

Type
systems

The INFDEV
team

Introduction

Issues with
Python

Possible
solutions

Static typing

Typing rules
and semantic
rules

Conclusion

Type systems

The INFDEV team

Hogeschool Rotterdam
Rotterdam, Netherlands

Type
systems

The INFDEV
team

Introduction

Issues with
Python

Possible
solutions

Static typing

Typing rules
and semantic
rules

Conclusion

Introduction

Type
systems

The INFDEV
team

Introduction

Issues with
Python

Possible
solutions

Static typing

Typing rules
and semantic
rules

Conclusion

Lecture topics

- Issues with Python
- Issues with Python and possible solutions
- Static typing

Type
systems

The INFDEV
team

Introduction

Issues with
Python

Possible
solutions

Static typing

Typing rules
and semantic
rules

Conclusion

Issues with Python

Type
systems

The INFDEV
team

Introduction

Issues with
Python

Possible
solutions

Static typing

Typing rules
and semantic
rules

Conclusion

Lack of...

- Lack of constraints: how can we specify that a function only takes integers as input
- Lack of structure: how can we specify that a variable will certainly support some methods
- Lack of assurances: how can we guarantee that programs with evident errors are not run

Type
systems

The INFDEV
team

Introduction

Issues with
Python

Possible
solutions

Static typing

Typing rules
and semantic
rules

Conclusion

What is wrong with this?

```
1 def f(x):  
2     return (x * 2)  
3 f("nonsense")
```

What is wrong with this?

```
1 def f(x):  
2     return (x * 2)  
3 f("nonsense")
```

The function clearly works with integers, but is given a string

What is wrong with this?

```
1 x = input()
2 if (x > 100):
3     print("dumb")
4 else:
5     print("dumber")
```


What is wrong with this?

```
1 x = input()
2 if (x > 100):
3     print("dumb")
4 else:
5     print("dumber")
```

The comparison is nonsensical if `x` is not a number

What is wrong with this?

```
1 def g(car):  
2     return car.drive(2)  
3 g(-1)
```

What is wrong with this?

```
1 def g(car):  
2     return car.drive(2)  
3 g(-1)
```

We expect something with a drive method, but get an integer instead

Type
systems

The INFDEV
team

Introduction

Issues with
Python

Possible
solutions

Static typing

Typing rules
and semantic
rules

Conclusion

Possible solutions

Type
systems

The INFDEV
team

Introduction

Issues with
Python

Possible
solutions

Static typing

Typing rules
and semantic
rules

Conclusion

Testing?

- Testing the program should be enough

Type
systems

The INFDEV
team

Introduction

Issues with
Python

Possible
solutions

Static typing

Typing rules
and semantic
rules

Conclusion

Testing?

- Testing the program should be enough
- Right?

Type
systems

The INFDEV
team

Introduction

Issues with
Python

Possible
solutions

Static typing

Typing rules
and semantic
rules

Conclusion

Testing?

- Testing the program should be enough
- Right?
- No. The number of possible execution paths is immense (order of billions), and each test only takes one.
- Testing can only guarantee presence of bugs, but not their absence!

How many times would we need to test to be sure there is no error?

```
1 if (randint(0,100000) > 99999):  
2     g(-1)  
3 else:  
4     g(mercedesSL500)
```


How many times would we need to test to be sure there is no error?

```
1 if (randint(0,100000) > 99999):  
2     g(-1)  
3 else:  
4     g(mercedesSL500)
```

≥ 100000

Testing?

- We want our programming languages to perform checks for us
- Clearly nonsensical programs should be rejected before we can even run them
- It is safer and easier to spend more time “talking” with the IDE than hoping to find all errors with tests

Type
systems

The INFDEV
team

Introduction

Issues with
Python

Possible
solutions

Static typing

Typing rules
and semantic
rules

Conclusion

Static typing

Introduction

- The language verifies^a, before running code, that all variables are correctly used
- "Correctly used" means that they are guaranteed to support all operations used on them
- This is by far and large the most typical solution to increase safety and productivity

^aBy means of the **compiler**.

What is static typing?

- When declaring a variable, we also specify what sort of data it will contain
- The **sort** of data contained is called **TYPE** of the variable
- Types can be either primitives (int, string, etc.), custom (classes), or compositions (functions, list of elements of a given type, etc.)

What is static typing?

- Especially in mainstream languages, the specification of the type of a variable is done by hand by the programmer
- In other languages (mostly functional languages like F#, Haskell, etc.) the type of variables is automatically 'guessed' by the compiler
- In our case our programs will become a bit more verbose but better specified
- Still, static typing is not necessarily connected with verbosity

A variable declaration in C# or Java is prefixed by the type of the variable

- `int x;` declares an integer variable
- `string s;` declares a string variable
- `float f;` declares a floating point variable
- ...

Static typing

Type
systems

The INFDEV
team

Introduction

Issues with
Python

Possible
solutions

Static typing

Typing rules
and semantic
rules

Conclusion

```
1 def f(x):  
2     return (x * 2)
```

Becomes, typed:

```
1 int f(int x) {  
2     return (x * 2);  
3 }
```

What has improved and why?

Static typing

Type
systems

The INFDEV
team

Introduction

Issues with
Python

Possible
solutions

Static typing

Typing rules
and semantic
rules

Conclusion

```
1 def f(x):  
2     return (x * 2)
```

Becomes, typed:

```
1 int f(int x) {  
2     return (x * 2);  
3 }
```

What has improved and why?

The second definition encodes information about what goes in and what comes out of the function

Is this still possible to write (as it was in Python)?

```
1 int f(int x) {  
2     return (x * 2);  
3 }  
4 f("nonsense");
```

Is this still possible to write (as it was in Python)?

```
1 int f(int x) {  
2     return (x * 2);  
3 }  
4 f("nonsense");
```

No: we get a compiler error because a string cannot be used where a number is expected

Static typing

Type
systems

The INFDEV
team

Introduction

Issues with
Python

Possible
solutions

Static typing

Typing rules
and semantic
rules

Conclusion

```
1 x = input()
2 if (x > 100):
3     print("dumb")
4 else:
5     print("dumber")
```

Becomes, typed:

```
1 int x = Int32.Parse(Console.ReadLine());
2 if (x > 100) {
3     Console.WriteLine("safe");
4 } else {
5     Console.WriteLine("safer");
6 }
```

What has improved and why?

Static typing

Type
systems

The INFDEV
team

Introduction

Issues with
Python

Possible
solutions

Static typing

Typing rules
and semantic
rules

Conclusion

```
1 x = input()
2 if (x > 100):
3     print("dumb")
4 else:
5     print("dumber")
```

Becomes, typed:

```
1 int x = Int32.Parse(Console.ReadLine());
2 if (x > 100) {
3     Console.WriteLine("safe");
4 } else {
5     Console.WriteLine("safer");
6 }
```

What has improved and why?

The variable declaration specifies what is allowed (and what is not) inside the variable.

Static typing

Type
systems

The INFDEV
team

Introduction

Issues with
Python

Possible
solutions

Static typing

Typing rules
and semantic
rules

Conclusion

```
1 def g(car):  
2     return car.drive(2)  
3 g(-1)
```

Becomes, typed:

```
1 int g(Car car) {  
2     return car.drive(2);  
3 }  
4 g(-1);
```

What has improved and why?

Static typing

Type
systems

The INFDEV
team

Introduction

Issues with
Python

Possible
solutions

Static typing

Typing rules
and semantic
rules

Conclusion

```
1 def g(car):  
2     return car.drive(2)  
3 g(-1)
```

Becomes, typed:

```
1 int g(Car car) {  
2     return car.drive(2);  
3 }  
4 g(-1);
```

What has improved and why?

The function declaration specifies that `car` is an instance of the `Car` class. We will thus get a compiler error.

Type
systems

The INFDEV
team

Introduction

Issues with
Python

Possible
solutions

Static typing

Typing rules
and semantic
rules

Conclusion

Typing rules and semantic rules

How do we describe them?

- How do we describe such relations clearly?
- We use the so-called **typing rules**, which specify what may be done and what not
- Typing rules are quite intuitive: they state that if one or more premises are true, then the conclusion is true as well

Typing rules and semantic rules

Type
systems

The INFDEV
team

Introduction

Issues with
Python

Possible
solutions

Static typing

Typing rules
and semantic
rules

Conclusion

$$\frac{A \wedge B}{C}$$

If A and B are true, then we can conclude C

Typing rules and semantic rules

Type
systems

The INFDEV
team

Introduction

Issues with
Python

Possible
solutions

Static typing

Typing rules
and semantic
rules

Conclusion

I wish to buy a pretty car \wedge I have 120000 euros
I buy a Mercedes SL500

How do we read this rule?

Typing rules and semantic rules

Type
systems

The INFDEV
team

Introduction

Issues with
Python

Possible
solutions

Static typing

Typing rules
and semantic
rules

Conclusion

$$\frac{\text{I wish to buy a pretty car} \wedge \text{I have 120000 euros}}{\text{I buy a Mercedes SL500}}$$

How do we read this rule?

If I have 120000 euros and I wish to buy a pretty car, then I buy a Mercedes SL500

Type
systems

The INFDEV
team

Introduction

Issues with
Python

Possible
solutions

Static typing

Typing rules
and semantic
rules

Conclusion

It is raining \wedge I have my umbrella with me
I open my umbrella

How do we read this rule?

Typing rules and semantic rules

Type
systems

The INFDEV
team

Introduction

Issues with
Python

Possible
solutions

Static typing

Typing rules
and semantic
rules

Conclusion

It is raining \wedge I have my umbrella with me
I open my umbrella

How do we read this rule?

If it is raining, and I have my umbrella with me, then I open my umbrella

Type
systems

The INFDEV
team

Introduction

Issues with
Python

Possible
solutions

Static typing

Typing rules
and semantic
rules

Conclusion

Reading typing rules

Let us apply this machinery to programming languages

Reading typing rules

- Let us apply this machinery to programming languages
- We will effectively give the specification of a modern compiler
- This looks like a “broadly scoped” execution of the program, and it is indeed such
- This process is called type checking

Reading typing rules

- We want to specify this in the typing rule notation
- The typing rules manipulate a stack of declarations which we will call D
- Each typing rule will add or remove variable declarations and return the type of the current expression
- Instead of coupling each variable with its value, we couple it with its type

Typing rules and semantic rules

Type
systems

The INFDEV
team

Introduction

Issues with
Python

Possible
solutions

Static typing

Typing rules
and semantic
rules

Conclusion

- The simplest typing rule is the one that finds a variable declaration
- A declaration adds to the declarations D the variable, connected with its type

$$\overline{\langle (T \text{ } v;), D \rangle \rightarrow \langle \text{void}, D[v \mapsto T] \rangle}$$

Typing rules and semantic rules

Type
systems

The INFDEV
team

Introduction

Issues with
Python

Possible
solutions

Static typing

Typing rules
and semantic
rules

Conclusion

1

```
int x;
```

Declarations:

PC
1

Typing rules and semantic rules

Type
systems

The INFDEV
team

Introduction

Issues with
Python

Possible
solutions

Static typing

Typing rules
and semantic
rules

Conclusion

1

```
int x;
```

Declarations:

PC	x
2	int

Typing rules and semantic rules

Type
systems

The INFDEV
team

Introduction

Issues with
Python

Possible
solutions

Static typing

Typing rules
and semantic
rules

Conclusion

- When we look the variable up, its type is whatever type was found connected to it in the declarations
- This does further nothing to the declarations
- Let's assume that x is a variable name

$$\overline{\langle x, D \rangle} \rightarrow \overline{\langle D[x], D \rangle}$$

Typing rules and semantic rules

Type
systems

The INFDEV
team

Introduction

Issues with
Python

Possible
solutions

Static typing

Typing rules
and semantic
rules

Conclusion

```
1 int x = 10;  
2 x = (x + 5);
```

Declarations:

PC
1

Typing rules and semantic rules

Type
systems

The INFDEV
team

Introduction

Issues with
Python

Possible
solutions

Static typing

Typing rules
and semantic
rules

Conclusion

```
1 int x = 10;  
2 x = (x + 5);
```

Declarations:

PC	x
2	int

Typing rules and semantic rules

Type
systems

The INFDEV
team

Introduction

Issues with
Python

Possible
solutions

Static typing

Typing rules
and semantic
rules

Conclusion

1
2

```
int x = 10;  
x = (x + 5);
```

Declarations:

PC	x
3	int

Typing rules and semantic rules

Type
systems

The INFDEV
team

Introduction

Issues with
Python

Possible
solutions

Static typing

Typing rules
and semantic
rules

Conclusion

- Another simple typing rule is the one that types a constant value
- It does nothing to the declarations
- Let's assume that `i` is an integer constant

$$\overline{\langle i, D \rangle} \rightarrow \overline{\langle \text{int}, D \rangle}$$

Typing rules and semantic rules

Type
systems

The INFDEV
team

Introduction

Issues with
Python

Possible
solutions

Static typing

Typing rules
and semantic
rules

Conclusion

- Let's assume that `f` is a floating point constant

$$\overline{\langle f, D \rangle \rightarrow \langle \text{float}, D \rangle}$$

Typing rules and semantic rules

Type
systems

The INFDEV
team

Introduction

Issues with
Python

Possible
solutions

Static typing

Typing rules
and semantic
rules

Conclusion

- Let's assume that `s` is a string constant

$$\frac{}{\langle s, D \rangle \rightarrow \langle \text{string}, D \rangle}$$

Typing rules and semantic rules

Type
systems

The INFDEV
team

Introduction

Issues with
Python

Possible
solutions

Static typing

Typing rules
and semantic
rules

Conclusion

- Let's assume that `b` is a boolean constant

$$\overline{\langle b, D \rangle} \rightarrow \overline{\langle \text{bool}, D \rangle}$$

Type
systems

The INFDEV
team

Introduction

Issues with
Python

Possible
solutions

Static typing

Typing rules
and semantic
rules

Conclusion

Reading typing rules

More complex typing rules compose together the types of different statements

Typing rules and semantic rules

Type
systems

The INFDEV
team

Introduction

Issues with
Python

Possible
solutions

Static typing

Typing rules
and semantic
rules

Conclusion

- The typing rule for operators such as + requires the operands to be compatible
- The type of both operands is often the same, for example int or float
- The resulting type is then the type of both operands
- Operands do not modify the current declarations

$$\frac{\langle a, D \rangle \rightarrow \langle \text{int}, D \rangle \wedge \langle b, D \rangle \rightarrow \langle \text{int}, D \rangle}{\langle (a + b), D \rangle \rightarrow \langle \text{int}, D \rangle}$$

$$\frac{\langle a, D \rangle \rightarrow \langle \text{float}, D \rangle \wedge \langle b, D \rangle \rightarrow \langle \text{float}, D \rangle}{\langle (a + b), D \rangle \rightarrow \langle \text{float}, D \rangle}$$

$$\frac{\langle a, D \rangle \rightarrow \langle \text{string}, D \rangle \wedge \langle b, D \rangle \rightarrow \langle \text{string}, D \rangle}{\langle (a + b), D \rangle \rightarrow \langle \text{string}, D \rangle}$$

Typing rules and semantic rules

Type
systems

The INFDEV
team

Introduction

Issues with
Python

Possible
solutions

Static typing

Typing rules
and semantic
rules

Conclusion

- The type of both operands could differ, but still be compatible (for example adding an `int` and a `float`)
- The resulting type is then the most generic type of the operands
- Operands do not modify the current declarations

$$\frac{\langle a, D \rangle \rightarrow \langle \text{int}, D \rangle \wedge \langle b, D \rangle \rightarrow \langle \text{float}, D \rangle}{\langle (a + b), D \rangle \rightarrow \langle \text{float}, D \rangle}$$
$$\frac{\langle a, D \rangle \rightarrow \langle \text{float}, D \rangle \wedge \langle b, D \rangle \rightarrow \langle \text{int}, D \rangle}{\langle (a + b), D \rangle \rightarrow \langle \text{float}, D \rangle}$$

Typing rules and semantic rules

Type
systems

The INFDEV
team

Introduction

Issues with
Python

Possible
solutions

Static typing

Typing rules
and semantic
rules

Conclusion

- Statements in a sequence both modify, top-to-bottom, the declarations
- Usually we expect the statements to simply return nothing, that is `void`
- Further we cannot say anything about what they each do

$$\frac{\langle a, D \rangle \rightarrow \langle \text{void}, D_1 \rangle \wedge \langle b, D_1 \rangle \rightarrow \langle \text{void}, D_2 \rangle}{\langle (a; b), D \rangle \rightarrow \langle \text{void}, D_2 \rangle}$$

Typing rules and semantic rules

Type
systems

The INFDEV
team

Introduction

Issues with
Python

Possible
solutions

Static typing

Typing rules
and semantic
rules

Conclusion

```
1 int x = 10;  
2 int y = 20;  
3 x = (x + y);
```

Declarations:

PC
1

Typing rules and semantic rules

Type
systems

The INFDEV
team

Introduction

Issues with
Python

Possible
solutions

Static typing

Typing rules
and semantic
rules

Conclusion

```
1 int x = 10;  
2 int y = 20;  
3 x = (x + y);
```

Declarations:

PC	x
2	int

Typing rules and semantic rules

Type
systems

The INFDEV
team

Introduction

Issues with
Python

Possible
solutions

Static typing

Typing rules
and semantic
rules

Conclusion

```
1 int x = 10;  
2 int y = 20;  
3 x = (x + y);
```

Declarations:

PC	x	y
3	int	int

Typing rules and semantic rules

Type
systems

The INFDEV
team

Introduction

Issues with
Python

Possible
solutions

Static typing

Typing rules
and semantic
rules

Conclusion

```
1 int x = 10;  
2 int y = 20;  
3 x = (x + y);
```

Declarations:

PC	x	y
4	int	int

Typing rules and semantic rules

Type
systems

The INFDEV
team

Introduction

Issues with
Python

Possible
solutions

Static typing

Typing rules
and semantic
rules

Conclusion

- The typing rule for an if-then-else requires the condition to be a boolean expression, and assumes the type of both the then and the else bodies
- The type of both the then and the else bodies must be the same (usually void, something else in case of function returns)
- It does not add anything to the declarations, even though the bodies of the then and the else might declare local variables

$$\frac{\langle c, D \rangle \rightarrow \langle \text{bool}, D \rangle \wedge \langle A, D \rangle \rightarrow \langle T, D' \rangle \wedge \langle B, D \rangle \rightarrow \langle U, D' \rangle \wedge T = U}{\langle (\text{if } c \{ A \} \text{else} \{ B \}), D \rangle \rightarrow \langle T, D \rangle}$$

Typing rules and semantic rules

Type
systems

The INFDEV
team

Introduction

Issues with
Python

Possible
solutions

Static typing

Typing rules
and semantic
rules

Conclusion

```
1 int x = 10;  
2 int y = 20;  
3 if (x > y) {  
4     string z = "x";  
5     Console.WriteLine(z);  
6 } else {  
7     string z = "y";  
8     Console.WriteLine(z);  
9 }
```

Declarations:

PC
1

Typing rules and semantic rules

Type
systems

The INFDEV
team

Introduction

Issues with
Python

Possible
solutions

Static typing

Typing rules
and semantic
rules

Conclusion

```
1 int x = 10;  
2 int y = 20;  
3 if (x > y) {  
4     string z = "x";  
5     Console.WriteLine(z);  
6 } else {  
7     string z = "y";  
8     Console.WriteLine(z);  
9 }
```

Declarations:

PC	x
2	int

Typing rules and semantic rules

Type
systems

The INFDEV
team

Introduction

Issues with
Python

Possible
solutions

Static typing

Typing rules
and semantic
rules

Conclusion

```
1 int x = 10;  
2 int y = 20;  
3 if (x > y) {  
4     string z = "x";  
5     Console.WriteLine(z);  
6 } else {  
7     string z = "y";  
8     Console.WriteLine(z);  
9 }
```

Declarations:

PC	x	y
3	int	int

Typing rules and semantic rules

Type
systems

The INFDEV
team

Introduction

Issues with
Python

Possible
solutions

Static typing

Typing rules
and semantic
rules

Conclusion

```
1 int x = 10;
2 int y = 20;
3 if (x > y) {
4     string z = "x";
5     Console.WriteLine(z);
6 } else {
7     string z = "y";
8     Console.WriteLine(z);
9 }
```

Declarations:

x	y		PC
int	int		4

Typing rules and semantic rules

Type
systems

The INFDEV
team

Introduction

Issues with
Python

Possible
solutions

Static typing

Typing rules
and semantic
rules

Conclusion

```
1 int x = 10;
2 int y = 20;
3 if (x > y) {
4     string z = "x";
5     Console.WriteLine(z);
6 } else {
7     string z = "y";
8     Console.WriteLine(z);
9 }
```

Declarations:

x	y		PC	z
int	int		5	string

Typing rules and semantic rules

Type
systems

The INFDEV
team

Introduction

Issues with
Python

Possible
solutions

Static typing

Typing rules
and semantic
rules

Conclusion

```
1 int x = 10;
2 int y = 20;
3 if (x > y) {
4     string z = "x";
5     Console.WriteLine(z);
6 } else {
7     string z = "y";
8     Console.WriteLine(z);
9 }
```

Declarations:

PC	x	y
6	int	int

Typing rules and semantic rules

Type
systems

The INFDEV
team

Introduction

Issues with
Python

Possible
solutions

Static typing

Typing rules
and semantic
rules

Conclusion

```
1 int x = 10;
2 int y = 20;
3 if (x > y) {
4     string z = "x";
5     Console.WriteLine(z);
6 } else {
7     string z = "y";
8     Console.WriteLine(z);
9 }
```

Declarations:

x	y		PC
int	int		7

Typing rules and semantic rules

Type
systems

The INFDEV
team

Introduction

Issues with
Python

Possible
solutions

Static typing

Typing rules
and semantic
rules

Conclusion

```
1 int x = 10;  
2 int y = 20;  
3 if (x > y) {  
4     string z = "x";  
5     Console.WriteLine(z);  
6 } else {  
7     string z = "y";  
8     Console.WriteLine(z);  
9 }
```

Declarations:

x	y		PC	z
int	int		8	string

Typing rules and semantic rules

Type
systems

The INFDEV
team

Introduction

Issues with
Python

Possible
solutions

Static typing

Typing rules
and semantic
rules

Conclusion

```
1 int x = 10;  
2 int y = 20;  
3 if (x > y) {  
4     string z = "x";  
5     Console.WriteLine(z);  
6 } else {  
7     string z = "y";  
8     Console.WriteLine(z);  
9 }
```

Declarations:

PC	x	y
9	int	int

Typing rules and semantic rules

Type
systems

The INFDEV
team

Introduction

Issues with
Python

Possible
solutions

Static typing

Typing rules
and semantic
rules

Conclusion

- The typing rule for a `while` loop requires the condition to be a boolean expression, and assumes the type of the body
- The type of the body can be anything (usually `void`, something else in case of function returns)
- It does not add anything to the declarations, even though the body might declare local variables

$$\frac{\langle c, D \rangle \rightarrow \langle \text{bool}, D \rangle \wedge \langle B, D \rangle \rightarrow \langle T, D_1 \rangle}{\langle (\text{while } c \{ A \}), D \rangle \rightarrow \langle T, D \rangle}$$

Typing rules and semantic rules

Type
systems

The INFDEV
team

Introduction

Issues with
Python

Possible
solutions

Static typing

Typing rules
and semantic
rules

Conclusion

- The typing rule for a class declaration adds the class declaration to the declarations with all its attributes and methods
- When adding the declaration of the class, we have to check that the types of the method bodies match their declarations
- Assume that C is the class name, a_i is the i -th attribute in the class (of type A_i), and m_j is the j -th method in the class (with type M_j)

$$\frac{D_1 := D[C \mapsto [..., a_i \mapsto A_i, ..., m_j \mapsto M_j, ...]] \wedge \langle M_j \ m_j, D_1[\text{this} \mapsto C] \rangle \rightarrow \langle M'_j, D_2 \rangle \wedge M_j = \langle (\text{class } C \{ ..., A_i \ a_i, ..., M_j \ m_j, ... \}), D \rangle \rightarrow \langle T, D_1 \rangle}{\langle (\text{class } C \{ ..., A_i \ a_i, ..., M_j \ m_j, ... \}), D \rangle \rightarrow \langle T, D_1 \rangle}$$

- When type checking a method declaration (within a class declaration we type check its body and compare the result with the type of the declaration
- Assume that C is the class name, p_i is the i -th parameter of the method (of type P_i), and b is the method body
- The type of a method is of the form $P_1 \times P_2 \times \cdots \times P_n \rightarrow R$, where P_i is the type of the i -th parameter and R is the return type

$$\frac{\langle (b), D[..., p_i \mapsto P_i, ...] \rangle \rightarrow \langle R, D_1 \rangle}{\langle (R \text{ m}(..., p_1 \text{ } p_1, ...)b), D \rangle \rightarrow \langle (P_1 \times P_2 \times \cdots \times P_n \rightarrow R), D \rangle}$$

Typing rules and semantic rules

Type
systems

The INFDEV
team

Introduction

Issues with
Python

Possible
solutions

Static typing

Typing rules
and semantic
rules

Conclusion

```
1 class Counter {  
2     private int cnt;  
3     public Counter() {  
4         this.cnt = 0;  
5     }  
6     public void incr(int diff) {  
7         this.cnt = (this.cnt + diff);  
8     }  
9 }
```

Declarations:

PC
1

Typing rules and semantic rules

Type
systems

The INFDEV
team

Introduction

Issues with
Python

Possible
solutions

Static typing

Typing rules
and semantic
rules

Conclusion

```

1 class Counter {
2     private int cnt;
3     public Counter() {
4         this.cnt = 0;
5     }
6     public void incr(int diff) {
7         this.cnt = (this.cnt + diff);
8     }
9 }

```

Declarations:

PC	this
4	Counter

Classes:

Counter
Counter=Counter \rightarrow Counter
cnt=int
incr=(Counter \times int) \rightarrow void

Typing rules and semantic rules

Type
systems

The INFDEV
team

Introduction

Issues with
Python

Possible
solutions

Static typing

Typing rules
and semantic
rules

Conclusion

```

1  class Counter {
2      private int cnt;
3      public Counter() {
4          this.cnt = 0;
5      }
6      public void incr(int diff) {
7          this.cnt = (this.cnt + diff);
8      }
9  }
```

Declarations:

PC	this
5	Counter

Classes:

Counter
Counter=Counter \rightarrow Counter
cnt=int
incr=(Counter \times int) \rightarrow void

Typing rules and semantic rules

Type
systems

The INFDEV
team

Introduction

Issues with
Python

Possible
solutions

Static typing

Typing rules
and semantic
rules

Conclusion

```

1  class Counter {
2      private int cnt;
3      public Counter() {
4          this.cnt = 0;
5      }
6      public void incr(int diff) {
7          this.cnt = (this.cnt + diff);
8      }
9  }
```

Declarations:

PC	diff	this
7	int	Counter

Classes:

Counter
Counter=Counter \rightarrow Counter
cnt=int
incr=(Counter \times int) \rightarrow void

Typing rules and semantic rules

Type
systems

The INFDEV
team

Introduction

Issues with
Python

Possible
solutions

Static typing

Typing rules
and semantic
rules

Conclusion

```

1  class Counter {
2      private int cnt;
3      public Counter() {
4          this.cnt = 0;
5      }
6      public void incr(int diff) {
7          this.cnt = (this.cnt + diff);
8      }
9  }
```

Declarations:

PC	diff	this
8	int	Counter

Classes:

Counter
Counter=Counter \rightarrow Counter
cnt=int
incr=(Counter \times int) \rightarrow void

Typing rules and semantic rules

Type
systems

The INFDEV
team

Introduction

Issues with
Python

Possible
solutions

Static typing

Typing rules
and semantic
rules

Conclusion

```

1  class Counter {
2      private int cnt;
3      public Counter() {
4          this.cnt = 0;
5      }
6      public void incr(int diff) {
7          this.cnt = (this.cnt + diff);
8      }
9  }
    
```

Declarations:

PC
9

Classes:

Counter
Counter=Counter \rightarrow Counter
cnt=int
incr=(Counter \times int) \rightarrow void

Typing rules and semantic rules

Type
systems

The INFDEV
team

Introduction

Issues with
Python

Possible
solutions

Static typing

Typing rules
and semantic
rules

Conclusion

- When type checking a `return` statement, we typecheck its argument
- The type of the argument is also the type of the `return` statement
- There is no change to the declarations

$$\frac{\langle x, D \rangle \rightarrow \langle T, D \rangle}{\langle (\text{return } x), D \rangle \rightarrow \langle T, D \rangle}$$

Typing rules and semantic rules

Type
systems

The INFDEV
team

Introduction

Issues with
Python

Possible
solutions

Static typing

Typing rules
and semantic
rules

Conclusion

- Statements in a sequence might contain return statements
- In this case one of them might not return void
- Their sequence will assume the non-void type

$$\frac{\langle a, D \rangle \rightarrow \langle T, D_1 \rangle \wedge \langle b, D_1 \rangle \rightarrow \langle \text{void}, D_2 \rangle}{\langle (a; b), D \rangle \rightarrow \langle T, D_2 \rangle}$$

$$\frac{\langle a, D \rangle \rightarrow \langle \text{void}, D_1 \rangle \wedge \langle b, D_1 \rangle \rightarrow \langle T, D_2 \rangle}{\langle (a; b), D \rangle \rightarrow \langle T, D_2 \rangle}$$

Typing rules and semantic rules

Type
systems

The INFDEV
team

Introduction

Issues with
Python

Possible
solutions

Static typing

Typing rules
and semantic
rules

Conclusion

- Statements in a sequence might contain return statements
- They might both return a non-void type
- Their sequence will assume the non-void type of both, which must be the same

$$\frac{\langle a, D \rangle \rightarrow \langle T, D_1 \rangle \wedge \langle b, D_1 \rangle \rightarrow \langle U, D_2 \rangle \wedge T = U}{\langle (a; b), D \rangle \rightarrow \langle T, D_2 \rangle}$$

Typing rules and semantic rules

Type
systems

The INFDEV
team

Introduction

Issues with
Python

Possible
solutions

Static typing

Typing rules
and semantic
rules

Conclusion

```
1 class Counter {  
2     private int cnt;  
3     public Counter() {  
4         this.cnt = 0;  
5     }  
6     public int incr(int diff) {  
7         this.cnt = (this.cnt + diff);  
8         return this.cnt;  
9     }  
10 }
```

Declarations:

PC
1

Typing rules and semantic rules

Type
systems

The INFDEV
team

Introduction

Issues with
Python

Possible
solutions

Static typing

Typing rules
and semantic
rules

Conclusion

```

1  class Counter {
2      private int cnt;
3      public Counter() {
4          this.cnt = 0;
5      }
6      public int incr(int diff) {
7          this.cnt = (this.cnt + diff);
8          return this.cnt;
9      }
10 }
```

Declarations:

PC	this
4	Counter

Classes:

Counter
Counter=Counter \rightarrow Counter
cnt=int
incr=(Counter \times int) \rightarrow int

Typing rules and semantic rules

Type
systems

The INFDEV
team

Introduction

Issues with
Python

Possible
solutions

Static typing

Typing rules
and semantic
rules

Conclusion

```

1  class Counter {
2      private int cnt;
3      public Counter() {
4          this.cnt = 0;
5      }
6      public int incr(int diff) {
7          this.cnt = (this.cnt + diff);
8          return this.cnt;
9      }
10 }
```

Declarations:

PC	this
5	Counter

Classes:

Counter
Counter=Counter \rightarrow Counter
cnt=int
incr=(Counter \times int) \rightarrow int

Typing rules and semantic rules

Type
systems

The INFDEV
team

Introduction

Issues with
Python

Possible
solutions

Static typing

Typing rules
and semantic
rules

Conclusion

```

1  class Counter {
2      private int cnt;
3      public Counter() {
4          this.cnt = 0;
5      }
6      public int incr(int diff) {
7          this.cnt = (this.cnt + diff);
8          return this.cnt;
9      }
10 }
```

Declarations:

PC	diff	this
7	int	Counter

Classes:

Counter
Counter=Counter → Counter
cnt=int
incr=(Counter×int) → int

Typing rules and semantic rules

Type
systems

The INFDEV
team

Introduction

Issues with
Python

Possible
solutions

Static typing

Typing rules
and semantic
rules

Conclusion

```

1  class Counter {
2      private int cnt;
3      public Counter() {
4          this.cnt = 0;
5      }
6      public int incr(int diff) {
7          this.cnt = (this.cnt + diff);
8          return this.cnt;
9      }
10 }
```

Declarations:

PC	diff	this
8	int	Counter

Classes:

Counter
Counter=Counter \rightarrow Counter
cnt=int
incr=(Counter \times int) \rightarrow int

Typing rules and semantic rules

Type
systems

The INFDEV
team

Introduction

Issues with
Python

Possible
solutions

Static typing

Typing rules
and semantic
rules

Conclusion

```

1  class Counter {
2      private int cnt;
3      public Counter() {
4          this.cnt = 0;
5      }
6      public int incr(int diff) {
7          this.cnt = (this.cnt + diff);
8          return this.cnt;
9      }
10 }
```

Declarations:

PC	ret	diff	this
9	int	int	Counter

Classes:

Counter
Counter=Counter → Counter
cnt=int
incr=(Counter×int) → int

Typing rules and semantic rules

Type
systems

The INFDEV
team

Introduction

Issues with
Python

Possible
solutions

Static typing

Typing rules
and semantic
rules

Conclusion

```

1  class Counter {
2      private int cnt;
3      public Counter() {
4          this.cnt = 0;
5      }
6      public int incr(int diff) {
7          this.cnt = (this.cnt + diff);
8          return this.cnt;
9      }
10 }
```

Declarations:

PC
10

Classes:

Counter
Counter=Counter \rightarrow Counter
cnt=int
incr=(Counter \times int) \rightarrow int

Typing rules and semantic rules

Type
systems

The INFDEV
team

Introduction

Issues with
Python

Possible
solutions

Static typing

Typing rules
and semantic
rules

Conclusion

- Sometimes we may look a attribute a up from an instance c of a class
- This assumes the type of the attribute, which needs to be looked up in the class descriptor found in the declarations
- No declaration is further modified

$$\frac{\langle c, D \rangle \rightarrow \langle C, D \rangle \wedge \langle a, C \rangle \rightarrow \langle A, C \rangle \wedge T = U}{\langle (c.a), D \rangle \rightarrow \langle A, D \rangle}$$

Typing rules and semantic rules

Type
systems

The INFDEV
team

Introduction

Issues with
Python

Possible
solutions

Static typing

Typing rules
and semantic
rules

Conclusion

```
1 class Counter {  
2     public int cnt;  
3     public Counter() {  
4         this.cnt = 0;  
5     }  
6 }  
7 Counter c = new Counter();  
8 int a = c.cnt;
```

Declarations:

PC
1

Typing rules and semantic rules

Type
systems

The INFDEV
team

Introduction

Issues with
Python

Possible
solutions

Static typing

Typing rules
and semantic
rules

Conclusion

```

1  class Counter {
2      public int cnt;
3      public Counter() {
4          this.cnt = 0;
5      }
6  }
7  Counter c = new Counter();
8  int a = c.cnt;
    
```

Declarations:

PC	this
4	Counter

Classes:

Counter
Counter=Counter → Counter cnt=int

Typing rules and semantic rules

Type
systems

The INFDEV
team

Introduction

Issues with
Python

Possible
solutions

Static typing

Typing rules
and semantic
rules

Conclusion

```

1  class Counter {
2      public int cnt;
3      public Counter() {
4          this.cnt = 0;
5      }
6  }
7  Counter c = new Counter();
8  int a = c.cnt;
    
```

Declarations:

PC	this
5	Counter

Classes:

Counter
Counter=Counter → Counter cnt=int

Typing rules and semantic rules

Type
systems

The INFDEV
team

Introduction

Issues with
Python

Possible
solutions

Static typing

Typing rules
and semantic
rules

Conclusion

```

1  class Counter {
2      public int cnt;
3      public Counter() {
4          this.cnt = 0;
5      }
6  }
7  Counter c = new Counter();
8  int a = c.cnt;
    
```

Declarations:

PC
6

Classes:

Counter
Counter=Counter → Counter cnt=int

Typing rules and semantic rules

Type
systems

The INFDEV
team

Introduction

Issues with
Python

Possible
solutions

Static typing

Typing rules
and semantic
rules

Conclusion

```

1  class Counter {
2      public int cnt;
3      public Counter() {
4          this.cnt = 0;
5      }
6  }
7  Counter c = new Counter();
8  int a = c.cnt;
    
```

Declarations:

PC	c
8	Counter

Classes:

Counter
Counter=Counter → Counter cnt=int

Typing rules and semantic rules

Type
systems

The INFDEV
team

Introduction

Issues with
Python

Possible
solutions

Static typing

Typing rules
and semantic
rules

Conclusion

```

1  class Counter {
2      public int cnt;
3      public Counter() {
4          this.cnt = 0;
5      }
6  }
7  Counter c = new Counter();
8  int a = c.cnt;

```

Declarations:

PC	a	c
9	int	Counter

Classes:

Counter
Counter=Counter → Counter cnt=int

Typing rules and semantic rules

Type
systems

The INFDEV
team

Introduction

Issues with
Python

Possible
solutions

Static typing

Typing rules
and semantic
rules

Conclusion

- Sometimes we may call a method m up from an instance c of a class and with parameters p_i
- This assumes the return type of the method, provided that all parameter types match the types expected by the method
- No declaration is further modified

$$\frac{\langle c, D \rangle \rightarrow \langle C, D \rangle \wedge \langle m, C \rangle \rightarrow \langle (P_1 \times P_2 \times \cdots \times P_n \rightarrow R), C \rangle \wedge \langle p_i, D \rangle \rightarrow \langle P'_i, D \rangle \wedge P_i = P'_i}{\langle (c.m(..p_i..)), D \rangle \rightarrow \langle R, D \rangle}$$

Typing rules and semantic rules

Type
systems

The INFDEV
team

Introduction

Issues with
Python

Possible
solutions

Static typing

Typing rules
and semantic
rules

Conclusion

```
1 class Counter {  
2     private int cnt;  
3     public Counter() {  
4         this.cnt = 0;  
5     }  
6     public void Incr(int diff) {  
7         this.cnt = (this.cnt + diff);  
8     }  
9 }  
10 Counter c = new Counter();  
11 c.Incr(5);
```

Declarations:

PC
1

Typing rules and semantic rules

Type
systems

The INFDEV
team

Introduction

Issues with
Python

Possible
solutions

Static typing

Typing rules
and semantic
rules

Conclusion

```

1  class Counter {
2      private int cnt;
3      public Counter() {
4          this.cnt = 0;
5      }
6      public void Incr(int diff) {
7          this.cnt = (this.cnt + diff);
8      }
9  }
10 Counter c = new Counter();
11 c.Incr(5);

```

Declarations:

PC	this
4	Counter

Classes:

Counter
Counter=Counter \rightarrow Counter
Incr=(Counter \times int) \rightarrow void
cnt=int

Typing rules and semantic rules

Type
systems

The INFDEV
team

Introduction

Issues with
Python

Possible
solutions

Static typing

Typing rules
and semantic
rules

Conclusion

```

1  class Counter {
2      private int cnt;
3      public Counter() {
4          this.cnt = 0;
5      }
6      public void Incr(int diff) {
7          this.cnt = (this.cnt + diff);
8      }
9  }
10 Counter c = new Counter();
11 c.Incr(5);

```

Declarations:

PC	diff	this
6	int	Counter

Classes:

Counter
Counter=Counter \rightarrow Counter
Incr=(Counter \times int) \rightarrow void
cnt=int

Typing rules and semantic rules

Type
systems

The INFDEV
team

Introduction

Issues with
Python

Possible
solutions

Static typing

Typing rules
and semantic
rules

Conclusion

```

1  class Counter {
2      private int cnt;
3      public Counter() {
4          this.cnt = 0;
5      }
6      public void Incr(int diff) {
7          this.cnt = (this.cnt + diff);
8      }
9  }
10 Counter c = new Counter();
11 c.Incr(5);

```

Declarations:

PC
9

Classes:

Counter
Counter=Counter \rightarrow Counter
Incr=(Counter \times int) \rightarrow void
cnt=int

Typing rules and semantic rules

Type
systems

The INFDEV
team

Introduction

Issues with
Python

Possible
solutions

Static typing

Typing rules
and semantic
rules

Conclusion

```

1  class Counter {
2      private int cnt;
3      public Counter() {
4          this.cnt = 0;
5      }
6      public void Incr(int diff) {
7          this.cnt = (this.cnt + diff);
8      }
9  }
10 Counter c = new Counter();
11 c.Incr(5);

```

Declarations:

PC	c
11	Counter

Classes:

Counter
Counter=Counter \rightarrow Counter
Incr=(Counter \times int) \rightarrow void
cnt=int

Typing rules and semantic rules

Type
systems

The INFDEV
team

Introduction

Issues with
Python

Possible
solutions

Static typing

Typing rules
and semantic
rules

Conclusion

```

1  class Counter {
2      private int cnt;
3      public Counter() {
4          this.cnt = 0;
5      }
6      public void Incr(int diff) {
7          this.cnt = (this.cnt + diff);
8      }
9  }
10 Counter c = new Counter();
11 c.Incr(5);

```

Declarations:

c		PC	ret	arg ₁	this
Counter		11	null	int	Counter

Classes:

Counter
Counter=Counter → Counter
Incr=(Counter×int) → void
cnt=int

Typing rules and semantic rules

Type
systems

The INFDEV
team

Introduction

Issues with
Python

Possible
solutions

Static typing

Typing rules
and semantic
rules

Conclusion

```

1  class Counter {
2      private int cnt;
3      public Counter() {
4          this.cnt = 0;
5      }
6      public void Incr(int diff) {
7          this.cnt = (this.cnt + diff);
8      }
9  }
10 Counter c = new Counter();
11 c.Incr(5);

```

Declarations:

c		PC	ret	arg ₁	this
Counter		11	void	int	Counter

Classes:

Counter
Counter=Counter → Counter
Incr=(Counter×int) → void
cnt=int

Typing rules and semantic rules

Type
systems

The INFDEV
team

Introduction

Issues with
Python

Possible
solutions

Static typing

Typing rules
and semantic
rules

Conclusion

```

1  class Counter {
2      private int cnt;
3      public Counter() {
4          this.cnt = 0;
5      }
6      public void Incr(int diff) {
7          this.cnt = (this.cnt + diff);
8      }
9  }
10 Counter c = new Counter();
11 c.Incr(5);

```

Declarations:

PC	c
12	Counter

Classes:

Counter
Counter=Counter \rightarrow Counter
Incr=(Counter \times int) \rightarrow void
cnt=int

Typing rules and semantic rules

Type
systems

The INFDEV
team

Introduction

Issues with
Python

Possible
solutions

Static typing

Typing rules
and semantic
rules

Conclusion

- We may call a static method m from class C and with parameters p_i
- This assumes the return type of the method, provided that all parameter types match the types expected by the method
- We do not need to look up the class because it is already specified in the call
- No declaration is further modified

$$\frac{\langle m, C \rangle \rightarrow \langle (P_1 \times P_2 \times \dots \times P_n \rightarrow R), C \rangle \wedge \langle p_i, D \rangle \rightarrow \langle P'_i, D \rangle \wedge p_i = P'_i}{\langle (C.f(..p_i..)), D \rangle \rightarrow \langle R, D \rangle}$$

Typing rules and semantic rules

Type
systems

The INFDEV
team

Introduction

Issues with
Python

Possible
solutions

Static typing

Typing rules
and semantic
rules

Conclusion

```
1 class Utils {  
2     static public int AddThree(int a,int b,int c) {  
3         return ((a + b) + c);  
4     }  
5 }  
6 int x = Utils.AddThree(10,20,30);
```

Declarations:

PC
1

Typing rules and semantic rules

Type
systems

The INFDEV
team

Introduction

Issues with
Python

Possible
solutions

Static typing

Typing rules
and semantic
rules

Conclusion

```

1 class Utils {
2     static public int AddThree(int a,int b,int c) {
3         return ((a + b) + c);
4     }
5 }
6 int x = Utils.AddThree(10,20,30);
    
```

Declarations:

PC	a	b	c
3	int	int	int

Classes:

Utils
AddThree=(int×int×int) → int

Typing rules and semantic rules

Type
systems

The INFDEV
team

Introduction

Issues with
Python

Possible
solutions

Static typing

Typing rules
and semantic
rules

Conclusion

```
1 class Utils {  
2     static public int AddThree(int a,int b,int c) {  
3         return ((a + b) + c);  
4     }  
5 }  
6 int x = Utils.AddThree(10,20,30);
```

Declarations:

PC
5

Classes:

Utils
AddThree=(int×int×int) → int

Typing rules and semantic rules

Type
systems

The INFDEV
team

Introduction

Issues with
Python

Possible
solutions

Static typing

Typing rules
and semantic
rules

Conclusion

```

1  class Utils {
2      static public int AddThree(int a,int b,int c) {
3          return ((a + b) + c);
4      }
5  }
6  int x = Utils.AddThree(10,20,30);
    
```

Declarations:

PC	x
7	int

Classes:

Utils
AddThree=(int×int×int) → int

Type
systems

The INFDEV
team

Introduction

Issues with
Python

Possible
solutions

Static typing

Typing rules
and semantic
rules

Conclusion

Reading typing rules

- The constructor of a class is simply a specially named static method
- It has no further typing rules

Typing rules and semantic rules

Type
systems

The INFDEV
team

Introduction

Issues with
Python

Possible
solutions

Static typing

Typing rules
and semantic
rules

Conclusion

```
1 class CounterFrom {  
2     public int cnt;  
3     public CounterFrom(int cnt0) {  
4         this.cnt = cnt0;  
5     }  
6 }  
7 CounterFrom c = new CounterFrom(100);
```

Declarations:

PC
1

Typing rules and semantic rules

Type
systems

The INFDEV
team

Introduction

Issues with
Python

Possible
solutions

Static typing

Typing rules
and semantic
rules

Conclusion

```

1  class CounterFrom {
2      public int cnt;
3      public CounterFrom(int cnt0) {
4          this.cnt = cnt0;
5      }
6  }
7  CounterFrom c = new CounterFrom(100);

```

Declarations:

PC	cnt0	this
4	int	CounterFrom

Classes:

CounterFrom
CounterFrom=(CounterFrom×int) → CounterFrom cnt=int

Typing rules and semantic rules

Type
systems

The INFDEV
team

Introduction

Issues with
Python

Possible
solutions

Static typing

Typing rules
and semantic
rules

Conclusion

```

1 class CounterFrom {
2     public int cnt;
3     public CounterFrom(int cnt0) {
4         this.cnt = cnt0;
5     }
6 }
7 CounterFrom c = new CounterFrom(100);

```

Declarations:

PC	cnt0	this
5	int	CounterFrom

Classes:

CounterFrom
CounterFrom=(CounterFrom×int) → CounterFrom cnt=int

Typing rules and semantic rules

Type
systems

The INFDEV
team

Introduction

Issues with
Python

Possible
solutions

Static typing

Typing rules
and semantic
rules

Conclusion

```

1  class CounterFrom {
2      public int cnt;
3      public CounterFrom(int cnt0) {
4          this.cnt = cnt0;
5      }
6  }
7  CounterFrom c = new CounterFrom(100);

```

Declarations:

PC
6

Classes:

CounterFrom
CounterFrom=(CounterFrom×int) → CounterFrom cnt=int

Typing rules and semantic rules

Type
systems

The INFDEV
team

Introduction

Issues with
Python

Possible
solutions

Static typing

Typing rules
and semantic
rules

Conclusion

```

1 class CounterFrom {
2     public int cnt;
3     public CounterFrom(int cnt0) {
4         this.cnt = cnt0;
5     }
6 }
7 CounterFrom c = new CounterFrom(100);

```

Declarations:

PC	c
8	CounterFrom

Classes:

CounterFrom
CounterFrom=(CounterFrom×int) → CounterFrom cnt=int

Type
systems

The INFDEV
team

Introduction

Issues with
Python

Possible
solutions

Static typing

Typing rules
and semantic
rules

Conclusion

Conclusion

Type
systems

The INFDEV
team

Introduction

Issues with
Python

Possible
solutions

Static typing

Typing rules
and semantic
rules

Conclusion

Looking back

- Python is brittle and breaks easily
- Static typing is a way to run a coarse simulation of the program
- If type checking fails, then the program cannot be guaranteed to run correctly, and we get a compiler error
- Safer programming, but at the cost of being able to run less programs that might still be valid

Type
systems

The INFDEV
team

Introduction

Issues with
Python

Possible
solutions

Static typing

Typing rules
and semantic
rules

Conclusion

The best of luck, and thanks for the
attention!