Tópicos Avançados em Engenharia de Software – Linguagens e Compiladores

Compilador NOPL para Framework C#

Paulo Bertoldi Renaux

Metodologia

1. Pesquisa sobre a estrutura
   - Leitura de código similar
   - Leitura de arquivos .h e .cpp das estruturas do grafo

Metodologia

2. Implementação

- Adaptação de trechos similares
- Desenvolvimento de funções para obter o termo correspondente a um valor de um atributo em uma estrutura (exemplo ao lado)
- Comparação com exemplo em Framework C#

```
std::string FormatName(Conjunction* conjunction)
{
    if (!conjunction)
        return "Condition.SINGLE";

    std::string name = "";

    switch (conjunction->getConjunctionId())
    {
        case 1: { // Conjunction::AND_CONJUNCTION
            name = "Condition.CONJUNCTION";
        } break;

        case 2: { //Conjunction::OR_CONJUNCTION
            name = "Condition.DISJUNCTION";
        } break;

        default: {
            name = "Condition.SINGLE";
        } break;
    }
```

Metodologia

4. Melhorias feitas

- Leitura de código fonte
  - Separação de conceitos & DRY
- Identificação dos passos

```
std::string FormatName(Type* type);
std::string FormatName(Factor* factor);
std::string FormatName(Symbol* symbol);
std::string FormatName(Conjunction* conjunction);
std::string FormatName (Strategy* strategy);
std::string FormatValue(Attribute* attribute);
std::string FormatValue(Factor* factor);
std::string Identation (int amount);
std::string generateCodeNestedPremise(Instance* instance, Premise* premise);
std::string ExternalInstanceFormat (std::string discard, std::string preAppend, std::string source);
std::string getStrategy(Instance* instance);
std::string ExternalInstanceFormat (std::string discard, std::string preAppend, std::string source);
```

```
filestream << Identation(level) << "#region instance declarations" << std::endl;
for (std::map<std::string, Instance*>::iterator it = instances->begin(); it != instances->end(); ++it)
{…
}
filestream << Identation(level) << "#endregion" << std::endl;
```

```
filestream << Identation(level) << "#region attribute declarations" << std::endl;
for (std::map<std::string, Attribute*>::iterator it = attributes->begin(); it != attributes->end(); ++it)
{…
}
filestream << Identation(level) << "#endregion" << std::endl;
```

```
filestream << Identation(level) << "#region method pointer declarations" << std::endl;
for (std::map<std::string, Method*>::iterator it = methods->begin(); it != methods->end(); it++)
{…
}
filestream << Identation(level) << "#endregion" << std::endl;
```

Metodologia

- Identificação dos passos

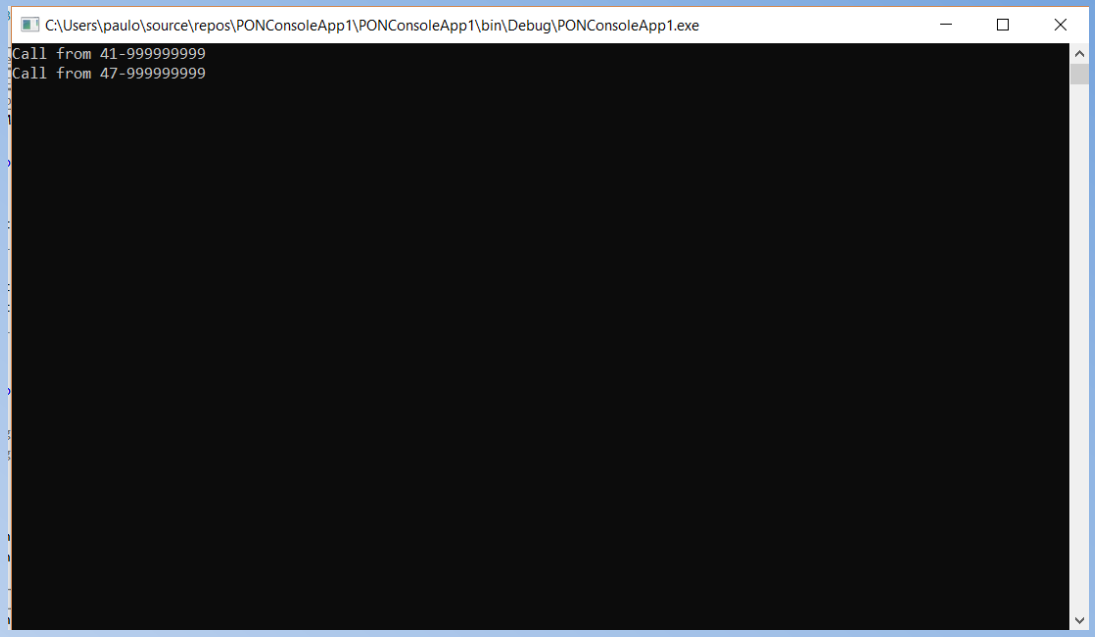
```
class MainFBE : NOPApplication
{
    instance declarations
    attribute declarations
    method pointer declarations
    public MainFBE() : base (SchedulerStrategy.PRIORITY) { startApplication(); }

    public override void initFactBase()
    {
    method pointers
    instances
    }

    public override void initRules()
    {
        rules from instance this:

        rules from instance sectorA:

        rules from instance sectorB:

    }
    public override void codeApplication() { }
    public override void initSharedPremises() { }
    public override void configureStartApplicationAction() { }
    methods accessed by MethodPointer's
}
```

```
#region rules from instance sectorA:
rlFireAlarmA

rlFireAlarmB

#endregion
```

```
var cdsectorA_rlFireAlarmA = new Condition (Condition.CONJUNCTION);
var acsectorA_rlFireAlarmA = new Action();

var cdsectorA_rlFireAlarmA_0 = new SubCondition (Condition.CONJUNCTION, exclusive: false);
cdsectorA_rlFireAlarmA_0.addPremise(new Premise(sectorA.alarmA.atStatus, NBoolean.TRUE_NOP, Premise.EQUAL, exclusive: false));
cdsectorA_rlFireAlarmA_0.addPremise(new Premise(sectorA.atIntruderDetected, NBoolean.FALSE_NOP, Premise.EQUAL, exclusive: false));
cdsectorA_rlFireAlarmA.addSubCondition(cdsectorA_rlFireAlarmA_0);
var cdsectorA_rlFireAlarmA_1 = new SubCondition (Condition.DISJUNCTION, exclusive: false);
cdsectorA_rlFireAlarmA_1.addPremise(new Premise(sectorA.sensorA1.atState, NBoolean.TRUE_NOP, Premise.EQUAL, exclusive: false));
cdsectorA_rlFireAlarmA_1.addPremise(new Premise(sectorA.sensorA2.atState, NBoolean.TRUE_NOP, Premise.EQUAL, exclusive: false));
cdsectorA_rlFireAlarmA.addSubCondition(cdsectorA_rlFireAlarmA_1);
acsectorA_rlFireAlarmA.addInstigation(new Instigation(sectorA.sirenA1.mpmtFire, new List<Attribute>(){new NInteger(10)}));
acsectorA_rlFireAlarmA.addInstigation(new Instigation(sectorA.sirenA2.mpmtFire, new List<Attribute>(){new NInteger(30)}));
acsectorA_rlFireAlarmA.addInstigation(new Instigation(sectorA.mpmtNotifyInvasion)); // No arguments
var sectorA_rlFireAlarmA = new Rule ("", scheduler, cdsectorA_rlFireAlarmA, acsectorA_rlFireAlarmA, IsDerived:false);
```

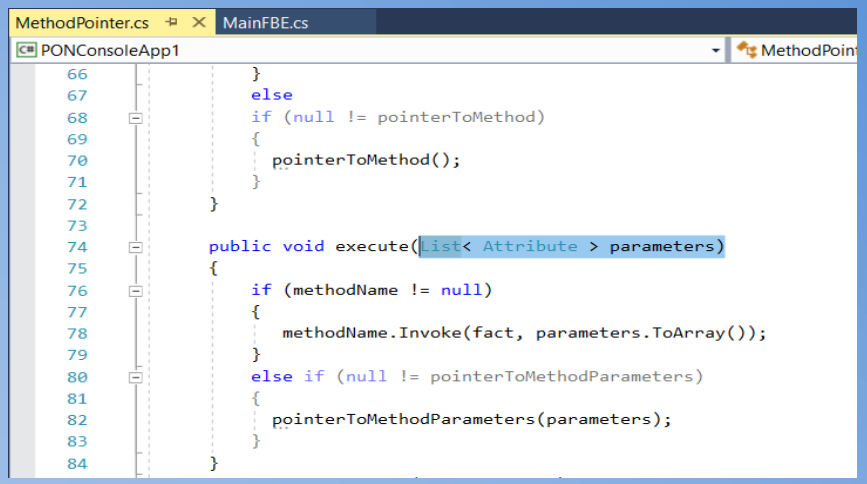
Resultados

- Compilação OK
- Funcionamento OK



Resultados (correções necessárias)

- Chamada de funções com parâmetros – OK



- Necessário uso correto (tipo e quantidade) de parâmetros ao atribuir MethodPointer à instigação

Resultados (correções necessárias)

- Estratégia do escalonador

```
//
// main:  constructor and initFactBase start
// other: constructor start
//
if (isMainFBE)
{
    // Framework C# Scheduler in construction, most reliable is SchedulerStrategy.NO_ONE
    filestream << Identation(level) << "public "<< mainFBE << "() : base (SchedulerStrategy.NO_ONE) { startApplication(); }" << std::endl << std::endl;
    filestream << Identation(level) << "public override void initFactBase()" << std::endl << Identation(level) <<  "{" << std::endl;
}
else
{...
}
```

- Acesso à propriedade Strategy

```
std::string getStrategy(Instance* instance)
{
    std::string strategy = "";

    /* New function added to compiler
     * Update on Fbe.h:
     *      std::map<std::string, Property*>* getProperties();
     *
     * Update on Call.cpp:
     *      std::map<std::string, Property*>* Fbe::getProperties() { return this->properties; }
     */
    std::map<std::string, Property*> *propertiesMap = (instance->getFbe())->getProperties();
    for (std::map<std::string, Property*>::iterator propIt = propertiesMap->begin(); propIt != propertiesMap->end(); propIt++)
    {
        if (propIt->second->getType() == Property::STRATEGY_PROPERTY)
        {
            strategy = FormatName((Strategy *) propIt->second);
        }
    }

    if (strategy.empty())
    {
        strategy = "SchedulerStrategy.NO_ONE";
    }

    return strategy;
}
```