50.054 Semantic Analysis

ISTD, SUTD

Learning Outcomes

1. Articulate the meaning of program semantics
2. List different types of program semantics.
3. Explain the limitation of static analysis.

Recap

Recall the compiler pipeline

```
[Lexing] -->[Parsing] --> [Semantic Analysis] --> [Optimization]
                                                         |
                                                         ↓
                                          [Target Code Generation]
```

In fact

```
                                         -----------------
                                         ↓               |
[Lexing] -->[Parsing] --> [Semantic Analysis] --> [Optimization]
                                                         |
                                                         ↓
                                          [Target Code Generation]
```

Syntax Analysis vs Semantic Analysis

- Syntax Analysis - verifies the given code conforms to the grammar rules
- Semantic Analysis - verifies the given code behaves according to the *expectation*

The program semantics define the behavior of a program.

Dynamic Semantics

Define the meaning and behaviors of the given program. The term “behavior” could mean

1. How does the program get executed?
2. What does the program compute / return?

Static Semantics

Describe a set of properties that the given program holds. For example,

```
x = input;
y = 1;
z = 0;
if (x - y) { // ill-type, int can't be used in place of an if condition.
    z = 1;
} else {
}
```

Goals of Semantic Analysis

1. Fault Detection
2. Optimization

Optimization

```
x = input;
y = 0;
s = 0;
while (y < x) {
    y = y + 1;
    t = s;  // t is not used.
    s = s + y;
}
return s;
```

```
1: x <- input
2: y <- 0
3: s <- 0
4: b <- y < x
5: ifn b goto 10
6: y <- y + 1
7: t <- s // t is not used
8: s <- s + y
9: goto 4
10: rret <- s
11: ret
```

Fault Detection

```
x = input;

while (x >= 0) {
    x = x - 1;
}
y = Math.sqrt(x); // error, can't apply sqrt() to a negative number.
return y;
```

Fault Detection

```
1: x <- input
2: t1 <- 0 < x
3: t2 <- 0 == x
4: t3 <- t1 + t2 // t1 or t2
5: ifn t3 goto 8
6: x = x - 1
7: goto 2
8: y <-  Math.sqrt(x) // x is definitely negative
9: rret <- y
10: ret
```

Different types of semantics analysis

- Dynamic (semantic) analysis - find faults and ascertains quality by supplying actual inputs to the target programs.
    - Testing
    - Run-time verification - finding bugs by log checking
    - Fuzzing

Different types of semantics analysis

- Static (semantic) analysis
  - Type checking
  - Name analysis
  - Control flow analysis
  - Data flow analysis
  - Model checking
  - Abtract Intepretation
  - Symbolic Execution (???)

Rice's theorem

All non-trivial semantic properties of programs are undecidable, i.e. there exists no algorithm that can decide all semantic properties for all given programs.

e.g.

```
def f(path):
  p = open(path, "r")
  x = 1
  if eval(p):
    x = -1
  return x
```

if there exists an algorithm *A* which statically decides `x`'s sign, we solve the halting problem in general!

Limitation of Static Analysis

Static analyses are computing a sound and conservative approximation of the run-time properties.

- Are you're a programmer?
-Yes
-So your code must be well optimized and secure.