THE UNIVERSITY OF TEESSIDE
SCHOOL OF COMPUTING
MIDDLESBROUGH
CLEVELAND TS1 3BA

VOCAL INTERFACE TO A COMPUTER ANIMATION SYSTEM

BSc (Honours) Computer Studies

March 2004

Julien Loiseaux

Supervisor: Prof. Marc Cavazza
Second Reader: T. P. Davison

Abstract

The aim of this project is to develop an implemented prototype of a vocal interface to a computer animation system.
The main purpose of building such an interface is to improve the interaction between human beings and artificial actors using speech recognition.
This interface is embedded in the Interactive Story Telling System, used by the university, which is based on the Unreal ™game [1] engine.
An analysis of the Interactive Story Telling System especially at the speech recognition and Natural Language Processing layers is provided.
A research on both Speech Recognition systems and Natural Language Processing is conducted to find out what is the best way to get the best performance.
We look especially at the different modes of speech recognition and the accuracy of speech recognition systems.
Regarding the Natural Language Processing approach, we look at a brief history of Natural Language Processing in which several concepts used to build past systems is reviewed, before introducing its main concepts and its embedding in the interactive storytelling system.
A corpus (set of sentences) of 300 sentences has been implemented using the ©BabelTech lexicon editor.
However three versions of the corpus were implemented. The first one is syntactic based; the second is thematic based , which gave low recognition results because of their complexity. The third and final version is based on plain text and alternatives.
To increase the flexibility of the system an extension of those sentences is provided in the Corpus using alternatives.
The Natural Language Processing is dealt using templates implemented using the Ear SDK from ©BabelTech.
Templates are based on speech acts, which aim at associating a list of keywords

[1] Epic Games

to a specific meaning.
The first version of the templates is containing all the themes and their relevant speech acts.
A first working prototype based on six relevant themes is provided reaching the performances expected within the given time.
Tests have been carried out at all the stages of the project using applications provided by Mr. Steven Mead and Mr. Fred Charles.
A review of Testing techniques and results has been provided.

Acknowledgements

First, I would like to express my gratitude to my supervisor, Prof. Marc Cavazza, whose expertise, understanding, and patience helped me to go through this project.

I would like also to thank Mr. Fred Charles and Mr. Steven Mead for the assistance they provided at all levels of the project.

Contents

Chapter 1

Introduction

To start with, Speech is one of the many ways a human being can interact with another one. The aim of speech recognition is to provide an interface to allow a human being to interact with a machine using speech.

1.1 Speech recognition

Speech recognition can be defined as

> "the process of converting an acoustic signal, captured by a microphone or a telephone, to a set of words."
>
> (V. Zue and R. A. Cole. Spoken language input)[16].

Once recognized, the words or set of words recognized can be used as input to any number of different applications. The recognized words can be used to control computers or other machines, for data entry and for text processing.

1.2 Natural Language Processing

Natural Language Processing (NLP) is intending to analyse and represent naturally occurring texts to achieve human-like language processing:

> "NLP is a range of computational techniques for analysing and representing naturally occurring texts at one or more levels of linguistic analysis for the purpose of achieving human-like language processing for knowledge intensive applications."
>
> Woojin Paik. Natural language processing[13].

1.3 Interactive Story Telling System overview

1.3.1 The system

The aim of the Interactive Story Telling System (ISS) is to create dynamic narratives with which the user can interact. The system is divided into three layers, the user layer, the character layer and the 3D environment layer as described in the figure 1.1: The user layer will be the most exploited, this layer is made up itself into 2 layers: the speech recognition layer and the Natural Language Processing (NLP) layer [11].

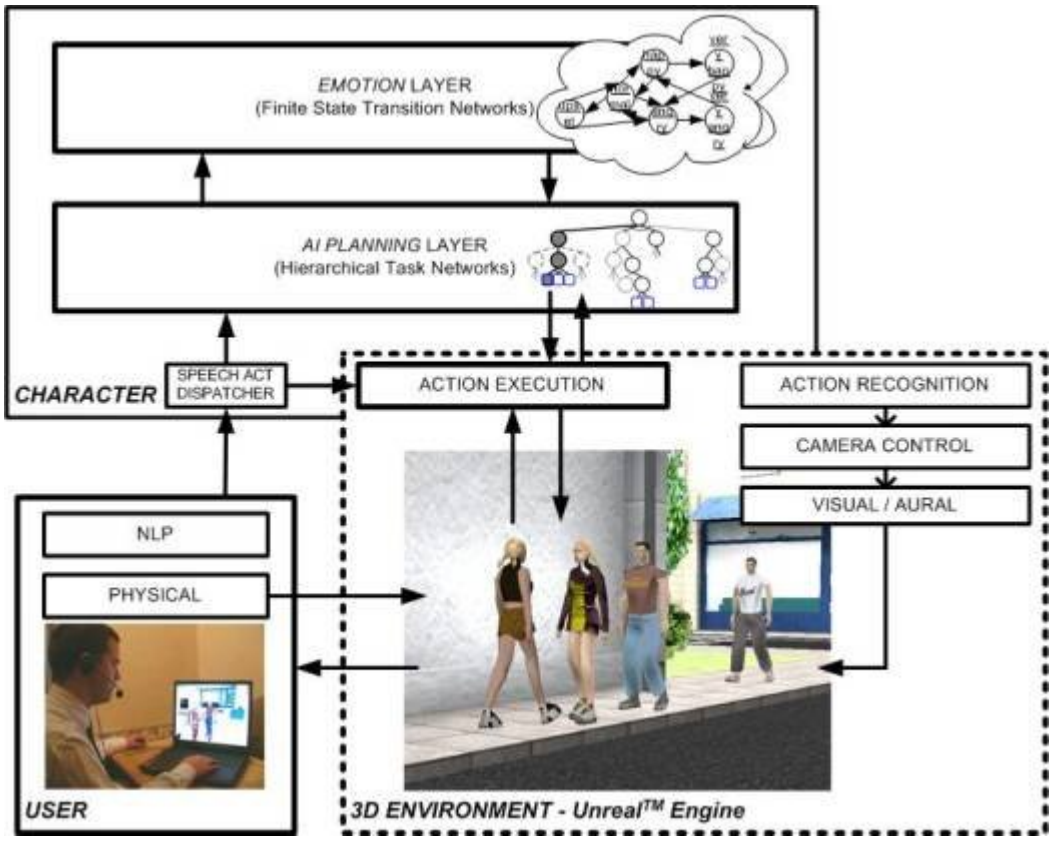

Figure 1.1: Character-based Interactive Storytelling[3]

1.3.2 The speech recognition layer

The speech recognition layer is providing tools to develop a Finite State Grammar (FSG), which is the set of sentences (corpus) to be recognized by the automatic speech recognition (ASR) system.

1.3.3 The natural language processing layer

This layer aims at attempting to map the output from the speech recognition layer and carries out the actual speech recognition act [11]. It is based on templates that contain all the sentences to be validated from the ear system and all the speech acts of the scenario.

1.4 Aims and contributions

The main aim of this project is to build a prototype of a vocal interface with an efficient speech recognition accuracy within an interactive storytelling system. To do so, here are the contributions:

- Analysing the interactive storytelling system.
- Finding ways to improve the accuracy and the performance of the vocal interface by:
  - Studying and understanding the basics of speech recognition and natural language processing principles.
  - Having a look at existing speech recognition systems.
  - Implementing those concepts in the system.
- Producing a Corpus with about 300 sentences which fix to the plot used by the Interactive Storytelling System by:
  - Doing a Review of James Bond Movies Villains Sentences
  - Extending those sentences to make the system more flexible

Another goal is to propose a Methodology which can be re-used in other Speech Recognition systems by summarizing the different steps which lead to an efficient system.

1.5 The structure of this report

In this report, after dealing with the methodology and research, we will have a look on how to build a corpus, the embedding of the system in Unreal^{TM1} to finally talk about the testing and refinement of the product.

[1]Epic Games

Chapter 2

Methodology

Constraint: As a part of a scientific publication this project was managed by a non-negotiable main deadline on mid April.
A first version of the Corpus had to be handed in on mid December.
A first beta version of the system had to be operational on the beginning of March.

The project development has been split into several steps.
The first was to do some research on speech recognition and natural language processing to see how the problem can be solved and if people have already solved this kind of problem.
By doing the research, a learning of the development tools in which the system has to be implemented was done as well.
Regarding the design an analysis on how we can deal with the problem was done, and different ways to solve the problem have been proposed.
The three main steps was to find a way to build the corpus first, next implement the templates and finally mix them together to have the best efficiency.
Although a testing and refinement phase is necessary at the end of the project, a lot of tests were done on going the project. The Figure 2.1 is illustrating the different methodology steps.

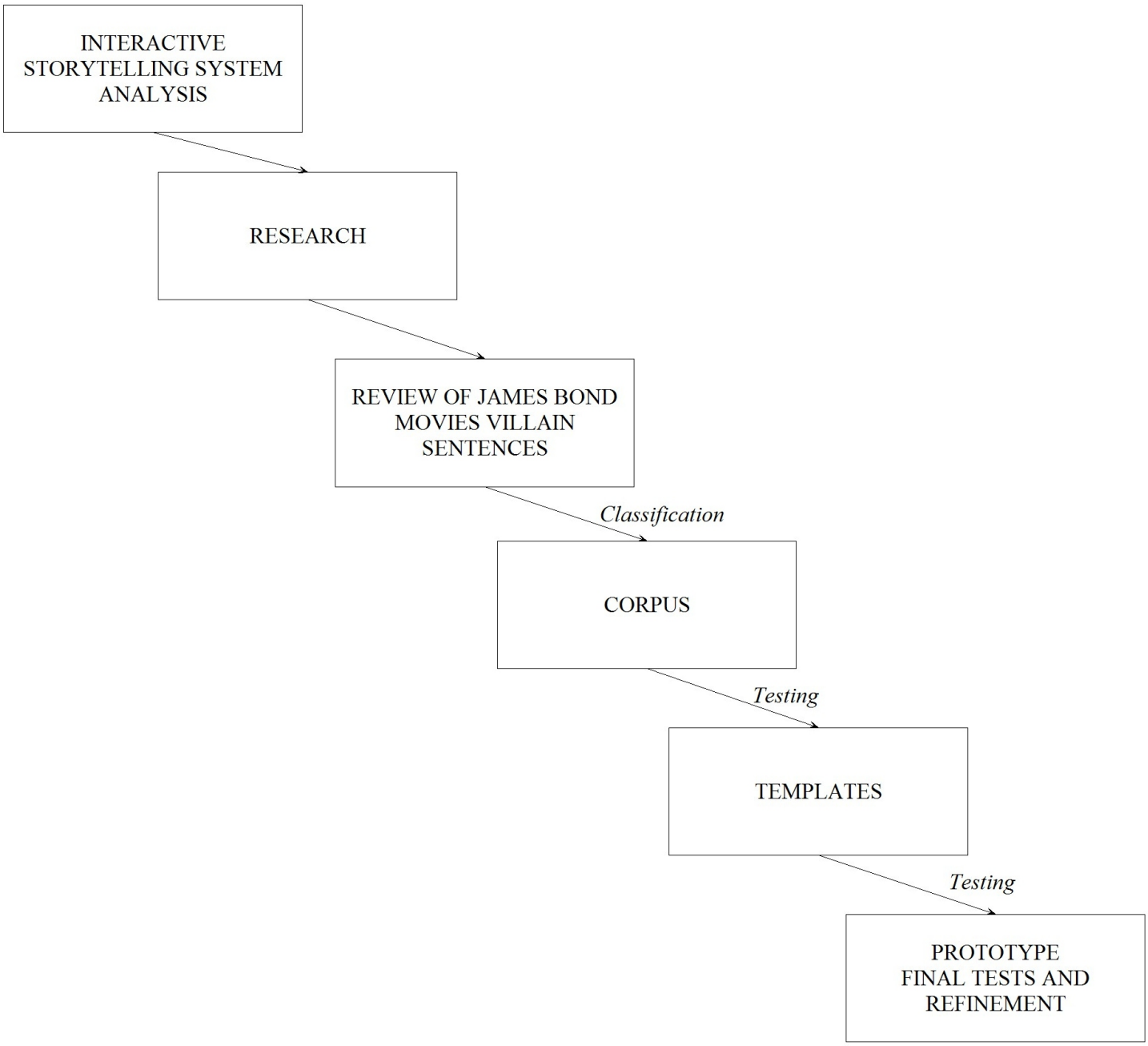

Figure 2.1: Methodology Chart

Chapter 3

Research

3.1 Speech Recognition System

In this part, we will focus on speech recognition systems in order to find out a system that best fix with the project.

3.1.1 Mode

There are several modes in which a speech recognition system can be used [7]:

- **Dependent systems:** In this system, the system has to be trained and accustomed to the voice of the user, recording sessions by the user is necessary. This system cannot be used in our case because it is not always the same user who is using the system.
- **Independent systems:** Those systems do not require a training phase, which fix with the aim of the project. However we are losing a little bit of accuracy.
- **Isolated Word Recognition:** In this recognition mode, each word is surrounded by a silence, the system is not required to know the beginning and the end of each word, each word is compared to a list of word models, it is the less greedy system in term of CPU requirement.
- **Continuous Speech Recognition:** Contrary to the previous mode, this mode requires more CPU and is user-friendlier. It is based on the assumption that the system is able to recognize a sequence of word in a sentence. We are losing in recognition accuracy but it is user-friendlier.

- **Keyword Spotting:** This is the more interesting speech recognition mode. Indeed it is a mix between continuous and isolated speech recognition and it improves the accuracy. Those systems are able to recognize words and group of words corresponding to a particular command or speech acts. For example in a Video Renting Machine, if we assume that the user asks for western movies, the user has many different ways to ask his question, it could be: "Show me the list of western movies" or "Can you please give me the list of movies with cowboys". The words "western" and "cowboys" are corresponding to a specified action which can be in this example to display the list of western movies. In our example we can consider the keyword spotting as "multiple" since a list of several keywords stands for the same meaning.

3.1.2 Existing systems

Speech recognition is starting to have a lot of applications using a dependent recognition system. For example, speech to text software used for pc dictation or to control pc operating systems. Speech recognition is also used in telephony and calls centres.

3.1.3 Accuracy

Nowadays, there is no speech recognition system that has 100 per cent accuracy. The accuracy of the speech recognition system that we are using is relying on those statements:

- **Vocabulary size:** The size of the vocabulary is a really important point in speech recognition, the more the size of the vocabulary is important the more the user can talk in different manner. However, if there are too many words the system is more led to make errors. If two words of different meanings have close pronunciations it can raise problem in the system. This means that we need to identify which vocabulary is more likely to be used by the user.

- **Language Models:** The way we deal with syntactic and semantic constraints is an important feature for the accuracy of the system. How the words and the set of words are split within the speech acts is important.

  Other accuracy improvement features will be dealt further in this report.

3.2 Natural Language Processing approach

3.2.1 Purpose

Natural Language Processing (NLP) is intending to analyse and represent naturally occurring texts to achieve human-like language processing [4]. In other words, NLP helps to define relevant logical grammar rules considering semantic, syntactic and lexical features. Natural language is used in a lot of human computer interaction systems.

3.2.2 History

This part is mainly based on the The Handbook of artificial intelligence by Avron Barr and Edward A. Feigenbaum [1].
The concept was introduced by Dr. Alan Turing in 1950[9], a British mathematician by proposing a test for intelligence. The aim is to answer to the question "Can this machine trick the human to think that its human", the machine used to do this test was a natural language program which attempted to converse with a human subject. The subject is placed in a room and is not aware whether he is talking to a human or a computer. The communication was done by typing on a terminal.
The first researches done in Natural Language Processing, during the middle of the 20th century [1], were about finding a way to resolve worldwide translation problems. This research called machine translation attempted to simulate the functions of a human translator using a computer. It was based on a search in a bilingual dictionary by finding an equivalent word in the output language. After processing sentences using this way, this system tried to put the output strings in the output language's word order. Problems occurred and this project was abandoned.
In the 1960s, researchers tried to find new ways to deal with the machine translation problem by viewing the human language as a involving knowledge of different kinds: The structure of the sentence, the meaning of the words, the rules of conversation and etcetera. The goal was to make a computer performed as a human, by modelling the human language processing using knowledge storing.
The early Natural language software was based on processing the language only in a specific domain. The main programs were Green's BASEBALL, Lindsay's SAD-SAM, Bobrow's STUDENT, and Weizenbaum's ELIZA.
Here is an example of an early application of NLP: ELIZA (Weizenbaum, 1966).

ELIZA aimed at enabling the user to discuss with the machine. The user played the role of a patient and the machine a psychotherapist.

Sample Dialogue with Eliza[10] :

...
USER: Men are all alike.
ELIZA: In what way?
USER: They're always bugging us about something or other.
ELIZA: Can you think of a specific example?
USER: Well, my boyfriend made me come here.
ELIZA: Your boyfriend made you come here.
USER: He says I'm depressed much of the time.
ELIZA: I am sorry to hear you are depressed.
USER: It's true, I am unhappy.
ELIZA: Do you think coming here will help you not to be unhappy?
USER: I need some help; that much seems certain.
ELIZA: What would it mean to you if you got some help?
USER: Perhaps I could learn to get along with my mother.
ELIZA: Tell me more about your family.
...

ELIZA was based upon predeclared keywords and pattern matching that gave information about the objects and the relationships between the latest. The results of such a system were impressive because of a limited discourse.

In the 70s Knowledge based Natural Language systems were introduced by dealing with both semantics and syntactic aspects of Natural Language processing. Those systems are most based on the theory of generative grammar introduced by Chomsky (1957). The idea was to parse the grammar of the sentences to determine their meaning in order to generate an appropriate response. By determining the functions of the words, the system builds a data structure that attempts to get the meaning of the sentence. But the practical use of grammar in natural language systems is complex and based upon the definition of the parser.

In order to be able to answer about the sample of rocks brought back from the moon, William Wood's LUNAR program were one of the first NL program which attempts to deal with the problems of English grammar using an augmented network parser. By integrating syntactic and semantic analysis with a body of a world-limited domain, those kinds of systems dealt with more complex aspects of language and discourse than previous programs.

The idea was to represent knowledge in a procedural way within the system. Semantics were integrated as programs in a computer language, this is called, procedural representations, in other words, it aimed at associating the definition of the words as actions executed by program fragments.
Semantic networks, which aim at linking parts of world knowledge together through semantics, have also been used in a lot of Natural language program (MARGIE and SAM (Schank 1975; Schank and Abelson 1977)).
During the 1980s empiricism and finite state models went back from the 50s on account of that the IBM Thomas J. Watson research centre introduced the rise of probabilistic models of speech recognition.
In 1994, the British national corpus was made available[12]. Now, the World Wide Web is used as a huge hyper linked corpus. Currently a lot of research is done in Natural Languages and due to the improvement of the computer performance some areas starts to be commercial. The current approaches in Natural language processing are often a combination of rule, statistical and corpus based methods.

3.2.3 Natural Language Processing concepts

The following concepts are key points for the analysis phase of the project [14]:

- **Morphology:**
  It is the way the words are constructed (prefixes and suffixes). A system has to differentiate for example the plural from the singular (e.g. flower / flowers).

- **Syntax:**
  It is how the relationships between the words are structured.
  The system has to be able to know the order of the words in a sentence. For example without considering syntax, a system can output "I am cannot be serious". Although syntax is not the meaning, word order is important because the sequence of words helps to determine their functions.

  > "Syntax can be defined as the arrangement of patterning of words"
  >
  > George W. Smith, Computer and human language[15].

- **Semantics:**
  It stands for the meanings of words, sequence of words and expressions. In the sentence "How would I know Mr. Bond?", the system has to be able to

associate expressions to a meaning, in this example it could be the sequence "How would I know" associates with the meaning denial and the expression Mr. Bond associates with the meaning actor.

> "Semantics constructs are usually more specific than syntactic rules and often resolve syntactic ambiguities."
>
> George W. Smith, Computer and human language[15].

.

- **Discourse:**
  It embodies the relationships across different sentences or thoughts (contextual effects).
- **Pragmatic:**
  It is the studies of how language is used to achieve specific goals.
- **Ambiguity[10]:**
  Ambiguity is an important issue in NLP; the issue is that a sequence of words can have different meanings. The expression "of course" can have different meanings it can stands for "yes, I agree" or ironically "no I disagree". Those problems can be resolved by using speech acts that allow the system to deal with the consequences of the speech. But we have to bear in mind that speech acts can be in some cases ambiguous; indeed one phrase can correspond to several speech acts[15].

3.2.4 Natural Language Processing in an Interactive Storytelling system

NLP is an important feature in an interactive storytelling system. This part is mainly based on interactive storytelling publications by Marc Cavazza, Fred Charles, and Steven J. Mead.

> "Interactive storytelling can be seen as a natural extension of the implementation of autonomous actors. As virtual characters become more intelligent, the action can increasingly rely on their automatic behaviour, generating a larger diversity of story than with current authoring methods. This dynamic computation of the action also makes

> possible various forms of user intervention, whose consequences on the story can then be propagated, as the plot is re-computed."
>
> Marc Cavazza, Fred Charles, and Steven J. Mead, Interactive Storytelling: From Computer Games to Interactive Stories[5].

Natural language in interactive storytelling is used as a paradigm for influence of plans that are used to drive the behaviour of characters in the story[11]. The main point is that the system aims at influencing the behaviour of characters rather than instructed them like in a conventional natural language system.
The user is interfering with characters to advice them. A planning system is used to drive the characters and modify the story, which is generated from the interaction between the character plans. This planning system is mainly character-based and represents each character role in the story. To do so, the system is using a knowledge representation called Hierarchical Task Networks (HTNs) which describes the behaviour of each character in the story.
The system is supporting re-planning and interleaving of planning and execution enabling an agent to re-plan new solutions as the situation is altered due to other agents or user interaction. Indeed, an agent task network can be directly searched using a real-time variant of the graph-search algorithm AO*[4]. Agent plans are generated as the semantics of the Natural Language instructions.

There are two main interactions within the system; one is physical interaction the other natural language interaction. Physical interaction is about allowing the user to drop or pick up resources, which modify the plot of the story. Using Natural Language interaction, the user is able to interfere with the story and modify characters plans. Although the user is considering as an active spectator by influencing and assist the development of the story, conventional use of speech recognition as character controlling (e.g. ordering a character to move from a place to one another) is not considered.

As briefly specified in the introduction the system has two layers the speech recognition layer the natural language processing layer. As the input can modify several stages of the planning process, the communicative nature of the input has to be identified. To do so, speech acts are used to categorize the Natural language input. The semantic of the speech acts is compared to the sub-goal node in the agent's plan.
The natural language processing layer aims at attempting to map the output from

the speech recognition layer and perform speech act recognition which influences the HTNs.
The system will attempt to identify the surface form of the advice then it will take the semantic information to produce a speech act. The system has to identify the context in which the utterance is presented and interpret it accordingly. The interpretation of a speech act is not only modifying the plot of the story but it is also depending on the current plan of the story.

3.3 Outcome

Here is a list of the main statements resulting from the research, that has to be considered for the implementation:

- **Speech Recognition system:**
  The speech recognition system will be based on the keyword spotting principles (See 3.1.1).

- **Accuracy:**
  The corpus has to have a large set of flexible sentences and a high specific vocabulary(See 3.1.3).

- **Grammar Validation:**
  As proposed in the William Woods LUNAR program (See 3.2.2), a way to validate the basic english syntactic rules of the sentences can be attempted.

- **Discourse:**
  The discourse (See 3.2.3) has to be considered, it will be managed by regrouping sentences by themes.

- **Speech Acts:**
  The Natural Language Processing of the Interactive Storytelling system will be managed using speech acts (See 3.2.4) by considering ambiguities and NLP concepts(See 3.2.3).

Chapter 4

Building the corpus

4.1 Aim of the Corpus

A corpus in speech recognition is a set of sentences; it aims at referencing all the sentences that the user can say. So, for one sentence we have to consider other ways to say the sentence to make the system flexible. Three versions of the corpus were implemented. The first one is syntactic based, the second is thematic based. The third and final version is based on plain text and alternatives.

4.2 The tool

To build the corpus a finite state grammar development tool "©BabelTech lex editor" is used. It is based on a mark-up language, which allows us to build a speech structure. It has some interesting features to be considered such as, alternatives and optional structuring:

- **Classification:** The mark-up language allows a classification of the different utterances classes: Example: <noun> <verb> <actor> and etcetera.
- **Alternatives:** The tag alt("word1" "word2")alt aims at providing alternatives for a given meaning or a given grammatical type. For example: alt ("james" "mister_bond" "james_bond")alt are the different ways to say James Bond.
- **Sequence:** A sequence helps at building sentences by associating different utterances or classes. Example: seq("hello" "my" "name" "is" <actor>)seq will produce for instance the sentence "hello my name is James Bond".

- **Optional:** The optional tag aims at defining some part of the sentences that can be said or not. Example: seq(opt ("hello")opt "my" "name" "is" <actor>)seq will output the sentence "hello my name is James Bond", or simply "my name is James Bond".

Those features improve the flexibility of the system. This example describes the power of those features:
alt("hello" "good_morning" "hi")alt opt(<ACTOR>)opt
where <ACTOR> is a class which contains all the names of the actors who are in the scenario and the different possibility to name them (James bond, mister bond, James, goldfinger and etcetera).
This simple line of code allows the user to say hello in different ways:
hello, good morning, Hi, hello bond, hi bond, good morning, bond, hello goldfinger and etcetera.

A screenshot of the ©BabelTech lex editor is available in appendix B.

4.3 Themes

In the Interactive Story Telling application example, the user plays the role of the villain in a short James Bond movie scenario. Thus, the first thing to do is to collect a suitable number of James Bond villain replies (about 300), and classify them by theme.
Obviously each of those replies has been extended to allow the user to say the sentences in different ways in the purpose of making the system more flexible. After a review of several James Bond movies dialogues, in the final version sentences have been classified into twenty main themes as follow:

- **Denials:**
  This theme is regrouping all the sentences that deals with a denial in which in which the villain refuses to tell an information to James Bond or ironically refuses to tell him the answer.
- **Introduction:**
  This theme is used to introduce actors to James Bond.
- **Threat:**
  This theme is a series of threating replies toward James Bond.

- **Challenge:**
  The sentences contained in this theme aims at challenging James Bond.
- **Agreements Answer:**
  This theme includes all the possibility that the user can say to agree with Mr. Bond.
- **Disagreements Answer:**
  This theme includes all the possibility that the user can say to disagree with Mr. Bond.
- **Greetings hi:**
  This theme includes all the possibility that the user can say to welcome Mr. Bond.
- **Greetings bye:**
  All the ways to say bye to Mr. Bond.
- **Complain:**
  This theme contains several sentences which express a complain toward Mr. Bond.
- **Offensive:**
  This theme aims at offending Bond.
- **Disagreement action:**
  When the user wants to stop Mr. Bond from doing an action.
- **Agreement actions:**
  When the user wants Mr. Bond to carry on his action.
- **Drinks questions:**
  Allow the user to propose a drink to Mr. Bond.
- **Command action Threat:**
  Allow the user to command Mr. Bond by threatening him, sentences like "put your hands on you head" and etcetera.
- **Misunderstanding:**
  When the user does not understand what Mr. Bond is talking about, he can ask him to repeat.

- **Thanks:**
  All the way to say thanks to an actor.
- **Compliment:**
  Several Compliments
- **Incredulity:**
  When the user does not trust an actor.
- **Advice:**
  When the user wants to advise Mr. Bond.
- **Romance:**
  This theme contains several romantic sentences if the villain is a girl.

4.4 Grammar classification and rules

4.4.1 Syntactic Based Grammar

The first idea was to define syntactic rules in the corpus to parse most of the sentences in the future. Basic English grammar rules have been reviewed and an implemented version has been produced.
The structure of this corpus is based on splitting grammar entities into phrasal groups. Here are basic grammar entities: subjects, verbs, nouns, pronouns, preposition, quantifier, auxiliary, adjective and etcetera.
Once those entities defined, we group them into phrasal groups as follow:
Nominal Compliment, Nominal Phrase, Prepositional Phrase, Verbal Phrase and etcetera. For example the sentence “I never fail mr bond” is made of a verbal phrase (VP) and a nominal phrase (NP):
seq(
<VP>
<NC>
)seq
A verbal phrase is defined as being:
<VP>=
seq(
opt(<ADVERB>)opt
opt(<SUBJECT>)opt
opt(<ADVERB>)opt

opt(<AUXILIARY>)opt
<VERB>)seq;
Each of those classes is containing a list of relevant words, for example, the class verb is containing a list of verbs:
<VERB> =
alt(
have
hope
fail
admiring
dreaming
be
expect
die
choose
introduce
let
allow
see
buy
)alt;
A nominal complement can be :
<NC>=
seq(
rep(
opt(<ADJECTIVE>)opt
)rep
<NOUN>
rep(
opt(<NOUN>)opt
)rep
opt(<NAME>)opt
)seq
Other examples are available in Appendix C[1]. The Figure 4.1 is describing the structure of an example of syntactic grammar.

[1]All the full versions of the sources are available on the attached CD

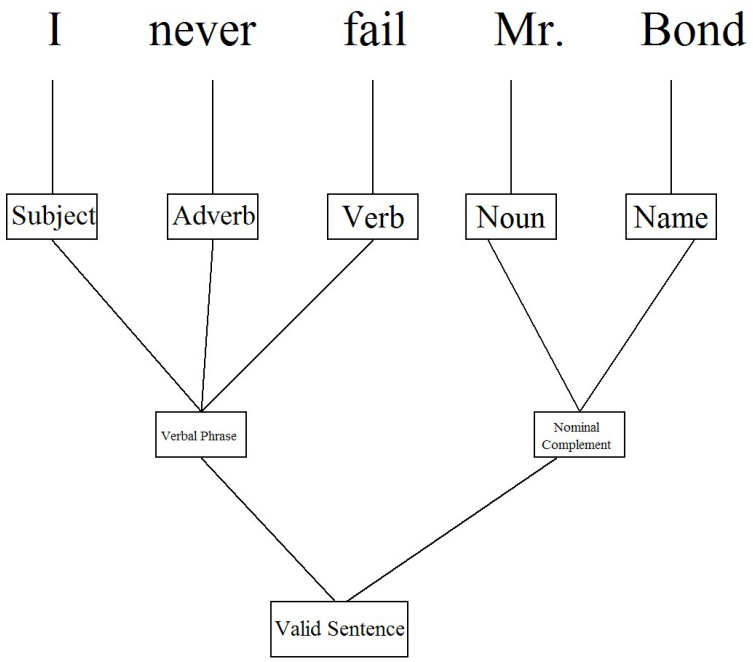

Figure 4.1: Syntatic grammar rules

It has been agreed that the idea of building an English Grammar is too complex for such a specific project. It involves too much rules which force to specify too many optional statements that deeply deteriorate the speech recognition performance.
The other problem raised is semantics because, although syntax is determining the function of the words within a sentence (3.2.3), the semantics are not fully described. However simple rules should not to be ignored, they are useful in some repeat pattern cases.

4.4.2 Thematic Based Grammar

The thematic based grammar is the structure that attempts to build the corpus based on syntactic basic rules but by integrating semantic classification. The idea was to categorize the corpus in themes to do a pre-parsing before validating the speech acts by the templates.
All the grammar entities are split into categories. Those categories are themselves split into sub categories. It reminds an object oriented class pattern on account of there are hierarchical links between categories.

A diagram describing theses classes is available in Appendix D.
There are two main part in the design of this corpus:
The first one defines the common grammar entities for all the sentences and the second part is thematic based containing the specific entities for each theme. For example, the common category "<VERB>" is split into sub categories:
"subject plus verb", "do", "want".
Inside the category "want" there are different patterns that match with the meaning "want". As a result, an affirmation like "This is my best friend" can be also formulated "that s my best friend", "Here is my best friend" and etcetera.
Simply by setting the words "this is", "that s", and "here is" in the same category.
Thematic classes are containing the specific phrases for a given theme. In Appendix D you can see a chart that describes the building of this corpus. In Addition to this diagram an excerpt of the source code is available in Appendix E.
For example the threat theme is dispatched into several classes threatverb, threathnoun, threatadj which respectively contain the specific verbs, nouns and adjectives for the theme threat.
This implies that the corpus is able to differentiate nouns, adjectives and verbs in a theme. We have the two concepts semantics and syntax mixed together.
If we look at the sentence "it may / can / might / will / could be your last" (please refer to line 1030 in the Appendix E), the sentence is made of the classes "subvbe" and "threatnoun" which are associated together in a sequence. The class subvbe (Please refer to Appendix E line 117) is containing the phrases which are composed by a subject, an auxiliary and a verb like "it may be" for example. The second part, threatnoun is containing specific nouns, which are involved in the threat theme (Please refer to Appendix E line 606) like "your_last".
As specified in part 6.1.2, performance was really slow because of the complexity of this corpus and that is why the idea of building such a complex corpus was abandoned.
The idea of classifying the corpus in this way was a good idea to show how to deal with a complex speech structure and how to classify it and the problems that were raised.

4.5 Final version of the Corpus

It has been decided that the classification of the themes will be fully managed at the templates level. The final version of the corpus is based on plain text sentences and alternatives.
The final version of the corpus is about 300 sentences (without alternatives) and contains a dictionary of 400 words.
An excerpt of the final version of the corpus is available in Appendix F.
The final corpus is containing 3 classes: ACTOR, TITLE and AUXILIARY as well as alternatives and optional(Please refer to appendix F line 10).
If we look at this example: “You are just a stupid secret agent”
seq(
<PRONOUN> “are” opt(“just” “nothing_but”)opt “a” alt(“silly” “dumb” “stupid”)alt alt(“secret_agent” “policeman”)alt
)seq
Using optional and alternatives a sentence can be said in many different ways which make the system flexible. In this example we can have 18 different ways to say the sentence :

you are a silly policeman.
you are a stupid policeman.
you are a dumb policeman.
you are a silly secret agent.
you are a stupid secret agent.
you are a dumb secret agent.
you are just a silly secret agent.
you are just a stupid secret agent.
you are just a dumb secret agent.
you are just a silly policeman.
you are just a stupid policeman.
you are just a dumb policeman.
you are nothing but a silly secret agent.
you are nothing but a stupid secret agent.
you are nothing but a dumb secret agent.
you are nothing but a silly policeman.
you are nothing but a stupid policeman.
you are nothing but a dumb policeman.

Another issue was raised while testing this version of the corpus during the template tests. Indeed, the system is clearly better at recognizing group of words than isolated words.
In the previous example, defining the utterances as "you_are_just", "you_are_nothing_but", "a_stupid_policeman","a_silly_secret_agent" and etcetera, will improve the accuracy of the system. That is why in the prototype version of the corpus, group of words are defined rather than isolated words.

Chapter 5

Embedding in Unreal

5.1 The aim of the templates

The system is using the ear SDK (©BabelTech), as a platform to transform speech recognition utterances into speech acts appropriate for the Artificial Intelligence planning layer. To provide suitable speech acts, templates have to be implemented by defining sentence/action-based pattern.
There are two different types of templates used for different purposes:
The speech acts templates define speech acts recognized by the ear SDK. Its structure consists in classes, which contain words or specific phrases linked to a specific act. The matching templates define sentences using speech act template classes. Thus, once generated and recognized these acts can be used to modify the unreal scenario.

5.2 Implementing the templates

5.2.1 The structure

The templates are managed into two natural language understanding (.nlu) files. The first one "templates.nlu" is defining the relation between words or group of words recognized by the ear SDK and the speech acts. In this file the first part is to declare a set of main speech acts representing the themes defined in section 4.3 as follow:
enum E_SentenceClass {
eSA_INTRO
eSA_AGREEANS

eSA_GREETHI
eSA_GREETBY
eSA_THANKS
eSA_DENIAL
eSA_THREAT
eSA_COMPLAINS
eSA_INCRED
eSA_ADVICE
eSA_CHALLENGE
eSA_MISUNDER
eSA_DRINKS
eSA_OFFENSIVE
eSA_DISAGREEACT
eSA_GUNDROPING
eSA_HANDSOHEADS
eSA_MOVOUT
eSA_AGREEACT
eSA_COMPLIMENT
eSA_DISAGREEANS
eSA_ROMANCE
};

Those speech acts consist into an enumeration of classes in which the order is predefined to allow the speech act dispatcher to recognize speech acts only identified by enum numbers.

As the ear SDK is sending speech recognition data using UDP, it increases the performance of the system, indeed, instead of sending long text data, only the identifier number of those classes is sent. Another series of enumeration list is necessary to identify sub speech acts classes.

For instance, if we have the theme Drinks, this main theme will be Drinks and the sub speech acts classes will be eDrinkSake, eDrinkMartini and etcetera to allow the speech dispatcher to associate specific acts to expression pronounced by the user:

enum E_Drink {
eDrinksake
eDrinkVodka
eDrinkStir
eDrinkMartini

};
Using this methodology, the system is able to identify specific acts, which is primordial in such an environment. The last part of the templates file is about the definition of all the words or group of words that correspond to a specific theme and sub speech act classes. In our case drinks we have:
template tDrink =
"sake" eDrinksake [ ] +
"vodka" eDrinkVodka [ ] +
"stirred" eDrinkStir [ ] +
"martini" eDrinkMartini [ ];
The series of enumeration eDrink* is linked to the main speech act theme eSA_Drinks. Here is a chart describing the above example:

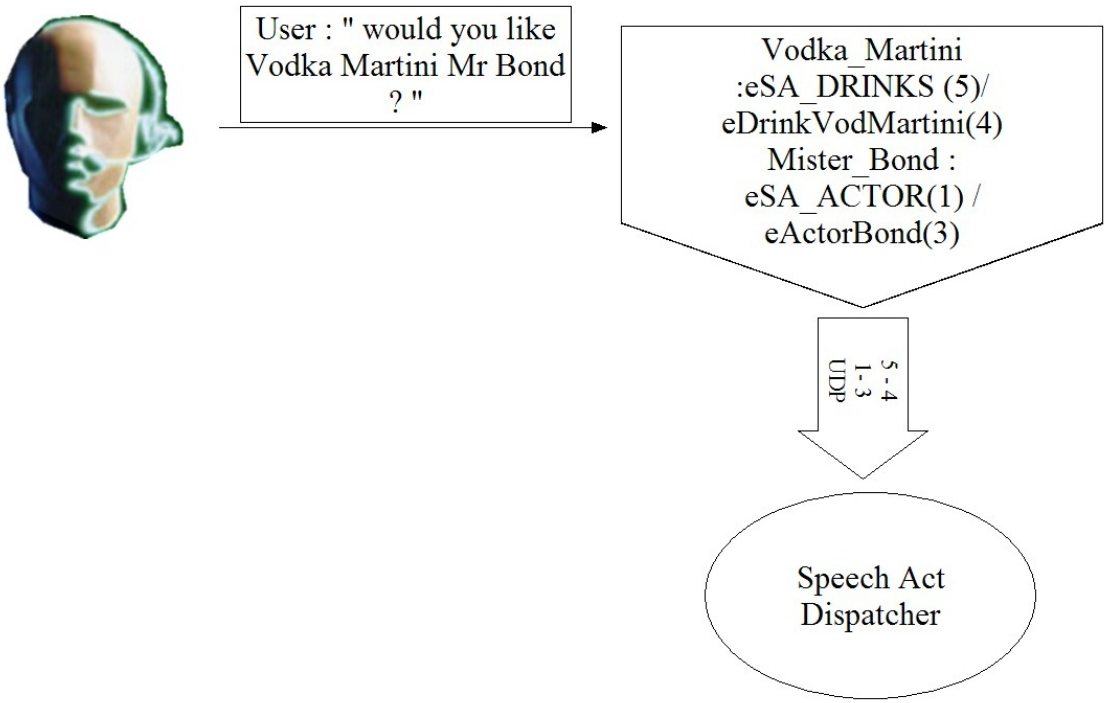

Figure 5.1: Chart describing the processing of the templates

There is a second natural language understanding file (sentences.nlu) that aims at defining and build all the sentences that was defined previously in the corpus. Each sentence is identified by:

- A unique number.

- The speech act the sentence is referring to (e.g. eSA_DRINKS).
- The name of the template used itself (e.g. tDrink).

Using this methodology, the system is not only able to recognize the main theme but can also recognize sub utterances within a given theme.

5.2.2 First version of the templates and problems raised

The first version of the templates was based upon the general idea of associating a word to a speech act as described in the previous section and was aimed at having a better knowledge of the system.
The first version of the templates is dealing with the all the themes described before (Please refer to 5.2.1 and 4.3).
An excerpt of the implementation of this first version is available in Appendix G.
Let us take as an example the sentence "it is insulting to think i haven t anticipated your every move" (Please refer to Appendix G (Source Code Line 429)):
sentence s0156 =
eSA_COMPLAINS
["it" "is" tCompl "to" "think " "i" "havent " "anticipated " " your " "every" "move "]
[^tDenials];
The word which recognized the speech act in this example is "insulting" which is defined as eComplinsult ((Please refer to Appendix G (Source Code Line 299)).
The first problem raised was that if the system does not recognize the word "insulting" the sentence was not validated.
Although some of the sentences were validated by the system, the system was recognizing words defined with double quote as isolated words (Please refer to 6.2.2 Testing Results). This gave low accuracy results.
A lot of errors occurred also because of the case sensitivity of the language, the templates have to reflect exactly the corpus.
The above problems played a role also on the meaning of the sentences; indeed during the first test session a lot of sentences were meaningless.

5.2.3 Design of the templates

The flexibility of the templates and the meaning of the sentences to be output are the key points in this part.
As specified in the article of the magazine EDN:

> "In general, sentences are easier to recognize than words, given that a sentence has more variation from other sentences than words do from words. Longer responses, such as "Buy stocks" or "View my portfolio," are easier to recognize than shorter ones, such as "Buy" or "View".
>
> Nicolas Cravotta. Speech recognition it's not what you say; it's how you say it[6].

To avoid the problems specified in the previous part we have to consider that group of words are better recognized by the system than isolated words.
Analysing all the sentences to see what they have grammatically in common was the first step.
Let us consider the theme denials as an example for this part. If we look at the sentences which are in the theme Denials (Please refer to Appendix H line) we can see that we can split the sentences into two main parts. The beginning of the sentences, which is grammatically important, but with no meaning and the second part of the sentences that will identify the speech act.
If we look at those sentences :

- Why would you like to know
  Why would you like to know
  Why are you interested
  Why do you care
  sentence s0082 =
  eSA_DENIALS
  [tstartQuestw tDenialsProp]
  [^tstartQuestw ^tDenialsProp];

- Why do you care,Mr Bond
  Why would you like to know,Mr Bond
  Why are you interested Mr Bond
  (Appendix H line 488) sentence s0083 =
  eSA_DENIALS
  [tstartQuestw tDenialsProp tActor]
  [^tstartQuestw ^tDenialsProp ^tActor];

  The first part of the sentences is starting by a question word associated with a verb and the second part is a group of words that defines the meaning of the sentence. It is this group of words that as to be linked to the

speech act.

To do so, a template which contains the question tags which starts by “W” was created. This part includes group of words such as “why_are”, “why_do” and etcetera. The second part will consist of large group of words, which are defined in the template Denials Propositions. This template contains the expression ”are_you_interested”, ”you_care”, ”you_like_to_know”.

The interesting bit here is that if the first part of the sentence is unfortunately unrecognised by the system, the second part is meaningful without his first part. As specified in the part structure (Please see 5.2.1), although those expressions are in the same template, an enumeration type identifies them. For example for the expression ”you_like_know”, it will be recognized as Denials proposition and also as a “eDenialKnow”. The Figure 5.2 describes the process the templates in the prototype version of the templates.

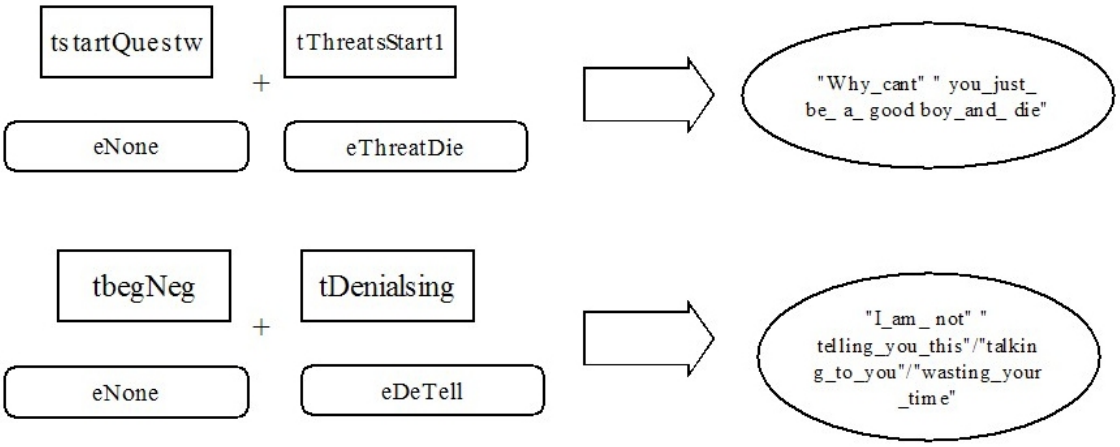

Figure 5.2: Templates Processing Chart

As the figure 5.2 shows it, the first part of the sentence in the first example on the figure is tstartQuestw containing all the appropriate question tags which start with a W (Please refer to appendix H Line 313), and the second part tThreatsStart1 which contains specific threat theme long expressions(Please refer to appendix H Line 225).

A Class definition diagram of all the classes embedded in the prototype version is available in Appendix I. This diagram is describing the whole templates definition of the prototype.

Chapter 6

Testing and refinement

6.1 Testing the Corpus

6.1.1 Efficiency testing techniques

Several tests have been done on the corpus using the ©BabelTech Lex Editor and a microphone. The tests were based on telling all the sentences that were in the corpus using a microphone. By testing all the sentences and writing the result in a table an average of the number of the sentence recognized on one hundred was produced.

6.1.2 Testing results

- **Thematic based grammar Corpus Test:** The first problem encountered during this test was the delay between the pronunciation of a sentence and the recognition. The delay was about one minute; the recognition was too long because of the complexity of the corpus structure. This problem was caused by a too much use of the operating system resources, when the system is loading and recognizing speech the central processing unit was 100 per cent. On 100 sentences only 35 was recognized without errors. Again because of the complexity of the structure the system gets quickly confused. For example if the user says the sentence "Failure is not tolerate" the system recognized "amusement a date of".

- **Final corpus version Test:**
  By decreasing the structure of the corpus, the performance of the system was better than before. Indeed the recognition delay was about 1 to 2 seconds, and the CPU usage never reached 100 per cent, it was about 50 to a maximum of 90 per cent during the recognition process. On 100 sentences an average of 65 sentences was recognized depending on the complexity of the sentences. However the main reason of the speech recognition errors was caused by the isolated words structure in the corpus.
- **Test on the Latest version of the corpus used for the current prototype:**
  During the last test the system was able to recognize an average of 75 sentences on 100. Structuring the corpus by group of words was the solution to the problems encountered in the previous tests. Indeed, we have a better recognition because there were no conflicts anymore between the isolated words.

6.2 Testing the templates

6.2.1 Efficiency testing techniques

The first point was to compiled the two natural language understanding (".nlu") files using an application called Natlang provided by Mr. Steven Mead. The application aims at debugging the code and checking the validity of the sentence by typing them within the keyboard as illustrated by the Figure 6.1.

In this example, the expression "hello" and "hello mister_bond" are valid sentences. The application is displaying relevant information on the recognized speech acts: The classification field is indicating the main speech acts of the sentence in this case [2]. It corresponds to eSA_GREETHI (greetings). The template number, which in our case is 9 and corresponds to "tGreetHi". And finally the instance which contains the specific speech act "eGreethiHi". After this first testing was done another important test was to use a test application provided by Fred Charles. The first step was to export the finite state grammar of the corpus into an ear application, then specifying the path of the templates file. After that the application launches the ear sdk

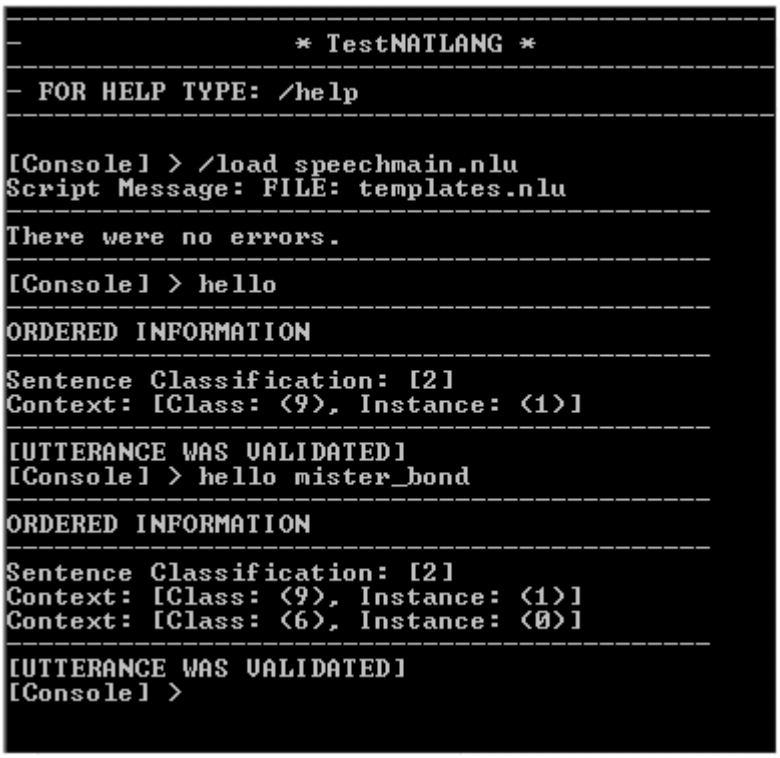

Figure 6.1: NatLang Screen Shot

system and unreal. After by using a microphone the aim is to pronounce all the sentences provided in the templates and see if they are all properly recognized.
A screenshot of the application is available in Appendix I.
The application is able to provide log files to display a description of the sentences well recognized and validated by the system. This application was really helpful to improve the design of the templates.

6.2.2 Testing results

- **Test on the first version of the templates :**
  Small expressions like “hello”, “I agree” were well recognized. But all the complex or even normal sentences were confusing the system. For example “I want to know that”, was implemented as follow:
  “I” “want” “to” “know” tDenials “that”.
  Again the system is confused by isolated words.
- **Test on the latest version of the templates used for the current prototype:**

The latest version of the templates based on the part design of the templates is structured by group of words. All the sentences were well recognized with an average of about 77 sentences recognized on 100.

6.3 Outcome

Currently the system has appreciable recognition accuracy (77%). The accuracy can be improved by sub-categorizing again the class Sentence in the templates (Example: tbegGeneBe and etcetera (Appendix I, SENTENCE)) to allow a better validation of the sentences by the system.
For example, if we take the sentence “Why_cant” “you_just_be_a_good _boy_and_die”.
The sequence “why_cant” (Appendix H.1, line 313) is included in the class tstartQuestw which also contains other sequences like “what_are ”,“ whats ” and etcetera.
That means in some rare cases the sentence “what_are” “you_just_be_a_good _boy_and_die” can be recognized by the system as a valid sentence.
Even if the system will validate the sentence in the right way by recognizing the relevant speech acts defined by the phrase “you_just_be_a_ good _boy_ and_die” the sentence is not grammatically correct.
By subcategorising the class tstartQuestw the syntax of the sentence can be better parsed.
Although the system can still produced in some cases grammatically invalid sentences, it is recognizing the relevant speech acts most of the time, which is more important in such a system.
The flexibility of the system can be improved as well by extending the prototype corpus and the prototype templates using the final version of the corpus produced previously.

Chapter 7

Conclusion

We have presented a vocal interface to a computer animation system and methodologies to build such an interface.
A research on Speech recognition systems and Natural language methodologies has been done as well as a review of the use of Natural language in an interactive storytelling system. Several methodologies have been implemented and tested to see which one can provide the best output.

A working prototype version based on six themes has been produced with an average of 77% sentences well recognized and validated by the system (Please refer to 6.2.2). However, the prototype could be extent to twenty themes. Some part of the templates definition can be improved (Please refer to 6.2.3) for the system to better parse the syntax.
A corpus has been as well produced containing 20 themes, 300 sentence and 400 words in the dictionary.

The final step consists in extending the templates by adding the alternatives included in the final version of the corpus in order to make the system even more flexible.
The constraints that have had to be considered are that to build a specific vocal interface to a computer animation system, a complex syntactic grammar structure is not required.
However, it could have been pertinent to reuse a pre-built English grammar definition to parse the sentences.
But building such a complex grammar alone was not appropriate due to the time schedule of the project, it requires more time.

The accuracy of the system is based upon three key points, the flexibility and the size of the corpus, the way of dealing with the speech acts and the structure of the sentences.

Bibliography

[1] Avron Barr and Edward A. Feigenbaum. *The Handbook of artificial intelligence, v.I.* William Kaufman, Inc., Los Altos, Calif, 1981, 1981.

[2] David J. Buerger. *LaTeX for Engineers and Scientists.* McGraw-Hill, New York, NY, USA, 1990.

[3] Marc Cavazza. Virtual unreality: Storytelling in virtual environments. ACM VRST, 2003.

[4] Marc Cavazza, Fred Charles, and Steven J. Mead. Non-instructional linguistic communication with virtual actors. In *Proceedings IEEE International Workshop on Robot and Human Interactive Communication Vlizy, France*, pages 26–31, 2001.

[5] Cavazza M. Charles F. and Mead S.J. Interactive storytelling: From computer games to interactive stories. *The Electronic Library*, pages 103–112, 2002.

[6] Nicolas Cravotta. Speech recoginition it's not what you say; it's how you say it. *EDNMAG*, pages 79–88, June 24, 1999. `http://www.reed-electronics.com/ednmag/contents/images/45962.pdf`.

[7] Olivier Deroo. A short introduction to speech recognition, 2003. `http://www.babeltech.com/download/SpeechRecoIntro.pdf`.

[8] Antoni Diller. *LaTeX Line by Line: Tips and Techniques for Document Processing.* Wiley Professional Computing. Wiley, Chichester, UK, 1993. Optionally accompanied by disk with examples, ISBN 0-471-93797-5.

[9] Sam Hsiung. An introduction to natural language processing. *Generation 5*, December 19, 1999. `http://www.generation5.org/content/1999/nlpoverview.asp`.

[10] Daniel Jurafsky and James H. Martin. *SPEECH and LANGUAGE PROCESSING: An Introduction to Natural Language Processing, Computational Linguistics, and Speech Recognition*. Prentice-Hall, 2000.

[11] Steven Mead, Marc Cavazza, and Fred Charles. Influential words: Natural language in interactive storytelling. 10th International Conference on Human-Computer Interaction, Crete, Greece, 2003.

[12] Diego Molla. Introduction to natural language processing, overview of language technology. In *Lecture*. Macquarie University, Sydney, 2003. `http://www.comp.mq.edu.au/units/comp248/lectures/comp248-2003-W01-L2.pdf`.

[13] Woojin Paik. Natural language processing (nlp). In *Lecture*, 2002. `http://www.cs.umb.edu/cs670/lecture-10302002.pdf`.

[14] Ellen Riloff. Lecture: Introduction to nlp, 2003. `http://www.cs.utah.edu/classes/cs5340/slides/introduction.pdf`.

[15] George W. Smith. *Computers and Human Language*. Oxford University Press, Oxford, 1991.

[16] V. Zue and R. A. Cole. Spoken language input. In *Survey of the State of the Art in Human Language Technology*, pages 1–57, 1996.

Appendix A

Project specification

Development of a vocal interface to a computer animation system.

The objective of the project is to produce a speech recognition grammar that aims at improving the communication between the user and the machine within an animation system.

It will improve the way the user is communicating with virtual characters in a video game like environment.
The speech recognition grammar will be built using the Babel lexicon editor (©BabelTech).

The speech recognition grammar has to be implemented to manage whatever the game is, or the software that uses the grammar is. Thus, the grammar has to contain all the English basic words (like auxiliary, standard verbs and etcetera).

The first thing to do is to analyse how the English grammar and typical sentences are built, and examine how to build simple sentences within the lexicon editor: this is the analysis phase.

An analysis of the linguistic processing using by the Interactive Storytelling System has to be done as well in order to be aware of what the ISS needs.
The design phase will consist of defining how to create a Finite State Grammar template which will contain all the grammar rules and definitions into classes, so grammar rules has to be defined.

During this phase several charts have to be done to explain how the words are linked together or not.
Once the design is done, it has to be validated and tested to prevent post failure in the next steps of the project.
The Implementation phase will use previous research work to implement the FSG template.
The testing and refinement step will, refine and test if the FSG file is correct, and it aims to detect any errors and correct them.
The animation system that will be used for the end test will be the Interactive Storytelling System used by the university.
If time permits it, a tool will be implemented to allow the user to dynamically change the FSG file without being compelled to write any FSG code.
The final report will be written ongoing the project.
The minimum objective of my project is to produce a flexible speech recognition grammar template to be used by an animation system.

Proposed time schedule:
Analysis - 3 weeks
Design and Interim Report - 6 weeks
Implementation - 6 weeks
Testing and Refinement - 5 weeks
Writing Report - 2 weeks

Appendix B

BabelTech Lexicon Editor screenshot

Figure B.1: Babel Tech Lexicon Editor Screenshot

Appendix C

Excerpt of the syntactic based Corpus

C.1 Grammar Definition

...

```
<ARTICLE> =
alt(
        the
        a
)alt;
<PREPOSITION> =
alt(
        seq(
                alt(
                    next
            of
                        to
                        on
                        in
                        with
                        for
                        against
                )alt
        )seq
)alt;
<NOUN> =
alt(

    opt( <QUANTIFIER> )opt
    opt( <ARTICLE>)opt
    opt( <CARDINAL>)opt
    chance
    ppk
    good
    men
    women
    time
    sense
    form
```

```
44      "name_s"
45      no
46      last
47      witticism
48      policeman
49      mister
50          information
51          sex
52          violence
53      gun
54  )alt;
55
56  <SUBJECT> =
57  alt(
58      it
59      I
60          you
61          we
62          they
63  )alt;
64
65  <PRONOUN> =
66  alt(
67      my
68          me
69          your
70          myself
71          us
72  )alt;
73  <VERB> =
74  alt(
75      have
76      hope
77      fail
78      admiring
79      dreaming
80      be
81      expect
82      die
83      choose
84          introduce
85          let
86          allow
87          see
88          buy
89          located
90          give
91          finding
92          talking
93          go
94          corpses
95          misjudged
96          come
97      going
98      need
99  )alt;
100 <ADVERB> =
101 alt(
102     each
103     always
104     absolutely
105     well
106     carefully
107     unfortunately
108     never
109         only
110 )alt;
111 <ADJECTIVE> =
112 alt(
```

```
113       opt( <ARTICLE>)opt
114       opt( <CARDINAL>)opt
115       opt( <QUANTIFIER>)opt
116       "fifty_fifty"
117       golden
118       gratuitous
119       my
120           nice
121           stupid
122           just
123   )alt;
124   <AUXILIARY> =
125   alt(
126       be
127       been
128       could
129       have
130           can
131           are
132       must
133       am
134   )alt;
135   <CONJUCTION> =
136   alt(
137       and
138   )alt;
139
140   <CARDINAL>=
141    alt(
142       first
143           second
144       one
145       )alt;
146
147   <COUNTRY>=
148    alt(
149       Japan
150       )alt;
151   <NAME> =
152   alt(
153           Bond
154           James
155           James_Bond
156       Ernst_Stravo_Blodfeld
157   )alt;
158
159   <QUANTIFIER> =
160     alt(
161   some
162     )alt;
163   #Nominal Compliment
164   <NC>=
165   seq(
166      rep(
167       opt( <ADJECTIVE> )opt
168      )rep
169      <NOUN>
170      rep(
171       opt( <NOUN> )opt
172      )rep
173      opt( <NAME> )opt
174      )seq;
175   #Nominal Phrase
176   <NP>=
177   seq(
178
179         rep(
180           opt( <ADJECTIVE> )opt
181         )rep
```

```
182       <NOUN>
183       rep( opt( <NOUN> )opt )rep
184
185   )seq;
186
187   #Preposition Phrase
188   <PP>=
189   seq(
190      <PREPOSITION>
191        alt(
192           opt( <AUXILIARY> )opt
193           <VERB>
194           alt(  <COUNTRY>
195                  <NC>
196                  <NP>
197           )alt
198         )alt
199       opt( <ADVERB>)opt
200    )seq;
201   #Verbal Phrase
202   <VP>=
203   seq(
204      opt( <ADVERB> )opt
205      opt( <SUBJECT> )opt
206      opt( <ADVERB> )opt
207      opt( <AUXILIARY> )opt
208      <VERB>
209      )seq;
210
```

...

C.2 Some sentences examples

...

```
260   # Choose your next witticism carefully mr bond, it could be your last.
261       seq(
262       <VP>
263       <PRONOUN>
264       <PP>
265       )seq
266
267       seq(
268        <VP>
269        <PRONOUN>
270        <NOUN>
271       )seq
272   #no mr bond I expect you to die
273            seq(
274             <NC>
275        <VP>
276        <SUBJECT>
277        <PP>
278           )seq
279
280   # allow me to introduce myself, I am ernt stqvro blofeld
281               seq(
282       <VP>
```

```
283       <PRONOUN>
284       <PP>
285       <PRONOUN>
286       )seq
287
288       seq(
289        <VP>
290          )seq
291   #good to see you mr bond, I hope we are going to have some gratuitous sex
      and violence
292
293           seq(
294            <NOUN>
295        <PP>
296        <SUBJECT>
297        <NC>
298         )seq
299
300        seq(
301        <VP>
302        <VP>
303        <PP>
304        <CONJUCTION>
305        <NOUN>
306         )seq
307
```

...

Appendix D

FSG definition Chart (Complex Corpus)

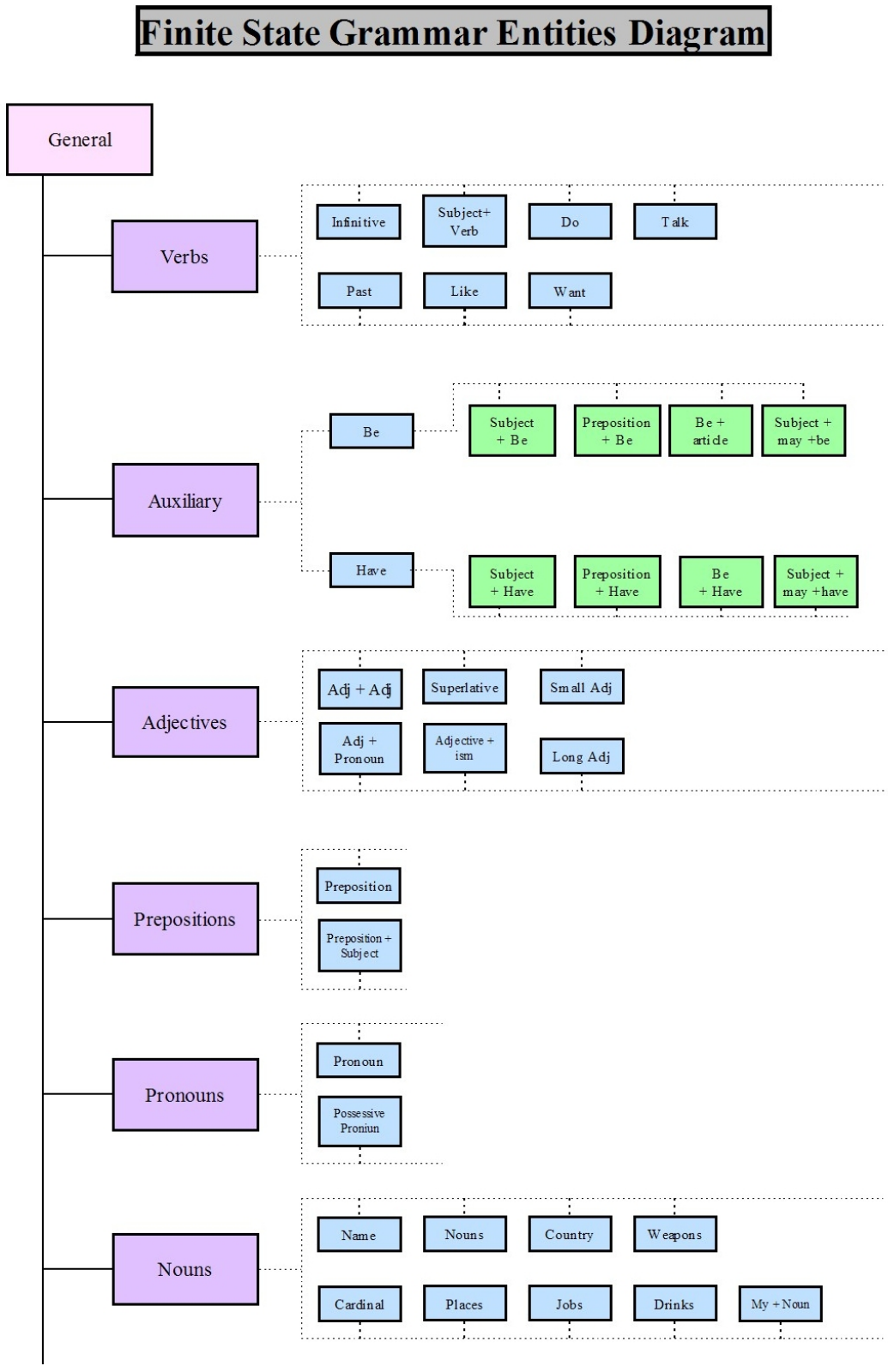

Figure D.1: Finite State Grammar Definition Chart

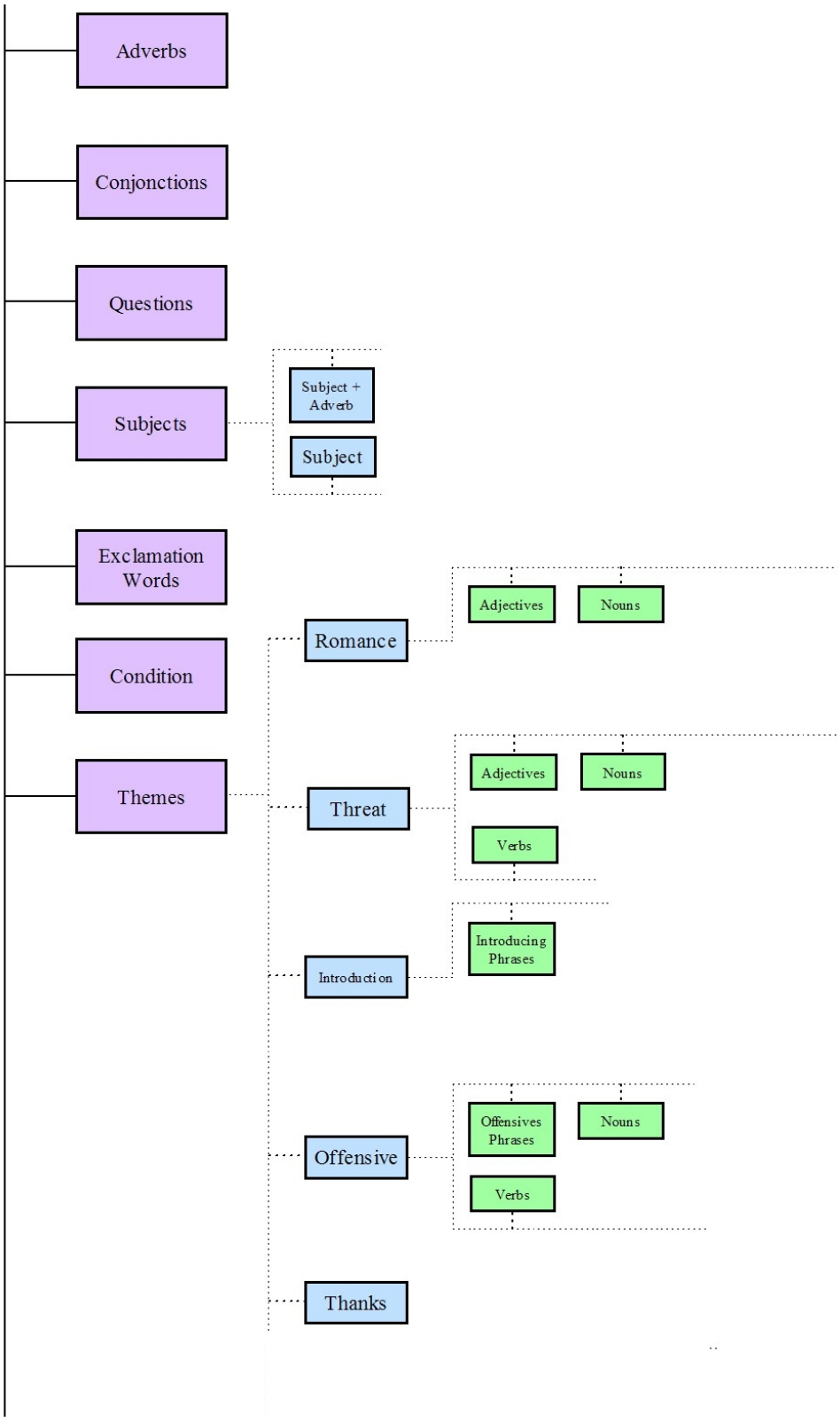

Figure D.2: Finite State Grammar Definition Chart

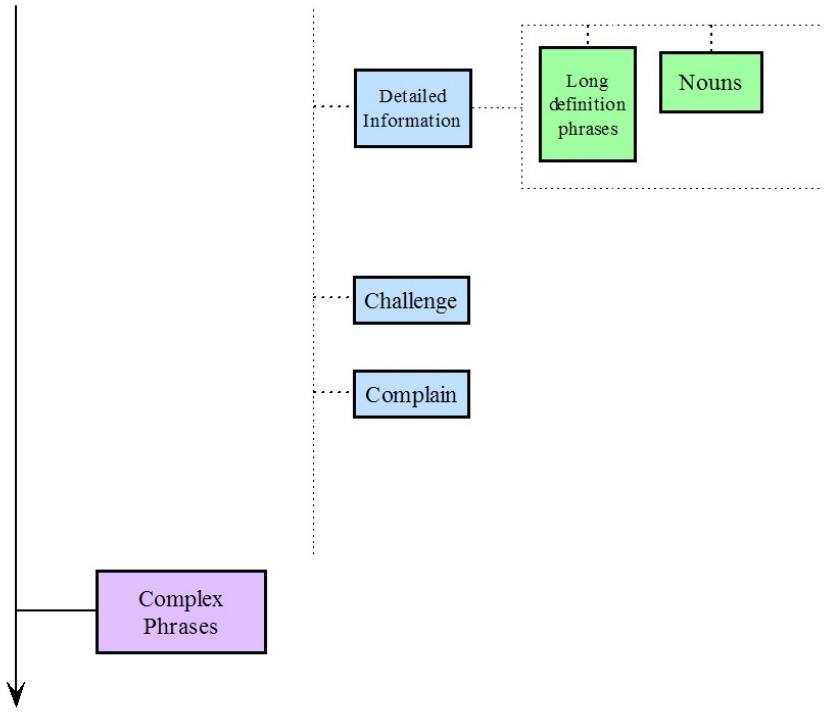

Figure D.3: Finite State Grammar Definition Chart

Appendix E

Thematic based corpus excerpt

E.1 Some grammar rules definition examples

...

```
<name>=
alt(
Colonel_ourumov
to_think
        Bond
        James
        James_Bond
    Ernst_Stravo_Blodfeld
    Domino
    Tanaka
    Tiger
    Number_three

)alt;
#Preposition
<prep>=
alt(
that
next
of
to
on
in
in_the
with
for
against
from
)alt;
#prep + sub
<prepcomb>=
alt(
for_this
in_the
to_the
with_your
for_me
to_you
```

```
to_me
without_me
with_me
with_you
each_of
        )alt;
<adjectiveism>=
alt(
    witticism
)alt;
#Adjectives combination
<adjcomb>=
alt(
unpleasant_surprise
fatal_weakness
simple
very_simple
)alt;

<condition>=
alt(
if_he
if_you
   )alt;
# to be :: Subject + be
<subbe>=
    alt(
you_will
ythey_are
this_is
I_was
it_is
I_am
I_am_not
you_are
you_are_that
you_were
    )alt;
# preposition +  be
<prepbe>=
alt(
just_be
     that_are
     who_are
   )alt;

# to be ::  be + word
<iscomb>=
alt(
is_quite
is_not
is_always
is_the
is
```

...

```
# Subject proba auxiliary
<subvbe>=
alt(
it_will_be
it_could_be
it_might_be
it_can_be
it_may_be
   )alt;
```

...

```
<subverb>=
alt(
I_said
I_expect
he_promises
I_beg
they_belong_to
you_have_lost
it_may_help
it_might_help
it_can_help
you_cant
you_mustnt
we_couldnt
we_wont
we_will_not
we_cant
we_cannot
          I_could
      I_can
      I_think
      you_think
      you_know
      it_will_help
      you_get
      men_always_come
      women_come
   )alt;


<adverb>=
alt(
slowly
always
again
totally
    each
    always
    absolutely
    well
    carefully
    unfortunately
    never
        completely
        only
    too
)alt;
<conj>=
alt(
or
and
that
for
but
just_for
)alt;
#Conj combination
```

...

E.2 Specific theme grammar definition

...

```
591 #Threat
592 <threatvb>=
593 alt(
594 toss
595 kill_me
596 Throw_down
597 Fire
598     fail
599    )alt;
600 <threatadj>=
601 alt(
602 fifty_fifty
603 a_fifty_fifty
604 stupid
605    )alt;
606 <threatnoun>=
607 alt(
608 you_fool
609 the_limbs
610 failure
611 your_last
612    )alt;
```

...

E.3 Some sentences examples from the Threat Theme

...

```
973 #----------threats----------------
974 <THREAT>=
975 Alt(
976 #you will die for this
977 seq(
978 <subbe>
979 <iverb>
980 <prepcomb>
981 )seq
982 #threat
983 #you are mine now
984 seq(
985 <subbe>
986 <pronposs>
987 <times>
988 )seq
989 #your fatal weakness
990 seq(
991 <pronoun>
992 <adjcomb>
```

```
)seq
#Why cant you just be a good boy and die?
seq(
<whqueststst>
<prepbe>
<goodness>
<conj>
<iverb>
)seq
#you were supposed to die for me
seq(
<subbe>
<pret>
<toverb>
<prepcomb>
)seq
#but sorry
seq(
<conj>
<nouns>
opt(
<name>
)opt
)seq

        #Choose/pick your next witticism carefully Mr Bond,
        seq(
        <iverb>
        <pronoun>
        <prep>
        <adjectiveism>
        <adverb>
        opt(
<name>
)opt
        )seq

    #it may/can/might/will/could be your last

    seq(
        <subvbe>
        <threatnoun>
        )seq


...
```

Appendix F

Corpus final version excerpt

F.1 Classes Definitions

...

```
<TITLE> =
alt(
        "mister"
        "miss"
)alt;
<PRONOUN> =
alt(
        "I"
        "you"
        "he"
        "she"
        "we"
        "you"
        "they"
)alt;
<AUXILIARY> =
alt(
        "could"
        "would"
        "am"
        "is"
        "are"
        "do"
        "will"
        "may"
        "might"
)alt;
<NEGATION> =
seq(
        opt( <AUXILIARY> )opt
        alt(
                "not"
        )alt
)seq;
<ACTOR> =
seq(
        opt( <TITLE> )opt
        alt(
```

```
48                  "james␣bond"
49                  "bond"
50                  "double␣o␣seven"
51                  "goldfinger"
52          )alt
53  )seq;
```

...

F.2 Denial Theme

...

```
57  #----------------DENIAL
    ------------------------------------------------------------------------------

58      #How would I know, Mr Bond
59
60          seq(
61                  "how␣would" <PRONOUN> "know" opt( <ACTOR> )opt
62          )seq
63
64          #How would I be aware of that, Mr Bond
65
66          seq(
67                  "how␣would" <PRONOUN> "be" "aware" "of" "that" opt( <ACTOR>
    )opt
68          )seq
69
70          #I don't␣have␣a␣clue
71          seq(
72                  "I" "don_t" "have" "a" alt( "clue" "hint" )alt
73
74          )seq
75
76          #I don't␣have␣a␣clue
77          seq(
78                  "I" "don_t" "have" "a" "clue" "of" "what" "you" "are" "
    talking" "about"
79          )seq
80          #I have no idea
81          seq(
82                  <PRONOUN> "have␣no␣idea"
83          )seq
84
85          #I would not know
86          seq(
87                  <PRONOUN> <NEGATION> "know"
88          )seq
89
90          #I could not tell you
91          seq(
92                  <PRONOUN> <NEGATION> "tell" <PRONOUN>
93          )seq
94
95
96          #Do you seriously think I would tell you
97          seq(
98                  "do" "you" "seriously" "think" <PRONOUN> "would" "tell" <
    PRONOUN>
```

```
99          )seq
100         #I am not telling you this
101         seq(
102                 <PRONOUN> <NEGATION> "telling" <PRONOUN> alt( "this" "that"
    )alt
103         )seq
104
105         #Why would I tell you
106         seq(
107                 "why␣would" <PRONOUN> "tell" <PRONOUN>
108         )seq
109         #I am not talking to you, Mr Bond
110         seq(
111                 <PRONOUN> <NEGATION> "talking" "to" <PRONOUN> opt( <ACTOR> )
    opt
112         )seq
113         #Why would I give you such informatio
114         seq(
115                 "why" "would" <PRONOUN> "give" <PRONOUN> "such" "information
    "
116         )seq
117
118
119         #This is no business of yours
120         seq(
121                 "this" "is" "no" "business" "of" "yours"
122         )seq
123         #This is none of your business
124         seq(
125                 "this" "is" "none" "of" "your" "business"
126         )seq
127
128         #It s not your business
129         seq(
130                 "it" <NEGATION>  "your" alt( "business" "deal")alt
131         )seq
132
133
134
135         #You're␣wasting␣your␣time
136         seq(
137                 <PRONOUN> "are" "wasting" "your" "time"
138         )seq
139         #You're␣wasting␣my␣time
140         seq(
141                 <PRONOUN> "are" "wasting" "my" "time"
142         )seq
143         #You don't␣want␣to␣know
144         seq(
145                 <PRONOUN> <NEGATION> "want" "to" "know"
146         )seq
147
148         #You are not serious
149         seq(
150                 <PRONOUN> <NEGATION> alt("serious" "sincere" "honest")alt
151         )seq
152         #You are joking
153         seq(
154                 <PRONOUN> "are" "joking"
155         )seq
156         #is it a joke ?
157         seq(
158                 "is" "it" "a" "joke"
159         )seq
160
161         #You must be joking
162         seq(
163                 <PRONOUN> "must" "be" "joking"
164         )seq
```

```
165         #Are you serious, Mr Bond
166         seq(
167                 "are" <PRONOUN> "serious" opt( <ACTOR> )opt
168         )seq
169
170         #Why would you like to know
171
172         seq(
173                 "why␣would" <PRONOUN> "like" "to" "know"
174         )seq
175
176         #Why do you care, Mr Bond
177         seq(
178                 "why␣do" <PRONOUN> "care" opt( <ACTOR> )opt
179         )seq
180         #Why are you interested
181         seq(
182                 "why␣are" <PRONOUN> "interested"
183         )seq
184         #Why do you want to know
185         seq(
186                 "why␣do" <PRONOUN> "want" "to" "know"
187         )seq
188
189         # Why will I share this piece of information with you ?
190         seq(
191                 "why" "will" "i" "share" alt( "this␣piece␣of␣information" "
    that" )alt "with" "you"
192         )seq
193
194         #I won t tell you 007
195         seq(
196                 <PRONOUN>  "won_t" "tell" <PRONOUN>
197         )seq
198         # you are too curious bond
199         seq(
200                 "you" "are" "too" "curious" opt( <ACTOR> )opt
201         )seq
202
203         # are u sure you want to get involved in that ?
204
205         seq(
206                  "are" "you" "sure" "you" "want" "to" "get" "involved" "in"
    "that" opt( <ACTOR> )opt
207         )seq
208
209         #You are not able to know that
210         seq(
211          <PRONOUN> <NEGATION> "able" "to" "know" "that"
212         )seq
213         #You don t need to know that
214         seq(
215          <PRONOUN> "don_t" "need" "to" "know" "that"
216         )seq
217
218         #I can t tell you
219         seq(
220         <PRONOUN> "can_t" "tell" <PRONOUN>
221         )seq
222
223         #It is confidential
224         seq(
225         "it" "is" alt( "confidential" "private" "secret" )alt
226         )seq
227
228
229         #Let s go down to business
230         seq(
231                 "lets" "get" "down" "to" "business"
```

```
232         )seq
233
234         # It is not in your interest to know that
235         seq(
236                 "it" "is" "not" "in" "your" alt( "interest" "concern" "
    preoccupation" )alt  "to" "know" "that"
237         )seq
238
239         # you should not worry about that
240         seq(
241                 "you" "should" "not" "worry" "about" "that"
242         )seq
243
244         #don't␣ask
245         seq(
246         "don_t" "ask"
247         )seq
248         #you 'd␣better␣forget␣about␣that
249         seq(
250         "you_d" "better" "forget" "about" "that"
251         )seq
252         #Which game are you playing ?
253         seq(
254         "which" "game" "are" "you" "playing"
255         )seq
```

...

Appendix G

Templates first version source code Excerpt

G.1 Templates First Version Definition Example

...

```
3   enum E_SentenceClass {
4           eSA_INTRO
5           eSA_AGREEANS
6           eSA_GREETHI
7           eSA_GREETBY
8           eSA_THANKS
9           eSA_DENIAL
10          eSA_THREAT
11          eSA_COMPLAINS
12          eSA_INCRED
13          eSA_ADVICE
14          eSA_CHALLENGE
15          eSA_MISUNDER
16          eSA_DRINKS
17          eSA_OFFENSIVE
18          eSA_DISAGREEACT
19          eSA_GUNDROPING
20          eSA_HANDSOHEADS
21          eSA_MOVOUT
22          eSA_AGREEACT
23          eSA_COMPLIMENT
24          eSA_DISAGREEANS
25          eSA_ROMANCE
26  };
```

...

```
101
102 enum E_Compl {
```

```
103          eCompludid
104          eCompldoneth
105          eComplome
106          eCompldare
107          eComplinsult
108          eComplno
109  };
110
111  enum E_Incre {
112          eIncretrust
113          eIncresure
114          eIncrerely
115  };
116
117  enum E_Adv {
118          eAdvatt
119          eAdvcare
120  };
121
122  enum E_Chall {
123          eChallwarn
124          eChalldareu
125  };
126
127  enum E_Misund {
128          eMisundsor
129          eMisundsay
130          eMisundrep
131  };


     ...



292
293  /* Complains (8) */
294  template tCompl          =                    "you_did"
         eCompludid      [ ] +
295                                                   "did_you"
         eCompludid      [ ] +
296                                                   "done_that"
         eCompldoneth    [ ] +
297                                                   "to_me"
         eComplome       [ ] +
298                                                   "dare"
         eCompldare      [ ] +
299                                                   "insulting"
         eComplinsult    [ ] +
300                                                   "nice_one"
         eComplno        [ ] ;
301
302  /* Incredulity (9) */
303  template tIncre          =                    "believe"
         eIncretrust     [ ] +
304                                                   "trust"
     eIncretrust     [ ] +
305                                                   "sure"
     eIncresure      [ ] +
306                                                   "rely"
     eIncrerely      [ ] ;
307
308  /* Advice (10) */
309  template tAdv            =                    "attention"
         eAdvatt [ ] +
310                                                   "carefull"
         eAdvcare        [ ] ;
311
312  /* Challenge (11) */
313  template tChall          =                    "warned"
         eChallwarn      [ ] +
```

```
314                                                     "dare_you"
        eChalldareu     [ ] ;
315 /* Misunderstanding (12) */
316 template tMisund               =                   "sorry"
        eMisundsor      [ ] +
317                                                     "say_it"
        eMisundsay      [ ] +
318                                                     "repeat"
        eMisundrep      [ ] ;
319
320
```

...

G.2 Sentences Examples : Complains, Incredulity, Advice, Challenge, Misunderstanding Themes

...

```
372  /*
373  * Complains
374  */
375
376 //Do you want to explain why you did that
377
378 sentence s0056 =
379         eSA_COMPLAINS
380         [  "do" "you" "want" "to" "explain" "why" tCompl "that" ]
381         [ ^tCompl ];
382
383
384 //Would you mind explaining to me why did you that
385 sentence s0057 =
386         eSA_COMPLAINS
387         [  "would" "you" "mind" "explaining" "to" "me" "why" tCompl "that" ]
388         [ ^tCompl ];
389
390
391
392 //Would you mind explaining to me Have you done that
393 sentence s0058 =
394         eSA_COMPLAINS
395         [  "would" "you" "mind" "explaining" "to" "me" "why" "Have" "you"
    tCompl ]
396         [ ^tCompl ];
397
398 //how could you do that to me
399 sentence s0059 =
400         eSA_COMPLAINS
401         [  "how" "could" "you" "do" "that" tCompl ]
402         [ ^tCompl ];
403
404 //I can t believe you did that
405 sentence s0060 =
406         eSA_COMPLAINS
407         [  "i" "cant" "believe" tCompl "that" ]
408         [ ^tCompl ];
409
```

```
410 //How come you did that
411 sentence s0061 =
412         eSA_COMPLAINS
413         [  "how" "come" tCompl "that" ]
414         [ ^tCompl ];
415
416 //How come you ve done that
417 sentence s0062 =
418         eSA_COMPLAINS
419         [  "how" "come" "you" "ve" tCompl ]
420         [ ^tCompl ];
421
422
423 //How dare you
424 sentence s0063 =
425         eSA_COMPLAINS
426         [  "how" tCompl "you" ]
427         [ ^tCompl ];
428
429 //it is insulting to think i haven t anticipated ur every move
430 sentence s0064 =
431         eSA_COMPLAINS
432         [  "it" "is" tCompl "to" "think" "i" "havent" "anticipated" "your" "
    every" "move" ]
433         [ ^tCompl ];
434
435
436 //Nice one james
437 sentence s0065 =
438         eSA_COMPLAINS
439         [  tCompl tActor ]
440         [ ^tCompl ^tActor ];
441
442
443 /*
444   Incredulity
445
446 //I don t believe trust you
447 sentence s0066 =
448         eSA_INCRED
449         [  "i" "dont" tIncre you ]
450         [ ^tIncre ];
451
452 //Are you sure ?
453 sentence s0067 =
454         eSA_INCRED
455         [  "are" "you" tIncre ]
456         [ ^tIncre ];
457
458 //I can t rely on that mister bond
459 sentence s0068 =
460         eSA_INCRED
461         [  "i" "cant" tIncre "on" "that" tActor ]
462         [ ^tIncre ^tActor ];
463
464 */
465 /*
466  * Advice
467  */
468 sentence s0069 =
469         eSA_ADVICE
470         [  "Please" "pay" tAdv tActor ]
471         [ ^tAdv ^tActor ];
472
473 sentence s0070 =
474         eSA_ADVICE
475         [  "be" tAdv tActor ]
476         [ ^tAdv ^tActor ];
477
```

```
478
479  /*
480   * Challenge
481   */
482
483  sentence s0071 =
484          eSA_CHALLENGE
485          [  "i" "tChall" "you" ]
486          [ ^tChall ];
487
488
489  sentence s0072 =
490          eSA_CHALLENGE
491          [  "i" "double" tChall ]
492          [ ^tChall ];
493
494  /*
495   * Misunderstanding
496   */
497  //sorry
498  sentence s0073 =
499          eSA_MISUNDER
500          [  tMisund ]
501          [ ^tMisund ];
502  //Say it again please
503  sentence s0074 =
504          eSA_MISUNDER
505          [  tMisund "again" "please" ]
506          [ ^tMisund ];
507
508  //Repeat it please
509  sentence s0075 =
510          eSA_MISUNDER
511          [  tMisund "it" "please" ]
512          [ ^tMisund ];
513  // Repeat please
514  sentence s0076 =
515          eSA_MISUNDER
516          [  tMisund "please" ]
517          [ ^tMisund ];
518


     ...
```

Appendix H

Templates source code excerpt

H.1 Templates Definition Example : Denials and Threats

...

```
5  /*
6   * Enumerations
7   */
8  enum E_SentenceClass {
9          eSA_INTRO
10         eSA_AGREEANS
11         eSA_GREETHI
12         eSA_GREETBY
13         eSA_THANKS
14         eSA_DENIALS
15         eSA_THREAT
16         eSA_COMPLAINS
17         eSA_INCRED
18         eSA_ADVICE
19         eSA_CHALLENGE
20         eSA_MISUNDER
21         eSA_DRINKS
22         eSA_OFFENSIVE
23         eSA_DISAGREEACT
24         eSA_GUNDROPING
25         eSA_HANDSOHEADS
26         eSA_MOVOUT
27         eSA_AGREEACT
28         eSA_COMPLIMENT
29         eSA_DISAGREEANS
30         eSA_ROMANCE
31 };
32
```

...

```
68
69  enum E_Denials {
70          eDeKnow
71          eDeTell
72          eDeThink
73          eDeBus
74          eDeJok
75          eDeConf
76          eDeWaste
77          eDeNever
78          eDeSer
79  };

    ...


81  enum E_Threats {
82          eThreatDie
83          eThreatmnow
84          eThreatKill
85          eThreatWitt
86          eThreatPick
87          eThreatFight
88          eThreatNow
89          eThreatWeap
90          eThreatAtt
91          eThreatBus
92          eThreatHell
93          eThreatWait
94          eThreatChoice
95          eThreatWin
96          eThreatLast
97          eThreatMatter
98          };
99

    ...


184
185 template tDenialsProp   =         "i_know"
                  eDeKnow [ ] +
186                                  "i_know_that"
          eDeKnow [ ] +
187                                  "i_give_you_such_information"
                   eDeInf  [ ] +
188                                  "you_like_to_know"
                                eDeKnow [ ] +
189                                  "you_care"
                                   eDeKnow [ ] +
190                                  "you_interested"
                                 eDeKnow  [ ] ;
191
192
193 template tDenialsEnd    =         "tell"
          eDeTell [ ] +
194                                  "tell_you"
          eDeTell [ ] +
195                                  "know"
    eDeTell [ ] +
196                                  "seriously_think"
          eDeThink        [ ] ;
197
198 template tDenialsing    =        "telling_you_this"
                  eDeTell [ ] +
199                                  "talking_to_you"
                  eDeTell [ ] +
```

```
200                                     "wasting_your_time"
                eDeWaste        [ ] +
201                                     "wasting_my_time"
                eDeWaste        [ ] +
202                                     "you_talking_about"
        eDeTell [ ] ;
203
204 template tDenialsExpr   =       "no_business_of_yours"
                eDeBus  [ ] +
205                                 "none_of_your_business"
                eDeBus  [ ] +
206                                 "your_business"
        eDeBus  [ ] +
207                                 "serious"
                eDeSer  [ ] +
208                                 "joking"
                eDeJok  [ ] +
209                                 "able_to_know_that"
                eDeKnow [ ] +
210                                 "need_to_know_that"
                                        eDeKnow [ ] +
211                                 "need_to_know"
                eDeKnow  [ ] +
212                                 "want_to_know_that"
                                        eDeKnow [ ] +
213                                 "want_to_know"
                eDeKnow [ ] +
214                                 "confidential"
                                                eDeConf [ ] +
215                                 "never_heard_of_it"
                                                eDeNever   [ ] ;
216
```

...

```
217 /*
218  * Threats (7)
219  */
220
221 template tThreatsMid1           =         "gonna_die"
        eThreatDie      [ ] +
222                                           "mine_now"
        eThreatmnow     [ ] +
223                                           "you_now"
        eThreatmnow     [ ] ;
224
225 template tThreatsStart1         =         "i_expect_you_to_die"
        eThreatDie      [ ] +
226                                           "you_just_be_a_good_boy_and_die"
    eThreatDie       [ ] +
227                                           "supposed_to_die_for_me"
        eThreatDie      [ ] +
228                                           "try_to_kill_me"
                    eThreatKill      [ ] +
229                                           "
choose_you_next_witticism_carefully"    eThreatWitt   [ ] +
230                                             "go_and_pick_it_up"
                 eThreatPick      [ ] ;
231
232
233 template tThreatsFight          =         "to_fight"
        eThreatFight    [ ] +
234                                           "lets_fight"
        eThreatFight    [ ] ;
235
236 template tThreatsNow            =         "right_now"
        eThreatNow      [ ] ;
237
```

```
238 template tThreatsWeap           =         "golden_gun"
        eThreatWeap     [ ] +
239                                           "walther_ppk"
        eThreatWeap     [ ] ;
240
241 template tThreatsExpr           =         "you_only_live_twice"
        eThreatLive     [ ] +
242                                           "attack_me_with_everything"
        eThreatAtt      [ ] +
243                                           "unfinished_business"
        eThreatBus      [ ] +
244                                           "you_want_to_kill_me"
        eThreatKill     [ ] +
245                                           "kill_you"
        eThreatKill     [ ] +
246                                           "see_you_in_hell"
        eThreatHell     [ ] +
247                                           "you_waiting_for"
        eThreatWait     [ ] +
248                                           "win"
        eThreatWin      [ ] +
249                                           "no_choice"
        eThreatChoice   [ ] +
250                                           "it_may_be_your_last"
        eThreatLast     [ ] ;
251
252 template tThreatsquest          =         "the_matter"
    eThreatMatter   [ ] +
253                                           "your_choice"
    eThreatChoice   [ ] ;
254
255
256
257
```

...

```
280 /*
281  * Elements not recognized as speech acts
282  */
283
284  template tbegGeneBe    =       "you_are"                    eNone   [ ]
    +
285                                  "it_is"                      eNone   [ ]
    +
286                                  "i_am"                       eNone   [ ]
    +
287                                  "youre"                      eNone   [ ]
    +
288                                  "this_is"                    eNone   [ ]
    +
289                                  "you_were"                   eNone   [ ]
    ;
290
291 template tbegGeneOwn    =       "i_have"              eNone   [ ] +
292                                  "you_have"            eNone   [ ];
293
294 template tbegGeneiw      =       "i_would"             eNone   [ ] +
295                                  "i_will"              eNone   [ ];
296
297
298
299 template tbegNeg         =  "i_am_not"                 eNone   [ ] +
300                             "i_could_not"              eNone   [ ] +
301                             "i_would_not"              eNone   [ ] +
302                             "it_is_not"                eNone   [ ] +
303                             "this_is_not"              eNone   [ ] +
304                             "you_are_not"        eNone   [ ] +
```

```
305                             "i_wont"            eNone   [ ] +
306                             "i_will_not"        eNone   [ ] +
307                             "i_dont"            eNone   [ ] +
308                             "you_dont"          eNone   [ ] +
309                             "i_cant"            eNone   [ ] ;
310
311 tbegGeneTh=        "thing_is"                                  eNone   [ ]
    ;
312
313 template tstartQuestw   =     "why_would"                       eNone   [ ]
    +
314                             "why_do"                            eNone   [ ]
    +
315                             "why_are"                           eNone   [ ]
    +
316                             "why_cant"                          eNone   [ ]
    +
317                             "what_is"                           eNone   [ ]
    +
318                             "what_are"                          eNone   [ ]
    +
319                             "whats"                             eNone   [ ]
    +
320                             "how_would"                         eNone   [ ]
    +
321                              "whats"                            eNone   [ ]
    ;
322
323 template tstartQuesta   =   "are_you"
    eNone   [ ] +
324                             "do_you"                            eNone   [ ]
    ;
325
```

H.2 Sentences Examples : Threats and Denials

...

```
160 /*
161  * Threats
162  */
163
164 //you re gonna die ok
165 sentence s0025 =
166         eSA_THREAT
167         [  tbegGeneBe tThreatsMid1  ]
168         [ ^tThreatsMid1  ];
169 //you re gonna die James ok
170 sentence s0026 =
171         eSA_THREAT
172         [  tbegGeneBe  tThreatsMid1 tActor  ]
173         [ ^tThreatsMid1 ^tActor  ];
174
175 //you are mine now  ok
176 sentence s0027 =
177         eSA_THREAT
178         [  tbegGeneBe tThreatsMid1  ]
179         [ ^tbegGeneBe ^tThreatsMid1  ];
180
181
```

```
182 //Ihave you now      ok
183 sentence s0028 =
184         eSA_THREAT
185         [  tbegGeneOwn tThreatsMid1  ]
186         [ ^tbegGeneOwn ^tThreatsMid1  ];
187
188
189 //Mr Bond I expect you to die ok
190 sentence s0029 =
191         eSA_THREAT
192         [  tActor tThreatsStart1  ]
193         [ ^tActor ^tThreatsStart1  ];
194
195
196 //Why cant you just be a good boy and die?       ok
197 sentence s0030 =
198         eSA_THREAT
199         [   tstartQuestw tThreatsStart1  ]
200         [ ^tstartQuestw ^tThreatsStart1  ];
201
202
203 //you were supposed to die for me  ok
204 sentence s0031 =
205         eSA_THREAT
206         [  tbegGeneBe tThreatsStart1  ]
207         [ ^tbegGeneBe ^tThreatsStart1 ];
208
209
210 //Thing is james right now you have to fight **
211 //sentence s0032  =
212 //      eSA_THREAT
213 //      [   tbegGeneTh  tActor  tThreatsNow  tbegGeneOwn  tThreatsFight   ]
214 //      [  ^tbegGeneTh  ^tActor  ^tThreatsNow  ^tbegGeneOwn  ^tThreatsFight
     ];
215
216
217 //let s fight ok
218 sentence s0033 =
219         eSA_THREAT
220         [  tThreatsFight  ]
221         [ ^tThreatsFight  ];
222
223 //Choose you next witticism carefully Mr Bond, it may be your last ***
224 sentence s0034 =
225         eSA_THREAT
226         [   tThreatsStart1  tActor tThreatsExpr   ]
227         [ ^tThreatsStart1 ^tActor ^tThreatsExpr  ];
228
229
230 //You only live twice Mr Bond ok
231 sentence s0035 =
232         eSA_THREAT
233         [ tThreatsExpr tActor   ]
234         [ ^tThreatsExpr ^tActor   ];
235
236
237
238 //My golden gun against your Walther PPK.ok
239 sentence s0036 =
240         eSA_THREAT
241         [  "my" tThreatsWeap "against" "your" tThreatsWeap  ]
242         [ ^tThreatsWeap  ];
243
244
245
246 //Attack me With everything you have       ok
247 sentence s0037 =
248         eSA_THREAT
249         [   tThreatsExpr  tbegGeneOwn  ]
```

```
250         [ ^tThreatsExpr ^tbegGeneOwn ];
251
252 //You and I have unfinished business ok
253 sentence s0038 =
254         eSA_THREAT
255         [  "you_and" tbegGeneOwn tThreatsExpr   ]
256         [ ^tThreatsExpr ^tbegGeneOwn ];
257
258 //u want to kill me bond  ok
259 sentence s0039 =
260         eSA_THREAT
261         [   tThreatsExpr tActor  ]
262         [ ^tThreatsExpr ^tActor  ];
263
264 //I m gonna Kill you ok
265 sentence s0040 =
266         eSA_THREAT
267         [  tbegGeneBe "gonna" tThreatsExpr  ]
268          [ ^tbegGeneBe ^tThreatsExpr  ];
269
270 //I will kill you   ok
271 sentence s0041 =
272         eSA_THREAT
273         [  tbegGeneiw tThreatsExpr  ]
274         [ ^tbegGeneiw ^tThreatsExpr ];
275
276 //what s the matter james //what s your choice james ok
277 sentence s0042 =
278         eSA_THREAT
279         [   tstartQuestw  tThreatsquest  tActor  ]
280         [ ^tstartQuestw  ^tThreatsquest ^tActor  ];
281
282 //You have no choice ok
283 sentence s0044 =
284         eSA_THREAT
285         [   tbegGeneOwn tThreatsExpr  ]
286         [ ^tbegGeneOwn ^tThreatsExpr  ];
287
288
289 //See you in hell james ok
290 sentence s0045 =
291         eSA_THREAT
292         [  tThreatsExpr tActor  ]
293         [ ^tThreatsExpr ^tActor  ];
294
295 //See you in hell ok
296 sentence s1045 =
297         eSA_THREAT
298         [  tThreatsExpr ]
299         [ ^tThreatsExpr  ];
300
301
302 //You cant win ****
303 sentence s0046 =
304         eSA_THREAT
305         [  tbegNeg tThreatsExpr  ]
306         [ ^tbegNeg  ^tThreatsExpr ];
307
308
309 //try to kill me //go and pick it up ok
310 sentence s0047 =
311         eSA_THREAT
312         [ tThreatsStart1  ]
313         [ ^tThreatsStart1  ];
314
315
316
317 //What are you waiting for? ok
318 sentence s0049 =
```

```
319          eSA_THREAT
320          [   tstartQuestw tThreatsExpr  ]
321          [ ^tstartQuestw ^tThreatsExpr  ];
322
```

...

```
392  /*
393   *Denials
394   */
395
396  //How would I know  Mr Bond ok
397  //Why would I give you such information ok
398  sentence s0061  =
399          eSA_DENIALS
400          [   tstartQuestw tDenialsProp tActor   ]
401          [ ^tstartQuestw ^tDenialsProp ^tActor  ];
402
403  //I could not tell you //I would not know ok
404  sentence s0062 =
405          eSA_DENIALS
406          [  tbegNeg tDenialsEnd   ]
407          [ ^tbegNeg ^tDenialsEnd  ];
408
409
410
411  //Never heard of it *****
412  sentence s0064 =
413          eSA_DENIALS
414          [ tDenialsExpr ]
415          [ ^tDenialsExpr ];
416
417  //Do you seriously think i would tell you ok
418  sentence s0065 =
419          eSA_DENIALS
420          [  tstartQuesta tDenialsEnd tbegGeneiw tDenialsEnd ]
421          [ ^tstartQuesta ^tDenialsEnd ^tbegGeneiw ^tDenialsEnd ];
422
423  //I am not telling you this ok
424  //I am not talking to you
425  sentence s0066 =
426          eSA_DENIALS
427          [  tbegNeg tDenialsing  ]
428          [ ^tbegNeg ^tDenialsing ];
429          //I am not talking to you, Mr Bond
430  sentence s0068 =
431          eSA_DENIALS
432          [  tbegNeg tDenialsing tActor  ]
433          [ ^tbegNeg ^tDenialsing ^tActor ];
434
435  //Why would I tell you ok
436  sentence s0067 =
437          eSA_DENIALS
438          [   tstartQuestw "i" tDenialsEnd  ]
439          [ ^tstartQuestw ^tDenialsEnd  ];
440
441
442
443   //This is no business of yours //This is none of your business ok
444  sentence s0070 =
445          eSA_DENIALS
446          [  tbegGeneBe tDenialsExpr  ]
447          [ ^tbegGeneBe ^tDenialsExpr   ];
448
449
450  //It is not your buisiness //This is not ur buisiness  ok
```

```
451 sentence s0072 =
452         eSA_DENIALS
453         [  tbegNeg tDenialsExpr  ]
454         [ ^tbegNeg ^tDenialsExpr ];
455
456 //You're wasting // your time ok
457 sentence s0075 =
458         eSA_DENIALS
459         [  tbegGeneBe tDenialsing  ]
460         [ ^tbegGeneBe ^tDenialsing  ];
461
462
463 //You are not serious *  //You don't want to know *
464 //You are not able to know that ok
465 sentence s0078 =
466         eSA_DENIALS
467         [  tbegNeg  tDenialsExpr  ]
468         [ ^tbegNeg ^tDenialsExpr  ];
469
470 //You are joking //it is confidential ok
471 sentence s0079 =
472         eSA_DENIALS
473         [  tbegGeneBe tDenialsExpr  ]
474         [ ^tbegGeneBe ^tDenialsExpr   ];
475
476 //You must be joking ok
477 sentence s0080 =
478         eSA_DENIALS
479         [  "you" "must" "be" tDenialsExpr   ]
480         [ ^tDenialsExpr  ];
481
482 //Are you serious, Mr Bond ***
483 sentence s0081 =
484         eSA_DENIALS
485         [  tstartQuesta tDenialsExpr  tActor  ]
486         [ ^tstartQuesta ^tDenialsExpr  ^tActor  ];
487
488 //Why would you like to know ok
489 sentence s0082 =
490         eSA_DENIALS
491         [  tstartQuestw tDenialsProp   ]
492         [ ^tstartQuestw ^tDenialsProp   ];
493
494 //Why do you care, Mr Bond ok
495 sentence s0083 =
496         eSA_DENIALS
497         [   tstartQuestw tDenialsProp tActor  ]
498         [ ^tstartQuestw ^tDenialsProp ^tActor  ];
499
500 //Why are you interested ok
501 sentence s0084 =
502         eSA_DENIALS
503         [   tstartQuestw  tDenialsProp  ]
504         [ ^tstartQuestw ^tDenialsProp   ];
505
506 //Why do you want to know ****
507 sentence s0085 =
508         eSA_DENIALS
509         [   tstartQuestw tDenialsExpr  ]
510         [ ^tstartQuestw ^tDenialsExpr  ];
511
512 //I won t tell you bond ok//I can t tell youok
513 //You don t need to know that ok
514 sentence s0086 =
515         eSA_DENIALS
516         [  tbegNeg tDenialsEnd tActor ]
517         [ ^tbegNeg ^tDenialsEnd ^tActor  ];
518
519 //sentence s0086 =
```

```
520 //       eSA_DENIALS
521 //       [  tbegNeg tDenialsEnd ]
522 //       [ ^tbegNeg ^tDenialsEnd  ];
523
524
525
526 //what are u talkin about ok
527 sentence s0091 =
528         eSA_DENIALS
529         [   tstartQuestw tDenialsing   ]
530         [ ^tstartQuestw ^tDenialsing  ];
531
```

...

Appendix I

Templates definition charts

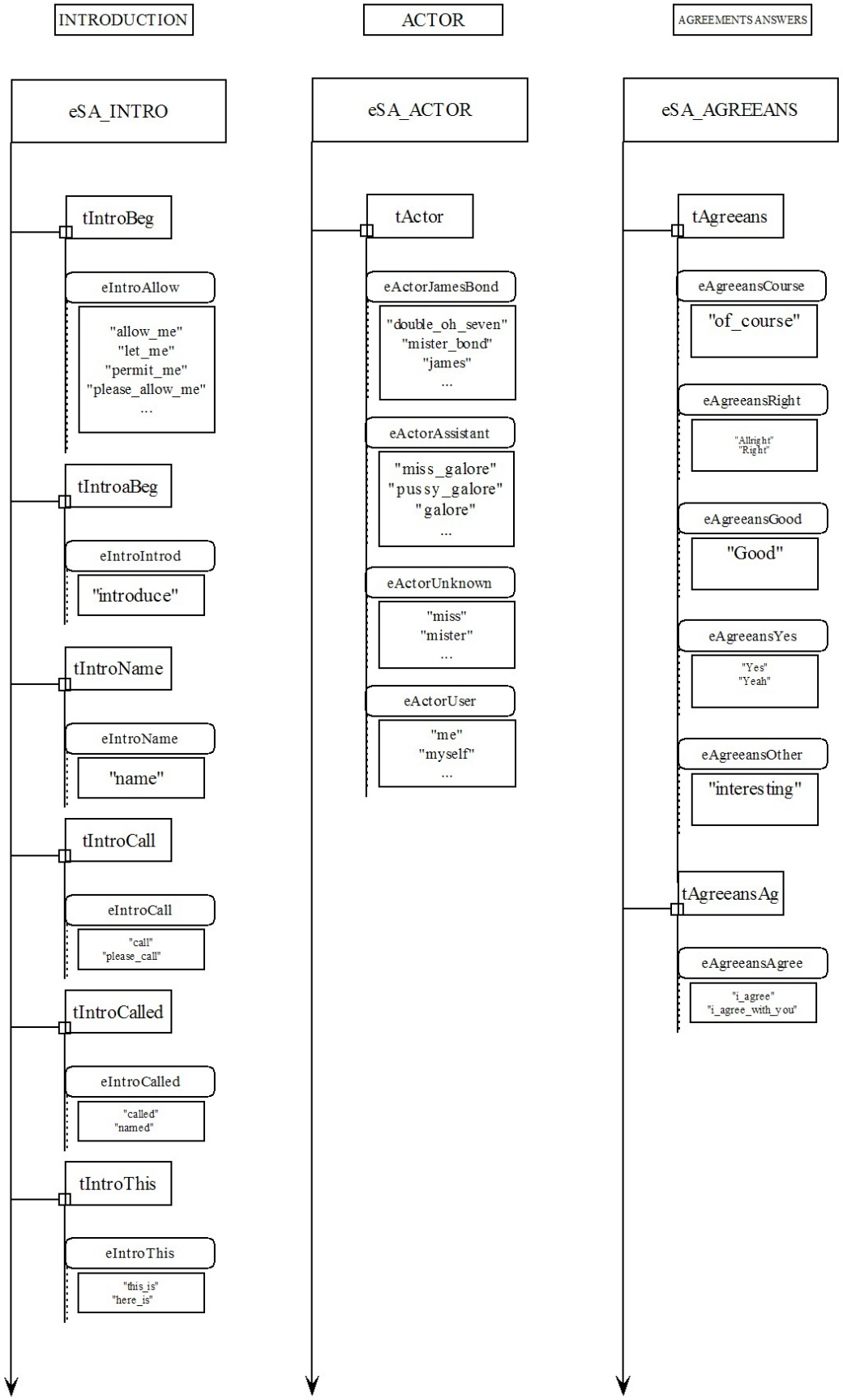

Figure I.1: Agreements, Actor and Introduction Templates Definion Chart

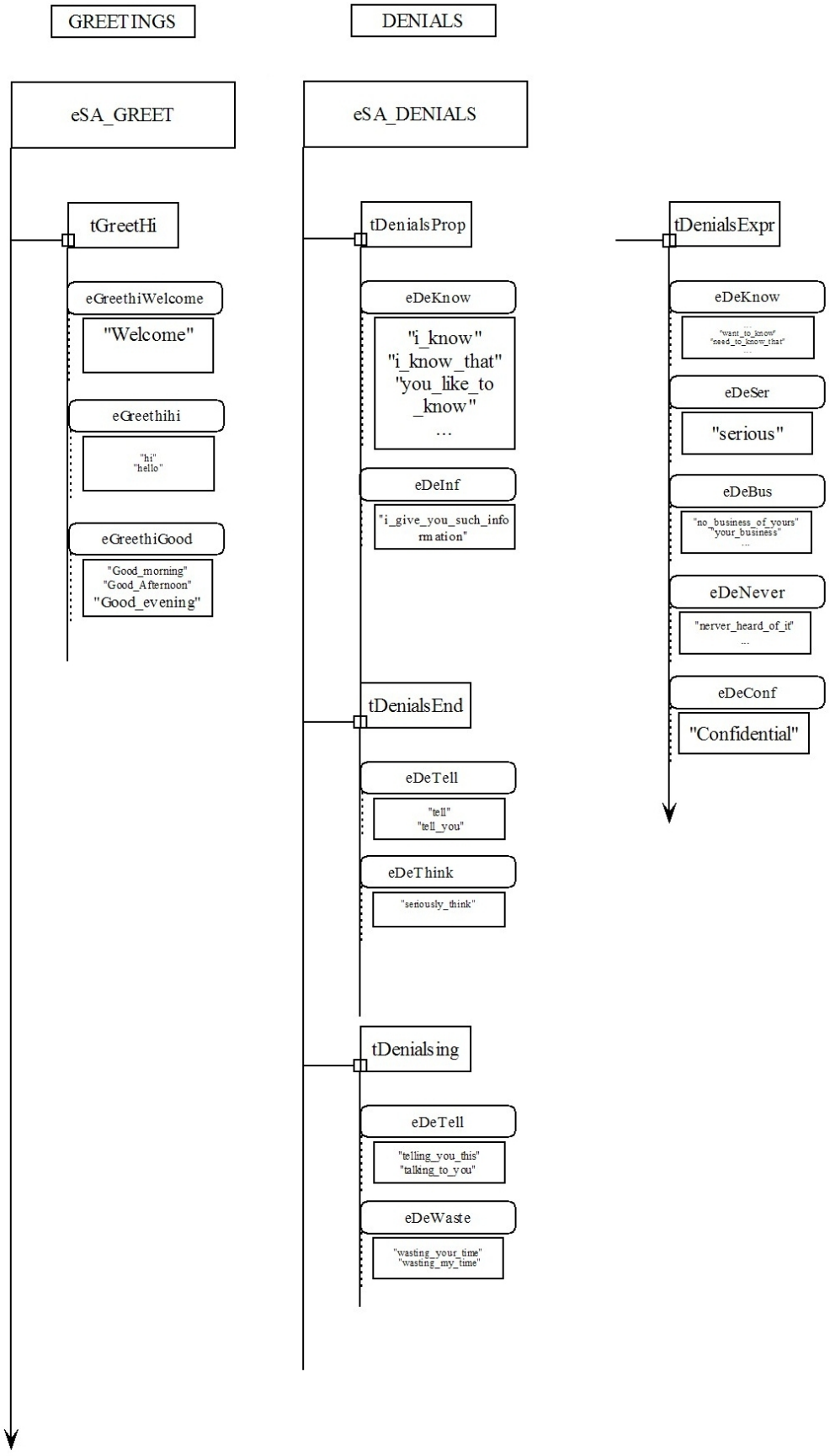

Figure I.2: Denials and Greetings Templates Definion Chart

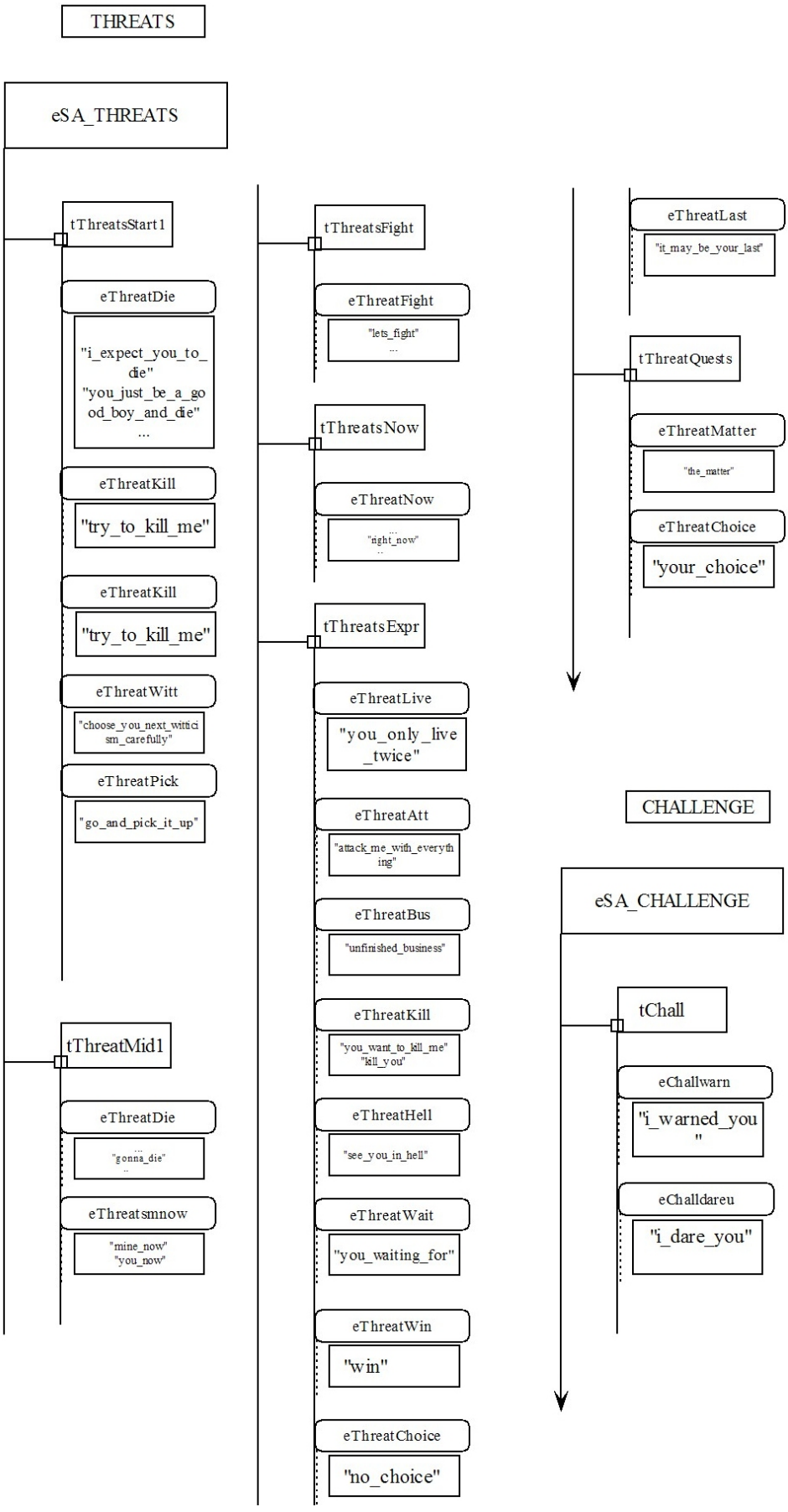

Figure I.3: Threats and Challenge Templates Definion Chart

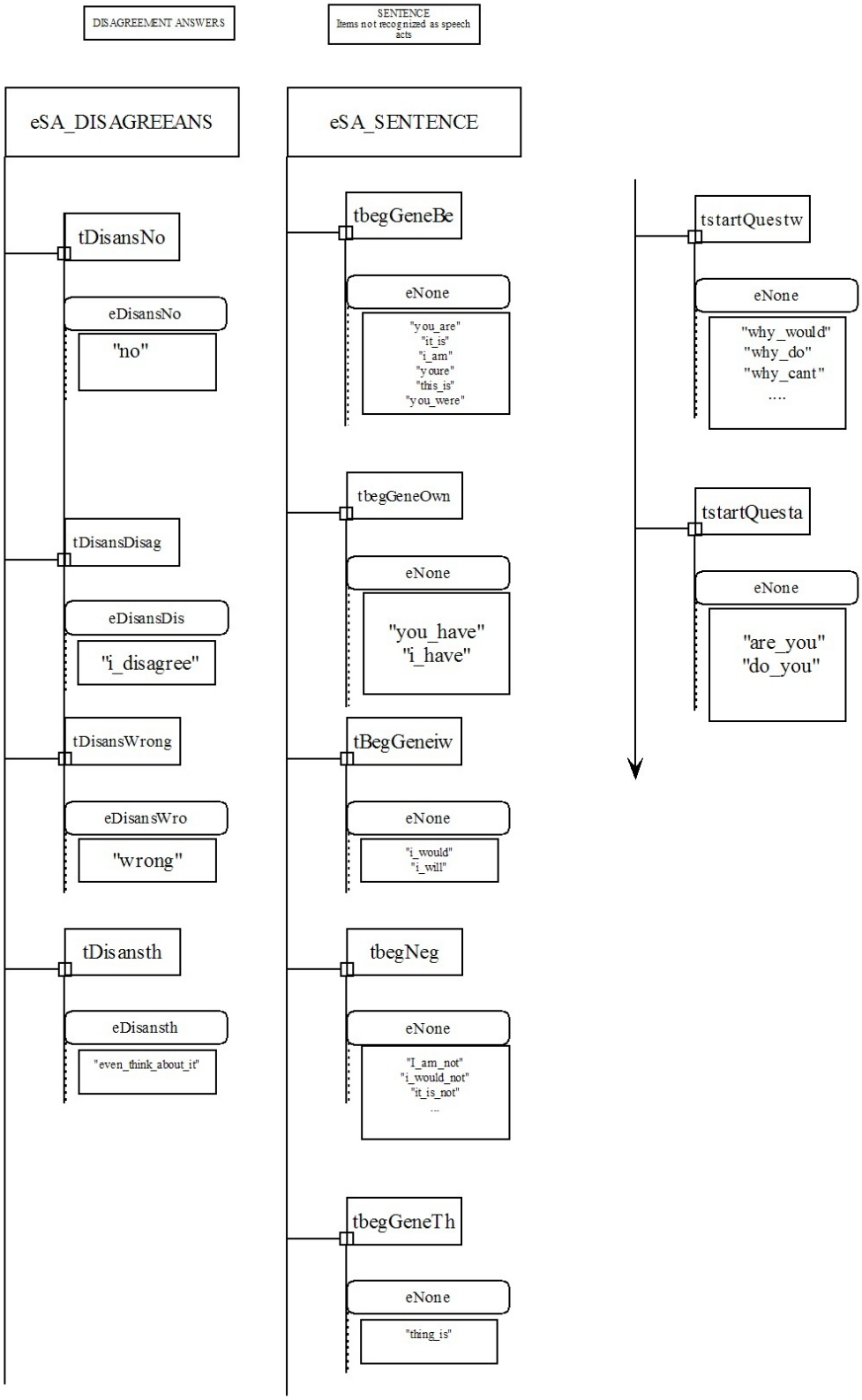

Figure I.4: Disagreement Answers and Sentences Templates Definion Chart

Appendix J

Talk to unreal application screenshot

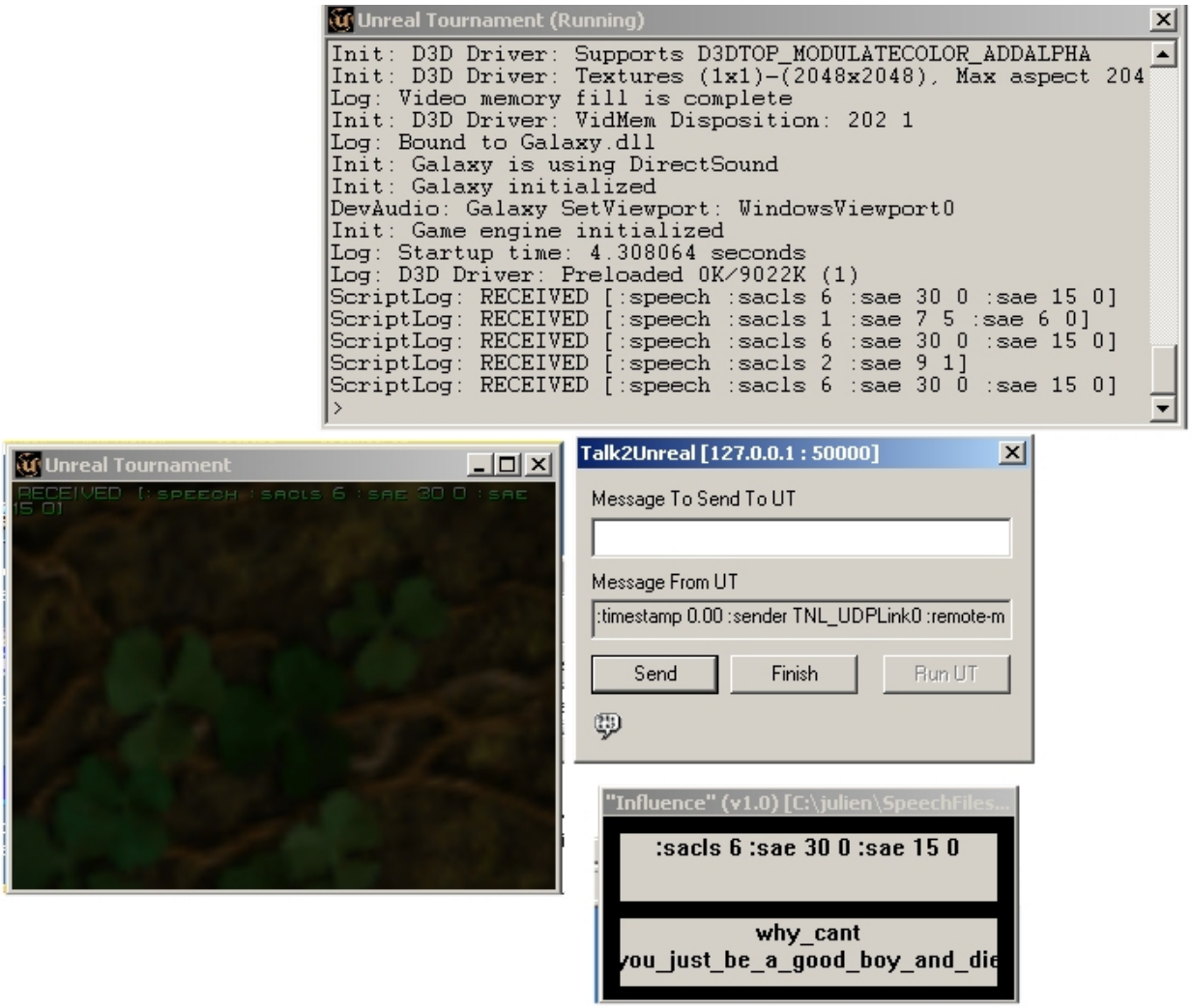

Figure J.1: Talk to unreal application screenshot