Juan Kenichi Sutan

Professor Wang

CIS 5590 Sec. 004

Project Report

May 2, 2024

NARS Translator: From Narsese to English

I. Introduction

Non-Axiomatic Reasoning System (NARS) is a project with the goal of building a general-purpose intelligent system, also known as Artificial General Intelligence. The logical part of NARS is known as Non-Axiomatic Logic (NAL) which is defined in the formal language Narsese. Narsese is rigidly defined in a context-free grammar that allows NARS implementation systems to generate unambiguous output. However, because of its complex syntax and many uses of symbols, it is not easily understandable for humans, both those who are familiar and unfamiliar with the language. This project hopes to undertake the creation of a NARS Translator tool, capable of translating sentences from Narsese to English that is easily understandable by readers with minimal knowledge of Narsese.

II. Literature Review

NAL is defined incrementally in multiple layers, which extends from the previous layer to have a larger range of expressions capable to be made (Wang, An introduction to NARS, n.d.). The layers of NAL extend from NAL-1 to NAL-9, and with each increasing layer, the capabilities of NAL are increased. The layers from NAL combine logical theory from a variety of logical systems while remaining independent.

At its core, NAL can be defined by statements that consist of terms and copulas. A term typically refers to a recurring pattern representing an object, concept, or variable, within the system's experience. A copula acts to link one term to another, and most commonly is an inheritance copula.

"Higher-order" operations are done when NAL expresses a statement on a statement. In this case, statements are taken as terms, and higher-order copulas are used to express derivations among

type	symbol	code	name	layer
basic copula	→	-->	inheritance	NAL-1
	↔	<->	similarity	NAL-2
	⇒	==>	implication	NAL-5
	⇔	<=>	equivalence	NAL-5
derived copula	⟜	{--	instance	NAL-2
	⟝	--]	property	NAL-2
	⟜⟝	{-]	instance-property	NAL-2
	⇏	=/>	predictive implication	NAL-7
	⇍	=\>	retrospective implication	NAL-7
	⇎	=|>	concurrent implication	NAL-7
	⇎	</>	predictive equivalence	NAL-7
	⇎	<|>	concurrent equivalence	NAL-7
tense	⇏	:/:	future	NAL-7
	⇍	:\:	past	NAL-7
	⇎	:|:	present	NAL-7
term connector	{}	{}	extensional set	NAL-2
	[ ]	[]	intensional set	NAL-2
	∩	&	extensional intersection	NAL-3
	∪	|	intensional intersection	NAL-3
	−	-	extensional difference	NAL-3
	⊖	~	intensional difference	NAL-3
	×	*	product	NAL-4
	/	/	extensional image	NAL-4
	\	\	intensional image	NAL-4
	⋄	_	image place-holder	NAL-4
statement connector	¬	--	negation	NAL-5
	∧	&&	conjunction	NAL-5
	∨	||	disjunction	NAL-5
	,	,	sequential conjunction	NAL-7
	;	;	parallel conjunction	NAL-7
term prefix	$	$	independent variable	NAL-6
	#	#	dependent variable	NAL-6
	?	?	query variable	NAL-6
	⇑	^	operator	NAL-8
punctuation	.	.	judgment	NAL-8
	!	!	goal	NAL-8
	?	?	question on truth-value	NAL-8
	¿	@	question on desire-value	NAL-8

Table 1 Symbols in Narsese Grammar (Wang, Non-Axiomatic Logic — A Model of Intelligent Reasoning (Second Edition), 2023)

statements. At higher layers of NAL, more copulas which are variants of the inheritance copula are introduced, and some copulas may be “higher-order” copulas.

NAL is expressed using the formal language Narsese, which has a defined formal grammar and syntax. Most Narsese content will not be immediately understandable to someone without prior knowledge of the grammar, as it uses a variety of symbols that have previously defined meanings (see Table 1).

NARS is implemented in several different versions. The most popular implementation of NARS is OpenNARS, made in Java, but implementations in C and Python also exist. In the case of OpenNARS, the implementation provides a GUI that allows users to enter input in the form of Narsese statements. The system will print a list of derivations that are occurring onto the GUI to be presented to the user.

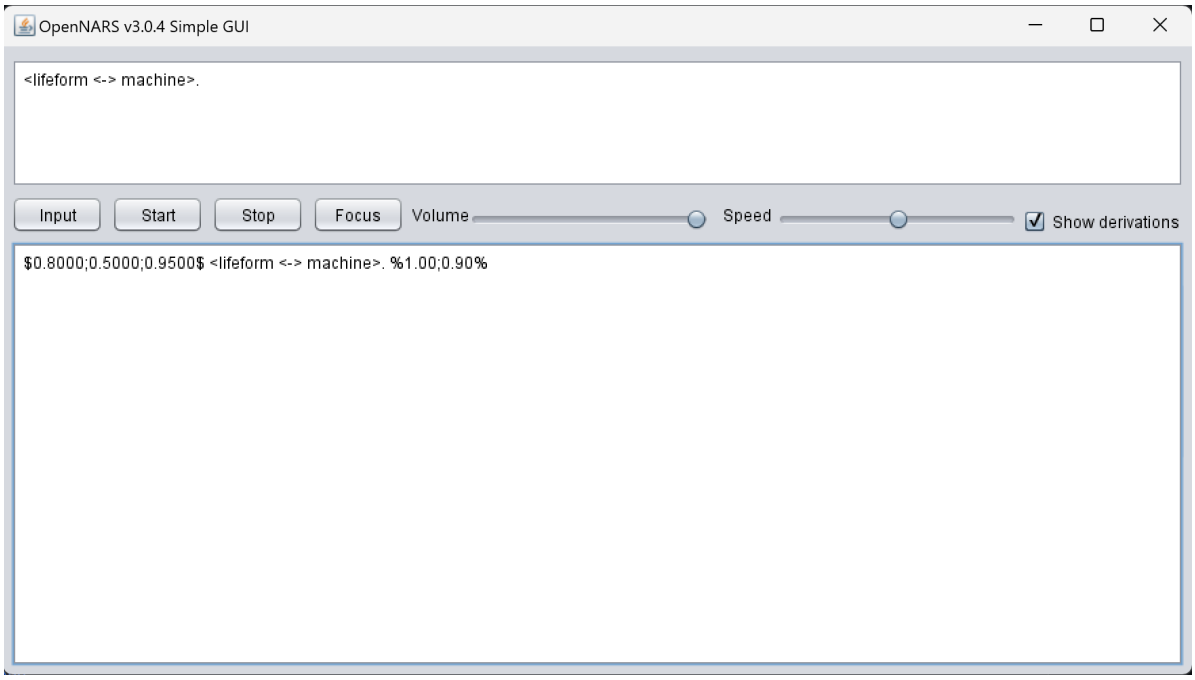

Figure 1 OpenNARS GUI (The OpenNARS authors, 2020)

III. Project Outline

The NARS Translator project can be divided into several steps:

1. A working proof of concept
2. Expansion to cover all NAL layers
3. Integration with a NARS implementation

A working proof of concept will be made that provides a limited functionality of the imagined final product. Depending on the final release of the NARS Translator alongside a NARS implementation, the proof of concept and subsequent versions will be made in the language of the implementation. For example, a version hoping to integrate with the OpenNARS platform should be created in Java, etc.

The NARS Translator will be made using a rule-based approach. A series of rules will be defined as to how a statement will be processed. Through those rules, the sentence will be processed into a result string that best resembles an English language sentence. Due to the inherent variability of term naming by the end-user, it is impossible to always create perfectly grammatically correct English translations. However, results should strive to be as clear, unambiguous, and correct as possible despite this uncertainty.

Following a working proof of concept, the limited functionality should be expanded to cover more cases. An idealized workflow would be to continuously develop more rules to cover cases from NAL-1 iteratively until NAL-9. However, in some cases, it may make more sense for the developer to ignore iterative updates and implement features as efficiently as possible, in any order.

A complete NARS Translator tool can then be integrated into existing versions of NARS implementations. Given the rule-based approach that is generally computationally inexpensive,

the NARS Translator could run alongside a NARS implementation in real-time. This would allow the system to provide derivations and their English translations, allowing for better understandability of the system's reasoning process, especially for novice or unfamiliar users.

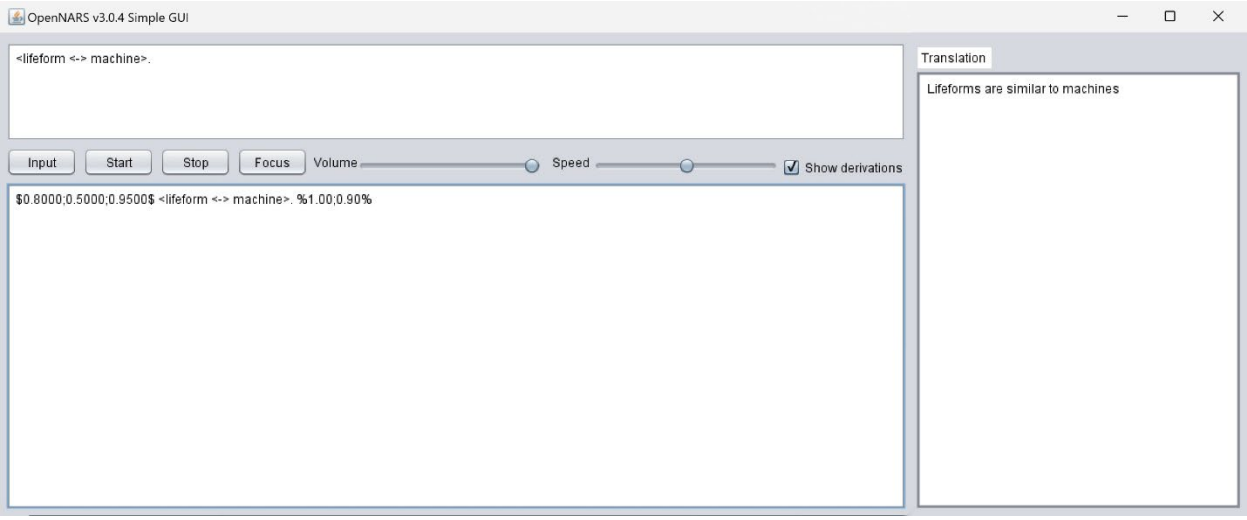

Figure 2 OpenNARS GUI w/ Imagined Translator Tool Integration

IV. Proof of Concept

As part of this project, a proof of concept of the NARS Translator tool has been created. The initial product was created using Java. However, a later iteration was made in Python. The most up-to-date version of the tool is the version in Python, which has an expanded capability set. The main objective of the proof of concept is to create a snapshot of the completed NARS Translator, with capabilities at least to completely translate the "Detective example - who is the criminal?" by OpenNARS.

As of the time of writing, the NARS Translator tool can receive input in the form of Narsese sentences and will categorize the type of sentence based on a set of rules. The sentence will first be checked if it consists of multiple sentences, or if it is a higher-order statement. In the case of it

being a higher-order statement, a recursive case will be met and the sentences acting as terms will be re-entered into the process as individual statements. Afterward, the tool will check if a frequency value is present. In the current version, the tool is not capable of considering confidence values. Once a frequency value has been retrieved, it will retrieve a sentence type based on the punctuation, followed by the type of copula it contains.

```
<human --> lifeform>.
Result: Humans are lifeforms
<{tim} --> (/,livingIn,_,{graz})>.
Result: Tim is living in graz
<sunglasses --> (&,[black],glasses)>.
Result: Sunglasses are black glasses
<{?who} --> murder>?
Result: Who is (a/the) murder?
<{tim} --> (/,livingIn,_,{graz})>. %0%
Result: Tim is not living in graz
<{tim} --> (/,livingIn,_,{graz})>. %50%
Result: There is a 50% chance that tim is living in graz
<$1 --> [aggressive]>.
Result: Someone/something that is aggressive
<<$1 --> [aggressive]> ==> <$1 --> murder>>.
Result: Someone/something that is aggressive implies that someone/something is murder
<<$1 --> (/,livingIn,_,{graz})> ==> <$1 --> murder>>.
Result: Someone/something that is living in graz implies that someone/something is murder
<lifeform <-> machine>.
Result: Lifeforms are similar to machines
<Tim {-- human>.
Result: Tim is human
```

Figure 3 NARS Translator Example Input and Output

Once the initial processes are done, the sentence will then be broken into the terms that make it up. In some cases, the sentence may have a compound term that contains more than one term (Example: <{tim} --> (/,livingIn,_,{graz})>. See Figure 3). In this case, the second term is saved by the tool as a “verb” for the sake of simplicity. This assumes that every implication statement will follow the general rule of Term 1 is Verb Term 2. Although this rule may not necessarily hold as the capabilities of the tool are expanded, it remains functional for now.

After terms are sanitized from any symbols or other non-alphanumeric characters, it is passed on to a construction function that will assemble a translation. Using a variety of switch cases relying on the previous values gathered for sentence type and copula, it will assemble the terms into a sentence in the English language.

V. Current Limitations

The proof of concept, given its nature to be an incomplete snapshot of a finished product, contains many limitations to its capabilities. This section will attempt to list out the limitations that the proof of concept was not able to achieve in regards to the original goal of being able to translate the “Detective example - who is the criminal?” Limitations outside of those are intended to be corrected prior to the completion of the tool.

```
<{tim} --> (/,livingIn,_,{graz})>.
Tim is living in graz

<{tim} --> (/,livingIn,_,{graz})>. %0%
Tim is not living in graz

<<(*,$1,sunglasses) --> own> ==> <$1 --> [aggressive]>>.
1sunglasses are owns implies that someone/something is aggressive

<(*,{tom},(&,[black],glasses)) --> own>.
Tomblackglasses are owns

<<$1 --> [aggressive]> ==> <$1 --> murder>>.
Someone/something that is aggressive implies that someone/something is murder

<<$1 --> (/,livingIn,_,{graz})> ==> <$1 --> murder>>.
Someone/something that is living in graz implies that someone/something is murder

<sunglasses --> (&,[black],glasses)>.
Sunglasses are black glasses

<{?who} --> murder>?
Who is (a/the) murder?
```

Figure 4 Output of Detective example - who is the criminal?

The proof of concept currently still experiences some poor translations from the given example. The tool is currently not able to process compound terms existing as the first term of a sentence. As a result, the example <(*,{tom},(&,[black],glasses)) --> own>. was translated very poorly (see Figure 4). The remaining mistake also follows a similar pattern, which is due to a compound term existing as the first term. However, more complex statements such as the 2 out of the 3 higher-order statements were translated correctly.

VI. Future Work

As the project is still in very early stages, there is still much work ahead to complete a finished product in the form of a NARS Translator integrated with a NARS implementation. The existence of this document hopes that in the future the project can be completely realized, as the general steps and a framework have been described and created.

The most immediate work that should be perfected is the proof of concept. The most proximate version should attempt to solve the issues listed in Chapter V and achieve the initial goal of completely translating the “Detective example - who is the criminal?” Afterward, appropriate iterations should be made to complete all NAL layers.

In regards to integration with a NARS implementation, the completion of a NARS Translator tool in any programming language should not pose much of a roadblock to importing the tool to an implementation based in another language. With a complete rule-based translating logic completed, especially with the assistance of generative AI such as ChatGPT, converting said logic to different programming languages would not be a difficult task. However, additional work in updating relevant GUI will be necessary as well as establishing proper input and output flows.

VII. Relevant Code

To access code relevant to the NARS Translator- Python, the public repository can be found at the following link: https://github.com/kenichisutan/nars-translator-python.

To access code relevant to the OpenNARS Translator (Java), the public repository can be found at the following link: https://github.com/kenichisutan/opennars-translator.

VIII. References

The OpenNARS authors. (2020). OpenNARS (Version 3.0.4) [Computer software]. Github. https://github.com/opennars/opennars.

Wang, P. (2023). Non-Axiomatic Logic — A Model of Intelligent Reasoning (Second Edition). Draft made accessible as part of CIS 5590.004 Spring 2024, Temple University.

Wang, P. (n.d.). *An introduction to NARS*. Retrieved 01 18, 2024, from A Logical Model of Intelligence: https://cis.temple.edu/~pwang/NARS-Intro.html