Department of Computer Science
Technical University of Cluj-Napoca

Intelligent Systems

Laboratory activity 2018-2019

Project title: Sentence Generating as Planning
Tool: Fast-Downward

Name:Macocian Adrian Radu
Group:30433
Email:rmacocian@yahoo.com

Contents

Chapter 1

Rules and policies

Lab organisation.

1. Laboratory work is 20% from the final grade.
2. There are 4 deliverables in total (see Table 1.1).
3. The scheduling of your work is listed in Table 1.
4. Before each deadline, you have to load your work (latex documentation/code) on moodle.cs.utcluj.ro. Work sent by email will not be graded!
5. Realistic and original scenarios are encouraged. Well known toy problems (salesmen, map colouring, logistic planning, wumpus, sudoku, queens, missionaries and cannibals, various logic puzzle, etc.) do not worth much for your grade. Your scenario should be realistic and should be business oriented. That means that you should imagine a real client asking you to perform some investigation in the AI domain. Analysing a realistic AI task is complex and can be very demanding but all students who put in the time and effort got there eventually. Note that the focus is both on programming and on modelling the reality into a formal representation. You should be aware that computing interacts with many different domains. Solutions to many AI problems require both computing skills and domain knowledge.
6. *Laptop policy*: you can use your own laptop as long you have Linux. One goal of the laboratory is to increase your competency in Linux. It is **your** task to set static IPs:

IP:	192.168.1.[51..98]
MASK:	255.255.255.0
GATEWAY:	192.168.1.2
DNS:	192.168.1.2
PROXY	192.168.1.2:3128

Another option is tu use to

Network:	isg
Password:	inteligentaartificiala

7. *Group change policy.* Maximum number of students in a class is 14.
8. *For students repeating the class*: For validating the grade obtained in previous years, a discussion is mandatory in the first week. I usually have no problem to validate your previous grades, as long you request this in the first week. Failing to do so, leads to the grade 1 for the laboratory work in the current semester.

Table 1.1: Deliverables.

Deliverable	Description	Deadline
Proposal	Description of your proposed project	W_5
Midway report	Review of related work, details of the proposed method, and preliminary results if available	W_7.
Final report	A full academic paper, including: problem definition and motivation, background and related work, details of the proposed method, details of experiments and results, conclusion and future work, references and appendix	W_{13}
Presentation	Present your work to the colleagues, instructors	W_{14}

Table 1.2: Lab scheduling.

Activity	**Deadline**
Installing the tool.	W_2
Running and understanding examples attached to each tool.	W_3
Selecting an adequate scenario to be implemented using the tool. *Describing the specifications* of your own scenario. Describing the top level design of your scenario.	W_4
Identifying and describing the knowledge bases (data sets, articles, studies, etc.) planned to be used for realistic modelling of your scenario.	W_5
Implementing your scenario. We try to evaluate how much your solution reflects the reality. This is a quantitative criteria: we evaluate how many aspects from real world have been covered by your solution.	W_{10}
Exploiting the expressivity of the tool. We try to evaluate how many capabilities provided by the tool have been enacted within your implementation. This is a qualitative criteria: a complex realistic scenario requires more expressivity.	W_{11}
Validating the correctness/efficiency of your solution through graphs and/or experiments.	W_{12}
Comparing your solution with related work. Illustrate the advantages and weak points of your solution (description and code showing advantages/disadvantages). *Demonstrating the running scenario and Latex documentation.* You have to show some competency on writing documentation in Latex. For instance, you have to employ various latex elements: lists, citations, footnotes, verbatim, maths, code, etc.	W_{13}
Public presentation and feedback provided by the supervisor to clarify the good/bad issues related to student activity/results during the semester.	W_{14}

Grading. Assessment aims to measure your knowledge and skills needed to function in realistic AI-related tasks. Assessment is based on your written report explaining the nature of the project, findings, and recommendations. Meeting the deadlines is also important. Your report is comparable to ones you would write if you were a consultant reporting to a client.

Grade inflation makes difficult to distinguish between students. It also discourages the best students to do their best. In my quest for "optimal ranking of the students", I do not use the following heuristics:

- "He worked hard at the project". Our society do not like anymore individuals that are *trying*, but individual that *do* stuff. Such heuristic is not admissible in education, except the primary school.
- "I knew he could do much better". Such a heuristic is not admissible because it does not encourage you to spread yourself.
- 7 means that you: i) constantly worked during classes, ii) you proved competent to use the tool and its expressivity for a realistic scenario, iii) you understood theoretical concepts on which the tool rely on.
- 8, 9 mean that your code is large enough and the results proved by your experiments are significant.
- 10 means that you did very impressive work or more efficient that I expected or handled a lot of special cases for realistic scenarios.
- 5 means that you managed to develop something of your own, functional, with your own piece of code substantially different from the examples available.
- You obtain less than 5 in one of the following situations:
  1. few code written by yourself (i.e 10 formulas in First Order Logic, <10 operators in PDDL).
  2. too much similarity with the provided examples.
  3. non-seriousity (i.e. re-current late at classes, playing games, worked for other disciplines, poor/unprofessional documentation of your work, etc.)[1].
- You get 2 if you present the project but fail to submit the documentation or code before the deadline. You get 1 if you do not present your project before the deadline. You get 0 for any line of code taken from other parts that appear in section *My own code.* For information on TUCN's regulations on plagiarism do consult the active norms.

  If your grade is 0, 1, or 2, you do not satisfy the preconditions for participating to the written exam. The only possibility to increase your laboratory grade is to take another project in the next year, at the same class, and to make all the steps again.

However, don't forget that focus is on learning, not on grading. Consider this lab as an opportunity to practice your skills on real-world problems. This laboratory best serves as a test to see weather you have at the moment the potential for graduate studies. The lab also provides you with an insight from research methodology. The lab project aims to: 1) develop your critical thinking; 2) enhance your ability to work independently; 3) develop your technical writing and presentation skills.

[1]Consider non-seriosity as a immutable boolean value that is unconsciously activated in my brain when one of the above conditions occurs for the first time.

Plagiarism. Most of you consider plagiarism only a minor form of cheating. This is far from accurate. Plagiarism is passing off the work of others as your own to gain unfair advantage.

During your project presentation and documentation, I must not be left with doubts on which parts of your project are your work or not. Always identify both: 1) who you worked with and 2) where you got your part of the code or solution.

Describe clearly the starting point of your solution. List explicitly any code re-used in your project. List explicitly any code adapted from other sources. List explicitly any help (including debugging help, design discussions) provided by others (including colleagues or teaching assistant). Keep in mind that it is your own project and not the teaching assistant's project. Learning by collaborating does remain an effective method. You can use it, but don't forget to mention any kind of support. Learning by exploiting various knowledge-bases developed by your elder colleagues remain also an effective method for "learning by example". When comparing samples of good and poor assignments submitted by your colleagues in earlier years try to identify which is better and why. You can use this repository of previous assignments, but don't forget to mention any kind of inspiration source.

The assignment is designed to be individual and to pose you some difficulties (both technological and scientific) for which you should identify a working solution by the end of the semester. Each semester, a distinct AI tool is assigned to a small group of students. Your are strongly encouraged to collaborate, especially during the installation phase and example understanding phase (W_1-W_4). The quicker you get throughout these preparatory stages, the more time you have for your own project.

Class attendance. We are all grown-ups, when and whether you attend lecture is up to you[2]. Keep in mind the exam can include any topic that was covered during class, explained on the board, or which emerged from discussions among participants.

First class is the most important one. It is the class in which we clarify the rules. Future mails or requests for such clarifications are not welcome. After the first class, the necessary presumption is that these rules are common knowledge. You should be aware that "Ignorantia legis neminem excusat".

Instead, attendance to laboratory classes is mandatory. Missing lab assignments or midterm leads to minimum grade for that part. You are free to manage your laboratory classes - meaning that you can submit the project earlier or send your work earlier - as long as you meet all the constraints and deadlines. However, it is mandatory to participate at the final public presentation of your project.

Re-assessment of your work. Your project is developed based on your interaction with instructor, interaction with similar examples, interaction with scientific literature, but also on your scientific interests[3] and your own deliberation. All these processes require some time steps and scheduling. That's why, developing another project in 3-4 days of the re-examination period does not provide the skills that I am interested you to have for AI. Consequently, the project will be assessed only once, in week 13. Consider that i) project assessment, ii) midterm, iii) Kahoot points, iv) lecture quizzes, v) other assignments are activities that take place within the allocated 14 weeks of the semester. Asking for a project re-assessment is similar to asking for an extra Kahoot game only for you.

[2]However, you should be aware that when signing the study contract, you have some (moral) obligations towards the people that are paying your classes through the Ministry of Education budget. These people asked you to spend 8 hours daily for learning. By signed the university contract, you practically agree to study 8 hours/day.

[3]I am forced to assume that, by studying computer science you do have such scientific curiosity.

Chapter 2

AI projects and tools (W_1)

The teaching objectives for this week are:

1. To identify existing projects for undergraduate level in the AI domain.
2. To get awareness of the effort expected for the laboratory activity by browsing past projects at TUCN and other universities.
3. To get used with the technical instrumentation used during this semester.

At the beginning of each class, please write what do you expect to learn/achieve/understand/develop during the following two hours.

My personal objectives for this class are:

1. To Find a suitable tool that I can easily understand and work with

This laboratory follows a project-based learning methodology for learning artificial intelligence.

An ideal project should be one that demonstrates some creativity, attempts to answer an interesting research question, offers an interesting AI solution to a real scenario and validates the solution through graphs and experiments. These aims cannot be achieved without reading relevant AI articles for your scenario.

2.1 AI undergraduate projects

This section aims to provide you starting ideas on student AI projects. Browse the following resources:

1. https://www.quora.com/What-are-basic-artificial-intelligence-projects-for-beginners
2. Project proposals in AI at Roskilde University, Denmark
3. Machine learning project suggestions at School of Computer Science, Carnegie Mellon University (http://www.cs.cmu.edu/ epxing/Class/10701/project.html)
4. Challenges in artificial intelligence domain at HacckerRank
5. Repository of AI assignments at AI-repository
6. Collection of projects at MIT open courseware for the Knowledge-Based Applications Systems class MITOpenCoursware

The projects are intended to let you look in depth at some area of artificial intelligence that may only be covered briefly in class or [6]. You can see the AI lab as an opportunity for you to explore an interesting problem of your choice in the context of a real-world scenario. Hence, we encourage you to do some original research in the AI domain that is of interest to you. To give you an idea, about how your fellows work as students, take a look at the examples below:

1. A machine learning approach for identifying patterns at Eurovision musical competition.
2. Performance comparison of AI algorithms in the Wumpus world.
3. A multi-agent system for playing the board game Risk. Assess the relative strength of each player and the strategic value of each country.
4. Intersection situation awareness and normative reasoning.
5. Summarize an article written in wikipedia or other technical text.
6. Checking if a small text is entailed by a larger text.
7. Extracting arguments in a persuasive or legal text.
8. A knowledge-based computer purchasing adviser.
9. Build an ontology from a collection of documents using machine learning.
10. Legal assistant in case of divorces. Splitting marital property between spouse.
11. Advisor for installing a wind turbine.
12. Compliance checking of architectural plans for buildings.
13. Knowledge based system for automobile fault diagnosis.
14. Real estate analyzer system.
15. Fake opinion detection system.

Projects at other universities:

1. A project submitted by Greg Barish entitled "Approaches to integrating abstractions in graphplan-based planning systems" for the Artificial Intelligence Planning class at University of Southern California. Final Report
2. A project submitted by Victor L. Williamson entitled "What to do With a Patient Who Has Chest Pain?" Final report

You may also check AI books like "Programming Collective Intelligence" by Toby Segaran [7]. It contains many interesting examples.

2.2 AI-related competitions

As part of your laboratory work, we encourage you to participate to various students competitions in AI:

1. Machine learning competitions: Kaggle
2. Ontology development competition: Ontology Competition
3. Competition on computational models of argumentation: ICCMA
4. Competitive programming HacckerRank
5. Student StarCraft AI tournament 2016 SSCAIT
6. Power trading agent competition PTAC
7. Ontology alignment evaluation initiative OAEI2015

Disseminating your work. Presenting your results at student conferences increases your chances to obtain a master scholarship. One possible student conference is:

1. Computer Science Students Conference at UBB and TUCN: CSSC

2.3 Running latex

LATEX is a typesetting system suitable for producing scientific and mathematical documents. Three starting points are:

- Getting started with LaTeX (David R. Wilkins)
- LaTeX Tutorial (Jeff Clark)
- Very Brief Introduction to Latex by Radu Slavescu

Kile is one latex editor. If installed, just type `kile` in the terminal.

2.4 Linux support

Becoming familiar with the Linux requires patience. You must have the desire to try and figure things out on your own, rather than having everything done for you. Two starting references are:

- Introduction to Linux A Hands on Guide, Machtelt Garrels
- Brief Synopsis of Linux by Radu Slavescu

Table 2.1 lists basic commands.

Table 2.1: Quickstart linux commands.

Command	Meaning
pwd	display present working directory
ls	displays the files in the current working directory
cd	change the directoris
chmod +x	set a file as executable
man	read man pages of a command
ssh	connects to a secure shell

2.5 Exercises

1. Compile the `is.tex` file in order to start writing your notes. Recall that this documentation is also a *support for you*, during the design and implementation of you ideas. Make an habit in writing down your ideas from the first week in a professional manner.

2. Identify 3 Web resources with ideas on student AI projects.

3. Think at one of your hobbies. Would be possible to develop something on that line?

4. Identify a media source for AI-news. Investigate interesting ideas in that news. Do the AI-technologies behind these ideas appear in the AIMA book?

5. Identify AI journals in Science Direct and Springer Verlag. Browse some abstracts from these journals.

6. Identify an AI competition. Consider making a team of 2-3 students to participate at that competition.

7. Imagine that you are the founder of a start-up. What kind of innovative project would be feasible for you company?

8. Write a short list (3 to 5) of possible projects for this lab. Be ambitious.

9. Display network information for your workstation.

10. Connect via `ssh` to another workstation.

Solution to exercise 2

Solution to exercise 3

Solution to exercise 4

Solution to exercise 5

Solution to exercise 6

Solution to exercise 7

Solution to exercise 8

Solution to exercise 9

Solution to exercise 10

Chapter 3

Installing the tool (W_2)

The teaching objectives for this week are:

1. To install the tool on a Linux system
2. To get aware of tool plugins, extensions, and running parameters.

My personal objectives for this class are:

1. Being able to install the tool with no difficulty
2. Understanding some basics about the tool

The AI lab follows the problem-based learning model [4]. In this model, you have to:

1. identify a real world situation that has no clear (or right, or efficient) solution;
2. develop a viable solution to the identified problem;
3. get new information through self-directing learning;
4. take the decisions needed during the lifecycle of your project

The first sub-problem that you are facing is to install the tool on a Linux distribution.

First, you need to find the version of your operating system. To find the kernel version, type in a terminal `uname -a`. To find the Linux distribution type `lsb_release -a`.

Then you need to download the distribution of the tool adequate for your system.

Extract the archive in your own directory. Avoid using spaces for the directory name. Identify a README file or Installation instructions for the tool.

When browsing the requirements, check first whether they have been not already installed. For instance, Java might be already in the system. You can check this with `java -version`.

List here the steps done for installing the tool. A typical installation could look like:

```
./configure
make
make install
export PATH=PATH:......
```

List here the exact commands for running the tool. These lines will save you precious time in the next week.

Finally, you need to check the installation.

In case something goes wrong:

1. Try to install a newer/older version of the tool
2. Try to obtain the source code and compile it.
3. Install the proper version for the software requirements (i.e., correct LISP or Prolog),
4. Take a look into the installation/running scripts or in the `makefile` file.
5. Try to identify a version of the tool written in other programming language.

It is your interest not to get stuck with this installation. It is therefore important to identify a solution such that you can start shaping your project from the next laboratory.

3.1 Exercises

1. List the steps done for installing the tool.
2. List the exact commands needed to run the tool.
3. What other AI tools can use the output of your tool? Can you identify such tools on the Web? Try to assess the difficulty level and risks of integrating such tools for your project.

Installing the dependecies, getting the tool and then building it

The tool has a very huge range of commands for running

Solution to exercise 3

Chapter 4

Running and understanding examples (W_3)

The teaching objectives for this week are:

1. To run and understand the toy examples provided by the tool
2. To identify what realistic problems are adequate for your tool

My personal objectives for this class are:

1. Finally getting the tool to work
2. Understanding how PDDL works with Fast-Downward

Describe the examples that you run. What type of problems can be supported by the tool?

4.1 Exercises

1. Detail one example that you run. Which are the input and the output? Describe the structure of the code.
2. Describe the real world problems that can be solved by your tool/algorithm.

I ran the example of a robot having to move 4 balls from one room to another

The easiest solutions would be getting through a maze or programming a small robot

Chapter 5

Understanding conceptual instrumentation (W_4)

The teaching objectives for this week are:

1. To understand the algorithm(s) on which your tool relies.
2. To get used with writing algorithms in Latex

Latex provides various environments for writing algorithms, including: `algorithm`, `algorithmic`, `algorithmicx`, `algpseudocode`, `algorithm2e`.

The following example algorithm has been taken from [3].

Algorithm 1: Required steps for AHP-based ontology evaluation.

Input: $\mathcal{O}$ - set of candidate ontologies; $\mathcal{W}$ - set of keywords of the domain; $\mathcal{T}$ - the criteria tree

Output: $\langle ontology, [evaluation\ values] \rangle$, association list with ontologies as keys and evaluation values as pairs

$\mathcal{W}_{\mathcal{D}} \leftarrow Update(\mathcal{W}, Wordnet)$
foreach $o \in \mathcal{O}$ **do**
    $DomainCoverage(o, \mathcal{W}_D)$
Define a *domain coverage threshold* $\delta \geq 0$
$\mathcal{O}_\delta \leftarrow Select(\mathcal{O}, \delta)$
foreach $o \in \mathcal{O}_\delta$ **do**
    foreach *non-leaf criterion* $k \in \mathcal{T}$ **do**
        complete the PC matrix to determine the weights of its sub-criteria
    foreach *leaf (atomic) criterion* $k \in \mathcal{T}$ **do**
        $OntologyMetrics(k, o)$
    Normalize ontology measurements to obtain weights for alternatives
    $WeightedSum(o, PC)$

5.1 Exercises

1. Big O complexity
2. Which are the latex options to write algorithms? Describe in one paragraph the main features of one such package for algorithms.

Solution to exercise 1

Solution to exercise 2

Chapter 6

Project description (W_5)

The teaching objectives for this week are:

1. To have a clear description of what you intend to develop.
2. To point to specific resources (datasets, knowledge bases, external tools) that support the development of your idea and which minimise the risk of failure.
3. To identify related work (articles) that are relevant or similar to your approach.

My personal objectives for this class are:

1. Find some useful resources for my project
2. Have a clearer view on how i want to implement the project

To encourage the development of AI skills, you were required to come up with a significant semester project You have to apply ideas from the course to a problem of your own.

Which domain to choose is a decision that only you can make. The more aware of the tool capabilities, the more adequate the decision. Realistic and original scenarios are encouraged. Well known toy problems (salesmen, map coloring, logistic planning, wumpus, sudoku, queens, missionaries and canibals, etc.) do not worth much for your grade. Your scenario should be realistic and should be business oriented.

Select clearly defined problems and not generic ones (i.e I will do something in the medical domain). Note that the focus is both on programming and on modelling the reality into a formal representation.

Before specifying your project you must understand as much as possible about the application domain (medicine, bank, human resource management, etc). You must also understand functionality required by the stakeholders of your system. Let the problem drive the modelling - the more you understand the domain, the more technical solutions need to be solved by you.

Consider answering to the following questions:

1. What will your system do?
2. Which is the scope of coverage your system aims for?
3. What will be the input of your program?
4. What will be the output of your program?
5. What will be the knowledge of your system?

6. Which would be the narrative description of running scenario(s)?

7. Which are the stakeholders of your system?

8. Which are the assumptions?

Example 1 (What will system do)

Example 2 (Scope of the program)

In this problem-based model you learn what you need to know in order to solve a problem. However, note that you have total flexibility in stating your project objectives. For instance, if you consider that studying more than one computational technology (or AI algorithms) brings more benefits, you are encourage to do it. You just have to frame your task under one scenario umbrella. One example: you can investigate the problem of *fake review detection* with various machine learning algorithms: decision trees, naive bayes, neural networks, ensemble learning. This road helps you to study and compare the above algorithms on your own problem. A second example: you can help a *robot to escape from a maze* with various search algorithms (A*, greedy, deep first, uniform cost) or computation technologies (constraint satisfaction problems, planning, searching with observations). Hence, both the problem-based model or a more algorithmic approach to AI (if formulated under the umbrella of a single scenario) are accepted for this laboratory.

6.1 Narrative description

The porpose of this project is designing a planner capable of generating natural language sentances. Natural language generation is the computation of given data into actual natural language. To do this i will use a TAG grammar trees as the domain of the problem which will then be used to generate sentances. I find this application usefull in generation of closed captions or creating computer generated messages for various events are similar to a spoken language.

6.2 Facts

Every language is different and it's almost impossible to express all the gramatical rules of a language in code. The main resource of this application will be the TAG language domain which will try to represent as many gramatical rules as possible. It is also impossible to represent all the words of a language in a domain so for the purpose of this application I will try to use just the necessary ones. The problem arrises in setting the goal state of the file.

6.3 Specifications

For the project I will be using Fast-Downward. The domain definition will be done in PDDL and working the Fast-Downward software and changing it for this application will be done in Python

6.4 Knowledge acquisition

The data is and will be gathered from the Fast-Downward website for developing and using the software, For general information on domain and planning i will use JAIR.org and most importantly i will try to learn from the study of Alexander Koller and Jorg Hoffmann "Waking Up a Sleeping Rabbit: On Natural-Language Sentence Generation with FF"

How do represent knowledge? Your system relies on a knowledge base. You have to describe how do you represent this knowledge. You might choose between different logics: propositional logic, first order logic, modal logics, description logics, epistemic logics, temporal logics, and so on.

Where are you getting the required knowledge/data Point towards the knowledge bases that you plan to exploit. The existence of these sources are required to prove that your approach is realistic.

Examples of knowledge sources include:

- Data sets: i.e., https://archive.ics.uci.edu/ml/datasets.html
- Statistics: i.e., http://ec.europa.eu/eurostat
- Ontology repositories in OWL or RDF format.

If you will be using books, give their reference. If you hope to exploit people for elicitation, give their names. If you aim to use data sources or knowledge repositories list them and be sure that you have access to the needed knowledge. Indicate what you have accomplished so far in knowledge acquisition.

6.5 Exercises

1. Sum up what is the aim of your project in a Twitter-sized phrase (140 characters)
2. Recall or identify an engineering methodology to write specifications. Cite this methodology and employ it for specifying your project.
3. Which are the differences between requirements and specifications?
4. Identify similar scenarios proposed by your colleagues. Think at some form of collaboration with one of your colleagues having similar interests.

The aim of the project is generating natural sentences from a set of given origin words

Solution to exercise 2

The requirements are the what is the expected ou and specifications are the real

One of my colleagues is writing a story generator so my sentence generator should come really in handy

Chapter 7

Preliminary results (W_7)

This section corresponds to the midway report in week 7. The teaching objectives for this week are:

1. To prove that you have managed to write few lines of code of your own.
2. To prove that the knowledge or data required are already obtained.

These objectives decreases the risk to fail. You should be aware that failing to meet the above objectives in week 7 indicates high risks in obtaining relevant results at the end of the semester. Take urgent measures to overcome these difficulties.

7.1 Exercises

1. Write the preliminary results explaining any realizations or insights found during the research of the subject.
2. Discuss new information and questions found during the domain investigation or during coding.

I studied a lot of research articles, but all of them rely on TAG-grammar to generate sentences but that's hard to implement and it's not ideal if we need those sentences to be randomly generated and still have some meaning. So after struggling with TAG-grammar I came to the conclusion to completely start over and write

Finding a suitable way of laying out the sentence so that it can be completely generated by a planner and some way of setting up the initial state and goal of the problem specification

Chapter 8

Implementation details (W_9)

The teaching objectives for this week are:

1. Illustrate each aspect of the reality that you have modelled in your solution.
2. To explain the relevant code from your scenario.

My personal objectives for this class are:

1. Making my code and solution for generating acceptable
2.

Projects in artificial intelligence consist of developing new solutions.

8.1 Relevant code

Provide the relevant code (see an example in Fig. 8.1). You can use "verbatim" package or "listing" package. Complement the code with its corresponding textual description.

The eager student may use concepts from *literate programming*.

8.2 Common bad practice in AI undergraduate projects

The over-estimated AI programmer.

Bad practice: Excepting few genial students, you tend to overestimate your AI-programming abilities. That is, you start to write a large amount of code. (Here large might be 20 lines). When testing it, nothing run. You start to debug a line or to remove it. Your program will not run this time too. You remove or comment another line. And so on, until you have a single line of code. If you are lucky, that could run. But you lose a lot of time in this enterprise.

Solution: In the early stage of writing code, write a line of code and test it. If it works, write another line and test it. And so on. That is, you are exploiting the interactive environments provided by AI tools or languages like LISP and PROLOG. You should hold your horses and have the most possible skeptical attitude towards your code. As you get experience, your will be noticing that writing AI-declarative code is more effective than procedural one.

The eyewash bug. *Bad practice*: You spend most of your programming time to develop a GUI for your AI-system. Don't bother. I am sympathetic with Sania Twain's view on GUIs: "You don't impress me much". Such things are indeed important in computer science, but not relevant in this AI class.

```
(:action addPred
    :parameters
                    (?verb - Verb)
    :precondition
                    (not (isPredicate ?verb))
    :effect
                    (and (isPredicate ?verb)
                         (used ?verb))
)

(:action addTermination
    :parameters
                    (?subject - noun
                     ?predicate - verb
                     ?termination - termination)
    :precondition
                    (and (isPredicate ?predicate)
                         (subject ?subject)
                         (needsTermination ?subject)
                         (or (and (terminationS ?predicate)
                                  (s ?termination))
                             (and (terminationEs ?predicate)
                                  (es ?termination))))
    :effect
                    (and (hasTermination ?predicate)
                         (isTermination ?termination)
                         (used ?termination))
)
```

Figure 8.1: Modelling arguments in Racer.

The not-organised student. You are not organised, if something like this will happen to you:

- You do not find your project and yield "Someone removed my project!". Most of the time your are logged with a different user as usual. Check this with `who am i`. This is not a rhetorical question, but a Linux command.
- You are working in a different directory. Type `pwd` and `ls` to check that your executables are indeed in the current working directory. If you have been lazy to set your PATH variable, you might just forgot to type `./` for executing the command in the current directory.

The omniscient student. You are in this category if you fail to add references. Reading relevant references is mandatory to deliver a decent project.

8.3 Exercises

1. What latex packages can be used to format code?
2.

Solution to exercise 1

Solution to exercise 2

Chapter 9

Graphs and experiments (W_{11})

The objectives for this week are:

1. To describe and interpret each experiment that you have performed

My personal objectives for this class are:

1.

2.

An experiment investigates how some variables are related. Usually, experiments verify a previously formulated hypothesis. Such hypothesis may investigate how your software degrades its performance with larger inputs. You will need to run simulations to see how your implementation is affected by different inputs.

Note that running experiments mean more than testing your solution. It helps to describe and prove how did you test your implementation. Moreover, during this lab, you will often need to: 1) generate random data for your algorithms, 2) measure their performance (number of operations, execution time), 3) draw charts, 4) interpret the obtained results.

The eager student might want to take a look at literature on how to design computer experiments, such as [2]. Section 5.6 from [2] might be of particular interest for some of you. If your experiments include a stochastic parameter, you need to include a test for statistical significance. This is important to prove that your outputs are not a random effect.

You should develop a test suite that can be used to show your code works correctly under a various conditions/problems/scenarious.

9.1 Evaluation metrics

Graphs always impress teachers...

Figure 9.1: Increasing the accuracy with the number of samples.

Chapter 10

Related work and documentation (W_{12})

The teaching objectives for this week are:

1. To compare your results to related work.
2. To discuss the advantages and limitations of your solution.
3. To deliver a professional documentation of your work.

My personal objectives for this class are:

1.
2.

This chapter convinces me that you know how your work fits into the larger domain area.

10.1 Related approaches

As stated earlier in this documentation most of the the other people that tried to create sentence generators using domain planners were relying on TAG-grammars with each individual action that a TAG node can make being a predicate

10.2 Advantages and limitations of your solution

The advantage of my solution is that it can take a set of random words and just generate a natural sentence and it takes into account all possible scenarios of a simple sentence. The limitations are that it requires a lot of input in the problem specification and that it is of a pretty big dimension (almost 1000 lines of code in the domain specification)

10.3 Possible extensions of the current work

It still can be upgraded by adding new categories for nouns (food, location, etc.)

Chapter 11

Project demo and documentation (W_{13})

The teaching objectives for this week are:

1. Deliver the technical documentation of your project
2. Demonstrate your running scenario to the instructor

My personal objectives for this class are:

1.
2.

Demonstrate in 4-5 minutes your running scenario to the instructor. The demo should take place on a Linux distribution

From your final report, remove the text/examples/algorithms/rules/bibliographic references/etc - keep only your notes. If the documentation does not meet minimum standard for readability and scientific discourse, it will be classified by the furious teaching assistant as unacceptable and therefore rejected.

11.1 Exercises

1. How you would advocate your project to a possible client?
2. Write five highlights of your results (maximum 85 characters including spaces).
3.

Solution to exercise 1

Solution to exercise 2

Chapter 12

Results dissemination and feedback (W_{14})

The teaching objectives for this week are:

1. To practice public presentation.
2. To get used with the beamer template for making scientific presentations
3. To get feedback from your colleagues.

My personal objectives for this class are:

1.

2.

Learning is enhanced by constructive feedback on the strong/weak points of your performance during AI laboratory. This feedback focuses on the scientific relevance of your results and it aims to complement the feedback encapsulated in the grade. Do not take criticism personally. It is the project that is being critisised, and not your competence or intelligence.

12.0.1 Public presentation

10 slides in beamer format for a timeslot of 5 minutes presentation plus 5 minutes questions. Questions may be posed by your colleagues or the teacher.

One of the main difficulties when designed slides is how to find the right amount of technical details to be included. No technical details rise the question of "bla bla story telling". To much technical details may bore the audience and also you may fail to fit within the time assigned.

Two introductory tutorials on beamer are [5] and [1]. The beamer manual is:

Presentation template

During presentation, it is a mistake to focus on the tool that you have been used. During 5 minutes, you have to focus only on your results and to market your work. Don't forget to include technical details and graphs.

A method for disseminating your research consists of writing 3-5 highlights. Highlights consist of a set of bullet points that convey the core findings of your work. For examples, see http://www.elsevier.com/highlights.

You are now playing the role of your project advocator. Always keep in mind that you have to market your project only, and not the tool that you have relied on.

Aspect	Self-assessment
How did you manage to master the tool?	
How realistic was your scenario?	
Relevance of the running experiments	
Knowledge and skills achieved	
Capacity to market your effort and results through documentation and presentation	

Table 12.1: Self-assessment. Assess each aspect, with: enthusiastic, satisfactory, unsatisfactory, bad

12.0.2 Self-assessment

'What did you do well? Give examples'

'Where do you think the assignment is weak?'

12.0.3 Formative feedback

The last week is an opportunity for interested students to obtain a formative feedback. This is not an opportunity to negotiate your grade. In the previous week you had your chances to advocate your work.

12.0.4 Problem-based learning

One scope was to engage you in a kind of self-directed learning. The rationale is that you are more heterogeneous and you have different learning experiences and maturity levels. The focus was not on mastering AI algorithms but to apply them in practice. By practice I mean realistic scenarious. Ideally, you should have developed more awareness of the social, environmental, economic aspects of a real problem.

Appendix A

Your original code

Domain.pddl

```
    (define (domain easyEx)

(:requirements :typing :adl)

    (:types adjective - adjective
            noun
            living - noun
            human animal - living
            location - noun
            pronoun - human
            article
            definiteArticle - article
            indefiniteArticle - article
            Verb - verb
            termination - termination
    )

    (:predicates
        (subject ?x)
        (hasSubject ?x)
        (predicate ?x)
        (terminationS ?x)
        (terminationEs ?x)
        (isPredicate ?x)
        (isHuman ?x)
        (isDone ?x)
        (usedByHuman ?x)
        (usedByLiving ?x)
        (usedByNoun ?x)
        (needsTermination ?x)
        (hasTermination ?x)
        (isTermination ?x)
        (used ?x)
        (s ?x)
        (es ?x)
        (a ?x)
```

```
    (an ?x)
    (needsHuman ?x)
    (needsLocation ?x)
    (isFood ?x)
    (isLocation ?x)
    (needsFood ?x)
    (needsLiving ?x)
    (needsNoun ?x)
    (needsAdjective ?x)
    (startsWithVowel ?x)
    (plural ?x)
    (hasArticle ?x)
)

(:action addPred
    :parameters
                    (?verb - Verb)
    :precondition
                    (not (isPredicate ?verb))
    :effect
                    (and (isPredicate ?verb)
                         (used ?verb))
)


(:action addSubjHuman                                                    ;Add subject
    :parameters
                    (?subject - human
                     ?predicate - verb)
    :precondition
                    (and (isPredicate ?predicate)
                         (or (usedByLiving ?predicate)
                             (usedByNoun ?predicate)
                             (usedByHuman ?predicate))
                         (not (subject ?subject))
                         (not (used ?subject))
                         (not (hasSubject ?predicate)))
    :effect
                    (and (subject ?subject)
                         (hasSubject ?predicate)
                         (used ?subject))
)

(:action addSubjLivingIndefinite                                                ;A
    :parameters
                    (?article - indefiniteArticle
                     ?subject - living
                     ?predicate - Verb)
    :precondition
                    (and (or (and (startsWithVowel ?subject)
```

```
                                    (an ?article))
                               (and (not (startsWithVowel ?subject))
                                    (a ?article)))
                          (isPredicate ?predicate)
                          (usedByLiving ?predicate)
                          (not (isHuman ?subject))
                          (not (plural ?subject))
                          (not (subject ?subject))
                          (not (used ?subject))
                          (not (hasSubject ?predicate)))
    :effect
                     (and (subject ?subject)
                          (hasSubject ?predicate)
                          (used ?subject)
                          (used ?article)
                          (hasArticle ?subject))
)

(:action addSubjLivingDefinite                                                    ;Add
    :parameters
                     (?article - definiteArticle
                      ?subject - living
                      ?predicate - Verb)
    :precondition
                     (and (isPredicate ?predicate)
                          (not (subject ?subject))
                          (not (isHuman ?subject))
                          (usedByLiving ?predicate)
                          (not (plural ?subject))
                          (not (used ?subject))
                          (not (hasSubject ?predicate)))
    :effect
                     (and (subject ?subject)
                          (hasSubject ?predicate)
                          (used ?subject)
                          (used ?article)
                          (hasArticle ?subject))
)

(:action addSubjLivingPlural                                                    ;Add s
    :parameters
                     (?subject - living
                      ?predicate - Verb)
    :precondition
                     (and (isPredicate ?predicate)
                          (not (subject ?subject))
                          (plural ?subject)
                          (not (used ?subject))
                          (usedByLiving ?predicate)
                          (not (hasSubject ?predicate)))
```

```
    :effect
                    (and (subject ?subject)
                         (hasSubject ?predicate)
                         (used ?subject))
)

(:action addSubjObjectIndefinite                                                ;A
    :parameters
                    (?article - indefiniteArticle
                     ?subject - noun
                     ?predicate - Verb)
    :precondition
                    (and (or (and (startsWithVowel ?subject)
                                  (an ?article))
                             (and (not (startsWithVowel ?subject))
                                  (a ?article)))
                         (isPredicate ?predicate)
                         (usedByNoun ?predicate)
                         (not (isHuman ?subject))
                         (not (plural ?subject))
                         (not (hasSubject ?predicate)))
    :effect
                    (and (subject ?subject)
                         (hasSubject ?predicate)
                         (used ?subject)
                         (used ?article)
                         (hasArticle ?subject))
)

(:action addSubjObjectDefinite                                                  ;Add
    :parameters
                    (?article - definiteArticle
                     ?subject - noun
                     ?predicate - Verb)
    :precondition
                    (and (isPredicate ?predicate)
                         (not (plural ?subject))
                         (not (isHuman ?subject))
                         (usedByNoun ?predicate)
                         (not (used ?subject))
                         (not (hasSubject ?predicate)))
    :effect
                    (and (subject ?subject)
                         (hasSubject ?predicate)
                         (used ?subject)
                         (used ?article)
                         (hasArticle ?subject))
)

(:action addSubjObjectPlural                                                    ;Add s
```

```
    :parameters
                    (?subject - noun
                     ?predicate - Verb)
    :precondition
                    (and (isPredicate ?predicate)
                         (plural ?subject)
                         (not (used ?subject))
                         (usedByNoun ?predicate)
                         (not (hasSubject ?predicate)))
    :effect
                    (and (subject ?subject)
                         (hasSubject ?predicate)
                         (used ?subject))
)

(:action addTermination
    :parameters
                    (?subject - noun
                     ?predicate - verb
                     ?termination - termination)
    :precondition
                    (and (isPredicate ?predicate)
                         (subject ?subject)
                         (needsTermination ?subject)
                         (or (and (terminationS ?predicate)
                                  (s ?termination))
                             (and (terminationEs ?predicate)
                                  (es ?termination))))
    :effect
                    (and (hasTermination ?predicate)
                         (isTermination ?termination)
                         (used ?termination))
)

;------------------Add Object------------------

(:action addObjectPlural
    :parameters
                    (?subject - noun
                     ?predicate - Verb
                     ?object - noun)
    :precondition
                    (and (isPredicate ?predicate)
                         (subject ?subject)
                         (plural ?object)
                         (not (needsTermination ?subject))
                         (not (hasArticle ?subject))
                         (not (plural ?object))
                         (not (isHuman ?object))
                         (not (used ?object))
```

```
                        (or (and (needsLocation ?predicate) (isLocation ?object))
                            (and (needsFood ?predicate) (isFood ?object))
                            (needsNoun ?predicate)))
    :effect
                    (and (used ?object)
                        (isDone ?predicate))
)

(:action addObjectWithDefiniteArticle
    :parameters
                    (?subject - noun
                     ?predicate - Verb
                     ?article - definiteArticle
                     ?object - noun)
    :precondition
                    (and (isPredicate ?predicate)
                        (subject ?subject)
                        (not (needsTermination ?subject))
                        (or (and (needsLocation ?predicate) (isLocation ?object))
                            (and (needsFood ?predicate) (isFood ?object))
                            (needsNoun ?predicate))
                        (not (plural ?object))
                        (not (isHuman ?object))
                        (not (used ?object)))
    :effect
                    (and (used ?object)
                        (isDone ?predicate))
)

(:action addObjectWithIndefiniteArticle
    :parameters
                    (?subject - noun
                     ?predicate - Verb
                     ?article - indefiniteArticle
                     ?object - noun)
    :precondition
                    (and (isPredicate ?predicate)
                        (subject ?subject)
                        (or (and (needsLocation ?predicate) (isLocation ?object))
                            (and (needsFood ?predicate) (isFood ?object))
                            (needsNoun ?predicate))
                        (not (needsTermination ?subject))
                        (not (plural ?object))
                        (not (isHuman ?object))
                        (or (and (startsWithVowel ?object)
                                 (an ?article))
                            (and (not (startsWithVowel ?object))
                                 (a ?article)))
                        (not (used ?object)))
    :effect
```

```
                    (and (used ?object)
                         (isDone ?predicate))
)

(:action addObjectWithSubjArticlePlural                                    ;Add object
    :parameters
                    (?article - article
                     ?subject - noun
                     ?predicate - Verb
                     ?object - noun)
    :precondition
                    (and (isPredicate ?predicate)
                         (subject ?subject)
                         (plural ?object)
                         (not (needsTermination ?subject))
                         (hasArticle ?subject)
                         (used ?article)
                         (or (and (needsLocation ?predicate) (isLocation ?object))
                             (and (needsFood ?predicate) (isFood ?object))
                             (needsNoun ?predicate))
                         (not (used ?object)))
    :effect
                    (and (used ?object)
                         (isDone ?predicate))
)

(:action addObjectWithSubjArticleWithDefiniteArticle                          ;Add obj
    :parameters
                    (?article1 - article
                     ?subject - noun
                     ?predicate - Verb
                     ?article - definiteArticle
                     ?object - noun)
    :precondition
                    (and (isPredicate ?predicate)
                         (subject ?subject)
                         (used ?article1)
                         (hasArticle ?subject)
                         (not (plural ?object))
                         (not (isHuman ?object))
                         (not (needsTermination ?subject))
                         (or (and (needsLocation ?predicate) (isLocation ?object))
                             (and (needsFood ?predicate) (isFood ?object))
                             (needsNoun ?predicate))
                         (not (used ?object)))
    :effect
                    (and (used ?object)
                         (isDone ?predicate))
)
```

```
(:action addObjectWithSubjArticleWithIndefiniteArticle                          ;Add ob
    :parameters
                    (?article1 - article
                     ?subject - noun
                     ?predicate - Verb
                     ?article - indefiniteArticle
                     ?object - noun)
    :precondition
                    (and (isPredicate ?predicate)
                         (subject ?subject)
                         (used ?article1)
                         (hasArticle ?subject)
                         (or (and (needsLocation ?predicate) (isLocation ?object))
                             (and (needsFood ?predicate) (isFood ?object))
                             (needsNoun ?predicate))
                         (not (plural ?object))
                         (not (isHuman ?object))
                         (not (needsTermination ?subject))
                         (or (and (startsWithVowel ?object)
                                  (an ?article))
                             (and (not (startsWithVowel ?object))
                                  (a ?article)))
                         (not (used ?object)))
    :effect
                    (and (used ?object)
                         (isDone ?predicate))
)

(:action addObjectTerminationPlural
    :parameters
                    (?subject - noun
                     ?predicate - verb
                     ?termination - termination
                     ?object - noun)
    :precondition
                    (and (isPredicate ?predicate)
                         (subject ?subject)
                         (needsTermination ?subject)
                         (hasTermination ?predicate)
                         (not (plural ?object))
                         (not (isHuman ?object))
                         (plural ?object)
                         (or (and (needsLocation ?predicate) (isLocation ?object))
                             (and (needsFood ?predicate) (isFood ?object))
                             (needsNoun ?predicate))
                         (isTermination ?termination)
                         (not (used ?object)))
    :effect
                    (and (used ?object)
                         (not (isTermination ?termintaion))
```

```
                        (isDone ?predicate))
)

(:action addObjectTerminationWithDefiniteArticle                                    ;A
    :parameters
                    (?subject - noun
                     ?predicate - Verb
                     ?termination - termination
                     ?article - definiteArticle
                     ?object - noun)
    :precondition
                    (and (isPredicate ?predicate)
                         (subject ?subject)
                         (or (and (needsLocation ?predicate) (isLocation ?object))
                             (and (needsFood ?predicate) (isFood ?object))
                             (needsNoun ?predicate))
                         (needsTermination ?subject)
                         (not (plural ?object))
                         (not (isHuman ?object))
                         (hasTermination ?predicate)
                         (isTermination ?termination)
                         (not (used ?object)))
    :effect
                    (and (used ?object)
                         (not (isTermination ?termintaion))
                         (isDone ?predicate))
)

(:action addObjectTerminationWithIndefiniteArticle
    :parameters
                    (?subject - noun
                     ?predicate - Verb
                     ?termination - termination
                     ?article - indefiniteArticle
                     ?object - noun)
    :precondition
                    (and (isPredicate ?predicate)
                         (subject ?subject)
                         (or (and (needsLocation ?predicate) (isLocation ?object))
                             (and (needsFood ?predicate) (isFood ?object))
                             (needsNoun ?predicate))
                         (needsTermination ?subject)
                         (hasTermination ?predicate)
                         (not (plural ?object))
                         (not (isHuman ?object))
                         (isTermination ?termination)
                         (or (and (startsWithVowel ?object)
                                  (an ?article))
                             (and (not (startsWithVowel ?object))
                                  (a ?article)))
```

```
                         (not (used ?object)))
    :effect
                    (and (used ?object)
                         (not (isTermination ?termintaion))
                         (isDone ?predicate))
)

(:action addObjectWithSubjArticlePlural
    :parameters
                    (?article - article
                     ?subject - noun
                     ?predicate - Verb
                     ?termination - termination
                     ?object - noun)
    :precondition
                    (and (isPredicate ?predicate)
                         (subject ?subject)
                         (plural ?object)
                         (hasArticle ?subject)
                         (used ?article)
                         (or (and (needsLocation ?predicate) (isLocation ?object))
                             (and (needsFood ?predicate) (isFood ?object))
                             (needsNoun ?predicate))
                         (needsTermination ?subject)
                         (hasTermination ?predicate)
                         (isTermination ?termination)
                         (not (used ?object)))
    :effect
                    (and (used ?object)
                         (not (isTermination ?termintaion))
                         (isDone ?predicate))
)

(:action addObjectWithSubjArticleWithDefiniteArticle
    :parameters
                    (?article1 - article
                     ?subject - noun
                     ?predicate - Verb
                     ?termination - termination
                     ?article - definiteArticle
                     ?object - noun)
    :precondition
                    (and (isPredicate ?predicate)
                         (subject ?subject)
                         (used ?article1)
                         (hasArticle ?subject)
                         (not (plural ?object))
                         (not (isHuman ?object))
                         (needsTermination ?subject)
                         (hasTermination ?predicate)
```

```
                        (isTermination ?termination)
                        (or (and (needsLocation ?predicate) (isLocation ?object))
                            (and (needsFood ?predicate) (isFood ?object))
                            (needsNoun ?predicate))
                        (not (used ?object)))
    :effect
                    (and (used ?object)
                        (not (isTermination ?termintaion))
                        (isDone ?predicate))
)

(:action addObjectWithSubjArticleWithIndefiniteArticle
    :parameters
                    (?article1 - article
                     ?subject - noun
                     ?predicate - Verb
                     ?termination - termination
                     ?article - indefiniteArticle
                     ?object - noun)
    :precondition
                    (and (isPredicate ?predicate)
                        (subject ?subject)
                        (used ?article1)
                        (hasArticle ?subject)
                        (or (and (needsLocation ?predicate) (isLocation ?object))
                            (and (needsFood ?predicate) (isFood ?object))
                            (needsNoun ?predicate))
                        (not (plural ?object))
                        (not (isHuman ?object))
                        (needsTermination ?subject)
                        (hasTermination ?predicate)
                        (isTermination ?termination)
                        (or (and (startsWithVowel ?object)
                                 (an ?article))
                            (and (not (startsWithVowel ?object))
                                 (a ?article)))
                        (not (used ?object)))
    :effect
                    (and (used ?object)
                        (not (isTermination ?termintaion))
                        (isDone ?predicate))
)

;------------------Add Object Adjective-----------------

(:action addObjectAdjective
    :parameters
                    (?subject - noun
                     ?predicate - Verb
                     ?object - adjective)
```

```
    :precondition
                    (and (isPredicate ?predicate)
                         (subject ?subject)
                         (not (needsTermination ?subject))
                         (not (hasArticle ?subject))
                         (needsAdjective ?predicate)
                         (not (used ?object)))
    :effect
                    (and (used ?object)
                         (isDone ?predicate))
)

(:action addObjectAdjectiveWithSubjArticle                                    ;Add objec
    :parameters
                    (?article - article
                     ?subject - noun
                     ?predicate - Verb
                     ?object - adjective)
    :precondition
                    (and (isPredicate ?predicate)
                         (subject ?subject)
                         (hasSubject ?predicate)
                         (hasArticle ?subject)
                         (not (needsTermination ?subject))
                         (used ?article)
                         (needsAdjective ?predicate)
                         (not (used ?object)))
    :effect
                    (and (used ?object)
                         (isDone ?predicate))
)

(:action addObjectAdjectiveTermination
    :parameters
                    (?subject - noun
                     ?predicate - verb
                     ?termination - termination
                     ?object - adjective)
    :precondition
                    (and (isPredicate ?predicate)
                         (subject ?subject)
                         (needsTermination ?subject)
                         (hasTermination ?predicate)
                         (needsAdjective ?predicate)
                         (isTermination ?termination)
                         (not (used ?object)))
    :effect
                    (and (used ?object)
                         (not (isTermination ?termintaion))
                         (isDone ?predicate))
```

```
)

(:action addObjectadjectiveTerminationWithSubjArticle
    :parameters
                    (?article - article
                     ?subject - noun
                     ?predicate - Verb
                     ?termination - termination
                     ?object - adjective)
    :precondition
                    (and (isPredicate ?predicate)
                         (subject ?subject)
                         (hasArticle ?subject)
                         (used ?article)
                         (needsAdjective ?predicate)
                         (needsTermination ?subject)
                         (hasTermination ?predicate)
                         (isTermination ?termination)
                         (not (used ?object)))
    :effect
                    (and (used ?object)
                         (not (isTermination ?termintaion))
                         (isDone ?predicate))
)

;------------------Add Object Human------------------

(:action addObjectHuman
    :parameters
                    (?subject - noun
                     ?predicate - Verb
                     ?object - human)
    :precondition
                    (and (isPredicate ?predicate)
                         (subject ?subject)
                         (not (hasArticle ?subject))
                         (not (needsTermination ?subject))
                         (needsHuman  ?predicate)
                         (not (used ?object)))
    :effect
                    (and (used ?object)
                         (isDone ?predicate))
)

(:action addObjectHumanWithSubjArticle                                  ;Add object hu
    :parameters
                    (?article - article
                     ?subject - noun
                     ?predicate - Verb
                     ?object - human)
```

```
    :precondition
                    (and (isPredicate ?predicate)
                         (subject ?subject)
                         (hasArticle ?subject)
                         (not (needsTermination ?subject))
                         (used ?article)
                         (needsHuman  ?predicate)
                         (not (used ?object)))
    :effect
                    (and (used ?object)
                         (isDone ?predicate))
)

(:action addObjectHumanTermination
    :parameters
                    (?subject - noun
                     ?predicate - verb
                     ?termination - termination
                     ?object - human)
    :precondition
                    (and (isPredicate ?predicate)
                         (subject ?subject)
                         (needsTermination ?subject)
                         (hasTermination ?predicate)
                         (needsHuman  ?predicate)
                         (isTermination ?termination)
                         (not (used ?object)))
    :effect
                    (and (used ?object)
                         (not (isTermination ?termintaion))
                         (isDone ?predicate))
)

(:action addObjectHumanTerminationWithSubjArticle
    :parameters
                    (?article - article
                     ?subject - noun
                     ?predicate - Verb
                     ?termination - termination
                     ?object - human)
    :precondition
                    (and (isPredicate ?predicate)
                         (subject ?subject)
                         (hasArticle ?subject)
                         (used ?article)
                         (needsHuman  ?predicate)
                         (needsTermination ?subject)
                         (hasTermination ?predicate)
                         (isTermination ?termination)
                         (not (used ?object)))
```

```
    :effect
                    (and (used ?object)
                         (not (isTermination ?termintaion))
                         (isDone ?predicate))
)

;------------------Add Object Living-----------------

(:action addObjectLivingPlural
    :parameters
                    (?subject - noun
                     ?predicate - Verb
                     ?object - living)
    :precondition
                    (and (isPredicate ?predicate)
                         (subject ?subject)
                         (plural ?object)
                         (not (isHuman ?object))
                         (not (needsTermination ?subject))
                         (not (hasArticle ?subject))
                         (needsLiving ?predicate)
                         (not (used ?object)))
    :effect
                    (and (used ?object)
                         (isDone ?predicate))
)

(:action addObjectLivingWithDefiniteArticle
    :parameters
                    (?subject - noun
                     ?predicate - Verb
                     ?article - definiteArticle
                     ?object - living)
    :precondition
                    (and (isPredicate ?predicate)
                         (subject ?subject)
                         (not (needsTermination ?subject))
                         (not (plural ?object))
                         (not (isHuman ?object))
                         (needsLiving ?predicate)
                         (not (used ?object)))
    :effect
                    (and (used ?object)
                         (isDone ?predicate))
)

(:action addObjectLivingWithIndefiniteArticle
    :parameters
                    (?subject - noun
                     ?predicate - Verb
```

```
                    ?article - indefiniteArticle
                    ?object - living)
    :precondition
                   (and (isPredicate ?predicate)
                        (subject ?subject)
                        (needsLiving ?predicate)
                        (not (plural ?object))
                        (not (isHuman ?object))
                        (not (needsTermination ?subject))
                        (or (and (startsWithVowel ?object)
                                 (an ?article))
                            (and (not (startsWithVowel ?object))
                                 (a ?article)))
                        (not (used ?object)))
    :effect
                   (and (used ?object)
                        (isDone ?predicate))
)

(:action addObjectLivingWithSubjArticlePlural                                    ;Add o
    :parameters
                   (?article - article
                    ?subject - noun
                    ?predicate - Verb
                    ?object - living)
    :precondition
                   (and (isPredicate ?predicate)
                        (subject ?subject)
                        (plural ?object)
                        (hasArticle ?subject)
                        (not (plural ?object))
                        (not (isHuman ?object))
                        (not (needsTermination ?subject))
                        (used ?article)
                        (needsLiving ?predicate)
                        (not (used ?object)))
    :effect
                   (and (used ?object)
                        (isDone ?predicate))
)

(:action addObjectLivingWithSubjArticleWithDefiniteArticle                       ;A
    :parameters
                   (?article1 - article
                    ?subject - noun
                    ?predicate - Verb
                    ?article - definiteArticle
                    ?object - living)
    :precondition
                   (and (isPredicate ?predicate)
```

```
                         (subject ?subject)
                         (used ?article1)
                         (not (plural ?object))
                         (not (isHuman ?object))
                         (not (needsTermination ?subject))
                         (hasArticle ?subject)
                         (needsLiving ?predicate)
                         (not (used ?object)))
    :effect
                    (and (used ?object)
                         (isDone ?predicate))
)

(:action addObjectLivingWithSubjArticleWithIndefiniteArticle                    ;
    :parameters
                    (?article1 - article
                     ?subject - noun
                     ?predicate - Verb
                     ?article - indefiniteArticle
                     ?object - living)
    :precondition
                    (and (isPredicate ?predicate)
                         (subject ?subject)
                         (used ?article1)
                         (not (needsTermination ?subject))
                         (hasArticle ?subject)
                         (needsLiving ?predicate)
                         (not (plural ?object))
                         (not (isHuman ?object))
                         (or (and (startsWithVowel ?object)
                                  (an ?article))
                             (and (not (startsWithVowel ?object))
                                  (a ?article)))
                         (not (used ?object)))
    :effect
                    (and (used ?object)
                         (isDone ?predicate))
)

(:action addObjectLivingTerminationPlural
    :parameters
                    (?subject - noun
                     ?predicate - verb
                     ?termination - termination
                     ?object - living)
    :precondition
                    (and (isPredicate ?predicate)
                         (subject ?subject)
                         (needsTermination ?subject)
                         (hasTermination ?predicate)
```

```
                         (plural ?object)
                         (not (isHuman ?object))
                         (needsLiving ?predicate)
                         (isTermination ?termination)
                         (not (used ?object)))
    :effect
                    (and (used ?object)
                         (not (isTermination ?termintaion))
                         (isDone ?predicate))
)

(:action addObjectLivingTerminationWithDefiniteArticle
    :parameters
                    (?subject - noun
                     ?predicate - Verb
                     ?termination - termination
                     ?article - definiteArticle
                     ?object - living)
    :precondition
                    (and (isPredicate ?predicate)
                         (subject ?subject)
                         (needsLiving ?predicate)
                         (needsTermination ?subject)
                         (hasTermination ?predicate)
                         (not (plural ?object))
                         (not (isHuman ?object))
                         (isTermination ?termination)
                         (not (used ?object)))
    :effect
                    (and (used ?object)
                         (not (isTermination ?termintaion))
                         (isDone ?predicate))
)

(:action addObjectLivingTerminationWithIndefiniteArticle
    :parameters
                    (?subject - noun
                     ?predicate - Verb
                     ?termination - termination
                     ?article - indefiniteArticle
                     ?object - living)
    :precondition
                    (and (isPredicate ?predicate)
                         (subject ?subject)
                         (needsLiving ?predicate)
                         (needsTermination ?subject)
                         (hasTermination ?predicate)
                         (not (plural ?object))
                         (not (isHuman ?object))
                         (isTermination ?termination)
```

```
                         (or (and (startsWithVowel ?object)
                                  (an ?article))
                             (and (not (startsWithVowel ?object))
                                  (a ?article)))
                         (not (used ?object)))
    :effect
                    (and (used ?object)
                         (not (isTermination ?termintaion))
                         (isDone ?predicate))
)

(:action addObjectLivingTerminationWithSubjArticlePlural
    :parameters
                    (?article - article
                     ?subject - noun
                     ?predicate - Verb
                     ?termination - termination
                     ?object - living)
    :precondition
                    (and (isPredicate ?predicate)
                         (subject ?subject)
                         (plural ?object)
                         (hasArticle ?subject)
                         (used ?article)
                         (needsLiving ?predicate)
                         (needsTermination ?subject)
                         (not (plural ?object))
                         (not (isHuman ?object))
                         (hasTermination ?predicate)
                         (isTermination ?termination)
                         (not (used ?object)))
    :effect
                    (and (used ?object)
                         (not (isTermination ?termintaion))
                         (isDone ?predicate))
)

(:action addObjectLivingTerminationWithSubjArticleWithDefiniteArticle
    :parameters
                    (?article1 - article
                     ?subject - noun
                     ?predicate - Verb
                     ?termination - termination
                     ?article - definiteArticle
                     ?object - living)
    :precondition
                    (and (isPredicate ?predicate)
                         (subject ?subject)
                         (used ?article1)
                         (hasArticle ?subject)
```

```
                              (needsTermination ?subject)
                              (hasTermination ?predicate)
                              (not (plural ?object))
                              (not (isHuman ?object))
                              (isTermination ?termination)
                              (needsLiving ?predicate)
                              (not (used ?object)))
        :effect
                         (and (used ?object)
                              (not (isTermination ?termintaion))
                              (isDone ?predicate))
    )

    (:action addObjectLivingTerminationWithSubjArticleWithIndefiniteArticle
        :parameters
                         (?article1 - article
                          ?subject - noun
                          ?predicate - Verb
                          ?termination - termination
                          ?article - indefiniteArticle
                          ?object - living)
        :precondition
                         (and (isPredicate ?predicate)
                              (subject ?subject)
                              (used ?article1)
                              (hasArticle ?subject)
                              (needsLiving ?predicate)
                              (needsTermination ?subject)
                              (not (plural ?object))
                              (not (isHuman ?object))
                              (hasTermination ?predicate)
                              (isTermination ?termination)
                              (or (and (startsWithVowel ?object)
                                       (an ?article))
                                  (and (not (startsWithVowel ?object))
                                       (a ?article)))
                              (not (used ?object)))
        :effect
                         (and (used ?object)
                              (not (isTermination ?termintaion))
                              (isDone ?predicate))
    )
)
```

Example problem.pddl

```
    (define (problem Ana-are)
    (:domain easyEx)
    (:objects
                He - pronoun
                She - pronoun
```

```
            Ana - human
            George - human
            clock - noun
            enjoy - verb
            eat - verb
            like - verb
            apple - noun
            see - verb
            s - termination
            es - termination
            this - definiteArticle
            the - definiteArticle
            a - indefiniteArticle
            an - indefiniteArticle
            tasty - adjective
            highschool - noun
            dog - living
)
(:init  (needsTermination Ana)
        (needsTermination George)
        (needsTermination He)
        (needsTermination She)
        (needsTermination apple)
        (needsTermination clock)
        (needsTermination highschool)
        (needsTermination dog)
        (s s)
        (es es)
        (a a)
        (an an)
        (isHuman George)
        (usedByLiving enjoy)
        (usedByLiving eat)
        (usedByLiving see)
        (isHuman Ana)
        (isHuman He)
        (isHuman She)
        (isLocation highschool)
        (isFood apple)
        (startsWithVowel apple)
        (terminationS like)
        (terminationS eat)
        (terminationS see)
        (needsFood eat)
        (needsLocation enjoy)
        (needsNoun like)
        (needsNoun see)
        (terminationEs enjoy)
)
(:goal (and (isDone eat) (isDone enjoy) (isDone see) (isDone like)))
```

)

Appendix B

Quick technical guide for running your project

To run the generator you need to specifiy all the words that the tool should use and specify what the words are (isLocation, isFood, isHuman, etc) as well as what nouns can use what verbs. Then the user just needs to run fast downward to solve the planning problem

Appendix C

Check list

1. Your original code is included in the Appendix .
2. Your original code and figures are readable.
3. All the references are added in the Bibliography section.
4. All your figures are referred in text (with command ref), described in the text, and they have relevant caption.
5. The final documentation describes only your project. Don't forget to remove all tutorial lines in the template (like these one).
6. The main algorithm of your tool is formalised in latex in chapter 5.

Bibliography

[1] Charles T Batts. A beamer tutorial in beamer. *The University of North Carolina at Greensboro, Department of Computer Science*, 2007.

[2] Kai-Tai Fang, Runze Li, and Agus Sudjianto. *Design and modeling for computer experiments*. CRC Press, 2005.

[3] A. Groza, I. Dragoste, I. Sincai, I. Jimborean, and V. Moraru. An ontology selection and ranking system based on the analytic hierarchy process. In *Symbolic and Numeric Algorithms for Scientific Computing (SYNASC), 2014 16th International Symposium on*, pages 293–300, Sept 2014.

[4] Cindy E Hmelo-Silver. Problem-based learning: What and how do students learn? *Educational psychology review*, 16(3):235–266, 2004.

[5] Andrew Mertz and William Slough. Beamer by example. *The PracTEX Journal*, 4, 2005.

[6] Stuart Jonathan Russell, Peter Norvig, John F Canny, Jitendra M Malik, and Douglas D Edwards. *Artificial intelligence: a modern approach*, volume 2. Prentice hall Upper Saddle River, 2003.

[7] Toby Segaran. *Programming Collective Intelligence: Building Smart Web 2.0 Applications*. O'Reilly Media, 2007.