EasyChair Preprint

№ 6189

An Object-Oriented Language Based on C++ :Leelus Typed Language.

Frank Appiah

July 30, 2021

An Object-Oriented Language Based on C++ :Leelus Typed Language.

Frank Appiah[1]

Abstract. This teaches typed language theory for interpretable artificial intelligence method for Multi-agent perspective. This is based on Object-Oriented C++ language. In describing, readers will be looking at code constructs as object classes. There are only three variables in the language argumentation. The argumentatives includes action, location and temporal with a variant types still including three arguments- rank, interest 1 and interest 2.

Keywords: language, object, typed theory, typed language,.

1 Introduction.

This section describes the content of this letter in terms of aims, objectives, method and planned outputs. There is brief discussion on each main term. I will describe the structural code. There are 5 type definitions in the Leetype header(h). A type definition is made with the declaration:

typedef datatype var_name;

In Leelus type language, each data type is declared as a character pointer to represent an array of characters. The type definitions are as follows in Leelus:

```
typedef char* action;
typedef char* temporal;
typedef char* interest;
typedef char* location;
typedef char* rank;
```

There are 8 header structures in the Leelus Typed Language namely:

[1] Frank Appiah is with King's College London, Strand, London, England, UK. Department of Informatics.

- Leetype.h
- Enactage.h
- EnactEL.h
- Enact.h
- Interestact.h

- Locationact.h
- Temporalrank.h
- IsImplies.h

Each header is implemented in its class structure. The class structures are namely:

- Leetype.c
- Enactage.c
- EnactEL.c
- Enact.c
- Interestact.c
- Locationact.c
- Temporalrank.c
- IsImplies.c

The accessors of the classes are the getters and setters. Each class construct is overloaded in addition to the default constructor of itself. An exact Enact class has a constructor as follows:

Enact(interest instr1, interest instr2, rank r) ;

The Enact class constructor has three arguments in passing values of two interest and rank value. A possible value for interest are some kind alternative words like buy or sell. An agent is interested in buy or sell in economics or business or commerce perspective. Here, to make a buy or sell decision needs the entity to rank in the 9th position to do so. Without the possibility of this decision making, there will be a negotiation on reputation or intention to things to happen. With these correct passing values, then it starts to reason about his intention by model transformation on code of practice.
Another class that needs to be discussed is Enactage with constructor:

Enactage(action a, location l, temporal t) ;

An Enactage is a construction of action, location and temporal values. If Enact of interests are made by a ranking entity then what will the enact age be? This question is about the exact action in some location that will temporarily hold for sometime. So Enactage construct

is an object structure to assign the three variables of variant values in the parameter passing.

Enact **implies** Enactage :Logical mode

The declaration is imperative. With variable assignments, the variables are assigned values. These values are passes to the constructors. There is then an interpretation of intelligence to reason of stored decisions.
The last constructor of importance is enactEL class constructor. It takes two arguments of Enactage and Enact:

enactEL(Enactage E, Enact L) ;

EnactEL can be a representation of enact expression language in generalized proposition of legality.

1.1 Aims.

– This research project aims to undertake an intelligent autonomous agent from the perspective of procurement world. Then reason with its logically basics to create an autonomous agent.

1.2 Objectives.

– The main objective is to use a gender -alized enact functions that are program -med as logical functions with an enact program.This is executed to run the enact parameters against the enact facts described as engagement functions of enact.
The behavior relationships of chain of commands are represented mathema -tically to create behavior dominance of agent interest. This creates autonomous objects.

1.3 Methodology.

– An abstract view of agent is used in representing the agent environment of procurement enact function. Mathematical logic tools are used to describe the procurement agent view. There is also the need to semantically compute the description of logic. This is made with Object-Oriented Leelus typed language. The logical structure is considered to be class structures.

1.4 Planned Outputs

– A console application written in C++ called LEEMapper. A report on demons -tration of

execution run on enact programs is not exploited here. The typed language is used to construct the application itself.

2 Conclusion

This report is about Leelus Typed Language. In this report, I show the header structures of Leelus Typed Language about 8 of them. The details of which are in the appendix section.

Further Reading

1. Josuttis, N. M. (2012). The C++ standard library: a tutorial and reference.
2. Coplien, J. O. (1991). Advanced C++ programming styles and idioms. Addison-Wesley Longman Publishing Co., Inc..
3. Stroustrup, B. (2000). The C++ programming language. Pearson Education India.
4. Horowitz, E., Sahni, S., & Rajasekaran, S. (1997). Computer algorithms C++: C++ and pseudocode versions. Macmillan.
5. Eckel, B. (2021). Thinking in C++.

Appendix : Leelus Header Code.

```
/ File:   LEEType.h
// Author: appiah
//
// Created on August,2020 00:3


#ifndef _LEETYPE_H
#define    _LEETYPE_H

typedef char* action;
typedef char* location;
```

```
//
// File:   Enact.h
// Author: appiah
//
//
#include "LEEType.h"


#ifndef _ENACTAGE_H
#define    _ENACTAGE_H

//using namespace std;
```

```
typedef char* temporal;
typedef char* rank;
typedef char* interest;

#endif    /* _LEETYPE_H */
```

```
class Enactage{
public:
    Enactage();
    Enactage(action a, location l, temporal t);
    void setAction(action a);
    void setLocation(location l);
    void setTemporal(temporal t);
    action getAction();
    location getLocation();
    temporal getTemporal();
private:
    action act;
    location loc;
    temporal temp;
};

#endif    /* _ENACTAGE_H */
```

```
// File:   enactEL.h
// Author: appiah
//
#include "LEEType.h"
#include "Enactage.h"
#include "Enact.h"

#ifndef _ENACTEL_H
#define    _ENACTEL_H


class enactEL{
public:
    enactEL();
    enactEL(Enactage E, Enact L);
    Enactage getEnactage();
    Enact getEnactmentL();
private :
    Enactage E;
    Enact L;
};

#endif    /* _ENACTEL_H */
```

```
// File:   Enactage.h
// Author: appiah
//
//

#include "LEEType.h"
#ifndef _ENACT_H
#define    _ENACT_H

using namespace std;


class Enact{
public:
    Enact();
    Enact(interest inrs1, interest inrs2);
    Enact(interest inrs1, interest inrs2, rank
r);
    void setInterests(interest inrs1, interest
inrs2);
    void setRank(rank r);
    rank getRank();
    interest* getInterests();
private:
    rank R;
    interest IRS[2];
    interest IT1;
    interest IT2;
};

#endif    /* _ENACT_H */
```

```
//
// File:   interestact.h
// Author: appiah
//
//

#include "LEEType.h"
#include "Enactage.h"
#include "Enact.h"

#ifndef _INTERESTACT_H
#define    _INTERESTACT_H

class interestact{
public:
    interestact();
    interestact(Enactage ia, Enact en);
protected:
    Enactage I;
    Enact A;
};



#endif    /* _INTERESTACT_H */
```

```
//
// File:   locationact.h
// Author: appiah
//
//
#include "Enactage.h"

#ifndef _LOCATIONACT_H
#define    _LOCATIONACT_H

class locationact{
public:
    locationact();
    locationact(Enactage locact);
    Enactage getEngagement();
private:
    Enactage locAct;
};

#endif    /* _LOCATIONACT_H */
```

```
// File:   ImplieStruct.h
// Author: appiah
//
//

#include "Enact.h"
#include "Enactage.h"


#ifndef _TEMPORALRANK_H
#define    _TEMPORALRANK_H



class temporalrank{
public:
    temporalrank();
    temporalrank(Enact ee, Enactage eage);
    Enact getEnactment();
    Enactage getEngagement();
    char* toString();
protected:
    Enact T;
    Enactage R;
};
```

```
// File:   IsImplies.h
// Author: appiah
//
//

#include "LEEType.h"
#include "temporalrank.h"
#include "Enact.h"
#include "enactEL.h"
#include "locationact.h"
#include "Enactage.h"
#include "interestact.h"

//using namespace lee::ture;

#ifndef _ISIMPLIES_H
#define    _ISIMPLIES_H


class IsImplies{
public:
    IsImplies();
    void setEnact1(Enactage E, Enact L);
    void setInterestAct1(Enactage I, action
A);
```

```
#endif    /* _TEMPORALRANK_H */
```

```
    void setLocationAct1(location Loc, action
Ac);
    void setTemporalRank1(temporal T, rank
R);
    enactEL getEnact();
    locationact getLocationAct();
    temporalrank getTemporalRank();
    interestact getInterestAct();
    void setEnact(enactEL EL);
    void setInterestAct(interestact IA);
    void setLocationAct(locationact LA);
    void setTemporalRank(temporalrank TR);
private:
    temporalrank TR;
    locationact LA;
    enactEL EL;
    interestact IA;
    Enactage E;
    Enact L;
    rank R;
    action A;
    Enactage I;
    location Loc;
    temporal T;
};
#endif    /* _ISIMPLIES_H */
```