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

Prof Frank Appiah AKA FAEng PhD
King's College
London
England.

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,.

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:

- 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 acessors 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.

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.

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.

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.

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.

Computer Codes

Typographical Conventions for C++

C++:Normal Text	**C++:Keyword**	**C++:Extensions**	C++:Data Type
C++:Decimal	C++:Octal	C++:Hex	C++:Float
C++:Char	C++:String	C++:String Char	C++:Comment
C++:Symbol	C++:Preprocessor	C++:Prep. Lib	
Doxygen:Normal Text	**Doxygen:Tags**	**Doxygen:Word**	**Doxygen:HTML Tag**
Doxygen:Description	Doxygen:Comment	Doxygen:Identifier	Doxygen:HTML Comment
Doxygen:Types	Alerts:Normal Text	**Alerts:Alert**	

```
SUPPLEMENTARY MATERIAL ON ARTICLE: MAPPING OF LEE ON ENACT AND ENGAGEMENT.

       C++ PROGRAMMING: LEELUS Type Language
==========================================
       HEADER STRUCTURES
==========================================
//
// File:   LEEType.h
// Author: appiah
//
// Created on August,2020 00:3


#ifndef _LEETYPE_H
#define _LEETYPE_H

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

#endif  /* _LEETYPE_H */


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


#ifndef _ENACTAGE_H
#define _ENACTAGE_H

//using namespace std;

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 */

```

```
 57 //
 58 // File:   enactEL.h
 59 // Author: appiah
 60 //
 61 #include "LEEType.h"
 62 #include "Enactage.h"
 63 #include "Enact.h"
 64
 65 #ifndef _ENACTEL_H
 66 #define _ENACTEL_H
 67
 68
 69 class enactEL{
 70 public:
 71     enactEL();
 72     enactEL(Enactage E, Enact L);
 73     Enactage getEnactage();
 74     Enact getEnactmentL();
 75 private :
 76     Enactage E;
 77     Enact L;
 78 };
 79
 80 #endif  /* _ENACTEL_H */
 81
 82
 83 //
 84 // File:   Enactage.h
 85 // Author: appiah
 86 //
 87 //
 88
 89 #include "LEEType.h"
 90 #ifndef _ENACT_H
 91 #define _ENACT_H
 92
 93 using namespace std;
 94
 95
 96 class Enact{
 97 public:
 98     Enact();
 99     Enact(interest inrs1, interest inrs2);
100     Enact(interest inrs1, interest inrs2, rank r);
101     void setInterests(interest inrs1, interest inrs2);
102     void setRank(rank r);
103     rank getRank();
104     interest* getInterests();
105 private:
106     rank R;
107     interest IRS[2];
108     interest IT1;
109     interest IT2;
110 };
111
112 #endif  /* _ENACT_H */
113
114 //
115 // File:   interestact.h
116 // Author: appiah
117 //
118 //
119
120 #include "LEEType.h"
121 #include "Enactage.h"
```

```
122 #include "Enact.h"
123
124 #ifndef _INTERESTACT_H
125 #define _INTERESTACT_H
126
127 class interestact{
128 public:
129     interestact();
130     interestact(Enactage ia, Enact en);
131 protected:
132     Enactage I;
133     Enact A;
134 };
135
136
137
138 #endif  /* _INTERESTACT_H */
139
140 //
141 // File:   locationact.h
142 // Author: appiah
143 //
144 //
145 #include "Enactage.h"
146
147 #ifndef _LOCATIONACT_H
148 #define _LOCATIONACT_H
149
150 class locationact{
151 public:
152     locationact();
153     locationact(Enactage locact);
154     Enactage getEngagement();
155 private:
156     Enactage locAct;
157 };
158
159 #endif  /* _LOCATIONACT_H */
160
161
162
163 //
164 // File:   ImplieStruct.h
165 // Author: appiah
166 //
167 //
168
169 #include "Enact.h"
170 #include "Enactage.h"
171
172
173 #ifndef _TEMPORALRANK_H
174 #define _TEMPORALRANK_H
175
176
177
178 class temporalrank{
179 public:
180     temporalrank();
181     temporalrank(Enact ee, Enactage eage);
182     Enact getEnactment();
183     Enactage getEngagement();
184     char* toString();
185 protected:
186     Enact T;
```

```
187     Enactage R;
188 };
189 #endif  /* _TEMPORALRANK_H */
190 
191 
192 //
193 // File:   IsImplies.h
194 // Author: appiah
195 //
196 //
197 
198 #include "LEEType.h"
199 #include "temporalrank.h"
200 #include "Enact.h"
201 #include "enactEL.h"
202 #include "locationact.h"
203 #include "Enactage.h"
204 #include "interestact.h"
205 
206 //using namespace lee::ture;
207 
208 #ifndef _ISIMPLIES_H
209 #define _ISIMPLIES_H
210 
211 
212 class IsImplies{
213 public:
214     IsImplies();
215     void setEnact1(Enactage E, Enact L);
216     void setInterestAct1(Enactage I, action A);
217     void setLocationAct1(location Loc, action Ac);
218     void setTemporalRank1(temporal T, rank R);
219     enactEL getEnact();
220     locationact getLocationAct();
221     temporalrank getTemporalRank();
222     interestact getInterestAct();
223     void setEnact(enactEL EL);
224     void setInterestAct(interestact IA);
225     void setLocationAct(locationact LA);
226     void setTemporalRank(temporalrank TR);
227 private:
228     temporalrank TR;
229     locationact LA;
230     enactEL EL;
231     interestact IA;
232     Enactage E;
233     Enact L;
234     rank R;
235     action A;
236     Enactage I;
237     location Loc;
238     temporal T;
239 };
240 #endif  /* _ISIMPLIES_H */
241 
242 
243 
244 ========================================
245              C++ CLASSES
246   SOURCE CODES ON LEE IMPLEMENTATION.
247 ========================================
248 
249 
250 #include "Enact.h"
251 
```

```
252 Enact::Enact(){
253
254 }
255
256 Enact::Enact(interest inr1, interest inr2):
257 IT1(inr1),
258 IT2(inr2){
259
260 }
261
262 Enact::Enact(interest inr3, interest inr4, rank r){
263     IT1=inr3;
264     IT2=inr4;
265     R=r;
266 }
267
268 interest* Enact::getInterests(){
269     IRS[0]=IT1;
270     IRS[1]=IT2;
271
272     return IRS;
273 }
274
275 rank Enact::getRank(){
276     return R;
277 }
278
279 void Enact::setInterests(interest i1, interest i2){
280     IT1=i1;
281     IT2=i2;
282 }
283
284 void Enact::setRank(rank r)
285 {
286     R=r;
287 }
288
289
290
291 #include "enactEL.h"
292
293 enactEL::enactEL(){
294
295 }
296
297 Enactage enactEL::getEnactage(){
298     return E;
299 }
300
301 Enact enactEL::getEnactmentL(){
302     return L;
303 }
304
305 enactEL::enactEL(Enactage e, Enact l) {
306     L.setInterests(l.getInterests()[0], l.getInterests()[1]);
307     L.setRank(l.getRank());
308     E.setAction(e.getAction());
309     E.setLocation(e.getLocation());
310     E.setTemporal(e.getTemporal());
311 }
312
313
314 #include "Enactage.h"
315
316 Enactage::Enactage(){
```

```
317
318 }
319
320 Enactage::Enactage(action a, location l, temporal t):
321 act(a),
322 loc(l),
323 temp(t){
324
325 }
326 action Enactage::getAction(){
327     return act;
328 }
329
330 temporal Enactage::getTemporal(){
331     return temp;
332 }
333
334 location Enactage::getLocation(){
335     return loc;
336 }
337
338 void Enactage::setAction(action a){
339     act=a;
340 }
341
342 void Enactage::setLocation(location l){
343     loc=l;
344 }
345
346 void Enactage::setTemporal(temporal t){
347     temp=t;
348 }
349
350 #include "interestact.h"
351
352 interestact::interestact(){
353 //    A=new Enact();
354  //   I=new Enactage();
355 }
356
357 interestact::interestact(Enactage ia, Enact en):
358 A(en),
359 I(ia){
360
361 }
362
363 #include "IsImplies.h"
364 #include "interestact.h"
365 #include "temporalrank.h"
366 #include "enactEL.h"
367 #include "Enact.h"
368 #include "Enactage.h"
369
370 IsImplies::IsImplies(){
371 }
372
373 enactEL IsImplies::getEnact(){
374     return EL;
375 }
376
377 interestact IsImplies::getInterestAct(){
378     return IA;
379 }
380
381 locationact IsImplies::getLocationAct(){
```

```
382     return LA;
383 }
384
385 temporalrank IsImplies::getTemporalRank(){
386     return TR;
387 }
388
389 void IsImplies::setEnact(enactEL el){
390     EL=el;
391 }
392
393 void IsImplies::setEnact1(Enactage e, Enact l){
394     E.setAction(e.getAction());
395     E.setLocation(e.getLocation());
396     E.setTemporal(e.getTemporal());
397     L.setInterests(l.getInterests()[0],l.getInterests()[1]);
398     L.setRank(l.getRank());
399 }
400
401 void IsImplies::setInterestAct(interestact ia){
402     IA=ia;
403 }
404
405 void IsImplies::setInterestAct1(Enactage i, action a){
406     char* ir=i.getAction();
407     char* lr1=i.getLocation();
408     I.setAction(ir);
409     I.setLocation(lr1);
410     I.setAction(a);
411     A=a;
412 }
413
414 void IsImplies::setLocationAct1(location loc, action A){
415     Loc=loc;
416     A=A;
417 }
418
419 void IsImplies::setTemporalRank1(temporal T, rank R){
420     T=T;
421     R=R;
422 }
423
424 void IsImplies::setTemporalRank(temporalrank tr){
425     TR=tr;
426 }
427
428
429 #include "locationact.h"
430
431 locationact::locationact(){
432
433 }
434
435 locationact::locationact(Enactage la){
436      locAct=la;
437 }
438
439 Enactage locationact::getEngagement(){
440     return locAct;
441 }
442
443
444
445 #include "temporalrank.h"
446
```

```
447 temporalrank::temporalrank(){
448
449 }
450
451 temporalrank::temporalrank(Enact ee, Enactage age){
452     R=age;
453     T=ee;
454 }
455
456 Enact temporalrank::getEnactment(){
457     return T;
458 }
459
460 Enactage temporalrank::getEngagement(){
461     return R;
462 }
463
464 char* temporalrank::toString(){
465     return "rank_i---> t";
466 }
467
```