**Temple University**

May 10, 2010

**Learning and Translating NARS**

**Safi Haq**

 To start, N.A.R.S must be understood in its differences with other reasoning systems. NARS is it is built with the capability to adapt to its environment and to work with insufficient knowledge and resources. This distinguishes it from other intelligence systems. While other systems are built with the invariant that resources are always sufficient and accessible and the system itself is fixed. This allows NARS a variety of powerful tools to help the system deal with different situations with an expected outcome. The system can then use its existing database to infer new knowledge and further expand its understanding of its environment. This is all done in real time, an advantage gained by assuming insufficient knowledge and building logic based n experience. With this understanding in mind, one can exchange existing databases from other systems to test the limitations of either system. If one database cannot be translated into the others syntax, or if inference in one system results in an inability to translate to the original, then a limitation has been found, which if corrected, can improve the system.

 After reading about NARS and its structure, I felt I had a much better grasp of what intelligence systems are trying to achieve and what NARS was specifically advantageous at. Looking at the single step testing cases help me gain a better understanding of how the system worked. Even from the start, it was clear that the system had very powerful capabilities. Allowing frequency and confidence values allowed for a spectrum of relations where only bivalent relations existed in other systems, reading about other intelligence systems made it clear that this would become important when testing and translating data from other languages since they might not contain frequency or confidence values. Being the most obvious restriction between translations, something as simple in NARS as revision will becomes quite difficult in languages not allowing the spectrum of truth values that NARS allows. This meant that translation from NARS to another language with statements requiring specific frequency and confidence values would need a sort of disposal of specifics to continue, this needed to be noted when working with other databases.

 My understanding of set theory initially became an issue when trying to understand the grammar of NARS. Not being used to thinking of elements, sets and relations with this sort of semantic language, it became clear that problems might arise simply from my lack understanding set theory. As in, a possible limitation between translations might not be the exclusion of a grammar rule from either semantic language, but me simply not forming the connection of that grammar rule to the exiting standard rule set. This was another concern to watch out for while translating, that several translations would need to be done while testing grammar rules between systems in order to insure an existing limitation.

 The sample single step cases helped me gain an understanding of the NAL syntax in its usage. From the first example NAL1, we can see can see clear advantages of NARS compared to other languages, especially in its usage of frequency and confidence values. The first case is an example of revision. Here we see the advantage of weighted evidence for statements. With other logic semantic, the relation would either have to exist or not exist. There is no way to counter evidence which does no render existing information false, NARS on the other hand can fairly balance the given evidence against its database and infer a statement accordingly. We can see other simple cases being worked on, also showing the advantages of the frequency and confidence values and the consistency of the values across different types of simple inference. The advantage of the frequency and confidence values can also be seen in the outputs, both of the questions and single step inference statements, showing the underlying experience based logic.

With a different system, the output would simply be bivalent. Allowing only the possibility for the system to return a statement which it thinks is entirely true or entirely false, NARS on the other hand can return a spectrum of assurance, allowing output that more closely resembles human intelligence and allowing it to infer more specifically based on statements it had previously inferred. Imagine another system inferring a statement which is partially true, then returning entirely true. If the “partialness” of that statement is significant in inference, then any further inference by the system will need to take it into account, this is possible with NARS. In fact, the assurance of a statement and the number of inference steps taken are taken into account.

The other sample cases show the other semantics of NAL eventually leading to procedural inference. The variety of syntax allows NAL to create statements that can better represent different forms of knowledge. This includes allowing various relations to objects, their properties and their set nature. This allows NARS to accurately represent logic and reasoning based on evidence that is similar to human logic. The system can work with simple syntax while representing complex relationships. For instance, rules of comparison and set and inheritance relation allow logic to easily be represented while holding the invariants of such relations. By keeping complex ideas in a simple format without the loss of information, NARS can infer new knowledge in real time, and in a way that resembles human intelligence.

By studying the semantics language that NARS uses, I began to understand the reason by a spectrum of truth values is needed and how it leads to a more accurate representation of information. By inferring statement containing truth values, NARS can benefit greatly in accuracy even inferring on atomic statements, for instance the case of revision can easily be handled by NARS and it is at this stage that the experience building advantage really became clear to me. The development of NAL with truth values allows for the more complicated, and compound statements be verified in consistency allowing a better and more accurate analysis to be made by the system when inferring new knowledge. We can see how something as simple as revision of the properties of an element within a set can be accurately represented and the system can develop information quickly in real time while handling the various invariants of relations and variance of consistency of truth values by using such a concise language. And the advantage gained by using NAL and truth values result in real time accurate inference.

Now understanding NARS and NAL, it was important to see where the future of NARS would lead before starting translations. By adding additional existing databases and frameworks of logic, a NARS system can learn and develop knowledge, and the inference of such knowledge will lead to a better understanding of the framework and database and possibly even add knowledge to it. This allows NARS systems to grow and develop experience based databases based on other systems, and the combination of retrieving and developing information can allow NARS to become a much more powerful system by not limiting its resources to its own database or language. Simply put, the integration of other systems into NARS can equip a NARS system with a larger range of tools and a larger set of resources.

Reading An Evidential Path Logic for Multi-Relational Networks by Dr. Rodriguez helped me gain an understanding of the graph nature of logic in NARS. After learning the logic itself, it was important at this point to understand the relational nature of NAL and how it is organized in a graph network. The paper also introduced me to RDF and semantic web, talking about the organization of RDF in terms of relations. I noticed right away that, unlike NARS which had specific relation rules like similarity and inheritance, RDF seemed to have the most basic of syntax. It used triples consisting of subjects, relations and objects. But the relations were just an input, there was no easy way to check between relations like in NARS by syntax alone, the relation needed to be “understood” by an outside user or taken into account. I could not simply assume the syntax, if correctly written, would correctly handle relations. This would need to be managed correctly by me. I could not simply take any existing data element (in the RDF case a triple) and easily translate into Narsese by comparing semantic rules, I could however compare the “relation” input in each triple and find a suitable set of semantic rules in Narsese to allow accurate representations of every triple. And then it would be simple enough to take the resulting statements that NARS would infer and translate them into triples because of their basic format.

The article also talked about the large number of resources that current semantic webs waste by working in Axiomatic systems, and how much can be gained by adding a truth value to the triples and mimicking NARS evidential system by allowing experience based information. The article really helped me understand the current nature of such systems and the advantages that are offered by implementing them into NARS. The resource usage alone makes it worthwhile, but allowing inference on existing frameworks and databases shows how NARS can be used to use and improve other existing systems.

LOAN was the next system I covered, and it was here I learned the importance of selecting languages. The introduction, LOAN for Narsese Users, helped me understand the difference of different semantic frameworks and the usability by the average user to understand them. Here LOAN was created to handle data in a more “scribbleable” manner than its predecessors, which allows the average computer user to be able to understand the language. That was its goal of course, although looking at the translation from Narsese to LOAN it became clear that some overhead of knowledge was required for the average user in order to understand the more complex syntax rules of LOAN. The similarities to Narsese were obvious, including the addition of truth values to an RDF type language. The paper also gave an insight into RDF as a framework, talking about the various elements that are included in RDF. The triples, the Dublin core prefixes, and various rules to handle Uniform Resource Identifiers. This is not relevant to Narsese translations directly, but in terms of a semantic web, this will represent the sources of information on the internet. By reading and studying LOAN, I gained an insight into the necessity for simplicity (which helped me understand the vague nature of RDF triples) and the ability to merge various databases together.

From then on, it was necessary to learn about semantic webs in general and where NARS could offer improvement, help existing systems reach significant goals and use existing databases to expand its own knowledge. A semantic web is a method of defining information on the internet, by having fixed definitions. Systems can utilize such information using queries to understand the usage of the internet both in terms of data networks and actual user usage. By being able to access such information, a system will now have access to a significantly larger and ever growing database. For NARS systems, this allows for more complex analysis to be performed while inferring. Even while assuming insufficient knowledge, with a significantly larger dataset about a given question, the system can more accurately infer statements. This leads to a more intelligent system and NARS experience based statements and procedural content allows it to handle the constantly changing database and massive amount of information in real time. The improvements by equipping a semantic web to NARS are significant enough to add to any system, especially those working with the internet.

By utilizing Resource Description Frameworks or RDF, a semantic web can network and define information for users to query. This is often done using SPARQL, and RDF query language. By using SPARQL with an existing database, a user can query various types of information from a database using simple language returning RDF triples. And by utilizing the triples, a user can understand the relationship and nature of the queries. Something as simple as returning all of the users of a website will return a series of triples with the subject “website”, relation “users of” and object “User Name”. This is important to users, as RDF will allow the source off all the triples to also be accessed and thus each triple will link the “User Name” and “website” with a known relation, allowing a user to map the website by its user database. By performing more queries a user can further map relations of a site, and even expand further mapping relations of other sites. Thus allowing a real time networking of the internet, on a specified localized level, allowing the user to control the level of localization and level of information it requires. By replacing the user with a NARS system, and adding truth values to the triples, it is easy to see the potential for NARS to understand and map the relationship of a website or relations between websites even beyond the capabilities of SPARQL queries, by inferring new information. By utilizing such a NARS system, a user can truly understand the relations of the internet and a given localized level. And with the way NARS is developed, with the intention of adapting with insufficient knowledge, this can all be done in real time and the user can gain accurate information that would not have existed previously.

There are five main challenges currently facing the semantic web community, and here are the potential ways I believe that NARS can combat some of them. This is of course with the implementation of equipping NARS with a semantic web, not simply adding truth values to triples and doing first order inference. The first of which is “vastness”, when talking about the internet one must understand that there are massive amounts of data to be dealing with. With about 50 billion pages of data to handle, any system trying to utilize this must be able to handle tremendous amounts of information in order to correctly understand the given information. Luckily, since NARS is built with the understanding that it must adapt to situations and it already assumes insufficient knowledge, it is able to handle such a problem. By treating a NARS system like a child, and feeding it appropriate localized data, a NARS system can infer on such data and return a localized value. Now with relatively small data this can easily be done in real time, but even with massive amounts of data, as talked about by Dr. Rodriguez in An Evidential Path Logic for Multi-Relational Networks, NARS implementation within a system greatly decreases path walks to obtain and infer knowledge in existing databases in RDF. This means that NARS would greatly improve the speed and efficiency with which resources are handled and by handling local data, a system could further reduce costs when handling that particular semantic web.

“Vagueness” refers to imprecise data, an example being “red”, “young” or “short”. As humans, we have an understanding of what “red” refers to, and we can even tell a system where “red” exists vaguely within color spectrum. But within a localized setting (say a campus in America), “red” will actually trigger a specific location on the color spectrum, what is “red” for one person might simply be “reddish” for another in another setting. And a proper semantic web trying to understand its database needs to be able to handle such vague information more accurately. I believe that by utilizing the truth value system in NARS, a more accurate “fuzzy logic” representation of information can be derived instead of relying on bivalent information. By utilizing truth values and given information, NARS can infer the overlapping vague ideals and represent them much more accurately than in current systems.

Inconsistency is a major problem within a semantic web’s database. If there can exist two sources of information, one being contradictory to the other, then the information the system holds or infers becomes invalid. We can see here that something as simple as a contradiction cannot be handled in current systems, and with the size and dataset of the internet, it is highly unlikely that a system would never face one. NARS has an existing framework for handling such contradictions, and with the truth value system, can accurately represent frequency and confidence when given a contradiction allowing it to handle contradictions. Here we can see the distinct advantage of using non-Axiomatic experience based logic rather than the bivalent standard. Allowing for the truth values to change as they do in procedural inference, NARS can allow for even more accuracy with revisions, compared to current systems which cannot even handle them.

“Deceit” is when a producer of information is intentionally deceiving the system by sending false information. Understanding the previous challenge of handling inconsistency with current systems, one can see how serious this challenge becomes. With NARS, it is not directly handled within the system. The system might work to adapt to its surroundings and infer assuming insufficient knowledge, but if that knowledge is faulty, then the system will infer using false statements. Of course dealing with the semantic web allows NARS to have a massive database to counter every new source of info with. Utilizing the truth value system, if the majority of information is correct and verified, than the system will infer based on this information. The small amounts of deceit will not drastically ruin inference or logic handling. We can also see the advantage of utilizing a truth system here, allowing frequency and confidence to change based on the total number of positive evidence and the total number of evidence. This allows NARS to handle deceit in an acceptable manner, and is sufficient for the usage of a semantic web.

Understanding how semantic webs work, and the ways that equipping NARS with one could improve both NARS and the existing semantic web, it was time to look at various existing databases, in order to truly utilize the potential of a semantic web framework, I thought it would be wise to use a changing massive database. DBpedia is one such attempt at a semantic web framework. The project attempts to develop a structured framework for data for Wikipedia, which is currently published in RDF, and allow semantic web agents to access this data through queries using SPARQL. By utilizing linking and interlinking with queries, users can perform advanced inference with a massive database. Wikipedia currently has millions of articles, constantly being updated in several languages. This includes querying with unknown data and data patterns, allowing NARS to perform real time inference on every Wikipedia article. The project is certainly a significant database to equip to any NARS system.

Now actually working with DBpedia, we will query two president and their starting dates in office. The SPARQL query results in the following.

PREFIX skos: <http://www.w3.org/2004/02/skos/core#>

PREFIX dbpedia2: <http://dbpedia.org/property/>

SELECT ?presName, ?startDate WHERE {

 ?presName skos:subject <http://dbpedia.org/resource/Category:Presidents\_of\_the\_United\_States>;

 dbpedia2:presidentStart ?startDate.

}LIMIT 2

The prefix informs us where this information is coming from what schema is being used, so the query parser can figure out how “startDate” should be used. The Limit is used to limit the results. The following list is returned.

[:George\_Washington](http://dbpedia.org/snorql/?describe=http%3A//dbpedia.org/resource/George_Washington) ![[http]]()1789

[:Thomas\_Jefferson](http://dbpedia.org/snorql/?describe=http%3A//dbpedia.org/resource/Thomas_Jefferson) ![[http]]()1801

Note that both presidents have a resource identifier and are linked to their DBpedia resource locations. The now queried information can be displayed as the following RDF triple <George Washington> <start date ><1789> and <Thomas Jefferson> <start date ><1801>. Here we see “George Washington” as the subject, “start date” as the relation to the object “1789”, representing in the following graphs.

 **“Start Date”**

 **“Start Date”**

Translating this into Narsese gives us,

IN: <(\*,GeorgeWashington, StartDate) --> 1789 >.

IN: <(\*,ThomasJefferson, StartDate) --> 1801>.

Running a single step of inference results in.

IN: <(\*,GeorgeWashington,StartDate) --> 1789>. %1.00;0.90% {0 : 1}

IN: <(\*,ThomasJefferson,StartDate) --> 1801>. %1.00;0.90% {0 : 2}

1

 OUT: <GeorgeWashington --> (/,1789,\_,StartDate)>. %1.00;0.90% {1 : 1}

 So George Washington inherits the extensional image of 1789 over Start Date. Represented by the same RDF triple <Start Date> <start date ><1789> since it can still represents the president who started, among the set of start dates, on 1789. If we expand the original triple into more detail, possible arranging the presidents into a set of presidents, inference would have been very different with the new addition of information. We see the vagueness of RDF immediately, and while this is an advantage when searching through millions of triples, the queries require a user to interpret them. While NARS does not assume knowledge, DBpedia does, especially when not defining literals. This needs to be taken into consideration when trying to translate between DBpedia. Also note that while the frequency and confidence values remained the same throughout inference, they would not have through multiple steps of inference. And at a certain point, the user will need to translate the spectrum the truth value represents into a bivalent statement in order to work in RDF. One can see the necessity of adding truth values to triples in order to equip DBpedia with a NARS system and the potential that the massive database holds for inference.