CSE P 504

Advanced topics in Software Systems

Abstract Interpretation

Today

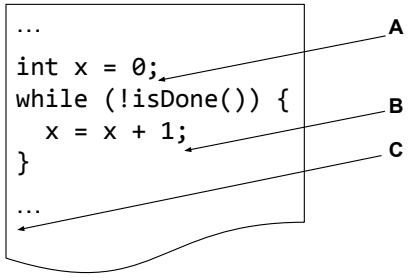
Abstract interpretation

- Intuition
  - Use cases
  - Abstract values, domain and stores
- A bit more formal
  - Lattices
  - Abstraction function
  - Concretization function
  - Transfer function (vs. lub vs. glb)
  - Galois connection
- Exercise: concrete examples

Abstract interpretation (intuition)

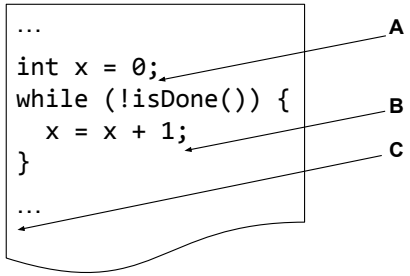
Properties of an ideal program analysis

- Soundness
- Completeness
- Termination



Properties of an ideal program analysis

- Soundness
- Completeness
- Termination



Abstract interpretation sacrifices completeness (precision)

A first example

Program

```
x = 0;
y = read_even();
x = y + 1;
y = 2 * x;
x = y - 2;
y = x / 2;
```

A first example: one concrete execution

Program

```
x = 0;
y = read_even();
x = y + 1;
y = 2 * x;
x = y - 2;
y = x / 2;
```

Concrete execution

```
{x=0;   y=undef}
```

A first example: one concrete execution

Program

```
x = 0;
y = read_even();
x = y + 1;
y = 2 * x;
x = y - 2;
y = x / 2;
```

Concrete execution

```
{x=0;   y=undef}
{x=0;   y=8}
```

A first example: one concrete execution

Program

```
x = 0;
y = read_even();
x = y + 1;
y = 2 * x;
x = y - 2;
y = x / 2;
```

Concrete execution

```
{x=0;  y=undef}
{x=0;  y=8}
{x=9;  y=8}
{x=9;  y=18}
{x=16; y=18}
{x=16; y=8}
```

A first example: program analysis questions

Program

```
x = 0;
y = read_even();
x = y + 1;
y = 2 * x;
x = y - 2;
y = x / 2;
```

Are all assignments necessary?

A first example: SSA form

Program

```
x = 0;
y = read_even();
x = y + 1;
y = 2 * x;
x = y - 2;
y = x / 2;
```

SSA form

```
x1 = 0;
y1 = read_even();
x2 = y1 + 1;
y2 = 2 * x2;
x3 = y2 - 2;
y3 = x3 / 2;
```

X_1 is never read.

A first example: program analysis questions



Program

```
x = 0;
y = read_even();
x = y + 1;
y = 2 * x;
x = y - 2;
y = x / 2;
```

SSA form

```
x1 = 0;
y1 = read_even();
x2 = y1 + 1;
y2 = 2 * x2;
x3 = y2 - 2;
y3 = x3 / 2;
```

What facts can you deduce about y and x after execution (for any possible execution)?

https://homes.cs.washington.edu/~rjust/courses/CSEP504/ai1.html

A first example: symbolic reasoning

Program

```
x = 0;
y = read_even();
x = y + 1;
y = 2 * x;
x = y - 2;
y = x / 2;
```

SSA form

```
x1 = 0;
y1 = read_even();
x2 = y1 + 1;
y2 = 2 * x2;
x3 = y2 - 2;
y3 = x3 / 2;
```

$y_3 = x_3 / 2$
$y_3 = (y_2 - 2) / 2$

A first example: symbolic reasoning

Program

```
x = 0;
y = read_even();
x = y + 1;
y = 2 * x;
x = y - 2;
y = x / 2;
```

SSA form

```
x1 = 0;
y1 = read_even();
x2 = y1 + 1;
y2 = 2 * x2;
x3 = y2 - 2;
y3 = x3 / 2;
```

$y_3 = x_3 / 2$
$y_3 = (y_2 - 2) / 2$
$y_3 = (2 * x_2 - 2) / 2$
$y_3 = (2 * (y_1 + 1) - 2) / 2$
$y_3 = (2 * y_1 + 2 - 2) / 2$
$y_3 = y_1$
$x_3 = y_1 * 2$

Symbolic reasoning shows simplification potential.

A first example: abstract interpretation

Program

```
x = 0;
y = read_even();
x = y + 1;
y = 2 * x;
x = y - 2;
y = x / 2;
```

Abstract value

Abstract domain (e: even, o: odd, u: unk)

A first example: abstract interpretation

Program

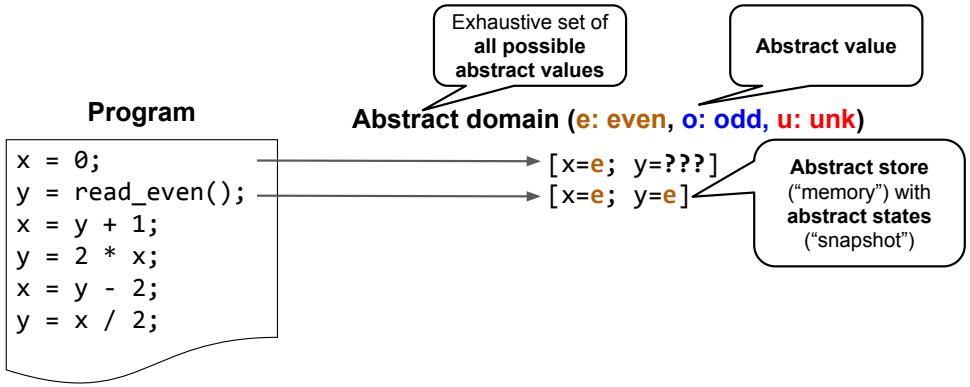
```
x = 0;
y = read_even();
x = y + 1;
y = 2 * x;
x = y - 2;
y = x / 2;
```

Exhaustive set of **all possible abstract values**

Abstract value

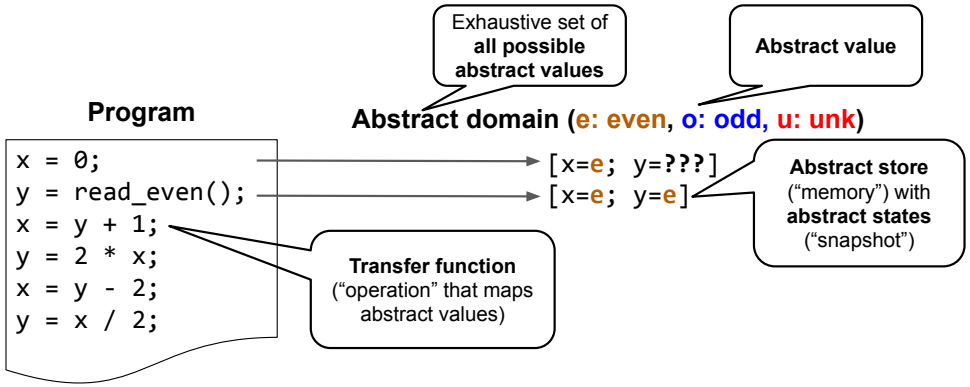
Abstract domain (e: even, o: odd, u: unk)

A first example: abstract interpretation



We usually model undefined values (y); will revisit this later.

A first example: abstract interpretation



A first example: abstract interpretation



Program

```
x = 0;
y = read_even();
x = y + 1;
y = 2 * x;
x = y - 2;
y = x / 2;
```

+	unk	odd	even
unk			
odd			
even			

What is the transfer function for <lhs> + <rhs>?

https://homes.cs.washington.edu/~rjust/courses/CSEP504/ai2.html

A first example: abstract interpretation

Program

```
x = 0;
y = read_even();
x = y + 1;
y = 2 * x;
x = y - 2;
y = x / 2;
```

+	unk	odd	even
unk	unk	unk	unk
odd	unk	even	odd
even	unk	odd	even

A first example: abstract interpretation



Program

```
x = 0;
y = read_even();
x = y + 1;
y = 2 * x;
x = y - 2;
y = x / 2;
```

Abstract domain (e: even, o: odd, u: unk)

```
[x=e; y=???]
[x=e; y=e]
[x=???; y=???]
```

What is the abstract value of x and y after (abstract) execution?

A first example: abstract interpretation



Program

```
x = 0;
y = read_even();
x = y + 1;
y = 2 * x;
x = y - 2;
y = x / 2;
```

Abstract domain (e: even, o: odd, u: unk)

```
{x=e; y=e}
{x=e; y=u}
```

Convince yourself that this is true.

A first example: abstract interpretation

Program

```
x = 0;
y = read_even();
x = y + 1;
y = 2 * x;
x = y - 2;
y = x / 2;
```

Abstract domain (e: even, o: odd, u: unk)

```
{x=e; y=e}
{x=o; y=e}
{x=o; y=e}
{x=e; y=e}
{x=e; y=u}
```

This result is accurate but imprecise.

A first example: abstract interpretation



Program

```
x = 0;
y = read_even();
x = y + 1;
y = 2 * x;
x = y - 2;
y = x / 2;
```

Abstract domain (e: even, o: odd, u: unk)

```
{x=e; y=e}
{x=o; y=e}
{x=o; y=e}
{x=e; y=e}
{x=e; y=u}
```

What abstract domain would allow us to conclude that y is even?

Abstract interpretation (a bit more formal)

Set, semilattice, lattice

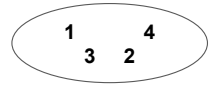
Set, semilattice, lattice

Set

Set, semilattice, lattice

Set

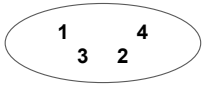
- **unordered** collection of **distinct** objects



Set, semilattice, lattice

Set

- unordered collection of distinct objects

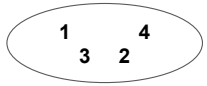


Partially ordered set

Set, semilattice, lattice

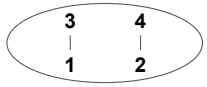
Set

- unordered collection of distinct objects



Partially ordered set

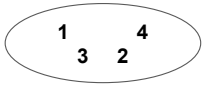
- Binary relationship $\leq$:
  - Reflexive: $x \leq x$
  - Anti-symmetric: $x \leq y \wedge y \leq x \Rightarrow x = y$
  - Transitive: $x \leq y \wedge y \leq z \Rightarrow x \leq z$



Set, semilattice, lattice

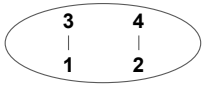
Set

- unordered collection of distinct objects



Partially ordered set

- Binary relationship $\leq$:
  - Reflexive: $x \leq x$
  - Anti-symmetric: $x \leq y \wedge y \leq x \Rightarrow x = y$
  - Transitive: $x \leq y \wedge y \leq z \Rightarrow x \leq z$



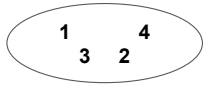
Join semilattice

Meet semilattice

Set, semilattice, lattice

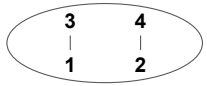
Set

- unordered collection of distinct objects



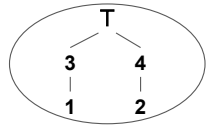
Partially ordered set

- Binary relationship $\leq$:
  - Reflexive: $x \leq x$
  - Anti-symmetric: $x \leq y \wedge y \leq x \Rightarrow x = y$
  - Transitive: $x \leq y \wedge y \leq z \Rightarrow x \leq z$



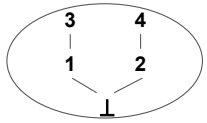
Join semilattice

- Partially ordered set with least upper bound (join)



Meet semilattice

- Partially ordered set with greatest lower bound (meet)



Set, semilattice, lattice

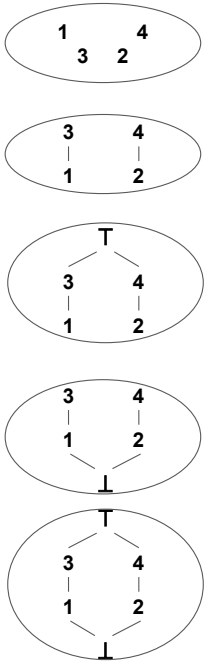
Set
- unordered collection of distinct objects

Partially ordered set
- Binary relationship $\leq$:
  - Reflexive: $x \leq x$
  - Anti-symmetric: $x \leq y \wedge y \leq x \Rightarrow x = y$
  - Transitive: $x \leq y \wedge y \leq z \Rightarrow x \leq z$

Join semilattice
- Partially ordered set with least upper bound (join)

Meet semilattice
- Partially ordered set with greatest lower bound (meet)

Lattice
- Both a join semilattice and a meet semilattice

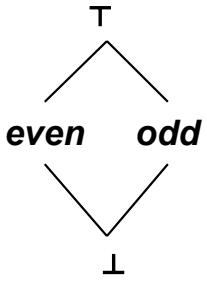


Lattice: example

Abstract domain: *even, odd, unknown*

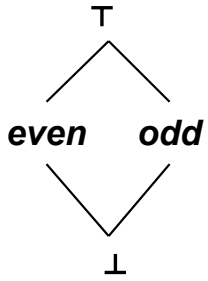
Lattice: example

Abstract domain: *even, odd, unknown ($\top$), undefined ($\bot$)*



Lattice: example

Abstract domain: *even, odd, unknown ($\top$), undefined ($\bot$)*



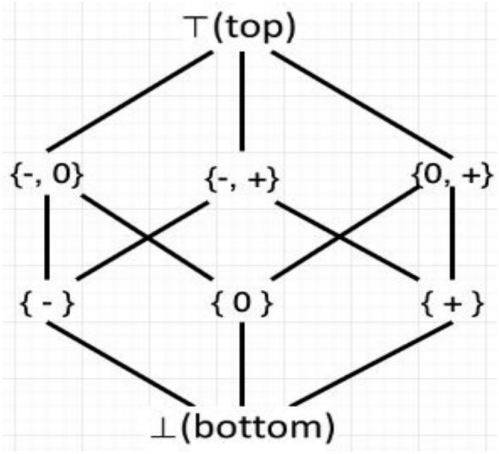
How would you characterize the difference between unknown and undefined? Why does this difference exist and matter?

Lattice: example



Define an abstract domain and lattice for a **sign analysis** (positive vs. negative numbers).

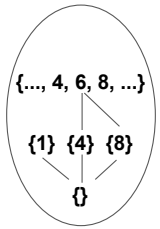
Lattice: example



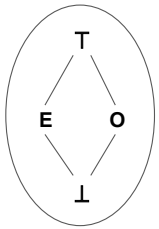
What are the pros/cons of adding the additional abstract values (beyond -, +)?

Abstraction function

Concrete ($P(\mathbb{N})$)



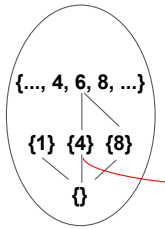
Abstract



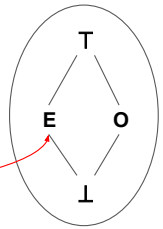
What is the abstraction (α) of {4}?

Abstraction function

Concrete ($P(\mathbb{N})$)



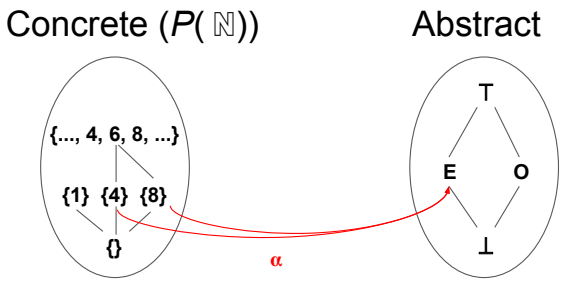
Abstract



α

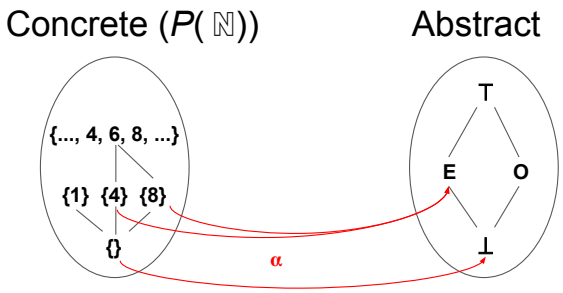
What is the abstraction (α) of {8}?

Abstraction function



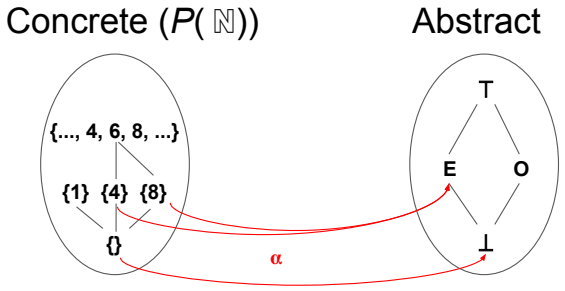
What is the abstraction (α) of {}?

Abstraction function



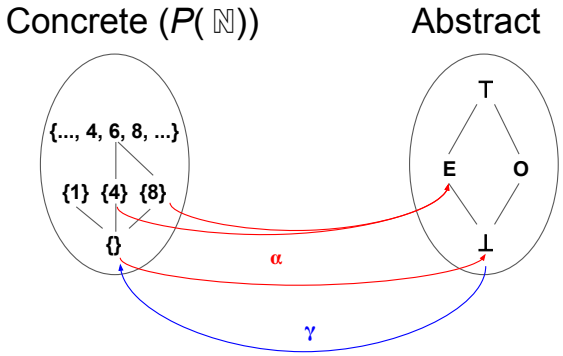
Why do we need an abstraction function?

Concretization function



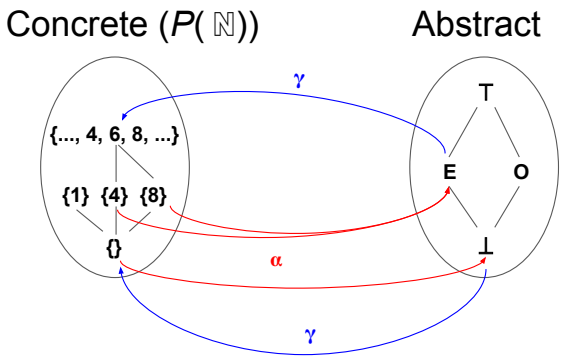
What is the concretization (γ) of ⊥?

Concretization function

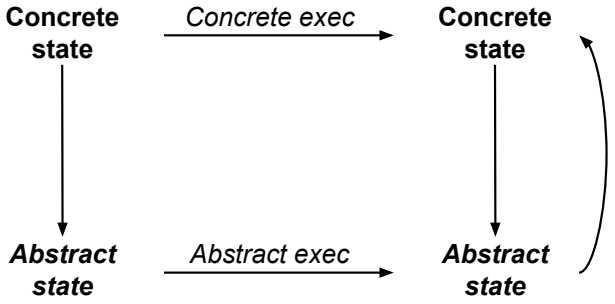


What is the concretization (γ) of **E** ?

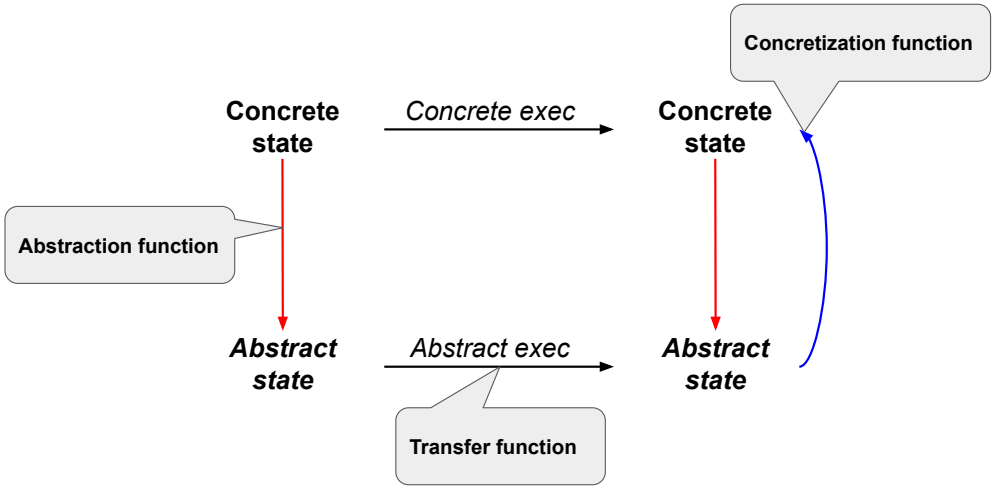
Concretization function



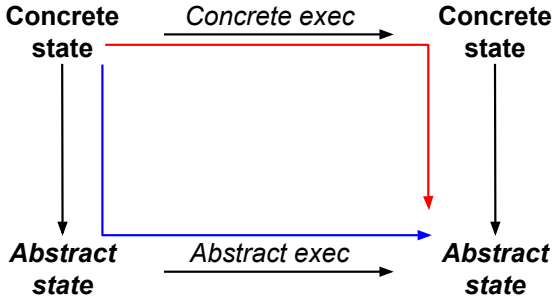
Transfer function



Transfer function

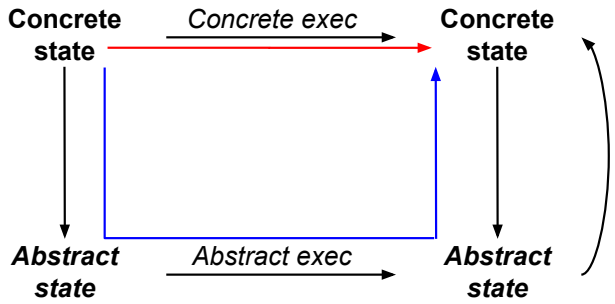


Abstract interpretation: approximation



Do both paths lead to the same abstract state?

Abstract interpretation: approximation



Do both paths lead to the same concrete state?

Abstract interpretation: soundness example

Abstract domain: {*odd, even$_2$, even$_4$, is2, unk*}

x=16 → 16 / 2 →

Abstract interpretation: soundness example

Abstract domain: {*odd, even$_2$, even$_4$, is2, unk*}

x=16 → 16 / 2 → **x=8** → 8 / 2 →

Abstract interpretation: soundness example

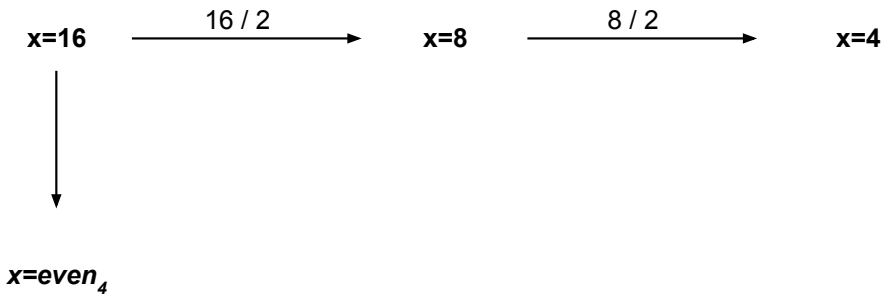
Abstract domain: {*odd, even$_2$, even$_4$, is2, unk*}

x=16 → 16 / 2 → **x=8** → 8 / 2 → **x=4**

↓

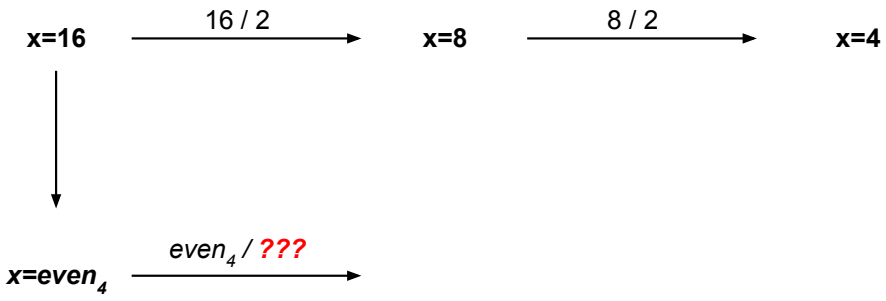
Abstract interpretation: soundness example

Abstract domain: {*odd, even$_2$, even$_4$, is2, unk*}



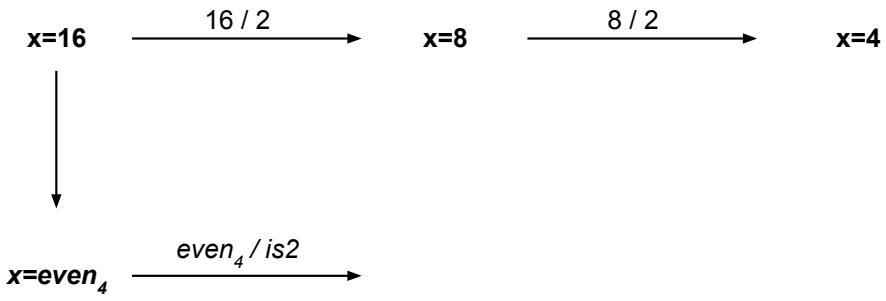
Abstract interpretation: soundness example

Abstract domain: {*odd, even$_2$, even$_4$, is2, unk*}



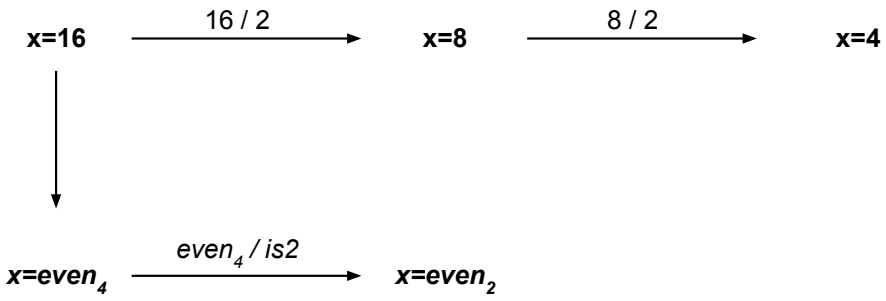
Abstract interpretation: soundness example

Abstract domain: {*odd, even$_2$, even$_4$, is2, unk*}



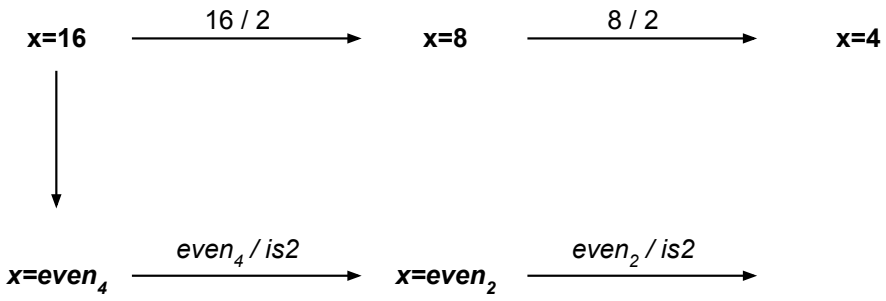
Abstract interpretation: soundness example

Abstract domain: {*odd, even$_2$, even$_4$, is2, unk*}



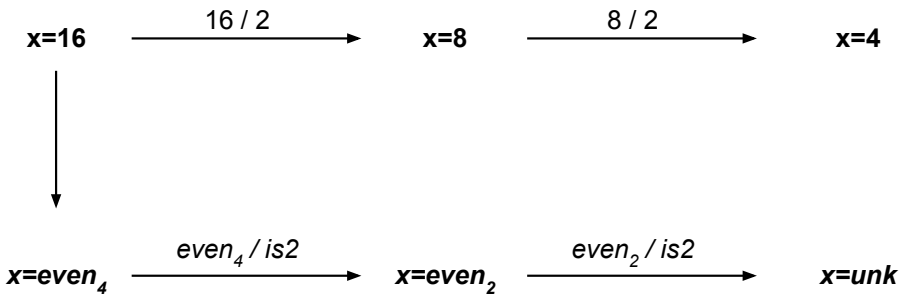
Abstract interpretation: soundness example

Abstract domain: {*odd, even$_2$, even$_4$, is2, unk*}



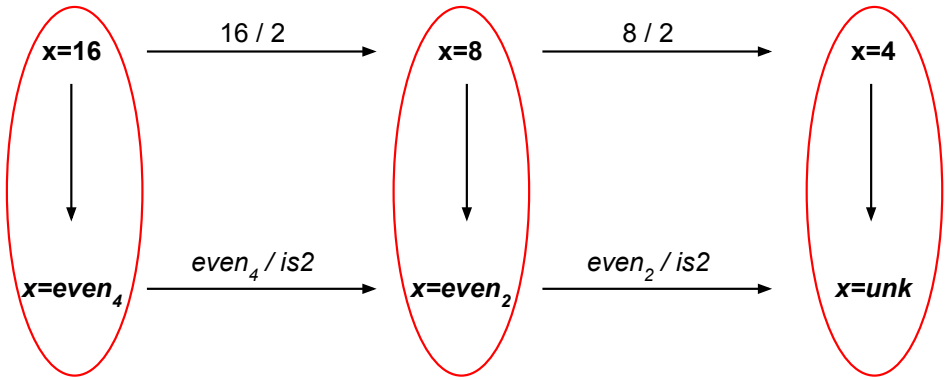
Abstract interpretation: soundness example

Abstract domain: {*odd, even$_2$, even$_4$, is2, unk*}



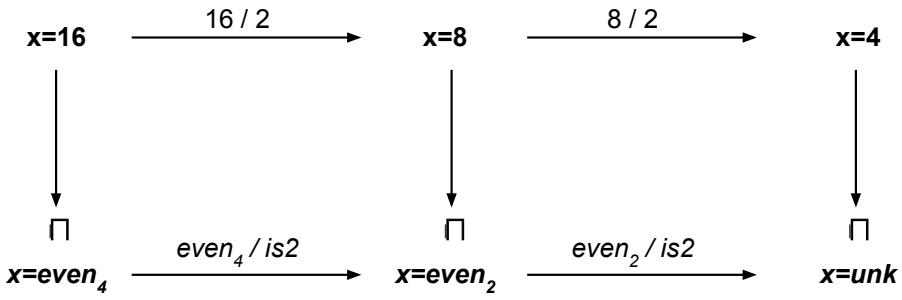
Abstract interpretation: soundness example

Abstract domain: {*odd, even$_2$, even$_4$, is2, unk*}

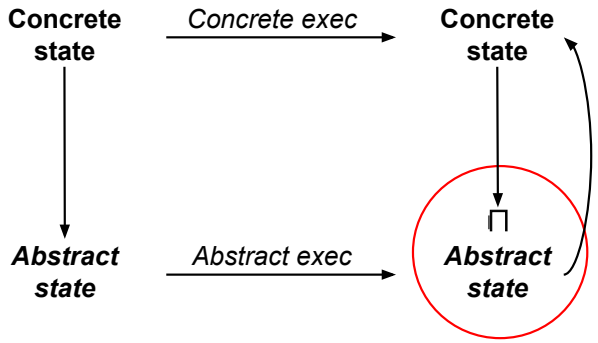


Abstract interpretation: soundness example

Abstract domain: {*odd, even$_2$, even$_4$, is2, unk*}

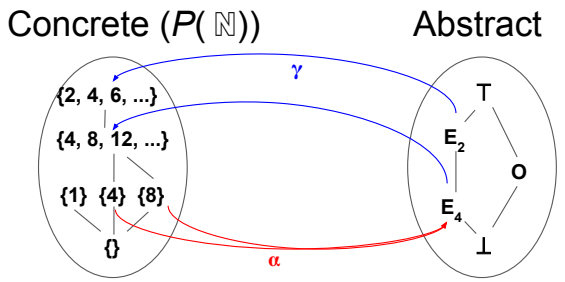


Abstract interpretation: soundness



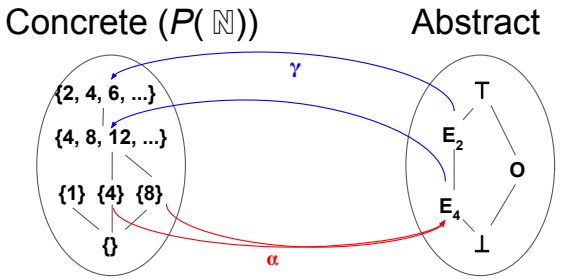
What properties must be satisfied by the abstraction, concretization, and transfer functions?

Sound approximation: properties



What properties must α and γ satisfy?

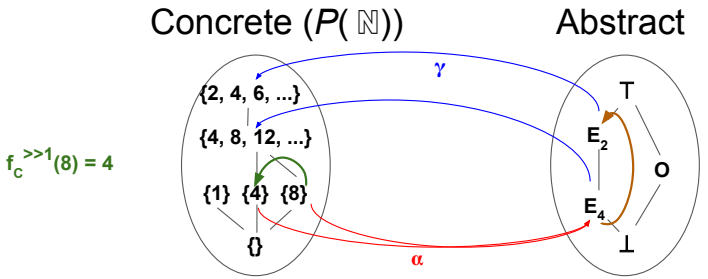
Sound approximation: galois connection



Galois connection

- α: *C -> A*
- γ: A -> C
- $\forall c \in C: c \leq \gamma(\alpha(c))$

Sound approximation: properties

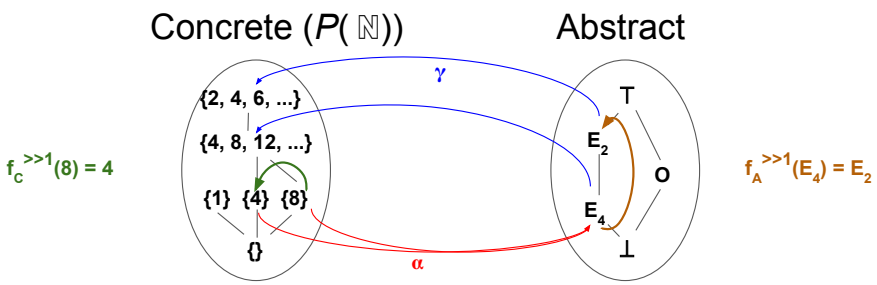


$f_C^{>>1}(8) = 4$

$f_A^{>>1}(E_4) = E_2$

Contrast the transfer function for a shift operation (>>1) with its concrete counterpart.

Sound approximation: consistency



Transfer function

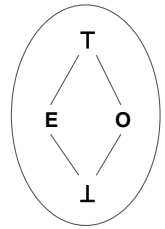
- Consistent with concrete execution
  - c: concrete state; c' = f_C(c)
  - a: α(c)
  - a' = f_A(a)
  - c'' = γ(a')
  - c' ≤ c''

Sound approximation: properties

Transfer function

- f_A^+: A x A -> A

+	E	O	T	...
E	E	O	T	
O	O	E	T	
T	T	T	T	
...				



Lub

- lub: A x A -> A

lub(E, O) = T

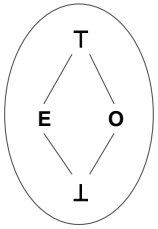
What properties must the lub function satisfy?

Sound approximation: monotonicity

Transfer function

- f_A^+: A x A -> A
- may not be monotone

+	E	O	T	...
E	E	O	T	
O	O	E	T	
T	T	T	T	
...				



Lub

- lub: A x A -> A
- must be monotone

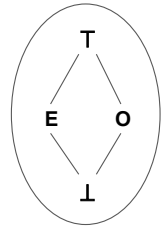
lub(E, O) = T

Sound approximation: join (lub) vs. meet (glb)

Transfer function

- f_A^+: A x A -> A
- may not be monotone

+	E	O	T	...
E	E	O	T	
O	O	E	T	
T	T	T	T	
...				



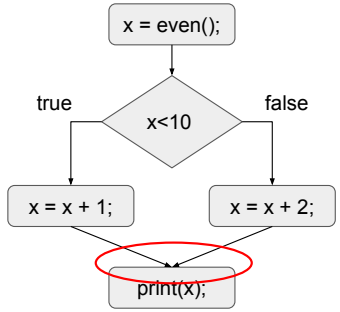
Lub

- lub: A x A -> A
- must be monotone

lub(E, O) = T

```
int x = even();

if (x < 10) {
  x = x + 1;
} else {
  x = x + 2;
}
print(x);
```



Exercises: concrete examples

Small-group exercise



- Work through two examples:
  - Join vs. meet operation (`f(int a, int b, int c): int`)

```
if (cond) {
  x = a * b;
} else {
  x = a * c;
}
return x;
```

Which parameters (`a, b, c`)
- will definitely be used?
- may be used?

(`cond` is independent of the parameters)

  - Termination/fix point iteration

```
int x = 2;
while (x < 10) {
  x = x + 2;
}
return x;
```

Is the value of `x` after the loop an even number? Use an abstract domain with {odd, 2, $even_2$, and $even_4$}. Why and how does the analysis terminate?