Abstraction Binding Logic:

A Logical Framework based off Set Theory, First & Second Order Logic, and the Lambda Calculus

Walter Gress

2022-10-8

## Basic Principles

Let : be the clarification operator

Let { } be a set

Let all the standard rules and operations of set theory stand as a formal foundation

The clarification operator is similar to the inference operator or assignment operator but used to provide a mapping of an old set to a new set.

A simple example follows:

Set A = { *element* 1, *element* 2, *element* 3 }

Set B : *some operator* A

For example, the first order logic operation “for all” ∀ when placed on a set A with the clarification operator directed towards set B allows set B to represent all the elements of set A as set B.

The “there exists operation” ∃ on set A with the clarification to set B provides a similar operation. Set B now represents the application of “there exists an element in set A” and maps that to the abstraction set B.

Via the clarification operation operations can be bound to sets and these new set representations or abstractions can be represented as a new set. According to this new formal language, there will be a variety of operators that can be bound to sets and abstracted using the clarification operator to provide operations of inference from first and second order logic and elements of the lambda calculus with the goal of building a formal contest-free grammar based off set theory. This context-free grammar will allow us to build tools based freely off contest to create architectures with markers for context.

What this paper will show is that via the addition of this simple clarification operator as an extension to set theory, this operator can be used to bind inference operations, lambda calculus functions, and much more such as second order logic operations and simple set theory operations already existent into a framework that allows for not only abstraction of concepts like set theory provides but continual, constructive re-abstraction.

## Abstraction Binding Logic and integrating First-Order Logic

The founding principles of First-Order Logic (Propositional Logic)

Consider the symbol P, Q, etc. in propositional logic to represent a proposition where a proposition is defined as a statement in propositional logic (a premise or a conclusion). (see citation). Premises are assumed to be true, or axiomatic. Using premises and propositional logic we can symbolically represent verbal proofs symbolically.

Propositional logic has the following operations that can be performed upon their respective propositions to come to a conclusion.

For example, given the premises P and Q,

P: It’s raining then it’s cloudy

Q: It’s cloudy

P → Q , P (insert symbol) Q

Other operators include the negation operator, which reverses the polarity of a premise

For example, ¬P

Also, the logical AND and OR operators which can be referenced here: (insert citation)

These operators are ∧∨. More can be found about their operations over truth operations over truth values here: (insert citation).

There are quite a few proofs that can be demonstrated through simple propositional logic, unfortunately its lack of variables limits us with regard to certain proofs so we will move on to first-order logic, an extension of propositional logic.

The principles of First Order Logic

**First-order logic**—also known as **predicate logic**, **quantificational logic**, and **first-order predicate calculus**—is a collection of [formal systems](https://en.wikipedia.org/wiki/Formal_system) used in [mathematics](https://en.wikipedia.org/wiki/Mathematics), [philosophy](https://en.wikipedia.org/wiki/Philosophy), [linguistics](https://en.wikipedia.org/wiki/Linguistics), and [computer science](https://en.wikipedia.org/wiki/Computer_science). First-order logic uses [quantified variables](https://en.wikipedia.org/wiki/Quantification_(logic)) over non-logical objects, and allows the use of sentences that contain variables, so that rather than propositions such as "Socrates is a man", one can have expressions in the form "there exists x such that x is Socrates and x is a man", where "there exists*"* is a quantifier, while *x* is a variable.[[1]](https://en.wikipedia.org/wiki/First-order_logic#cite_note-1) This distinguishes it from [propositional logic](https://en.wikipedia.org/wiki/Propositional_logic), which does not use quantifiers or [relations](https://en.wikipedia.org/wiki/Finitary_relation);[[2]](https://en.wikipedia.org/wiki/First-order_logic#cite_note-2) in this sense, propositional logic is the foundation of first-order logic. (citation wikipedia)

Set Binding and First Order Logic

A = { element 1, element 2, ... }

∀x, x = A, ∃y ∈x, ∃z ∈x, λ.x = y

Is simplified by the new operation into sets for nodes to operate on

A = { ? }

B : ∀xA The set B now maps to “for all the variables x on A”

C : ∃xA The set C now maps to “there exists a variable x on A”

D: λC The set D now maps to the lambda operation bound to the set C

And for example

E: λB The set E now maps to the lambdas operation bound to the set B

And set F then could be

F: D ∪ E

Each set on the left hand side represent a simple remapping of a unary or binary operation performed on the right hand side. .Thus each l.h.s. set represents an abstraction connection to one or more destination abstractions.

This starts with the empty set, and atomic sets, sets of one elements which are built by operation to deliver abstraction.

A : { element 1 }

B : { element 2 }  
C : A ∪ B

Language is broken down into its abstractions and logical relationships to other extractions

“Rob Snow mines Dragon Glass”

Requires an abstraction of not only each word and how it maps to every sub-element and related abstraction, but also a relationship that determines its logical meaning so it can provide context for other sentences.

Just the word Rob will create thousands of links to other abstractions, if not millions. The entire sentence is a set, an abstraction.

Take the set “Rob” for starters

A : { Rob }

B : { R }  
C : { o }  
D : { b }

E : { capitalized roman letters }

F : B ∈ E

G :

The sets provide abstraction, but the operations provide the multiple branches to further abstractions. The highest principle is to not create complicated operations within in a single abstraction, otherwise we expand our search space and make it unmanageable. By keeping our trees binary or unary, we avoid the many pitfalls that occur with circular reference, self-reference, and unlimited recursion. Loops of thought and circular reference will occur since we are using a graph model, but the ability to search linearly or bilinearly down a path to provide hints to prevent self-reference allows us to the opportunity to provide both localized (BFS like) search and DFS like search, or peering down the proverbial Rabbit hole for deeper answers and memories that have been abstracted away. These deeper memories can be then reabstracted through the integration process as new symlinks when they are more important to the present scope of processing and “conscious” thought.

## Abstraction Binding Logic and Second Order Logic

## Abstraction Binding Logic and the Lambda Calculus

We propose that through the use of set theory and ZFC theorems, the foundations of mathematics and linguistics, including the lambda calculus can be constructed. Arithemtic operations will built up and taught to the network through a method similar to their proof in ZFC.