Approximate Program Synthesis

James Bornholt
Emina Torlak
Luis Ceze
Dan Grossman

University of Washington

Writing approximate programs is hard

Precise
Implementation

Writing approximate programs is hard

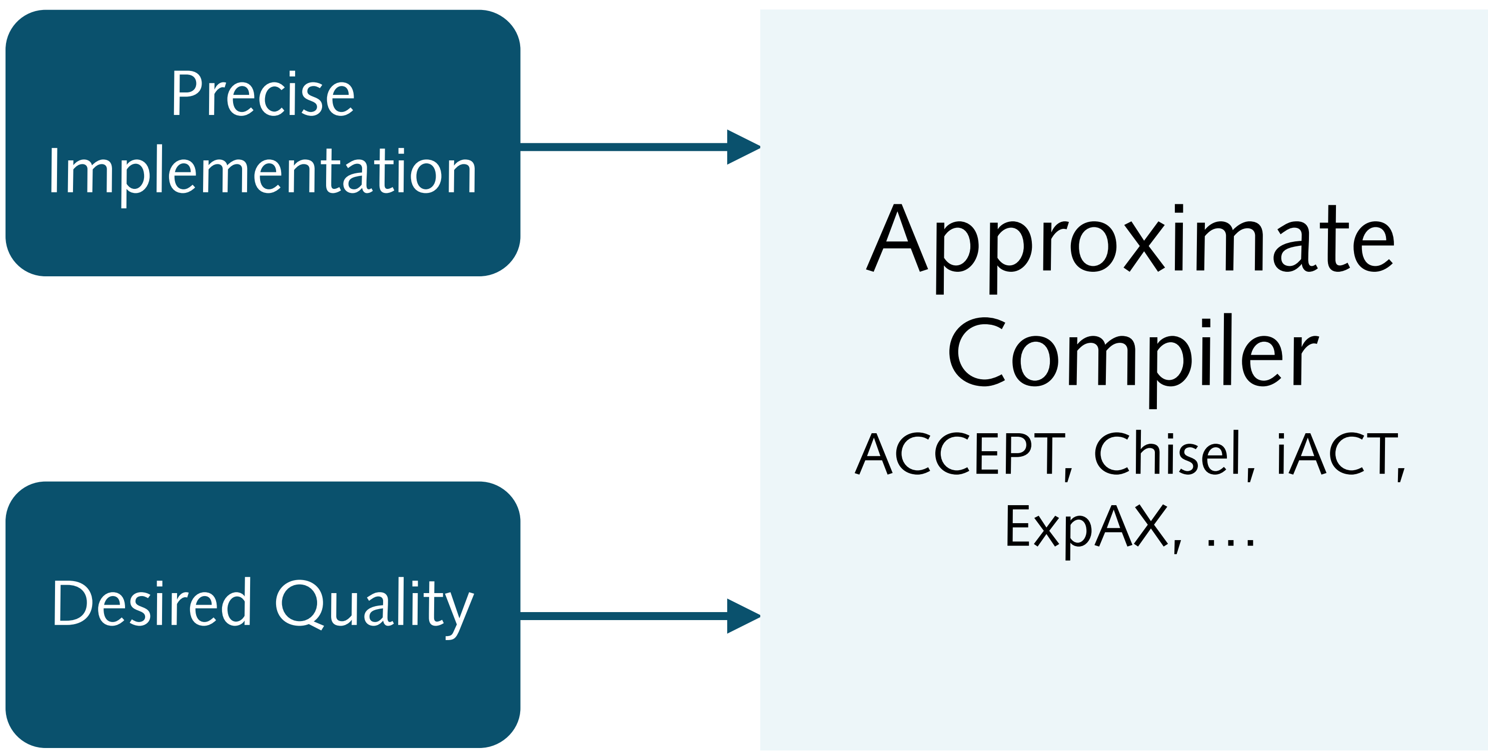
Precise
Implementation

Writing approximate programs is hard

Precise Implementation

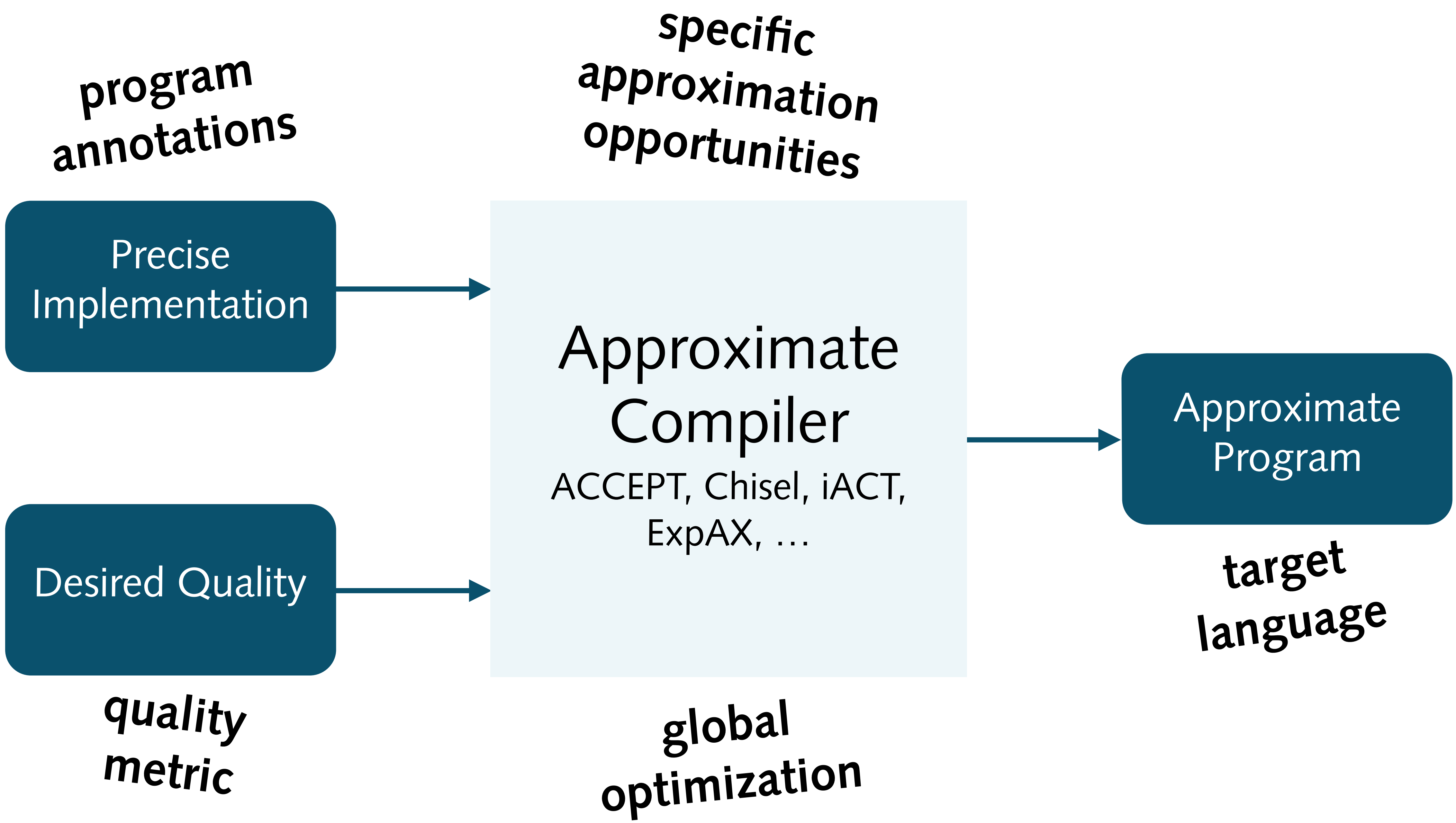
Desired Quality

Writing approximate programs is hard

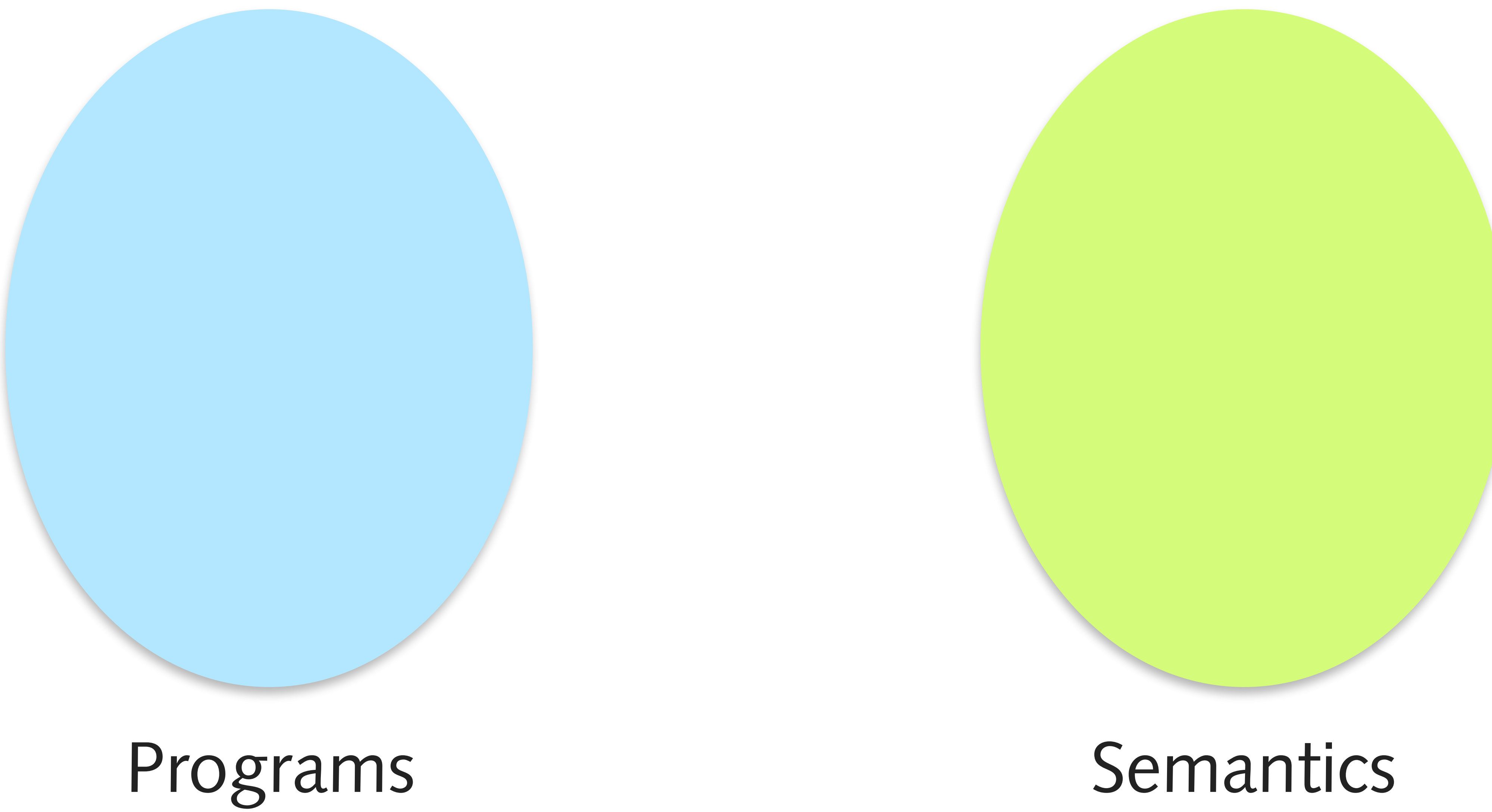


Writing approximate programs is hard

Writing approximate programs is hard

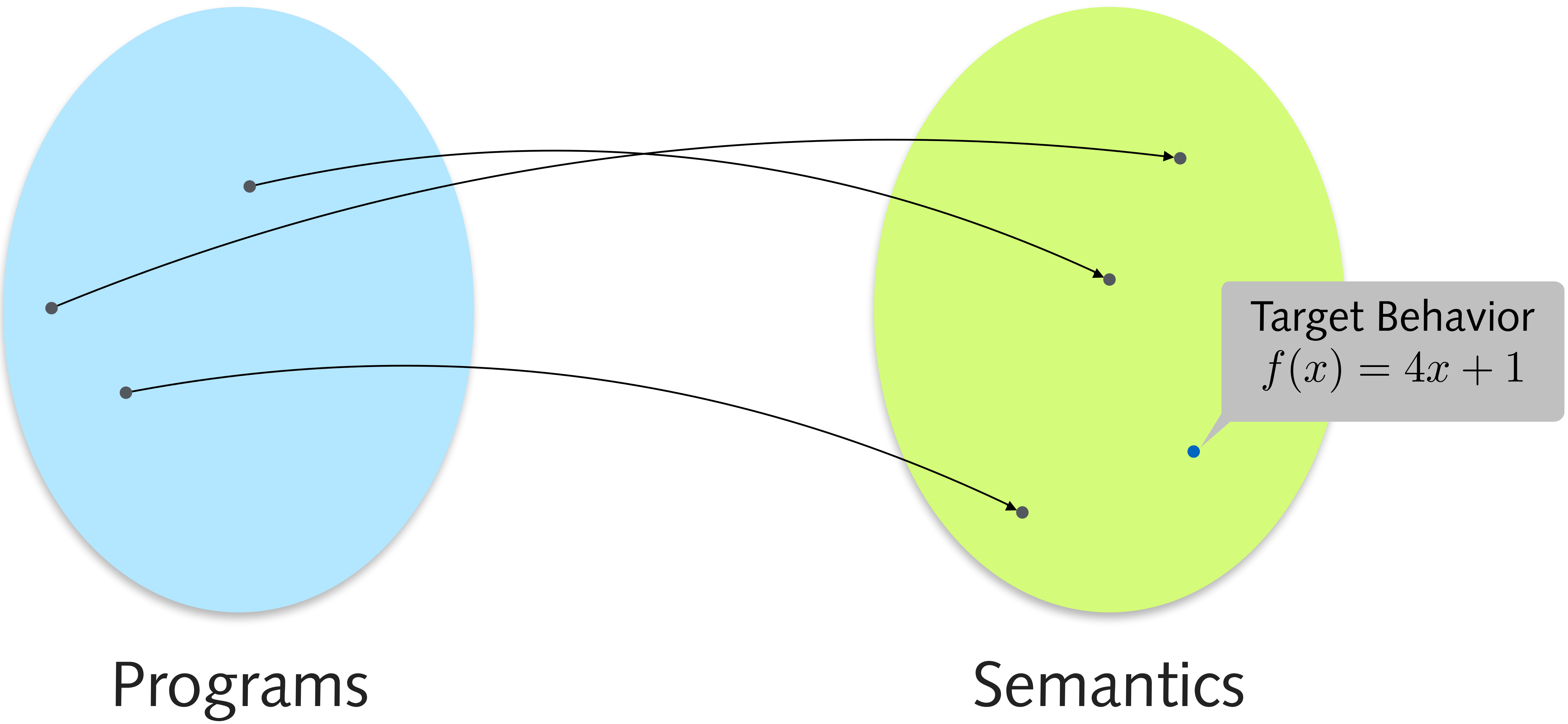


Synthesis: write programs automatically

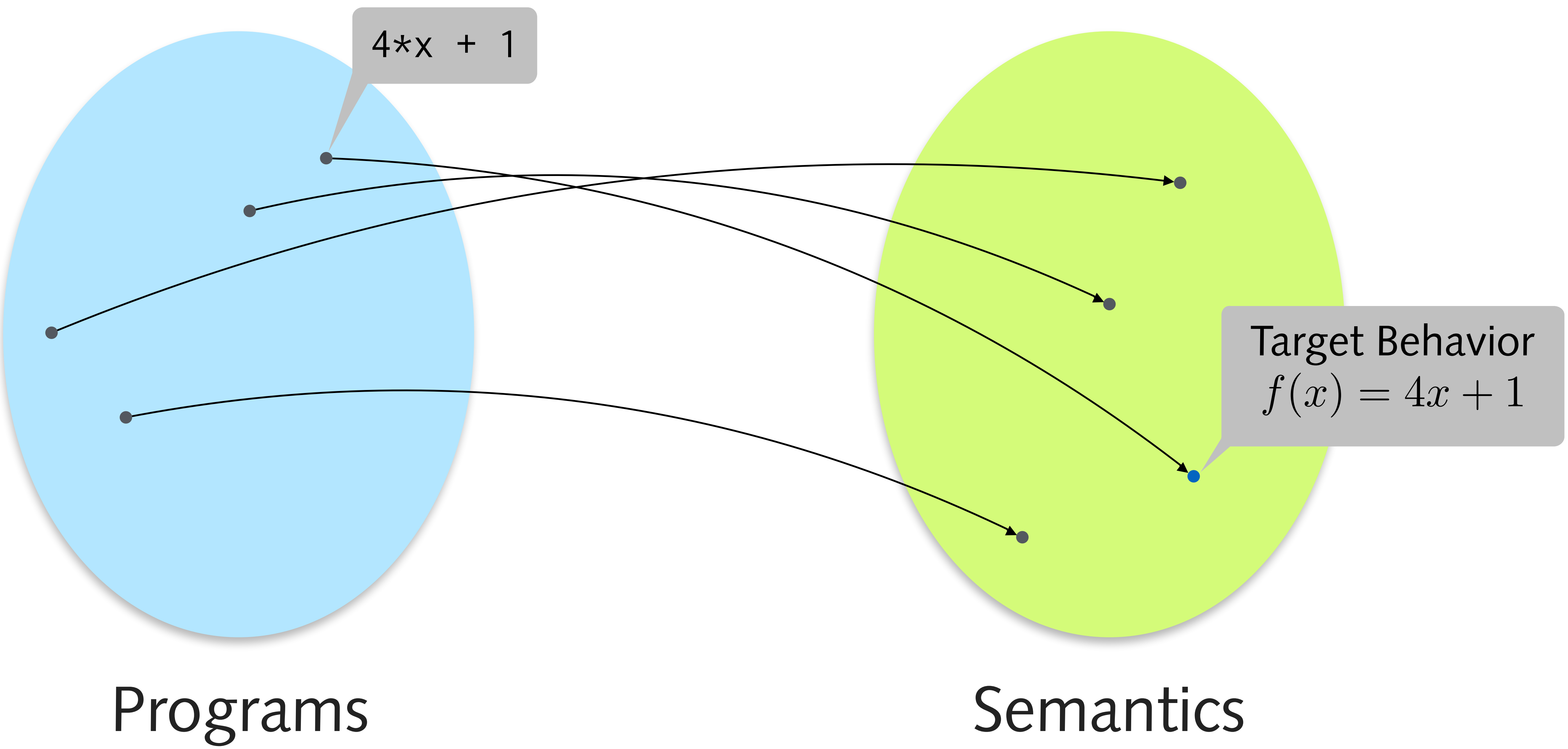


Synthesis: write programs automatically

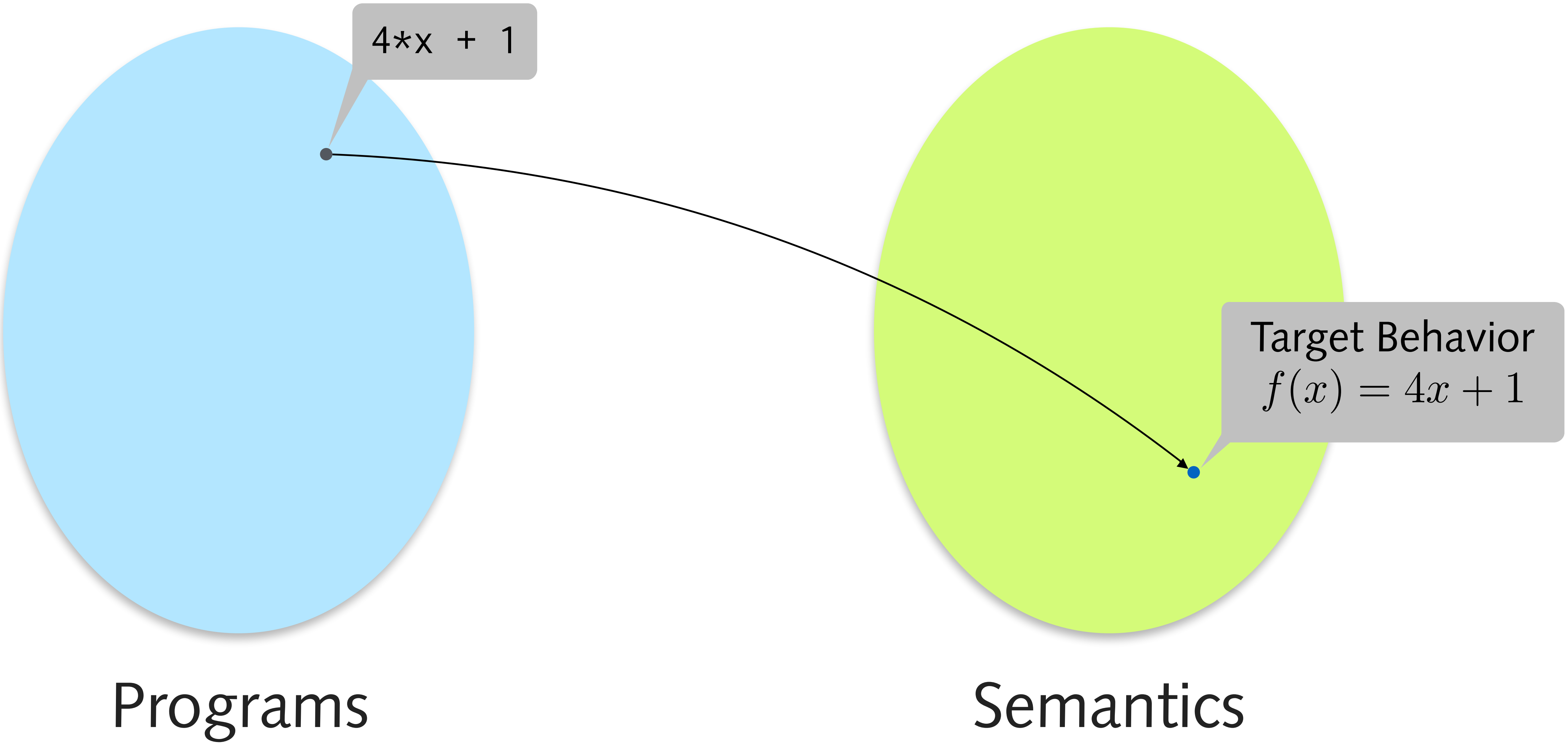
Synthesis: write programs automatically



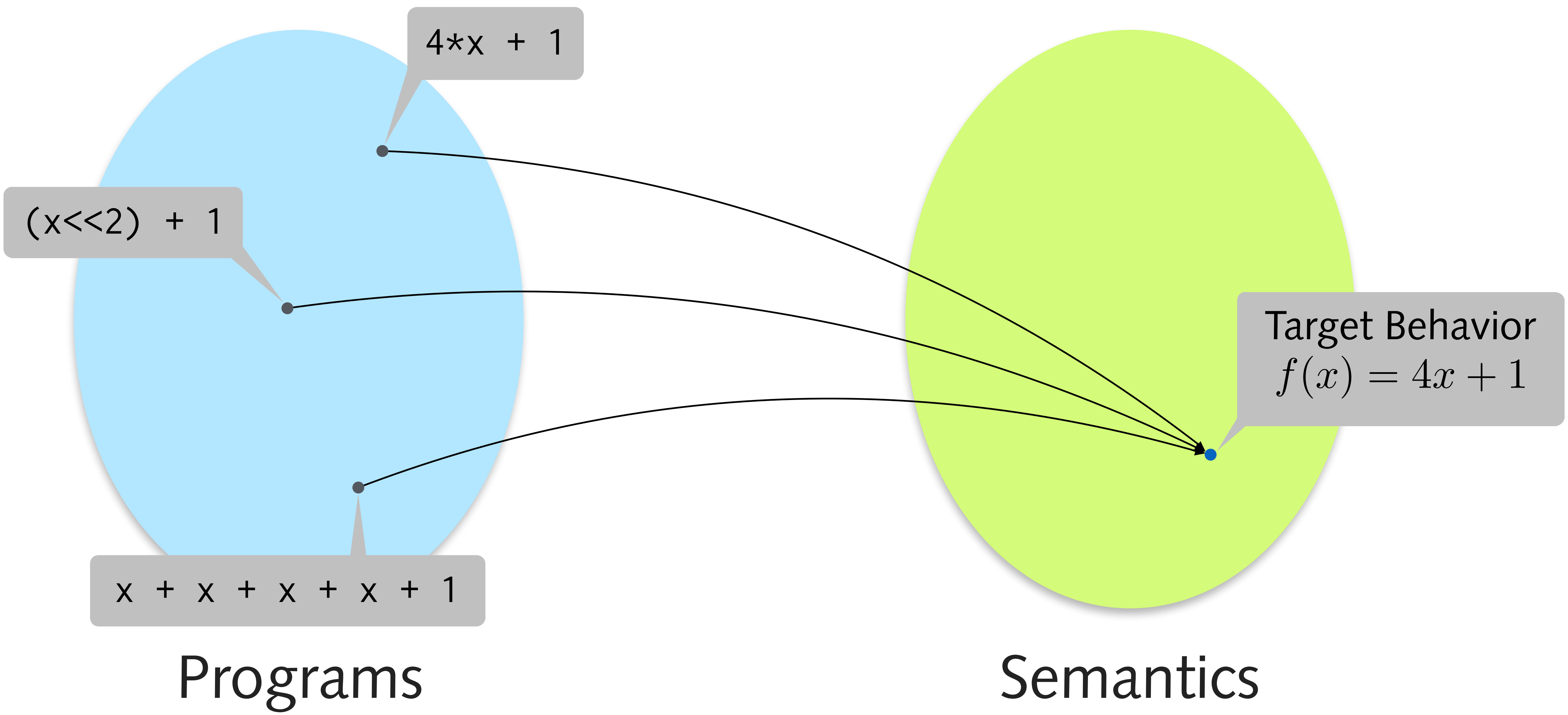
Synthesis: write programs automatically



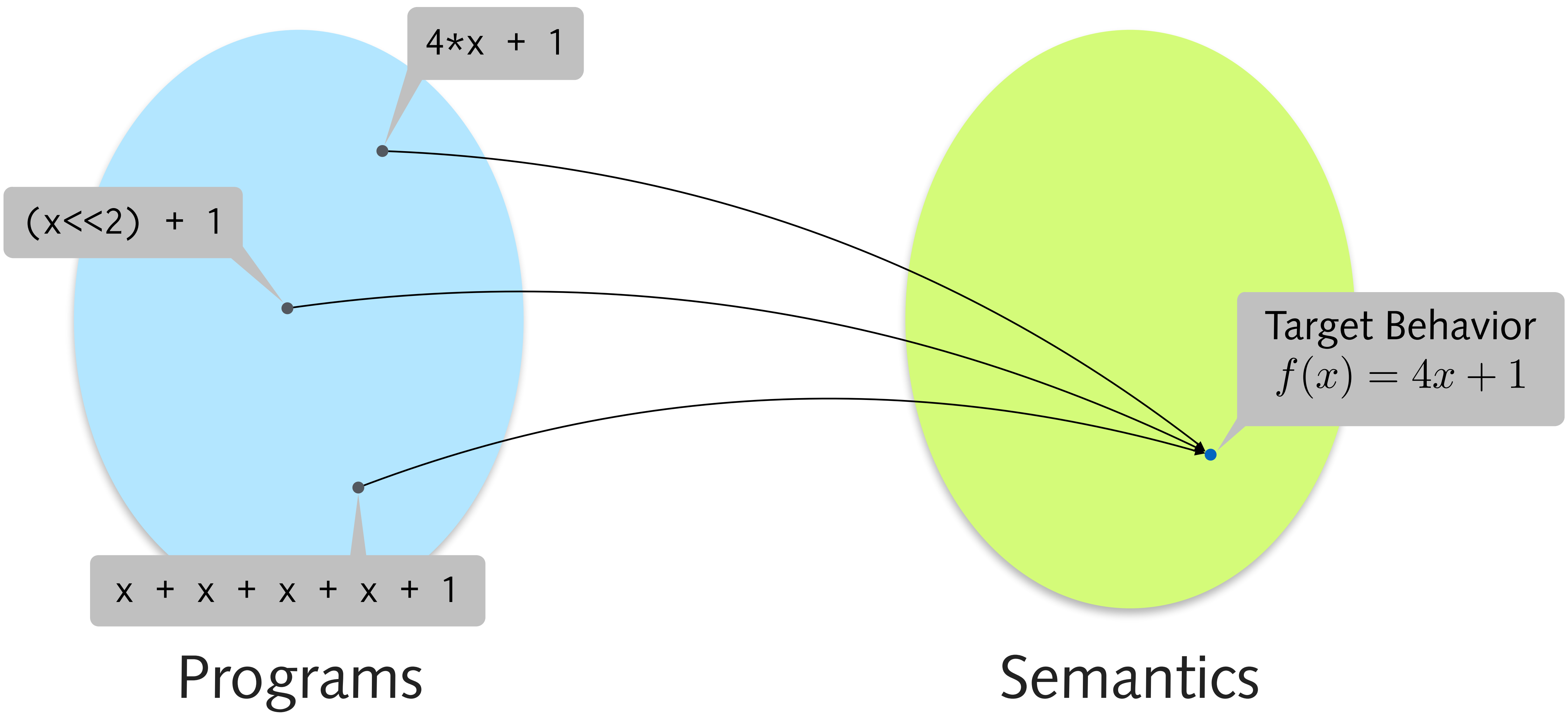
Synthesis: write programs automatically



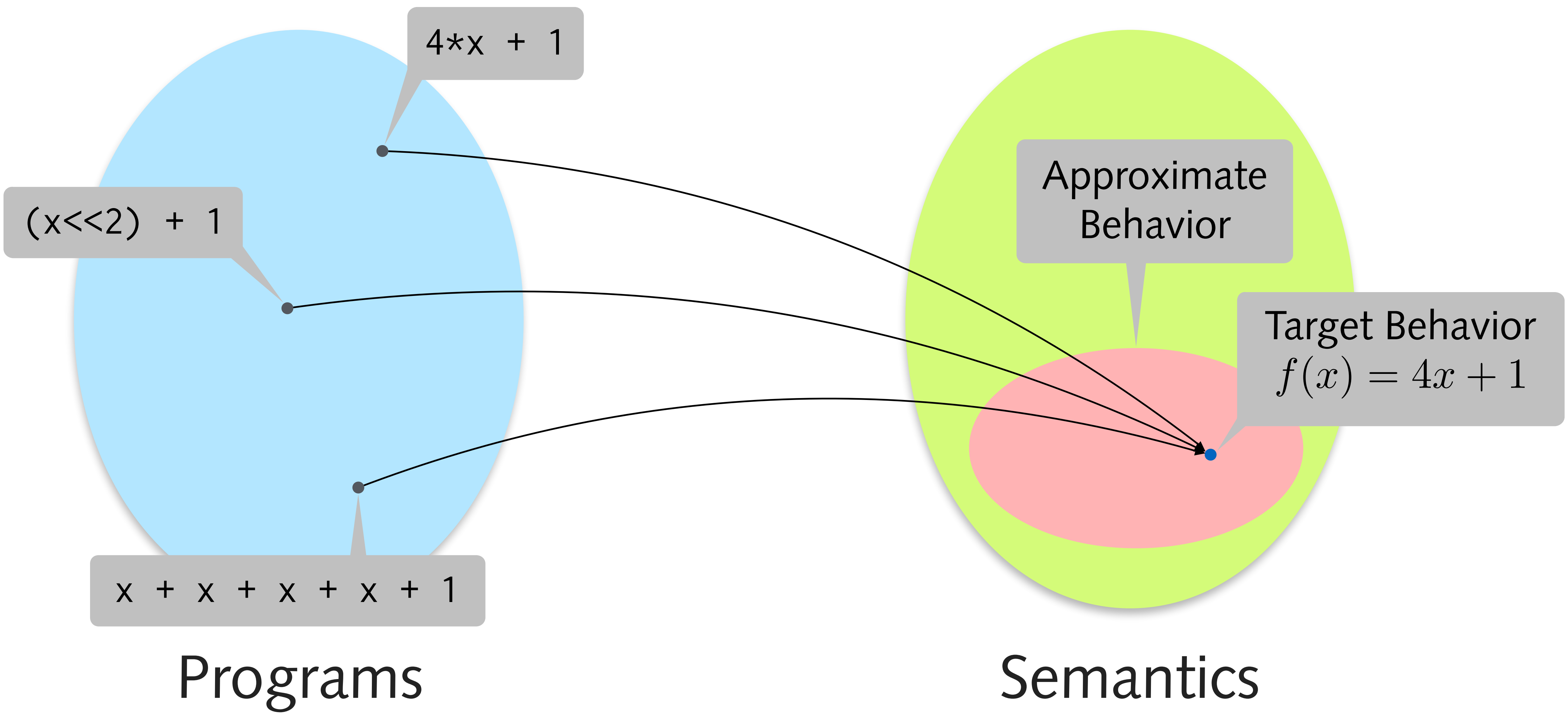
Synthesis: write programs automatically



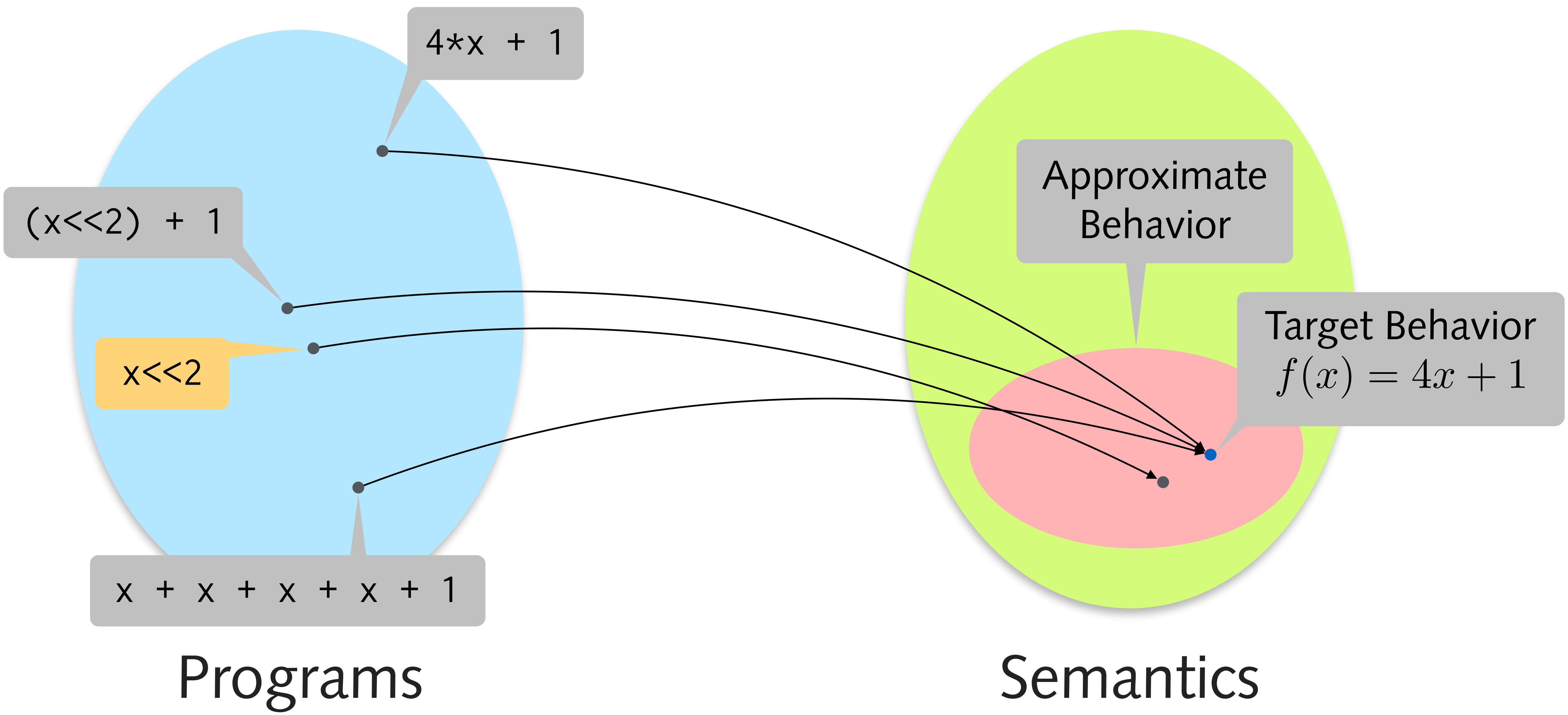
Synthesizing approximate programs



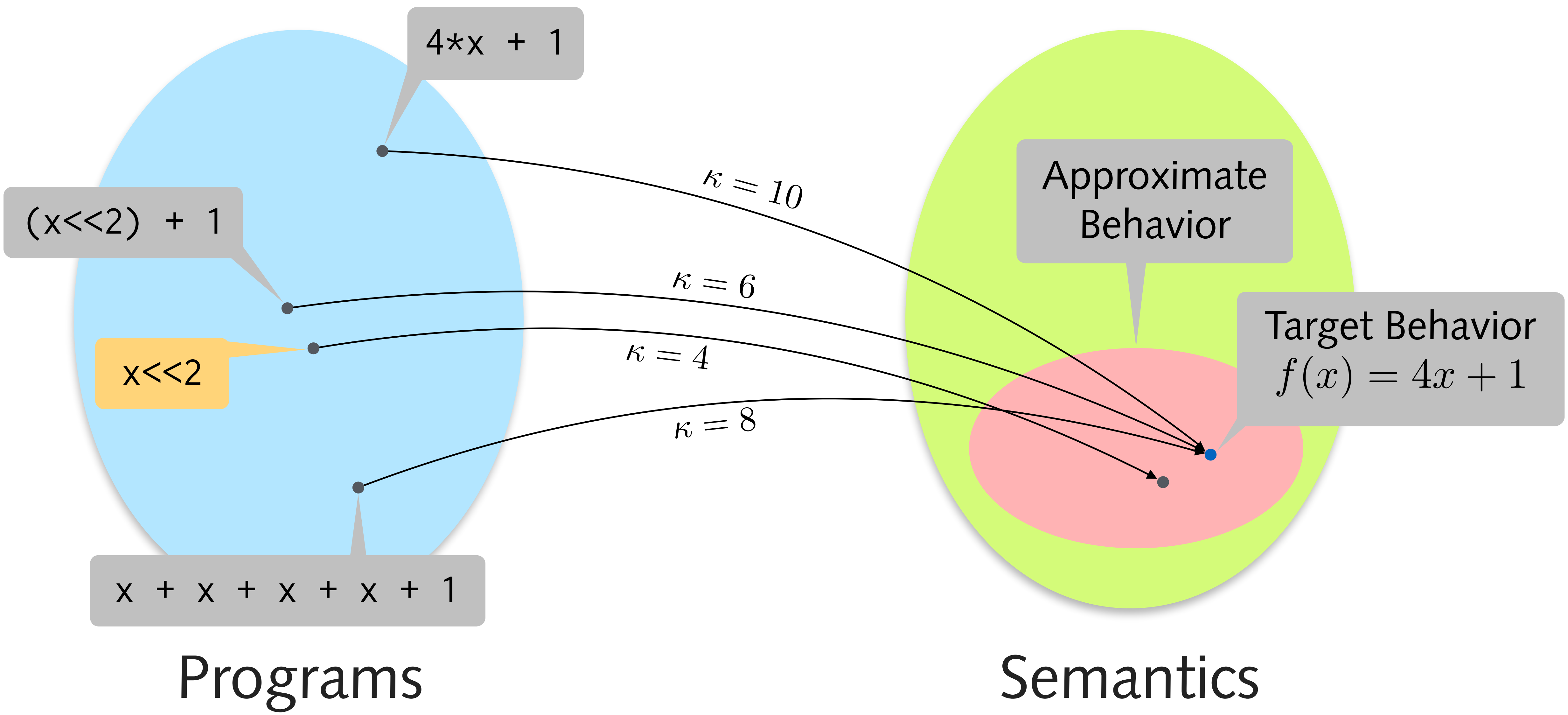
Synthesizing approximate programs



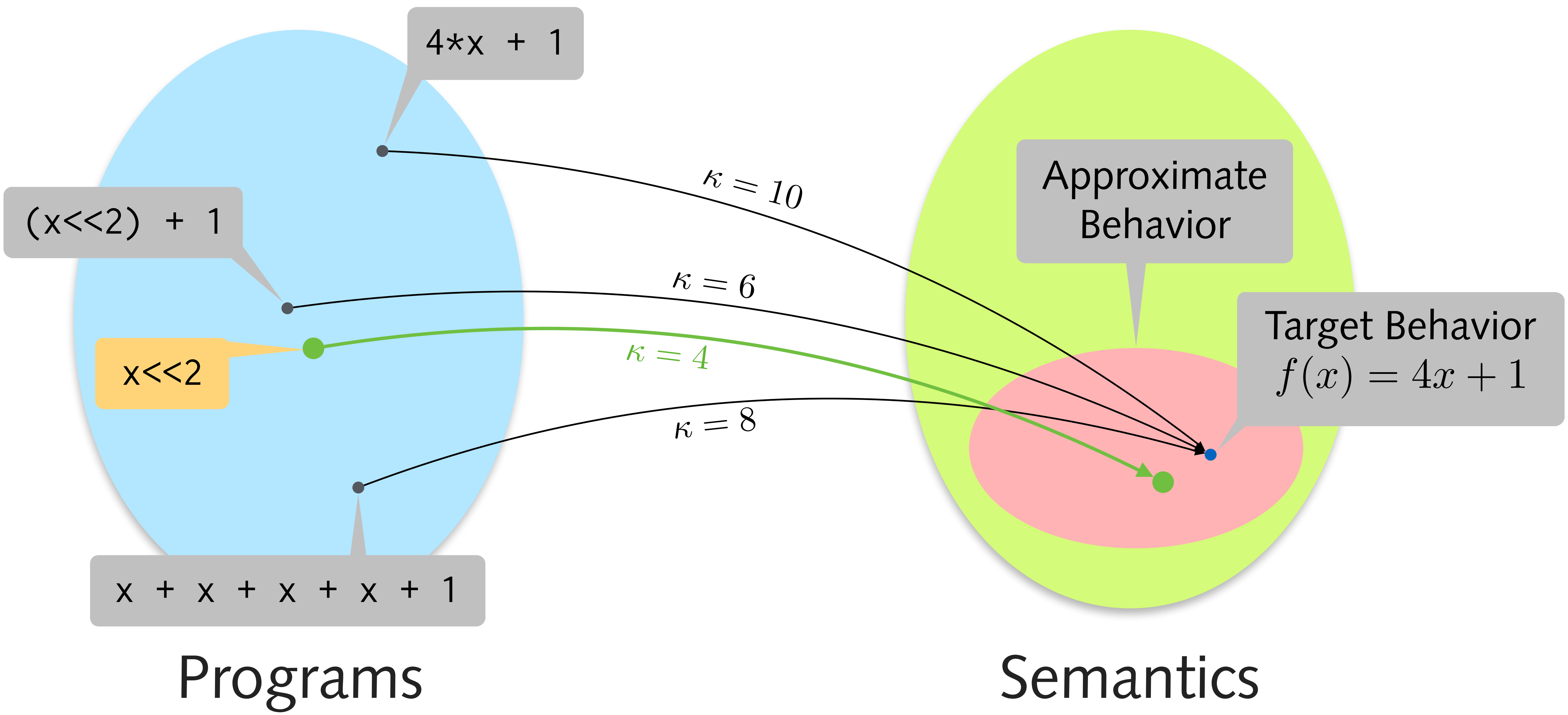
Synthesizing approximate programs



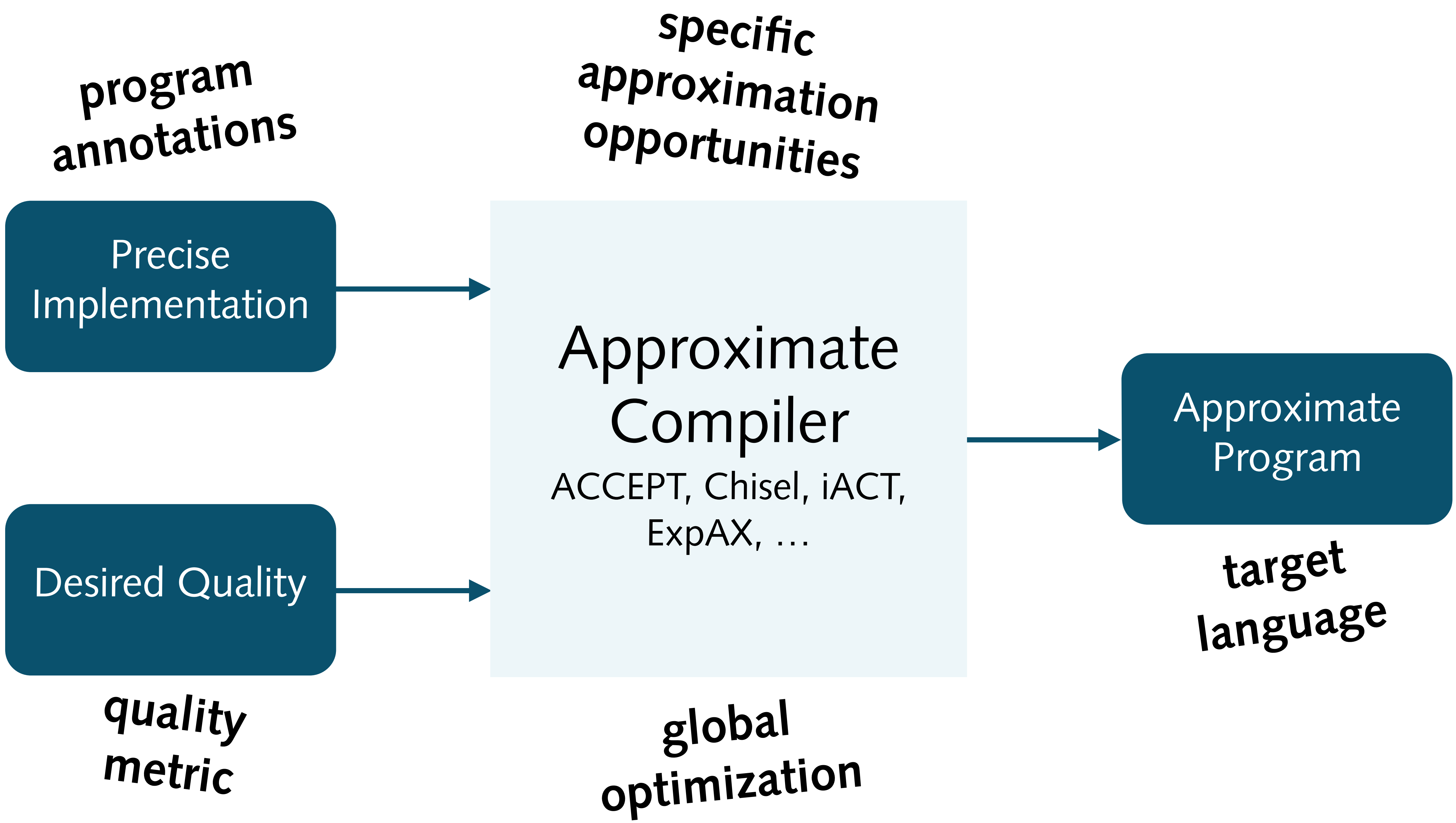
Synthesizing approximate programs



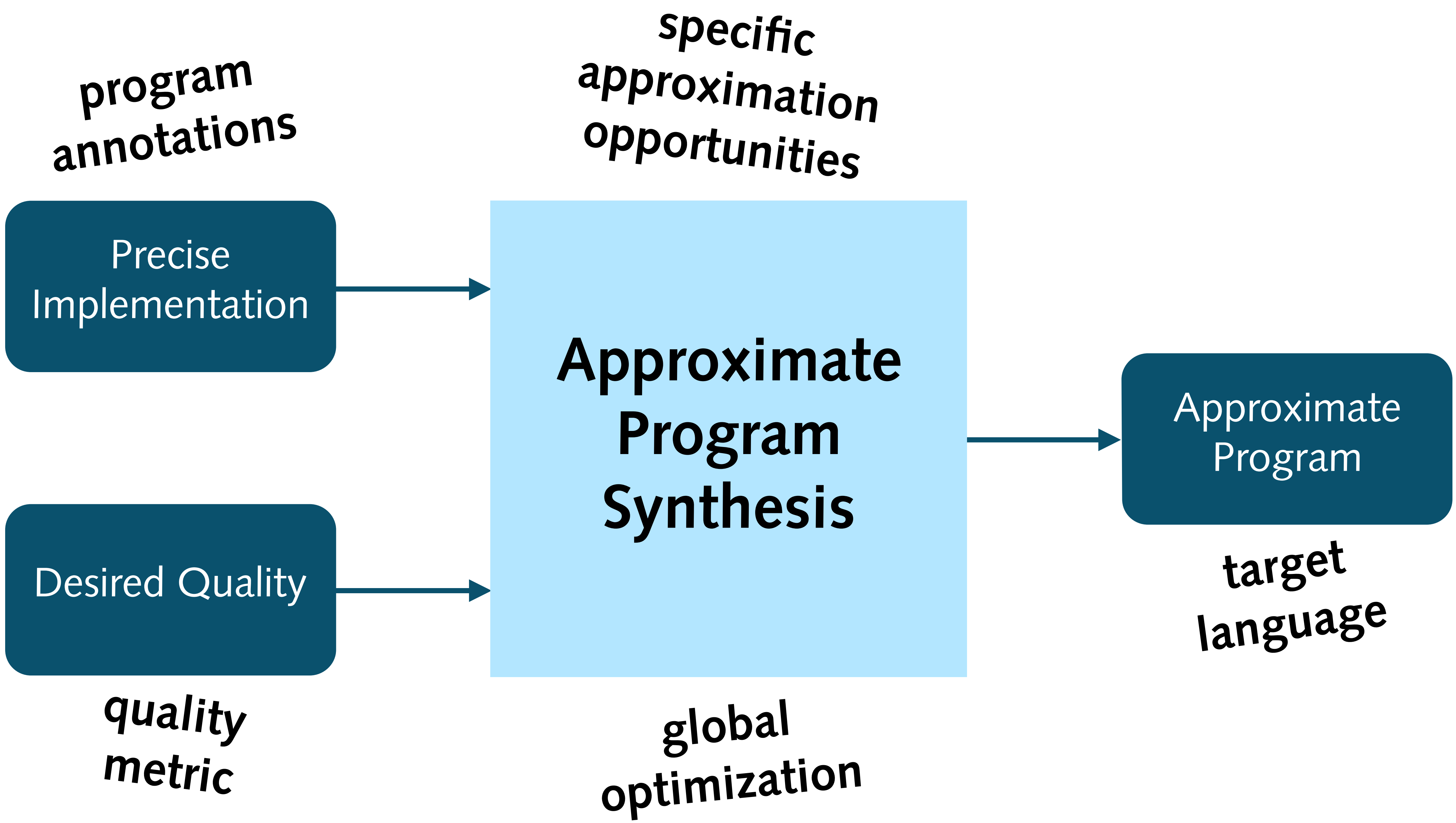
Synthesizing approximate programs



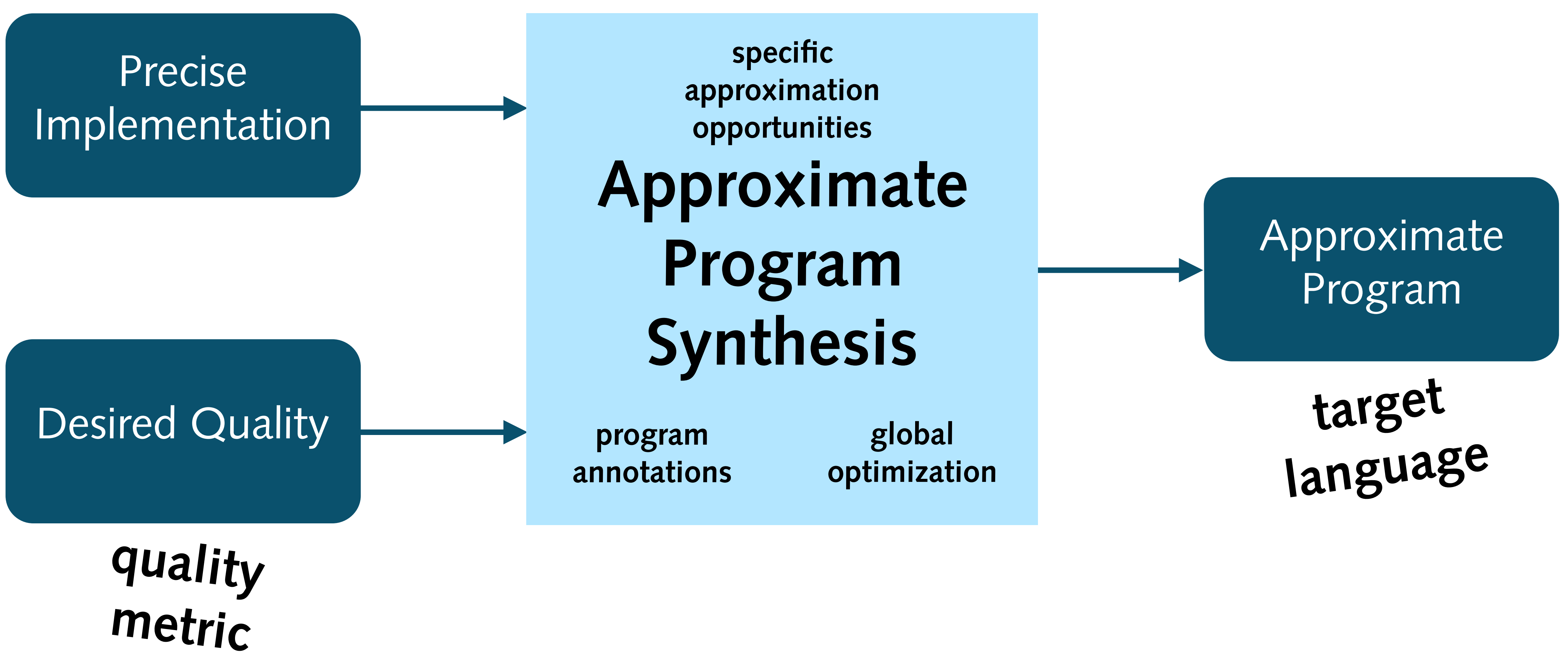
Synthesis automates approximation



Synthesis automates approximation



Synthesis automates approximation



Existing synthesizers don't scale enough

Approximate benchmarks

- fft
- kmeans
- inversek2j
- sobel

† Alur et al. *Syntax-Guided Synthesis*. FMCAD 2013.

Existing synthesizers don't scale enough

Approximate benchmarks

- fft
- kmeans
- inversek2j
- sobel

Off-the-shelf synthesizers†

- Symbolic
- Stochastic
- Brute-force

† Alur et al. *Syntax-Guided Synthesis*. FMCAD 2013.

Existing synthesizers don't scale enough

Approximate benchmarks

- fft
- kmeans
- inversek2j
- sobel

↓

Off-the-shelf synthesizers[†]

- Symbolic
- Stochastic
- Brute-force

```
float dist(float p[3], float c[3]) {
    float r = 0;
    r += (p[0] - c[0])*(p[0] - c[0]);
    r += (p[1] - c[1])*(p[1] - c[1]);
    r += (p[2] - c[2])*(p[2] - c[2]);
    float ret = sqrt(r);
    return ret;
}

float f(float p[3], float c[3]) {
    ??
}
```

[†] Alur et al. *Syntax-Guided Synthesis*. FMCAD 2013.

Existing synthesizers don't scale enough

Approximate benchmarks

- fft
- kmeans
- inversek2j
- sobel

Off-the-shelf synthesizers†

- Symbolic
- Stochastic
- Brute-force

```
float dist(float p[3], float c[3]) {
    float r = 0;
    r += (p[0] - c[0])*(p[0] - c[0]);
    r += (p[1] - c[1])*(p[1] - c[1]);
    r += (p[2] - c[2])*(p[2] - c[2]);
    float ret = sqrt(r);
    return ret;
}

float f(float p[3], float c[3]) {
    ??
}
```

```
+ - * / & | ^
<< >> …
```

† Alur et al. *Syntax-Guided Synthesis*. FMCAD 2013.

Existing synthesizers don't scale enough

Approximate benchmarks

- fft
- kmeans
- inversek2j
- sobel

Off-the-shelf synthesizers†

- Symbolic
- Stochastic
- Brute-force

```
float dist(float p[3], float c[3]) {
    float r = 0;
    r += (p[0] - c[0])*(p[0] - c[0]);
    r += (p[1] - c[1])*(p[1] - c[1]);
    r += (p[2] - c[2])*(p[2] - c[2]);
    float ret = sqrt(r);
    return ret;
}

float f(float p[3], float c[3]) {
    ??
}
```

```
+ - * / & | ^
<< >> …
```

assert $\forall p, c.\ |\texttt{dist}(p, c) - \texttt{f}(p, c)| < 50\%$

Reference program

Program being synthesized

† Alur et al. *Syntax-Guided Synthesis*. FMCAD 2013.

Existing synthesizers don't scale enough

Approximate benchmarks

- fft
- kmeans
- inversek2j
- sobel

Off-the-shelf synthesizers[†]

- Symbolic
- Stochastic
- Brute-force

```
float dist(float p[3], float c[3]) {
    float r = 0;
    r += (p[0] - c[0])*(p[0] - c[0]);
    r += (p[1] - c[1])*(p[1] - c[1]);
    r += (p[2] - c[2])*(p[2] - c[2]);
    float ret = sqrt(r);
    return ret;
}

float f(float p[3], float c[3]) {
    ??
}
```

`+ - * / & | ^ << >> …`

assert $\forall p, c.\ |\mathtt{dist}(p, c) - \mathtt{f}(p, c)| < 50\%$

Reference program

Program being synthesized

† Alur et al. *Syntax-Guided Synthesis*. FMCAD 2013.

Existing synthesizers don't scale enough

Programs

```
float dist(float p[3], float c[3]) {
    float r = 0;
    r += (p[0] - c[0])*(p[0] - c[0]);
    r += (p[1] - c[1])*(p[1] - c[1]);
    r += (p[2] - c[2])*(p[2] - c[2]);
    float ret = sqrt(r);
    return ret;
}

float f(float p[3], float c[3]) {
    ??
}
```

+ - * / & | ^
<< >> …

assert $\forall p, c.\ |\texttt{dist}(p, c) - \texttt{f}(p, c)| < 50\%$

Reference program

Program being synthesized

Existing synthesizers don't scale enough

Programs

```
float dist(float p[3], float c[3]) {
    float r = 0;
    r += (p[0] - c[0])*(p[0] - c[0]);
    r += (p[1] - c[1])*(p[1] - c[1]);
    r += (p[2] - c[2])*(p[2] - c[2]);
    float ret = sqrt(r);
    return ret;
}
```

```
float f(float p[3], float c[3]) {
    ??
}
```

```
+ - * / & | ^
<< >> …
```

assert $\forall p, c.\ |\texttt{dist}(p,c) - \texttt{f}(p,c)| < 50\%$

Reference program

Program being synthesized

Existing synthesizers don't scale enough

Programs

```
float dist(float p[3], float c[3]) {
    float r = 0;
    r += (p[0] - c[0])*(p[0] - c[0]);
    r += (p[1] - c[1])*(p[1] - c[1]);
    r += (p[2] - c[2])*(p[2] - c[2]);
    float ret = sqrt(r);
    return ret;
}
```

```
float f(float p[3], float c[3]) {
    ??
}
```

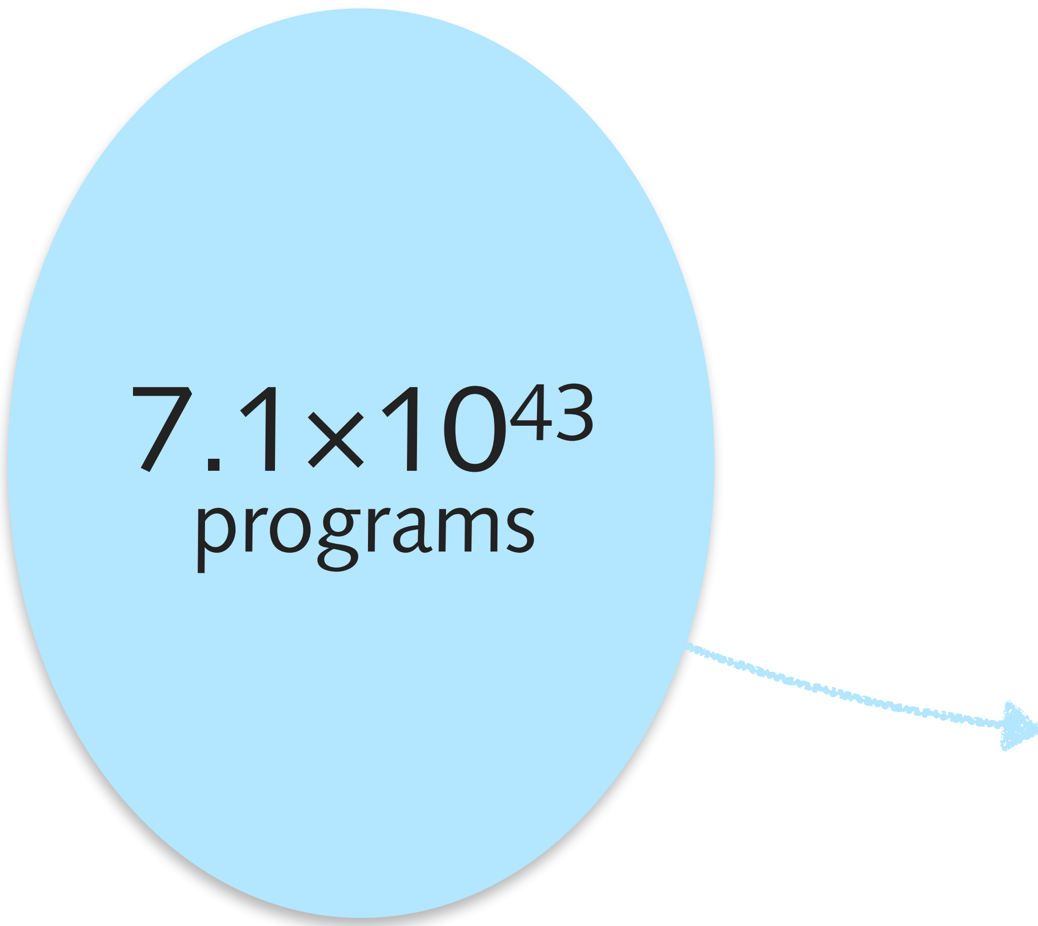
+ - * / & | ^
<< >> …

assert $\forall p, c.\ |\mathtt{dist}(p, c) - \mathtt{f}(p, c)| < 50\%$

Reference program

Program being synthesized

Use reference programs to guide synthesis



Programs

```
float dist(float p[3], float c[3]) {
    float r = 0;
    r += (p[0] - c[0])*(p[0] - c[0]);
    r += (p[1] - c[1])*(p[1] - c[1]);
    r += (p[2] - c[2])*(p[2] - c[2]);
    float ret = sqrt(r);
    return ret;
}
```

```
float f(float p[3], float c[3]) {
    ??
}
```

+ - * / & | ^
<< >> …

assert $\forall p, c.\ |\texttt{dist}(p, c) - \texttt{f}(p, c)| < 50\%$

Reference program

Program being synthesized

Use reference programs to guide synthesis

7.1×10^{43}
programs

Programs

```
float dist(float p[3], float c[3]) {
    float r = 0;
    r += (p[0] - c[0])*(p[0] - c[0]);
    r += (p[1] - c[1])*(p[1] - c[1]);
    r += (p[2] - c[2])*(p[2] - c[2]);
    float ret = sqrt(r);
    return ret;
}
```

```
float f(float p[3], float c[3]) {
    ??
}
```

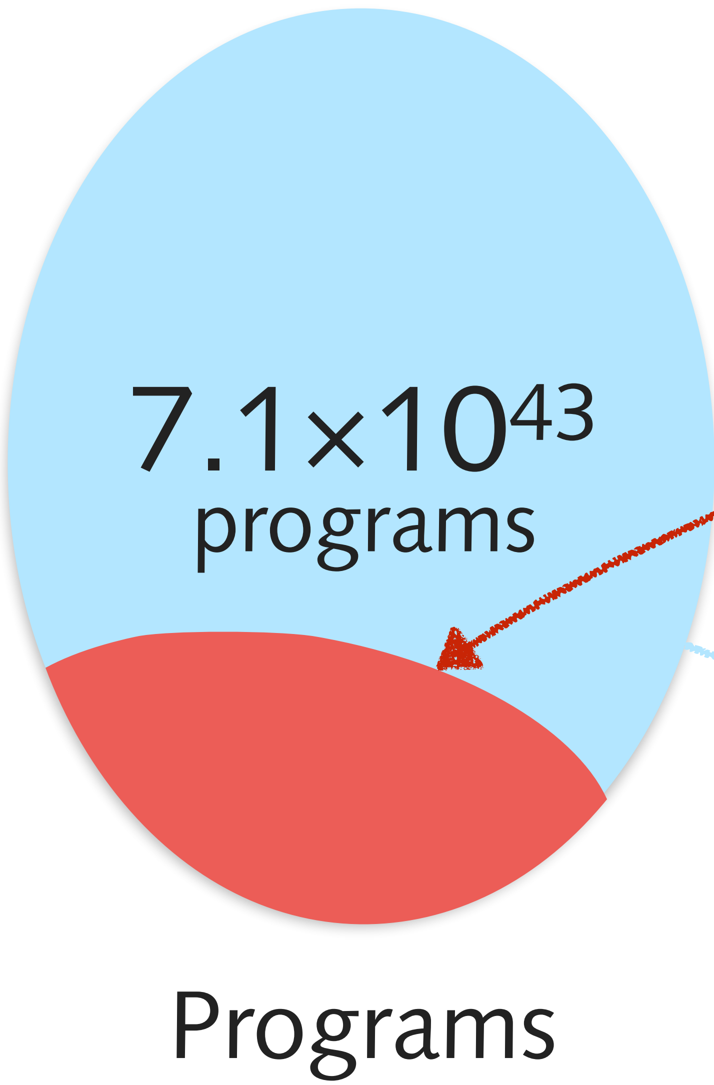
+ - * / & | ^
<< >> …

assert $\forall p, c.\ |\texttt{dist}(p, c) - \texttt{f}(p, c)| < 50\%$

Reference program

Program being synthesized

Use reference programs to guide synthesis




```
float dist(float p[3], float c[3]) {
    float r = 0;
    r += (p[0] - c[0])*(p[0] - c[0]);
    r += (p[1] - c[1])*(p[1] - c[1]);
    r += (p[2] - c[2])*(p[2] - c[2]);
    float ret = sqrt(r);
    return ret;
}
```

```
float f(float p[3], float c[3]) {
    ??
}
```

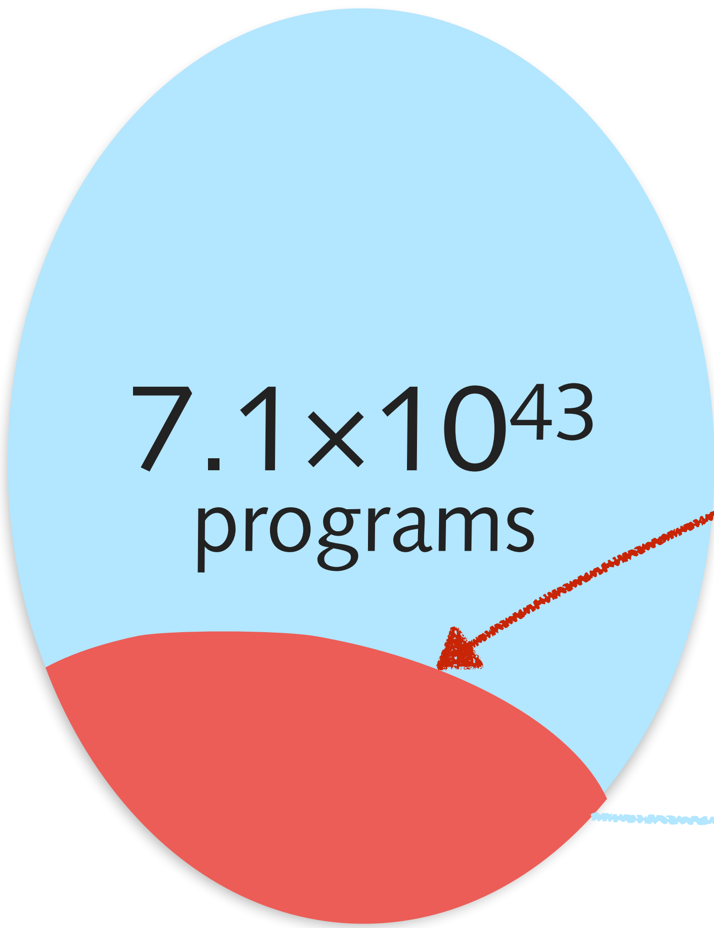
```
+ - * / & | ^
<< >> …
```

assert $\forall p, c.\ |\texttt{dist}(p, c) - \texttt{f}(p, c)| < 50\%$

Reference program

Program being synthesized

Use reference programs to guide synthesis



Programs

```
float dist(float p[3], float c[3]) {
    float r = 0;
    r += (p[0] - c[0])*(p[0] - c[0]);
    r += (p[1] - c[1])*(p[1] - c[1]);
    r += (p[2] - c[2])*(p[2] - c[2]);
    float ret = sqrt(r);
    return ret;
}
```

```
float f(float p[3], float c[3]) {
    ??
}
```

```
+ - * / & | ^
<< >> …
```

assert $\forall p, c.\ |\mathtt{dist}(p, c) - \mathtt{f}(p, c)| < 50\%$

Reference program

Program being synthesized

Synthesis produces good approximations

Spec: < 50% average error

Benchmark	Speedup	Error
fft_s	11.4×	21.3%
fft_c	12.0×	28.9%
dist3	1.6×	14.9%
$sobel_x$	10.6×	0%
$sobel_y$	10.7×	0%
$inversek2j_1$	34.8×	16.3%
$inversek2j_2$	10.0×	18.5%

Synthesis produces good approximations

Spec: < 50% average error

Benchmark	Speedup	Error
fft_s	11.4×	21.3%
fft_c	12.0×	28.9%
dist3	1.6×	14.9%
sobel_x	10.6×	0%
sobel_y	10.7×	0%
inversek2j_1	34.8×	16.3%
inversek2j_2	10.0×	18.5%

Missed compiler optimization

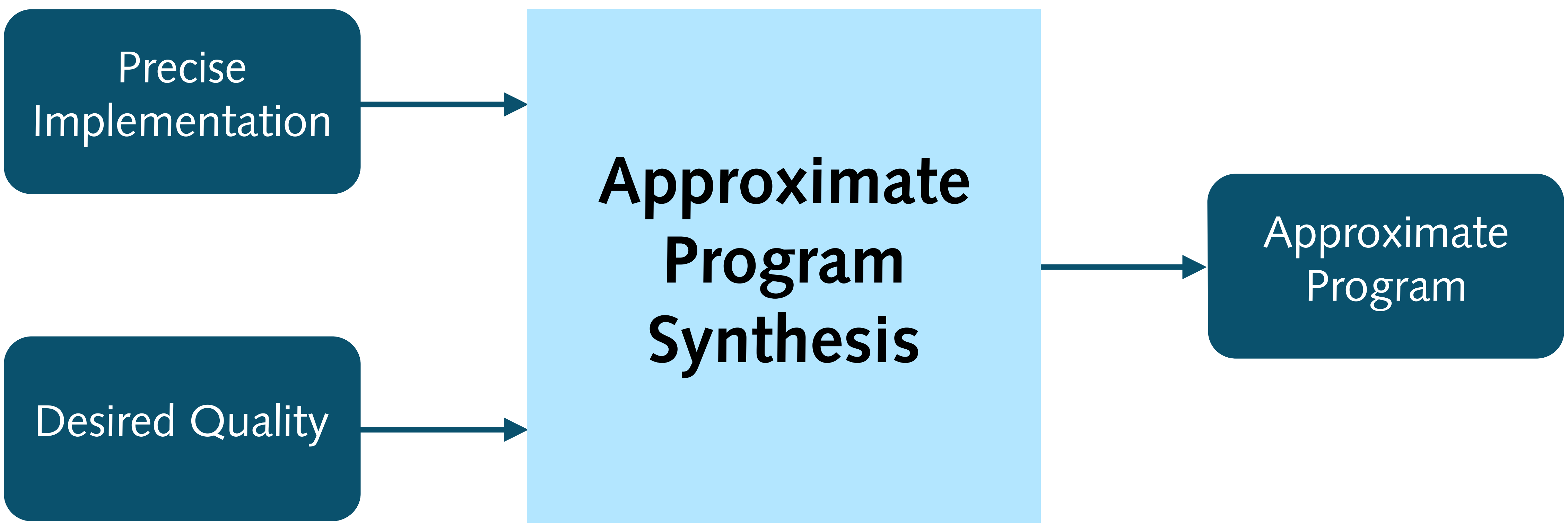
Synthesis produces clever approximations

```
float dist_approx(int a[3], int b[3]) {
    int c1 = abs(b[0] - a[0]);
    int c2 = abs(b[1] - a[1]);
    int c3 = abs(a[2] - b[2]);
    int c4 = c1 | c2;
    int c5 = abs(c3 > c4 ? c3 : c4);
    return (float)c5;
}
```

3D Euclidean distance

1.6× faster, 14.9% error

Thanks!


Precise Implementation
Desired Quality
Approximate Program Synthesis
Approximate Program