UNIVERSITY OF OSNABRUECK

BACHELORS THESIS

A Comparison of Lojban and Atomese Knowledge Representation

Author:
Roman TREUTLEIN

Supervisor:
Prof. Dr. phil. Kai-Uwe Kuehnberger and Dr. Ben Goertzel

A thesis submitted in fulfillment of the requirements for the degree of Bachelor of Science

in the

Research Group Name
Department or School Name

July 29, 2017

Declaration of Authorship

I, Roman TREUTLEIN, declare that this thesis titled, "A Comparison of Lojban and Atomese Knowledge Representation" and the work presented in it are my own. I confirm that:

- This work was done wholly or mainly while in candidature for a research degree at this University.
- Where any part of this thesis has previously been submitted for a degree or any other qualification at this University or any other institution, this has been clearly stated.
- Where I have consulted the published work of others, this is always clearly attributed.
- Where I have quoted from the work of others, the source is always given. With the exception of such quotations, this thesis is entirely my own work.
- I have acknowledged all main sources of help.
- Where the thesis is based on work done by myself jointly with others, I have made clear exactly what was done by others and what I have contributed myself.

Signed:

Date:

"Thanks to my solid academic training, today I can write hundreds of words on virtually any topic without possessing a shred of information, which is how I got a good job in journalism."

Dave Barry

University of Osnabrueck

Abstract

Faculty Name
Department or School Name

Bachelor of Science

A Comparison of Lojban and Atomese Knowledge Representation

by Roman TREUTLEIN

The fundamental goal of NLP is to translate the semantics and pragmatics of natural language text into a formal language. For this you primarly need a formal language that can express the meaning of any natural language text as well as a way to translate that into your formal language. Ongoing research in this area has lead to the development of various ontologies capable of expressing the meaning of any English text such as SUMO or CYC. Aswell as FSP (Frame Semantic Parsing) that aims to extract semantic structure from the text. FrameNet is a semantic ontology that can be used for FSP but is lacking the coverage of SUMO or CYC. Lojban, a construceted logical language could be another potential resource to be used in this endevour. It's unambiguos grammar based on predicate logic makes it a relativly easy language to work with for Computers. That through active use by a small but dedicated community has been refined to be easy to use and precise. This has also resulsted in various Texts being translated from English to Lojban providing a good data source for machine learning. In this thesis we will examine if during the mapping from Lojban to Atomese we can retain it's simple and short representation. By taking a closer look at a translation system that does exactly that.

Acknowledgements

The acknowledgments and the people to thank go here, don't forget to include your project advisor...

Contents

Declaration of Authorship iii

Abstract vii

Acknowledgements ix

1 Foundations 1
1.1 Introduction to OpenCog . . . 1
1.2 Introduction to Atomese . . . 1

2 Relevant Linterature 3
2.1 FrameNet . . . 3
2.2 SUMO . . . 3
2.3 CYC . . . 3

3 Introduction to Lojban 5
3.1 Background . . . 5
3.2 Grammar . . . 5

4 Methodology 7
4.1 I did stuff . . . 7

5 Lojban Atomese Mappings 11
5.1 Similarities . . . 11
5.1.1 Basic Structure . . . 11
Instance Handling: . . . 11
5.1.2 Predicate to Object conversion . . . 12
5.1.3 2.1.3 Intensional and Extensional Inheritance . . . 12
5.2 Lojban . . . 13
5.2.1 Modifying Arguments in a Commonsensical way. . . . 13
Empty Arg Places. . . . 13
Additional Arg Places . . . 13
5.2.2 Tanru . . . 14
5.2.3 Statement Predicate Conversion . . . 14
5.2.4 Metalinguistic Expressions . . . 15
5.2.5 Attitudinals . . . 16
5.2.6 Same word different location different meaning ??? . . . 17
5.2.7 Utilizing a working memory . . . 17
5.3 Atomese . . . 17
5.3.1 Explicit Truth Values . . . 17
5.3.2 Less Implicit Information . . . 17

6 Conclusion and future directons **21**
6.1 Conclusion 21
6.2 Future Work 21

A AppendixA **23**

List of Figures

List of Tables

For/Dedicated to/To my…

Chapter 1

Foundations

1.1 Introduction to OpenCog

To better understand this thesis it is important to understand the context in which it was created. OpenCog is an open-source project aimed at creating a framework for and an implementation of Artifical General Intelligence. Naturally such a system needs to be able to understand natural language. There does exist an NLP-System for english language comprehensive in OpenCog that is under continous development to slowly eliminate any hardcoded information and replace it by learning subsystems. The first step in the NLP-Pipeline is to grammatically analyse the English text using the Link Grammar Parser (ref). The resulting parse is then used as an input to RelEx. RelEx is an English language semantic dependency relationship extractor. RelEx already creates an output using Atomese Syntax but it is not yet represented in a way that could be used for reasoning. The final step is then to convert this output into PLN style Atomese using RelEx2Logic. This last step is the one that is still most problematic. Because it is just a set of handwritten rules the coverage and accuracy of this is lacking. While this might be fixable with enough effort the final goal of OpenCog is to not depend on such handwritten rules if possible and instead learn them when needed. Ohter Systems in OpenCog are PLN (Probabilistic Logic Network),OpenPsy (Emotion Modeling),MOSES (Meta Optimizing Semantic Evolutionary Search),ECAN (Economic Attention Network),DeSTIN (Deep Spatio Temporal Inference Network)

1.2 Introduction to Atomese

Atomese is the language of the OpenCog Platform and although it can also represent programs here we will only focus on the knowledge representation aspects. The language was designed with the goal of using it in conjunction with PLN to do reasoning. Fundamentally it is also based on Predicate Logic extended with Probability Theory. Each Statement is represented by a labeled Hypergraph. The primary Element being an Atom that has 2 main subtypes namely Nodes and Links. Example Graph:

```
ContextLink tv
        ConceptNode      earth      tv
    InheritanceLink tv
                ConceptNode      mouse      tv
                ConceptNode      animal      tv
```

Every Atom has a TruthValue which can come in various form the most basic one being a "Simple TruthValue" cosisting of a mean Strenght and a Confidence Value. But in the folloing i will often leave out the TV when writing Atomese.

The 2 Node Types used by PLN are ConceptNode's and PredicateNode's

Chapter 2

Relevant Linterature

2.1 FrameNet

2.2 SUMO

2.3 CYC

Chapter 3

Introduction to Lojban

3.1 Background

Originally a language called Loglan was developed by James Cook Brown to test the Sapphire-Worth Hypothesis as such it was designed to be fundamentally different from natural languages. In 1987 the community split leading to the development of what we now know as Lojban. The underlying design of the Lojban grammar is based on predicate logic as such each statement normally consists of 1 predicate comparable with a verb and 1 or more arguments applied to it. Great care was taken to ensure that the resulting syntax would not allow for multiple interpretations as is common in natural languages. This with the fact that there are no exceptions to any of the rules leads to a language that is significantly easier to deal with for a computer. And makes it feasible to write a system that could syntactically translate Lojban into a different Knowledge representation Language. The primary object of Lojban syntax is the bridi describing one predicate relation and consisting of one selbri/predicate and a number of sumti/arguments.

mi : me/we the speaker(s) do : you the listener(s) tavla : x1 talks to x2 about x3

mi tavla do = I am talking to you

As we can see the Predicate has a number of argument places x1-x3 (Some predicates have up to 5 argument places by default) one of which we have not filled in the above statement.

3.2 Grammar

Chapter 4

Methodology

4.1 I did stuff

Chapter 5

Lojban Atomese Mappings

5.1 Similarities

First we will look at the parts in which Lojban and Atomese are similar as well as give some examples.

5.1.1 Basic Structure

As both are based on Predicate Logic the simplest example is just a Predicate with simple Arguments. In Lojban that would be a root word as a Predicate and pronouns for Arguments. In Atomese they are represented by either a PredicateNode or a ConceptNode. Which are then linked using a EvalutionLink. In this simple case each Word can be directly converted to a Node and linked appropriately:

Lojban	Atomese

mi jimpe ti
Definition:
jimpe : x1 understands fact x2 about subject x3
ti : this here; immediate demonstrative it;
	`EvaluationLink tv`
	`  PredicateNode "understand"`
	`  ListLink`
	`  ConceptNode "I"`
	`  CondeptNode "this"`

Instance Handling:

In Lojban or natural languages the meaning of words is very often Context dependent. Depending on the time and location the words I,you,this,that,... can refer to different things. To make this distinction clear in Atomese we will be using 1 general Concept and 1 instance Concept. Where in a given Context the Instance inherits from the General Concept. Similarly we will have a Instance of a Predicate and a General Version. Where the instance only describes the specific relation in the given Context whereas the general version describes all relations of this kind. So the Example above actually looks more like:

```
ContextLink
    ConceptNode "context"
    SetLink
        EvaluationLink tv
            PredicateNode "understanding_instance_789"
            ListLink
                ConceptNode "me_instance_123"
```

```
            ConceptNode "this_instance_456"
        InheritanceLink
            ConceptNode "me_instance_123"
            ConceptNode "me"
        InheritanceLink
            ConceptNode "this_instance_456"
            ConceptNode "this"
        ImplicationLink
            PredicateNode "understanding_instance_456"
            PredicateNode "understanding"
```

5.1.2 Predicate to Object conversion

As there are no Nouns in Lojban if you want to talk about Dogs for example you have to extract this Concept from the associated Predicate. Which is done using words of the class LE in this case "lo" which is the most commonly used.

Lojban	Atomese

lo gerku
Definition:
gerku : x1 is a dog of breed x2 lo :
veridical descriptor: the one(s) that
really is(are) ...
	`SatisfyingSetLink tv`
	`  EvaluationLink`
	`  PredicateNode "is a dog"`
	`  ListLink`
	`  VariableNode "$1"`

The meaning of "lo gerku" is any object that could be placed into the first place of "gerku". So we have an object that is a dog of an unspecified breed. Note that while i have been using the Singular the Lojban phrase does not specify the number in this example. It can be done but just using "lo" you could be talkinga about 1 or many dogs.

Similarly in Atomese the SatisfyingSetLink extracts the Set of all Atoms that could be substituted for the VariableNode. The fuzzy MemberShip of that Atom in the set is then based on the TruthValue of the given Graph that contains the Atom. So in the example above Atomes that would result in a high TruthValue of the EvalutionLink (i.e. Dogs) would be the defining Members of the Resulting Set.

5.1.3 2.1.3 Intensional and Extensional Inheritance

In Atomese there are 2 types of Inheritance Intensional and Extensional. For an extensive explanation read "Probabilistic Logic Networks" Book. In short Extensional Inheritance describes a situation where the elements in Set A are also elements in Set B. Which is why it is also called a subset relationship. On the other hand Intensional Inheritance says something about the properties of the set. So if the Properties of the Set A are also Properties of the Set B than A intensionally Inherits from B. Though most often we just use an Inheritance Link which is defined as mix of both i.e.:

```
OrLink
    Extensional Inheritance Link
      X
```

```
    Y
  Intensional Inheritance Link
    X
    Y
```

Lojban has a similar distinction in the word class LE. The Words beginning with lo (lo,lo'e,lo'i,loi) are all used to describe objects which are a subset of the objects that fit in the Predicate. On the other hand the words starting with le (le,le'e,le'i,lei) are all used for objects which can only be described as something else. For example a box is not a table but depending on the Situation it could be used/described like one.

5.2 Lojban

5.2.1 Modifying Arguments in a Commonsensical way.

In Lojban there are a lot of ways to fiddle with the argument places of a given Predicate which are all designed to make it simpler to express various things.

Empty Arg Places.

As we have seen in the above example it is possible to leave an argument slot free. By default it is then filled by the pro-sumit "zo'e" which has a value that makes the statement true in the given context. This defaulting either happens on trailing unfilled argument places or for the x1 place if there is no argument in front of the Predicate. In Atomese we choose not to represent these at all as they contain no important Information. If one explicitly uses "zo'e" then it will also be present in the Atomese output as there is the possibility of using quantifiers or relative phrase with it.

Additional Arg Places

In Lojban it can easily happen that a given Predicate is missing an Argument Slot to easily express what you want to say. In these cases it is possible to integrate the Argument Slot of another Predicate into Relation. Example:

mi viska do fi'o kanla [fe'u] ti (example 5.1 CLL) I see you [modal] eye: this one I see you with this eye.
viska: x1 sees x2 under conditions x3 kanla: x1 is an/the eye of body x2 ti: this here; immediate demonstrative it; indicated thing/place near speaker.

In this case the speaker states that he see you with his eye. Unfortunately "viska" has no argument place for the eye that you use to see. So we will have to take it from the predicate "kanla" and add it to the Statement. This is done using "fi'o predicate fe'u object". If we didn't have that option we would have to use the more complicated sentence:

mi pilno lo kanla pe ti lo nu mi viska do I use an eye associated with this for the event of me seeing you. pilno: x1 uses/employs x2 [tool, agent, ...] for purpose x3.

When Translating to Atomese the Primary Predicate will imply the Secondary One.

5.2.2 Tanru

In english you have adjectives and adverbs to modify nouns and verbs. Whereas in lojban since everything is based on Predicates the only way to achieve a similar result is by having one predicate modify another.

ti blanu tsani This is a blue type-of sky. This is a blue sky.
blanu: x1 is blue tsani: x1 is an expanse of sky/the heavens at place x2

Similar to english the left predicate modifies the right one. So in the case of "blanu zdani" the resulting predicate is still primarily about a house and also retains all argument places from "zdani" but now has some properties of blueness. In Atomese the naive solution would be to have the instance predicate imply booth underlying predicates namely blanu and tsani but this would cause issue as the following sentence would have the same meaning as the first example.

ti tsani blanu This is blue in a sky type-of way. This is sky blue.

So to differentiate between the primary Predicate and the modifying one the instance Predicate normally implies the general form of the primary Predicate but intensionally implies the modifying Predicate. That could then be interpreted as the new Predicate implying all things related to the primary Predicate but only some properties of the modifying one. Even with this encoding something is lost as they way this modification works differs slightly on a case by case basis. Hopefully the system will be able to learn these differences even without them being explicitly hinted at in the translation output.

5.2.3 Statement Predicate Conversion

When saying things like "I like to understand things." we make statements about other statements. So we need a way to convert a statement into a object that we can use as an Argument. The first step is to abstract the statement into a predicate using the words of class NU. Combined with the LE Class we can then make a object described by a whole statement.

mi nelci lo nu mi jimpe I like the event of me understanding I like it if i understand something.
nu: abstractor: generalized event abstractor; x1 is an event of [bridi].

In this case "nu" takes the statement "mi jimpe" and abstracts it into a predicate : x1 is the event of "me understanding". In Atomese the statement is represented by a predicate that only applies to the situation described in the statement. So we now have to convert that into a predicate that associates a concept with this predicate. For this we use another predicate that has as it's first argument the concept and as a second argument the predicate of the event. But since the event of me understanding is not 1 specific event. We cannot use the given predicate directly instead any predicate that fulfils the same constraints as the given one would be valid.

If we but it all Together we get the following:

Atomese	Explanation

EquivalenceLink
  EvaluationLink
  PredicateNode "nu_mi_jimpe"
  ListLink
  VariableNode "Arg"
  MemberLink
  VariableNode "Arg"
  SatisfyingSetLink
  VariableNode "$2"
  ExistsLink
  VariableListLink
  TypedVariableLink
  VariableNode "PredVar"
  TypeNode "PredicateNode"
  AndLink
  ImplicationLink
  PredicateNode "event_inst"
  PredicateNode "event"
  AndLink
  EvaluationLink
  PredicateNode "event_arg1"
  ListLink
  PredicateNode "event_inst"
  ConceptNode "$2"
  EvaluationLink
  PredicateNode "event_arg2"
  ListLink
  PredicateNode "event_inst"
  VariableNode "PredVar"
  EvaluationLink
  PredicateNode "jimpe_arg1"
  ListLink
  VariableNode "PredVar"
  ConceptNode "mi_instance"
  ImplicationLink
  VariableNode "PredVar"
  PredicateNode "jimpe"
  InheritanceLink
  ConceptNode "mi_instance"
  ConceptNode "mi"
	This is our abstracted Predicate and it is defined by it's equivalence to the following Term.
	
	Any Argument that applies to the predicate is also a Member of the Set of all Events the fullfile the Constraints
	
	We can only define that if there exists at least one Predicate fulfilling the conditions.
	
	Here we define a instance Predicate for this specific Event.
	
	The First Argument is a ConceptNode that describes one event.
	
	The second argument is the Predicate defining the Event
	
	And as a constraint a sepcific instance of 'mi' hast to apply to this predicated.
	
	Additionally it has to be of a specific Type.

5.2.4 Metalinguistic Expressions

In lojban you can have Statements that are not about the current topic but comment on the discourse itself. Those Statements are marked with the non-mathematical

parenthesis “to .. toi” for remarks made by the speaker or “to’i ... toi” for remarks made by an editor. These Statements can’t affect anything in the normal discourse but they can reference it by using already assigned KOhA and GOhA. If they themselves assign KOhA/GOhA those assignments get reverted after the discursive statement finishes.

Similarly we have Statements surrounded by “sei ... se’u” they are like “to ... toi” with the difference that they do not stand on their own but are attached semantically to a part of the normal discourse. Usually the part SEI is attached to could fill an unfilled slot of the Predicate used in the SEI Statement, if it is itself a statement it would first have to converted into a Object using “lo nu/du’u”. Which is most often the case because when attaching a SEI statement to a sumti/object it acts very much like a relative clause.

Figuring out which unfilled slot is filled is not as easy as in other cases. Most rules similar to this one always consider the first unfilled slot as the relevant one. But when filling out the slot one has to consider the Type of argument one is dealing with. Explain this earlier in the generic Lojan intro?. In Lojban there are 3 main types of arguments Clauses , Entities and Numbers with the first to having further subtypes. Each argument slot has a Type that restricts what argument can be placed in it. In normal conversations these rules are often ignored as the intended meaning is relatively easy to comprehend for a human. Example:

mi djica lo plise I want apples. djica: x1 (entity) desires/wants x2 (clause) plise: x1 (entity) is an apple

Here the x2 place of djica should contain a clause but instead has an entity (the LE phrase has the same type ase the argument slot it extracts). It is not clear if you want to have/eat/buy/... the apples. Though this is likely apparent from context.

So in the case of sei where you already have a argument with a certain type it belongs into the first unfilled slot with the same type.

5.2.5 Attitudinals

In lojban the word class UI is used to express and make explicit the attitude or related notions off the speaker. Practically they act the same as SEI statements but with the subject of the Statement always being the speaker. Almost all UI can be translated into a SEI statement with equivalent meaning. Since this is only true for most UI the translator can’t just replace all UI with equivalent SEI. Instead UI are treated as Predicates which apply to the attached sentence part. And we provide the Systems with the knowledge of equivalent UI and SEI statements so far as they exist. For example:

do ui jimpe = do sei mi cinmo be lo ka gelki se’u jimpe ui: attitudinal: happiness - unhappiness cinmo: x1 feels emotion x2 (ka) about x3. gelki: x1 is happy/merry/glad/gleeful about x2 ka: abstractor: property abstractor; x1 is quality/property exhibited by [bridi].

Not providing the system with such a mapping would make it a lot harder for the system to learn the semantics. Since there would be multiple ways to express the same meaning it would have to additionally learn those equivalences.

5.2.6 Same word different location different meaning ???

In Lojban there are Words that based on their location in the Syntax have different but related meanings. I good example of this are words of the class PU.

Lojban	Explanation
mi pu jimpe I in the past understood	Here it set's the tense of the statement
mi jimpe pu do I understood something in your past/before you	Here it adds a tense place to the predicate
mi jimpe i pu bo do jimpe I understood before You understood	Here it describes the relationship between 2 statements

5.2.7 Utilizing a working memory

As in most spoken languages in Lojban the order of sentences matters and there are easy ways to refer to already mentioned things pro-nouns. Contrary to Atomese that like many logic systems has no inherent order to the set of Statements. One of the ways this is used in Lojban are the Word Classes KOhA and GOhA for Objects and Predicates respectively.

ko'a goi la .alis. klama le zarci .i ko'a blanu It-1 also-known-as Alice goes to the store. It-1 is blue

First we define that it-1/ko'a refers to Alice and then we use that to refer to alice in an easy way. In Atomese on the other hand all occurrences of ko'a have to be replaced by the Concept that represents Alice in this Context.

5.3 Atomese

5.3.1 Explicit Truth Values

One significant difference is that in Atomese every Atom has an associated probabilistic Truth Value.

5.3.2 Less Implicit Information

Another significant difference is the amount of implicit Information in a given Lojban / Atomese Statement which would be best demonstrated with an example.

Lojban	Atomese
mi jimpe The statement is true in a specific Context The actual statement The predicate describing this situation implies a general version The “mi” in this context inherits from a more general Concept of “mi” The Context in which the statement is true is in some relation to the current Context This relation is a combination of a temporal offset and a spatial offset The temporal offset implies some direction “PU” and distance “ZI” The spatial offset implies some direction “FAhA” and distance “VA”	ListLink AnchorNode "StatementAnchor" ListLink ListLink ContextLink SetLink ConceptNode "1Wukqb4HQsev0ocNQL EvaluationLink (stv 0.750000 0.90 PredicateNode sumti1 " ListLink PredicateNode "YHCDzFH2Pubhw ConceptNode "ojrdBR1KFW2HlQDC ImplicationLink (stv 1.000000 0.900 PredicateNode "YHCDzFH2PubhwYkYQ PredicateNode "jimpe" InheritanceLink (stv 1.000000 0.900 ConceptNode "ojrdBR1KFW2HlQDCGF9q ConceptNode "mi" EvaluationLink PredicateNode "IpjvEZLtI0Cuw0q4irK ListLink ConceptNode "1Wukqb4HQsev0ocNQL ConceptNode "NowAndHere" ImplicationLink PredicateNode "IpjvEZLtI0Cuw0q4irK AndLink PredicateNode "XvyfNcezKnE9qcp PredicateNode "B43HePzTgxqsweY ImplicationLink PredicateNode "XvyfNcezKnE9qcpLY PredicateNode "ZI" ImplicationLink PredicateNode "XvyfNcezKnE9qcpLY PredicateNode "PU" ImplicationLink PredicateNode "B43HePzTgxqsweYXx PredicateNode "VA" ImplicationLink PredicateNode "B43HePzTgxqsweYXx PredicateNode "FAhA"

The information about the Context is left completely implicit in the Lojban version as well as things like "mi" not referring to the same object all the time.

Chapter 6

Conclusion and future directons

6.1 Conclusion

In Conclusion there are quite a few differences between Lojban and Atomese even though they are both based on predicate Logic. Understanding these differences does not only help in translating between the 2 but is also helpful in learning how to best represent things in the very flexible Atomese Language, as we can leverage the great amount of effort already put into designing Lojban.

6.2 Future Work

Since Lojban can be considered an Ontology it could be of interesting to take a closer look at how well reasoning on this works as compared to other Ontologies like SUMO or Cyc. The hypotheses would be that because Lojban is actually used by a group of human Speakers that have continuously developed the language for ease of use, that the Lojban to Atomese output would lend itself especially well for reasoning, because it can represent conceptually simple things in a logically simple way.

Appendix A

AppendixA