Semantic 10주차

2022년 5월 11일 수요일 오전 10:07

Semantics

- Syntax and type이 valid한 프로그램
- 프로그램 의미의 정확한 이해를 제공한다
- Programming language는 syntax + type system + semantics가 잘 정의 될때 완성된다

접근 방법

- **Operational semantics**
  - 프로그램의 동작 과정을 정의
- **Denotaional semantics**
  - 프로그램의 의미를 함수 형태로 정의
  - 이번에 공부할 접근 방법
  - Statements as state transfoming mathematical functions 상태 변환 함수로 정의
  -
- **Axiomatic semantics**
  - 프로그램의 시작 상태와 종료상태를 논리적 선언 ,assertion 형태로 정의

Short circuit Evalution

c,c++가 사용하는 방법

AND

- **a and b**
  - 만약 a가 false라면 뒤에 b 결과 상관없이 false 할수있다
  - 즉 a가 참일때만 b를 계산하는게 short circuit and이다
  -

OR

- **a or b**
  - 만약 a가 참이면 b를 계산하지 않아도 전체식이 True인것을 확인할수있다
  - 일반 or(a&b 모두 계산하는 or)과는 다르다

프로그램을 더 간결하게 만들수있다.

Example: what is assumed here?

```
Node p = head;
while (p != null && p->info != key)
      p = p.next;

if (p == null)
      ...       // key not found in the list
else
     ...        // found it
```

화면 캡처: 2022-05-11 오전 10:40

Linked list에서 주어진 key를 찾는 프로그램

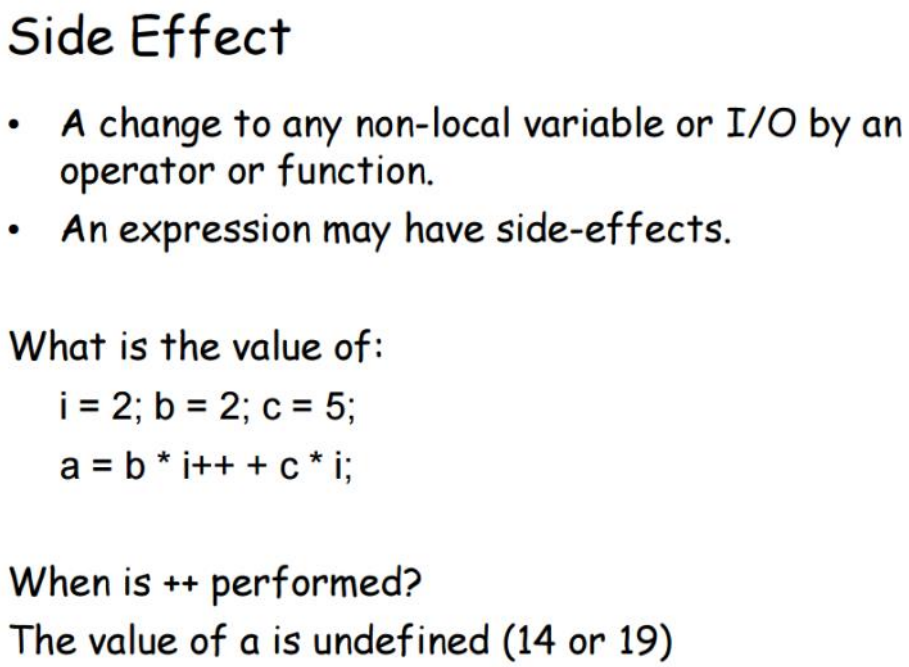
만약 p==NULL이면

- Short circuit 사용한다고 하면 p!=NULL에서 False로 판명되고 빠져나온다
- 그러나 short circuit이 아니라면 뒤의 조건 p-> info != key를 수행하게 되는데
    - NULL-> info는 잘못된 값이다 -> 프로그램이 정상 작동이 안된다

수학에서는 표현식과 연산자에 의해 의미가 결정되지만 computer에서는 이 법칙이 성립하지는 않는다 즉 수학과 완전히 같지 않다 왜냐하면 computer는 유한한 메모리를 가지고 있다 즉 길이에 제한이 있다.

예_) (a+b)+c =a+(b+c)가 수학에서는 같지만 computer에서는 같지 않다

그럼 언제 불일치 할까?
-> **overflow -> computer에서는 수학과 같지 않음을 알수있는 예**



화면 캡처: 2022-05-11 오전 10:58

Non local variable(global variable?) 또는 IO 또는 function에서 값이 바뀌는 부작용

What is the state of a running program?

프로그램의 상태를 정의한다.

The state of a program : the collection of all active objects and their current values

== State를 함수로 보라 Name - > Value 변수이름을 주면 변수이름과 대응하는 값을 반환하는 함수가 state라고 할수있다 , 변수들의 값을 다 모아놓은 것이다

```
// compute the factorial
of n
1    void main ( ) {
2     int n, i, f;
3      n = 3;
4      i = 1;
5      f = 1;
6      while (i < n) {
7              i = i + 1;
8              f = f * i;
9     }
10  }
```

n	i	f
undef	undef	undef
3	undef	undef
3	1	undef
3	1	1
3	1	1

화면 캡처: 2022-06-05 오후 1:02

```
// compute the factorial
of n
1    void main ( ) {
2     int n, i, f;
3      n = 3;
4      i = 1;
5      f = 1;
6      while (i < n) {
7              i = i + 1;
8              f = f * i;
9     }
10  }
```

n	i	f
3	3	6
3	3	6

화면 캡처: 2022-05-11 오전 11:15

선언만되어있고 값이 할당 되어있지 않으면 undef로 가고 할당이 될때 undef가 할당된 값으로 변경이 된다

Factorial 함수에서 state를 분석해 보자
-> pdf자료 참고해라

- Int n,I,f 선언이라서 undef의 상태가 된다
- N =3,i=1,f=1이 정의되면 n I f 의 state가 3 1 1로 바뀐다
- While(i<n)은 변화 없다
- i=i+1 하면 i는 1에서 2가 된다
- f=f*I 하면 f는 1에서 2가 된다
- 이걸 n번 만큼 반복을 한다

이 방법으로 일일이 다 프로그램 검사하는건 불가능하다

그래서 3가지 접근 방법으로 semantic 접근해서 정의한다

Assignment: Copy Semantics vs. Reference Semantics

a = b;

- Copy
  - a, b have same value.
  - Changes to either have no effect on the other.
  - Used in imperative languages.
- Reference
  - a, b point to the same object. (a and b are alias for the same object)
  - A change in object state affects both
  - Used by many object-oriented languages.

화면 캡처: 2022-05-11 오전 11:21

Copy와 reference semantic가 있다.

Control Flow semantics

Turing complete : 의미있는 함수

Turing complete하려면 3가지를 만족해야한다

- **Statement sequencing**
  - S1, S2
  - S1계산 결과가 S2에 input으로 들어갈수있는 상황?
  - 순차적으로 계산
- **Conditional statement**
  - If ~ else ~
  - 조건에따라 skip 또는 다른 선택 가능
  - if문이 참이면 if문의 statement가 다음 state되고 else라면 else 문의 statement가 다음 state됨
- **Lopping statement**
  - While(expression)
  - Expression이 false면 while문 terminate한다

모든 프로그램을 goto없이 구성 할수있다 -> 증명한 사람 있다. Structed programming

Semantic interpretation

많은 프로그램은 의미 없는 프로그램이다
알고리즘은 partial function[값이 정의 되지 않은 domain있는 함수]이다

Historical Problem

Valid program had different meanings on different machines

- *More than one (e.g.) size of an int or float*

Problem was lack of precision in defining meaning => Semantics

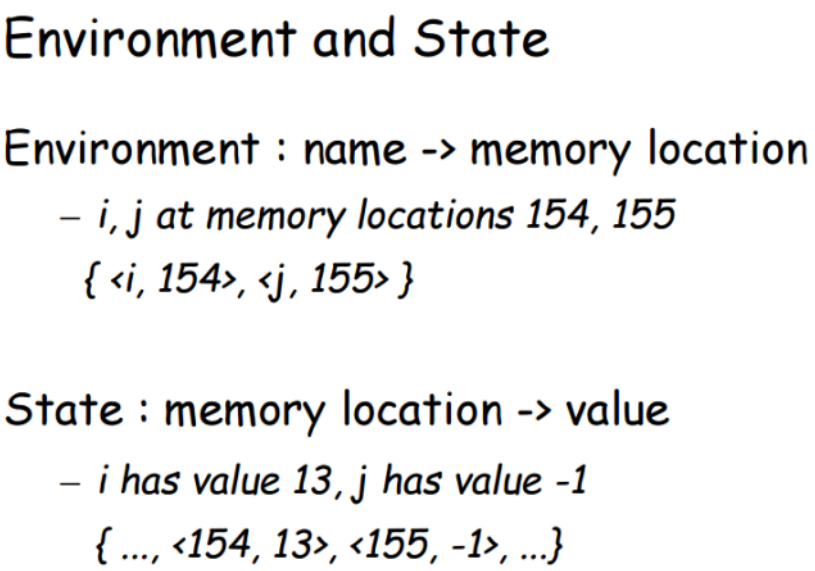
화면 캡처: 2022-05-11 오후 12:42

```
while (true)
    ;
return a;
```

화면 캡처: 2022-05-11 오후 12:40

0으로 나누거나 최대값에 +1 하거나 무한 루프등 의미 없는 프로그램들이다

방법은 위의 3가지 방법 으로 접근할수있다

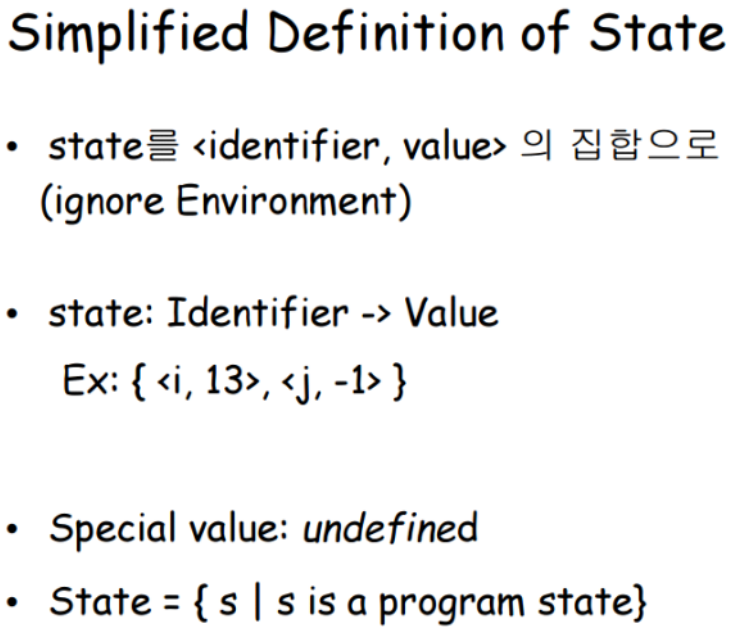


화면 캡처: 2022-05-11 오후 12:43

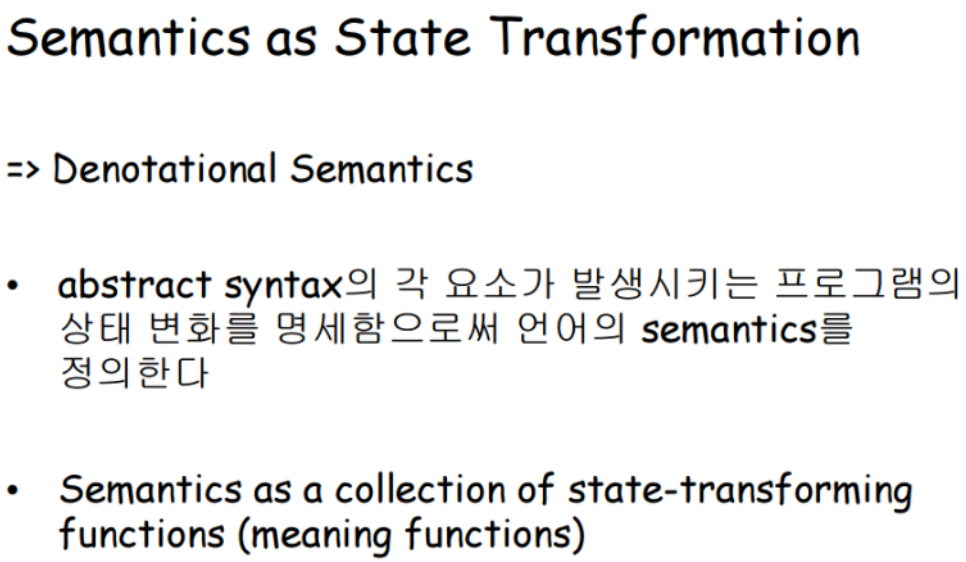
Evironment : name -> memory location 대응 시킴 <I,154>

State : memory location -> value
: I has value 13 {....<154,13>....}

<identifier,value>의 집합으로 정의한다



화면 캡처: 2022-05-11 오후 12:47



화면 캡처: 2022-05-11 오후 12:48

State transforming functions == meaning functions

Clite Semantics

의미 있는 함수 M
프로그램을 실행했을때 그 프로그램의 상태

State is state의 집합이다 대소 의미 구분 해야한다

M : Program -> State
M : Statement * State - > State
M: Expression * State - > value

프로그램의 의미 : 실행했을때의 정의가 무엇인가
Declarations dec; Statement body 로 이루어져 있다.

초기 상태가 주어졌을때의 body가 프로그램이다

초기 상태 : 선언된 변수를 다 초기화 한 상태

Meaning Rule 8.1

The meaning function for a program

Abstract syntax

Program = Declarations *decpart*; Statement *body*

Meaning function M for Program

M: Program -> State // signature (or type)

M(Program p) = M(p.body, InitialState(p.decpart))

InitialState(p.decpart) is

$\{<v_1, undef_1>, \ldots, <v_m, undef_m>\}$

화면 캡처: 2022-05-18 오후 3:14

Undef는 값 자체의 타입은 담고있는 변수에 따라서 타입다르다

Implementation of Meaning Function M

```
//state는 hashmapd을 상속한다
Public class State extends HashMap{....}

State M(Program p)
{
	//program = Declarations decpart; Statement body

	//가정 : 모든 변수가 초기화 되어있는 상태일때 body가 program이다
	return M(p.body , initialState(p.decpart));
}
```

Type checker에서는 <변수이름,type>을 map으로 표현했다.
M function은 <변수이름, 값>으로 표현한다

InitialState() 함수

모든 변수 선언을 초기화하는 함수

initialState() 함수

```
State initialState (Declarations d) {
    State state = new State( );
    for (Declaration decl : d)
        state.put(decl.v,  Value.mkValue(decl.t));
    }
    return state;
}
```

화면 캡처: 2022-05-18 오후 3:22

Meaning function for Statement

Abstaract Syntax

Statement = Skip | Block | Assignment | Loop | Conditional

M : Statement * State -> State

M: *Statement* x *State* -> *State*

// specification
M(Statement s, State state)
= M((Skip)s, state) if s is a *Skip*
= M((Assignment)s, state) if s is an *Assignment*
= M((*Conditional*)s, state) if s is an *Conditional*
= M((*Loop*)s, state) if s is an *Loop*
= M((*Block*)s, state) if s is an *Block*

화면 캡처: 2022-05-18 오후 3:26

종류에 따라서 의미가 달라진다 그래서 자바로 나타내면

```
// Java 구현

State M(Statement s, State state) {
    if (s instanceof Skip) return M((Skip)s, state);
    if (s instanceof Assignment) return M((Assignment)s, state);
    if (s instanceof Block) return M((Block)s, state);
    if (s instanceof Loop) return M((Loop)s, state);
    if (s instanceof Conditional) return M((Conditional)s, state);
    throw new IllegalArgumentException( );
}
```

화면 캡처: 2022-05-18 오후 3:26

Skip 의 경우

The state is unchanged

상태가 안바뀐다 , 상태 그대로 유지

```
M(Skip s, State state)
{
    return state;
}
```

Assignment 의 경우

Assignment = Variable target; Expression source;

Assignment는 input state에 있는 target variable에 source expression 값을 저장한다

```
State M(Assignment a, State state) {
    return state.onion(a.target, M(a.source, state));
}
// onion
// M(a.source, state)
```

화면 캡처: 2022-05-18 오후 3:36

Onion : 변수와 값을 주면 변수의 값을 파라매터로 넘긴 값으로 바꿔주는 함수

Conditioal 의 경우

Meaning Rule 8.4

Conditional = Expression test;
Statement thenbranch, elsebranch

The meaning of a conditional is:
- *If the test is true, the meaning of the thenbranch;*
- *Otherwise, the meaning of the elsebranch*

화면 캡처: 2022-05-18 오후 3:39

```
State M(Conditional c, State state) {
    if (M(c.test, state).boolValue( ))
        return M(c.thenbranch, state);
    else
        return M(e.elsebranch, state);
}
```

화면 캡처: 2022-05-18 오후 3:39

Expression Semantics

We ignore the issue of side Effect for now

Expressions

M: Expression x State → Value

Expression = Variable | Value | Binary | Unary
Binary = BinaryOp op; Expression term1, term2
Unary = UnaryOp op; Expression term
Variable = String id
Value = IntValue | BoolValue | CharValue | FloatValue

화면 캡처: 2022-05-18 오후 3:42

M : Expression * state -> Value라고 하는 이유 (정확하지는 않음)

x+3 이라는 표현식이거나 x+y라는 표현식 있다면 x또는 y의 값을 알아야한다 즉 state [변수와 값 대응한거]가 필요하다 그래야 계산을 할수 있으니까

아마 위의 내용도 이런 맥락이지 않을까 생각한다

// java 구현

```
Value M(Expression e, State state) {
    if (e instanceof Value)  return (Value)e;
    if (e instanceof Variable)  return (Value)(state.get(e));
    if (e instanceof Binary) {
        Binary b = (Binary)e;
        return applyBinary(b.op, M(b.term1, state),
                                 M(b.term2, state);
    }
    ...
}
```

화면 캡처: 2022-05-18 오후 3:46

어떤 의미 인지확인한다 -> 값을 명확히 해준다?

stackFrame의 set함수가 변수 ,값이 undef되어있는것을 바꿔준다
-> 선언만 되어있으면 아직 undef 그리고 할당이 되거나 값이 바뀌면 그때마다 변수와 대응하는 값을 바꿔준다.