صالح زارع زاده ۶۱۰۳۹۶۱۰۹

پروژه کامپایلر فاز ۴ سمنتیک انالایزر

هدف از این فاز پروژه هندل کردن ارور هایی است که در ۲ فاز قبلی نمیتوانستیم هندل کنیم که با مشخص کردن تایپ ها و استتیک تایپ است

در ابتدا باید همه ی تایپ ها را بدست اوریم درواقع باید کلاس هایی که در برنامه تعریف شده و کلاس های پایه را در نظر بگیریم سپس باید ارور های مربوط به ان ها را هندل کنیم که عبارت اند از

ارث بری از تایپی که وجود ندارد

ارث بری از تایپ های استرینگ و بولین و اینتیجر زیرا نباید مقادیر متغییر های ان ها تغییر کند

نبودن کلاس پایه (main)

و نبودن تابع مین یا گرفتن ورودی در آن

حال باید تابع ها و ابجکت ها (متغیر ها) را برسی کنیم که ارور های مربوط به ان ها عبارت اند از: تعریف یک ابجکت از تایپی که وجود ندارد

تعریف ورودی توابع از تایپی که وجود ندارد

دوباره تعریف کردن ابجکت ها با اسم یکسان در کلاس های پدر

دوباره تعریف کردن توابع با اسم یکسان به صورتی که تایپ ورودی ها و یا تعداد ان ها و یا تایپ خروجی یکی نباشد

تعریف تایپ خروجی یک تابع به صورتی که در تایپ های موجود نباشد

و ساب تایپ نبودن تایپ ابجکتی که یک تابع برمیگرداند از تایپ خروجی ان

سپس یک مپ از هر کلاس به توابع ان میسازیم

و برای اینکه هر کلاس را بررسی کنیم اتریبیوت هایی که در این کلاس و کلاس های پدر هست را در یک سیمبول تیبل ذخیره میکنیم که برای مثل استک عمل میکند سپس گذر دوم را شروع میکنیم

استتیک تایپ ها را با یک تابع بازگشتی به دست میاوریم که هرقسمت را به تعدادی بخش تقسیم میکنیم که تایپ ان ها را بدست اوریم و با ترکیب آن ها میتوانیم تایپ آن را بدست آوریم برای مثال

Symbol eq\_class::type\_checking()

{

Symbol type1 = this->e1->type\_checking();

Symbol type2 = this->e2->type\_checking();

if (not (type1 == Int || type1 == Bool || type1 == Str ) or not(type2 == Int || type2 == Bool || type2 == Str) or type1 != type2)

{

semant\_error(this) << "Illegal comparison with a basic type.\n";

type = Object;

return type;

}

type = Bool;

return type;

}

مساوی از دو اکسپرژن تشکیل شده زمانی که تایپ هر کدام از ان ها را به دست اوریم اگر تایپ ان ها یکی باشد و اینت یا استرینگ یا بولین باشند این درست است در غیر این صورت باید ارور بدهیم و برای جایگزین کردن تایپی که درست باشد و از انتقال ارور جلوگیری کند برای ان استتیک تایپ ابجکت را در نظر میگیریم

و برای مثال : دیسپچ

Symbol dispatch\_class::type\_checking()

{

type = this->expr->type\_checking();

if (type != SELF\_TYPE and classTable.find(type) == classTable.end())

{

semant\_error(this) << "Dispatch on undefined class " << type << ".\n";

type = Object;

return type;

}

method\_class\* method ;

bool exist = false;

std::vector<Class\_> parents = get\_parents(type);

for (int i = 0; i < parents.size(); i++)

{

for (int j = 0; j < methodTable[parents[i]].size(); j++)

{

if (methodTable[parents[i]][j]->getName() == this->name)

{

method = methodTable[parents[i]][j];

exist = true;

break;

}

}

if(exist)

break;

}

if (not exist)

{

semant\_error(this) << "Dispatch to undefined method " << this->name << ".\n";

type = Object;

return type;

}

Formals formals = method->getFormals();

int k1 = this->actual->first(), k2 = formals->first();

bool error = false;

while (this->actual->more(k1) && formals->more(k2))

{

Symbol actual\_type = this->actual->nth(k1)->type\_checking();

Symbol formal\_type = formals->nth(k2)->getType();

if (not conform(actual\_type,formal\_type))

{

error = true;

semant\_error(this) << "In call of method " << this->name << ", type " << actual\_type << " of parameter " << formals->nth(k2)->getName() << " does not conform to declared type " << formal\_type << ".\n";

}

k1 = this->actual->next(k1);

k2 = formals->next(k2);

if (this->actual->more(k1) xor formals->more(k2))

{

error = true;

semant\_error(this) << "Method " << this->name << " called with wrong number of arguments.\n";

break;

}

}

if (error)

{

type = Object;

return type;

}

if (method->getReturnType() != SELF\_TYPE)

type = method->getReturnType();

return type;

}

در ابتدا تایپ اکسپرژن را برسی میکنیم سپس برسی میکنیم که ایا تابعی که فراخوانی شده توسط ان اکسپرژن در کلاس ان یا کلاس پدر و جد های ان هست یا خیر و اگر نبود ارور میدهیم و اگر بود بررسی میکنیم که ایا تعداد ورودی های داده شده و تایپ ان ها با تعداد ورودی های تابع و تایپ ان ها یکی است یا خیر و اگر نبود ارود میدهیم و در غیر اینصورت اگر ریترن تایت تابع سلف تایپ نبود استتیک تایپ را ریترن تایپ و در غیر این صورت تایپ اکسپرژن در نظر میگیریم

در همه مراحل اگر با ارور مواجه شویم استتیک تایپ را ابجکت در نظر میگیریم