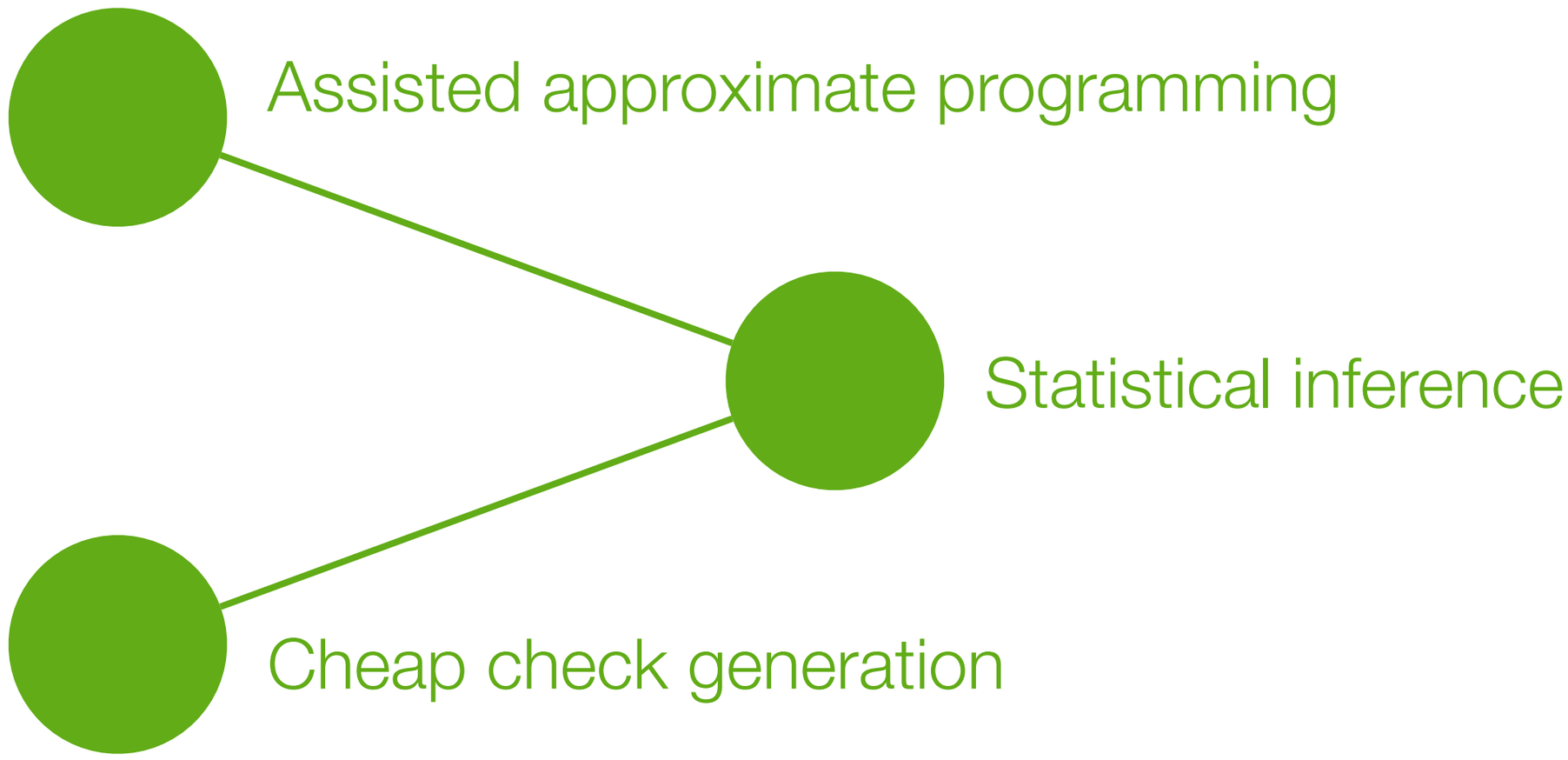
First steps: statistical inference with constraints and objectives?



First steps: statistical inference with constraints and objectives scalably?



$$\Pr[d(f(x), f'(x)) \leq b] \geq p$$


Assisted approximate programming
Statistical inference
Cheap check generation