iJournals: International Journal of Software & Hardware Research in Engineering
ISSN:2347-4890, ijournals.in/ijshre
Volume 13 Issue 8, August 2025

Merging Connectionism and Logicism in Knowledge Representation

Author: Weihan Huang

Master of Computer Science Department, State University of New York, at Buffalo, U.S.A.

Master of Physics Department, National Hsing Hua University, Taiwan

Email:weihanh@yahoo.com.tw

DOI: 10.26821/IJSHRE.13.08.2025.130801

ABSTRACT

There are two main schools in knowledge representation: Connectionism and Logicism. As two independent formalisms, they are often put separately in knowledge representation. While Logicism expresses in symbolic formulas, Connectionism expresses in graphs or networks. The purpose of this paper is to investigate that it is possible we can merge the two formalisms, Connectionism and Logicism, into one formalism? Firstly, in this paper I will introduce Logicism representations. Examples are proposition logic, first order predicate logic, second order predicate logic. Next, I will introduce Connectionism. Examples of Connectionism are directed graph, semantic network, artificial neural network. And lastly, I will try to merge the two formalisms into one formalism according to three mathematical theorems.

Keywords: Connectionism, Knowledge Representation, Logicism, Merge

1. INTRODUCTION TO LOGICISM

1.1 Introduction to Proposition Logic [1]

A proposition is a symbol that takes value of true or false. For example, let proposition P represent "John is here", it can only take one value, either P is true, i.e. John is here; or P is false, i.e. John is not here. Similarly, we can let proposition Q represent "Mary is here". These two representations are just the abstract conceptualization.

Next let's introduce the proof system. It is mainly by Modus Ponens Rule: if we know A, and we know A->B, then we can infer that B. Then we can start to prove in an axiomatic system. Assume we have axioms

(1) P or Q

(2) not Q

Please note that (P or Q) is equivalent to (not Q -> P). So we can have

(3) not Q -> P

By (2) and (3) and Modus Ponens Rule, we can deduce that

(4) P

We have P, not Q.. i.e. We have "John is here" and "Mary is not here". firstly, like algebra, we firstly transform natural language sentences to propositions containing only one segment of letter, then we do "reasoning" to get the results P and not Q.

This is a brief introduction to proposition logic. Next I will introduce a more complicated and powerful logic: First Order Predicate Logic.

1.2 Introduction to First Order Predicate Logic [2]

The language of first order logic contains quantifiers, predicates, logic operators, and first order terms. Quantifiers are like For-all and exists. Predicates are like IsAncestor(X,Y), IsParent(X,Y). Logical operators are {And, Or, Not, ->}. First order terms contain variables, objects and functions. Objects are like John, Mary. Functions are like fatherOf(X). motherOf(Y).

The example of "Ancestor" can be written by the first order logic:

(5) For-all X, Y, IsParent(X,Y)->IsAncestor(X,Y)

(6) For-all X, Y, Z

(IsParent(X,Y)andIsAncestor(Y,Z))

->IsAncestor(X,Z)

(7) For-all X IsParent(father(X),X).

(8) For-all X IsParent(mother(X),X).

i.e. first order terms are inside predicates. And first order logic is by quantifying over first order terms. Higher order logic is by quantifying over predicates.

Assume we know that the father of John is Peter, and the mother of Peter is Mary. By (7) For-all X IsParent(father(X),X), we have

(9) IsParent(Peter,John).

Similarly, we have

(10) IsParent(Mary,Peter).

Therefore by (5) and (9), we have

(11) IsAncestor(Peter,John).

And by (10)(11) and (6), we have

(12) IsAncestor(Mary,John).

This is the brief introduction to first order predicate logic, the second case of Logicism formalism. Next, I will introduce second order predicate logic, i.e. the third case of Logicism formalism.

1.3 Introduction to Second Order Predicate Logic [3]

After seeing the first two cases of Logicism formalism, we are now arriving at the third case of Logicism formalism : the Second Order Predicate Logic.

Firstly, let's review the differences of pre-order predicate, in-order predicate and post-order predicate.

Pre-order Predicate :

Predicate argument1 argument2

In-order Predicate:

argument1 Predicate argument2

Post-order Predicate:

argument1 argument2 Predicate

The equality predicate = is actually an in-order predicate. For example:

"3=4" is false

"2+5=7" is true

Note that the equality predicate is reflexive, symmetric, and transitive.

(13) A=A (reflexive)

(14) A=B -> B=A (symmetric)

(15) (A=B and B=C) -> A=C (transitive)

And if A=B, then it is replicable with B for any argument A of any predicate.

(16) Forall A, Forall B, Forall Predicate (A=B) -> (Predicate(..,A,..) <--> Predicate(..,B,..)).

(17) Forall A, Forall B, Forall Predicate, Forall Function, (A=B) -> (Predicate (….Function(..,A,..),..) <--> (Predicate (…Function(..,B,..)…)).

Please heed that it is true for any predicate, so we have the "Forall Predicate". i.e. The Forall Quantifying is over all predicates, so in this case it is Second Order Predicate Logic. So here we reach the Second Order Predicate Logic, the third case of Logicism formalism. Next, we will go to the Connectionism formalisms.

2 INTRODUCTION TO CONNECTIONISM

2.1 Introduction to Directed Graph [4]

To represent the knowledge in the knowledge database uniformly, I am using a directed graph [5] as the knowledge representation.

Firstly, we need to distinguish two types of nodes "unique node" and "separate node". A "unique node" is always referred to the unique one node in all contexts. Examples of unique nodes are like "I", "You", "Weihan", "Little Prince", "Book of Natural Language Programming". And a "separate node" is a node that is not a unique node. Examples of "separate node" are "author", "reader", "name".

A node has links fanned out and links fanned in. And a link contains a string and one from-node and one to-node.

Example of "I am Weihan" is represented in the following graph:

"I" --be --> "Weihan"

<--be –

Example of "Book of Natural Language Programming" is

-own-> “name” -be-> "Natural Language Programming"

"Book <-belong to- <-be- Language Programming"

Natural Language-own->“author”-be->“Weihan”

<-belong to- <-be- Programming”

-own-> “reader” -be-> "Little Prince"

<-belong to- <-be-

We can read from the left to the right that "The author of Book Natural Language Programming is Weihan.” and read from right to left that “Weihan is the author of Book Natural Language Programming".

To represent the procedural semantics of sentence like "we wirte ChineseProgram", we can use a list to do it.

"How" -type-> "verb" -name-> "write" -type-> "we write ChineseProgram"

-procedure->

{decide the directory name;

decide the program nname;

decide the program cocodes;

edit the codes;

compile the codes;

run the codes;}

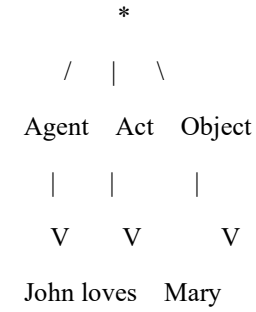
This is a brief introduction to Directed Graph. Next, I will introduce the Semantic Network.

2.2 Introduction to Semantic Network [6]

A representative model of semantic network is

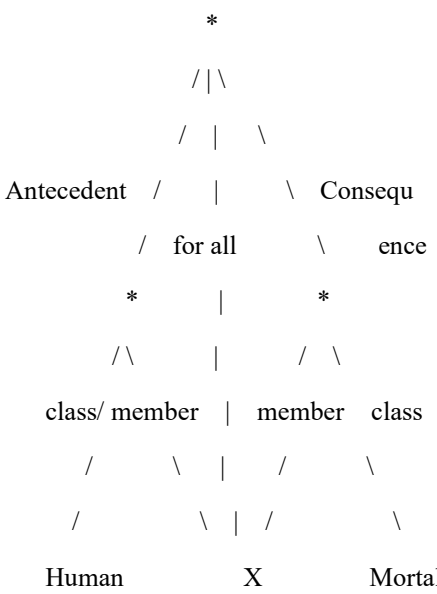
SNePS(Semantic Network Processing System) [6]. Its main purpose is to investigate knowledge representation and reasoning for natural language competent agent. One example in proposition logic that is expressed in graph is shown below.

Proposition Logic



First Order Predicate Logic

And another example in first order predicate logic that is expressed in graph is shown below.

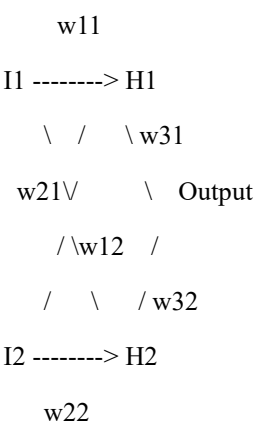


Since predicate logics can be expressed in graph here, it means that Logicisim can be merged into Connectionism by SNePS. This means that we can merge the two formalism Logicism and Connectionism into one formalism Connectionism by SNePS!

In the following I will introduce the third case of Connectionism: the artificial neural network.

2.3 Introduction to Artificial Neural Network[7]

A simple example of artificial neural network is the back propagation model for computing the logic function XOR. In this model, there are 3 layers of nodes. The first layer contains two input nodes I1 and I2. The second layer(the hidden layer) contains two nodes H1 and H2. The third layer contains one node Output, the result of XOR function of the two input A and B.



The forward computing is by the following formulas.

H1=f(I1*w11+I2*w12)

H2=f(I1*w21+I2*w22)

Output=f(H1*w31+H2*w32)

w11..w32 are weights of the link in the network, they are initially random values. They are changed by the comparison of computed Output and the standard solution of XOR Output in the training stage. The

standard solution of XOR output is given by the following truth table

I1	I2	XOR-Output
1	1	0
1	0	1
0	1	1
0	0	0

After the training stage, the weights should be such that the computed Output matches the standard solution of XOR Output.

This is the last example of Connectionism. Next I will show how to merge the 2 formalisms, Logicism and Connectionism, by 3 mathematic theorems.

3 THE MERGING THEOREMS

3.1 Merging theorem 1

Every logic sentence can be expressed as a tree[8].

Proof : We will prove it by construction of a tree from a logic sentence recursively. Here => means construction of the tree. N->T means T is a child of the node N.

A tree of (forAll X Formula) => node(forAll X)
-> A tree of(Formula)

A tree of (exists X Formula) =>
node(exists X) -> A tree of(Formula)

A tree of (not Formula) =>
node(not) -> A tree of(Formula)

A tree of (Formula1 and Formula2) =>
node(and) -> A tree of(Formula1)
-> A tree of(Formula2)

A tree of (Formula1 or Formula2) => node(or)
-> A tree of(Formula1)
-> A tree of(Formula2)

A tree of (Formula1 -> Formula2) =>
node(->) -> A tree of(Formula1)
-> A tree of(Formula2)

A tree of
(Predicate(arg1,arg2,..,argK)) =>
node(Predicate)-> A tree of(arg1)
->A tree of(arg2)
....
-> A tree of(argK)

A tree of
(Function(arg1,arg2,..,argK))=> node(Function) -> A tree of(arg1)
->A tree of(arg2)
....
-> A tree of(argK)

A tree of object => node(object)

A tree of Variable => node(Variable)

Q.E.D.

3.2 Merging theorem 2

Every directed graph can be expressed in first order logic.

Proof : We will prove it by construction of a directed graph in first order logic.

For each node I, there is a corresponding predicate IsVertex(node(I)).

For each directed link from node From to node To, there is a corresponding predicate

IsDirectedLink(

node(From), Value, node(To)).

where Value can be a string or number. It is a string for Semantic Network, and it is a number for Artificial Neural Network.

Q.E.D.

A predicate IsConnected(node(I),node(J)) is defined when node I,J are connected.

IsDirectedLink(node(I),Value,node(J)) -> IsConnected(node(I),node(J)).

IsDrecitedLink(node(I),Value,node(J))

and IsConnected(node(J),node(K)) -> IsConnected(node(I),node(K))

3.3 Merging theorem 3

Every logic tree can be expressed as a logic sentence.

Proof : We will prove it by construction of a logic sentence from a logic tree recursively. Here => means construction of the logic sentence. N->T means is T a child of the node N.

node(forAll X) -> A tree T =>

A sentence of (forAll X a sentence of(T))

node(exits X) -> A tree T =>

A sentence of (exists X a sentence of(T))

node(not X) -> A tree T =>

A sentence of (not a sentence of(T))

node(and) -> A tree T1

-> A tree T2 =>

A sentence of (a sentence of(T1) and a sentence of(T2))

node(or) -> A tree T1

-> A tree T2 =>

A sentence of (a sentence of(T1) or a sentence of(T2))

node(->) -> A tree T1

-> A tree T2 =>

A sentence of (a sentence of(T1) -> a sentence of(T2))

node(Predicate) -> A tree T1

-> A tree T2

……

-> A tree Tk =>

A sentence of (Predicate ((a sentence of T1), (a sentence of T2),

.. (a sentence of Tk)).

node(Function) -> A tree T1

-> A tree T2

……

-> A tree Tk =>

A sentence of (Function ((a sentence of T1), (a sentence of T2),

.. (a sentence of Tk)).

node(object) -> object

node(Variable) -> Variable

Q.E.D.

3.4 Discussion of the Merging of Connectionism and Logicism

By Merging theorem 1, every logic sentence can be expressed by a tree, because every tree is a directed graph, so we can have that every logic sentence can be expressed in a directed graph. This means that we can merge Logicism to Connectionism.

By merging theorem 2 and 3, every directed graph and logic tree can be expressed in first order logic. Therefore, we have merged Connectionism to

Logicism. Actually, by theorem 2 we can build the axioms in axiomatic system for those predicates of Vertexes and IsDirectedLinks. Then we can use it to reason about the connectivity of two nodes in graph by the definition of the predicate IsConnected(node(I),node(J)).

4 CONCLUSION

We have reviewed two well established formalisms : Connectionism and Logicism. The general pattern of Logicism is its symbolic formula with the ability of inference. And also for the school of Connectionism, the general pattern of Connectionism is graph and connectivity. All of these formalisms have the same features, firstly they all have uniform representations, and secondly they become powerful when the small segments are aggregated to be large size ones. However the two well-established formalisms Connectionism and Logicism are often put separately and independently in knowledge representation. So the purpose of this paper is to investigate that is it possible we can merge the two formalisms, Connectionism and Logicism, into one formalism? And I am glad that the final answer is positive by one Connectionism formalism SNePS, which can express predicate logics in graph. Hence SNePS merges the two formalism Connectionism and Logicism into one formalism Connectionism. Lastly, my first merging theorem is close to SNePS for it can also express predicate logics in graphs, and my second and third merging theorem can express graphs in logics. Therefore the three merging theorems proposed in this paper can merge the two formalisms Connectionism and Logicism into one formalism.

6. REFERENCES

[1] Proposition logic
https://en.wikipedia.org/wiki/Propositional_calculus
[2] First order predicate logic
https://en.wikipedia.org/wiki/First-order_logic
[3] Second order predicate logic
https://en.wikipedia.org/wiki/Second-order_logic
[4] Directed graph
Weihan Huang, 2025 "Natural Language Understanding by Natural Language Programming", International Journal of Software & Hardware Research in Engineering(IJSHRE) Volume 13, Issue 4 April 2025, pp.21
https://ijournals.in/wp-content/uploads/2025/04/3.IJSHRE-130401-Weihan.pdf
[5] Directed graph
https://en.wikipedia.org/wiki/Directed_graph
[6] Semantic network
https://en.wikipedia.org/wiki/SNePS
https://cse.buffalo.edu/sneps/
[7] Artificial neural network
https://en.wikipedia.org/wiki/Neural_network_(machine_learning)
[8] Tree
https://en.wikipedia.org/wiki/Tree_(graph_theory)