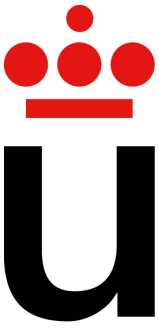

Universidad Rey Juan Carlos

ESCUELA TÉCNICA SUPERIOR DE INGENIERÍA INFORMÁTICA

GRADO EN INGENIERÍA INFORMÁTICA

Modelling and estimation of criminal sentences using Logic Programming

TRABAJO FIN DE GRADO

2022-2023

Author: **Elena Martínez Fernández**

Supervised by: Joaquín Arias Herrero
Alberto Fernández Gil

Abstract

In Spain, the legal system continues to be an area where the use of more traditional methods is prioritized over the use of new technologies. Although there are several valid reasons why this is still the case, we cannot deny the intrusion of the new artificial intelligences in environments where they had never had a place before, so it would not be unreasonable to think of a future where these two worlds are united.

In this project we have searched for a way to remotely approach that future, developing a system to obtain an estimate of the sentence as a result of the commission of a series of crimes. Thus, we make use of logic programming to model and justify the legal bases that will lead to a final minimum and maximum penalty, focusing also on obtaining a natural language reasoning that makes up the final result. Specifically, the main part of the project has been implemented thanks to the s(CASP) system. Moreover, we wanted to show another innovative way to model the law through the Blawx application. Both tools will be explained in detail throughout this paper.

The reason why we have sought to express the final result in natural language was to try to make the tool also (and mainly) available to the average person who has no knowledge in computer science; we seek to offer in the future a web tool that can help both students in the field of law and ordinary people with simple curiosity; and maybe, if the tool were to become professional, help workers in the justice system who wish to quickly obtain an approximate calculation of the penalty.

Resumen

En España, el sistema judicial sigue siendo un sector en el que se prioriza el uso de métodos más tradicionales frente al uso de las nuevas tecnologías. Aunque existen varias razones válidas por las que esto sigue siendo así, no podemos negar la introducción de nuevas inteligencias artificiales en entornos en los que nunca antes habían tenido cabida, por lo que no sería descabellado pensar en un futuro en el que estos dos mundos estén unidos.

En este trabajo hemos buscado una forma de aproximarnos remotamente a ese futuro, desarrollando un sistema para obtener una estimación de la pena como resultado de la comisión de una serie de delitos. Así, hacemos uso de la programación lógica para modelar y justificar las bases legales que conducirán a un rango con una pena final mínima y máxima, centrándonos también en la obtención de un razonamiento en lenguaje natural que componga el resultado final. En concreto, la parte principal del proyecto se ha implementado gracias al sistema s(CASP). Más allá, hemos querido mostrar otra forma novedosa para interpretar la ley a través de la aplicación de Blawx. Ambas herramientas serán explicadas en profundidad a lo largo de la memoria.

La razón por la que hemos buscado expresar el resultado final en lenguaje natural ha sido para intentar que la herramienta esté también (y principalmente) al alcance de la persona media que no tenga conocimientos en informática; buscamos ofrecer en el futuro una herramienta web que pueda ayudar tanto a estudiantes de la rama del derecho como a personas de a pie con simple curiosidad; y tal vez, si la herramienta llegara a profesionalizarse, ayudar a los trabajadores de la justicia que deseen obtener rápidamente un cálculo aproximado de la pena.

Contents

List of Figures

Chapter 1

Introduction

New technologies have burst into all areas of our lives and have had a direct impact on every aspect of our private lives as well as on the public administration, including the justice system. Today, we can already speak of a new concept born from the introduction of ICT to law: e-justice. This new concept involves the use of a wide variety of technological elements in all the main and related processes of justice delivery. The contribution of technological innovations and tools to e-justice can be analyzed on the basis of four main uses of ICTs related to the administration of justice: information, management, relationship and decision (Sacoto Romo and Cordero Moscoso 2021).

Thus, legal technology is starting to include various applications, from artificial intelligence to blockchain technology, to enhance and improve the efficiency and effectiveness of legal services (Abou El-Eid 2022). Its ultimate goal is to achieve an administration of justice in which the main operational feature is the digitization of processes from their beginning to their termination, getting rid of barriers of time and space and automating those repetitive and time-consuming processes (Sacoto Romo and Cordero Moscoso 2021).

Other works related to this area show different ways to take advantage of technological advances related to machine learning or AI. For example, to predict court decisions through tools that automatically analyze legal materials (Aletras et al. 2016) or with

tools that provide text classification methods that can help legal professional workers (Sulea et al. 2017). Following this line, we wanted to develop through this project one more tool to add to this list, in this case focusing on the modelling of criminal penalties.

1.1 Main objectives

From the very beginning, the main purpose of this project has been to merge the legal field with the current knowledge of computer science, in order to find a new perspective of the justice world by adapting the current procedures to the new technologies. For this end, several ideas were considered in order to develop some form of computer tool that could be useful in the legal field. Finally, the chosen approach was the creation of a software which, through the introduction of input values corresponding to the occurrence of some facts, would determine the perpetration, or not, of certain crimes and the resulting prison sentence.

1.2 Secondary objectives

The project aims to achieve two secondary objectives. In the first place, the aim is to develop using the programming language s(CASP) the required logical rules so that, with certain input data, it gives us as output the minimum and maximum penalty for a series of crimes.

On the other hand, we aim to use and to introduce the Blawx interface, which is based on the use of s(CASP) and other tools for the codification of legislation and contracts. Thus, the previously developed logical rules will be encoded in the Blawx interface, with the purpose of creating a project that adapts the application of Google's Blockly library in the definition of certain crimes.

1.3 Methodology

Once the main objective of the project was set and since the base of this project is the Spanish Penal Code, the first step was to do a research on this group of laws and find out our approach. At this point we came up with the idea of using Prolog, since it was an already familiar language. Based on this idea, we were introduced to the existence of s(CASP), which will be explained later on and whose functionality had to be studied in order to carry out the project. Thus, with the use of logic programming through this system, we developed a software that would provide us a list of the crimes committed by a certain subject according to some input values and the total prison penalty assigned to them. To do this, it was necessary to first specify which part of the law was going to be modeled so that it could be studied in depth and find out how to formulate this knowledge into logical rules. Once this part was finished, it was then required to establish both the way in which the penalty was going to be estimated and which legislative articles would come into play in order to study them and be able to implement these. Thus, the next step was to develop the code for the calculation of the penalty and ultimately verify the correct working of the whole system.

Next, to fulfill the project, it was decided to introduce another way of implementing the law upon finding out about the existence of Blawx, a web-based tool that will be explained in detail in the following sections. It was necessary to learn about its functioning and examine several examples provided on the website itself in order to model the chosen laws. Therefore, it was possible to enhance the project with a new, relatively unknown tool that can be very useful when coding legislative text.

Chapter 2

Background

2.1 Logic programming

Before we present the Ciao and s(CASP) languages, it is necessary to know the origin of both: the programming language known as Prolog. It is a language that is often used to implement artificial intelligence and expert systems. It is part of what is known as declarative programming, which is characterized by problem solving through the use of rules (Jones Pérez and Yong Morales 2007). This means that Prolog is based on a predicate based logic, particularly on a subset of this logic called "Horn clauses". The elements of this language can be either facts, if we are dealing with statements that we consider true in our program, or rules, if we are referring to logical implications that will necessarily have at least one antecedent and one consequent. A program of this type would be, therefore, formed by predicates, and each of them formed by a name and arguments (Cubero and Berzal 2017).

A very simple example that is often used to provide a first approach to logic is “All men are mortal. Socrates is a man. Therefore, Socrates is mortal”. In First Order Logic this would be expressed like this:

$$(((\forall x) Man(x) \rightarrow Mortal(x)) \wedge Man(socrates)) \rightarrow Mortal(socrates)$$

It is stated on the one hand that, if X is a man, then X is mortal, and, on the other hand,

that Socrates is a man, therefore it is concluded that Socrates is a mortal. In Prolog, this would be expressed in the following way.

```
1 man(socrates). %%This clause corresponds to a fact
2 mortal(X) :- man(X). %%This one corresponds to a rule
```

Thus, if the query `?- mortal(socrates)` is made, the result will be positive.

In a project that uses Prolog, the system will assume that everything that is not explicitly stated (facts or conclusions that can be proved using the rules) is false; when running a query, Prolog tries to prove the objectives (or antecedents) of a rule in the specified order. Each goal specifies a subset of executable clauses (possible solutions), called choice points; Prolog selects the first one and continues executing the program until it determines whether the goal is true or false with those variables. In case it turns out to be false, Prolog undoes everything executed using the backtracking strategy, placing the program in the same state in which it was just before reaching that choice point, and continues the execution by trying the next one. This continues until there are no more targets or choice points annotated (Soto Romero 2019).

It was important to explain the basics of how Prolog works, since Ciao is a modern implementation of Prolog that is built on a simple, logic-based kernel designed to be portable, extensible and modular. The Ciao system is described as an environment that allows the development of programs using Prolog and other derived languages. Based on this system, the Ciao language has been developed, designed to allow extensions and restrictions in a modular way and focused on enabling modular program development, manipulation and automatic optimization. Ciao also includes a large number of features, such as standalone compiler (ciaoc), interactive shell (ciaosh or ciao), an interpreter of scripts written in Prolog (ciao-shell), several libraries and many others (Bueno et al. 2004).

Lastly, we need to describe the theoretical framework of the main language used in this project; that is s(CASP), a system that acts as an interpreter for programs using constrained logic. The result is a model capable of creating variables that are kept during execution, as well as in answer sets, and also can run programs without grounding,

which, in this context, refers to the act of assigning concrete values to obtain concrete results from abstract query expressions. s(CASP) is the result of an approach based on other models such as s(ASP) and CLP, since the constraints typical of CLP are incorporated into the execution model s(ASP). The s(CASP) interpreter is reimplemented in Ciao, since this way Prolog is in charge of all the operations and can handle them natively, avoiding the need to interpret them (Arias et al. 2018). In this project we have chosen to use the s(CASP) interpreter due, firstly, to the convenience of having Prolog do all the operations that can be done in natively and, secondly, to the fact that the interpreter provides a justification tree for the answer to a query, which will be useful for the user to know why one answer or another is given and is possible due to the fact that s(CASP) supports both negation as failure (NAF) and classical negation.

2.1.1 Local installation

The rules required in this project could be generated either in the Playground of the s(CASP) interpreter (`https://ciao-lang.org/Playground/scasp.html`), a web-based application created from the Ciao Playground and running entirely in the user's browser, or they could be generated locally by performing the corresponding installation. Due to certain limitations of the Ciao Playground (and therefore also in the s(CASP) Playground), such as a slower execution than natively, and for a more complete operation that does not depend on a browser, the system has been installed natively. To do this, we had to perform the necessary changes in the host computer to meet the requirements for the installation.

In our case, due to the fact that we are working with the Windows operating system, the first requirement was to install WSL. This acronym stands for Windows Subsystem for Linux and it is a compatibility layer that allows running Linux applications on Windows without the need of a virtual machine in the traditional sense. It differs from a virtual machine in the fact that there is not a full instance of a virtualized operating system, but instead uses a combination of system call translation and process-level virtualization to provide a Linux runtime environment inside Windows. The second requirement relies on the installation of a modern C compiler, such as gcc. gcc (GNU

Compiler Collection) provides compilers for several programming languages and is used to translate the source code of a program written in a high-level programming language into machine code, which is the executable format understood by the computer. Optionally, the third and final requirement was to install Emacs, a text editor that allows you to edit and manipulate the text of the development environment efficiently.

Once the specific requirements explained in the Ciao manual for installation have been met, the only missing piece is Ubuntu. Within WSL, users can choose to install a Linux distribution, such as Ubuntu, which operates as a fully functional Linux environment within Windows. This allows the user to run Linux applications and commands in a familiar environment without having to switch completely to the Linux operating system. Thus, we already would have everything we need to install Ciao by entering certain commands in the Ubuntu terminal, and then, once Ciao is installed, enter the command `ciao get gitlab.software.imdea.org/ciao-lang/sCASP` to obtain s(CASP) and be able to start using it through the terminal.

2.2 Blawx

The second part of this project is based on the Blawx system. It is a web-based tool created by Jason Morris of Lexpedite Legal Technologies and it is founded on the idea that by writing rules in a programming language and simultaneously writing them in natural language, a user achieves more efficient rules and makes it easier for other people to use them in future implementations. The creator of the application himself argues that the best tools to express a representation of laws are logical programming languages.

Therefore, Blawx is a web tool for the representation of logical declarative knowledge, characterized by being open source and user-friendly and designed specifically for coding and testing rules. It is implemented as a group of applications, offering the user a web server that stores the Blawx codes, a visual development environment based on Google's Blockly where the user can write and test the codes and, lastly, a reasoner

based on SWI-Prolog and s(CASP) that will answer the queries and justify the answers.

It is because of this last point that it has been decided to use Blawx in this part of the project, due to the close relationship it has with the first part. Basically, it allows us to express the idea that we have previously pursued by developing the rules using s(CASP) in a new and original way, allowing us to offer a more intuitive and approachable alternative to the common user, while, at the same time, we can make use of and publicize such an interesting tool such as Blawx.

Chapter 3

Modelling the criminal law

In order to illustrate the functionality of the application we have decided to focus on three specific crimes that have been considered to be significant enough to provide a first approximation of the software. The crimes chosen are sexual assault, robbery and assault and battery.

3.1 Using s(CASP)

This part of the project consisting of the rules defined in Prolog can be divided into two main parts: firstly, the definition of the offenses to conclude, according to the input values, whether a crime has been committed or not, and secondly, the calculation of the total sentence.

3.1.1 Committed crimes

Legislation regarding the modification of the sentence according to the degree of attempt and responsibility

In the first place, when calculating the penalty, it has been considered that the offenses can be divided into three levels according to the degree of attempt and responsibility. The first of these is defined in Article 62 of the Spanish Penal Code:

> To the perpetrators of attempted crime shall be imposed the penalty lower by one or two degrees than the one indicated by the Law for the consummated crime, to the extent deemed appropriate, considering the danger inherent to the attempt and the degree of execution reached.

On the other hand, regarding the degree of responsibility, we will be focusing on the figure of the accomplice, regulated in the Article 63:

> To the accomplices of a consummated or attempted crime shall be imposed the penalty lower in degree than that fixed by the Law for the perpetrators of the same crime.

To simplify the development of the rules, we have decided to establish a reduction of one degree in the sentence in case the attempt is completed and two degrees in case it is unfinished. On the other hand, if we are talking about a subject qualified as an accomplice, we will reduce the penalty by one degree.

Reducing the sentence by one or two degrees means that this sentence is going to be decreased, as it is explained in the the Article 70 of the Spanish Penal Code, so that the new minimum penalty limit will be half the minimum of the original penalty and the new maximum limit will be the one previously established as the minimum limit. For instance, if a sentence is originally 3 to 5 years, when reduced by one degree the resulting sentence would be 1 year and 6 months to 3 years. To reduce the original penalty by two degrees, one would just apply the same process to the latter result.

Legislation regarding the crime of sexual assault

We will begin by discussing the crime of sexual assault. According to article 178 of the Spanish Penal Code:

> 1. Anyone who performs an act that threatens the sexual freedom of another person without that person's consent shall be punished by imprisonment for one to four years as a perpetrator of sexual assault. Consent shall only be considered to exist when it has been freely expressed through acts which, in view of the circumstances of the case, clearly express the will of the person.
> 2. In any case, any acts of sexual nature that are carried out using violence, intimidation or abuse of a situation of superiority or vulnerability of the victim, as well as those that are performed on people who are sensory deprived or whose mental situation is abused and those that are performed when the victim's will is nullified for any cause, shall be considered sexual assault.

It is worth mentioning several aspects related to this crime. Firstly, despite other elements of the law define several cases with specific penalties, for the development of the project it has been decided to include all scenarios corresponding to the first paragraph which proposes imprisonment of one to four years.

The matter of consent

On the other hand, the concept of consent has been of interest when defining the rules, since it was recently introduced by the Organic Law 10/2022. Consent is therefore understood as a "free expression through acts which, in view of the circumstances of the case, clearly expresses the will of the person". In any case involving an attack on the sexual freedom of another person and without this free expression of will, we would be dealing with a case of sexual assault. Similarly, if in such a situation any of the circumstances of the second paragraph of this article are present, we can automatically qualify the crime of sexual assault. The problem surrounding the concept of consent has to do with the difficulty of proof in court and the scenarios where not saying "no" is not the same as consenting, which have led to the popularization of the slogan "solo sí es sí" (only yes is yes) (IndeGranada 2019). In these cases, the courts must follow

the jurisprudence that has been created by the Supreme Court through the decisions and arguments used in its judgments. Subsequently, we will analyze how this issue has been translated into our logical rules.

Specification of the input values

The three files required to ensure the correct and complete performance of the program are `delitos.pl`, `delitos.pred.pl` and `delitos.database.pl`. We shall begin with the contents of the latter, since it is the file that contains the input values for each of the subjects. First of all, it is important to add the commands `#include('delitos.pl')` and `#include('delitos.pred.pl')` at the beginning of the file in order to link it to the other two and to be able to dump the information from one into the other. Next, through the predicate `acusado(d01)` we determine the existence of a variable `d01` and we define it as our subject, who has been accused and whose input values will be processed to specify the crimes they have committed. We are going to use the predicate `evidence` to specify the input values, in order to determine that certain information is true. Thus, by introducing the predicate `evidence(d01, violencia_ag)`, we are telling the system that `d01` has been involved in a sexual act performed with the use of violence. If instead, we wish to deny a fact, we specify it with '-' before the predicate `evidence`, whereby `-evidence(d01, acto_de_contenido_sexual)` denotes that `d01` has not been involved in any act of sexual content. This idea for proving the occurrence or non-occurrence of certain facts has been inspired by the articles explaining the development of the s(LAW) framework, also based on s(CASP) and created to model the applicable legislation (Arias et al. 2020).

```
1 evidence(d01, acto_de_contenido_sexual).
2 evidence(d01, violencia_ag).
3 -evidence(d01, intimidacion_ag).
4 -evidence(d01, superioridad).
5 -evidence(d01, vulnerabilidad_victima).
6 -evidence(d01, privacion_sentido).
7 -evidence(d01, situacion_mental).
8 -evidence(d01, anulacion_voluntad).
9 -evidence(d01, jurisprudencia_confirma).
```

Thus, the input values are saved in order to find out, in this case, whether `d01` has committed a crime of sexual assault or not; by knowing that the facts `acto_de_contenido_sexual` and `violencia_ag` are certain, and due to the subsequent logical rules, we can't get its respective conclusion.

The degree of the crime is also established the same way at this point of development with the predicate `evidence`. For instance, if it is stated that the facts `autor_ag` and `complice_rob` are certain, the degree of these offenses will be those corresponding to them.

Definition of the facts

To complete this part, in the file `delitos.pl` the definition of the rules of these facts is given by the predicate `evidence`. For example:

```
1 acto_de_contenido_sexual(D):- evidence(D, acto_de_contenido_sexual).
2 n_acto_de_contenido_sexual(D):- -evidence(D, acto_de_contenido_sexual).
3 acto_de_contenido_sexual(D):- not evidence(D, acto_de_contenido_sexual),
4                               not -evidence(D, acto_de_contenido_sexual),
5                               not n_acto_de_contenido_sexual(D).
6 n_acto_de_contenido_sexual(D):- not acto_de_contenido_sexual(D).
```

In this case, all the possible alternatives to prove the certainty of `acto_de_contenido_sexual(D)` are established, using the input values and the predicate `evidence`, whether it appears as affirmative, denied or does not appear at all.

Specification of the crime of sexual assault and its requirements

The elaboration of the rules for each of the offenses begins with the most elementary definition of these offenses. It can be established that there has been sexual assault in three ways, depending on the degree of the crime, and, therefore, depending on whether the penalty will be modified or not, a calculation that we will explain later on.

Therefore, the predicate `agresion` includes, firstly, a variable that represents the subject associated with some input values and defined in the file `delitos.database.pl` and, secondly, a variable corresponding to the modification of the penalty: "autor" if it is not modified, "complice" if it is reduced by one degree or "tentativa" if it is reduced by two.

```
agresion(D,autor):-
    autor_ag(D),
    agresion_rec(D).
agresion(D,complice):-
    complice_ag(D),
    agresion_rec(D).
agresion(D,tentativa):-
    tentativa_ag(D),
    agresion_rec(D).

agresion_rec(D):-
    acto_de_contenido_sexual(D),
    requisito_agresion(D).
```

Thus, defendant `d01` has committed sexual assault if it is proven that `agresion(d01, X)` is fulfilled for any of the three options (X being "autor", "complice" or "tentativa"). For this purpose, depending on the fact confirmed as certain, one way or the other will continue to be explored. In the case of `d01`, since the fact proven to be true is `autor_ag`, the possibility of fulfilling the predicate `agresion(d01, autor)` is considered, since the other two options cannot be satisfied once both `complice_ag` and `tentativa_ag` are proven to be false. After checking `autor_ag(d01)`, the system continues to the next rule, and so on until all of them are proven true. In this case, the second and last rule is `agresion_rec(D)`, which can be satisfied by two paths, one of them being the one shown in line 11. `agresion_rec` is considered to be fulfilled if `acto_de_contenido_sexual(D)` and `requisito_agresion(D)` can be proved. The first one is a fact, so the input values themselves will confirm whether it can be demonstrated or not. On the other hand, `requisito_agresion(D)` can be proved by three different means.

```
requisito_agresion(D):-
    violencia_ag(D).
requisito_agresion(D):-
    intimidacion_ag(D).

requisito_agresion(D):-
    abuso_situacion(D).
abuso_situacion(D):-
```

```
9       superioridad(D).
10  abuso_situacion(D):-
11      vulnerabilidad_victima(D).
12
13  requisito_agresion(D):-
14      condicion_especifica_victima(D).
15  condicion_especifica_victima(D):-
16      privacion_sentido(D).
17  condicion_especifica_victima(D):-
18      situacion_mental(D).
19  condicion_especifica_victima(D):-
20      anulacion_voluntad(D).
```

By checking the input values, we will find out which facts are true and, consequently, which rules are fulfilled, and we will confirm or deny the `requisito_agresion(D)`.

Modelling the consent

Lastly, the interpretation taken to elaborate on the idea of consent found in the article is as follows. According to the Penal Code, a sexual assault exists whenever an act of sexual content has occurred without consent. Section 2 of article 178 sets out the situations where it is assumed that consent can never be given, and all of them have been included in the predicate `requisito_agresion`. However, and because there may be cases where consent has not been given and are not specified in the article, we must therefore provide an alternative to the first predicate `agresion_rec(D)`.

```
1  agresion_rec(D) :-
2      acto_de_contenido_sexual(D),
3      not consentimiento(D).
```

So, through the fact `consentimiento`, we consider sexual assault all those acts of sexual content that have occurred without consent. The fact is defined as follows:

```
1  consentimiento(D):- evidence(D, jurisprudencia_confirma).
2  n_consentimiento(D):- -evidence(D, jurisprudencia_confirma).
3  consentimiento(D):- not evidence(D, jurisprudencia_confirma),
4                  not -evidence(D, jurisprudencia_confirma),
5                  not n_consentimiento(D).
6  n_consentimiento(D):- not consentimiento(D).
```

While similar to the other fact definitions, this one is different due to the introduction of a new predicate: `jurisprudencia_confirma`. The idea we aim to convey implies that consent will exist as long as there is jurisprudence that supports its validity. This model could also be used with other ambiguous issues in the judicial field where it would be convenient to look at the past case law of a given event in order to know what ought to be done in that particular situation. Thus, depending on whether the value of the `jurisprudencia_confirma` input is positive or not, we will be able to estimate what a judge could determine in this case, taking as an example the past decisions of the Supreme Court in similar cases.

Legislation regarding the crime of robbery

The second crime used in the software development is the crime of robbery, under Article 237 of the Spanish Penal Code.

> Those who, with the intent of profit, seize other people's movable property by using force to gain access to or leave the place where the property is located, or violence or intimidation against individuals, either when committing the crime, to protect the escape, or against those who come to the victim's aid or who pursue him, are guilty of the crime of robbery.

Since each of the possible situations determined in the article has a different sentence, we have decided to specifically modulate the case of robbery with violence and intimidation, which, according to Article 242 of the Penal Code, is punishable by a prison sentence of two to five years:

> 1. Those who commit robbery with violence or intimidation against another person shall be punished with a prison sentence of two to five years, without prejudice to the sentence that may be imposed for the acts of physical violence committed by the perpetrator.

This decision to focus only on robbery of these type is based on the fact that the development of the other cases (robbery with the use of force or in an inhabited house)

was trivial and did not contribute any matter of interest to this explanation. They can be modeled in a future full development of the software.

Specification of the crime of robbery and its requirements

```
robo(D, autor):-
    autor_rob(D),
    apoderacion_cosa(D),
    requisito_robo(D).
robo(D, complice):-
    complice_rob(D),
    apoderacion_cosa(D),
    requisito_robo(D).
robo(D, tentativa):-
    tentativa_rob(D),
    apoderacion_cosa(D),
    requisito_robo(D).
```

The same dynamics as for the crime of assault apply in this case. The software will confirm that the defendant `d01` has committed a crime of robbery if it is satisfied that `robo(d01, autor)`, `robo(d01, complice)` or `robo(d01, tentativa)` are true. To do this, besides confirming the statement denoting the degree of the penalty (`autor_rob`, `complice_rob` or `tentativa_rob`), it is necessary to verify that the statement `apoderacion_cosa(d01)` is true, as well as `requisito_robo(d01)`, which we can further confirm according to the following rules:

```
requisito_robo(D):-
    objeto(D),
    momento(D).
objeto(D):-
    violencia(D).
objeto(D):-
    intimidacion(D).
momento(D):-
    comision(D).
momento(D):-
    huida(D).
```

Legislation regarding the crime of assault and battery

Finally, the last crime used to provide a first approach to this software is the crime of assault and battery, defined in Article 147 of the Spanish Penal Code:

> 1. Anyone who, by any means or procedure, causes an injury to someone else causing harm to their bodily integrity or their physical or mental health, shall be punished, as the offender of the crime of assault and battery, with a prison sentence of three months to three years or a fine of six to twelve months, as long as the injury objectively requires medical or surgical treatment for its healing, in addition to initial medical assistance. The simple surveillance or medical follow-up of the course of the injury will not be considered medical treatment.

Although the punishment for this crime could consist of a fine, for the purpose of the system's development we have decided to overlook this possibility and focus on the prison sentence, since this is what our application has been created for. In a future development it could be possible to add other types of penalties in addition to imprisonment.

Specification of the crime of assault and battery and its requirements

Thus, following the example of the previous offenses, the assault and battery offense would be modulated as follows:

```
lesion(D, 0):-
    autor_les(D),
    lesion_sec(D).
lesion(D, 1):-
    complice_les(D),
    lesion_sec(D).
lesion(D, 2):-
    tentativa_les(D),
    lesion_sec(D).

lesion_sec(D):-
    menoscabo_fisico(D),
    requisito_lesion(D).
lesion_sec(D):-
    menoscabo_mental(D),
    requisito_lesion(D).
```

```
17
18 requisito_lesion(D):-
19     asistencia_facultativa(D),
20     tratamiento_medico(D).
21 requisito_lesion(D):-
22     asistencia_facultativa(D),
23     tratamiento_quirurgico(D).
```

Expressing the results in natural language

Once the contents of the files `delitos.pl` and `delitos.database.pl` have been explained, the remaining file to be described is `delitos.pred.pl`. This file is essential to provide the user a justification tree expressed in natural language displaying the stages performed when carrying out a query. The contents of this file are organized in two main parts. The first one of them makes it possible to avoid a justification tree that is excessively vast and not very efficient. Using the `show` instruction, and as long as we call the s(CASP) interpreter with the `--short` option, we can tell the system exactly which predicates to display in the justification tree. For example, `# show agresion/2` or `# show acto_de_contenido_sexual/1`.

Each predicate must include the number of arguments that go with it according to its definition. Thus, we are telling the system that, if we call s(CASP) with the `--short` instruction, only the specified predicates will be shown, and only if they are proved to be true (because we specified it that way). To see the rest, refer to the Appendix B.

On the other hand, the thickest part of the file consists of a translation of each of the predicates into natural language, in such a way that, following the order established by the rules, the message corresponding to the rule fulfilled will be displayed.

```
1 #pred agresion(X, Y) :: '@(X) has been accused of sexual assault'.
2 #pred acto_de_contenido_sexual(X) ::
3     '@(X) has committed an act of sexual nature'.
4 #pred requisito(X) :: 'a requirement is met'.
5 #pred violencia_ag(X) ::
6         'the sexual act has been committed with the use of violence'.
7 #pred intimidacion_ag(X) ::
```

```
8           'the sexual act has been committed with the use of intimidation'.
9   #pred abuso_situacion(X) ::
10          'the sexual act has been committed taking advantage of the situation'.
11  ...
```

And so on. These lines of the code allow us to display a justification tree that can be understood by any user thanks to the use of common language. Te rest of the code for the justification tree of sexual assault and the other crimes is shown in the Appendix B.

3.1.2 Calculation of the final sentence

Legislation regarding the penalty calculation

The previously presented code allows us to find out, based on the input values, whether a specific subject has committed any of the crimes under consideration. To estimate the final penalty, it is therefore necessary to elaborate on the contents of the files that conform the software.

In order to determine the resulting penalty as the sum of several offenses, we will consider real competition ("concurso real"), defined in Article 73 of the Spanish Penal Code:

> Those responsible for two or more crimes or misdemeanors will be imposed all the penalties corresponding to the various offenses to be fulfilled simultaneously, if possible, by the nature and effects of the same.

Thus, we will consider as the final penalty the sum of the penalties as specified by the law and bearing in mind the minimum of 6 months and the maximum of 20 years set forth in Article 36 of the Penal Code.

Output information in natural language

Initially, it would not be necessary to add anything to the file `delitos.database.pl`, since it only specifies the input values and these are the same throughout the whole development of the software. On the other hand, in `delitos.pred.pl` it would only be necessary to add the line by which we specify how we want to display the result of the penalty calculation in natural language:

```
#pred penaTotal(N, Delitos, [MinA, MinM, MinD], [MaxA, MaxM, MaxD]) ::
        'The final sentence will range from @(MinA) years, @(MinM) months
        and @(MinD) days to @(MaxA) years, @(MaxM) months and @(MaxD)
        days in prison'.
```

Rules definition for the calculation

The essential part of the calculation is included in the file `delitos.pl`. There, we defined the rules required to reach the final result. We will start by explaining the main rule:

```
penaTotal([Delito-Grado|Resto], [MinA, MinM, MinD], [MaxA, MaxM, MaxD]).
```

This predicate's input value is `[Delito-Grado|Resto]`, a list composed of pairs formed by the crime committed and the value specifying the degree used for the future modifying of the sentence. For example, an input value could be `[agresion-autor, robo-tentativa]`. To get this list we used a predicate `delito`, which, with the input value of the accused subject, will return the desired list.

```
delito(D, [agresion-X, robo-Y, lesion-Z]):-
    acusado(D),
    agresion(D,X),
    robo(D,Y),
    lesion(D,Z).
delito(D, [agresion-X, robo-Y]):-
     acusado(D),
     agresion(D,X),
     robo(D,Y).
...
```

And so on with the rest of the combinations. By doing so, we address all possible groups of offenses among those we have taken into account in our work. Thus, when we ask to solve a query, it will first call the predicate `delito` to generate the list that we will send as input value to `penaTotal` in the second part of the query.

On the other hand, the output of `penaTotal` is composed of two sets of values corresponding to the minimum and maximum penalty, expressed in years, months and days.

```
penaTotal([Delito-Grado|Resto], [MinA, MinM, MinD], [MaxA, MaxM, MaxD]):-
    penaTotal_rec([Delito-Grado|Resto], MinProvisional, MaxProvisional),
    minimo(MinProvisional, 0.5, Min),
    minimo(MaxProvisional, 20, Max),
    conversor(Min, MinA, MinM, MinD),
    conversor(Max, MaxA, MaxM, MaxD).
```

The predicate `penaTotal` is formed by five calls to three different rules. First, the predicate `penaTotal_rec`, which is given the list of crime-grade pairs as input values and, as output, returns the range of the sentence in years according to that list. Thus, if the list is empty, the result will be a range from 0 to 0 years:

```
penaTotal_rec([], 0, 0).
```

Alternatively, it will function as a recursive predicate that allows the increasing of the penalty according to the offenses found in the list.

```
penaTotal_rec([DelitoR-GradoR|RestoR], MinimoR, MaximoR):-
    pena(DelitoR, MinC, MaxC),
    modifica(GradoR, MinC, MaxC, MinMod, MaxMod),
    penaTotal_rec(0, RestoR, MinResto, MaxResto),
    MinimoR is MinMod + MinResto,
    MaximoR is MaxMod + MaxResto.
```

Other required predicates

The predicate `pena`, on the other hand, consists of the facts that provide the sentence stated by the Penal Code for each of the crimes in its original form, as they

were previously discussed. Therefore, for sexual assault we get a range from 1 to 5 years, for robbery, from 2 to 5 and for assault and battery from 0.5 (6 months) to 3 years. For instance, for the crime of assault it would be modelled like this: `pena(agresion,1,4)`.

`modifica` is the predicate that will allow us to modify the penalty according to the degree attached to it. Thus, once `pena` provides the original penalty assigned to that crime of the `DelitoR-GradoR` pair, `modifica` modifies it thanks to the variable `GradoR` depending on whether it corresponds to `autor`, `complice` or `tentativa`. As explained above, the variable remains original in the first case, the penalty is lowered by one degree in the second, and by two degrees in the third and last case.

```
1 modifica(autor, MinC, MaxC, MinC, MaxC).
2 modifica(robo, MinC, MaxC, MinMod, MaxMod):-
3     inf_enGrado(MinC, MaxC, MinMod, MaxMod).
4 modifica(tentativa, MinC, MaxC, MinMod, MaxMod):-
5     inf_enGrado(MinC, MaxC, MinTemp, MaxTemp),
6     inf_enGrado(MinTemp, MaxTemp, MinMod, MaxMod).
```

On the other hand, `inf_enGrado` allows us to determine what the result would be when it comes to reducing the penalty by one degree, as stipulated in Article 70.1.2 of the Spanish Penal Code.

Thus, `penaTotal_rec` performs a first query to find out the general sentence corresponding to the first crime in the list `[DelitoR-GradoR]`, modifies it as indicated by the variable `GradoR`, and calls `penaTotal_rec` again, this time with the rest of the list, until it reaches the empty list. This way, the lower and upper ranges of the penalties are added, resulting in the variables `MinimoR` and `MaximoR`, which will correspond to `MinProvisional` and `MaxProvisional` in the original `penaTotal` query.

The only remaining steps to reach the resulting sentence range are two. First of all, we must avoid exceeding the penalty limits imposed by Article 36 of the Penal Code, using some simple predicates that help us to choose the minimum or maximum value between two offered. Secondly, the last step will allow us to provide the values corresponding to the resulting penalty range in years, months and days, instead of a decimal figure that could be unintuitive. Both predicates are formed by simple calcu-

lations and and available in the Appendix B, as it is `inf_enGrado()`.

Finally we now have the output values ready to be returned in the query made to `penaTotal`.

3.2 Blawx

3.2.1 Definition of the rules

The creation of a Blawx project involves three fundamental parts: the definition of the rules, the design of the code and the creation of the required tests. Therefore, the first step to be taken is the definition of the rules using the editor provided for this purpose and following the guidelines described in the CLEAN documentation (https://github.com/lexpedite/clean/). We decided to focus only on the crime of sexual assault to illustrate the functionality of Blawx since out of the three it is the most interesting one and the other two would just have to follow the same model. Thus, the definition of the rules would be set out as follows:

```
Crimes Committed

A person has been accused of sexual assault if
1.
  (1) the person has committed an act of sexual nature
    (a) it has been committed with the use of violence
    (b) it has been committed with the use of intimidation
    (c) it has been committed taking advantage of the situation
      (i) because the person held a position of superiority
      (ii) because the victim was in a vulnerable position
    (d) it has committed when the victim had a specific condition
      (i) because they were unable to act independently
      (ii) because their mental situation was abused
      (iii) because their will was overridden
  (2) it happened without the victim's consent.
```

The rules follow the same structure as the one used previously in s(CASP).

3.2.2 Code design

Once the rules that compose the project have been presented, it is time to design the code in blocks for one of these rules. The interface is displayed as follows:

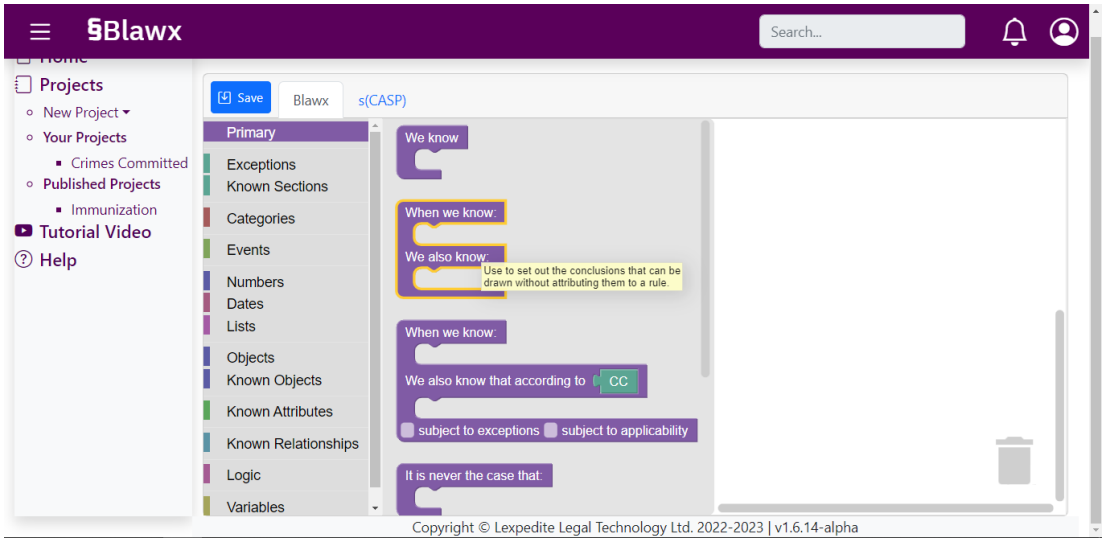

Figure 3.1: Blawx code design interface

On the left the toolbox can be seen, formed by a set of drop-down menus. Each of these drop-downs corresponds to a group of blocks based on their functionality. In order to use these blocks, we simply drag them to the workspace. The Blawx work manual explains the use and functionality of each of the blocks in more detail. However, we will only explain whatever is relevant to the development of our idea.

By clicking on the Section button, a drop-down menu appears with the rules framework previously defined. By clicking on each line of the framework, a new workspace for designing the respective code is displayed. Below, we will get into each one of them.

Commission of sexual assault

The first line of the rules matches the definition of sexual assault or, also, what would be `assault_rec(D)` in Prolog code:

```
1  1 A person has been accused of sexual assault if
```

In this workspace, we first establish the facts that we know with certainty using a "We know" block. It allows us to create the categories `person` and `sexual_intercourse`, with the latter one representing the act of sexual content mentioned in the corresponding article of the Penal Code. Next, with the block that allows creating new attributes, we establish the ones that the objects belonging to this category will have and give them a natural language translation that will be used later on. An object of type `person` will always be associated with the boolean attribute `guilty`, which will allow us to determine whether that person is indeed guilty of sexual assault or not. On the other hand, `sexual_intercourse` has two attributes `alleged_ofender` and `victim` of type `person`, which represent what their own names imply.

The other big block that we find in the description of this rule corresponds to the block "It is never the case that", through which we set out a combination of statements that cannot all be true at the same time. Thus, we ensure that the two individuals involved in the `sexual_intercourse` event must necessarily be two different objects.

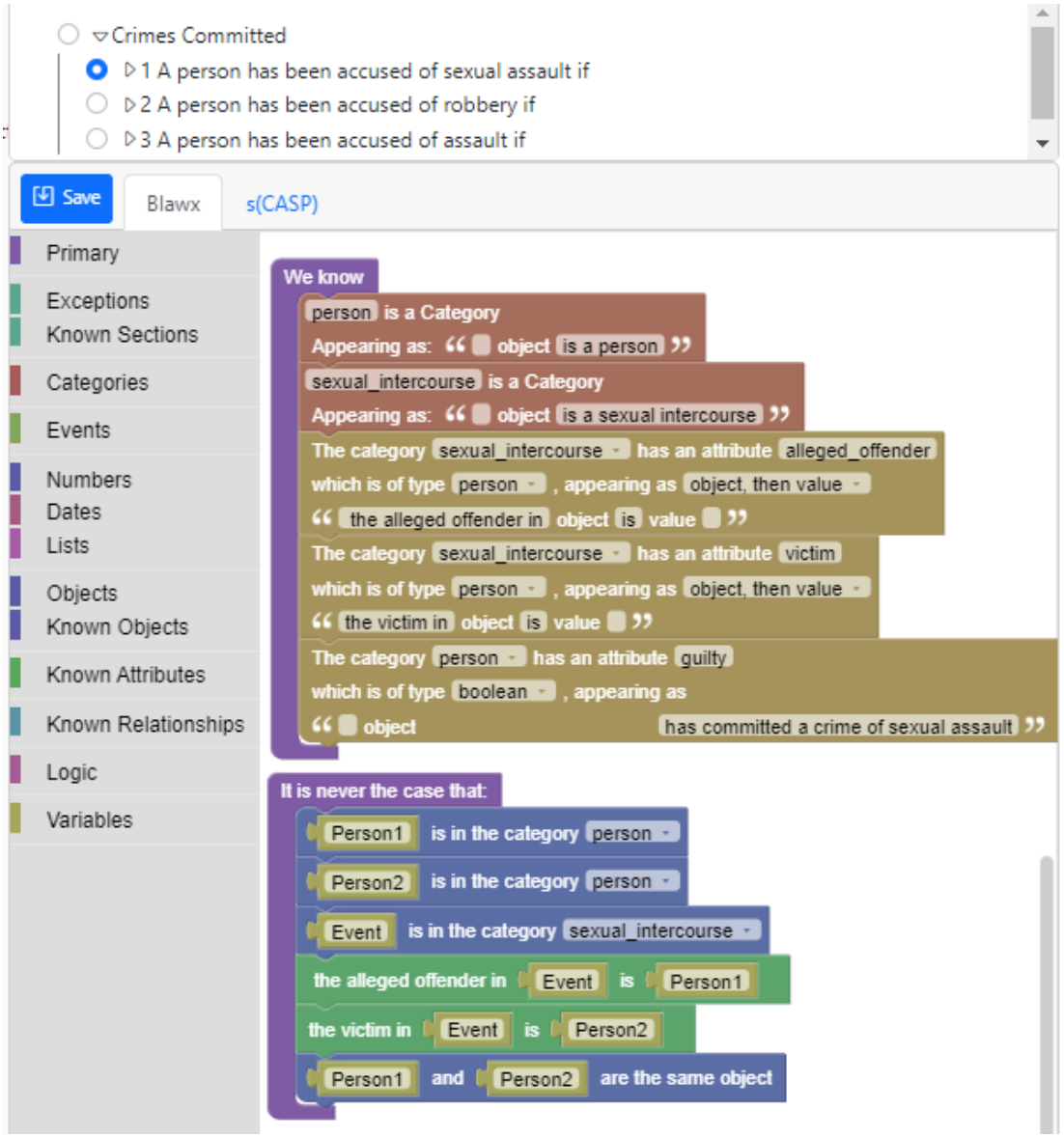

Figure 3.2: Sexual assault rule 1 on Blawx

Although similar, the Blawx block code does not correspond exactly to what is expressed in the Prolog code. An example for this is that in Prolog we needed two implementations of the rule `agresion_rec(D)` if we wanted to specify that a sexual assault existed if either one or the other was fulfilled (OR operator); instead, in Blawx we can put it all in the same rule.

Thus, the first line is split up into two different rules:

```
1  1 A person has been accused of sexual assault if
```

```
2   (1) the person has committed an act of sexual nature
3   (2) it happened without the victim`s consent.
```

Requirement of the act of sexual nature

The first rule corresponds to the predicate `acto_de_contenido_sexual(D)` from s(CASP) code. There is no need to write Blawx code regarding this line. On the other hand, one could also say that this rule includes the predicate `requisito_agresion(D)` since it displays the four means whereby this rule is fulfilled:

```
1  ...
2   (1) the person has committed an act of sexual nature
3    (a) it has been committed with the use of violence
4    (b) it has been committed with the use of intimidation
5    (c) it has been committed taking advantage of the situation
6    (d) it has committed when the victim had a specific condition
```

Sexual assault through the use of violence

The first of the cases where the requirement to determine the existence of a sexual assault could be met was the one where the act of sexual content had been committed with the use of violence. This has been coded in the form of a "We know" block and a "When we know... we know..." block. We already are familiar with the first one; in this case, we use it to create a new attribute of type boolean, `violence`, associated to an object of type `sexual_intercourse`. With it, we will be able to define, through the corresponding input value, whether there has been violence in that sexual act or not.

On the other hand, the second block is used to determine that, in fact, if the alleged aggressor has performed the act of sexual content with violence, it is possible to con-

clude that we are dealing with a sexual assault. This block also allows us to specify the section through which we are confirming the existence of the crime.

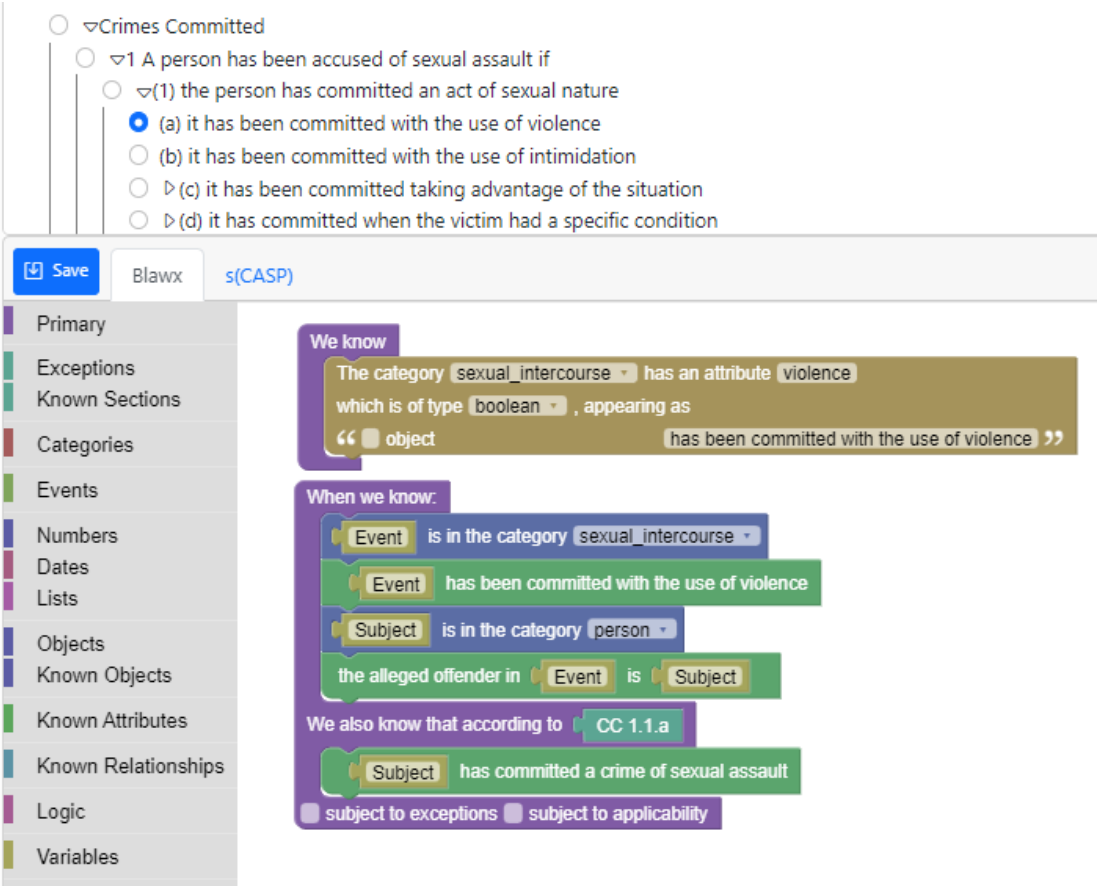

Figure 3.3: Sexual assault rule 1.1.a on Blawx

Sexual assault through the use of intimidation

The same applies in case (b) whereby the assault requirement is met if the act of sexual content has been perpetrated with the use of intimidation.

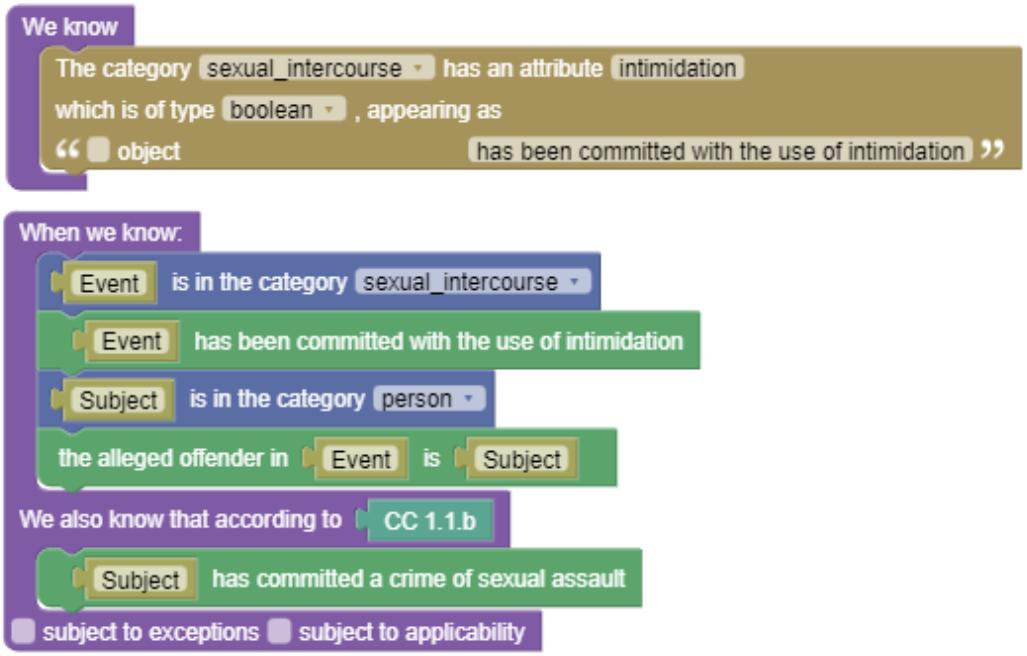

Figure 3.4: Sexual assault rule 1.1.b on Blawx

Sexual assault through the abuse of a situation

In third place we encounter the rule that would correspond to the predicate in Prolog `abuso_situacion(D)`. In this case, there is also no need to include any Blawx code. This rule is further deployed in two other rules corresponding to the contexts in which the alleged aggressor is considered to have taken advantage of the situation:

```
1 ...
2   (c) it has been committed taking advantage of the situation
3    (i) because the person held a position of superiority
4    (ii) because the victim was in a vulnerable position
```

These rules are modularized as follows:

Figure 3.5: Sexual assault rules 1.1.c.i and 1.1.c.ii on Blawx

This time, the category to which the new `position_superiority` and `vulnerable_position` attributes are associated is person. However, one of the rules applies in the case that the person object is the alleged aggressor and the other when it is the victim of the act, respectively. Thus, we can conclude that there has been sexual assault if the person object corresponding to the alleged aggressor of the event has acted using his position of superiority or, on the other hand, if the person object corresponding to the victim of the event was in a vulnerable position.

Sexual assault through some specific condition of the victim

Lastly, within the requirements for a sexual assault we find rule (d), corresponding to the Prolog predicate `condicion_especifica_victima(D)`. Similar to how we have coded rule (c) above, it is not necessary to write any specific Blawx code here, since all that is necessary is already stated in the three rules that derive from it:

```
...
  (d) it has committed when the victim had a specific condition
   (i) because they were unable to act independently
   (ii) because their mental situation was abused
   (iii) because their will was overridden
```

The three have been coded in a similar way, since they are all based on a specific condition that the victim had at the time of the act, i.e., an attribute associated with an object of type person that corresponds to the victim's figure. The three attributes corresponding to each of the three rules respectively are `unable`, `mental_situation` and `will_overriden`. Since they follow the same model as the two previous rules, we show how the look like in the Appendix A.

Therefore, all the situations whereby a requirement that qualifies an act as sexual assault can be met are codified.

Sexual assault based on lack of consent

Lastly, we must describe the case where, even if none of these situations are met, given a lack of consent, the existence of a sexual assault can still be considered.

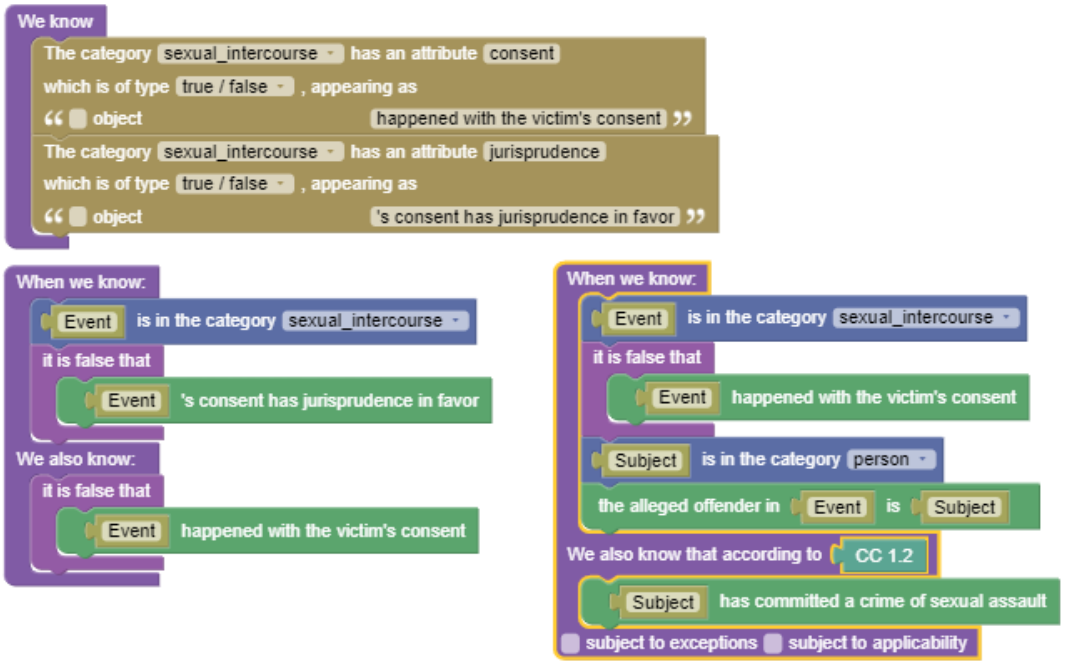

Figure 3.6: Sexual assault rule 1.2 on Blawx

The interesting thing about this rule is that we introduce a new Blawx tool: "It is false that". After creating the boolean attribute `jurisprudence` and `consent` associated with objects of type `sexual_intercourse`, we use them in a negative way in the first block "When we know... We also know..." as shown, creating a cause and effect relationship whereby, if there is no jurisprudence defending the validity of consent in that situation, then we could say that the act has occurred without the consent of the victim. The other block replicates the idea used on the requirements' section so that, if it is false that the sexual intercourse happened with the victim's consent, we should consider it to be a crime of sexual assault.

3.2.3 Test implementation

The last step to implement a Blawx project after defining the rules and building the code is the creation of the corresponding tests. In our case, as we have only focused on the crime of sexual assault for now, only one test called `guilty_sexual_assault` was needed. The tests in Blawx are defined in a similar way as with the code: using

blocks. In this way, to perform the final query we will use the block “Is it true that”, for which Blawx will try to find the values for its variables that would evaluate the statement as true. In our case, it is shown as follows:

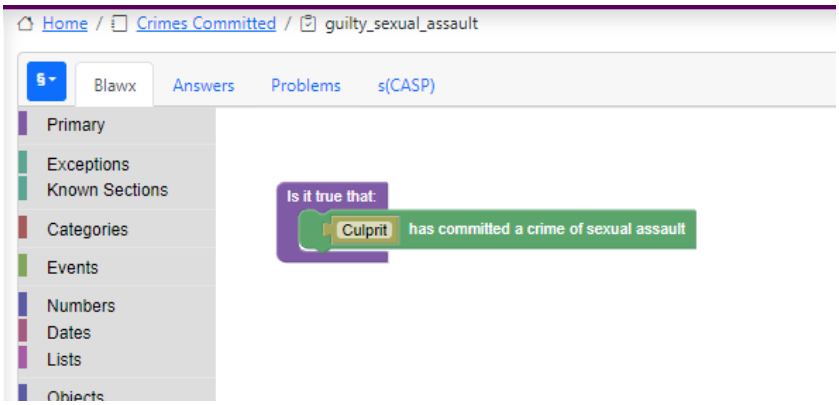

Figure 3.7: Blawx test for sexual assault

Subsequently, in the Chapter 4, the functionality of the test will be further explained.

3.2.4 Use of Blawx

Once created the test we will use to verify the functionality of our implementation, we must open the scenario editor to add the input values of our query and observe the results. Before inserting these values, in the View tab we can select the attributes we wish to hide in the selection of values. In our case, these would be the `guilty` attribute because it corresponds to the output value we are looking for and the `consent` attribute, because in the cases where it is relevant, its occurrence depends only on whether there is case law to prove it.

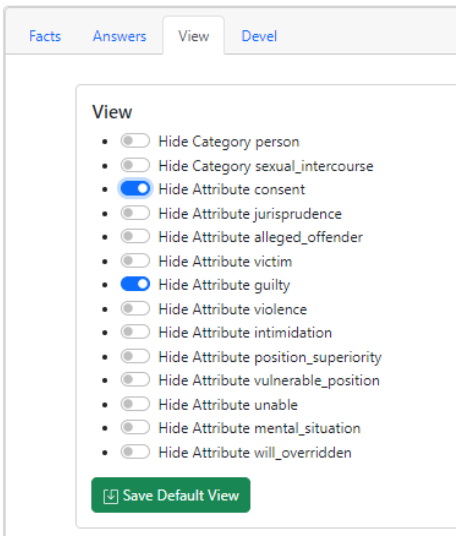

Figure 3.8: Blawx View tab

Next, to select the known facts that will function as input values, we use the drop-down menu where all possible attributes are presented. Once selected, Blawx enables us to determine the fact as true or false and to enter the variable we need.

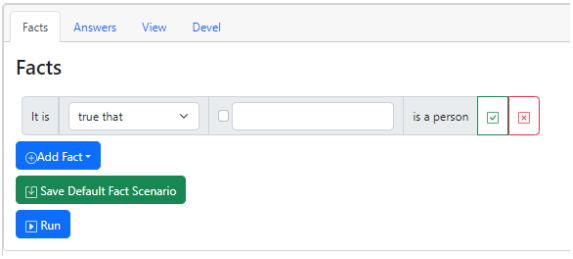

Figure 3.9: Blawx test facts definition

Once some variables have been added, the tool itself stores them in order to be used in other facts:

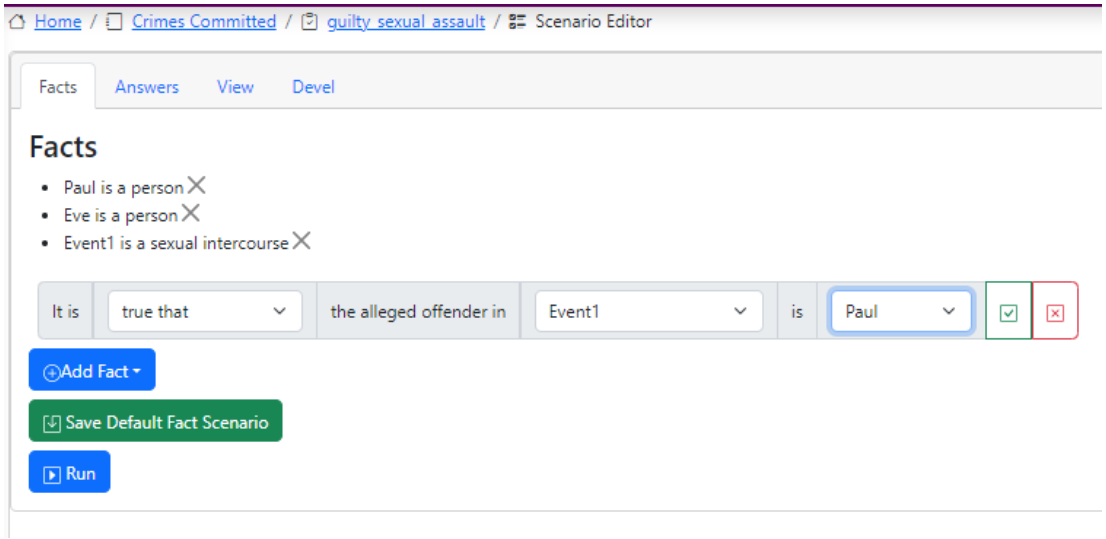

Figure 3.10: Blawx test facts definition with stored information

Once the facts needed for our specific situation are stated, we can run the program, and subsequently, it will show the output with the corresponding answers (or the absence of them).

Chapter 4

Examples

This chapter is intended to test the functionality of what has been achieved in the project. For that purpose, we will study seven different examples.

The files necessary to run the part of the software implemented in s(CASP) can be found in the following repository: https://github.com/elenais7/TFG-Elena-Martinez. Its functionality can be tested either through a terminal once the system is installed locally and following the instructions explained later in example 4.6, or in the Playground by copying the code from the `.pl` files and loading it in the page. The Blawx project is also available in the repository so that it is possible to import it into the system's web and run it as explained in Section 3.2.4.

4.1 Sexual assault by the abuse of the victim's mental situation

The first example we provide is based on a subject `a01` that was involved as the perpetrator in an act of sexual nature where the victim's mental situation was abused, resulting in a crime of sexual agression. In the file `delitos.database.pl` we specify the facts that are true for this subject, which in this case are `acto_de_contenido_sexual`,

`situacion_mental` and `autor_ag`.

In order to run the program in the Playground, the information of the three files need to be entered and loaded. Once this is done, we can ask the query `delito(a01, D)`, `penaTotal(D, [A,B,C],[X,Y,Z])`. and observe the results:

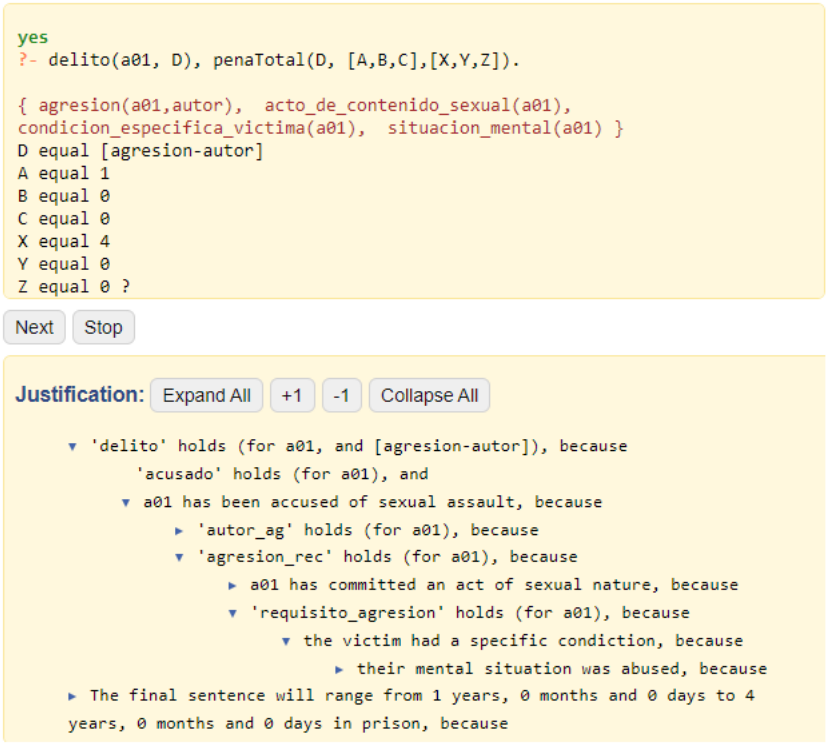

Figure 4.1: Playground results of example 4.1

In Blawx, we created an object of type person called `Paul` that corresponds to our defendant and, on the other hand, we need to create another one of the same type called `Eve` that would be the victim and to which attributes may also be associated. `Event1` corresponds to the act of sexual content. In this case, we have determined that `Eve`'s mental situation was abused, so that, according to the Penal Code, the act would correspond to a crime of sexual assault.

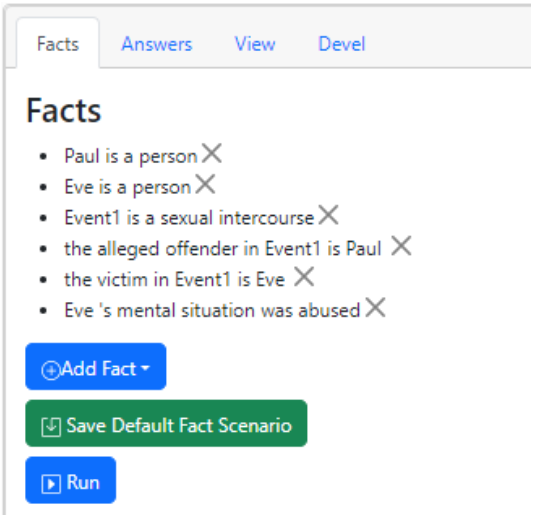

Figure 4.2: Blawx set up for example 4.1

The following is the program's response after the query has been executed:

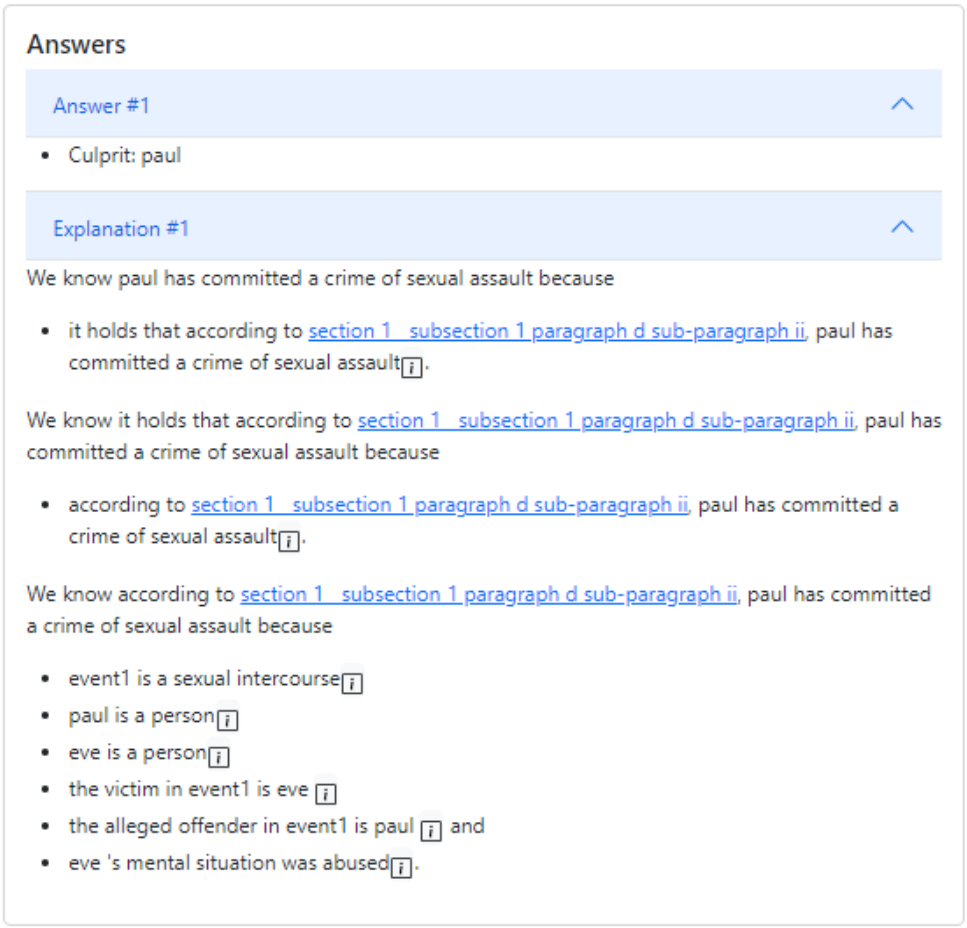

Figure 4.3: Blawx results of example 4.1

4.2 Sexual assault by the perpetrator's position of superiority

In the next example, the facts proven to be true are `acto_de_contenido_sexual`, `superioridad` and `autor_ag`. According to the laws explained, this also should be considered sexual assault and our software should come to the same conclusion. In Blawx, we check what happens if we introduce the fact that `Paul` held a position a superiority. In both cases, the answers are the same as the ones shown in the previous example, proving `a02` (`Paul`) as guilty of a crime of sexual assault. The images showing the outputs are shown in the Appendix A.

4.3 Not sexual assault by the victim's position of superiority

We want to compare the previous example to what would happen if it was the victim (`Eve`) who held a position of superiority (there should be no sexual assault). There is not much point in testing this in the Playground since there is no object that acts as a victim, so we will only be showing what would happen in Blawx.

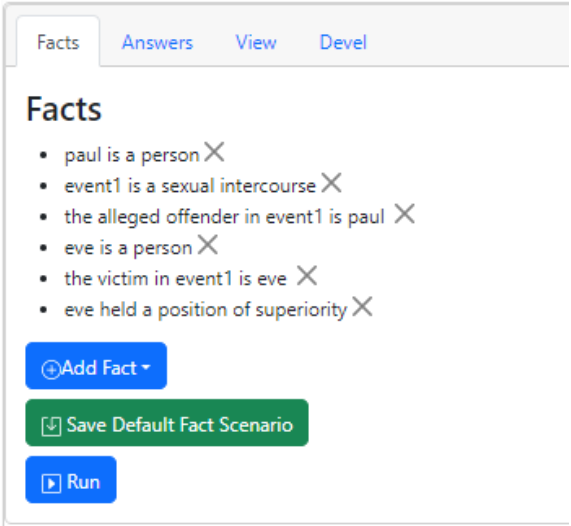

Figure 4.4: Blawx set up for example 4.3

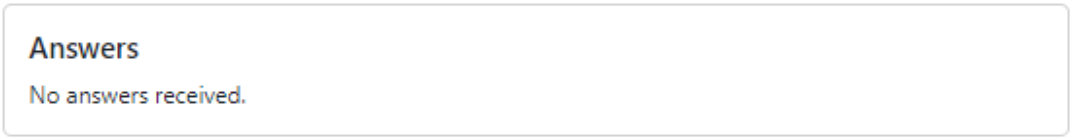

Figure 4.5: Blawx results of example 4.3

4.4 Not sexual assault when there's consent

At last, we need to test the subject of consent. Essentially, if there is case law in favor of the existence of consent in such case, then it should not be sexual assault, since it is considered to be a consensual sexual act.

In Blawx, the fact selection would look like this:

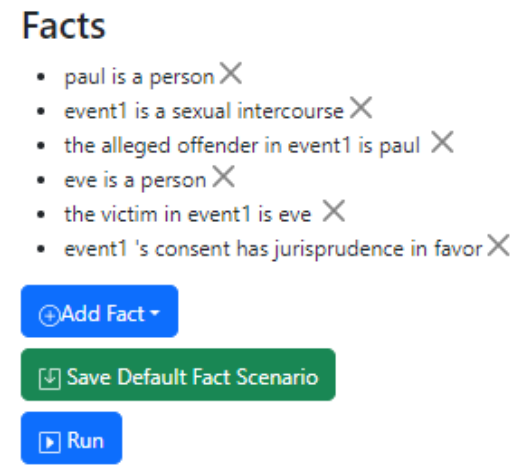

Figure 4.6: Blawx set up for example 4.4

In this case, both the Playground and Blawx show no answers, so we can conclude the defendant (`Paul`) has not committed sexual assault. The images in the Appendix A show the output in both cases.

4.5 Sexual assault when there's no consent

On the other hand, if there is no case law in favor of consent in this case, then it has not been a consensual sexual act but rather a sexual assault that has occurred. Here, as shown in previous examples, the Playground tells us the defendant `a05` has committed a crime of sexual assault and the answer in Blawx sets `Paul` as the culprit of the crime. These also can be found in Appendix A.

4.6 Sexual assault and robbery

Finally, we will test two cases in which there have been other crimes besides sexual assault, so they currently cannot be implemented in Blawx.

In this case, we will run the program both in the s(CASP) online Playground and locally and we will show how the result would look like in both situations.

First, the defendant `d01` for which the facts `acto_de_contenido_sexual`, `violencia_ag`, `apoderacion_cosa`, `violencia_rob`, `comisión`, `autor_ag` and `cómplice_rob` are demonstrated. According to this evidence, the accused would have committed the crimes of sexual assault with the corresponding degree of `autor` and robbery with the degree of `complice`. Following the explanation given in the previous chapter, the penalty for the first offense would be from 1 to 4 years and for the second, although it would originally be from 2 to 5 years, after the decrease of one degree it results in the range of 1 to 2 years. Therefore, the resulting total range would be 2 to 6 years.

To execute the query locally, it is necessary to enter the command `scasp -i --tree --short --human delitos.database.pl`: `i` to run it interactively, `tree` to print the justification tree, `short` for a more efficient and intuitive justification and `human` to show us the justification in natural language.

Subsequently, we can run our query `delito(d01, D), penaTotal(D, [A, B, C], [X, Y, Z])` and observe the results.

First, the terminal shows us the query we have performed with the new variables assigned, the number of answers found and the time it took to find them.

```
?- delito(d01, D), penaTotal(D, [A, B, C], [X, Y, Z]).
% QUERY:I would like to know if
     'delito' holds (for d01, and Var0) and
     The final sentence will range from Var1 years, Var2 months and Var3 days
 to Var4 years, Var5 months and Var6 days in prison.

        ANSWER: 1 (in 6926.212 ms)
```

Secondly, the justification tree is displayed.

```
JUSTIFICATION TREE:
d01 has been accused of sexual assault, because
    d01 has committed an act of sexual nature, and
    the sexual act has been committed with the use of violence.
d01 has been accused of robbery, because
    d01 has taken possession of a movable thing from another person, and
    a requirement in d01 robery is met, because
        the condition "objeto" holds, because
            d01 took is with the use of violence.
        and the condition "momento" holds, because
            they used it when taking the thing.
The final sentence will range from 2 years, 0 months and 0 days to 6 years,
 0 months and 0 years in prision, and
The global constraints hold, because
    the global constraint number 1 holds.
```

And lastly, the model with the facts and logical rules that have been involved and the result values.

```
MODEL:
{ agresion(d01,autor), acto_de_contenido_sexual(d01), violencia_ag(d01),
  robo(d01, complice), apoderacion_cosa(d01), objeto(d01),
  violencia_rob(d01), comision(d01) }
BINDINGS:
Var0 equal [agresion-autor,robo-complice]
Var1 equal 2
Var2 equal 0
Var3 equal 0
Var4 equal 6
Var5 equal 0
Var6 equal 0 ?
```

We can notice in the model the result corresponding to the variable `D` (Var0), which would be the list of crimes `[agresion-autor,robo-complice]`, and the variables corresponding to the range of the final sentence (Var1-Var6), which, effectively, show a range from 2 to 6 years, as expressed in natural language in the last line of the justification tree ("The final sentence will range from 2 years, 0 months and 0 days to 6 years, 0 months and 0 days in prison"). .

In the Playground, once the information of the three files has been entered and loaded, the query would be executed in the same way, giving the following result:

```
?- delito(d01,D), penaTotal(D, [A, B, C], [X, Y, Z]).

{ agresion(d01,autor),  acto_de_contenido_sexual(d01),  violencia_ag(d01),
robo(d01,complice),  apoderacion_cosa(d01),  objeto(d01),  violencia_rob(d01)
,  comision(d01) }
D equal [agresion-autor,robo-complice]
A equal 2
B equal 0
C equal 0
X equal 6
Y equal 0
Z equal 0 ?

yes
```

Figure 4.7: Playground results for d01

```
Justification: Expand All  +1  -1  Collapse All

▼ 'delito' holds (for d01, and [agresion-autor,robo-complice]), because
      'acusado' holds (for d01), and
    ▼ d01 has been accused of sexual assault, because
        ▸ 'autor_ag' holds (for d01), because
        ▼ 'agresion_rec' holds (for d01), because
            ▸ d01 has committed an act of sexual nature, because
            ▼ 'requisito_agresion' holds (for d01), because
                ▸ the sexual act has been committed with the use of violence, because
    ▼ d01 has been accused of robbery, because
        ▸ 'complice_rob' holds (for d01), because
        ▸ d01 has taken possession of a movable thing from another person, because
        ▼ a requirement in d01 robbery is met, because
            ▼ the condition "objeto" holds, because
                ▸ d01 took it with the use of violence, because
            ▼ and the condition "momento" holds, because
                ▸ they used it when taking the thing, because
▸ The final sentence will range from 2 years, 0 months and 0 days to 6 years, 0 months and 0 days in
prison, because
```

Figure 4.8: Playground results for d01 (justification tree)

4.7 Robbery and assault and battery

In the second case, defendant d02 has as confirmed facts acto_de_contenido_sexual, jurisprudencia_confirma, apoderacion_cosa, intimidación_rob, comision, menoscabo_mental, asistencia_facultativa, tratamiento_medico, cómplice_ag, tentativa_rob y complice_les. We have decided to include certain facts concerning sexual assault, as well as complice_ag to double-check the performance of the rules. Thus, the requirements for sexual assault are not met, but the crime of robbery in the degree of tentativa and assault and battery with the degree of complice are met. The corresponding ranges would be from 6 months to 1 year after reducing the penalty for robbery by two degrees and from 3 to 6 months in the case of injury when reduced by one degree. Hence, the final range will be from 9 months to 1 year and 6 months.

We run the query with the variable d02 and the following output is displayed:

```
?- delito(d02, D), penaTotal(D, [A,B,C], [X,Y,Z]).
% QUERY:I would like to know if
     'delito' holds (for d02, and Var0) and
     The final sentence will range from Var1 years, Var2 months and
     Var3 days to Var4 years, Var5 months and Var6 days in prison.

        ANSWER: 1 (in 5579.467 ms)

JUSTIFICATION_TREE:
d02 has been accused of assault, because
    d02 has caused a mental damage, and
    a requirement is met in d02 victim's recovery, because
        the victim of d02 has needed a first medical care, and
        the victim of d02 has also needed medical treatment.
d02 has been accused of robbery, because
    d02 has taken possession of a movable thing from another person, and
    a requirement in d02 robbery is met, because
        the condition "objeto" holds, because
            d02 took it with the use of intimidation.
        and the condition "momento" holds, because
            they used it when taking the thing.
The final sentence will range from 0 years, 9 months and 0 days to 1 years,
6 months and 0 days in prison, and
The global constraints hold, because
    the global constraint number 1 holds.

MODEL:
{ lesion(d02,complice), menoscabo_mental(d02), requisito_lesion(d02),
asistencia_facultativa(d02), tratamiento_medico(d02), robo(d02,tentativa),
```

```
apoderacion_cosa(d02), objeto(d02), intimidacion_rob(d02), comision(d02) }
BINDINGS:
Var0 equal [lesion-complice,robo-tentativa]
Var1 equal 0
Var2 equal 9
Var3 equal 0
Var4 equal 1
Var5 equal 6
Var6 equal 0 ?
```

We follow the same instructions on the Playground and observe the results.

```
?- delito(d02, D), penaTotal(D, [A, B, C], [X, Y, Z]).

{ lesion(d02,complice), menoscabo_mental(d02), requisito_lesion(d02)
, asistencia_facultativa(d02), tratamiento_medico(d02), robo(d02,
tentativa), apoderacion_cosa(d02), objeto(d02), intimidacion_rob(d02)
, huida(d02) }
D equal [lesion-complice,robo-tentativa]
A equal 0
B equal 9
C equal 0
X equal 1
Y equal 6
Z equal 0 ?

yes
```

Figure 4.9: Playground results for d02

Justification: Expand All | +1 | -1 | Collapse All

- 'delito' holds (for d02, and [lesion-complice,robo-tentativa]), because
  - 'acusado' holds (for d02), and
    - d02 has been accused of assault, because
      - 'complice_les' holds (for d02), because
      - 'lesion_sec' holds (for d02), because
        - d02 has caused a mental damage, because
        - a requirement is met in d02 victim´s recovery, because
          - the victim of d02 has needed a first medical care, because
          - the victim of d02 has also needed medical treatment, because
    - d02 has been accused of robbery, because
      - 'tentativa_rob' holds (for d02), because
      - d02 has taken possession of a movable thing from another person, because
      - a requirement in d02 robbery is met, because
        - the condition "objeto" holds, because
          - d02 took it with the use of intimidation, because
        - and the condition "momento" holds, because
          - they used it to secure the escape from the crime scene, because
- The final sentence will range from 0 years, 9 months and 0 days to 1 years, 6 months and 0 days in prison, because
- The global constraints hold, because

Figure 4.10: Playground results for d02 (justification tree)

Chapter 5

Conclusions and future work

Throughout this project, the aim was to develop some kind of tool that could introduce some aspects of computer science into the legal field. The result has been a system that, through some input values that correspond to certain past events, shows us a list of the crimes committed and the range of years corresponding to their prison sentences, so that we can obtain an estimate of what will later be decided by the courts.

Through the Prolog-derived language s(CASP) and the Blawx tool we have implemented two interpretations for the same idea.

For the first interpretation, we developed a code that allows us, through a sh-compatible terminal, to perform queries and obtain the results in natural language. By using natural language for the answers, we provide an insight into what an implementation of the system might look like once it is made available to the non-technical user. The implementation has been carried out thanks to the s(CASP) interpreter, by which we obtain the justification tree, key to the result.

The second, although more specific, also results in an intuitive interface through which we can test certain input values and observe the results. We have been fortunate to have the help of the system's creator Jason Morris, who has always been willing to answer any questions and address any problems that may have arisen during the project. Blawx

is an interesting and useful interface that also provides a new way of modularizing the legislative code and facilitates the medium user to understand how it works and use it.

During the course of this report, we have pointed out certain aspects that could be developed in a future, more advanced implementation. The s(CASP) code could serve as the basis for a web tool intended to be used by the non-technical user. The input values could be selected from a list of events, as well as the penalty modifying variants corresponding to the degree of attempt and responsibility of the crime. For example, the different types of responsability could be displayed along with an explanation that would let the user know which one comes closest to what they are looking for. It also could be expanded with other crimes or new variants of the crimes already presented, with other forms of concurrence of several crimes in addition to the real competition and even modifications of the penalty in case there are mitigating or aggravating factors.

On the other hand, the implementation in Blawx could be further developed in the same way. The crimes of assault and battery and robbery would follow the model of sexual assault, and could be complemented with other crimes. Having implemented several offenses, it would be interesting to see if the current Blawx design would allow us to develop a test to obtain a list of the crimes committed and, being more ambitious, the final test to obtain the range of penalties. Once we are certain that this is not currently possible, if lucky, a future project could involve working with Jason Morris so that Blawx can expand its functionality according to our needs.

In conclusion, the whole development of this project has cost a lot of effort but it has turned out to be satisfying and really interesting. The justice field needs new tools to speed up bureaucratic processes, as well as updates to those already in use, and ideas like this could open up new and unexplored possibilities that could be very useful if these parties would like to consider them. Until then, we can offer the tools to non-professional users who want to have a first approximation of this world.

Bibliography

Abou El-Eid, T. (2022). **The future of Legal Technology: How will legal technology change the legal practice?**

Aletras, N., Tsarapatsanis, D., Preoţiuc-Pietro, D., and Lampos, V. (2016). **Predicting judicial decisions of the European court of human rights: a natural language processing perspective**. In: *PeerJ Comput. Sci*. DOI: https://doi.org/10.7717/peerj-cs.93.

Arias, J., Carro, M., Chen, Z., and Gupta, G. (2020). **Justifications for Goal-Directed Constraint Answer Set Programming**. In: *Electronic Proceedings in Theoretical Computer Science* 325. DOI: https://doi.org/10.48550/arXiv.2009.10238, pp. 59–72.

Arias, J., Carro, M., Salazar, E., Marple, K., and Gupta, G. (2018). **Constraint Answer Set Programming without Grounding**. In: *Theory and Practice of Logic Programming* 18(3–4). DOI: https://doi.org/10.1017/S1471068418000285, pp. 337–354.

Bueno, F., Cabeza, D., Carro, M., Hermenegildo, M., López, P., and Puebla, G. (2004). **The Ciao Prolog System: A Next Generation Multi-Paradigm Programming Environment**. Tech. rep. School of CS, Technical University of Madrid CS and ECE Departments, University of New Mexico.

Cubero, J. C. and Berzal, F. (2017). **Sistemas Inteligentes de Gestión: Tutorial de PROLOG**. Departamento de Ciencias de la Computación e I.A., Universidad de Granada.

IndeGranada (2019). **El Tribunal Supremo subraya que el consentimiento sexual debe ser expreso para que no sea delito**. In: *El Independiente De Granada*. Available at https://www.elindependientedegranada.es/ciudadania/tribunal-supremo-subraya-que-consentimiento-sexual-debe-ser-expreso-que-no-sea-delito.

Jones Pérez, K. and Yong Morales, G. (2007). **Introducción al lenguaje de programación lógica Prolog**. Available at http://www.di-mare.com/adolfo/cursos/2007-2/pp-Prolog.pdf.

Sacoto Romo, M. and Cordero Moscoso, J. (2021). **E-justicia en Ecuador: inclusión de las TIC en la administración de justicia**. In: *FORO: Revista de Derecho* 36. DOI: https://doi.org/10.32719/26312484.2021.36.5, pp. 91–110.

Soto Romero, M. (2019). **Manual de practicas para la asignatura de Programacion Declarativa**. PhD thesis. Universidad Nacional Autonoma de México.

Sulea, O. M., Zampieri, M., Malmasi, S., Vela, M., Dinu, L. P., and Van Genabith, J. (2017). **Exploring the use of text classification in the legal domain**.

Appendix A

Additional screenshots

A.1 Blawx code implementation

The definition of the rules in Blawx would be set out as follows:

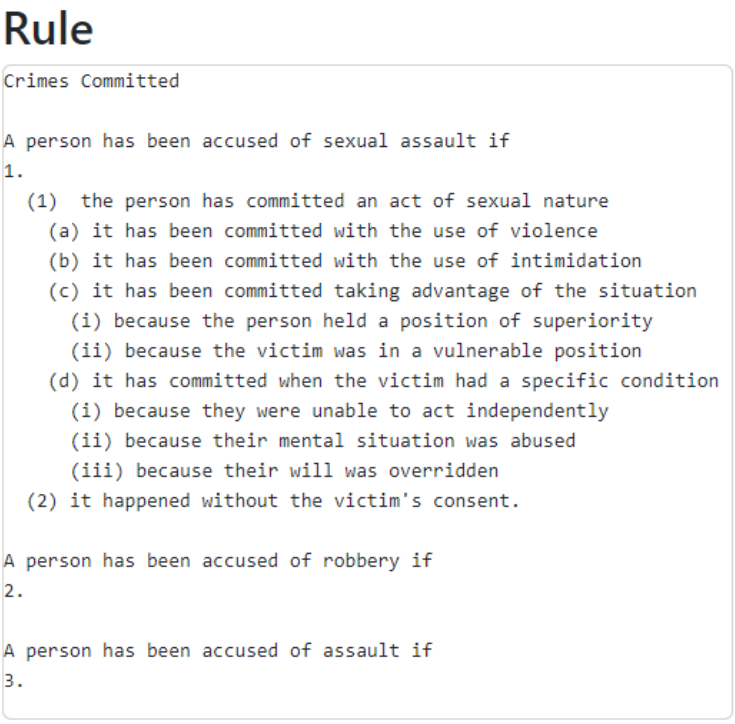

Figure A.1: Blawx rules as shown in the interface

The last three requirements for the crime of sexual assault developed in rule 1.1.d have been coded in a similar way as the previous rules in 1.1.c, since they are all based on a specific condition that the victim had at the time of the act, i.e., an attribute associated with an object of type person that corresponds to the victim's figure. The three attributes corresponding to each of the three rules respectively are `unable`, `mental_situation` and `will_overriden`.

Figure A.2: Sexual assault rules 1.1.d.i on Blawx

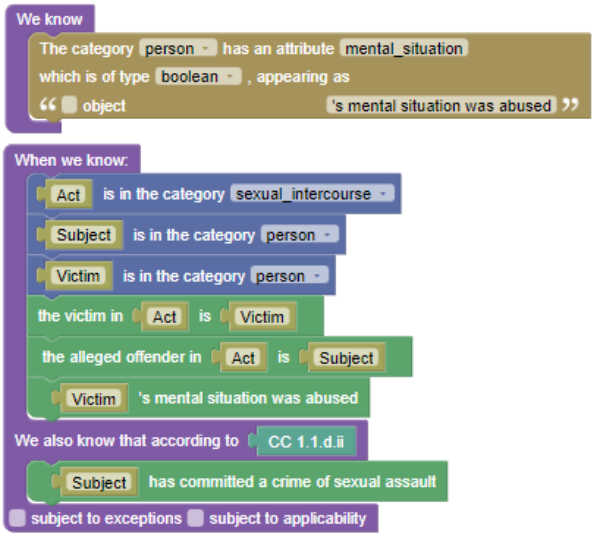

Figure A.3: Sexual assault rule 1.1.d.ii on Blawx

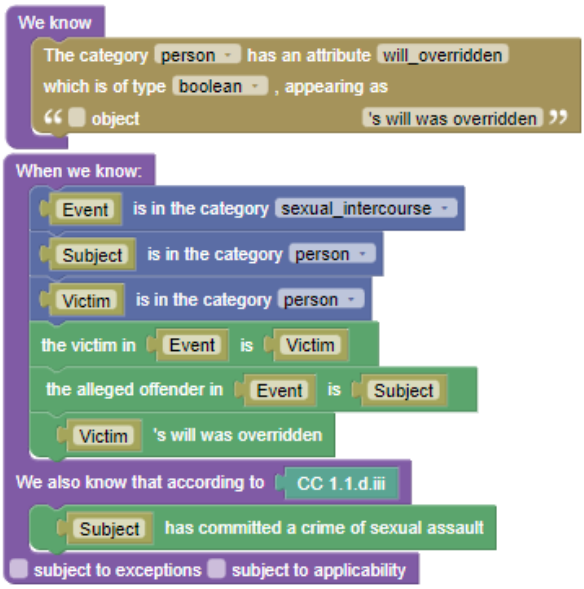

Figure A.4: Sexual assault rule 1.1.d.iii on Blawx

A.2 Rest of the examples results

In the example 4.2, the facts proven to be true were `acto_de_contenido_sexual`, `superioridad` and `autor_ag`. According to the laws explained, this should be considered sexual assault and our software should come to the same conclusion. In Blawx, we check what happens if we introduce the fact that `Paul` held a position a superiority. In both cases, the answers prove `a02` (`Paul`) as guilty of a crime of sexual assault.

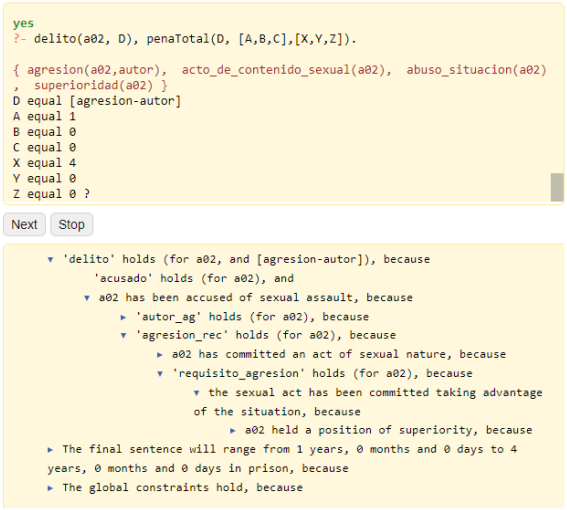

Figure A.5: Playground results of example 4.2

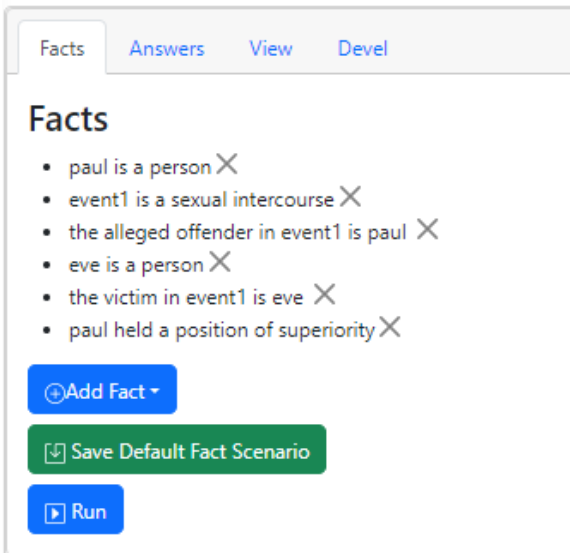

Figure A.6: Blawx set up for example 4.2

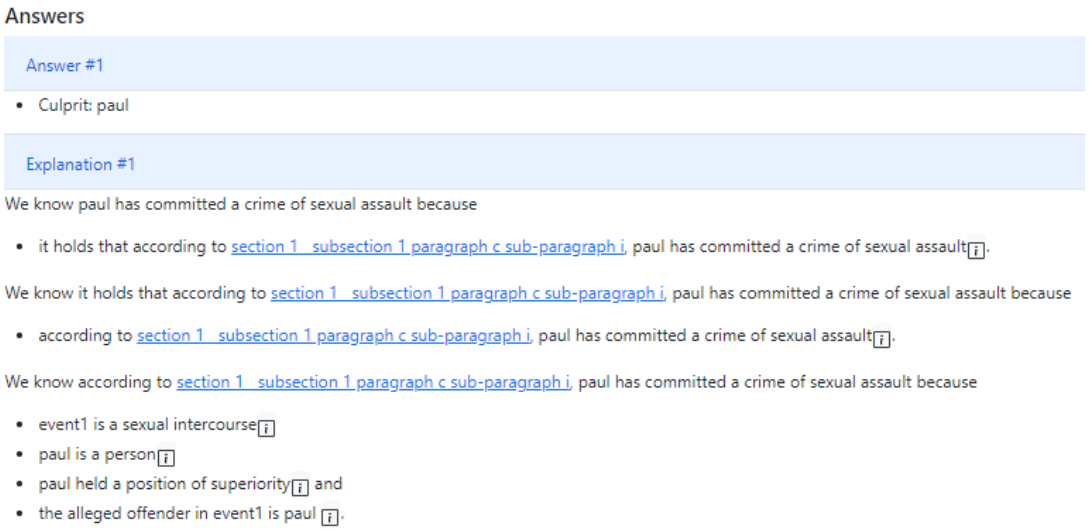

Figure A.7: Blawx results of example 4.2

Essentially, if there is case law in favor of the existence of consent in such case, then it should not be sexual assault, since it is considered to be a consensual sexual act. That is what happens in this case, where both the Playground and Blawx show no answers and so we can conclude the defendant (`Paul`) has not committed sexual assault.

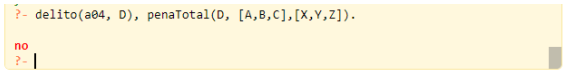

Figure A.8: Playground results of example 4.4

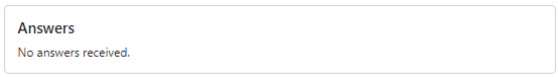

Figure A.9: Blawx results of example 4.4

In Example 4.5, according to this evidence, the accused would have committed the crimes of sexual assault with the corresponding degree of `autor` and robbery with the degree of `complice`. First, the terminal shows us the query we have performed with the new variables assigned, the number of answers found and the time it took to find them.

```
?- delito(d01, D), penaTotal(D, [A, B, C], [X, Y, Z]).
% QUERY:I would like to know if
     'delito' holds (for d01, and Var0) and
     The final sentence will range from Var1 years, Var2 months and Var3 days
 to Var4 years, Var5 months and Var6 days in prison.

        ANSWER: 1 (in 6926.212 ms)
```

Figure A.10: Terminal results for d01 (query)

Secondly, the justification tree is displayed.

```
JUSTIFICATION_TREE:
d01 has been accused of sexual assault, because
    d01 has committed an act of sexual nature, and
    the sexual act has been committed with the use of violence.
d01 has been accused of robbery, because
    d01 has taken possession of a movable thing from another person, and
    a requirement in d01 robbery is met, because
        the condition "objeto" holds, because
            d01 took it with the use of violence.
        and the condition "momento" holds, because
            they used it when taking the thing.
The final sentence will range from 2 years, 0 months and 0 days to 6 years, 0
 months and 0 days in prison, and
The global constraints hold, because
    the global constraint number 1 holds.
```

Figure A.11: Terminal results for d01 (justification tree)

And lastly, the model with the facts and logical rules that have been involved and the result values.

```
MODEL:
{ agresion(d01,autor),  acto_de_contenido_sexual(d01),  violencia_ag(d01),  r
obo(d01,complice),  apoderacion_cosa(d01),  objeto(d01),  violencia_rob(d01),
  comision(d01) }
BINDINGS:
Var0 equal [agresion-autor,robo-complice]
Var1 equal 2
Var2 equal 0
Var3 equal 0
Var4 equal 6
Var5 equal 0
Var6 equal 0 ?
```

Figure A.12: Terminal results for d01 (model and bindings)

In Example 4.6, the requirements for the crime of robbery in the degree of `tentativa` and assault and battery with the degree of `complice` are met. We run the query with the variable `d02` and the following output is displayed:

```
?- delito(d02, D), penaTotal(D, [A, B, C], [X, Y, Z]).
% QUERY:I would like to know if
     'delito' holds (for d02, and Var0) and
     The final sentence will range from Var1 years, Var2 months and Var3 days to Var4
 years, Var5 months and Var6 days in prison.

        ANSWER: 1 (in 7625.687 ms)

JUSTIFICATION_TREE:
d02 has been accused of assault, because
    d02 has caused a mental damage, and
    a requirement is met in d02 victim´s recovery, because
        the victim of d02 has needed a first medical care, and
        the victim of d02 has also needed medical treatment.
d02 has been accused of robbery, because
    d02 has taken possession of a movable thing from another person, and
    a requirement in d02 robbery is met, because
        the condition "objeto" holds, because
            d02 took it with the use of intimidation.
        and the condition "momento" holds, because
            they used it when taking the thing.
The final sentence will range from 0 years, 9 months and 0 days to 1 years, 6 months
and 0 days in prison, and
The global constraints hold, because
    the global constraint number 1 holds.

MODEL:
{ lesion(d02,complice),  menoscabo_mental(d02),  requisito_lesion(d02),  asistencia_f
acultativa(d02),  tratamiento_medico(d02),  robo(d02,tentativa),  apoderacion_cosa(d0
2),  objeto(d02),  intimidacion_rob(d02),  comision(d02) }
BINDINGS:
Var0 equal [lesion-complice,robo-tentativa]
Var1 equal 0
Var2 equal 9
Var3 equal 0
Var4 equal 1
Var5 equal 6
Var6 equal 0 ?
```

Figure A.13: Terminal results for d02

Appendix B

Encoding using s(SCASP)

B.1 `delitos.pl` file code

```
agresion(D,autor):-
    autor_ag(D),
    agresion_rec(D).
agresion(D,complice):-
    complice_ag(D),
    agresion_rec(D).
agresion(D,tentativa):-
    tentativa_ag(D),
    agresion_rec(D).

agresion_rec(D):-
    acto_de_contenido_sexual(D),
    requisito_agresion(D).

requisito_agresion(D):-
    violencia_ag(D).
requisito_agresion(D):-
    intimidacion_ag(D).

requisito_agresion(D):-
    abuso_situacion(D).
abuso_situacion(D):-
    superioridad(D).
abuso_situacion(D):-
    vulnerabilidad_victima(D).

requisito_agresion(D):-
    condicion_especifica_victima(D).
condicion_especifica_victima(D):-
    privacion_sentido(D).
```

```
31 condicion_especifica_victima(D):-
32     situacion_mental(D).
33 condicion_especifica_victima(D):-
34     anulacion_voluntad(D).
35
36 agresion_rec(D) :-
37     acto_de_contenido_sexual(D),
38     not consentimiento(D).
39
40
41 robo(D, autor):-
42     autor_rob(D),
43     apoderacion_cosa(D),
44     requisito_robo(D).
45 robo(D, complice):-
46     complice_rob(D),
47     apoderacion_cosa(D),
48     requisito_robo(D).
49 robo(D, tentativa):-
50     tentativa_rob(D),
51     apoderacion_cosa(D),
52     requisito_robo(D).
53
54 requisito_robo(D):-
55     objeto(D),
56     momento(D).
57 objeto(D):-
58     violencia_rob(D).
59 objeto(D):-
60     intimidacion_rob(D).
61 momento(D):-
62     comision(D).
63 momento(D):-
64     huida(D).
65
66
67 lesion(D, autor):-
68     autor_les(D),
69     lesion_sec(D).
70 lesion(D, complice):-
71     complice_les(D),
72     lesion_sec(D).
73 lesion(D, tentativa):-
74     tentativa_les(D),
75     lesion_sec(D).
76
77 lesion_sec(D):-
78     menoscabo_fisico(D),
79     requisito_lesion(D).
80 lesion_sec(D):-
81     menoscabo_mental(D),
82     requisito_lesion(D).
83
84 requisito_lesion(D):-
```

```
85      asistencia_facultativa(D),
86      tratamiento_medico(D).
87  requisito_lesion(D):-
88      asistencia_facultativa(D),
89      tratamiento_quirurgico(D).
90
91
92
93  %%Evidence definition
94
95  autor_ag(D):- evidence(D, autor_ag).
96  n_autor_ag(D):- -evidence(D, autor_ag).
97  autor_ag(D):- not evidence(D, autor_ag), not -evidence(D, autor_ag),
98      not n_autor_ag(D).
99  n_autor_ag(D):- not autor_ag(D).
100
101 autor_rob(D):- evidence(D, autor_rob).
102 n_autor_rob(D):- -evidence(D, autor_rob).
103 autor_rob(D):- not evidence(D, autor_rob), not -evidence(D, autor_rob),
104     not n_autor_rob(D).
105 n_autor_rob(D):- not autor_rob(D).
106
107 autor_les(D):- evidence(D, autor_les).
108 n_autor_les(D):- -evidence(D, autor_les).
109 autor_les(D):- not evidence(D, autor_les), not -evidence(D, autor_les),
110     not n_autor_les(D).
111 n_autor_les(D):- not autor_les(D).
112
113 complice_ag(D):- evidence(D, complice_ag).
114 n_complice_ag(D):- -evidence(D, complice_ag).
115 complice_ag(D):- not evidence(D, complice_ag),
116     not -evidence(D, complice_ag), not n_complice_ag(D).
117 n_complice_ag(D):- not complice_ag(D).
118
119 complice_rob(D):- evidence(D, complice_rob).
120 n_complice_rob(D):- -evidence(D, complice_rob).
121 complice_rob(D):- not evidence(D, complice_rob),
122     not -evidence(D, complice_rob), not n_complice_rob(D).
123 n_complice_rob(D):- not complice_rob(D).
124
125 complice_les(D):- evidence(D, complice_les).
126 n_complice_les(D):- -evidence(D, complice_les).
127 complice_les(D):- not evidence(D, complice_les),
128     not -evidence(D, complice_les), not n_complice_les(D).
129 n_complice_les(D):- not complice_les(D).
130
131 tentativa_ag(D):- evidence(D, tentativa_ag).
132 n_tentativa_ag(D):- -evidence(D, tentativa_ag).
133 tentativa_ag(D):- not evidence(D, tentativa_ag),
134     not -evidence(D, tentativa_ag), not n_tentativa_ag(D).
135 n_tentativa_ag(D):- not tentativa_ag(D).
136
137 tentativa_rob(D):- evidence(D, tentativa_rob).
138 n_tentativa_rob(D):- -evidence(D, tentativa_rob).
```

```
tentativa_rob(D):- not evidence(D, tentativa_rob),
    not -evidence(D, tentativa_rob), not n_tentativa_rob(D).
n_tentativa_rob(D):- not tentativa_rob(D).

tentativa_les(D):- evidence(D, tentativa_les).
n_tentativa_les(D):- -evidence(D, tentativa_les).
tentativa_les(D):- not evidence(D, tentativa_les),
    not -evidence(D, tentativa_les), not n_tentativa_les(D).
n_tentativa_les(D):- not tentativa_les(D).


acto_de_contenido_sexual(D):- evidence(D, acto_de_contenido_sexual).
n_acto_de_contenido_sexual(D):- -evidence(D, acto_de_contenido_sexual).
acto_de_contenido_sexual(D):- not evidence(D, acto_de_contenido_sexual),
    not -evidence(D, acto_de_contenido_sexual),
    not n_acto_de_contenido_sexual(D).
n_acto_de_contenido_sexual(D):- not acto_de_contenido_sexual(D).

violencia_ag(D):- evidence(D, violencia_ag).
n_violencia_ag(D):- -evidence(D, violencia_ag).
violencia_ag(D):- not evidence(D, violencia_ag),
    not -evidence(D, violencia_ag), not n_violencia_ag(D).
n_violencia_ag(D):- not violencia_ag(D).

intimidacion_ag(D):- evidence(D, intimidacion_ag).
n_intimidacion_ag(D):- -evidence(D, intimidacion_ag).
intimidacion_ag(D):- not evidence(D, intimidacion_ag),
    not -evidence(D, intimidacion_ag), not n_intimidacion_ag(D).
n_intimidacion_ag(D):- not intimidacion_ag(D).

superioridad(D):- evidence(D, superioridad).
n_superioridad(D):- -evidence(D, superioridad).
superioridad(D):- not evidence(D, superioridad),
    not -evidence(D, superioridad), not n_superioridad(D).
n_superioridad(D):- not superioridad(D).

vulnerabilidad_victima(D):- evidence(D, vulnerabilidad_victima).
n_vulnerabilidad_victima(D):- -evidence(D, vulnerabilidad_victima).
vulnerabilidad_victima(D):- not evidence(D, vulnerabilidad_victima),
    not -evidence(D, vulnerabilidad_victima),
    not n_vulnerabilidad_victima(D).
n_vulnerabilidad_victima(D):- not vulnerabilidad_victima(D).

privacion_sentido(D):- evidence(D, privacion_sentido).
n_privacion_sentido(D):- -evidence(D, privacion_sentido).
privacion_sentido(D):- not evidence(D, privacion_sentido),
    not -evidence(D, privacion_sentido), not n_privacion_sentido(D).
n_privacion_sentido(D):- not privacion_sentido(D).

situacion_mental(D):- evidence(D, situacion_mental).
n_situacion_mental(D):- -evidence(D, situacion_mental).
situacion_mental(D):- not evidence(D, situacion_mental),
    not -evidence(D, situacion_mental), not n_situacion_mental(D).
n_situacion_mental(D):- not situacion_mental(D).
```

```
193
194 anulacion_voluntad(D):- evidence(D, anulacion_voluntad).
195 n_anulacion_voluntad(D):- -evidence(D, anulacion_voluntad).
196 anulacion_voluntad(D):- not evidence(D, anulacion_voluntad),
197     not -evidence(D, anulacion_voluntad), not n_anulacion_voluntad(D).
198 n_anulacion_voluntad(D):- not anulacion_voluntad(D).
199
200 consentimiento(D):- evidence(D, jurisprudencia_confirma).
201 n_consentimiento(D):- -evidence(D, jurisprudencia_confirma).
202 consentimiento(D):- not evidence(D, jurisprudencia_confirma),
203     not -evidence(D, jurisprudencia_confirma), not n_consentimiento(D).
204 n_consentimiento(D):- not consentimiento(D).
205
206
207 apoderacion_cosa(D):- evidence(D, apoderacion_cosa).
208 n_apoderacion_cosa(D):- -evidence(D, apoderacion_cosa).
209 apoderacion_cosa(D):- not evidence(D, apoderacion_cosa),
210     not -evidence(D, apoderacion_cosa), not n_apoderacion_cosa(D).
211 n_apoderacion_cosa(D):- not apoderacion_cosa(D).
212
213 violencia_rob(D):- evidence(D, violencia_rob).
214 n_violencia_rob(D):- -evidence(D, violencia_rob).
215 violencia_rob(D):- not evidence(D, violencia_rob),
216     not -evidence(D, violencia_rob), not n_violencia_rob(D).
217 n_violencia_rob(D):- not violencia_rob(D).
218
219 intimidacion_rob(D):- evidence(D, intimidacion_rob).
220 n_intimidacion_rob(D):- -evidence(D, intimidacion_rob).
221 intimidacion_rob(D):- not evidence(D, intimidacion_rob),
222     not -evidence(D, intimidacion_rob), not n_intimidacion_rob(D).
223 n_intimidacion_rob(D):- not intimidacion_rob(D).
224
225 comision(D):- evidence(D, comision).
226 n_comision(D):- -evidence(D, comision).
227 comision(D):- not evidence(D, comision),
228     not -evidence(D, comision), not n_comision(D).
229 n_comision(D):- not comision(D).
230
231 huida(D):- evidence(D, huida).
232 n_huida(D):- -evidence(D, huida).
233 huida(D):- not evidence(D, huida),
234     not -evidence(D, huida), not n_huida(D).
235 n_huida(D):- not huida(D).
236
237
238 menoscabo_fisico(D):- evidence(D, menoscabo_fisico).
239 n_menoscabo_fisico(D):- -evidence(D, menoscabo_fisico).
240 menoscabo_fisico(D):- not evidence(D, menoscabo_fisico),
241     not -evidence(D, menoscabo_fisico), not n_menoscabo_fisico(D).
242 n_menoscabo_fisico(D):- not menoscabo_fisico(D).
243
244 menoscabo_mental(D):- evidence(D, menoscabo_mental).
245 n_menoscabo_mental(D):- -evidence(D, menoscabo_mental).
246 menoscabo_mental(D):- not evidence(D, menoscabo_mental),
```

```
247     not -evidence(D, menoscabo_mental), not n_menoscabo_mental(D).
248 n_menoscabo_mental(D):- not menoscabo_mental(D).
249
250 asistencia_facultativa(D):- evidence(D, asistencia_facultativa).
251 n_asistencia_facultativa(D):- -evidence(D, asistencia_facultativa).
252 asistencia_facultativa(D):- not evidence(D, asistencia_facultativa),
253     not -evidence(D, asistencia_facultativa),
254     not n_asistencia_facultativa(D).
255 n_asistencia_facultativa(D):- not asistencia_facultativa(D).
256
257 tratamiento_medico(D):- evidence(D, tratamiento_medico).
258 n_tratamiento_medico(D):- -evidence(D, tratamiento_medico).
259 tratamiento_medico(D):- not evidence(D, tratamiento_medico),
260     not -evidence(D, tratamiento_medico), not n_tratamiento_medico(D).
261 n_tratamiento_medico(D):- not tratamiento_medico(D).
262
263 tratamiento_quirurgico(D):- evidence(D, tratamiento_quirurgico).
264 n_tratamiento_quirurgico(D):- -evidence(D, tratamiento_quirurgico).
265 tratamiento_quirurgico(D):- not evidence(D, tratamiento_quirurgico),
266     not -evidence(D, tratamiento_quirurgico),
267     not n_tratamiento_quirurgico(D).
268 n_tratamiento_quirurgico(D):- not tratamiento_quirurgico(D).
269
270
271 %%Calculation of the sentence
272
273 delito(D, [agresion-X, robo-Y, lesion-Z]):-
274     acusado(D),
275     agresion(D,X),
276     robo(D,Y),
277     lesion(D,Z).
278 delito(D, [agresion-X, robo-Y]):-
279      acusado(D),
280      agresion(D,X),
281      robo(D,Y).
282 delito(D, [agresion-X, lesion-Y]):-
283     acusado(D),
284     agresion(D,X),
285     lesion(D,Y).
286 delito(D, [lesion-X, robo-Y]):-
287     acusado(D),
288     lesion(D,X),
289     robo(D,Y).
290 delito(D, [agresion-X]):-
291     acusado(D),
292     agresion(D,X).
293 delito(D, [robo-X]):-
294     acusado(D),
295     robo(D,X).
296 delito(D, [lesion-X]):-
297     acusado(D),
298     lesion(D,X).
299
300 pena(agresion, 1, 4).
```

```
301 pena(robo, 2, 5).
302 pena(lesion, 0.5, 3).
303
304 penaTotal([Delito-Grado|Resto], [MinA, MinM, MinD], [MaxA, MaxM, MaxD]):-
305      penaTotal_rec([Delito-Grado|Resto], MinProvisional, MaxProvisional),
306      maximo(MinProvisional, 0.5, Min),
307      minimo(MaxProvisional, 20, Max),
308      conversor(Min, MinA, MinM, MinD),
309      conversor(Max, MaxA, MaxM, MaxD).
310
311 penaTotal_rec([], 0, 0).
312
313 penaTotal_rec([DelitoR-GradoR|RestoR], MinimoR, MaximoR):- %concurso real
314     pena(DelitoR, MinC, MaxC),
315     modifica(GradoR, MinC, MaxC, MinMod, MaxMod),
316     penaTotal_rec(RestoR, MinResto, MaxResto),
317     MinimoR is MinMod + MinResto,
318     MaximoR is MaxMod + MaxResto.
319
320 modifica(autor, MinC, MaxC, MinC, MaxC).
321 modifica(complice, MinC, MaxC, MinMod, MaxMod):-
322     inf_enGrado(MinC, MaxC, MinMod, MaxMod).
323 modifica(tentativa, MinC, MaxC, MinMod, MaxMod):-
324     inf_enGrado(MinC, MaxC, MinTemp, MaxTemp),
325     inf_enGrado(MinTemp, MaxTemp, MinMod, MaxMod).
326
327 minimo(M,N,M):- M=<N.
328 minimo(M,N,N):- N<M.
329 maximo(M,N,M):- M>=N.
330 maximo(M,N,N):- N>M.
331
332 inf_enGrado(MinIn, MaxIn, MinFi, MinIn):-
333     MinFi is MinIn / 2.
334
335 conversor(N, A, M, D):-
336     A is floor(N),
337     Dif is N - A,
338     Meses is Dif * 12,
339     M is floor(Meses),
340     Dif2 is Meses - M,
341     Dias is Dif2 * 30.416,
342     D is round(Dias).
343
344 % ?- penaTotal([(agresion,0), (robo,1)], [Ai, Mi, Di], [Af, Mf, Df]).
345 % ?- delito(Def, Delitos), penaTotal(Delitos, [Ai, Mi, Di], [Af, Mf, Df]).
```

B.2 delitos.database.pl file code

```
acusado(d01). %sexual assault-autor + robbery-tentativa

evidence(d01, acto_de_contenido_sexual).
evidence(d01, violencia_ag).
-evidence(d01, intimidacion_ag).
-evidence(d01, superioridad).
-evidence(d01, vulnerabilidad_victima).
-evidence(d01, privacion_sentido).
-evidence(d01, situacion_mental).
-evidence(d01, anulacion_voluntad).
-evidence(d01, jurisprudencia_confirma).

evidence(d01, apoderacion_cosa).
evidence(d01, violencia_rob).
-evidence(d01, intimidacion_rob).
evidence(d01, comision).
-evidence(d01, huida).

-evidence(d01, asistencia_facultativa).
-evidence(d01, menoscabo_fisico).
-evidence(d01, menoscabo_mental).
-evidence(d01, asistencia_facultativa).
-evidence(d01, tratamiento_medico).
-evidence(d01, tratamiento_quirurgico).

evidence(d01, autor_ag).
-evidence(d01, complice_ag).
-evidence(d01, tentativa_ag).
-evidence(d01, autor_rob).
evidence(d01, complice_rob).
-evidence(d01, tentativa_rob).
-evidence(d01, autor_les).
-evidence(d01, complice_les).
-evidence(d01, tentativa_les).


acusado(d02). %robbery-tentativa + assault and battery-complice

evidence(d01, acto_de_contenido_sexual).
-evidence(d02, violencia_ag).
-evidence(d02, intimidacion_ag).
-evidence(d02, superioridad).
-evidence(d02, vulnerabilidad_victima).
-evidence(d02, privacion_sentido).
-evidence(d02, situacion_mental).
-evidence(d02, anulacion_voluntad).
evidence(d02, jurisprudencia_confirma).

evidence(d02, apoderacion_cosa).
-evidence(d02, violencia_rob).
evidence(d02, intimidacion_rob).
evidence(d02, comision).
-evidence(d02, huida).

```

```
-evidence(d02, menoscabo_fisico).
evidence(d02, menoscabo_mental).
evidence(d02, asistencia_facultativa).
evidence(d02, tratamiento_medico).
-evidence(d02, tratamiento_quirurgico).

-evidence(d02, autor_ag).
-evidence(d02, complice_ag).
-evidence(d02, tentativa_ag).
-evidence(d02, autor_rob).
-evidence(d02, complice_rob).
evidence(d02, tentativa_rob).
-evidence(d02, autor_les).
evidence(d02, complice_les).
-evidence(d02, tentativa_les).
```

B.3 delitos.pred.pl file code

```
# show agresion/2, acto_de_contenido_sexual/1, requisito/1, violencia_ag/1,
    intimidacion_ag/1, abuso_situacion/1, superioridad/1,
    vulnerabilidad_victima/1, condicion_especifica_victima/1,
    privacion_sentido/1, situacion_mental/1, anulacion_voluntad/1,
    not consentimiento/1.

# show robo/2, apoderacion_cosa/1, uso_fuerza/1, acceder/1,
    abandonar_lugar/1, objeto/1, violencia_rob/1, intimidacion_rob/1,
    comision/1, huida/1.

# show lesion/2, menoscabo_fisico/1, menoscabo_mental/1,
    requisito_lesion/1, asistencia_facultativa/1,
    tratamiento_medico/1, tratamiento_quirurgico/1.

#pred agresion(X, Y) :: '@(X) has been accused of sexual assault'.
#pred acto_de_contenido_sexual(X) ::
    '@(X) has committed an act of sexual nature'.
#pred requisito(X) :: 'a requirement is met'.
#pred violencia_ag(X) ::
    'the sexual act has been committed with the use of violence'.
#pred intimidacion_ag(X) ::
    'the sexual act has been committed with the use of intimidation'.
#pred abuso_situacion(X) ::
    'the sexual act has been committed taking advantage of the situation'.
#pred superioridad(X) :: '@(X) held a position of superiority'.
#pred vulnerabilidad_victima(X) ::
    'the victim was in a vulnerable position'.
#pred condicion_especifica_victima(X) ::
    'the victim had a specific condiction'.
#pred privacion_sentido(X) :: ' they were unable to act independently'.
```

```
31 #pred situacion_mental(X) :: 'their mental situation was abused'.
32 #pred anulacion_voluntad(X) :: 'their will was overridden'.
33 #pred consentimiento(X) :: 'it happened with the victim's consent'.
34
35 #pred robo(X,Y) :: '@(X) has been accused of robbery'.
36 #pred apoderacion_cosa(X) ::
37     '@(X) has taken possession of a movable thing from another person'.
38 #pred requisito_robo(X) :: 'a requirement in @(X) robbery is met'.
39 #pred objeto(X) :: 'the condition "objeto" holds'.
40 #pred violencia_rob(X) :: '@(X) took it with the use of violence'.
41 #pred intimidacion_rob(X) :: '@(X) took it with the use of intimidation'.
42 #pred momento(X) :: 'and the condition "momento" holds'.
43 #pred comision(X) :: 'they used it when taking the thing'.
44 #pred huida(X) :: 'they used it to secure the escape from the crime scene'.
45
46 #pred lesion(X, Y) :: '@(X) has been accused of assault'.
47 #pred menoscabo_fisico(X) :: '@(X) has caused a physical damage'.
48 #pred menoscabo_mental(X) :: '@(X) has caused a mental damage'.
49 #pred requisito_lesion(X) ::
50     'a requirement is met in @(X) victim`s recovery'.
51 #pred asistencia_facultativa(X) ::
52     'the victim of @(X) has needed a first medical care'.
53 #pred tratamiento_medico(X) ::
54     'the victim of @(X) has also needed medical treatment'.
55 #pred tratamiento_quirurgico(X) ::
56     'the victim of @(X) has also needed a surgical procedure'.
57
58 #pred penaTotal(Delitos, [MinA, MinM, MinD], [MaxA, MaxM, MaxD]) ::
59     'The final sentence will range from @(MinA) years, @(MinM) months
60     and @(MinD) days to @(MaxA) years, @(MaxM) months and @(MaxD)
61     days in prison'.
```