Chatbot

Generated by Doxygen 1.8.14

Contents

Chapter 1

Main Page

ELIZA & Knowledge-based agent documentation

This documentation was made to go hand in hand with the files of our projectso as to be of use to any future developer wanting to work on it while gaining a deeper insight into the different components of this project.

Chapter 2

Hierarchical Index

2.1 Class Hierarchy

This inheritance list is sorted roughly, but not completely, alphabetically:

Chapter 3

Class Index

3.1 Class List

Here are the classes, structs, unions and interfaces with brief descriptions:

Chapter 4

File Index

4.1 File List

Here is a list of all documented files with brief descriptions:

Chapter 5

Class Documentation

5.1 Action Class Reference

Represents an action undertaken by the knowledge-based agent.

```
#include <Action.h>
```

Public Member Functions

- String makeActionText (vector< Sentence > a)

    Converts an action its logical form to its string form.

Static Public Member Functions

- static vector< Sentence > makeActionQuery (KB kb, int t)

    This function generates an action from a query at a given time.

Public Attributes

- int id

Private Attributes

- String string_feed
- vector< Sentence > logic_feed

5.1.1 Detailed Description

Represents an action undertaken by the knowledge-based agent.

Due to the agent being a conversational one, the action will be represented by a string.

5.1.2 Member Function Documentation

5.1.2.1 makeActionQuery()

```
vector< Sentence > Action::makeActionQuery (
            KB kb,
            int t )  [static]
```

This function generates an action from a query at a given time.

Parameters

kb	Represents the knowledge base.
t	Elapsed time.

Returns

An action in it's logical form.

5.1.2.2 makeActionText()

```
String Action::makeActionText (
            vector< Sentence > a )
```

Converts an action its logical form to its string form.

Parameters

a	The action expressed in logical terms.

Returns

Text form of an action.

5.1.3 Member Data Documentation

5.1.3.1 id

```
int Action::id
```

Each action bears a unique number

5.1.3.2 logic_feed

```
vector<Sentence> Action::logic_feed  [private]
```

Represents the logical form of the action

5.1.3.3 string_feed

```
String Action::string_feed  [private]
```

Represents the text form of the action

The documentation for this class was generated from the following files:

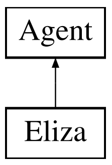
- src/Agent/KnowledgeBase/Action.h
- src/Agent/KnowledgeBase/Action.cpp

5.2 Agent Class Reference

Processes input speech and generates output.

```
#include <Agent.h>
```

Inheritance diagram for Agent:



Public Member Functions

- Agent (istream ∗input, ostream ∗output)
- void run (bool debug=false)

Public Attributes

- String name

    Agent name.
- istream ∗ inputStream

    pointer to input stream
- ostream ∗ outputStream

    pointer to output stream
- bool quit

    boolean to quit conversation

Protected Member Functions

- virtual String processInput (String input)=0
- virtual String greetUser ()=0

Protected Attributes

- ostream ∗ debugger

  Pointer to output stream for displaying debug information.
  Can be used to point to a file stream to write a log file.

5.2.1 Detailed Description

Processes input speech and generates output.

5.2.2 Constructor & Destructor Documentation

5.2.2.1 Agent()

```
Agent::Agent (
              istream ∗ input,
              ostream ∗ output )
```

Agent constructor that initializes I/O streams

Parameters

input	Pointer to input stream
output	Pointer to output stream

5.2.3 Member Function Documentation

5.2.3.1 greetUser()

```
virtual String Agent::greetUser ( )  [protected], [pure virtual]
```

Generates greeting at the beginning of the conversation.

Returns

output string.

Implemented in Eliza.

5.2.3.2 processInput()

```
virtual String Agent::processInput (
            String input )  [protected], [pure virtual]
```

Processes input string and generates response.

Parameters

input	User input

Returns

Processed output

Implemented in Eliza.

5.2.3.3 run()

```
void Agent::run (
            bool debug = false )
```

Runs agent until Agent.quit is true.

Parameters

debug	if true, displays running processes.

The documentation for this class was generated from the following files:

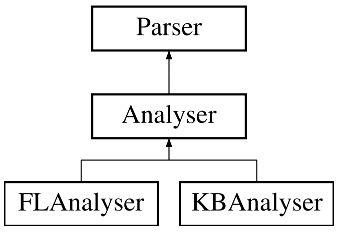
- src/Agent/Agent.h
- src/Agent/Agent.cpp

5.3 Analyser Class Reference

The analyser is responsible for the different parsing jobs that are used by the agent.

`#include <Analyser.h>`

Inheritance diagram for Analyser:



Public Member Functions

- ParseTree parse (String words, Grammar grammar)

  This function creates a parse tree given a set of words and a grammar.

Public Attributes

- Lexer lexer

Additional Inherited Members

5.3.1 Detailed Description

The analyser is responsible for the different parsing jobs that are used by the agent.

5.3.2 Member Function Documentation

5.3.2.1 parse()

```
ParseTree Analyser::parse (
             String words,
             Grammar grammar )
```

This function creates a parse tree given a set of words and a grammar.

Parameters

words	The words to be transformed into a parse tree.
grammar	The grammar of the language in question.

Returns

A parse tree.

5.3.3 Member Data Documentation

5.3.3.1 lexer

```
Lexer Analyser::lexer
```

The lexer that is used by the analyser.

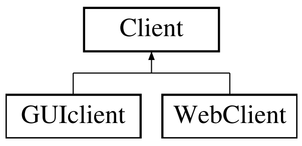
The documentation for this class was generated from the following file:

- src/Agent/KnowledgeBase/Analyser.h

5.4 Client Class Reference

```
#include <Client.h>
```

Inheritance diagram for Client:



5.4.1 Detailed Description

Project ChatBot

The documentation for this class was generated from the following file:

- src/Client/Client.h

5.5 Decomp Class Reference

Decomposition rule for a sentence.

```
#include <Decomp.h>
```

Public Member Functions

- Decomp (Key ∗key, String scriptLine, Thesaurus thes)
- void newReasmb (String reasmb)
- vector< String > decompose (String str)
- Reasmb ∗ nextRule ()

Public Attributes

- String pattern

  Decomposition REGEX pattern.
- vector< Reasmb ∗ > reassemb

  List of reassembly rules for decomposition.
- Key ∗ key

  Pointer to parent Key.
- bool mem

  save decomposition rule in memory ?

Private Attributes

- size_t reassembRule = -1

  current reassembly rule index in Decomp.reassemb, -1 if not assigned.

Friends

- ostream & **operator**<< (ostream &os, const Decomp &decomp)

5.5.1 Detailed Description

Decomposition rule for a sentence.

5.5.2 Constructor & Destructor Documentation

5.5.2.1 Decomp()

```
Decomp::Decomp (
            Key * key,
            String scriptLine,
            Thesaurus thes )
```

Default constructor, translates script pattern (special format) into REGEX.

Parameters

key	parent Key
scriptLine	output string from Script::extractPattern
thes	Thesaurus object

5.5.3 Member Function Documentation

5.5.3.1 decompose()

```
vector< String > Decomp::decompose (
            String str )
```

Decomposes a sentence according.

Parameters

str	input sentence

Returns

vector of matching expressions

5.5.3.2 newReasmb()

```
void Decomp::newReasmb (
            String reasmb )
```

Creates new reassembly object, adds it to Decomp.reassemb and links it with the parent Decomp object.

Parameters

reasmb	reassembly rule pattern.

5.5.3.3 nextRule()

```
Reasmb * Decomp::nextRule ( )
```

Returns

pointer to a random rassembly rule

The documentation for this class was generated from the following files:

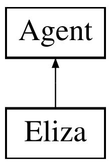
- src/Agent/ELIZA/Decomp.h
- src/Agent/ELIZA/Decomp.cpp

5.6 Eliza Class Reference

Agent based on Weizenbaum's ELIZA conversational agent.

`#include <Eliza.h>`

Inheritance diagram for Eliza:



Public Member Functions

- Eliza (istream ∗input, ostream ∗output, String sourcePath)

Public Attributes

- Script ∗ script

  Database parser for ELIZA.
- Memory memory

  Memory stack.

Private Member Functions

- String processInput (String input) override
- String greetUser () override
- vector< Key ∗ > collectKeys (String input)
- String processSentence (String input)
- String decomposeOnKey (Decomp ∗decomp, String input)

Additional Inherited Members

5.6.1 Detailed Description

Agent based on Weizenbaum's ELIZA conversational agent.

5.6.2 Constructor & Destructor Documentation

5.6.2.1 Eliza()

```
Eliza::Eliza (
            istream ∗ input,
            ostream ∗ output,
            String sourcePath )
```

Default constructor for Eliza. Calls inherited constructor and sets Agent.name and memory. See Agent.Agent()

5.6.3 Member Function Documentation

5.6.3.1 collectKeys()

```
vector< Key ∗ > Eliza::collectKeys (
            String input ) [private]
```

Collects keywords from user input.

Parameters

input	user input

Returns

keywords (sorted in descending order according to Key::rank)

5.6.3.2 decomposeOnKey()

```
String Eliza::decomposeOnKey (
            Decomp * decomp,
            String input )   [private]
```

Decomposes input string on given decomposition rule. Calls:

- vector<String> decomp::decompose(String input)
- Reasmb∗ decomp::nextRule()
- String reasmb::reassemble(vector<String> matches)

Parameters

decomp	pointer to decomposition rule
input	input string

Returns

reassembled string

5.6.3.3 greetUser()

```
String Eliza::greetUser ( )   [override], [private], [virtual]
```

Generates greeting at the beginning of the conversation.

Returns

output string.

Implements Agent.

5.6.3.4 processInput()

```
String Eliza::processInput (
            String input )   [override], [private], [virtual]
```

Processes input string and generates response.

Parameters

input	User input

Returns

Processed output

Implements Agent.

5.6.3.5 processSentence()

```
String Eliza::processSentence (
            String input )  [private]
```

Process individual sentences from input.

Parameters

input	broken user sentence

Returns

processed answer

The documentation for this class was generated from the following files:

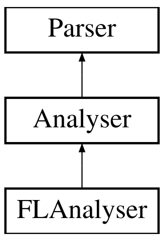
- src/Agent/ELIZA/Eliza.h
- src/Agent/ELIZA/Eliza.cpp

5.7 FLAnalyser Class Reference

This class uses the parse tree generated by the parser and translates it into a formal language (a small set of the English language in this case)

```
#include <FLAnalyser.h>
```

Inheritance diagram for FLAnalyser:



Public Member Functions

- vector< Sentence > interpret ()

  This function interprets the parse tree and gives equivalent possible intepretations of the tree.
- vector< Sentence > disambiguate (vector< vector< Sentence >> ps)

  This function disambiguates the different interpreted sentences in order to pick the one that makes the most sense.

Public Attributes

- ParseTree t

Additional Inherited Members

5.7.1 Detailed Description

This class uses the parse tree generated by the parser and translates it into a formal language (a small set of the English language in this case)

5.7.2 Member Function Documentation

5.7.2.1 disambiguate()

```
vector< Sentence > FLAnalyser::disambiguate (
            vector< vector< Sentence >> ps )
```

This function disambiguates the different interpreted sentences in order to pick the one that makes the most sense.

Parameters

ps	A set of possible sentences.

Returns

The correct intepretation of a sentence.

5.7.2.2 interpret()

```
vector< Sentence > FLAnalyser::interpret ( )
```

This function interprets the parse tree and gives equivalent possible intepretations of the tree.

Returns

Intepreted sentence.

5.7.3 Member Data Documentation

5.7.3.1 t

```
ParseTree FLAnalyser::t
```

Parse tree used by the formal language analyser.

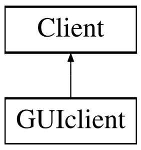
The documentation for this class was generated from the following files:

- src/Agent/KnowledgeBase/FLAnalyser.h
- src/Agent/KnowledgeBase/FLAnalyser.cpp

5.8 GUIclient Class Reference

```
#include <GUIclient.h>
```

Inheritance diagram for GUIclient:



Public Member Functions

- void Main (int exit_status)

5.8.1 Detailed Description

Project ChatBot

5.8.2 Member Function Documentation

5.8.2.1 Main()

```
void GUIclient::Main (
             int exit_status )
```

Parameters

exit_status	Project ChatBot GUIclient implementation
exit_status	

The documentation for this class was generated from the following files:

- src/Client/GUI/GUIclient.h
- src/Client/GUI/GUIclient.cpp

5.9 KB Class Reference

This class represents our knowledge base.

```
#include <KB.h>
```

Public Member Functions

- void tell (KB kb, vector< Sentence > s)

  This function adds a new rule to the knowledge base.
- vector< Sentence > ask (KB kb, vector< Sentence > s)

  This function queries the knowledge base and presents the best course of action for the agent to undertake.
- bool entails (KB kb, vector< Sentence > s)

  This function checks if a new rule/sentence is logical accordance with the existant rules.
- void forwardChain (KB kb, vector< Sentence > s)

  This function is responsible for deducing new sentences/rules from already existing ones by using the forward chain algorithm.
- bool backwardChain (KB kb, vector< Sentence > query)

  This function is used in itself by the query mechanism in order to infer whether a certain course of action is valid of production following the rules.
- vector< Sentence > train (vector< vector< Sentence >> examples)

  This function is used to train the agent (by altering the knowledge base) based on different given hypothesis.
- int nbRules (KB kb)

  Calculates the number of rules in a knowledge base.
- int nbFacts (KB kn)

  Calculates the number of facts present in a knowledge base.

Private Attributes

- vector< Sentence > facts
- KBAnalyser analyser
- CNF cnfclause
- vector< Rule > rules

5.9.1 Detailed Description

This class represents our knowledge base.

The knowledge base holds all the logical sentences and facts which can later be used to infer new facts, ask the database and train the aforementioned.

5.9.2 Member Function Documentation

5.9.2.1 ask()

```
vector< Sentence > KB::ask (
            KB kb,
            vector< Sentence > s )
```

This function queries the knowledge base and presents the best course of action for the agent to undertake.

Parameters

kb	The knowledge base.
s	A question in the form of a logical sentence.

Returns

An action in the form of a sentence.

5.9.2.2 backwardChain()

```
bool KB::backwardChain (
            KB kb,
            vector< Sentence > query )
```

This function is used in itself by the query mechanism in order to infer whether a certain course of action is valid of production following the rules.

Parameters

kb	The knowledge base
query	A question in the form of a logical sentence

Returns

true If the proposed query is valid.
false If the proposed query bears logical invalidity.

5.9.2.3 entails()

```
bool KB::entails (
            KB kb,
            vector< Sentence > s )
```

This function checks if a new rule/sentence is logical accordance with the existant rules.

Parameters

kb	The knowledge base.
s	A logical sentence.

Returns

true If the sentence is in accordance with the existant rules.
false If the sentence clashes with the existant rules.

5.9.2.4 forwardChain()

```
void KB::forwardChain (
            KB kb,
            vector< Sentence > s )
```

This function is responsible for deducing new sentences/rules from already existing ones by using the forward chain algorithm.

Parameters

kb	The knowledge base.
s	A logical sentence.

5.9.2.5 nbFacts()

```
int KB::nbFacts (
            KB kn )
```

Calculates the number of facts present in a knowledge base.

Parameters

kn	The knowledge base.

Returns

int The calculated number of facts.

5.9.2.6 nbRules()

```
int KB::nbRules (
             KB kb )
```

Calculates the number of rules in a knowledge base.

Parameters

kb	The knowledge base.

Returns

int The calculated number of rules.

5.9.2.7 tell()

```
void KB::tell (
             KB kb,
             vector< Sentence > s )
```

This function adds a new rule to the knowledge base.

Parameters

kb	The knowledge base.
s	A rule in the form of a logical sentence.

5.9.2.8 train()

```
vector< Sentence > KB::train (
             vector< vector< Sentence >> examples )
```

This function is used to train the agent (by altering the knowledge base) based on different given hypothesis.

Parameters

examples	A set of hypothesis.

Returns

The best possible hypothesis.

5.9.3 Member Data Documentation

5.9.3.1 analyser

`KBAnalyser KB::analyser [private]`

The First Order Logic analyser used by the knowledge base.

5.9.3.2 cnfclause

`CNF KB::cnfclause [private]`

A collection of facts expressed in their conjunctive normal form.

5.9.3.3 facts

`vector<Sentence> KB::facts [private]`

A collection of the facts contained in the knowledge base.

5.9.3.4 rules

`vector<Rule> KB::rules [private]`

A collection of rules used by the inference engine

The documentation for this class was generated from the following files:

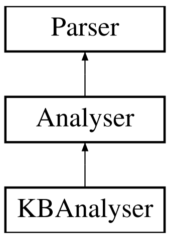
- src/Agent/KnowledgeBase/KB.h
- src/Agent/KnowledgeBase/KB.cpp

5.10 KBAnalyser Class Reference

This class represents the analyser responsible for parsing the First Order Logic language.

`#include <KBAnalyser.h>`

Inheritance diagram for KBAnalyser:



Public Member Functions

- KBAnalyser ()

    Constructs a new KBAnalyser object.

Additional Inherited Members

5.10.1 Detailed Description

This class represents the analyser responsible for parsing the First Order Logic language.

The documentation for this class was generated from the following files:

- src/Agent/KnowledgeBase/KBAnalyser.h
- src/Agent/KnowledgeBase/KBAnalyser.cpp

5.11 Key Class Reference

Public Member Functions

- Key (const String &name, int rank)

    Default constructor.
- Decomp $*$ newDecomp (String scriptLine, Thesaurus thesaurus)
- Decomp $*$ findDecomp (String str)

Public Attributes

- String name

    Unique key name (identifier)
- int rank

    Key rank.
- vector$<$ Decomp $*$ $>$ decomp

    List of decomposition rules for keyword.

Friends

- ostream & **operator**$<<$ (ostream &os, const Key &key)

5.11.1 Member Function Documentation

5.11.1.1 findDecomp()

```
Decomp * Key::findDecomp (
            String str )
```

Finds matching decomposition rule for a given string on a keyword using REGEX.

Parameters

str	Raw string

Returns

pointer to matching Decomp, or nullptr if no match was found for Key object.

5.11.1.2 newDecomp()

```
Decomp * Key::newDecomp (
            String scriptLine,
            Thesaurus thesaurus )
```

Creates new decomposition object, adds it to Key.decomp and links it with the parent Key object.

Parameters

scriptLine	output string from Script::extractPattern
thesaurus	Script.thes

Returns

pointer to created Decomp

The documentation for this class was generated from the following files:

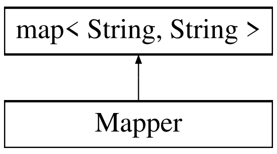
- src/Agent/ELIZA/Key.h
- src/Agent/ELIZA/Key.cpp

5.12 Mapper Class Reference

Hash table for pre/post script elements.

```
#include <Mapper.h>
```

Inheritance diagram for Mapper:



Public Member Functions

- void map (String src, String dst)
- String translate (String sentence)

5.12.1 Detailed Description

Hash table for pre/post script elements.

5.12.2 Member Function Documentation

5.12.2.1 map()

```
void Mapper::map (
            String src,
            String dst )
```

Adds a new element to hash table.

Parameters

src	key
dst	value

5.12.2.2 translate()

```
String Mapper::translate (
            String sentence )
```

Translates keywords in a sentence into their values from the hash table.

Parameters

sentence	string of words

Returns

Translated sentence

The documentation for this class was generated from the following files:

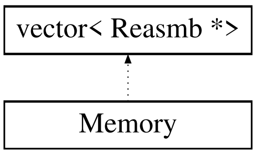
- src/Agent/ELIZA/Mapper.h
- src/Agent/ELIZA/Mapper.cpp

5.13 Memory Class Reference

FIFO stack of Reasmb objects.

`#include <Memory.h>`

Inheritance diagram for Memory:



Public Member Functions

- void save (Reasmb ∗)

  Saves new reassembly rule in memory.
- Reasmb ∗ pop ()

  Pops first reassembly rule from memory.

Private Attributes

- size_t max = 20

  Memory capacity.

5.13.1 Detailed Description

FIFO stack of Reasmb objects.

The documentation for this class was generated from the following files:

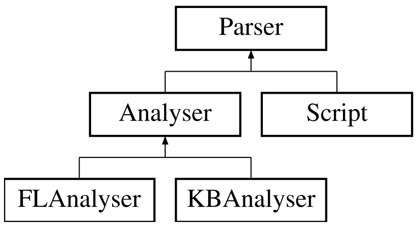
- src/Agent/ELIZA/Memory.h
- src/Agent/ELIZA/Memory.cpp

5.14 Parser Class Reference

Parses source file into appropriate data types.

`#include <Parser.h>`

Inheritance diagram for Parser:



Public Member Functions

- Parser (String sourcePath)

Public Attributes

- String sourcePath

  Source file path.

Protected Member Functions

- virtual void parse ()=0

5.14.1 Detailed Description

Parses source file into appropriate data types.

5.14.2 Constructor & Destructor Documentation

5.14.2.1 Parser()

```
Parser::Parser (
            String sourcePath )
```

Parser default constructor.

Parameters

sourcePath	source file path

5.14.3 Member Function Documentation

5.14.3.1 parse()

```
virtual void Parser::parse ( )  [protected], [pure virtual]
```

Called by constructor to parse data from source file.

Implemented in Script.

The documentation for this class was generated from the following files:

- src/Agent/Parser.h
- src/Agent/Parser.cpp

5.15 Percept Class Reference

This class represents a perception received by the agent (which is of an exclusive text form in our case).

```
#include <Percept.h>
```

Public Member Functions

- vector< Sentence > makePerceptSentence (Percept p)

  This function turns a percept into a logical sentence.

Public Attributes

- int id

Private Attributes

- String string_feed
- vector< Sentence > logic_feed

5.15.1 Detailed Description

This class represents a perception received by the agent (which is of an exclusive text form in our case).

5.15.2 Member Function Documentation

5.15.2.1 makePerceptSentence()

```
vector< Sentence > Percept::makePerceptSentence (
            Percept p )
```

This function turns a percept into a logical sentence.

Parameters

p	The percepts to be converted.

Returns

A logical sentence.

5.15.3 Member Data Documentation

5.15.3.1 id

```
int Percept::id
```

Each percept bears a unique number.

5.15.3.2 logic_feed

```
vector<Sentence> Percept::logic_feed  [private]
```

Represents the logical form of a percept.

5.15.3.3 string_feed

```
String Percept::string_feed  [private]
```

Represents the text form of a percept.

The documentation for this class was generated from the following files:

- src/Agent/KnowledgeBase/Percept.h
- src/Agent/KnowledgeBase/Percept.cpp

5.16 Reasmb Class Reference

Reassembly rule for decomposed sentence.

```
#include <Reasmb.h>
```

Public Member Functions

- **Reasmb** (Decomp $*$decomp, const String &rule)
- String reassemble (vector$<$ String $>$ matches)

Public Attributes

- Decomp $*$ decomp

  Pointer to parent decomposition rule.
- String rule

  Reassembly rule pattern.

Friends

- ostream & **operator**<< (ostream &os, const Reasmb &reasmb)

5.16.1 Detailed Description

Reassembly rule for decomposed sentence.

5.16.2 Member Function Documentation

5.16.2.1 reassemble()

```
String Reasmb::reassemble (
            vector< String > matches )
```

Reassembles sentence from matching expressions.

Parameters

matches	output of Decomp::decompose

Returns

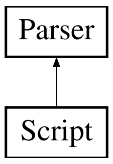
reassembled response

The documentation for this class was generated from the following files:

- src/Agent/ELIZA/Reasmb.h
- src/Agent/ELIZA/Reasmb.cpp

5.17 Script Class Reference

Inheritance diagram for Script:



Public Member Functions

- **Script** (const String &sourcePath)
- String pre_translate (String str)
- String post_translate (String str)
- Key ∗ getKey (String word)

Public Attributes

- String initial

    Initial string to greet user.
- String final

    Final string to bid farewell to user.
- Mapper pre

    Pre-processing map database.
- Mapper post

    Post-processing map database.
- vector< Key * > keys

    List of keywords in database.
- vector< String > quit

    List of strings that the user can enter to end the conversation.
- Thesaurus thes

    Thesaurus (list of synonyms) in database.

Private Member Functions

- void parse () override
- String extractPattern (String line, String key)
- Key * newKey (String scriptLine)

Friends

- ostream & **operator**<< (ostream &os, const Script &parser)

Additional Inherited Members

5.17.1 Member Function Documentation

5.17.1.1 extractPattern()

```
String Script::extractPattern (
            String line,
            String key )  [private]
```

Extracts pattern definition from a line from the script file.

Parameters

line	line from script file
key	script element identifier key

Returns

extracted pattern if the given key is in line, an empty string otherwise.

Usage example:

```
line = "reasmb: Do you feel strongly about discussing such things ?";
pattern = extractPattern(line, "reasmb"); // pattern = "Do you feel strongly about discussing
        such things ?"
pattern = extractPattern(line, "decomp"); // pattern = ""
```

5.17.1.2 getKey()

```
Key * Script::getKey (
            String word )
```

Finds keyword in database

Parameters

word	given word

Returns

associated Key object

5.17.1.3 newKey()

```
Key * Script::newKey (
            String scriptLine ) [private]
```

Creates new Key object and adds it to Script.keys

Parameters

scriptLine	output string from Script::extractPattern

Returns

pointer to created Key

5.17.1.4 parse()

```
void Script::parse ( )  [override], [private], [virtual]
```

Called by constructor to parse data from source file.

Implements Parser.

5.17.1.5 post_translate()

```
String Script::post_translate (
             String str )
```

Post-translates output string

Parameters

output	raw output string

Returns

processed output

5.17.1.6 pre_translate()

```
String Script::pre_translate (
             String str )
```

Pre-translates input string

Parameters

input	user's raw input

Returns

processed input

The documentation for this class was generated from the following files:

- src/Agent/ELIZA/Script.h
- src/Agent/ELIZA/Script.cpp

5.18 Sentence Class Reference

Class representing a sentence.

`#include <Sentence.h>`

5.18.1 Detailed Description

Class representing a sentence.

The documentation for this class was generated from the following file:

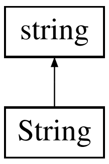
- src/Agent/KnowledgeBase/Sentence.h

5.19 String Class Reference

An extended string class with useful methods.

`#include <String.h>`

Inheritance diagram for String:



Public Member Functions

- **String** (const string &)
- **operator int** () const
- vector< String > split ()

  Splits string by whitespace into a vector of strings.
- vector< String > split (char)

  Splits string by a given character into a vector of strings.
- void lower ()

  Turns string into lowercase.
- void replaceStr (const String &src, const String &dst)

5.19.1 Detailed Description

An extended string class with useful methods.

5.19.2 Member Function Documentation

5.19.2.1 replaceStr()

```
void String::replaceStr (
            const String & src,
            const String & dst )
```

Replaces all instances of a string into another

Parameters

src	string to be replaced
dst	string to replace src

The documentation for this class was generated from the following files:

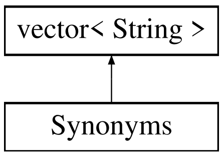
- src/utils/String.h
- src/utils/String.cpp

5.20 Synonyms Class Reference

List of synonyms.

`#include <Synonyms.h>`

Inheritance diagram for Synonyms:



Public Member Functions

- Synonyms (const String word)
  Default constructor: creates an empty list and adds.
- Synonyms (const vector< String > &__x)
  Constructor: creates list from a vector of words.
- String asRegex ()
- bool hasWord (String word)

Friends

- ostream & **operator**<< (ostream &os, const Synonyms &synonyms)

5.20.1 Detailed Description

List of synonyms.

5.20.2 Constructor & Destructor Documentation

5.20.2.1 Synonyms()

```
Synonyms::Synonyms (
          const String word )
```

Default constructor: creates an empty list and adds.

Parameters

word	into the list.

5.20.3 Member Function Documentation

5.20.3.1 asRegex()

```
String Synonyms::asRegex ( )
```

Translates synonyms list into a REGEX group expression.

Returns

words list separated by "|"

5.20.3.2 hasWord()

```
bool Synonyms::hasWord (
             String word )
```

Searches for a word in synonyms list.

Parameters

word	

Returns

True if word in list, False otherwise

The documentation for this class was generated from the following files:

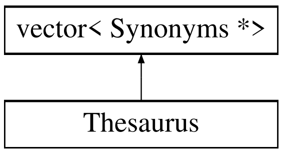
- src/Agent/ELIZA/Synonyms.h
- src/Agent/ELIZA/Synonyms.cpp

5.21 Thesaurus Class Reference

List of Synonyms objects.

```
#include <Thesaurus.h>
```

Inheritance diagram for Thesaurus:



Public Member Functions

- Synonyms ∗ findSynonyms (String word)

Friends

- ostream & **operator**<< (ostream &os, const Thesaurus &thesaurus)

5.21.1 Detailed Description

List of Synonyms objects.

5.21.2 Member Function Documentation

5.21.2.1 findSynonyms()

```
Synonyms * Thesaurus::findSynonyms (
             String word )
```

Finds Synonyms object containing a given word. Calls Synonyms::hasWord
If none found, returns a new synonyms list containing only the given word.

Parameters

word	

Returns

pointer to synonyms object containing word

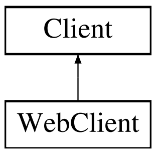
The documentation for this class was generated from the following files:

- src/Agent/ELIZA/Thesaurus.h
- src/Agent/ELIZA/Thesaurus.cpp

5.22 WebClient Class Reference

```
#include <WebClient.h>
```

Inheritance diagram for WebClient:



Static Public Member Functions

- static void send ()

5.22.1 Detailed Description

Project ChatBot

5.22.2 Member Function Documentation

5.22.2.1 send()

```
void WebClient::send ( )  [static]
```

Project ChatBot WebClient implementation

The documentation for this class was generated from the following files:

- src/Client/Web/WebClient.h
- src/Client/Web/WebClient.cpp

Chapter 6

File Documentation

6.1 src/Agent/KnowledgeBase/Action.h File Reference

```
#include "Sentence.h"
#include "KB.h"
```

Classes

- class Action

  Represents an action undertaken by the knowledge-based agent.

6.1.1 Detailed Description

Author

Ergi, Rand, Yuge

6.2 src/Agent/KnowledgeBase/Analyser.h File Reference

```
#include "../Parser.h"
```

Classes

- class Analyser

  The analyser is responsible for the different parsing jobs that are used by the agent.

6.2.1 Detailed Description

Author

Ergi, Rand, Yuge

6.3 src/Agent/KnowledgeBase/FLAnalyser.h File Reference

```
#include <vector>
#include "Analyser.h"
#include "Sentence.h"
```

Classes

- class FLAnalyser

  This class uses the parse tree generated by the parser and translates it into a formal language (a small set of the English language in this case)

6.3.1 Detailed Description

Author

Ergi, Rand, Yuge

6.4 src/Agent/KnowledgeBase/KB.h File Reference

```
#include <vector>
#include "Sentence.h"
#include "KBAnalyser.h"
#include "Rule.h"
```

Classes

- class KB

  This class represents our knowledge base.

6.4.1 Detailed Description

Author

Ergi, Rand , Yuge

6.5 src/Agent/KnowledgeBase/KBAgent.h File Reference

```
#include "../Agent.h"
#include "Percept.h"
#include "Action.h"
#include "KB.h"
```

6.5.1 Detailed Description

Author

Ergi, Rand, Yuge

6.6 src/Agent/KnowledgeBase/KBAnalyser.h File Reference

```
#include "Analyser.h"
#include "../Parser.h"
```

Classes

- class KBAnalyser

  This class represents the analyser responsible for parsing the First Order Logic language.

6.6.1 Detailed Description

Author

Ergi, Rand, Yuge

6.7 src/Agent/KnowledgeBase/Percept.h File Reference

```
#include "Sentence.h"
#include <vector>
#include <iostream>
```

Classes

- class Percept

  This class represents a perception received by the agent (which is of an exclusive text form in our case).

6.7.1 Detailed Description

Author

Ergi, Rand, yuge

6.8 src/Agent/KnowledgeBase/Rule.h File Reference

```
#include "Sentence.h"
#include <vector>
#include <iostream>
```

6.8.1 Detailed Description

Author

Ergi, Rand, Yuge

6.9 src/Agent/KnowledgeBase/Sentence.h File Reference

Classes

- class Sentence

  Class representing a sentence.

6.9.1 Detailed Description

Author

Ergi, Rand, Yuge

Index