Type system

2022년 5월 4일 수요일 오전 5:25

Type System

- 프로그래머가 새로운 데이터 타입을 정의하게 해준다
- Type system은 실행전에 type Error를 검출하고 예방하기 위해 필요하다
- CFG는 type Error 정의 불가능하다 타입 정보를 표현할수없어서 타입 시스템을 만들어서 타입을 체크한다

Designing a Type system for Clite

- Int , float, bool
- Not support Struct, Union...
- No nesting function, no locals
- Assignment에만 inplicit type conversion이 허용된다
  - Float x = 3; 하면 3.0으로 변환됨 이 과정만 허용된다
- Mixed mode operation은 허용 안함
  - 2 + 3.5 같은 예가 mixed mode operation이다
  - 실수 + 실수 , 정수 + 정수는 있지만 섞여 있으면 clite에서는 지원하지 않는다
- Strong type / static type 지원한다

Type checking == static symantics == type system 분석

Implementing a type system for clite

- Type 규칙 집합을 정의 해야한다
- 각 변수는 유니크하다
- 키워드는 변수명이 될수 없다 ex) if,for,int는 변수명 될수없다
- 변수는 reference되기 전에 선언 되어야 한다
  - Declaration => Statement

Parsing이 완료 되고 AST가 만들어지면 type checking이 가능한다
AST가 만들어진 이후 Type Checking을 한다

AST type checking 하는 방법

$$
\begin{aligned}
\mathit{Program} &= \mathit{Declarations}\ \texttt{decpart;}\ \ \mathit{Statements}\ \texttt{body;}\\
\mathit{Declarations} &= \mathit{Declaration}^*\\
\mathit{Declaration} &= \mathit{VariableDecl} \mid \mathit{ArrayDecl}\\
\mathit{VariableDecl} &= \mathit{Variable}\ \texttt{v;}\ \mathit{Type}\ \texttt{t}\\
\mathit{ArrayDecl} &= \mathit{Variable}\ \texttt{v;}\ \mathit{Type}\ \texttt{t;}\ \mathit{Integer}\ \texttt{size}\\
\mathit{Type} &= \ \texttt{int} \mid \texttt{bool} \mid \texttt{float} \mid \texttt{char}\\
\mathit{Statements} &= \mathit{Statement}^*\\
\mathit{Statement} &= \mathit{Skip} \mid \mathit{Block} \mid \mathit{Assignment} \mid \mathit{Conditional} \mid \mathit{Loop}\\
\mathit{Skip} &=\\
\mathit{Block} &= \mathit{Statements}^*\\
\mathit{Conditional} &= \mathit{Expression}\ \texttt{test;}\ \mathit{Statement}\ \texttt{thenbranch, elsebranch}\\
\mathit{Loop} &= \mathit{Expression}\ \texttt{test;}\ \mathit{Statement}\ \texttt{body}
\end{aligned}
$$

화면 캡처: 2022-05-04 오전 6:01

Type Rules

- 모든 변수는 선언 되어있어야 한다
- 모든 변수는 유니크한 이름을 가져야한다
- ★ 만약 선언부가 유효하고 실행부의 바디가 각 선언부에 대하여 유효하면 프로그램이 유효하다 -> 프로그램이 타입 오류가 없는지 검사한다는 의미
  - V(Program p)의 의미이다 -> V(declations) V(functions)등을 진행하잖아 그 의미이다.
- 실행문이 유효하다

//Type checker 관련 내용 종이에 적을 필요 있다

TypeMap을 이용해서 <변수,타입>을 mapping해서 타입이 일치하는지 선언이 된 변수인지 확인할수있다.

The validity of statement

- Skip은 항상 유효하다
- **Assignment**
  - 변수 = expression
  - 변수 타입이랑 expr이랑 타입이 같아야하고 expr타입이 맞아야 한다 (타입 호환성)이 맞아야한다
  - 예_) 타겟이 float이면 소스는 float이거나 int,
  - 예_ 타겟이 int이면 소스는 int이거나 char [implicit 변환]
- **Condition**
  - Expr이 유효하고 bool 타입을 가진다
  - Then 브랜치와 else 브랜치 실행문이 유효하다
- **Loop**
  - Expr 이 유효하고 bool 타입 가짐
  - Body가 유효함 -> Statement가 유효해야한다
- **Block (body)**
  - 모든 실행문이 유효하다

Validity of an Expression

: value,valiable,binary, unary

- 값(Value)은 항상 유효하다 [literal]
- 변수는 선언되어있으면 유효하다
- 바이너리: mixed mode 타입이 없으면 유효하다 ,no mixed mode operation
  - x+5 [x가 integer이면 맞지만 x가 다른 타입이면 맞지 않다]
  - -> 실제 구현한 프로그램에서는 바이너리 연산할때 지원하게 구현함
  - Relation operation은 같은 타입만 비교관계 할수있다
  - Op,term1,term2가 있다 term1과 term2는 같은 타입이어야한다

- **Unary**
  - !이면 bool ,
  - - 이면 int or float
  - Int(i) 캐스팅할때 이때 i는 char또는 float이다 -> 규칙 정하기 나름이다
  - Float(i)할때 char(i)할때 i는 inte여야한다
  - 즉 conversion

Implementing a type checker using JAVA

Type rules따른 유효성을 체크하기 위해 함수(overloaded)function V집합을 정의함

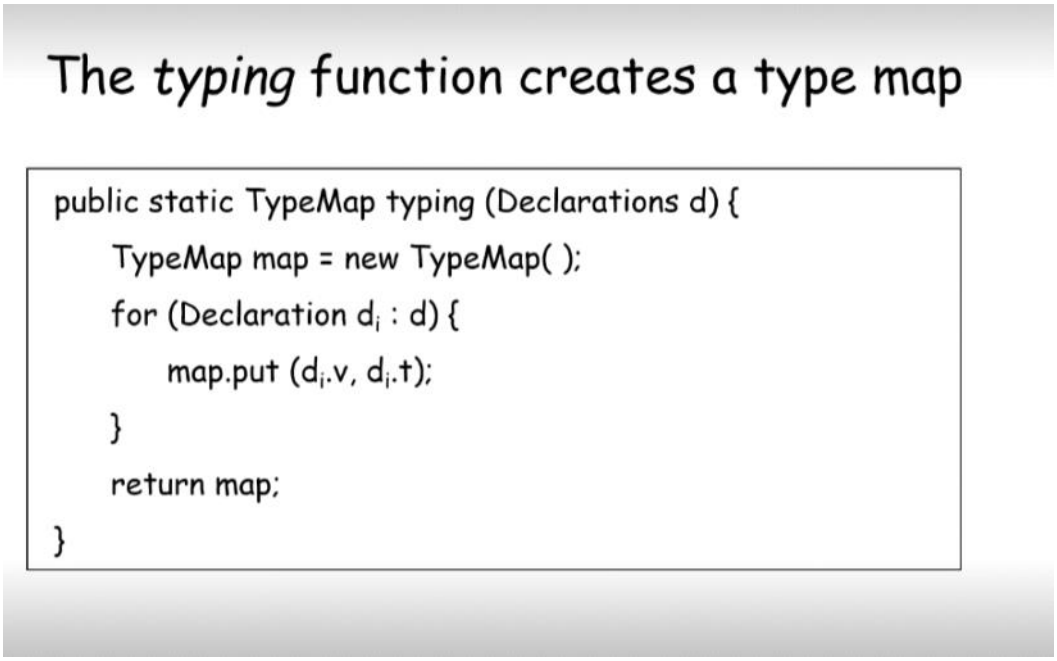
Return True or False
Based on Abstract Syntax 노드 타입 개수 만큼 V함수 존재

Type map

일종의 딕셔너리

<name, type>

- Typing(d)
  - 선언부 d를 위한 type map을 만드는 함수



화면 캡처: 2022-05-04 오전 10:15

- **typeOf(e)**
  - Type of Expression e를 리턴하고 계산하는 함수

All referenced variables must be declared : 각각의 type map에 존재하는 참조된 변수를 만듬으로써 이 룰을 지킨다

Validity of declarations

: 중복 선언 되면 안된다 그래서 확인을 한다

Declarions are valid if All declared variables have unique names.

```
public static void V (Declarations d) {
    for (int i=0; i<d.size() - 1; i++)
        Declaration di = d.get(i);

        // check if it has been declared elsewhere
        for (int j=i+1; j<d.size(); j++) {
            Declaration dj = d.get(j);
            check( ! (di.v.equals(dj.v)),
                "duplicate declaration: " + dj.v);
        }
}
```

화면 캡처: 2022-05-04 오전 10:18

A program is valid if

- **Its Declarations are valid and**
- **Its body of Statements is valid with respect to the type map for those Declarations**

```
public static void V (Program p) {   // V(Program)
    V (p.decpart);                        // V(Declarations)
    V (p.body, typing (p.decpart));   // V(Statements)
}
```

화면 캡처: 2022-05-04 오전 10:22

만약 오류라면 Exception handling으로 빠져 나온다

실제 프로그램 구현

```
Publicstatic void V(Program p)
{
        //Since the TypeMap Is an extension of HashMap, strictly obeying the formalized type rules is erroneous.
        //Concrete syntax guarentees that the final function declared must be main ,so that typechecking has been omitted.

        V(p.globals);          //전역변수에대한유효성체크
        V(p.functions);     //함수에대한유효성체크
        V(p.globals,p.functions);     //전역변수와함수이름겹치는지에대한처리
        for(Functionfi:p.functions)     //mapping하는작업
                V(fi,typing(p.globals,p.functions,fi));          //함수하나에대한유효성체크한다
}
```

Validity of a Statement

```
public static void V (Statement s, TypeMap tm) {
    if (s == null) throw new IllegalArgumentException("AST error: null stmt");
    if (s instance of Skip) return; // ok
    if (s instance of Assignment) {
      Assignment a = (Assignment) s;
      check( tm.containsKey(a.target), "undefined target in assignment: " + a.target);
      V(a.source, tm);      // check the validity of the source expression
      Type ttype = (Type)tm.get(a.target);   // get the declared type of the target
      Type srctype = typeOf(a.source, tm);  // evaluate the type of the source expr
      if (ttype != srctype) {
        if (ttype == Type.FLOAT) check( srctype == Type.INT, "type error..", a.target);
        else if (ttype == Type.INT) check (srctype == Type.CHAR, "type error ", a.target);
        else check(false, "mixed mode assignment to " + a.target);
    }
    return;
}
```

화면 캡처: 2022-05-04 오전 10:26

Statement 개선 버전

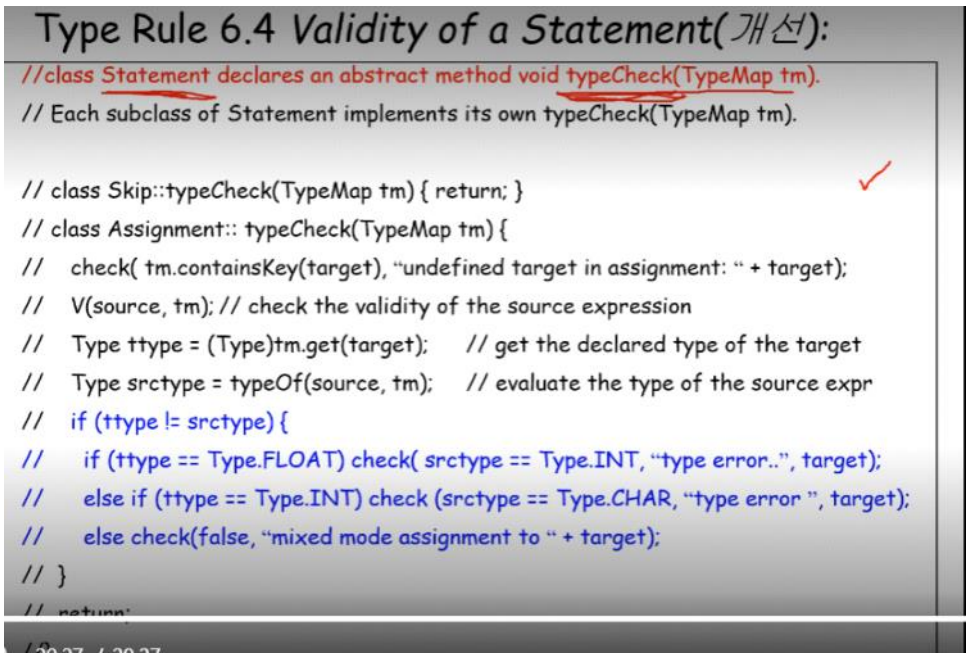
```
// 교재 프로그램에 대한 개선안
// class hierarchy 설계를 다음과 같이 변경할 수 있다.

public static void V (Statement s, TypeMap tm) {
    if (s == null) throw new IllegalArgumentException("AST error: null stmt");
    return s.typeCheck(tm);
}
//class Statement declares an abstract method void typeCheck(TypeMap tm).

// Each subclass of Statement implements its own typeCheck(TypeMap tm).
```

Rule 6.5 Validity of an Expression 검사 구현을 위한 class hierarchy도 유사한 방법으로 할 수 있다.

화면 캡처: 2022-05-04 오전 10:42



화면 캡처: 2022-05-04 오전 10:47

계층적인 class
Tycasting 할 필요 없다 , 모든 객체를 다 볼 수 있다는 전제가 존재

Vaildity of an Expression

```
public static void V (Expression e, TypeMap tm) {
    if (e instance of Value) return; // every value is valid
    if (e instance of Variable) {
     Variable v1=(Variable)e; check( tm.containsKey(v1), "undeclared variable: " + v1);
     return;
    }
    if (e instance of Binary) {
     Binary b = (Binary) e;    // downcasting - undesirable!
     Type typ1 = typeOf(b.term1, tm); Type typ2=typeOf(b.term2, tm);
     V(b.term1, tm); V(b.term2, tm);
     if (b.op.ArithmeticOp())
          check((typ1 == typ2) && (typ1 ==Type.INT || typ1 == Type.FLOAT),
          "type error for " + b.op);
     else if (b.op.RelationalOp())
          ...
     }
     ...
    throw new IllegalArgumentException("Should never come here");  }
```

화면 캡처: 2022-05-04 오전 10:25

ArithmeicOP : 산술연산은 정수거나 실수형 계산할때만 유효하니까 ty1과 ty2의 타입이 같은지 확인해주는 작업 필요하다

실제 구현에서는 mixed mode 지원 하게 함 - >Typetransfomer에서 typechecking이 끝나서 넘어오면 자동으로 형 변환해서 계산을 해준다

The typeOf() function

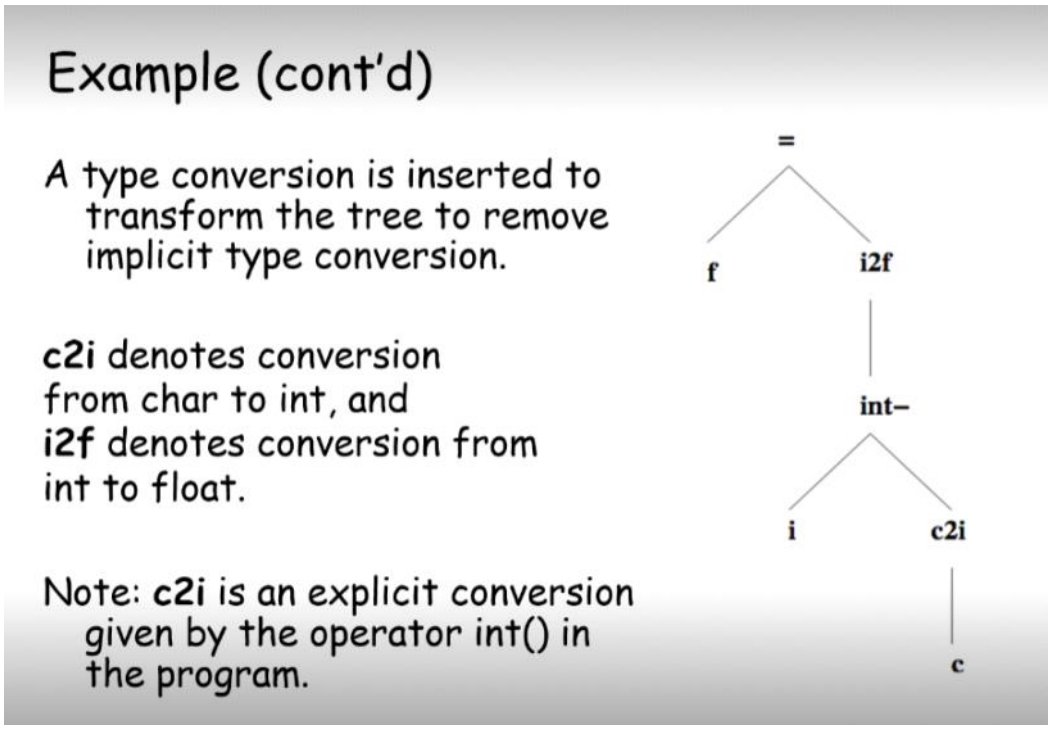
```
// Precond: e is a valid expression
public static Type typeOf (Expression e, TypeMap tm) {
    if (e instance of Value) return ((Value)e).type;
    if (e instance of Variable) {
     Variable v = (Variable)e;
     return tm.get(v);
    }
    if (e instance of Binary) {
     Binary b = (Binary) e;
     if (b.op.ArithmeticOp()) {
          return typeOf(b.term1, tm);
     else if (b.op.RelationalOp()) || b.op.BooleanOp())
          retun (Type.Bool);
    }
    if (e instanceof Unary) {
     Unary U = (Unary)e;
     if (u.op.NotOp()) return (Type.Bool);
     ...
     else if (u.op.intOp()) return (Type.INT);
     ...
    }
}
```

화면 캡처: 2022-05-04 오전 10:35

Type을 반환하는 함수이다
Value, Vaiable, Binary, Unary등에 있는 type을 return 해주는 함수이다.

Implicit Type conversion

Clite Assignment supports two implicit (widening) conversions for assignments.
– int to float or char to int.
All other type conversions require explicit type casting.
Implicit type conversions은 코드 생성 전에 제거되어야할 필요가 있다 생성되고 나면 explicit으로 바꿔줘야함
그리고 트리도 변경해서 i2f이런거 삽입해줘야함
우리는 필요하면 inserting explicit conversions를 삽입함으로싸 AST를 변형함



화면 캡처: 2022-05-04 오전 10:39