Article title: Building the Enactment Engagement Language Stored Program :Leemapper

Authors: Frank Appiah[1]

Affiliations: King's College London, Department of Informatics[1]

Orcid ids: 0000-0001-6732-9503[1]

Contact e-mail: appiahnsiahfrank@gmail.com

License information: This work has been published open access under Creative Commons Attribution License <http://creativecommons.org/licenses/by/4.0/>, which permits unrestricted use, distribution, and reproduction in any medium, provided the original work is properly cited. Conditions, terms of use and publishing policy can be found at <https://www.scienceopen.com/>.

Preprint statement: This article is a preprint and has not been peer-reviewed, under consideration and submitted to ScienceOpen Preprints for open peer review.

DOI: 10.14293/S2199-1006.1.SOR-.PPCADSN.v1

Preprint first posted online: 02 August 2021

Building the Enactment Engagement Language Stored Program: Leemapper.

Frank Appiah ¹

Abstract. This is about the necessary requirements needed to build an application of Lee class structures. The requirements look at the parameters needed to have a functional logic system. This is on Leemapper application and details of execution as outputs are also shown.

Keywords. *requirements, variables, definitions, functional, application.*

1 Introduction

Leelus[6] is a typed language theory for interpretable artificial intelligence method for multi-agent perspective. This is based on the Object-Oriented C++[1,2,3,4,5] language. In describing, readers will be looking at application constructs as instance classes. There are only three variables in the language argumentation. The argumentatives includes action, location and temporal with a variant type still including three arguments- rank, interest 1 and interest 2.

The action, location, rank and two interests are of character type dimensioned in array. It is programmed as a pointer definition. I will now described programmed actions with four values in namespace:

```
action a="buy", a1="sell", a2="sold", a3="trade";
```

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

The first programmed action declared as a is an equalisation of "buy" action. This action is both a business and economics term. A buy action is used to describe a process of buying a good or service or both (good and service). The second programmed action is declared as a1 and it is an equalisation of "sell" action. Sell action is used to describe a process of selling a good or service or both. The third programmed action value is sold and it is declared as a2. It is used to indicate whether a good or service is sold or not. The possibility of both (good and service) can be also be evaluated to sold action. Sold is the past tense of sell. Sold places a past action on something that sell as goods or services. Sold goods are less current and likely of a future condition of never getting a sell action again. Sold can be reversed unless the item or good is returned. It will create an unsold action in protecting terms. Validity of return is a warranty on a sold good or service. Validity is specified in temporal means and I will attend to it later. Validation of warranty is required to be in the specified time between the sold date and warrant date. After the warrant date makes the validity of return invalid and guarantee notice removed from the period. This policy of warranty is a temporal nature of sold goods/services. Invalid return of goods create a programmed unsold action to change to "invalid_return" action. The final programmed action is an equalisation of trade value declared as a3 variable. The value of trade is normally pursued as tradepaper by the author's experience. Here, an author or writer trades a book made of Papers to an organisation to sell on their behalf. Mostly, on split percentage of buy value in currency specified during the agreed price.

The next set of programmed interests are described in terms of their value in the program. The code for declaration is as follows :

```
interest ia="sell_pc", ia1="buy_pc";
```

Two values of interest are used in the program namely : sell_pc or buy_pc.

This means that an agent has value if the entity sells pc or either buys pc. Buy is the anti-sell and vice versa.

On the business side, interest is a return value on investment denoted by a principal value period in some year(s) at a certain percentage limited to 100%. Again in such ways, what is the interest of an agent in business? This is a means to evaluate the interest of an action to another. What is interesting in doing such business with you? Agent asks an organisation. In this present situation, the equalisation of interest value, "sell_pc" is declared as ia. The value of interest is having to be used to describe an agent that has interest in selling PCs at their entity. With this interest, an agent believes to generate some return of investment in the entity just by selling personal computers(pc). The second interest value is "buy_pc". It is

declared as ia1. An agent has interest in buying a PC and will stay in business if the entity continues to do so.

An agent can do both buying and selling PCs. There are three terminologies in describing this. These are business, economics and commerce. I will leave you to find it out. The focuses are in E-business, E-agent and E-commerce. Doing business over electronic domain will place focus on E-business (Business-2-Business and more). EEE-CAB or E³-cab is the abbreviation used in describing all the focuses of an agent doing buying or selling and both selling and buying.

Businesses cannot stay relevant at all times. It is important to rank them to store the value of their rank in such a process. The value of rank for an agent to stay relevant to an organisation interested in delivering always or sometimes quality of service is required. Quality of service is required in doing business at the highest level and as such it is catered for by the rank value. The declaration of rank is coded as follows :

```
rank r="rank_9";
```

A single rank is declared in this program. It is declared as r. The rank is a numeric value but here the number is used after an underscore. A business which scores 9 is better off than another which scores 12. Hence, a lower rank value is better off in terms of doing quality of service than a higher rank entity. Given a table of ranks in a period of annual competitions :

Location is required to have any action done. Without location, directions can be made in any physical universe. A building in even a physical universe can give direction to a location of situational action.

We are looking at two locations in this coded program. It is programmed that declared loc is valued at Tottenham for an agent to do business. This is the first location value.

The code of location in stored program is as follows :

```
location loc="tottenham", loc1="London";
```

The second declared variable is loc1. It has a value of "London". This again situated action is located in London locality only except loc value which is Tottenham. Agent parties now do businesses in Tottenham and London only. But other local businesses can be located elsewhere. Possibly Accra Biz does business with New York Biz. Again, Kumasi Biz does business with London Biz and so on.

Doing business cannot be a permanent activity. Holding to do business with an agent in an agency is temporary in some sense. After a certain period of time limits an organisation in doing business with the agent company. That requirement demands a stored temporal value to hold business going until otherwise. On elapse, business stops to other matters until further agreement is signed. Contract by agreement between agent and entity is required to complete in a closed location before opening another business day. The expected commissions necessary on demand by both parties is of importance. The declared variable in temporal dimension is temp and it is coded as:

```
temporal temp="Sat 12/09/2019 12:00pm";
```

The temp value is Saturday 12/09/2019 12:00pm. The agent can begin to do a buying action or selling action before or after the declared day of doing business. Entity might cease to work if the declared temporal value is elapsed. Again might start to do business with agents after the elapsed. Always or sometimes or never calculi on temporal value might be a requirement on the specification of logic programs.

2 Setting Variables in Coded Program

In order to run the program, it is required to set the variables above in section 1 to the object classes. The 6 classes are Enact, Enactage, Isimplies, enactEL, locationact, interestact and temporalrank. The methodized setLEEValues is algorithmitized as:

```
Enactage en1;
    en1.setAction(a);
    en1.setLocation(loc);
    en1.setTemporal(temp);

Enact en2;
    en2.setInterests(ia, ia1);
    en2.setRank(r);

Isimplies is;
    is.setEnact1(en1, en2);
```

```

enactEL EL(en1, en2);
    is.setEnact(EL);

Enactage EA(a, loc, temp);
    is.setInterestAct1(EA, a);

    is.setEnact1(EA, en2);

locationact LA(en1);
    is.setLocationAct1(loc, a);

temporalrank TR(en2, EA);
    is.setTemporalRank1(temp, r);

```

Algorithm on Application Building

1. Set the values of action, location and temporal in Enactage setters.
2. The interests and rank values of Enact class are set.
3. Enactage class, en1 and Enact, en2 class instances are set to the Isimplies instance, is.
4. An instance of enactEL is constructed with enact, en1 and Enactage, en2. EL instance is set to the Isimplies, is.
5. An instance of Enactage, EA is constructed and set as interestact with a to the Isimplies instance, is. EA and en2 are set as enact 1.
6. LA is an instance of locationact and it is constructed with en1 instance. Then set as locationact 1 on Isimplies instance, is.
7. An instance of temporalrank, TR is constructed with en2 and EA instances. Then set as temporal rank on Isimplies, is instance. Finally the main method is called.

In main method;

```

setLEEEValues();
    enterContinues();
setContextLanguage();
    enterContinues();
setLEEImplies();
    enterContinues();
setLEEImpliesLang();
    enterContinues();
setLEELinearOrd();
    enterContinues();
setLEEAbb();
    enterContinues();
setLEEStructLang();
    enterContinues();
setLEEAbbLanguage();
    enterContinues();

```

```

        setLEEDoubleArrow();
        enterContinues();
setLEEDoubleArrowStruct();
        enterContinues();
        setIsLanguage();
        enterContinues();
setLEEConsequentRun();
        enterContinues();
        setLEEComposite();
        enterContinues();

return (EXIT_SUCCESS);

```

After the setting of LEE values by setLEEValues (), 12 logic functional methods are called in the main method as well.

The details of logic function and function call name or method name are enumerated.

logic function << function callname/method name

```

(1):: Context Language"<<setContextLanguage();
(2):: LEE Implications"<<setLEEImplies();
(3):: LEE Imply Language"<<setLEEImpliesLang();
(4):: LEE Linear Ordering"<<setLEELinearOrd();
(5):: LEE Abbreviation(Abb)"<<setLEEAbb();
(6):: LEE AbbStruct Language"<<setLEEStructLang();
(7):: LEE Abbreviation Language"<<setLEEAbb();
(8):: LEE Double Arrow(DA)"<<setLEEDoubleArrow();
(9):: LEE DA Structures"<<setLEEDoubleArrowStruct();
(10):: LEE is_Language"<<setIsLanguage();
(11):: LEE Consequent Run"<<setLEEConsequentRun();
(12):: Enactment Formalizations"<<setLEEComposite();

```

3 Conclusion

This is about building an application by Lee classes. In this work, I did describe the variable declarations needed as requirements in successfully creating an application. The details of Lee values are thoroughly

discussed on its relevance in starting a functional logic program. This is a stored program in the sense that the program has all its functionality initially valued at default. No further changes to the program during execution unless it is changed and recompiled. The 12 logic program details are omitted for now but the exact functional name along the functional method are enumerated in simple format. The source code can be accessed at [9]. The screenshots of output are shown.

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++.
6. Frank Appiah (2021). An Object-Oriented Language Based on C++ :Leelus Typed Language. EasyChair Preprint no. 6189
7. Frank Appiah (2021). The Mapping of the Logical Structure, Enact to the Engagement Structure of Enact. EasyChair Preprint no. 6187
8. Frank Appiah (2021). Semantic Computation of the Propositional Model Composites in Enactment Logic. EasyChair Preprint no. 6186
9. Frank Appiah (2021). Leemapper accessed online at <https://github.com/fanhubgt/Leemapper>.

```

File Edit View Bookmarks Settings Help
~/home/appliah/netbeans-6.0/cnd1/bin
Enter number[go(1)/no(2)] to continue>>
Go Value:=#
(1)The exact of business implies the exact of machinery, paronymy, interest
(2)The exact of business interest implies the exact of business action or a
(3)The exact of business interest implies the exact of interest in a locat
(4)The exact rank of business interest implies the exact of the linear/temp
Enter number[go(1)/no(2)] to continue>>
Go Value:=#
(1) Exact_1_buap: tottenham Sat 12/09/2010 12:00pm implies Exact_1(sell_pc.bu
(2)buay implies sell_pc
(3)sell_pc implies tottenham
(4)sell_5 implies Sell 12/09/2010 12:00pm
Enter number[go(1)/no(2)] to continue>>
Go Value:=#
(1)The exact of business implies the exact of logic
(2)The exact of business interest is the implication of business action.
(3)The exact of business interest is in a location
(4)The exact of business rank is based on temporal dimensions
Enter number[go(1)/no(2)] to continue>>
Go Value:=#
(1) Exact_1_buap: tottenham Sat 12/09/2010 12:00pm implies Exact_1(sell_pc.bu
(2)Interest buy implies Action sell_pc
(3) Interest sell_pc implies Action In Location tottenham
(4) Exact interest rank 5 implies Indication in temporal dimension Sat 12/09/2010

```

```

File Edit View Windows Settings Help
- Some applications are disabled in this mode.
-

1. Run the following command to create the table:

mysql> CREATE TABLE IF NOT EXISTS `business` (
  `id` INT(11) UNSIGNED NOT NULL AUTO_INCREMENT,
  `name` VARCHAR(255) NOT NULL,
  `description` VARCHAR(255) NOT NULL,
  `location` VARCHAR(255) NOT NULL,
  `created_at` DATETIME NOT NULL,
  PRIMARY KEY (`id`),
  INDEX (`name`),
  INDEX (`description`),
  INDEX (`location`)
) ENGINE=InnoDB;

Enter mysql>USE business; to continue
to mysql>

mysql> INSERT INTO business (name, description, location, created_at) VALUES ('New York', 'The most of business entered in the last location of business in the world', 'New York', '2012-01-01 00:00:00');

Enter mysql>USE business; to continue
to mysql>

mysql> SELECT * FROM business;
+----+-----+-----+-----+
| id | name | description | location | created_at |
+----+-----+-----+-----+
| 1  | New York | The most of business entered in the last location of business in the world | New York | 2012-01-01 00:00:00 |
+----+-----+-----+-----+

Enter mysql>USE business; to continue
to mysql>

```