MAULANA AZAD NATIONAL INSTITUTE OF TECHNOLOGY, BHOPAL, MP, INDIA (462051)

BUILDING NATURAL LANGUAGE GENERATION SYSTEMS FOR DYNAMIC SPATIAL-TEMPORAL DATA

VIII SEMESTER EQUIVALENT RESEARCH INTERNSHIP REPORT, CONDUCTED AT UNIVERSITY OF BREMEN, GERMANY, A BACHELOR MAJOR PROJECT REPORT.

SUBMITTED IN PARTIAL FULFILLMENT OF THE REQUIREMENTS FOR THE AWARD OF THE DEGREE OF

BACHELOR OF TECHNOLOGY
(COMPUTER SCIENCE AND ENGINEERING)

SUBMITTED BY:

HARSHITA JHAVAR
111112003
(MANIT BHOPAL)

SUPERVISED BY:

Prof. SARITHA KHETHAWAT
(MANIT BHOPAL)
Dr. MEHUL BHATT
(UNIVERSITY OF BREMEN)

STUDENT'S DECLARATION

I hereby declare that the work presented in the report entitled "BUILDING NATURAL LANGUAGE GENERATION SYSTEMS FOR DYNAMIC SPATIO-TEMPORAL DATA" submitted by me, for the fulfillment of the requirements for the degree of Bachelor of Technology in Computer Science & Engineering at Maulana Azad National Institute of Technology, Bhopal, MP, India, is an authentic record of my work carried out at University of Bremen under guidance of Dr. Mehul Bhatt, University of Bremen and Professor Saritha Khethawat, MANIT Bhopal. Due acknowledgements have been given in the report to all material used. This work has not been submitted anywhere else for the reward of any other degree or diploma. This is to certify that the above declaration is correct to the best of my knowledge.

.....

Harshita Jhavar Place & Date: .....

MAULANA AZAD NATIONAL INSTITUTE OF TECHNOLOGY BHOPAL –462051

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING

CERTIFICATE

This is to certify that Harshita Jhavar student of final year B. Tech (Computer Science and Engineering) has successfully completed her major project 'BUILDING NATURAL LANGUAGE GENERATION SYSTEMS FOR DYNAMIC SPATIAL, TEMPORAL DATA' for the partial fulfilment of the degree - Bachelor of Technology in Computer Science and Engineering.

Prof. Saritha Khethawat
(Project Internal Advisor)
Dept. of Computer Science and Engineering,
MANIT BHOPAL

Prof. Manish Pandey
(Major Project Coordinator)
Dept. of Computer Science and Engineering,
MANIT Bhopal

ACKNOWLEDGEMENT

I feel it is my proud privilege to express my deep sense of gratitude and indebtedness to Dr. Mehul Bhatt, Uni-Bremen (external advisor); Jakob Suchan, Uni-Bremen (phd student), Prof. Saritha Khethawat, MANIT Bhopal (internal advisor) for providing their painstaking and untiring supervision. I thank them for their constructive criticism, valuable suggestions and constant encouragement at all stages of development of this project. I also would like to thank Prof. Manish Pandey and Prof. Nilay Khare for their formal procedural guidance for conducting a semester equivalent internship at Germany. I thank the entire team at Space Design Lab, Uni-Bremen for their help and for providing a conducive environment for proper development of the project and the necessary facilities for completion of the project. I also thanks University of Bremen and Santander Bank for providing me BISIP Santander scholarship at Germany. Last but not the least, I would like to express my appreciation towards my family members and friends for providing much needed support and encouragement.

Harshita Jhavar

ABSTRACT

Built Natural Language Generation engine for analysed dynamic spatial-temporal data, collected from various experiments conducted at lab. Generated text summaries of different lengths, in different tenses with different clause structures – simple, complex and compound. The engine is based on predicates built using regular grammar. After the identification of the parts of speech of different input data, the data tokens are arranged according to Grammar to generate a syntax tree and thereby, a sentence. The time taken to generate summaries with different tenses and to generate sentences with different clause structures was found. Being declaratively built in Prolog, this NLG engine proves to be highly fast, efficient, scalable and robust**.**

CONTENTS

LIST OF FIGURES

CHAPTER 1
INTRODUCTION

Natural Language Generation is a field which deals with generation of text from data. It is a subfield of artificial intelligence and computational linguistics that focusses on computer systems that can produce understandable texts in English or other human-languages. The application domain deals with producing documents, reports, explanations, narratives, messages and other kinds of texts.

This area of research is really fascinating and is built for real world applications. It brings a unique perspective towards fundamental issues in artificial intelligence, cognitive science and human - computer interaction. Questions like how linguistics and domain knowledge should be represented and how communication can take place between machine and human are answered by this field. From a practical perspective, NLG technology is capable of partially automating routine document creation, removing much of the drudgery associated with such tasks. It is also being used in the research laboratories as in this case, at Space Design Cognitive Solutions Lab at University of Bremen, Germany. The main motive is to explain complex information to people who do not have the background or time required to understand the raw data. In the longer term, NLG is also likely to play an important role in human–computer interfaces and will allow much richer interaction with machines than is possible today.

NLG is a subfield of natural language processing (NLP), which in turn can be seen as a subfield of both computer science and cognitive science. The relation between NLG and other aspects of NLP is further discussed below. From the broader perspective of computer science and cognitive science as a whole, NLG provides an important and unique perspective on many fundamental problems and questions, including the following:

- How should computers interact with people?
- What is the best way for a machine to communicate information to a human?
- What kind of linguistic behavior does a person expect of a computer he or she is communicating with, and how can this behavior be implemented?

These are basic questions in human–computer interaction, an area of computer science which is becoming increasingly important as the quality of computer software is judged more and more by its usability.

• How can typical computer representations of information – large amounts of low-level (often numeric) data – be converted into appropriate representations for humans, typically a small number of high-level symbolic concepts? What types of domain and world models and associated reasoning are required to ‘translate’ information from computer representations to natural language, with its human-oriented vocabulary and structure?

These questions are aspects of the larger question of how humanlike intelligence can be modelled and simulated on a computer, which is one of the main goals of artificial intelligence (AI).

Although work in natural language generation can provide insights in these related fields, it also draws on these fields for ideas. For example, many NLG systems use ideas developed within artificial intelligence, such as planning techniques and production rules, to determine the information content of a text; and most NLG systems use formal linguistic models of syntax to ensure that their output text is grammatically correct.

CHAPTER 2
PROBLEM IDENTIFIED

In order to understand experiments based on analysis of some type of big data, there should be a way such that machine generating the data should be able to generate human like reports of the same. The amount of human labour involved in analysis, writing summaries from those analysis, narratives is very high. Therefore, there should be a way by which text can be generated from the machine generated data. So, the field of Natural Language Generation comes as perfect solution to this problem.

There are various Natural Language Generators for different domains, like weather forecasting, baby health report generation etc. But there is no nlg engine which can read analysed dynamic spatio-temporal domain data. The scientists dealing with various visual computing, space design or image processing experiments have to either hard code the text which they intend to generate or they have to use the pre-defined templates. There exists no generic model through which the text can be generated.

Thus, space-time data, obtained from various experiments at Space Design Lab, University of Bremen, Germany, was used to test the NLG engine, built during this project, to find the rate of generation and other properties of this NLG engine.

CHAPTER 3
PROJECT PLANNING

MONTH	TASK	
February	Literature Survey on NLG	✓
March	NLG Engine Development	✓
April	NLG Engine Development	✓
May	Phase 1 Test with various test cases	✓
June	Report and Presentation	✓

- Every Monday and Friday, there was a review meeting at Space-Design Lab, Uni-Bremen with my external advisor and the team.
- Presented my weekly and monthly progress, in form of various presentations.
- Gave demo test at various stages.
- Internship started on February 1, 2015 and ended on June 30, 2015.

CHAPTER 4
LITERATURE SURVEY

For literature survey, I went through different books and papers related to my problem statement in an attempt to find solution to the problem statement. The names of books and papers have been mentioned in the references. Some of the key-points of literature survey are:-

4.1 Differences between Natural Language Understanding and Natural Language Generation:

Natural language generation is closely related to natural language understanding, which is the study of computer systems that understand English and other human languages. Both NL understanding and NL generation are concerned with computational models of language and its use; they share many of the same theoretical foundations and are often used together in application programs. Together, natural language understanding and natural language generation form the field of natural language processing (NLP).

At an abstract level, process of natural language generation can be thought of as being the inverse of the process of natural language understanding. Natural language generation is the process of mapping internal computer representations of information into human language, whereas natural language understanding is the process of mapping human language into internal computer representations. Thus, at least in general terms the two processes have the same end points, and their difference lies in the fact that they are concerned with navigating between these end points in opposite directions.

But the internal operations of these processes are quite different in character. Most fundamentally, the process of natural language understanding is best characterized as one of hypothesis. Given some input, which of the multiple possible interpretations at any given stage of processing is the appropriate one? Natural language generation is best characterised as a process of choice: Given the different means that are available to achieve some desired end, which should be used?

A major problem in building real NLU systems is the need to be able to handle grammatically incorrect or ill-formed input, which is not a problem in NLG, whereas a major problem in NLG is ensuring that generated text is easily comprehensible by humans, which is not a problem in NLU. A related problem for bidirectional grammar systems is that the internal representations used by NLG and NLU systems are very different. For example, the representations which most NLU parsers produce as output are quite different from the input representations required by most NLG realizers. This may change in the future.

It seems intuitively plausible that the same representations of knowledge about language should be used for both generation and analysis. Many of the fundamental tasks of NL generation, such as deciding what information should be communicated in a text, have no clear analogue in NL understanding. NL understanding is driven by the need to be able to handle the large variety of complex linguistic structures that make up any natural language. This aspect of coverage is of somewhat less importance in NL generation, and indeed the texts produced by most NL generation systems are much simpler in linguistic terms than the texts that NL understanding systems aspire to process. For work in NLG, it is often good enough to have one way of saying something, whereas any experiment in NLU has to take account of the fact that many different inputs may be used to express much the same thing.

4.2 The Application Perspective:

From an applications perspective, most current NLG systems are used either to present information to users, or to (partially) automate the production of routine documentation. Information presentation is important because the internal representations used by computer systems often require considerable expertise to interpret. Representations such as airline schedule databases, accounting spreadsheets, expert system knowledge bases, grid-based simulations of physical systems, and so forth are straightforward for a computer to manipulate but not always easy for a person to interpret. This means that there is often a need for systems which can present such data, or summaries of the most important aspects of the data, in an understandable form for non-expert users. When the best presentation of the data is in English, Spanish, Chinese, or some other human language, NLG technology can be used to construct the presentation system. Automating document production is important because many people spend large proportions of their time producing documents, often in situations where they do not see document production as their main responsibility. A doctor, for example, may spend a significant part of his or her day writing referral letters, discharge summaries, and other routine documents. Similarly, a computer programmer may spend as much time writing text (code documentation, program logic descriptions, code walkthrough reviews, progress reports, and so on) as writing code. Tools which help such people quickly produce good documents may considerably enhance both productivity and morale. When we build a complete NLG system, a major design decision that has to be taken is whether that system will operate in standalone mode, generating texts without any human information, or whether it will operate by producing what are in effect drafts of texts that are subsequently modified by a human author. This distinction is essentially the same as that found more broadly in work in expert systems, where it is often deemed appropriate to maintain the presence of a ‘human in the loop’, especially in contexts where a mistake on the part of the system could be life-threatening or disastrous in some other way. In the case of NLG, the reasons for including a human in the authoring process are generally less dramatic. Put simply, in many contexts it is simply not possible

to create texts of the appropriate quality or with the required content without human intervention.

4.3 Computer as Authoring Aid

In practice, current NLG technology and the limitations of the information available in host systems mean that it is often not possible for the system to create the final product; instead, the NLG system is used to produce an initial draft of a document which can then be further elaborated or edited by a human author. A variant of this approach is for the NLG system to focus on producing routine factual sections of a document (which human authors often find monotonous to write), leaving analytical and explanatory sections to the human author. Examples of NLG systems used as authoring assistants are the following:

• **FOG** (Goldberg, Driedger, and Kittredge, 1994), which helps meteorologists compose weather forecasts.

• **PLANDOC** (McKeown, Kukich, and Shaw, 1994), which helps telephone network engineers document the results of simulations of proposed network changes.

• **ALETHGEN** (Coch, 1996a), which helps customer-service representatives write response letters to customers.

• **DRAFTER** (Paris et al., 1995), which helps technical authors write software manuals.

As indicated above, it is probably true to say that most practical systems are required to operate in the 'computer as authoring aid' mode, sometimes for legal or contractual reasons. However, many experimental NLG systems have been developed with the aim of operating as standalone systems that require no human intervention when creating texts. Some such systems are as follows:

• **MODELEXPLAINER** (Lavoie, Rambow, and Reiter, 1997), which generates textual descriptions of classes in an object-oriented software system, using information extracted from a computer-aided software engineering database.

• **KNIGHT** (Lester and Porter, 1997), which explains information extracted from an artificial intelligence knowledge base.

• **LFS** (Iordanskaja et al., 1992), which summarises statistical data for the general public.

• **PIGLET** (Cawsey, Binstead, and Jones, 1995), which gives hospital patients explanations of information in their patient records.

Given the current state of NLG technology, systems which work in collaboration with a human author are more practical, allowing a symbiotic approach where the machine produces the more routine aspects of documents and leaves the more difficult-to- automate aspects to the human. However, advances in NLG technology should make it possible in the near future to build field able standalone systems in some domains, just as mail-merge systems are used to customise letters to individual recipients without a perceived need for each letter to be checked by a human.

4.4 Study of Existing NLG Systems

When developers build a complete NLG system, they do so with a certain application context in mind. The system serves some purpose. Stories of such systems generating meaningless letters to deceased persons or other inappropriate categories of recipients abound. The difficulty of predetermining all the relevant circumstances is one very good reason for maintaining the role of a human overseer in systems of any complexity.
These are built with the purpose of presenting information to the user in a straightforward and unbiased manner. However, some experimental systems have explored other uses of NLG technology, including the following:

- **Teaching.** The ICICLE system (McCoy, Pennington, and Suri, 1996) helps deaf people learn the syntax of written English.

- **Marketing.** The DYD system (van Deemter and Odijk, 1997) generates descriptions of a music CD which are intended to increase interest in and sales of that CD.

- **Behaviour Change.** The STOP system (Reiter, Robertson, and Osman, 1997) generates personalised letters which encourage people to stop smoking.

- **Entertainment.** The JAPE system (Binstead and Ritchie, 1997) generates jokes (more specifically, punning riddles).

All of the aforementioned systems are research prototypes. To be really successful in their domains of application, systems such as these need to embody models of how to teach effectively, how to persuade people to buy things, how people break addictive habits, and what people find amusing. Currently there are no precise computational models in any of these areas, which means that building effective systems is partially a research project in the computational modelling of teaching, persuasion, and so forth, as well as in language generation.

It is perhaps unfortunate that the popular media pay far more attention to systems which generate fictional forms such as jokes, novels, or poetry than to any other kind of NLG

system. For example, the JAPE system mentioned above has been reported upon in the UK in both the tabloid and the more serious press, and in radio and television broadcasts.

- ✓ **Weather Reporter:** The purpose of WEATHER-REPORTER is to provide retrospective reports of the weather over periods whose duration is one calendar month. It does this by taking as input a large set of numerical data collected automatically by meteorological devices, from which it produces short texts of one or two paragraphs in length.

The month was cooler and drier than average, with the average number of rain days. The total rain for the year so far is well below average. There was rain on every day for eight days from the 11th to the 18th.

Figure 4.1 The Macquarie weather summary for February 1995.

Figure 4.1 shows an example text that might be produced by WEATHER-REPORTER. This is fairly typical of the kinds of texts required. It provides information on temperature and rainfall for the month as a whole, and it provides descriptions of spells of weather that are noteworthy for one reason or another. Some of the texts can be quite different from this general form, depending on the nature of the data to be reported.

96,122,1,5,2.00,200,-14.41,-3.668,-1.431,.345,1023,15.41,15.82,20.07,-11.1,-2.878,104.2,.28,153.6,53.19,0,16.26
96,122,1,5,2.25,215,-10.72,-3.241,-1.35,.152,1023,15.3,15.78,20.07,-11.42,-2.762,105,.208,98.2,.822,0,17.05
96,122,1,5,2.50,230,-8.37,-1.282,-.904,2.15,1022,15.3,15.71,20.05,-11.66,-3.206,104.4,.2,141.6,42.96,0,17.7
96,122,1,5,2.75,245,-12.81,-2.11,-1.067,2.119,1022,15.33,15.79,19.99,-11.15,-3.093,104.8,.2,186.5,11.32,0,17.81
96,122,1,5,3.00,300,-13.68,-3,-1.35,1.075,1022,15.36,15.79,19.96,-10.63,-3.005,104.6,.402,285.8,61.45,0,18.47
96,122,1,5,3.25,315,-10.2,-2.457,-1.13,-.73,1022,15.32,15.66,19.92,-11.17,-3.263,103.6,.304,354.7,36.29,0,19.03
96,122,1,5,3.50,330,-9.33,-1.353,-.942,.902,1022,15.21,15.62,19.9,-10.95,-2.903,104.3,.313,302.2,34.69,0,19.16
96,122,1,5,3.75,345,-7.29,-.285,-.76,2.048,1022,15.24,15.63,19.87,-10.68,-3.27,104,.252,313,29.7,0,19.61
96,122,1,5,4.00,400,-6.822,-.365,-.653,1.531,1022,15.25,15.63,19.83,-9.93,-3.316,104,.331,274.2,52.98,0,20.42
96,122,1,5,4.25,415,-8.78,-.65,-.747,1.602,1023,15.35,15.66,19.79,-9.77,-2.656,103.3,.253,247.7,10.99,0,21.08
96,122,1,5,4.50,430,-8.73,-.641,-.741,1.785,1023,15.46,15.81,19.75,-9.16,-2.782,103.7,.2,295,29.15,0,21.3
96,122,1,5,4.75,445,-11.45,-2.671,-1.03,-.456,1022,15.46,15.82,19.74,-8.81,-2.464,103.7,.2,355.3,23.98,0,21.65
96,122,1,5,5.00,500,-13.12,-4.3,-1.306,-1.359,1022,15.42,15.75,19.76,-9.39,-2.49,103.4,.2,20.67,.188,0,21.83
96,122,1,5,5.25,515,-13.62,-4.621,-1.344,-.842,1022,15.32,15.67,19.81,-9.47,-2.703,103.7,.2,20.65,.183,0,21.98
96,122,1,5,5.50,530,-13.8,-3.534,-1.325,.943,1022,15.23,15.61,19.86,-10.92,-3.384,103.9,.2,20.65,.183,0,22.14
96,122,1,5,5.75,545,-14.7,-3.748,-1.419,.385,1022,15.06,15.47,19.9,-11.62,-2.868,104.4,.2,341.6,18.6,0,22.36
96,122,1,5,6.00,600,-13.61,-2.315,-1.287,2.038,1022,14.98,15.42,19.9,-12.37,-3.092,104.7,.2,298.6,5.173,0,22.54
96,122,1,5,6.25,615,-14,-2.894,-1.293,.669,1022,14.92,15.36,19.88,-12.48,-3.808,104.7,.591,320.3,21.07,0,22.87

The 22 data values in each record are as follows: the year; the day of the year as a value between 1 and 365; the month of the year; the day of the month; the time in 24 hour notation; the time in hours and minutes; four radiation readings; the barometric pressure; the temperature, referred to as 'dry bulb temperature'; the wet bulb temperature, which is used to calculate humidity; the soil temperature; the soil heat flux at two depths; relative humidity; average wind speed; wind direction; the standard deviation of wind direction; precipitation within the 15 minute period; and amount of sunlight.

Figure 4.2 Data Input For W-R System

(Source: *Dale, Robert; Reiter, Ehud (2000). Building natural language generation systems. Cambridge, U.K.: Cambridge University Press. ISBN 0-521-02451-X.)*

Figure 4.2 shows a fragment of the data that WEATHERREPORTER uses as input. Each line represents one data record, these being collected at 15-minute intervals by an automatic weather station. Retrospective weather reports similar to the one shown in automatically by meteorological data gathering equipment on the university campus. The WEATHERREPORTER design study is thus based on real input data and a real corpus of human-written texts, and this description of WEATHERREPORTER how humanlike texts can be generated from real input data using NLG techniques.

- ✓ **The FOG System** (Goldberg et al., 1994) generates textual weather forecasts from numerical weather simulations produced by a supercomputer and annotated by a human forecaster. More precisely, it takes as input a numerical prediction of how wind speeds, precipitation type and intensity, and other meteorological phenomena vary over a given region in a specified time interval, and produces as output a textual summary of this information. An example of a graphical rendition of some of FOG's input is shown in Figure 4.3, which depicts the predicted weather system over Northern Canada at 0600 GMT on 14 August 1998. An example of FOG's output is shown in Figure 4.4; this is a marine forecast issued at 04:25 EDT (09:25 GMT) on 13 August 1998, which describes the predicted weather over various points in Northern Canada. Because it is a marine forecast, it emphasises wind and precipitation information and says nothing about temperature; other types of forecasts would emphasise different information. The texts produced by FOG are not very exciting, but they are useful and more difficult to generate than might at first seem to be the case. One complexity in FOG that may not be immediately apparent is that it must decide how detailed the information it provides should be. In the example in Figure 4.4, for instance, FOG has decided to produce a single summary forecast that covers both East Brevoort and East Davis but a separate forecast for East Clyde. Also, FOG needs to decide when it can use inexact temporal terms like late this evening or early Friday evening and when it should use a more precise phrase such as at midnight. Another technically interesting aspect of FOG is that it can produce forecast texts in both English and French; in other words, it is a MULTILINGUAL system. Internally, FOG first produces a language-independent abstract representation of the forecast text and then maps this into each output language using appropriate grammatical and lexical resources. The FOG system was developed by CoGenTex, a specialist NLG software house, for Environment Canada (the Canadian weather service). It has been in everyday use since 1993.

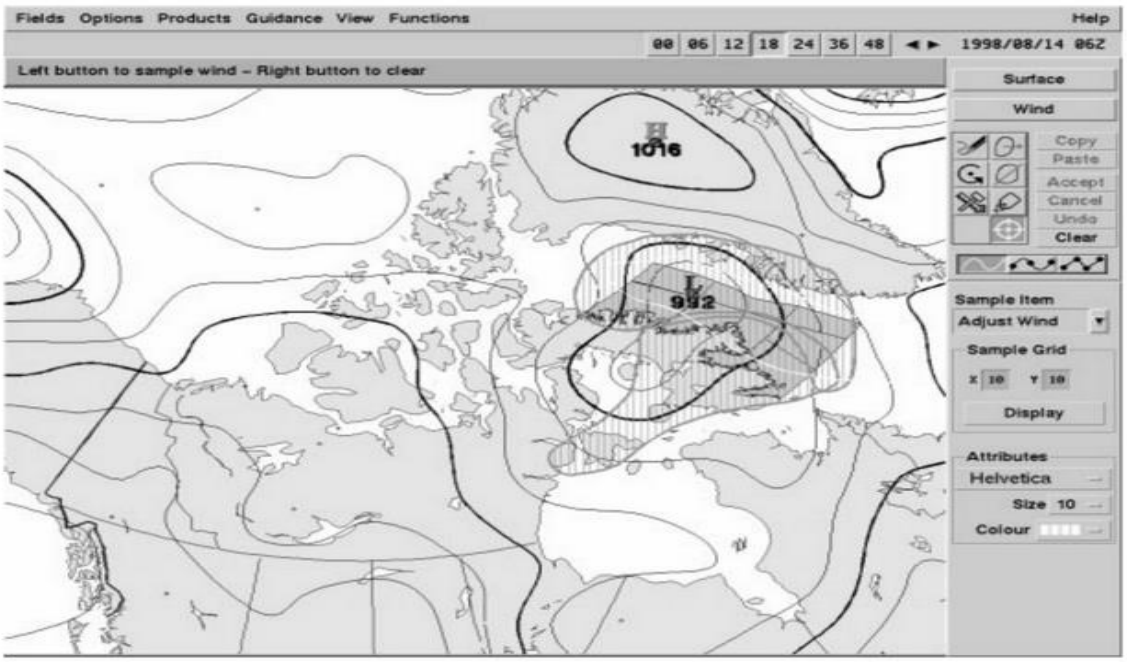

Fig 4.3 Fog Input From Canada (Source: Environment Canada)

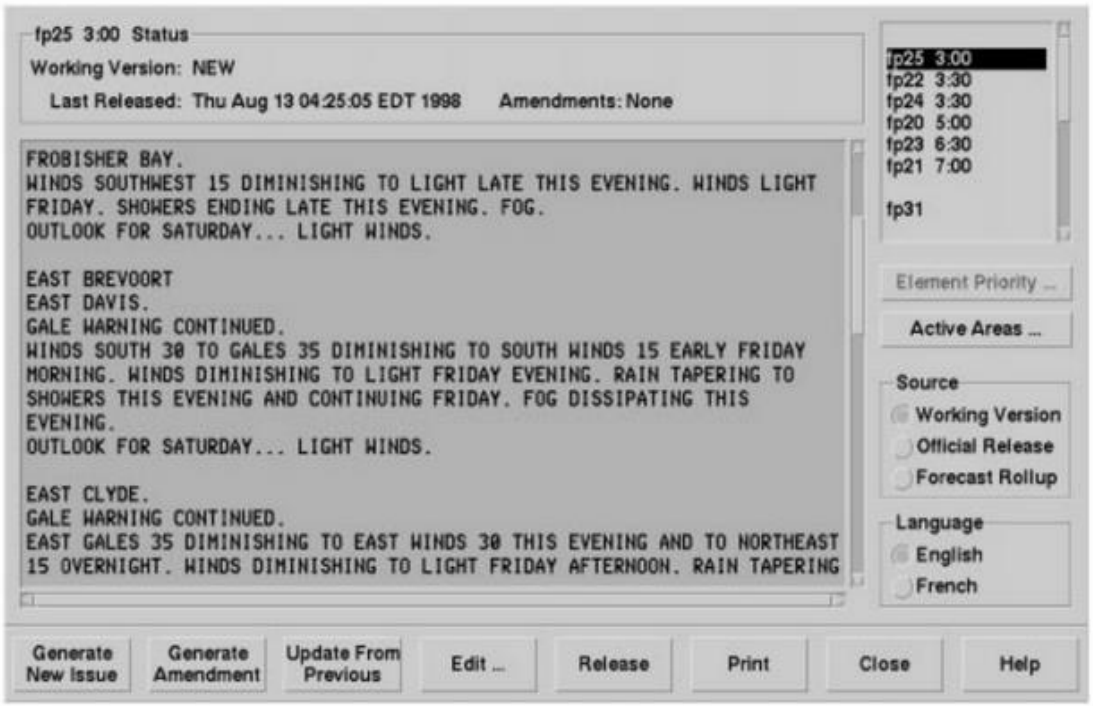

Fig 4.4 Some Examples of Weather Forecast (Source: Environment Canada)

CHAPTER-5
METHODOLOGY

5.1 Architecture of the NLG Engine

This NLG system is a 'Single-Interaction' mode engine where each invocation of the system produces a single sentence in order to produce spatio-temporal domain based text. The entire system has been built declaratively in prolog. The specification of the different types of processing in this NLG Engine are distributed across a number of distinct, well-defined and easily-integrated component modules. These modules interact with each other during the process of text generation. Less complex modularization made it easier to reuse the components efficiently amongst the different applications.

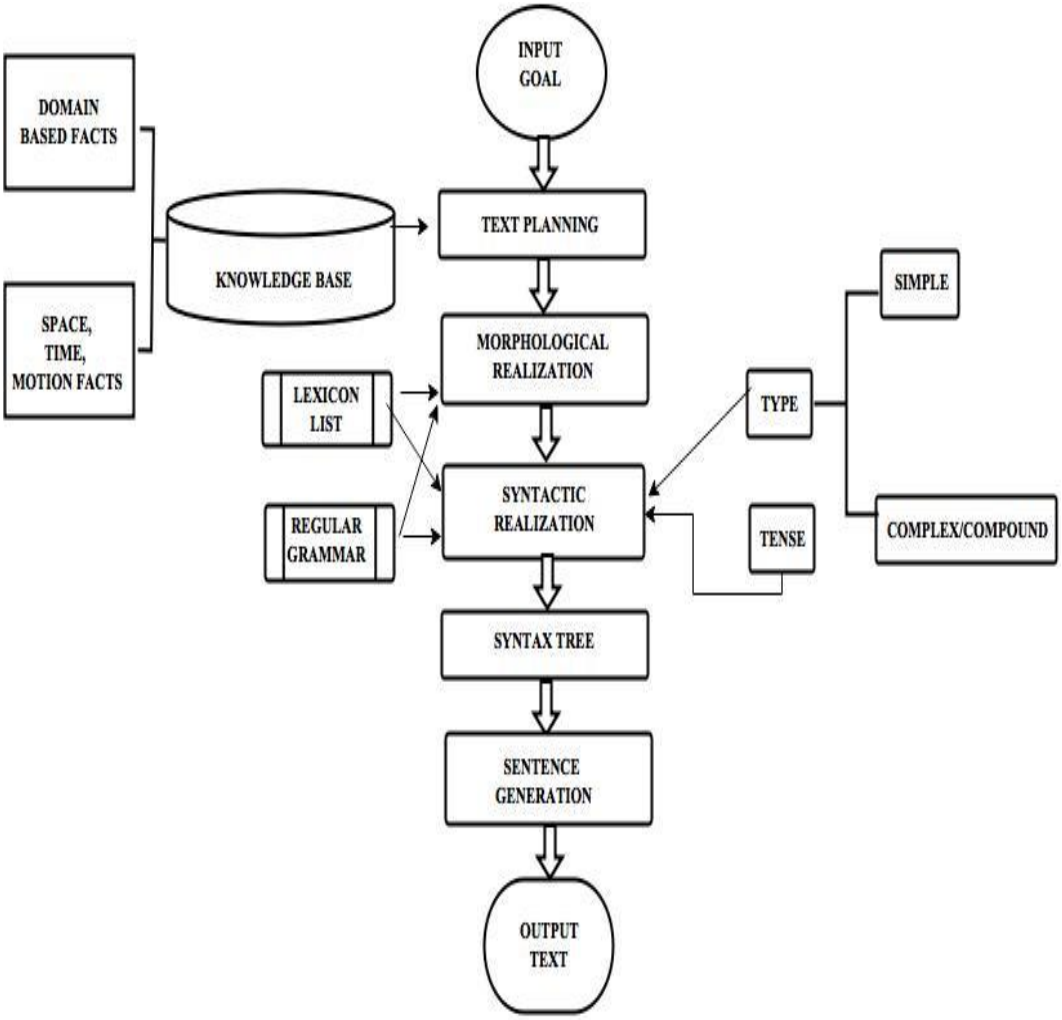

Fig 5.1 Architecture of NLG Engine

The production of the output utterances from this NLG engine follow the following architectural specification of the modules, executed in response to the user inputs:

5.1 Input Goal

The spatio-temporal domain experts are heavily involved to obtain the corrective summarised data value corresponding to the various analysis of the recorded data from the experiments. They try to best fulfill the communicative goal in the context of information to be conveyed for the recorded set of experiments. The input to our NLG engine is a structure that contains the information selected for content determination by them. Well, the input data to the engine comes from various visual-computing scientists. It is organized as:

5.1.1 Dictionary Data File

The dictionary data file consists of all the object, subject and motion-name list which are input in system as nouns and verbs as discussed in Morphological Realization in Section 5.3.1. esp. 5.3.1.1. A sample dictionary file has been attached in Section 6.1.

5.1.2 Domain Description

The domain description describes all the spatial relations between different objects which are read by machine as preposition. This file also holds the various analyzed properties of all the object which are read by machine as adjective. It holds the various analysis values for different motions recognized by the system and are read by the machine as adverbs. These identification of different parts of speech takes place in Morphological Realization as discussed in Section 5.3.1 esp 5.3.1.2. A sample domain description file has been attached in Section 6.1.

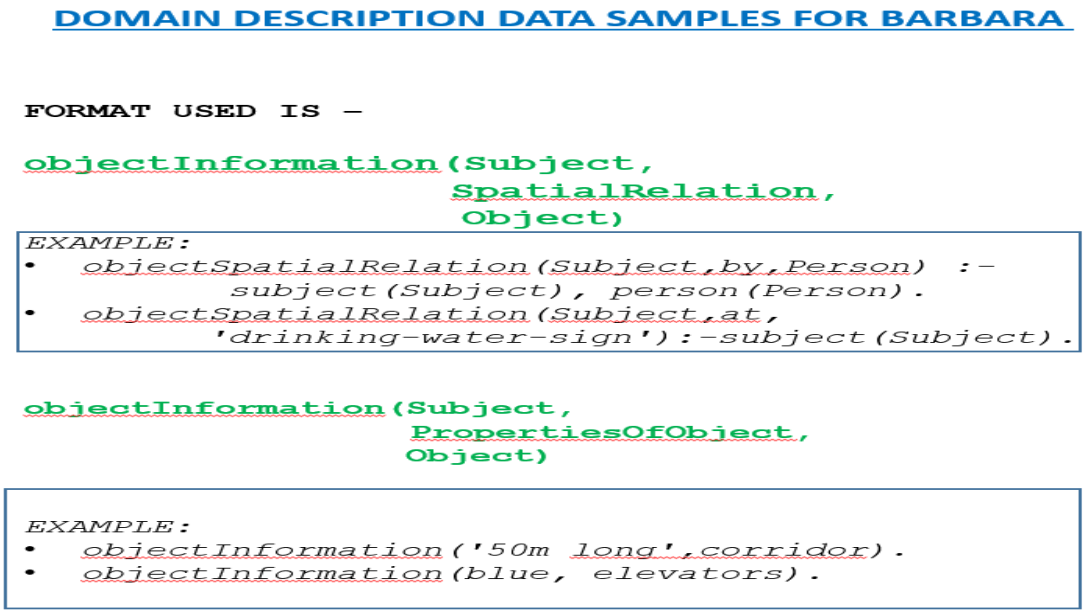

Fig 5.2 Format for Domain Description

5.1.3 High Level Data Analyzed Input for Different Case Scenarios

This file consists of the subject information, action information and object information. Which object comes in play in which time coordinate, which action(s) took place in which time coordinate are mentioned here. The decision of generating complex and compound or simple sentences takes place from this file. The actions happening in same time coordinate for different users or may be same user, are taken into complex sentence generation as described in stages below. A sample high level Data Analyzed Input file has been attached in Section 6.1 for a user 'Barbara'.

```
/*Subject Name*/
subject(barbara).

/*actionInformation(Subject,PropertiesOfMotion,Motion) for
barbara*/actionInformation(barbara,quickly,walk).

/*Compound Sentence Generation*/
compound(action(barbara, pass, mehul, 112), action(mehul, reach, sign, 115)).

/*(Complex) SimultaneousAction(Subject Name, Action, Object) of barbara*/
action(barbara,walk,corridor,1-1).
action(barbara, pass, mehul,1-2).

/*action(Subject, Action, Object) of barbara*/
action(barbara, focus, sign,11).
action(barbara, turn, left,12).
```

5.2 Text Planning: Structure of the Input Text

For every single interaction obtained, the spatio-temporal domain experts alot words corresponding to the analysed data which are taken as input into our NLG engine in the following format. The system tries to structure the high level data in the following format.

5.2.1 Input Structure For Single Motion:

- Properties of the subject in the interaction,
- Name of the subject in the interaction,
- Analysis of the motion identified,
- Name of motion identified,
- Object List
  - [Relation between the two consecutive results,
  - Properties of the object,
  - Name of the object],
- Expected Tense of Action.

An example of prolog format based data input for simple sentence structure as read by our system:

generateSimpleSentence ([
[],'Barbara', [], focus, on, 'Drinking Water', sign, at, corridor, end
], Sentence).

5.2.2 Input Structure For Simultaneous Motions:

Put_All_Action_Information[

- Properties of the subject in the interaction,
- Name of the subject in the interaction,
- Properties of the motion identified,
- Name of motion identified,
- Object_List
  - [Relation between the two consecutive results,
  - Properties of the object,
  - Name of the object]
- Expected Tense of Action]

] Till_All_Actions_Not_Over.

Example of prolog format based data input for simultaneous sentence structure read by our system for complex and compound sentences:

```
generateComplexSentence(
    [
        [
          [], barbara, [], walk, in, [], corridor, [], [], [], present_continuous
        ],
        [
        three, persons, [], pass, to, [], left, [], [], [], simple_present
        ],
        [
        one, person, [], pass, from, [], right, [], [], [], simple_present
        ]
    ], Sentence ).
```

```
generateCompoundSentence(
    [
        [
          [], barbara, [], walk, in, [], corridor, [], [], [], present_continuous
        ],
        [
          [], barbara, [], want, to, [], reach, [], blue, elevator, simple_present
        ],
    ], Sentence ).
```

5.3 Linguistic Realization

This stage allows detailed grammatical knowledge to be encapsulated within the morphological and syntactic realizer as described in the following sections:

5.3.1 Morphological Realization

Morphological realization deals with the content aspect of the syntactic realization. This stage involves the following:

5.3.1.1. Lexicon List –

The user is allowed to generate this list independently according to the domain of the analyzed data. This list consists of a list of all nouns and verbs, expressed in the form of prolog facts.

- Prototype of structure of a noun in the list:

```
noun (Value of Noun, Type of Noun).
```

- Living nouns are the names of anything that represents a living object and is involved in the experiment.
  Examples of living nouns in prolog used in our NLG engine are:

```
noun (barbara, living).
noun (user, living).
```

- Definite non-living nouns are anything that represents a specific landmark in the experiment. Examples of definite non-living nouns in prolog used in our NLG engine are:

```
noun (sign, definiteNonLiving).
noun (trashbins, definiteNonLiving).
```

- Indefinite non-living nouns are anything that represents the nouns that are talked in general but are not specifc. Examples of indefinite non-living nouns used in our NLG engine are:

```
noun (door, indefiniteNonLiving).
noun (board, indefiniteNonLiving).
```

- The spatial word list, consisting of directions and locations, are also mentioned under definite non-living nouns as they are specific in their meaning at an instance and are structured to be a part of the object phrase in our NLG system. Examples of spatial word list expressed as nouns:

```
noun(right, definiteNonLiving).
noun(left, definiteNonLiving).
noun(end, definiteNonLiving).
```

- The verbs are listed as facts in the list. They are structured as:
  Structure of a verb in the list:

```
verb ( Root Form of Verb, Present Form of Verb, Past Form of Verb, Continuous Form of Verb).
```

- The different examples are:

```
verb(walk,walks,walked,walking).
verb(pass,passes,passed,passing).
verb(do,does,did,doing).
```

These pre-existing well-structured verbs and nouns in our system are called during the stage of Syntactic Realization when these words play according to our domain specific Grammar to come up with a sentence.

5.3.1.2 Identification of the Part of Speech Category of Each Input Argument-

The input text is structured as explained in stage of Text Planning. The various part of speech from the input text are identified in the following correspondence:

- Properties of the subject in the interaction - Adjective
- Name of the subject in
- the interaction - Noun
- Analysis of the motion identified
  - Adverb
- Name of motion identified - Main Verb
- ObjectList
  - [Relation between the two
  - consecutive results - Preposition,
  - Properties of the object- Adjective,
  - Name of the object] - Noun
- Expected Tense of Action - Tense

Thus, all the parts of speech are identified. The constrained data structure used for input data made it easier and less complex for our NLG system to identify the various parts of speech. This information is now passed to the Syntactic Analyzer where the lexicons and their types identified during the morphological analyzer meet the rules written for generation in form of Grammar. An example of this in
prolog format is:

```
generateSentence(
                    [
                      [],     %%Read as Adjective
                      barbara,%%Read as Noun
                      quickly,%%Read as Adverb
                      pass,   %%Read as Verb
                      behind, %%Read as Preposition
                      [],     %%Read as Adjective
                      Tom,    %%Read as Noun
                      at,     %%Read as Preposition
                      meeting,%%Read as Adjective
                      room,   %%Read as Noun
                      present_continuous%% Read as Tense
                ]
                ,Sentence
              ).
```

5.3.2 Syntactic Realization

The parts of speech identified by the Morph Analyzer from the Morphological Realization are consulted with the simple context free grammar rules, executed according to the type of sentence, identified by the input structure:

- ✓ For Simple Sentences:

```
<Sentence> ::== <NounPhrase> <VerbPhrase>
<NounPhrase> :== [<det>] <adjective*> <Noun>
<VerbPhrase> :== [<auxillaryVerb>] <adverb*> <mainVerb>
c                    [<ObjectPhrase>]
<ObjectPhrase> :== [<preposition>] <nounPhrase>
```

- ✓ For Complex Sentences:

```
<Sentence> ::== <S> ('because')|('therefore')|('so')|('after') <S>
<S> ::== <NounPhrase> <VerbPhrase>
<NounPhrase> :== [<det>] <adjective*> <Noun>
<VerbPhrase> :== [<auxillaryVerb>] <adverb*> <mainVerb>
                 [<ObjectPhrase>]
<ObjectPhrase> :==  [<preposition>] <nounPhrase>
```

The above grammar predicates are implemented in prolog. The phrases are appended to generate the complete sentence. The following predicates come into execution:

- ✓ **Frame Noun Phrase**

  The determiner before the noun is obtained in consultation with the Lexicon list discussed in Morphological realization.

  - Living Nouns - no determiner is identified.
  - Definite Non Living Nouns - 'the' is identified as determiner.
  - Indefinite Non Living Nouns - 'a' is identified as determiner.

  Then, the chosen determiner is appended with the nouns and their corresponding adjectives as identified by Morphological Realizer. Thus, the subject phrase is framed. Similarly, all the object phrase, if any, follow the frame Noun Phrase rule to form the corresponding phrases for each object entered with its analyzed properties.

- ✓ **Frame Verb Phrase**

The verb phrases are obtained by following the input tense format and the corresponding word form and the auxiliary words are appended to the main verb. So, the verb phrase comes up with the helping verb, adverb and the verb. For complex/compound sentences, the connecting words are appended according to the grammar. They thus, append and become a part of verb/noun phrase. Thus, the subject phrase, verb phrase and object phrase are appended to generate a syntax tree out of it. The prolog example for this from our NLG engine is as follows:

generateSimpleSentence([little, barbara, quickly, focus, on, drinking-water, sign, at, corridor, end, present_continuous],Sentence).

/ barbara is a living noun, so no determiner, sign is definite non living noun, so 'the' is determiner, end is a definite living noun, so 'the' is determiner, focus has <is focussing> as its form of usage as tense is present_continuous,*

<little> + <barbara> = noun phrase

<is> + <quickly> + <focussing> = verb phrase

<on> + <the> + <drinking-water> + <sign> = Noun phrase

*<at> + <the> + <corridor> + <end> = Noun phrase */*

5.4 Syntax Tree

The syntax tree is obtained as a result of the morphological and syntactic realization. The structure of the syntax tree looks like:

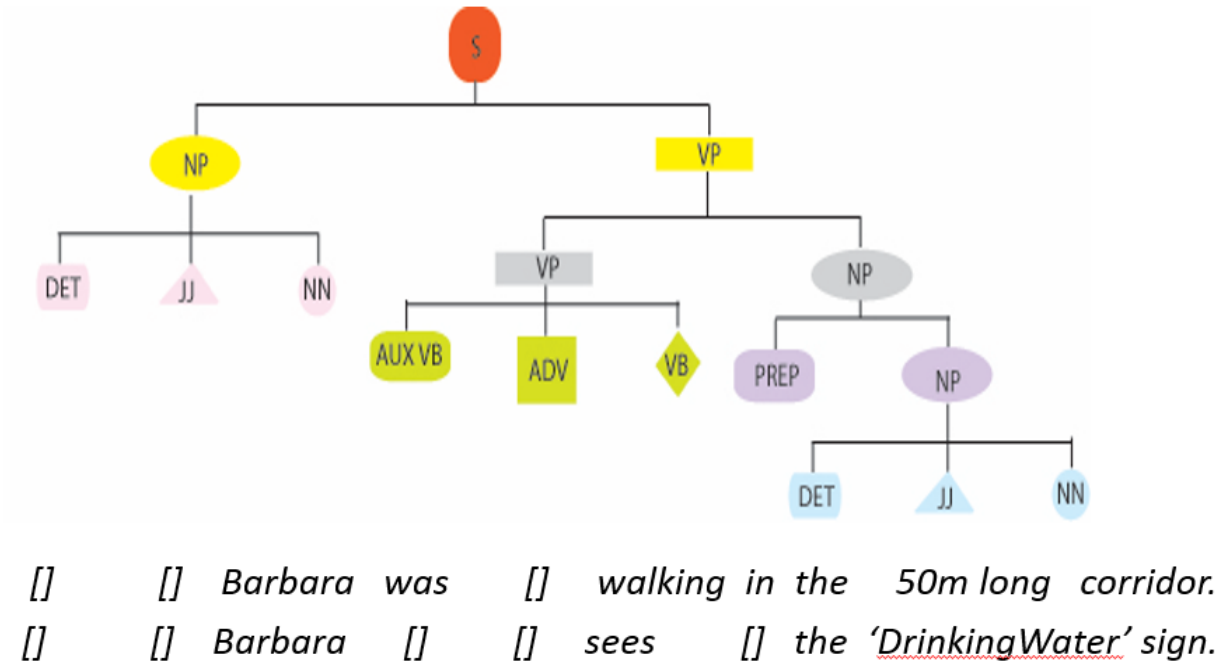

Fig 5.3 An example of Syntax Tree

The syntax tree can be generated from our system and can be used to understand the overall structure of the arrangement of lexicons used in sentence generation. The builtSyntaxTree predicate is responsible for the built of this tree.

An example of this is as follows:

```
?- builtSyntaxTree(
[
little,barbara, quickly, focus, on, drinking-water, sign,at,corridor,end,
present_continuous
],SyntaxTree).


SyntaxTree = [[[[]|little]|barbara],
[[[[is]|focussing]|quickly], on],
[[[the]|drinking-water]|sign],
[at, [[the|corridor]|end]]]]
```

Since the system is declaratively built, so any attribute of the syntax tree can be queried as shown in the example below. The various components are connected logically, thus, with querying directly in prolog, we can derive the particular component values. This can be used in order to retrieve the value of different analysis done corresponding to different identified objects.

```
?- grammar(
   [little, barbara, quickly,focus, on, drinking-water, sign, at, corridor,end,
   present_continuous]
   , [[[[]|little]|barbara], [[[[is]|focussing]|quickly], on], [OBJECT, LOCATION]]]).

LOCATION = [at, [[[the]|corridor]|end]],
OBJECT= [[[the]|drinking-water]|sign]
```

5.5 Sentence Generation

The syntax tree thus results in the expected sentence/text generation as output after the linearization of the syntax tree. The output goal of generation of text is achieved here. Some examples are shown below:

```
?- generateSimultaneous(
     [
     [[],barbara,[],walk,in,[],corridor,[],[],[],present_continuous],
     [three,persons,[],pass,to,[],left,[],[],[],simple_present],
     [one, person,[],pass,from,[],right,[],[],[],simple_present]
     ],Sentence
    ).
Sentence = 'While barbara is walking in the corridor ,
          three persons passes to the left and  one
          person passes from the right'
```

```
?- generateDescription(junction1,
|         [
|         [sign,board,[],[],on,[],right,[],[],[],simple_present],
|         [[],door,[],[],with,'Caution',sign,to,[],end,
           simple_present],
|         [[], corridor,[],[],to,[],right,towards,'Attrium',lobby,
           simple_present]
|         ],Sentence
|        ).
Sentence = 'At the junction1 , there is a sign board on the
right , a door with the Caution sign to the end and the
corridor to the right towards the Attrium lobby'
```

```
?- generateSimpleSentence([[],barbara,[],focus,on,exit,sign,at,
                corridor,end,simple_past],Sentence).
Sentence = 'barbara focussed on the exit sign at
        the corridor end'
```

```
?- generateDependentText([[[],barbara,[],walk,in,
                    large,corridor,[],[],[],
                    past_continuous],
                    [[],barbara,[],go,to,gift,
                     shop,[],[],[],
                     past_continuous]],Sentence
    ).
Sentence = 'barbara was walking in the corridor because
        barbara was going to the gift shop'
```

5.6 WHY PROLOG?

- More interactive.
- Direct logical specifications makes it more easier.
- Prolog applications are comparatively smaller to Java due to less complex libraries.
- Syntactic constraints are handled early and naturally, ensuring the expressibility of selected concepts.
- Order of adding content is flexible, allowing varied sentences without complicated search.
- Grammatical knowledge is stated once only, inside the generator.
- Enables efficient descriptions that refer implicitly to parameters from the context.
- Enables efficient descriptions that exploit the hearer's ability to recognize inferential links to material elsewhere in the sentence.

CHAPTER 6
IMPLEMENTATION

With all the predicates taken into account, the overall implementation of NLG engine looks like that in figure 6.1.

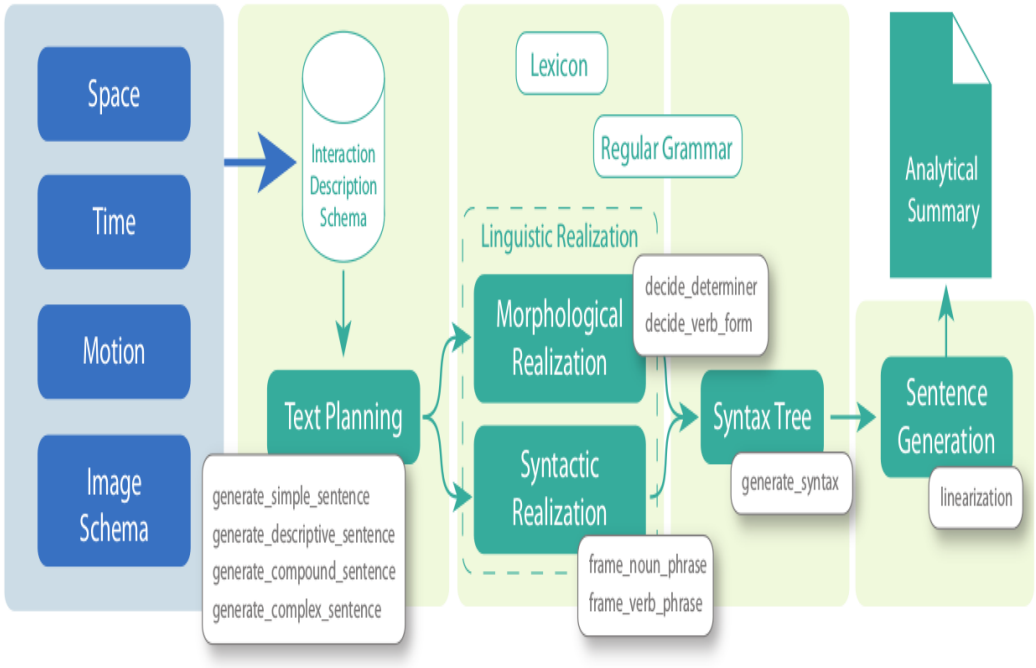

Fig 6.1 Implementation of NLG engine (Designed by: Jakob Suchan)

The above figure also represents an algorithmic logical flow of the system. Since, the system is declaratively built in prolog, it doesn't have any procedural algorithmic flow. It's components are connected logically to one another. The logical flow diagram is very well described in figure 6.1 which involves all the different predicates explicitly mentioned in the diagram. These predicates like frame verb phrase, frame noun phrase, generate simple sentence, generate descriptive sentence, decide_determiner have been discussed in Chapter 5 in detail with examples.

Fig 6.2 Big Picture of Entire Project

6.1 Real Case Scenario Application of The NLG Engine On Analysis of User 'Barbara':

- **DICTIONARY DATA FILE**

```
/* Noun(name,type), 25 Subjects*/
noun(barbara,living).
noun(mehul, living).

/*ObjectListOfNoun(name,type)., 14Objects, 10Directions*/
noun(hospital, definiteNonLiving).
noun(chairs, definiteNonLiving).
noun(trashbins, definiteNonLiving).
noun(sign, definiteNonLiving).
noun(corridor, definiteNonLiving).
noun(door, indefiniteNonLiving).
noun(lobby, definiteNonLiving).
noun(arms,definiteNonLiving).
noun(leg,definiteNonLiving).
noun(bell,indefiniteNonLiving).
noun('exit-sign', definiteNonLiving).
noun('drinking-water-sign',definiteNonLiving).
noun('caution-sign',definiteNonLiving).
noun('toilet-sign',definiteNonLiving).
noun(right, definiteNonLiving).
noun(left, definiteNonLiving).
noun(top, definiteNonLiving).
noun(bottom, definiteNonLiving).
noun(above, definiteNonLiving).
noun(below, definiteNonLiving).
noun(near, definiteNonLiving).
noun(far,definiteNonLiving).
noun(end, definiteNonLiving).
noun(beginning, definiteNonLiving).

/*verb(RootForm, Present, Past, Continuous) */
verb(walk,walks,walked,walking).
verb(pass,passes,passed,passing).
verb(focus,focusses,focussed,focussing).
verb(see,sees,saw,seeing).
verb(turn,turns,turned,turning).
verb(reach,reaches,reached,reaching).
verb(jump,jumps,jumped,jumping).
verb(run,runs,ran,running).
verb(stop,stops,stopped,stopping).
```

```
verb(dance,dances,danced,dancing).
verb(raise,raises,raised,raising).
verb(talk,talks,talked,talking).
verb(stand,stands,stood,standing).
verb(laugh,laughs,laughed,laughing).
verb(smile,smiles,smiled,smiling).
verb(stare,stares,stared,staring).
verb(point,points,pointed,pointing).
verb(shout,shouts,shouted,shouting).
verb(call,calls,called,calling).
verb(push,pushes,pushed,pushing).
verb(sit,sits,sat,sitting).
verb(kick,kicks,kicked,kicking).
verb(punch,puches,punched,punching).
verb(press,presses,pressed,pressing).
verb(go,goes,went,going).
```

➢ DOMAIN DESCRIPTION

```
person(X) :- subject(X).

/*objectInformation(Subject,SpatialRelation,PropertiesOfObject,Object) for Barbara*/
objectSpatialRelation(Subject,in,corridor) :- subject(Subject).
objectSpatialRelation(Subject,by,Person) :- subject(Subject), person(Person).
objectSpatialRelation(Subject,at,sign) :- subject(Subject).
objectSpatialRelation(Subject,to,left) :- subject(Subject).
objectSpatialRelation(Subject,to,right) :- subject(Subject).
objectSpatialRelation(Subject,[],trashbins) :- subject(Subject).
objectSpatialRelation(Subject,at,lobby) :- subject(Subject).
objectSpatialRelation(Subject,at,hospital) :- subject(Subject).
objectSpatialRelation(Subject,on,chairs) :- subject(Subject).
objectSpatialRelation(Subject,to,elevators) :- subject(Subject).
objectSpatialRelation(Subject,[],arms) :- subject(Subject).
objectSpatialRelation(Subject,with,leg) :- subject(Subject).
objectSpatialRelation(Subject,forward,door) :- subject(Subject).
objectSpatialRelation(Subject,[],bell) :- subject(Subject).
objectSpatialRelation(Subject,at,'exit-sign') :- subject(Subject).
objectSpatialRelation(Subject,at,'drinking-water-sign') :- subject(Subject).
objectSpatialRelation(Subject,at,'caution-sign') :- subject(Subject).
objectSpatialRelation(Subject,at,'toilet-sign') :- subject(Subject).
objectSpatialRelation(Subject,to,top) :- subject(Subject).
objectSpatialRelation(Subject,to,bottom) :- subject(Subject).
objectSpatialRelation(Subject,to,far) :- subject(Subject).
objectSpatialRelation(Subject,to,near) :- subject(Subject).
```

```
objectSpatialRelation(Subject,to,above) :- subject(Subject).
objectSpatialRelation(Subject,to,below) :- subject(Subject).
objectSpatialRelation(Subject,to,end) :- subject(Subject).
objectSpatialRelation(Subject,to,beginning) :- subject(Subject).

/*objectInformation(Subject,PropertiesOfObject, Object) for Barbara*/
objectInformation('50 metres long',corridor).
objectInformation([],Person) :- person(Person).
objectInformation([],sign).
objectInformation([],left).
objectInformation([],right).
objectInformation(two,trashbins).
objectInformation('Attrium Lobby', lobby).
objectInformation('Dallas',hospital).
objectInformation(green,chairs).
objectInformation(blue,elevators).
objectInformation(two,arms).
objectInformation(left,leg).
objectInformation(golden,door).
objectInformation(first,bell).
objectInformation([],'exit-sign').
objectInformation([],'drinking-water-sign').
objectInformation([],'caution-sign').
objectInformation([],'toilet-sign').
objectInformation([],top).
objectInformation([],bottom).
objectInformation([],beginning).
objectInformation([],end).
objectInformation([],far).
objectInformation([],near).
objectInformation([],above).
objectInformation([],below).
```

- **HIGH LEVEL ANALYSED DATA INPUT FOR BARBARA**

```
/*Subject Name*/
subject(barbara).

/*action(Subject, Action, Object) of barbara*/
action(barbara, focus, sign,11).
action(barbara, turn, left,12).
action(barbara,see,trashbins,13).
action(barbara, push, door,14).
action(barbara, raise, arms,15).
action(barbara, kick, legs,16).
```

```
action(barbara, sit, chairs,17).
action(barbara, jump, hospital,18).
action(barbara, press, bell,19).
action(barbara, reach, lobby,10).

/*(Complex) SimultaneousAction(Subject Name, Action, Object) of barbara*/
action(barbara,walk,corridor,1-1).
action(barbara, pass, mehul,1-2).
action(barbara, kick, tom,1-3).
action(barbara, push, swati,1-4).

/*Compound Sentence Generation*/
compound(action(barbara, pass, mehul, 112), action(mehul, reach, sign, 115)).
compound(action(barbara, turn, mehul, 114), action(mehul, pass, sign, 117)).
compound(action(barbara, see, mehul, 116), action(mehul, turn, sign, 118)).

/*actionInformation(Subject,PropertiesOfMotion,Motion) for barbara*/
actionInformation(barbara,quickly,walk).
actionInformation(barbara,[],pass).
actionInformation(barbara,[],focus).
actionInformation(barbara,[],turn).
actionInformation(barbara,[],see).
actionInformation(barbara,[],reach).
actionInformation(barbara,strongly,kick).
actionInformation(barbara,[],press).
actionInformation(barbara,excitedly,jump).
actionInformation(barbara,[],sit).
actionInformation(barbara,[],raise).
actionInformation(barbara,[],push).
```

- **GENERATED SUMMARY FOR USER BARBARA.**

The sample summary generated for Barbara is:

Barbara is walking in a 50m long corridor. Barbara sees the 'Drinking-Water' sign at the corridor end. While Barbara is walking in the corridor, Mehul passes Barbara. Barbara turns left at the end of the corridor. Barbara reaches the blue elevators.

A snapshot of the same is as follows:

```
?- generateSummary(barbara,simple_present,Simple,Complex,Compound).
Simple = [['barbara focusses at the sign'], ['barbara turns to the left'], ['bar
bara sees the two trashbins'], ['barbara pushes forward a golden door'], ['barba
ra raises the two arms'], ['barbara sits on the green chairs'], ['barbara jumps
excitedly at the Dallas hospital'], ['barbara presses a first bell'], [...]|...]
,
Complex = [['barbara passes by mehul and mehul reaches at the sign'], ['barbara
turns by mehul and mehul passes at the sign'], ['barbara sees by mehul and mehul
 turns at the sign']],
Compound = [['While barbara is raising the two arms , mehul raises the two arms
'], ['While barbara is walking quickly in the 50 metres long corridor , mehul p
asses by barbara'], ['While barbara is walking quickly in the 50 metres long cor
ridor , mehul passes by barbara'], ['While barbara is passing by mehul , mehul
 turns to the right'], ['While barbara is passing by mehul , mehul turns to the
right'], ['While barbara is kicking strongly by tom , tom pushes by barbara'],
['While barbara is kicking strongly by tom , tom pushes by barbara'], ['While
barbara is pushing by swati , swati pushes by barbara']].
```

Fig 6.3 Snapshot of summary generated for Barbara

CHAPTER 7
EMPIRICAL TEST

For the empirical test, we built a spatio-temporal domain based, moving image analysed data corpus. The corpus consists of 25 subjects, 500 interactions, 53 spatio-temporal domain based facts. We analysed generation time of 25 summaries on two basis - different tenses and type of clause structure of generated sentences for each summary(simple, complex, compound) . The test was done on ubuntu 14.04, Intel® Core™ i7-3630QM CPU @ 2.40GHz × 8. These figures suggest that the spatio-temporal domain based, declarative model built for language generation, is highly robust to render paragraphs in less than a second.

The time records noted for different tenses and different clause structures are shown in fig 7.1 and fig 7.2 respectively. The average time of generation of these records and other results are discusssed in Chapter 8 ‘Result’.

SUBJECT NAME	NO. OF INTERACTIONS	NO. OF SIMPLE SENTENCES IN SUMMARY	NO. OF COMPOUND SENTENCES IN SUMMARY	NO. OF COMPLEX SENTENCES IN SUMMARY	TIME FOR SUMMARY IN SIMPLE PRESENT, PAST OR FUTURE	TIME FOR SUMMARY IN CONTINUOUS PRESENT, PAST OR FUTURE	TOTAL LENGTH OF SUMMARY (Average Length of Sentence is 17.6 tokens)
barbara	20	13	3	8	79	76	24
mehul	32	10	3	7	73	92	20
jakob	16	11	3	16	88	91	30
swati	33	15	4	6	72	99	25
bob	21	16	3	8	73	89	27
tim	20	11	2	4	80	81	17
elisa	23	15	3	21	71	84	39
george	16	10	3	6	76	82	19
kin	18	13	3	8	82	73	24
raj	21	13	3	17	73	78	33
ana	20	15	3	12	81	92	30
javi	21	13	3	8	75	86	24
jimmy	15	11	2	11	76	78	24
tom	27	15	3	8	79	76	26
meera	22	14	3	11	70	89	28
ram	19	13	3	8	76	88	24
rashi	18	13	3	8	87	87	24
jack	17	13	3	11	72	81	27
john	15	10	3	14	85	95	27
joy	17	13	3	15	96	89	31
jason	18	12	3	15	81	83	30
tina	18	13	4	7	71	76	24
hoYa	15	10	3	8	76	81	21
shiv	22	14	3	11	71	82	28
harshita	16	10	3	16	82	84	29
					Average=77.8	Average=84.48	Average Length of Summary= 26.2
Average	20	12.64	3	10.56	Min=70	Min=73	Min=17
	No. Of Subjects = 25.				Max=96	Max=99	Max=39
	No. of Domain Facts=53.						
	No. Of Interactions=500						
	Average=20						

Fig 7.1 Empirical Test Records.

- Average time taken for generation of sentences
- Different clause structures on a scale of 100 generated sentences
- Time for :-
    - one simple sentence being approximately 0.52/msec,
        - one compound sentence being 1.23/msec
            - one complex sentence being 1.32msec

No Of Simple	No. Of Compound	No. of Complex	Time Taken	Rate	Time Taken	Rate
			Simple Form Of Tense		Continuous Form Of Tense	
100	0	0	52	0.52	54	0.54
0	100	0	123	1.23	126	1.26
0	0	100	132	1.32	14	1.4

Fig 7.2 Empirical Test Records For Different Clause Structures

The details of the entire corpus are mentioned in table 7.1 and table 7.2. The results are discussed in chapter 8 ‘Result’.

CHAPTER 8
RESULT

- ✓ With an average of 20 interactions, on an average 26.2 sentences per summary, with 17.6 tokens as the average length of each sentence, the system took average time of 77.8 msec and 84.48msec for generating summaries in simple present/past/future and continuous present/past/future respectively for all subjects, as shown in figure 7.1.

- ✓ We also found the average time taken for generation of sentences with different clause structures on a scale of 100 generated sentences, the time for one simple sentence being approximately 0.52/msec, one compound sentence being 1.23/msec and one complex sentence being 1.32msec as discussed in figure 7.2.

CHAPTER 9
CONCLUSION AND FUTURE WORK

9.1 CONCLUSION

- ✓ The NLG engine developed during the course of my internship is based on dynamic spatio temporal data domain.
- ✓ This engine is highly scalable. It can hold any length of data.
- ✓ The same predicates which are defined in prolog can be reused for generating more complex functionalities.
- ✓ The system can render paragraphs in seconds.
- ✓ It has reduced a lot of human effort.
- ✓ The same data input is able to produce different reports for different users. By reducing or increasing the amount of details, this is achievable.
- ✓ The system is efficient to generate analysis summaries and narrative reports in different tenses involving different clause structures of simple, complex, compound.

9.2 FUTURE WORK

- ✓ Coming up with 'Discourse Model' for language generation in the same domain.
- ✓ Sentences with Pronouns, Numbers (Singular/Plural).
- ✓ Forming Interrogative, Reflexive, Exclamatory, Negation.
- ✓ Increase the features of Grammar Involved.
- ✓ Building Natural Language Based, Declaratively built, Question-Answering System.

CHAPTER 10
REFERENCES

1. *D. Roy and E. Reiter. 2005. Connecting language to the world. Artificial Intelligence.*
2. *M Mitchell, K van Deemter, E Reiter (2010). Natural Reference to Objects in a Visual Domain. Proceedings of INLG-2010, pages 95-104*
3. *Yuan ren, Kees van Deemter, Jeff Z. Pan. Charting the potential of description logic for the generation of referring expressions. Proceedings of INLG-2010.*
4. *Barbara Di Eugenio, Michael Glass, Michael J. Trolio, Susan Haller. Simple Natural Language Generation and Intelligent Tutoring Systems. Proceedings of the 2005 conference on Artificial Intelligence in Education: Supporting Learning through Intelligent and Socially Informed Technology.*
5. *Albert Gatt and Ehud Reiter, 2009, SimpleNLG: A realisation engine for practical applications, ENLG '09: Proceedings of the 12th European Workshop on Natural Language Generation*
6. *R Kutlak, C Mellish, K Van Deemter, 2013, Content Selection Challenge-University of Aberdeen Entry, ENLG '13 Proceedings of the 14th European Workshop on Natural Language Generation*
7. *Jette Viethen, Margaret Mitchell, Emiel Krahmer 2013, Graphs and Spatial Relations in the Generation of Referring Expressions, ENLG '13 Proceedings of the 14th European Workshop on Natural Language Generation.*
8. *Ehud Reiter and Robert Dale, 2000, Building Natural Language Generation Systems, Cambridge University Press, Cambridge, UK.*
9. *Albert Gatt and Ehud Reiter, 2009, SimpleNLG: A realisation engine for practical applications, ENLG '09: Proceedings of the 12th European Workshop on Natural Language Generation*
10. *R Kutlak, C Mellish, K Van Deemter, 2013, Content Selection Challenge-University of Aberdeen Entry, ENLG '13 Proceedings of the 14th European Workshop on Natural Language Generation*
11. *Sina ZarrieÃŸ Kyle Richardson, 2013, An Automatic Method for Building a Data-to-Text Generator, ENLG '13 Proceedings of the 14th European Workshop on Natural Language Generation*
12. *Jette Viethen, Margaret Mitchell, Emiel Krahmer, 2013, Graphs and Spatial Relations in the Generation of Referring Expressions, ENLG '13 Proceedings of the 14th European Workshop on Natural Language Generation.*
13. *F. L. Aldama. The Science of Storytelling: Perspectives from Cognitive Science, Neuroscience, and the Humanities. Projections, 9(1):80–95, 2015. doi: doi:10.3167/proj.2015.090106.*
14. *J. F. Allen. Maintaining knowledge about temporal intervals. Commun. ACM, 26(11):832–843, 1983. ISSN 0001-0782*

15. *J. Bateman and M. Zock. Natural language generation, Oxford handbook of computational linguistics, pages 284–304, 2003.*
16. *J. A. Bateman. Situating spatial language and the role of ontology: Issues and outlook. Language and Linguistics Compass, 4(8):639–664, 2010. doi: 10.1111/j.1749-818X.2010.00226.x.*
17. *M. Bhatt. Reasoning about Space, Actions and Change: A Paradigm for Applications of Spatial Reasoning. In Qualitative Spatial Representation and Reasoning:Trends and Future Directions. IGI Global, USA, 2012. ISBN ISBN13: 9781616928681.*
18. *D. Waller and L. Nadel. Handbook of Spatial Cognition. American Psychological Association (APA),2013. ISBN 978-1-4338-1204-0.*
19. *H. Yu, N. Siddharth, A. Barbu, and J. M. Siskind. A Compositional Framework for Grounding Language Inference, Generation, and Acquisition in Video. J. Artif. Intell. Res. (JAIR), 52:601–713, 2015. doi: 10.1613/jair.4556.*
20. *M. Bhatt, C. P. L. Schultz, and M. Thosar. Computing narratives of cognitive user experience for building design analysis: KR for industry scale computer-aided architecture design. In C. Baral, G. D. Giacomo, and T. Eiter, editors, Principles of Knowledge Representation and Reasoning: Proceedings of the Fourteenth International Conference, KR 2014, Vienna, Austria, July 20-24, 2014. AAAI Press, 2014. ISBN 978-1-57735-657-8.*