

Translating UNL Expressions to Logical Expressions

M.Tech Dissertation

Submitted in partial fulfillment of the requirements
for the degree of

Master of Technology

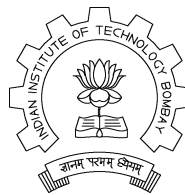
by

Suresh Kumar M

Roll No: 02305012

under the guidance of

Prof.Pushpak Bhattacharya



Department of Computer Science and Engineering
Indian Institute of Technology, Bombay

Mumbai

July 27, 2004

Acknowledgments

I express my deep sense of gratitude towards my guide **Prof.Pushpak Bhattacharya** for his support, guidance and encouragement throughout the work. I thank **Adam Pease** for generously providing his software for this project.

Suresh Kumar M
02305012

Abstract

Universal Networking Language(UNL) is an interlingua for machine translation. The meaning of a sentence is represented as a list of binary relations. Logical reasoning from this representation is not possible since the UNL representation do not have variables, quantifiers and implication explicitly represented. Logic formulas containing these constructs, can be derived from the UNL expressions. Reasoning can be done on these logic formulas.

In this report, we present an approach for converting the UNL representation of sentences that can be readily represented in logic, to predicate expressions. Concepts and relations from Suggested Upper Merged Ontology(SUMO) are used in these expressions.

Contents

1	Introduction	2
1.1	Outline	3
2	Universal Networking Language	4
2.1	UNL	4
2.2	UNL Knowledgebase	6
2.3	Representation	8
2.4	Inferencing	9
2.5	Conceptual Graphs	10
3	Suggested Upper Merged Ontology	12
3.1	SUMO	12
3.2	Relation with WordNet	14
4	Mapping	16
4.1	Noun UWs	16
4.2	Verb UWs	17
4.3	Adjectives and Adverbs	18
4.4	Relations	19
5	Translation to Logic	21
5.1	Restructuring	21
5.1.1	Implication	21
5.1.2	Conjunction and Disjunction	23
5.1.3	Negation	24
5.2	Using SUMO	24
5.3	Quantification and Translation	24
6	Conclusions	28
A	Marcus Problem	29

Chapter 1

Introduction

Universal Networking Language a.k.a UNL is an interlingua for machine translation of natural language sentences. The UNL consists of binary relations, attributes and universal words. The meaning of the natural-language sentences is represented in an intermediate form which can be translated to sentences in another language. The *Enconverter* system generates intermediate representation from the natural language sentences. Natural-language sentences from the intermediate representation are generated by the *Deconverter* system. The language-specific details for analysis, generation of natural language sentences are encoded as rules for these systems. This is the UNL model of machine translation. The system is called the UNL system.

The intermediate representation in UNL can be represented as a network of nodes interconnected with binary relations. The universal words or UWs are the nodes of the network. The binary relations from UNL represent the role of the UW in the sentence. The network is also represented, in linear form, as a list of predicates- each binary relation in the network forming a predicate with arguments as the UWs of the adjacent nodes of the binary relation. These UWs and binary relations - either in network form or linear form - are called *UNL expressions*.

Though the UNL expressions in linear representation has a structure similar to relations in predicate logic, logical reasoning from these expressions can't be done directly. To be specific, the reasoning mechanism of predicate logic can't be used directly. The reasoning mechanism in predicate logic is built around connectives(implication, conjunction, disjunction and negation), quantifiers and the notion of variable. These connectives, quantifiers and variables are not represented explicitly in UNL. Connectives are present as binary relations. Quantifiers are represented as UWs. Node identifiers associated with a UW are the equivalent of a variable. These clues can be exploited for translating the UNL expressions to logical expressions, and inferencing can be done on these expressions. The thesis is an attempt in this direction.

The scope of the work includes obtaining the logical representation from the UNL representation of the sentences that can be readily represented in logic i.e.,

- Sentences containing quantifiers *every, all, some*.
- Sentences containing *if, and, or* as connectives.

The concepts and relations from Suggested Upper Merged Ontology are used in the logical representation of the UNL expressions. Concept or relation to be used for a particular Universal

Word is determined by the relation between UW, WordNet and the SUMO Ontology.

1.1 Outline

Chapter 2 gives the necessary background on Universal Networking Language. Description of universal words and UNL expressions is presented. A knowledge representation formalism called Conceptual Graphs [Sow83] is described in this chapter. This formalism is very much similar to the semantic networks generated by UNL system.

Chapter 3 describes Suggested Upper Merged Ontology[NP01][SUM03]. A description of the top level categories in the ontology is presented. Its relation to WordNet is described. An inference engine used with SUMO ontology is described.

Chapter 4 describes the method for linking universal words to SUMO concepts. The structure of the UW, and the relation between SUMO concepts and WordNet synsets is used extensively for linking a UW to the corresponding SUMO concept.

Chapter 5 presents the approach for translation of UNL expressions to logical expressions. Principles from conceptual graph theory, discourse representation theory are used in the translation. Chapter 6 gives conclusions and the future directions.

Chapter 2

Universal Networking Language

UNL[Fou03] or Universal Networking Language is an *interlingua* used in Machine Translation of natural languages. The language consists of binary relations, attributes and Universal Words. The sentential information is represented as a semantic network of Universal Words interlinked with binary relations, and annotated with the attributes. Figure 2.1 is the UNL representation for *John ate rice with spoon*.

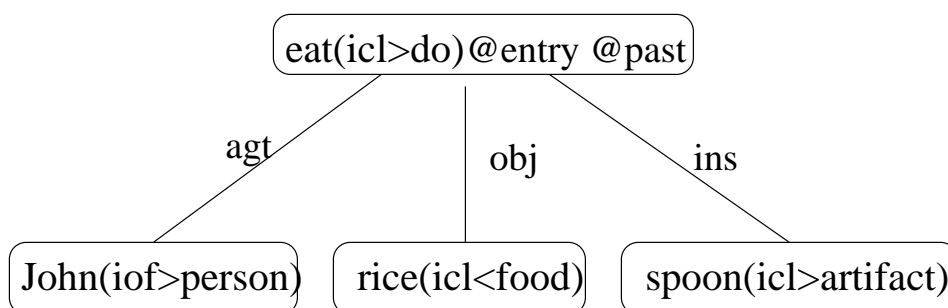


Figure 2.1: UNL representation for *John ate rice with spoon*


The UNL system has two components-*Enconverter* and *Deconverter*. Intermediate representation from sentences in natural language is generated by *Enconverter*. This intermediate representation can be translated to another natural language using *Deconverter*.

2.1 UNL

UNL consists of three categories in its vocabulary- universal words, relations and attributes. Universal Words or UWs correspond to words in a sentence. The relations represent the relationship between UWs present in the sentence. Attributes annotate the UWs to give the speakers view point.

Universal Words The Universal Words or UWs correspond to words in a sentence. UWs are formed such that they have a single meaning. Each UW has a headword(a word in a natural language), and an optional list of constraints. These constraints disambiguate

the meaning of the headword. The relations *icl*, *equ*, *iof*, *pof* are usually present as a constraint in the UW. These relations are subclass, equivalent, instance and part-of relations of UNL. The structure of the UW for “fruit” is shown below. Head-word is **fruit**. The constraint list defines the UW as a subclass of food, part of plant.

head-word	Constraint List
	
<code>fruit(icl>food>functional thing, icl>part of plant, pof>plant>living thing)</code>	

The relations in the constraint list serve as disambiguating information. The two UWs with “club” as the head-word are- `club(icl>weapon>tool)` is a base-ball or golf club. `club(icl>organization>group)` is formal association of people with similar interests.

The Universal Words thus defined are arranged in a hierarchy in UNL Knowledgebase[Un103]. The concepts at the top level of the hierarchy are explained in next section.

Relations The binary relations of UNL represent relationship between two UWs present in a sentence. There are 41 relations in UNL. The relations can be roughly grouped into 3 categories- the role of the UW in an action (*agt*, *obj*, *ins*, *cag*, *gol* etc), relations indicating time and place where an event occurred or an entity is present (*plc*, *tim*, *plt*, *tmt* etc), relations for expressing state or attributes of an entity (*aoj*, *mod*, *cao* etc). Description of a few relations follows.

- agt** defines a thing that initiates an action.
Ex: “Car runs” is `agt(run(icl>act(agt>volitional thing)), car(icl>vehicle))`.
- obj** defines a thing that is directly affected by an event or state.
Ex: “Man murdered” is `obj(murder(agt>thing, obj>thing), man(icl>human))`
- ins** defines an instrument to carry out an event.
Ex: “write with a pencil” is represented as
`ins(write(icl>express(agt>thing,obj>thing)), pencil(icl>stationery))`
- aoj** defines the attribute or state of a *thing*.
Ex: “leaf is read” is represented as `aoj(red(aoj>thing), leaf(pof>plant))`
- mod** defines a thing that restricts a focused thing.
Ex: “All Indians” is represented as `mod(Indian(icl>person), all(mod<thing))`.
- plc** defines a place where an event occurs, a state is true, or a thing exists.
Ex: “cook in the kitchen” is represented as
`plc(cook(icl>do), kitchen(pof>building))`
- tim** defines the time when an event occurs or a state is true.
Ex: “leave on Tuesday” is represented as `tim(leave(icl>do),Tuesday(icl>time))`
- rsn** defines a reason why an event or a state happens.
Ex: “Didn’t go because of rain” is represented as
`rsn(go(icl>do), rain(icl>weather))`
- con** defines an event or state that conditions a focused event or state.
Ex: “If u are tired, we will go home” is represented as
`con(go(icl>move(agt>thing,gol>place,src>place)), tired(aoj>thing))`

- and** defines a conjunctive relation between concepts.
 Ex: “John and Mary” is `and(Mary(iof>person), John(iof>person))`
- or** defines a disjunctive relation between concepts.
 Ex: “John or Mary” is `or(Mary(iof>person), John(iof>person))`

Attributes Attributes are used to describe what is said from the speaker’s point of view UWs and Relations describe objective things, events and states-of-affairs in the world. Attributes enrich this description by representing speech acts, propositional attitude of the speaker, time and reference with respect to speaker. Description of a few attributes follows.

@generic	a generic concept	ex: “ <i>Dog</i> is a faithful animal.”
@def	already referred in the discourse	ex: “ <i>The book</i> you lost”
@indef	non-specific class	ex: “There is <i>a book</i> on the desk”
@not	negation	ex: “John <i>do not like</i> Jim”
@past	happened in the past	ex: “It <i>was raining</i> yesterday.”
@future	will happen in future	ex: “I <i>will submit</i> the report tomorrow”

The attributes @generic, @def, @indef represent speakers view of reference i.e whether the expression refers to a particular entity or all the entities of that class. The attribute @generic refers to the whole class where as @def, @indef refer to a particular individual of that class. The attribute @not is the equivalent of a negation. The attributes @past, @future, @present represent the time of an event from the speaker’s view.

2.2 UNL Knowledgebase

The UNL knowledgebase[Unl03] is a language-based ontology. The Universal Words are defined and categorized based on their usage in a natural language. The UWs are arranged in a hierarchy with *icl*, *pof*, *equ*, *iof* relations which correspond to subclass, part-of, equivalence and instance relations respectively. The top level of the UW hierarchy is shown in Figure 2.2. **Universal Word** is the root of the hierarchy, which corresponds to the universal entity. All the other UWs are categorized based on the syntactic category of the headword. The categories are adjective concept, adverbial concept, nominal concept, verbal concept.

The UW **thing** is the generic noun UW. It is classified into **abstract thing**, **concrete thing**, **place(icl>thing)**, **volitional thing**. The concept **abstract thing** consists of UWs for events, attributes, information etc. The concept **concrete thing** consists of UWs for concepts which will have physical presence. This includes all the physical objects, substances, animals, plants etc. The class **place(icl>thing)** consists of UWs for places, positional attributes and directional attributes. The class **volitional thing** consists of concepts like human, animal etc. The icl hierarchy of the concept for man is `man>human>living thing>concrete thing>thing`, where > indicates icl relation.

At higher level, verbs are classified into three broad categories- ‘do’, ‘occur’ and ‘be’ verbs. The ‘do’ verbs define the concept of an event that is caused by someone or something. The ‘occur’ verbs define concept of an event that occur on their own. The ‘be’ verbs define the state of an event or object. The verbs *eat*, *flow*, *like* belong to ‘do’, ‘occur’ and ‘be’ class of verbs respectively. The UW for a verb is defined based on the argument structure of the verb, and is included in

2.3 Representation

The UNL representation of a few English sentences that can be readily represented in logic is given in figures from Figure 2.3 to Figure 2.8. These expressions are generated manually by following the examples given in the UNL specification[Fou03] and the documents in UNL available at [Unl03]. The UWs are represented by rectangular boxes. UNL binary relations are represented by a labeled arrow; the name of the binary relation as the label, head of the arrow pointing at second argument of the relation.

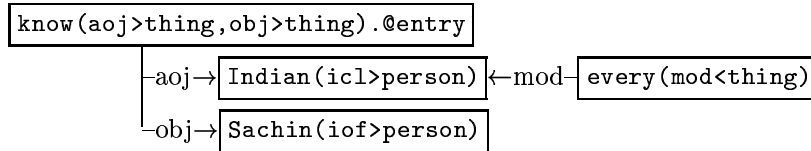


Figure 2.3: *Every Indian knows Sachin*

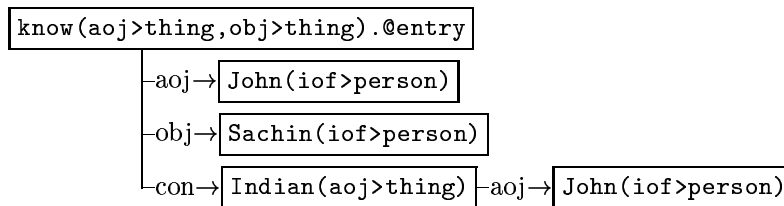


Figure 2.4: *If John is an Indian, then John knows Sachin*

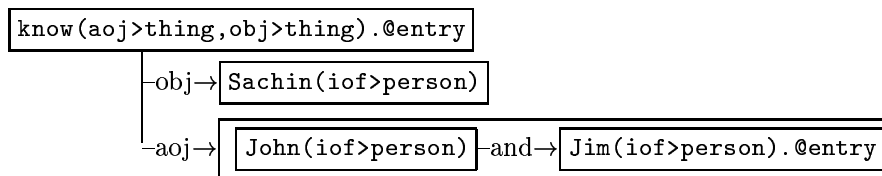


Figure 2.5: *John and Jim know Sachin*

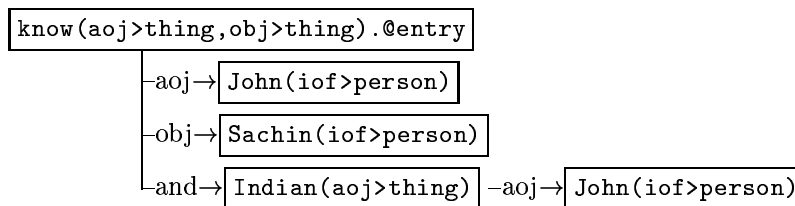


Figure 2.6: *John knows Sachin and John is an Indian*

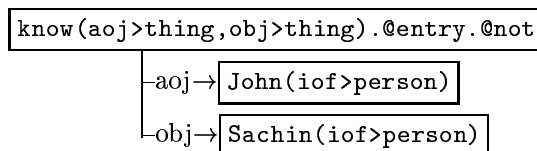


Figure 2.7: *John does not know Sachin*

Figure 2.3 is the UNL representation of *Every Indian knows Sachin*. The quantifier *every* is represented in UNL as `every(mod<thing)`. It is connected to the `Indian(icl<person)` through

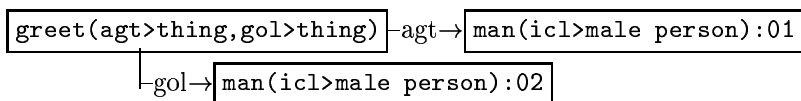


Figure 2.8: *Man kills man*

the modifier relation `mod`. The existential quantifier *some* is also represented similarly.

Figure 2.4 is the UNL representation of *If John is an Indian, John knows sachin*. The conditional in *if... then..* is represented by the relation `con`. The `con` relation relates an event or state with another event or state that acts as a condition. The conditioning event is the node at the tail of the edge.

Figure 2.5 is the UNL representation of *John and Jim know Sachin*. The connective *and* is represented as `and` relation between the UWs `John(iof>person)` and `Jim(iof>person)`. The meaning of the sentence is that both John and Jim know sachin. So a compound-UW is created. This compound-UW is connected to the UW `know(aoj>thing,obj>thing)`. Figure 2.6 is another example with *and* as connective. Here two clauses are connected with `and` relation. The representation of the connective *or* is similar.

UNL representation for *John does not know Sachin* is shown in Figure 2.7. The *negation* is represented as an attribute `@not` annotating the uw `know(aoj>thing,obj>thing)`.

2.4 Inferencing

Though UNL is intended as an interlingua for machine translation, the predicate like structure of UNL relations lends itself readily for question answering and inferencing tasks as well. Previous work in this direction[MRK⁺03] used subgraph matching for question answering; question is represented as a graph with the expected answer node as a variable. This query graph is matched against the UNL representation of a document for *retrieving* the sentences containing the query graph as subgraph. This is essentially *information extraction*, and not inferencing.

The subgraph matching approach is not suitable for logical inferencing. From the UNL representation of the short discourse “Every Indian knows Vajapaye. Sachin is an Indian” shown in 2.9, it can not find whether *Sachin knows Vajapaye* or not, as the relationship between these UWs is not present explicitly in the UNL representation.

```

[S:001]
mod( Indian(icl>person) , every(mod<thing) )
aoj( know(aoj>thing,obj>thing)@entry, Indian(icl>person) )
obj( know(aoj>thing,obj>thing)@entry, Vajapaye(iof>person) )
[/S]
[S:002]
sachin(iof>Indian)
[/S]
  
```

Figure 2.9: UNL representation of *Every Indian knows Vajapaye. Sachin is an Indian*.

The graph matching approach treats the UWs of quantifiers- `every(mod<thing)`, `all(mod<thing)` and `any(mod<thing)`- just like other UWs. These UWs should be handled separately. For this, the notion of quantifier is needed.

To introduce quantifiers- the concept of instantiation, variable or concept are also needed. The notion of a variable is implicitly present in the UNL. A variable can be associated with each unique UW present in the UNL representation of a sentence. Multiple nodes with the same UW are distinguished by a unique node identifier. The sentence “Man greets man” is represented in UNL as in Figure 2.8. The UW `man(icl>male person)` is used twice, to distinguish the man who greets and the receiver.

Giving a unique identifier for a UW node is only a notational change. Associating a variable with the UW node actually means *creating an instance of that UW*. But, can all the UWs be instantiated? `India(iof>country>region, iof>nation>society)` is a constant, `member(icl> person>human, pof>group)` is a relation, `aggressive(icl>uw(aoj>thing))` is also a constant.

The question of which concepts can be instantiated is an unresolved one, involving philosophical concerns and debates. Without going deeper into the details, principles from an existing ontology called Suggested Upper Merged Ontology can be adapted for our purpose. This requires us to find the closely related concept in SUMO for each UW.

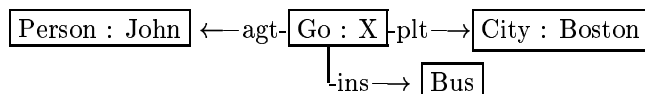
2.5 Conceptual Graphs

The nodes in the semantic network created by UNL can be transformed to include a quantifier, a variable or a constant. These kind of semantic networks are called *Conceptual graphs*[Sow83]. Conceptual graphs can be translated to equivalent predicate formulas.

Conceptual graphs are knowledge representation formalism designed to have a smooth translation from natural language and predicate logic. The formalism is a subclass of semantic networks and it is a more readable representation of predicate logic.

Conceptual graph is a bipartite, finite and directed graph of *concept nodes* and *relation nodes*. In the graphs, concept nodes represent classes of individuals, and relation nodes show how the concept nodes are related.

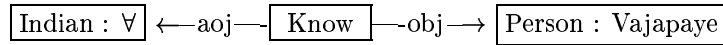
The conceptual graph representation of “John goes to Boston by bus” is



The rectangular boxes are called *concept nodes*. Each concept node in a CG consists of type and referent part. The referent part can be a variable, quantifier, a constant or another conceptual graph. `[Person:John]`, `[Go:X]`, `[City:Boston]` are concept nodes. `Person`, `Go`, `City` are the type part of the nodes. `John`, `X`, `Boston` are the referent part of the concept nodes respectively. If the referent part of the concept node is omitted, it takes *existential quantifier* by default. The concept node `[Bus]` assumes existential quantifier by default. The relation nodes are represented as labeled arcs between the concept nodes. `agt`, `plt`, `ins` are the relation nodes in the above. The direction of the arrow on the arc determines the argument position of the incident concept nodes. The concept node at the head of the arc will be the first argument

of the relation and the concept node at the tail of the arc fills the second argument position. For the relation **agt**, the concept nodes [Go:X] and [Person:John] are the first and second arguments respectively.

The conceptual graph representation for “Every Indian knows sachin” is



The quantifier *every* is shown as \forall in the referent part of the concept node [Indian: \forall].

Conceptual graph representation can be translated to formulas in logic. The method is given [AO00][F.S93]. Roughly, the method is

1. For a concept node C, let $C(x)$ be the predicate.
2. For a relation *Rel* in the conceptual graph with C_1, C_2, \dots as its argument nodes,
 - If the quantifier of all its argument nodes is \exists , then the logical representation of the relations is $Rel(x_1, x_2, \dots)$
 - If the quantifier of a node C_i is \forall , $C_i(x_i) \rightarrow Rel(x_1, \dots, x_i, \dots)$ is the logical representation of the relation.
3. The formula for the conceptual graph is conjunction of the predicates of the concept nodes and relations, prefixed with universal quantifier followed by existential quantifier.

For the conceptual graph of the sentence “John goes to Boston by bus”, the logical representation is

$\exists x, y \text{ person}(\text{John}) \wedge \text{city}(\text{Boston}) \wedge \text{Bus}(x) \wedge \text{Go}(y) \wedge \text{agt}(y, \text{John}) \wedge \text{plt}(y, \text{Boston}) \wedge \text{ins}(y, x)$.

For the conceptual graph of “Every Indian knows Vajapaye”, the logical representation is $\forall x \exists y \text{ Indian}(x) \wedge \text{person}(\text{Vajapaye}) \wedge \text{know}(y) \wedge (\text{Indian}(x) \rightarrow \text{aoj}(y, x)) \wedge \text{obj}(y, \text{Vajapaye})$.

Chapter 3

Suggested Upper Merged Ontology

According to Tom Gruber- “An Ontology is an explicit specification of a conceptualization”. An ontology consists of names of classes and relations among those classes. Based on the motivation for their creation, the ontologies are categorized as *logic-based*, *language-based* ontologies. Ontologies can also be labeled as upper ontology, domain-specific ontology. Domain-specific ontology defines concepts, relations for a particular domain. *Upper* ontology consists of concepts that are meta, generic and philosophical that can address a broad range of domain areas. IEEE is working towards developing a Standard Upper Ontology. Suggested Upper Merged Ontology[NP01][SUM03] is one of the starter documents submitted for the Standard Upper Ontology Working group[SUO04]. It is a formal, logic-based ontology.

3.1 SUMO

Suggested Upper Merged Ontology is an upper ontology proposed as a starter document for the development of Standard Upper Ontology. SUMO defines and organizes the abstract view of the world in a domain-independent and application-independent way. Domain-specific ontologies are built on top of SUMO.

SUMO consists of concepts, relations and axioms that constrain the meaning of the concepts and relations. The top level concepts are shown in 3.1. *Entity* is the universal concept. *Physical* and *Abstract* are the two disjoint subclasses of *Entity*. Events, situations and objects come under the *Physical* class. Attributes, functions, relations, numbers and other mathematical entities come under *Abstract* category. *Physical* category has events, objects that actually occur or exist at some point of time. *Process*, *Object* are its subcategories. *Object* roughly corresponds to our intuitive notion of object. It includes geographical regions, organism and other tangible entities . *Process* consists of entities that exist in time but are not objects. All events and situations belong to this category.

Abstract category consists of entities which do not exist in space or time. Entities belonging to this category are usually associated with some physical entity. Attributes, functions, relations, numbers belong to this category.

```

Entity
|=>Physical
|   |=>Process
|   |   |=>DualObjectProcess
|   |   |=>InternalChange
|   |   |=>ShapeChange
|   |   |=>IntentionalProcess
|   |   |   |=>Guiding
|   |   |   |=>ContentDevelopment
|   |   |=>Motion
|   |=>Object
|       |=>Collection
|       |=>Agent
|       |   |=>Group
|       |   |=>Organism
|       |=>SelfConnectedObject
|       |   |=>Food
|       |   |=>Substance
|       |=>Region
|       |   |=>GeographicalArea
|       |   |=>SpaceRegion
|=>Abstract
|   |=>Proposition
|   |   |=>Procedure
|   |=>Relation
|   |=>Quantity
|   |   |=>Number
|   |   |=>PhysicalQuantity
|   |=>Attribute
|   |   |=>InternalAttribute
|   |   |=>RelationalAttribute
|   |=>SetOrClass

```

Figure 3.1: The top level of SUMO concept hierarchy.

SUMO has a rich set of relations and functions. They can be grouped into temporal relations, spatial relations, case-relations and other mathematical relations. Relations *before*, *after*, *starts*, *finishes* etc define relationship between temporal objects. Relations *agent*, *patient*, *destination*, *instrument* etc denote the role played by the entity in an event. Relations *connects*, *between*, *located* etc denote spatial relation between entities.

Axioms are well-formed formulas in predicate calculus. Axioms provide additional constraints on the meaning of the concepts and relations. Figure 3.2 has an axiom for concept `Murder`. `?MURDER`, `?PERSON` are variables which are instances of type `Murder`, `Human` respectively. The meaning of the axiom is- if an instance of `murder` took place, there will be an instance of `Human` who is the object of `murder` i.e who is murdered.


```
(=>
  (instance ?MURDER Murder)
  (exists (?PERSON)
    (and (patient ?MURDER ?PERSON)
         (instance ?PERSON Human))))
```

Figure 3.2: An axiom in SUMO.

The Vampire inference system [VR] is used for inferencing on the SUMO ontology. It is a resolution-based system for automatic theorem proving in first order logic with equality. The system is able to provide inference using first order logic formulas. But it can't handle mathematical and temporal reasoning.

A few ontology modules are developed based on SUMO. Middle Level Ontology a.k.a MILO is developed as a bridge between the abstract content of the SUMO ontology to more domain-specific ontologies. Other modules available from [SUM03] are an ontology of geography, government and transportation.

3.2 Relation with WordNet

Ian Niles et al[Nil03] linked the synsets from WordNet1.6 to the corresponding concept in the ontology. The entries in the WordNet database are augmented with the related SUMO concept. The synset corresponding to the verb **breathe** is linked to the SUMO concept **Breathing**. The SUMO concept is the last entry in the synset record. The = symbol followed by the concept name is an indicator that the synset is mapped to a concept with the same meaning. The augmented WordNet record is shown below.

```
29 v 03 breathe 0 take_a_breath 0 respire 0 013 * 00003763 v 0000 * 00003142 v
0000 ^ 00003142 v 0103 ^ 00003763 v 0102 ~ 00002143 v 0000 ~ 00002343 v 0000 ~
00002841 v 0000 ~ 00003011 v 0000 ~ 00003142 v 0000 ~ 00003763 v 0000 ~ 00004868
v 0000 ~ 00005197 v 0000~ 00011570 v 0000 02 + 02 00 + 08 00 | draw air into,
and expel out of, the lungs; "I can breathe better when the air is clean"
&%Breathing=
```

Since SUMO is an upper ontology, every synset may not be linked to the concept with the same meaning. Some synsets are mapped to a subclass of a SUMO concept when there are slight difference in meaning of the synset and the concept. For example, synset corresponding to "choke"-*breathing with difficulty* is mapped to a subclass of **Breathing**, represented in the synset record as **Breathing+**.

```
29 v 01 choke 0 001 @ 00001740 v 0000 01 + 02 00 | breathe with great difficulty,
&%Breathing+
```

Synsets corresponding to individuals, places, countries etc are defined as an instance of a SUMO concept. For example, synset of **Gandhi** is defined as an instance of **Human** and represented in the synset record as **Human@**

18 n 03 Gandhi 0 Mahatma_Gandhi 0 002 @ 07524377 n 0000 @ 06861412 n 0000 |
(1869-1948) political and spiritual leader during India's struggle with Great
Britain for home rule; an advocate of passive resistance &%Human@

This relationship between WordNet synsets and SUMO concepts is exploited for mapping the
universal words to the corresponding SUMO concepts.

Chapter 4

Mapping

Each Universal Word should be linked to the SUMO concept with closest meaning, so that concepts from the SUMO ontology can be used as the representation of the UW in logic. This is akin to mapping ontologies i.e. finding concepts with similar meaning from two different ontologies. The lexical and structural similarities of the concepts are used in [NM00] for deciding the mapping.

For mapping UW to SUMO concept, the relation between WordNet and SUMO is used along with the lexical and structural comparison. First, a synset in WordNet for the UW is selected from the different senses of the head-word of the UW. Next, the SUMO concept associated with the synset is taken for mapping the UW. The method for finding the synset for the UW is different for nouns, verbs, adjectives and adverbs. Subsequent sections describe these methods.

4.1 Noun UWs

Algorithm 1 gives the algorithm for mapping a noun UW. The WordNet synset for the UW is selected from the synsets of the head-word of that UW by comparing the *icl* and *pof* hierarchy of the UW with the *hypernyms* and *holonyms* of the sense. Out of 4400 noun UWs present in UNL knowledgebase, 3000 noun UWs are mapped in this way. Remaining UWs are mapped manually. A few UWs along with the selected synset and SUMO concept are listed below.

Universal Word	Sense selected	SUMO concept
<code>beast(icl>animal{>living thing})</code>	<code>beast=>organism=>living thing</code>	<code>Animal</code>
<code>step(icl>movement>action)</code>	<code>step=>travel=>movement=>..</code>	<code>subclass Walking</code>
<code>fin(icl>concrete thing, pof>fish)</code>	<code>part of: fish</code>	<code>subclass Organ</code>

The head-word ‘beast’ of the UW `beast(icl>animal{living thing})` has 2 senses in WordNet. The hypernyms of the selected sense have the word `living thing` in common with the *icl* hierarchy of the UW. The head-word `fin` of the UW `fin(icl>concrete thing, pof>fish)` has 6 senses in WordNet. The holonyms of the selected synset have `fish` in common with the *pof* relations of the UW. The SUMO concept related to this UW is a subclass of `Organ` i.e the concept is not present in SUMO and have to be added as a subclass of `Organ`.

Algorithm 1 Mapping Noun UWs

Input: Universal Word(*uw*)**Output:** SUMO Concept(*concept*)

```
hw ← headword(uw)
icl ← parentsof(uw); pof ← isPartOf(uw)
syns ← synsets(hw)
for all S ∈ syns, while related synset not found do
  hypernyms ← hypernymsof(S)
  holonyms ← holonymsof(S)
  if ((icl ∧ hypernyms) ≠ {}) then
    S is the matching synset
  else if ((pof ∧ holonyms) ≠ {}) then
    S is the matching synset
  end if
end for
if S is the matching synset for UW then
  concept ← sumoConcept(S)
end if
```

4.2 Verb UWs

Algorithm 2 gives the algorithm for mapping verb UWs. This similar to the algorithm used for mapping nouns except that only hypernymy relation is used as holonyms are not available for verbs. Only 390 verb UWs are mapped this way, out of 1730 verb UWs present in the UNL knowledgebase. Remaining verb UWs are mapped manually. A few examples where the mapping is done by the algorithm are listed below.

Universal Word	Selected sense	SUMO Concept
<i>sigh</i> (<i>icl</i> > <i>breathe</i> (<i>agt</i> > <i>volitional thing</i>))	<i>sigh</i> => <i>breathe</i>	subclass Breathing
<i>shift</i> (<i>icl</i> > <i>move</i> (<i>agt</i> > <i>thing</i> , <i>gol</i> > <i>place</i>))	<i>shift</i> => <i>move</i>	Translocation
<i>rule</i> (<i>icl</i> > <i>decide</i> (<i>agt</i> > <i>thing</i> , <i>obj</i> > <i>thing</i>))	<i>rule</i> => <i>decide</i>	subclass Ordering
<i>settle</i> (<i>icl</i> > <i>resolve</i> (<i>agt</i> > <i>thing</i> , <i>obj</i> > <i>thing</i>))	<i>settle</i> => <i>resolve</i>	subclass Communication
<i>recommend</i> (<i>icl</i> > <i>propose</i> (<i>agt</i> > <i>thing</i> , <i>obj</i> > <i>thing</i>))	<i>recommend</i> => <i>propose</i>	subclass Communication

Only 390 verb UWs are mapped this way. The subclass hierarchy of verbs in UNL knowledgebase is very flat i.e most verb UWs are direct subclasses of **be**, **do**, **occur**. Hence, the *icl* list of the UW do not have anything in common with the hypernym list of the head-word of the UW. When the subclass hierarchy of the UW is more deeper, the WordNet synset for that UW is selected correctly. A few such cases are listed above.

Algorithm 2 Mapping Verb UWs

Input: Universal Word(*uw*)**Output:** SUMO Concept(*concept*)

```
hw ← headword(uw)
icl ← parentsof(uw)
syns ← synsets(hw)
for all S ∈ syns, while related synset not found do
  hypernyms ← hypernymsof(S)
  if ((icl ∧ hypernyms) ≠ {}) then
    S is the matching synset
  end if
end for
if S is the matching synset for UW then
  concept ← sumoConcept(S)
end if
```

4.3 Adjectives and Adverbs

Adjectives and adverbs are arranged in a subclass hierarchy in UNL knowledgebase. But WordNet do not have hypernymy relation for adjectives and adverbs. *icl* information can not be used for finding the sense of the headword. Algorithm 3 is used for mapping adjectives and adverbs.

Algorithm 3 Mapping adjectives and adverbs

Input: Universal Word(*uw*)**Output:** SUMO concept (*conc*)

```
headword ← headword(uw)
syns ← synsets(headword)
choices ← []; {Initializing choices to empty list}
for all S ∈ syns do
  c ← sumoConcept(S)
  add(c, choices)
end for
select the concept conc which appears maximum number of times in choices
```

All the three senses of **adequate** are mapped to **SubjectiveAssessmentAttribute**. So the UW **adequate(icl>(uw(aoj>thing))** is mapped to **SubjectiveAssessmentAttribute**. In fact, many of the adjectives and adverbs are mapped to the same concept **SubjectiveAssessmentAttribute**. This is because the representation of adjectives and adverbs in SUMO is very coarse.

4.4 Relations

The UNL relations define the relationship between the entities present in a sentence. These relations can be roughly classified into case-relations and spatio-temporal relations. The case-relations indicate the role played by an entity in the event. The relations `agt`, `obj`, `ins`, `gol` etc belong to this category. The spatio-temporal relations relate the entities, events and states with the time of their occurrence and the place where they occurred. The relations `dur`, `tim`, `tmf`, `tmt`, `plc`, `plf`, `plt` etc belong to this category.

Many of the case relations of UNL are available in SUMO as well. Relations which are not present in SUMO can be defined as a composition of relations present in SUMO. The following table lists some of the relations and their corresponding SUMO relation.

UNL Relation	Meaning	SUMO Relation
<code>agt(do, thing)</code>	a <i>thing</i> that initiates an <i>action</i>	<code>agent(Process, Agent)</code>
<code>aoj(uw(aoj>thing), thing)</code>	defines a <i>thing</i> that is in a state or has an <i>attribute</i>	<code>property(Entity, Attribute)</code>
<code>ben(do, thing)</code>	an indirectly related beneficiary or victim of an <i>event</i>	<code>destination(Process, Entity)</code>
<code>obj(do, thing)</code>	defines a <i>thing</i> that is directly affected by an <i>event</i>	<code>patient(Process, Entity)</code>
<code>ins(do, concrete thing)</code>	defines an <i>instrument</i> to carry out an <i>event</i>	<code>instrument(Process, Object)</code>
<code>cnt(thing, thing)</code>	defines an equivalent concept	<code>equal(Entity, Entity)</code>
<code>pur(do, do)</code>	the purpose of an agent of an event	<code>hasPurpose(Physical, Sentence)</code>
<code>coo(do, do)</code>	defines a co-occurring event for a focused event	<code>cooccur(Physical, Physical)</code>
<code>rsn (do, do)</code>	defines a reason why an <i>event</i> happens	<code>causes(Process, Process)</code>

Table 4.1: UNL relations and the corresponding SUMO relations.

The spatio-temporal relations of UNL didn't have an exactly matching relation in SUMO. But they can be defined with the help of the relations present in SUMO. The functions *whereFn*, *whenFn* and other temporal relations of SUMO are used for defining the UNL relations. A description of these relations follows.

whenFn(Entity) This function gives the time interval in which the entity exists.

whereFn(Entity, Time Interval) This function gives the place in which entity is present during that time interval.

beginFn(Time Interval) gives the beginning of a time interval.

endFn(Time Interval) gives the end of a time interval.

Using the above SUMO relations, some of the spatio-temporal relationships of UNL can be defined as follows:

plc(do, thing) (equal Thing (WhereFn do (WhenFn do)))

Which means, the place *where* the action *do* took place is equal to the entity denoted by *Thing*.

plf(do, thing) (equal Thing (WhereFn do (BeginFn (WhenFn do))))

Which means, the place *where* the action *do* began is equivalent to *Thing*. The *plf* relation is defined in the same way.

tim(do, time) (equal time (WhenFn do))

which means, the time *when* the action *do* took place is equal to *time*.

tmf(do, time) (equal time (BeginFn (WhenFn do)))

The time *when* the action *do* begins is equal to *time*. The *tmt* relation is defined similarly.

The relations *and*, *or* are mapped to the logical *and*, *or* respectively. But the meaning of these relations in natural language is not limited to the logical operations.

Of all the attributes, only the attributes that represent time from the speaker's point of view are mapped to SUMO. Other attributes for representing the speech acts, propositional attitudes of the speaker etc are not mapped as SUMO do not have concepts for representing the speech acts, propositional attitudes. The attributes of time and the corresponding SUMO functions are shown in 4.2

Attribute	Definition	SUMO Function
@past	Something happened in the past	pastFn
@future	Something will happen in future	futureFn
@begin	Beginning of an event or a state	beginFn

Table 4.2: Attributes of time and the related SUMO functions

Chapter 5

Translation to Logic

Before translating the UNL graph to logic formulas, some *restructuring* of the arguments is done for the relations **con**, **and**, **or**. This introduces the connectives \Rightarrow , \vee , \wedge . Next, a logical representation is associated with the nodes of the UNL graph using the concepts, relations from SUMO. Then, the whole graph is translated to logic representation using the equivalence of *Discourse Representation Structures* and *First order logic* given in [KR]. *Quantifiers* are added to the logical expression in this stage.

5.1 Restructuring

This phase introduces the connectives \Rightarrow , \vee , \wedge , \neg into the network representation of the UNL expression.

5.1.1 Implication

Implication is introduced in two cases- when the expression contains a universal quantifier like **all(mod<thing)**, **every(mod<thing)**, or when the expression has **con** relation.

Sentences with a universal quantifier have a logical representation containing an \Rightarrow . The logical expression associated with “Every Indian knows Vajapaye” is

$$\forall x(\text{Indian}(x) \Rightarrow \text{knows}(x, \text{Vajapaye})).$$

Same intuition is followed while translating the UNL representation to logical expression. An \Rightarrow will be introduced following the Conceptual Graph method. For the UNL graph G,

1. From G, remove the quantifier node and the relation linking it to the quantified node.
2. Antecedent of the implication is the quantified node
3. Consequent of the implication is the Graph G.

The sentence “Every Indian knows Vajapaye” is represented in UNL is in Figure 5.1. The corresponding restructured graph is in Figure 5.2. The nodes **every(mod<thing)**, **Indian(ic1>person)** are the quantifying node, quantified node respectively. The node **every(mod>thing)** and the

relation `mod` is removed from the initial semantic network. This semantic network is the *consequent* part of the \Rightarrow . The quantified node `Indian(icl>person)` will be the *antecedent*. The antecedent and consequent, thus formed, are the two nodes of the semantic network in Figure 5.2 with \Rightarrow as the only edge.

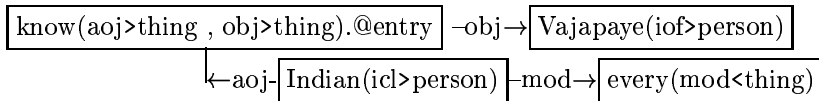


Figure 5.1: UNL representation for “Every Indian knows Vajapaye”

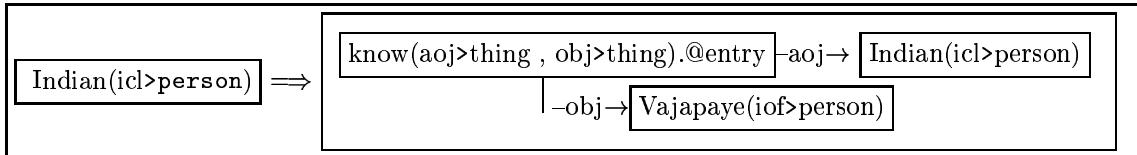


Figure 5.2: Introducing implication in UNL expression of Figure 5.1

Sentences with conditional clauses also have an \Rightarrow in their logical representation. The logical expression associated with the sentence “If John owns a donkey, he beats it” contains an implication. The expression is $\forall x(\text{donkey}(x) \vee \text{owns}(x, \text{John}) \Rightarrow \text{beats}(\text{John}, x))$. The conditional aspect of this kind of sentences is represented in UNL using `con` relation. The relation `con(uw1, uw2)` means `uw2` acts as a condition for the occurrence of the event or state expressed by `uw1`. The UNL representation of the above sentence Figure 5.3. This is slightly modified in order to avoid anaphora. The actual expression will have the pronouns `I(icl>person)` and `it(icl>thing)` as the `agt`, `obj` of the node `beat(agt>thing, obj>thing)`.

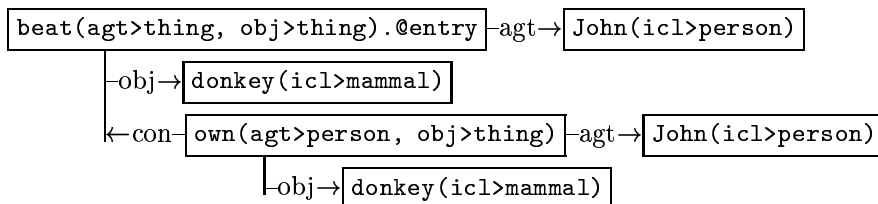


Figure 5.3: UNL representation for “If John owns a donkey, he beats It”.

The two arguments of the `con` relation are not the actual antecedent and consequent of an implication. As evident from the above example, the argument UWs are only part of the antecedent and consequent. The actual antecedent, consequent are the subgraphs formed by removing the `con` relation from the network. For the network `G` containing the edge `con(UW1, UW2)`, the antecedent and consequent are obtained as follows.

1. Remove the edge `con(UW1, UW2)`.
2. If `con` is the only path between `UW1` and `UW2`, step 1 creates two disjoint graphs `G1`, `G2`. `G1` contains `UW1` along with all its relations. `G2` contains `UW2` along with all its relations.

3. Otherwise, create graph G1 with only UW1 initially. Add each node N in G, reachable from UW1 if the path from UW1 to N does not contain UW2. Add edge E(N1,N2) to G1 if nodes N1 and N2 are present in G1 and edge E(N1,N2) is present in G. Similarly, create graph G2 with UW2 initially and expand G2 similar to G1.
4. Create a new graph with nodes G1 and G2, and an \Rightarrow relation from node G2 to node G1.

For the UNL expression shown in Figure 5.3, the restructured graph is shown in 5.4. The con relation between `beat(agt>thing, obj>thing)` and `own(agt>thing, obj>thing)` is restructured as an implication. The nodes `John(icl>person)` and `donkey(icl>mammal)` are reachable from both `own(agt>thing, obj>thing)` and `beat(agt>thing, obj>thing)`. So both these concepts are added to the consequent graph as well as the antecedent graph.

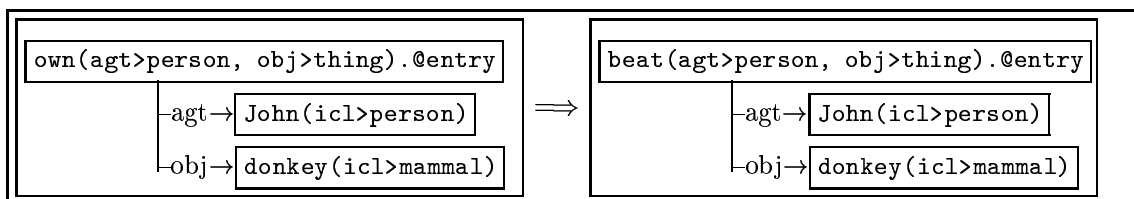


Figure 5.4: Introducing \Rightarrow in UNL representation in Figure 5.3.

5.1.2 Conjunction and Disjunction

The relations `and`, `or` represent the logical aspects of sentences involving conjunction and disjunction. These relations are used in two cases; when the sentence consists of two clauses joined by *and* or *or* as in the sentence “John is an Indian and John knows Sachin”; for sentences like “John and Jim know Sachin” to represent the conjunctive participation of both John and Jim in *know*. The UNL representation for this sentence is in 5.5. The corresponding restructured graph is in Figure 5.6.

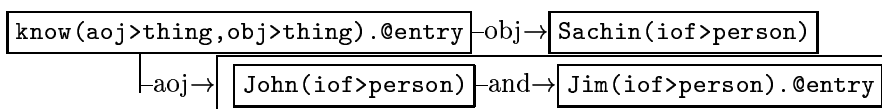


Figure 5.5: UNL representation of *John and Jim know Sachin*

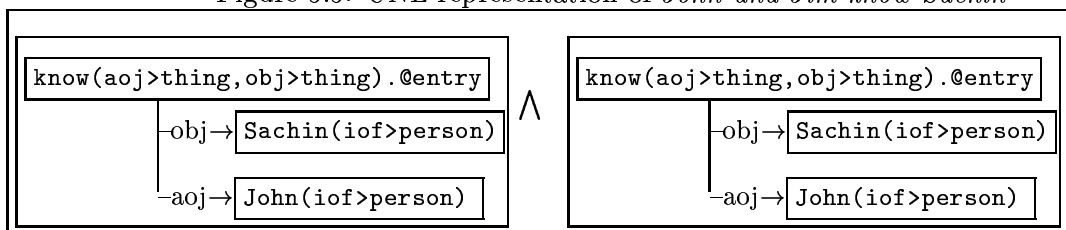


Figure 5.6: Restructured UNL graph.

Restructuring in this case is done by distributing the clauses of the compound UW to the main UNL graph. The compound UW is treated like a UNL graph, and the restructuring is

applied to that compound UW. This gives the clauses of the compound UW. The entry nodes of these clauses are joined with the nodes to which the compound-UW is initially related. The resulting graph for the sentence "John and Jim know Sachin" is shown in 5.6. The relation `or` is handled similarly.

5.1.3 Negation

Negation is represented in UNL as an attribute `@not`. The UNL graph for "John do not like Sachin" is shown in 5.7. The attribute `@not` is used while translating the graph to logic.

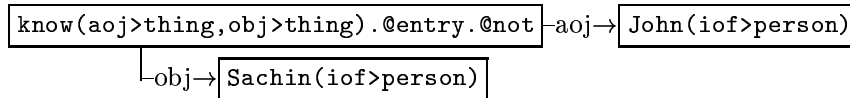
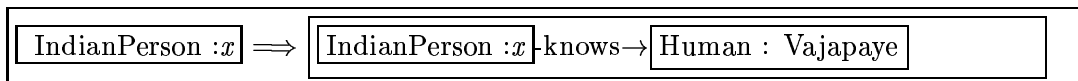


Figure 5.7: *John does not know sachin*

5.2 Using SUMO

Before translating the the UNL graph to logical expression, a logical expression has to be associated with each node in the graph. The logical expression for a UW node consists of a *type* and a *referent*. The referent is a variable or a constant. Initially, the type of the node is the associated *uw* and the referent is a variable. The actual logical expression of a UW node is decided based on the SUMO concept to which the *uw* of the node is mapped. The SUMO equivalents for the UNL relations are given in Chapter 4.0. For a node N with type *uw*, the logical expression is decided as below.

- If *uw* is mapped to a SUMO constant *C* of type *T*, the type and referent of the node N is changed to *T* and *C* respectively.
- If *uw* is mapped to a SUMO concept *T*, only the type of the node N is changed to *T*.
- If *uw* is mapped to a SUMO relation *R* with a replacing pattern graph *P*, replace the relations around *uw* that are in the pattern *P*, with the relation *R*.



The new representation of the graph in Figure 5.2 is shown above. The UW `Vajapaye(iof>person)` is mapped to the constant `Vajapaye` of type `Human`. For a node with UW `Vajapaye(iof>person)`, new type and referent will be `John` and `Human` respectively. The UW `know(aoj>thing,obj>thing)` is mapped to a relation `knows` with a pattern containing relations `aoj`, `obj`. So, this pattern around the UW is replaced with the relation `knows`.

5.3 Quantification and Translation

The algorithm for translating the restructured UNL expression to logical expression, and the method for associating quantifiers with variables are taken from Discourse Representation Theory [KR].

Similar to discourse representation structures, the restructured UNL graph will have clauses of the following structure.

- Simple graph containing only non-logical relations of UNL or SUMO.
- Two graphs connected with an implication.
- Two graphs connected with a disjunction.
- Two graphs connected with a conjunction.
- A graph containing a node with attribute @not.

The algorithm for translating the simple graph containing only non-logical relations of SUMO is given in Algorithm 4. The translation for the other categories is defined based on this algorithm. Input for the algorithm is the UNL graph, list of variables from an enclosing scope. Output is a logic formula and list of free variables in the formula. The logic representation of the input graph is the conjunction of the logical representation of the nodes, relations in it. The functions *type*, *referent* give the type and referent part of a node. The functions *head*, *tail* will give the originating, destination node of an edge.

Algorithm 4 Translating a simple graph to logical expression

Input: Graph $G(\text{Nodes}, \text{Relations})$, Enclosing Scope= $\{x_1, x_2.. \}$

Output: Formula, Referents .

```

Referents  $\leftarrow \{ \}$ ; ConstraintList  $\leftarrow \{ \}$ 
for all Node  $\in$  NodeSet do
  Node  $\leftarrow$  NodeSet.nextElement()
  Type  $\leftarrow$  type(Node); Var  $\leftarrow$  referent(Node)
  if Var not in EnclosingScope then
    Con  $\leftarrow$  instance(Type, Var)
    add Con to ConstraintList
  end if
end for
for all R  $\in$  Relations do
  rel  $\leftarrow$  label(R)
  Source  $\leftarrow$  head(R); x1  $\leftarrow$  referent(Source)
  Target  $\leftarrow$  tail(R); x2  $\leftarrow$  referent(Target)
  Con  $\leftarrow$  rel(x1, x2)
  add Con to ConstraintList
end for
Formula  $\leftarrow$  the conjunction of all constraints in ConstraintList
Referents is the list of variables present in the graph G.

```

The algorithm for translating graph containing $\Rightarrow, \vee, \wedge$ is given in 5. *Scope* of a graph is the list of variables created and used in the nodes of that graph. *Enclosing Scope* is the variables used

in the nodes of the enclosing scope. The antecedent of an \Rightarrow is the enclosing scope for the consequent. The input to the algorithm is the restructured UNL graph $G(Nodes, Relations)$, variables in the enclosing scope of the graph G . The function $translate(G, EnclosingScope, Scope)$ returns the equivalent formula for the graph G , and the list of free variables in the graph is returned in $Scope$. The function $translate(G, EnclosingScope, Scope)$ invokes the above algorithm for translating simple graphs to formula.

Algorithm 5 Translating to Logical Expressions.

Input: Graph $G(Nodes, Relations)$, EnclosingScope= $\{x_1 \dots\}$

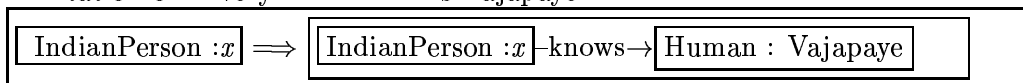
Output: Formula , Scope

```

if  $G$  do not contain logical connectives then
  Con  $\leftarrow$  translateSG( $G, EnclosingScope, Scope$ )
else if A node  $N$  in graph  $G$  has @not attribute then
  Con  $\leftarrow$  translate( $G, EnclosingScope, Scope$ )
  associate  $\exists$  with the free variables  $x_i$  in  $Con$ 
  Formula  $\leftarrow$  (not ( $\exists x_1, x_2..Con$ ))
end if
for all  $R$  in Relations do
  if  $R$  is of type  $A \Rightarrow B$  then
    Antecedent  $\leftarrow$  translate( $A, EnclosingScope, Scope$ )
    Consequent  $\leftarrow$  translate( $B, Scope, ScopeOut$ )
    if  $x \in$  Scope of Antecedent, associate a  $\forall$  with it.
    if  $y \in$  Scope of Consequent, associate a  $\exists$  with it.
    add condition of the form  $\forall x_1, x_2..(Antecedent \Rightarrow (\exists y_1, y_2.. Consequent));$ 
  else if  $R$  is of type  $A \vee B$  then
    Con1  $\leftarrow$  translate( $A, EnclosingScope, Scope$ )
    Con2  $\leftarrow$  translate( $B, Scope, ScopeOut$ )
    associate  $\exists$  with free variables  $x_i$  in  $Con1, Con2$ .
    add condition of the form  $\exists x_1, x_2..(Con1 \vee Con2)$ .
  else if  $R$  is of type  $A \wedge B$  then
    Con1  $\leftarrow$  translate( $A, EnclosingScope, Scope$ )
    Con2  $\leftarrow$  translate( $B, Scope, ScopeOut$ )
    associate  $\exists$  with free variables  $x_i$  in  $Con1, Con2$ .
    add condition of the form  $\exists x_1, x_2..(Con1 \vee Con2)$ .
  end if
end for
Formula of the graph is the conjunction of all the conditions accumulated above.

```

The quantifier \forall is associated with the free variables in the antecedent of an implication. \exists is the quantifier for the remaining free variables. The logical representation for the restructured representation of “Every Indian knows Vajapaye” in



is $\forall x(\text{IndianPerson}(x) \Rightarrow (\text{Human}(\text{Vajapaye}) \wedge \text{knows}(x, \text{Vajapaye}))$. The corresponding representation produced by the system is shown below. This is in SUO-KIF language which is used in creating the SUMO ontology. Variables are represented by a string prefixed with ‘?’ as in ‘?X’.

```
(forall (?X)
  (= > (instance ?X IndianPerson)
    (and (instance Vajapaye Human)
      (knows ?X Vajapaye)
    )))
```

The consequent part of the implication $\boxed{\text{IndianPerson} : x} \text{--knows--} \boxed{\text{Human:Vajapaye}}$ is in the scope of the antecedent graph $\boxed{\text{IndianPerson:x}}$. The quantifier \forall is associated with the variable x as described in the algorithm 5. The quantifier \exists would be given to any free variable in the consequent part of the implication.

Now that we arrived at the logical representation for a single sentence, add the logical representation of the UNL graph for “Sachin is an Indian” to the discourse. Logical reasoning is done by giving the resulting discourse consisting of the two formulas two a theorem prover. Since the syntax, concepts and relations from SUMO are used for the logical representation, the *vampire* inference system of SUMO is used for reasoning tasks. The discourse given to the theorem prover is

```
(forall (?X)
  (= > (instance ?X IndianPerson)
    (and (instance Vajapaye Human)
      (knows ?X Vajapaye))))
(instance Sachin IndianPerson)
```

For the query `(knows Sachin Vajapaye)`, the output of the inference system is an obvious *yes* with a long proof. This is only a two-sentence discourse. The UNL representation and the corresponding logical representation of Marcus problem is given in the Appendix A.

Chapter 6

Conclusions

The UNL expressions of natural language sentences- that can be readily represented in predicate logic- are translated to logic representation using concepts and relations from SUMO. Logical implication, conjunction and disjunction are introduced by restructuring the arguments of the UNL relations *con*, *and*, *or*. The method for introducing quantifiers is based on discourse representation theory. The method worked for a few simple sentences.

The UWs from the UNL knowledgebase are linked to the SUMO terms for obtaining the logical representation of the UWs. Concepts for nouns and verbs are represented well in SUMO. But the representation of adjectives and adverbs is very shallow.

The content of SUMO is readily available but the tools for inferencing and systems that use the ontology are not available earlier. This hampered the progress of the project in initial stages. The situation is changing now with the release of tools like CELT(Controlled English to Logic Translator).

Future Work

Controlled Language The approach is tested only for a few sentences. Increase the scope of the problem to include sentences from *controlled* languages. These *controlled* languages are designed such that they can be easily translated to logic formulas.

Conceptual Graphs Using conceptual graph theory and tools for reasoning directly on the UNL expressions without translating the UNL expressions to logic formulas, can be considered for further investigation.

Appendix A

Marcus Problem

This is a classical example for working with reasoning in predicate logic. The premises in the problem are

1. Marcus is a man.
2. Marcus is a Pompeian.
3. All Pompeians are romans.
4. Caesar is a ruler.
5. All Romans are either loyal to or hate Caesar.
6. If people try to assassinate a ruler, they are not loyal to the ruler.
7. Marcus tried to assassinate Caesar

From the above statements, we have to find whether “Marcus is loyal to Caesar or not”.

UNL representation

The UNL representation for the above sentences in the problem is

```
[S:001]
{org} Marcus is a man. {/org}
{unl}
aoj(Man(icl>male person),Marcus(iof>person).@entry)
{/unl}
[/S]
```

```
[S:002]
{org} Marcus is a pompein. {/org}
{unl}
aoj(pompein(aoj>thing), Marcus(iof>person).@entry)
{/unl}
[/S]
```



```
[S:002]
{org} All pompeins are romans. {/org}
{unl}
mod(pompein(icl>person).@entry, all(mod<thing>))
aoj(roman(aoj>thing), pompein(icl>person).@entry)
{/unl}
[/S]
```

```
[S:003]
{org} Caesar is a ruler. {/org}
{unl}
aoj(ruler(icl>status), Caesar(iof>person))
{/unl}
[/S]
```

```
[S:004]
{org} All Romans either loyal to or hate Caesar. {/org}
{unl}
aoj(:01.@entry, roman(icl>person))
or:01(loyal(aoj>volitional thing,obj>thing), hate(agt>volitional thing,obj>thing))
obj(:01.@entry, Caesar(iof>person))
mod(roman(icl>person), all(mod<thing>))
{/unl}
[/S]
```

```
[S:005]
{org} If people try to assassinate a ruler, they are not loyal to the ruler {/org}
{unl}
aoj(loyal(aoj>volitional thing,obj>thing).@entry.@not, people(icl>person))
obj(loyal(aoj>volitional thing,obj>thing).@entry.@not, ruler(icl>person))
con( loyal(aoj>volitional thing,obj>thing).@entry.@not, try(agt>thing,obj>thing).@entry)
agt(try(agt>thing,obj>thing):0S, people(icl>person))
pur(try(agt>thing,obj>thing):0S, assassinate(agt>thing,obj>thing):0B)
obj(assassinate(agt>thing,obj>thing):0B, ruler(icl>person) )
{/try}
[/S]
```

```
[S:006]
{org} Marcus tried to assassinate Caesar. {/org}
{unl}
agt(try(agt>thing,obj>thing):0T.@entry, Marcus(iof>person))
pur(try(agt>thing,obj>thing):0T.@entry, assassinate(agt>thing,obj>thing):0A)
obj(assassinate(agt>thing,obj>thing):0B, Caesar(iof>person) )
{/unl}
[/S]
```

SUMO mappings

Since the logical representation for the UW is given in terms of SUMO concepts, the SUMO concepts related to the UWs present in the above UNL expressions are listed below.

Marcus(iof>person)	[Human:Marcus]
Man(icl>human)	[SexAttribute:Male]
pompein(aoj>thing)	[EthnicGroup:Pompein]
pompein(icl>person)	[Human]-aoj→[EthnicGroup:Pompein]
roman(aoj>thing)	[EthnicGroup:Roman]
roman(icl>person)	[Human]-aoj→[EthnicGroup:Roman]
Caesar(iof>person)	[Human:Caesar]
ruler(icl>status)	[Position:Ruler]
ruler(icl>person)	[Human]-aoj→[Position:Ruler]
loyal(aoj>volitional thing,obj>thing) {aoj, obj}	loyal
hate(agt>volitional thing,obj>thing) {agt, obj}	dislikes
people(icl>person)	[Human]
try(agt>thing,obj>thing)	[Try]
assassinate(agt>thing,obj>thing)	[Killing]

The equivalent expression for the UWs is shown above. The nodes in the equivalent SUMO expression are enclosed in [] with type and referent separated by a colon. The equivalent expression for `loyal(aoj>volitional thing,obj>thing)` is a relation `loyal` obtained by replacing the the UW, the two relations `aoj`, `thing`. For `roman(icl>person)`, the equivalent expression is a graph with two nodes and a relation. Replace the UWs in the restructured graph with their equivalent expression from above.

Logical Representation

The logical representation produced for the UNL expressions are shown below. They are edited for better presentation.

```
;;Marcus is a man
(and (instance Marcus Human) (attribute Male Marcus))

;;Marcus is a pompein
(and (instance Marcus Human) (attribute Marcus Pompein))

;;All Pompeins are Romans.
(forall (?X1) (=) (and (instance ?X1 Human)
                      (attribute ?X1 Pompein))
            (attribute ?X1 Roman)))

;;Caesar is a ruler
(and (instance Caesar Human) (attribute Caesar Ruler))

;;All Romans are either loyal to or hate Caesar
```

```
(forall (?X3) (=> (and (instance ?X3 Human ) (attribute ?X3 Roman ))
                  (or (and (loyal ?X3 Caesar)(instance Caesar Human))
                      (and (hate ?X3 Caesar) (instance Caesar Human))))))
```

;;If People try to kill a ruler, they are not loyal to the ruler.

```
(forall (?X1 ?X2 ?T ?A ) (=> (and (instance ?X1 Human)
                                  (instance ?X2 Human) (attribute ?X2 Ruler)
                                  (instance ?T Try) (instance ?A Killing)
                                  (agent ?T ?X1)
                                  (hasPurpose ?T ?A)
                                  (patient ?A ?X2))
                              (not (loyal ?X1 ?X2))))
```

;; Marcus tried to assassinate Caesar

```
(exists (?X10 ?X11)
  (and (instance Marcus Human)
        (instance ?X10 Try)
        (instance ?X11 Killing)
        (instance Caesar Human)
        (agent ?X10 Marcus)
        (hasPurpose ?X10 ?X11)
        (patient ?X11 Caesar)))
```

Inference Engine

The logical expression produced shown above is loaded into the Vampire inference system. If the system arrives at a contradiction while proving a query, the response of the system is a simple “no”. If the query can be satisfied, the proof is produced. After loading the premises, queries are given to the inference system. The query should be in the SUO-KIF language. The question here is “Is Marcus is loyal to Caesar”. The query is (loyal Marcus Caesar). The response of the inference engine for this query as well its negation is shown below.

```
Query 1: Is Marcus Loyal to Caesar
<query> (loyal Marcus Caesar) </query>
<queryResponse>
  <answer result='no'> </answer>
  <summary proofs='0' />
</queryResponse>
```

```
Query 2: Is Marcus not loyal to Caesar
<query>(not (loyal Marcus Caesar)</query>
<queryResponse>
  <answer result='yes' number='1'>
    <proof>
      .....
      FALSE
    </proof> </answer>
  <summary proofs='1' />
</queryResponse>
```

Bibliography

- [AO00] Gianni Amati and Iadh Ounis. Conceptual graphs and first order logic. *The Computer Journal*, 43, 2000.
- [Fou03] UNDL Foundation. Universal networking language specifications, version 3 edition 2, July 1 2003. Retrived from <http://www.undl.org/unlsys/unl/UNLSpecifications.htm>.
- [F.S93] John F.Sowa. Relating diagrams to logic. In Bernard Moulim Guy W. Mineau and John F.Sowa, editors, *Conceptual Graphs for Knowledge Representation*, August 1993.
- [KR] Kamp and Reyle. *From Discourse to Logic: Introduction to Model-theoretic semantics of natural language, formal logic and Discourse representation theory*, volume 42 of *Studies in logic and Philosophy*. Kluwer Academic Publishers.
- [MRK⁺03] Amitabha Mukerjee, Achla M Raina, Kumar Kapil, Pankaj Goyal, and Pushpraj Shukla. Universal networking language- a tool for language-independent semantics? In *In Proc. of Convergence 2003*, 2003.
- [Nil03] Adam Pease Niles, I. Linking lexicons and ontologies:mapping wordenet to suggested upper merger ontology. In Chris Welty and Barry Smith, editors, *International Conference on Information and Knowledge Engineering*, June 2003.
- [NM00] N.F.Noy and M.A.Musen. PROMPT: Algorithm and tool for automated ontology merging and alignment. In *In Proceedings of the National Conference on Artificial Intelligence AAAI*, 2000.
- [NP01] I Niles and Adam Pease. Towards a standard upper ontology. In Chris Welty and Barry Smith, editors, *Proceedings of the 2nd International Conference on Formal Ontology in Infomation Systems(FOIS-2001)*, Ogunquit, Maine, October 17-19 2001.
- [Sow83] John F. Sowa. *Conceptual Structures: Information Processing in Mind and Machine*. Addison-Wesly, 1983.
- [SUM03] Suggested Upper Merged Ontology, 2003. Retrieved from <http://ontology.teknowledge.com>.

- [SUO04] Standard Upper Ontology Working Group, 2004. Retrieved from <http://suo.ieee.org>.
- [Unl03] UNL system, 2003. Retrived from <http://www.undl.org/unlsys/>.
- [VR] Andrei Voronkov and Alexandre Riazanov. Vampire- a theorem prover for first-order classical logic. From <http://www.cs.man.ac.uk/~riazanoa/Vampire/>.