Informe de Práctica Profesional

Fundación Inria Chile

Juan-Pablo Silva Peña
15 de abril de 2018

Rut: 18.784.802-4
Carrera: Ingeniería Civil en Computación
Correo: jpsilva@dcc.cl
Celular: (+56) 9 56799536

Índice de Contenidos

1. Resumen

El trabajo fue realizado en la fundación Inria Chile, un centro de investigación y desarrollo creado por Inria, institución francesa de prestigio mundial en el área de las ciencias de la computación y matemáticas aplicadas. Este consistió en el marco del desarrollo de Mogand, que tiene por objetivo entrenar inteligencia ética y emocional para resolver conflictos de alta complejidad.

El trabajo del estudiante consistió en la búsqueda extensiva de bibliografía sobre generación automática de escenarios y narrativas consistentes, diseñar y proponer una solución que pudiera modelar los elementos del juego y considerar conflictos éticos y morales en las generaciones y finalmente realizar una prueba de concepto con una cantidad limitada de elementos del juego. Al término de la práctica se escribió un documento en inglés describiendo propuestas de cómo modelar el juego en consideración de los conflictos ético. Se presentó un diseño de solución que utiliza inteligencia artificial para la planificación (o selección) automáticas de elementos del juego para presentar desafíos morales y cómo interactúan los distintos módulos. Además se citó a los artículos científicos correspondientes que presentaban estas ideas y cómo se usaron para el diseño de la solución final. Adicional a esto, se implementó un prototipo que modela las interacciones entre elementos del juego y muestra una de las opciones para la generación automática.

En cuanto a los aprendizajes obtenidos, en el ámbito técnico se destaca el aprendizaje de algoritmos de inteligencia artificial, específicamente planeamiento automático, el uso de SPARQL que fue usado para crear prototipos de las relaciones entre entidades del juego, y la profundización del lenguaje de programación Python. También se participó en variadas actividades internas de la fundación, donde se aprendió cómo se llevan a cabo las lluvias de ideas para proyectos y cómo se presentan estas ideas demostrando seguridad a pesar de no estar involucrado directamente en el proyecto. Finalmente se destaca enormemente el aprendizaje de cómo leer artículos científicos de manera eficiente, y la práctica del idioma ingles, en el que todos los artículos científicos estaban escritos, y en el cual se escribió el documento dejado en la fundación.

2. Introducción

2.1. Descripción de la empresa

La práctica profesional se realizó en la función Inria Chile, fundación sin fines de lucro, y centro de transferencia tecnológica en el área de las Tecnologías de la Información y de la Comunicación en Chile. Lleva a cabo sus actividades trabajando estrechamente con grupos de investigación tanto en Chile como en Francia, siendo la Universidad de Chile uno de estos grupos. Se encuentra ubicado en Avenida Apoquindo 2827, piso 12, Santiago, Chile. El trabajo se realizó de forma presencial en las instalaciones de la fundación, entre el 3 y 28 de julio de 2017.

El contacto con la fundación comenzó al ver una oferta de práctica en el foro de U-cursos del departamento de computación [1] donde se describía un proyecto que involucraba inteligencia artificial y un videojuego educativo. El contacto fue hecho mediante el profesor Alexandre Bergel, que participó de asesor de investigación en el proyecto. Dentro de la empresa el estudiante fue supervisado por Stephen Sinclair (postdoc e investigador) en los aspectos de investigación, y por Cristián Paris (desarrollo de negocios) en los aspectos administrativos y relación con la empresa.

2.2. Motivación de la empresa

Al momento de comenzar la práctica, la fundación estaba pronto a comenzar la ejecución de un voucher CORFO de parte de Instituto Conexiones, que trataba de la generación automática de escenarios que presentaran conflictos éticos y morales para el videojuego educativo *Mogand* que estaban desarrollando. En este contexto la fundación vio que el proyecto requería de extensiva investigación acerca de literatura en el área, por lo que requerían de un practicante para llevar a cabo esta encuesta para que luego propusiera un diseño o modelo capaz de cumplir con los requisitos del cliente.

La situación previa a la llegada del practicante era nula. No había una situación previa, por lo que todo lo que el practicante propuso, diseño e implementó fue hecho desde cero, investigando literatura del área y conversando en conjunto con el supervisor de investigación.

2.3. Descripción del trabajo realizado

El trabajo realizado se puede dividir en 3 partes. La primera consistió en su mayoría en la búsqueda y lectura de papers, investigando distintos algoritmos, modelos y técnicas sobre generación automática de narrativas (*Automated narratives*), narración interactiva (*Interactive Storytelling*), modelación de jugadores (*Player modeling*). Adicionalmente se investigó en las áreas de generación de conflictos (*Conflict generation*), generación de dilemas (*Dilemma generation*) y narrativas coherentes (*Coherent narratives*), para el correcto diseño de la solución final. Esta búsqueda de bibliografía y técnicas se mantuvo durante toda la práctica, pero se concentró mayormente las primeras 2 semanas. La segunda parte de la práctica consistió en la implementación de un modelo de datos que permitiera la generación de escenarios considerando el factor ético y moral de cada uno de ellos. Aquí se utilizó Python para modelar las entidades y la lógica del programa, y SPARQL para hacer consultas al grafo generado, ya que la modelación se basó en grafos semánticos. La última parte consistió en el diseño de una solución (en papel, ya que no se alcanzó a implementar) basada en planificación automática, para luego escribir un documento que contuviera el proceso de pensamiento durante la práctica para llegar a esas soluciones, la documentación de las soluciones, la explicación y herramientas que se podían usar para la implementación de esta última solución, trabajo futuro, y un resumen de los papers más útiles para el desarrollo

[1] `https://www.u-cursos.cl/uchile/2008/0/COMCADCC/1/foro/o/22256775`

del proyecto. Durante la práctica se leyeron más de 60 artículos, de los cuales se seleccionaron 12 como los más relacionados al problema que se trataba de resolver.

3. Problema

Al comienzo de la práctica, la fundación comenzó su participación en un proyecto de parte de Instituto Conexiones, *Mogand*. *Mogand* es un videojuego educacional en el cual se avanza resolviendo dilemas con diferentes grados de dificultad, aplicando o no, inteligencia ética y emocional, presentando a los jugadores desafiantes situaciones éticas y morales en las cuales deberán tomar decisiones que impactarán en el futuro desarrollo del juego. Enfatizando esta última parte de presentar a los jugadores situaciones que impactan el futuro. El objetivo del juego es enseñar a los niños y jóvenes el peso de las decisiones que uno toma, y no juzgarlos por ello, sino entender la razón de su decisión. Para ello hay un sistema de puntos que cambian en base a lo que el jugador decide a lo largo del juego, de esta forma se crea un perfil del jugador.

Lo solicitado a Inria Chile fue que estas "situaciones desafiantes" se generaran automáticamente, ya que escribir una a una y ponerse en todos los casos en inviable. Además, estas situaciones deben depender de las acciones pasadas del jugador. Es decir, si el jugador decide dar o no agua a cierto personaje, el futuro de la historia o las acciones que el jugador podrá hacer en el futuro cambian.

El problema es cómo se construye un sistema que sea capaz de hacer esto. Primero es necesaria una forma de modelar las entidades de forma consistente para efectivamente ser capaz de generar estos escenarios. Luego es necesaria una forma de saber cómo seleccionar cada una de estos escenarios, ya que para cada punto en la historia esta podría fluir en muchas direcciones, por lo que saber qué camino tomar es un problema importante.

4. Objetivos

Como se explicó en la sección 3, el contexto en que se desenvuelve la práctica es en el de un videojuego educativo que busca presentar a los jugadores situaciones desafiantes ética y moralmente, para luego trabajar junto a ellos y entender por qué esa decisión fue tomada. Dentro del juego estas decisiones impactan el futuro de la historia, por lo que cada acción que el jugador tome puede afectar el futuro.

El objetivo de la práctica es la investigación de literatura relacionada con el tema de generación automática de escenarios y narrativas. Luego que estas narrativas contengan conflictos éticos y morales para que sean decisiones desafiantes para el jugador. Finalmente, sobre la generación de narrativas conflictivas se requiere que las narrativas sean consistentes con las acciones que el jugador ha tomado en el pasado, y por su puesto, que cada generación sea consistente con la generación pasada, es decir, que la historia tenga sentido. Cada uno de estos requerimientos tiene un desafío asociado no menor que se explicará en los siguiente párrafos, y la forma de solucionarlos en el desarrollo de este informe.

4.1. Generación de escenarios

La generación de escenarios tiene la dificultad de que tomando el universo de entidades dentro del juego, debe haber una relación entre todas para que sea posible armar un escenario a partir de ellas. Por ejemplo, todo escenario necesita de personajes y un lugar en el cual ocurrir, todo escenario puede pero no requiere que se levante una acción, así como también puede haber o no objetos con los que el jugador puede interactuar. Por su puesto, no todos los personajes pueden aparecer en todas las ubicaciones, a cada ubicación se puede llegar de cierta forma, pero no saltar al final del juego desde la primera escena. Para que los escenarios no sean repetitivos, también debe estar la posibilidad de que en una corrida del juego ciertos personajes no aparezcan en lugares que en otra corrida sí lo hagan.

4.2. Conflictos

Los escenarios generados deben poseen conflictos éticos y/o morales dentro de ellos, si no el jugador no se verá presentado frente a decisiones desafiantes que tomar. Qué se define como conflicto, cómo se genera, qué lo genera, qué tipos de conflictos hay, qué es un conflicto interesante, entre otras preguntas, son un problema abierto de investigación. Por esta razón el desafío cae en manos de intentar aplicar investigación reciente sobre el tema al contexto del juego, ya que gran mayoría de la literatura habla sobre conflictos/drama en novelas, no necesariamente eso se relaciona con un videojuego, que es interactivo.

4.3. Considerar acciones pasadas

En términos tradicionales, considerar acciones pasadas en un juego no requiere mayor esfuerzo que hacer un árbol o grafo con los pasos a seguir después de cierto “estado”. Sin embargo, en este caso no se puede aplicar esto de forma directa. La razón es que los escenarios pasados y futuros son auto-generados. Al ser auto-generados se debe mantener consistencia en las acciones pasadas y lo que se puede hacer ahora en el presente, sumándole que se deben incluir conflictos éticos relacionados con las acciones que ha tomado el jugador, para plantearle decisiones difíciles de tomar. La dificultad de realizar esta tarea es evidente.

5. Metodología

Como se explicó en la sección 2.3, la práctica se dividió en 3 partes. La primera parte permaneció activa durante toda la práctica, pero en mayor medida durante las primeras 2 semanas, y consistió en la búsqueda de bibliografía pertinente y de seleccionar e identificar papers que pudieran ser de utilidad para el trabajo. Durante el transcurso de la práctica se leyeron más de 60 artículos, de los cuales solo 12 fueron usados directamente en la solución, pero 40 fueron seleccionados como útiles, y aportaron algún grado de entendimiento o noción, o poseía algún referencia a algún otro artículo que sí ayudó en la construcción de la solución. Las áreas de investigación que más relación tienen con el proyecto son las siguientes: generación automática de narrativas (*Automated narratives*), narración interactiva (*Interactive Storytelling*), modelación de jugadores (*Player modeling*), generación de conflictos (*Conflict generation*), generación de dilemas (*Dilemma generation*) y narrativas coherentes (*Coherent narratives*). Donde en particular, el área de narración interactiva contiene un relación especial con lo que se quería lograr finalmente. Adicionalmente también se leyeron papers de sicología y teoría de la narración (en novelas), pero finalmente este enfoque fue descartado ya que se consideró que quienes debían proveer ejemplos de conflictos y el guión general eran los clientes, y que estábamos ampliando demasiado el área de búsqueda.

Luego de leer bibliografía y tener un mayor entendimiento del área se procedió a la modelación del mundo del juego. Aquí se realizaron 3 iteraciones verificando que efectivamente los prototipos cumplieran con el mundo del juego. El enfoque general que se usó fue el de usar un *banco de objetos* (*object bank*) para el almacenamiento y posterior selección de entidades en cada escenario. En la primera iteración se pensó crear un objeto escenario que contuviera todos los elementos del juego en un cierto momento. Si bien este enfoque cumplía con las necesidades de encapsulación y de relación entre estados del juego, la complejidad era muy alta, tenía muchos objetos anidados y el número de relaciones aumentaba exponencialmente por cada nueva entidad. Además que para mantener la coherencia del juego se pensó en llevar un contador o *id* que representara numéricamente en qué parte del juego uno se encontraba, lo cual dificultaba la auto-generación de los escenarios, ya que se debía seleccionar desde antes cuántos escenarios habría y esto implicaba además de considerar acciones pasadas, considerar principio y fin.
Una segunda iteración simplificó el modelo y generalizó el contenido de los objeto para que fuera más flexible, para ello se eliminaron ciertas relaciones del modelo anterior para luego proponer el uso del concepto de "personajes involucrados". Usando este concepto se logró una mayor eficiencia en la generación de escenarios y la introducción de personajes accionables o secundarios, refiriéndose a si el jugador puede o no interactuar con ellos. Finalmente la última iteración generalizó el modelo de la iteración 2 y pulió detalles para dejarlo funcional. Este prototipo de implementó y se hicieron pruebas para confirmar que efectivamente era posible generar nuevos escenarios aleatoriamente.

Con el prototipo para la generación aleatoria se tenía la posibilidad de crear nuevos escenarios a partir de distribuciones aleatorias de entidades y conflictos, pero esto no representa necesariamente el camino más interesante ni el con más conflictos. Para lograr esto se propuso modelar un usuario, creando un modelo interactivo basado en planificación automática, que considerara *traits*, personalidades, deseos y relaciones (familiares, amorosas, etc) entre personajes. Con estas características y un sistema de puntos se propone una solución que solo requiere un estado inicial para generar, basado en reglas, nuevos escenarios congruentes con los anteriores. El estado inicial puede ser generado por el prototipo anterior. Solo se necesita un banco de personajes, y sus características o *backgrounds*.

Finalmente, esta solución se evaluó replicando el *gameplay* que se nos fue mostrado por los clientes. Como el modelo es capaz de replicar lo mostrado, además de poder tomar otras rutas y puntajes dependiendo de las acciones del jugador, se consideró una propuesta prometedora. Este último algoritmo de planificación automática no fue implementado debido a limitaciones en el tiempo, sin embargo sí fue implementado posterior a la practica por el equipo en la fundación basándose en lo descrito en el documento entregado.

6. Solución

Siguiendo la metodología mostrada en la sección 5, se explicará en detalle qué herramientas y qué fue exactamente el resultado de la modelación, y cómo funciona el modelo de planificación automática.

6.1. Resultados de investigación

Como se explicó anteriormente, la práctica contuvo una gran componente de investigación ya que se partía desde cero, con solo ideas de parte del cliente para lo que se quería, y de gran complejidad. La revisión de decenas de artículos principalmente sobre el área de *Interactive Storytelling* y *Player Modeling* proveyó ideas y técnicas que se han usado en otras modelaciones. En particular fueron particularmente útiles las investigaciones de Heather Barber (junto con su tesis doctoral) acerca de narrativas adaptativas con dilemas, y las investigaciones de Nicolas Szilas, que ha publicado sobre cómo se considera una narrativa interesante, y cómo es posible generarla, junto a su software IDtension para crear elementos dramáticos de forma interactiva.

Durante la revisión de literatura se pensó en utilizar redes neuronales recurrentes (RNN) para modelar situaciones y lenguaje natural, a la vez que se utilizaban otros algoritmos de aprendizaje de máquinas (machine learning, ML), para que *aprendieran* cómo generar estos conflictos en situaciones particulares. Estas ideas fueron desechadas rápidamente porque para que funcionen se requiere de un gran conjunto de datos y ejemplos, lo cual considerando el corto tiempo de la práctica (1 mes) más el corto tiempo del proyecto voucher CORFO (6 meses) y el pequeño tamaño y presupuesto de los clientes, no se podrían generar.

Como resultado de la investigación se identificó que gran parte de las modelaciones para narrativas dinámicas usan un modelo de 3 o más capas, donde cada capa cumple un propósito en el algoritmo de generación. En particular, hay una capa como banco de objetos y otra como *game knowledge*, donde se crean las relaciones entre personajes. También gran parte de las investigaciones utilizan un sistema de reglas para que las historias sigan una cierta dirección, por más libres y auto-generadas que sean deben seguir una lógica y un fin congruente.

6.2. Prototipos de generación

Para lograr generar escenarios consistentes primero es necesario modelar las entidades y establecer relaciones entre ellas. En total se hicieron 3 iteraciones, cada una mejorando sobre la solución anterior.

6.2.1. Prototipos iniciales

Para la primera modelación de las entidades del juego se buscó que los escenarios estuvieran autocontenidos, con objetos dentro de este representando el estado actual del juego más los dilemas, acciones y conflictos presentes en él. Un ejemplo de las relaciones posibles en este modelo es: partiendo desde una ubicación específica, pueden aparecer ciertos personajes y ciertos objetos. Luego dependiendo de los personajes puedo ejecutar acciones sobre ellos, y al hacerlo estaría creando un personaje "directo", significando con el que se interactúa directamente. Luego este define el problema y el problema también participa en la selección de los objetos en el nivel. Como se puede ver esto es bastante complejo y las dependencias no son fáciles de conseguir, ya que se debe hacer un banco de entidades diciendo qué se puede relacionar con qué.

En la segunda iteración se redujo el número de objetos interiores de 3 a 1, simplificando las dependencias eliminando objetos redundantes y reemplazando el objeto "acciones" por personajes participantes o pasivos en interacciones. Sin embargo este modelo sigue teniendo un problema, el cual es que requiere el estado del juego dentro del mismo escenario. Este estado de juego se refiere a un valor numérico indicando en qué parte del

juego el jugador se encuentra actualmente. En esta iteración también se dejó como parámetro individual el propósito del escenario, lo que debe lograr el jugador, el *goal*. Esto es un error en términos de consistencia del juego, ya que lo que se tiene como meta es algo de lo cual otras variables pueden depender lógicamente, como la ubicación. Solucionando estos problemas se diseñó la última iteración.

6.2.2. Solución generativa

La iteración final de la modelación para las entidades del juego propone solo 1 objeto interno al igual que la iteración 2. Sin embargo la principal diferencia de esta solución final es que los roles de qué depende de qué se invierten. Por ejemplo, antes los objetos y los personajes dependían del tipo de conflicto que se presentaría en el escenario, ahora es al revés, es el conflicto el que depende de qué personajes y objetos se encuentren en el escenario. La razón es que ahora es posible tener tipos de conflictos que pueden relacionarse con cierto tipo de personajes y no restringir la generación de personajes, sino de conflictos. El otro cambio significativo fue remover *GameStatus* del objeto escenario, cambiar su nombre a *PlayerStatus* y hacer que el mismo objeto escenario dependa de este. La razón fue que como el jugador tiene puntajes a lo largo del juego, se pueden usar estos puntos para seleccionar qué escenario usar, luego el *goal* actúa como delimitante sobre en qué punto del juego el jugador se encuentra. El diagrama de estas relaciones comparado con la primera iteración se encuentra en el apéndice A.

6.2.3. Implementación

Como prototipo de generación automática se implementó la solución propuesta en la última iteración para la modelación de entidades. Inicialmente se pensó utilizar tablas estructuradas, en formato *csv*, pero la cantidad de tablas redundantes y la mantención para dejarlas ordenadas y en una forma normal aceptable no era muy práctico. Siempre está la posibilidad de usar árboles como *json* o *xml*, pero por la naturaleza semántica de las relaciones se decidió experimentar un poco y utilizar *RDF*, que es un formato de metadata para grafos en la web semántica. Adicionalmente se pueden hacer consultas directas usando SPARQL, que es un lenguaje para consultas semánticas en grafos, igual que como SQL es para tablas estructuradas. La implementación consideró python como el lenguaje base, donde se usó un paradigma orientado a objetos, ya que este funciona bastante bien con estructuras en juegos y ayuda a encapsular funcionalidades. Un pequeño extracto del código para la generación aleatoria de conflictos se puede encontrar en Código 1. Finalmente se hizo un pequeño script que cargara el grafo de interacciones, generara todas las posibles combinaciones, luego se seleccionaran algunas al azar y se guardaran en los objetos creados. Luego comienza el juego simplificado, el script permite interacción donde se le presenta al jugador una situación y una pregunta o interacción que puede realizar. El jugador tiene 3 opciones y luego de actuar se le pregunta si quiere moverse a otro escenario. Un ejemplo del output para este programa se puede encontrar en el apéndice B. Este juego no tiene lógica ni historia, sino más bien fue creado para la modelación de entidades y generación de escenarios que luego pueda ser usado como base para el modelo interactivo que se presentará a continuación.

Código 1: Generación aleatoria de conflictos a partir de objetos y personajes presentes en el juego.

```
1 # Calculates all combinations of characters-items
2 def make_conflicts(self, ask, char_act):
3     c_list = list()
4     i_list = list()
5     for lc in range(len(self.in_characters) + 1):
6         for cc in itertools.combinations(self.in_characters, lc):
7             c_list.append(list(cc))
8
9     for li in range(len(self.in_items) + 1):
```

```
10          for ii in itertools.combinations(self.in_items, li):
11              i_list.append(list(ii))
12
13      # Creates a conflict for each of the combinations.
14      for _ask in ask:
15          for comb in itertools.product(c_list, i_list):
16              self.conflicts.append(Conflict(_ask, comb[0] + [char_act], comb[1]))
```

6.3. Modelo de planificación automática

El modelo de generación automática es capaz de generar diversos escenarios de forma aleatoria, pero para darle un sentido lógico a los hechos y poder crear un juego auto-generativo se requiere de otro modelo. Aquí se usó todo lo investigado, y aunque no se alcanzara a implementar, se propusieron herramientas y reglas (que debían ser conversadas con los clientes) para las variaciones autónomas del juego en respuesta a decisiones del jugador. En particular se tomó especial cuidado en que las acciones que el jugador tome puedan afectar el futuro desarrollo de la historia. Aquí se tomó como ejemplo una pequeña muestra del juego que los clientes nos habían facilitado para recrearla y mostrar cómo las acciones del jugador afectan su jugabilidad, particularmente qué opciones se le presentan en respuesta a lo que hizo anteriormente.

La solución propuesta utiliza técnicas de planificación automática. Lo que esto significa es tener un set de reglas que determinen el estado inicial del juego, luego tener reglas de relaciones y acciones, tales que estas reglas sean capaces de moldear el estado actual a uno siguiente. Cada estado posee una pre y post condición que permite cambiar el estado del juego a otro, solo si las precondiciones se cumplen. Un ejemplo muy simple sería que si estoy en la ciudad puedo ir al río y viceversa, pero si estoy en el volcán esto no es posible. El cumplimiento de estas condiciones se plantean como un problema SAT, es decir, un problema de satisfacibilidad booleana. Este problema es NP-completo, por lo que es complejo computacionalmente de resolver. Sin embargo, es importante notar que es realmente necesario chequear todas las condiciones y ver todas las posibles rutas que el jugador puede tomar para efectivamente crear narrativas auto-generadas consistentes y que dependan de las acciones pasadas. En la bibliografía acerca de *Interactive Storytelling* se usa bastante este enfoque, y existen herramientas y lenguajes de programación que permiten implementar estas condiciones fácilmente.

Se definió que cada personaje tiene su propia personalidad, deseos y relaciones con otros personajes. Además se define el concepto de propiedades para cada personaje, que pueden cambiar a lo largo del juego. Cada personaje conoce cosas y está enterado sobre ciertas cosas que ocurren, luego el personaje actúa o se comporta coherentemente respecto a esto. Por ejemplo en el caso de que el jugador mate a otro personaje, los personajes que sepan esto no querrán interactuar con él ni le pedirán favores, a menos que el otro personaje también sea una "mala persona". Lo mismo ocurre con relaciones amorosas: si Ben y Mary se encuentran en una relación, el deseo de uno de los dos se vuelve el deseo de ambos.

Todas estas ideas se llevan a cabo usando reglas lógicas, por ejemplo se tiene una función `at(Personaje, Lugar)` que es verdadera solo si Personaje se encuentra en Lugar. Una acción se representa con una función de la misma manera, por ejemplo la función `move(Personaje, Lugar1, Lugar2)`. Esta función se refiere a que un personaje se mueve de un lugar a otro, pero requiere pre y post condiciones. Primero se requiere que `at(Personaje, Lugar1)` y que los lugares sean contiguos en el mapa del juego, la postcondición es `!at(Personaje, Lugar1)` $\wedge$ `at(Personaje, Lugar2)`. Y de esta forma muchas otras reglas y funciones, como `request(·)` que muestra la interacción de 2 personajes, pero como precondición para hacerlo ambos deben estar en el mismo lugar. Todas estas reglas e ideas fueron diseñadas y concebidas por el practicante basandose en investigación en el área.

Este tipo de reglas se pueden verificar con herramientas como STRIPS[2], PDDL[3], *Java Theorem Prover*, y lenguajes orientados a acciones como GOLOG[4], o lenguajes basados en reglas como CLIPS[5]. Todos estos enfoques se han usado en la literatura con cierto nivel de éxito. Posterior a la práctica el proyecto continuó su desarrollo usando CLIPS. Un ejemplo del output que debería generar este programa se encuentra en el apéndice C. Aquí se muestra el estado inicial del juego y los personajes presentes, junto a sus propiedades, dónde están, sus deseos y gustos, entre otras. Luego se muestra la ejecución del juego donde cada *step* corresponde a una acción del jugador. Los resultados o consecuencias de las acciones se referencia con →, las opciones presentadas al jugador con un + y entre paréntesis junto a cada opción la razón de por qué esa opción está presente.

6.4. Documento final

Al final de la práctica se escribió un documento en ingles en formato de paper explicando muy similar al presente informe la metodología usada para llegar a la solución, los distintos modelos generados, sus ventajas, y para el caso del modelo de planificación automática, los detalles pertinentes a la lógica, reglas, funciones, acciones y técnicas o herramientas que permitieran implementar este modelo fácilmente en el futuro, lo cual efectivamente se hizo siguiendo las propuestas dejadas en el documento. Dentro del mismo documento se hizo un resumen de cada paper usado directamente en la solución, que se traduce a hacer un *survey* o encuesta exhaustiva sobre lo que existe actualmente en la literatura.

[2]Stanford Research Institute Problem Solver.
[3]Planning Domain Definition Language, un intento para estandarizar los lenguajes de planificación con inteligencia artificial.
[4]lenguaje de programación lógico para dominios dinámicos.
[5]C Language Integrated Production System, usado en *expert systems*.

7. Discusión y reflexión

Durante la práctica hubo numerosos obstáculos que presentaron un desafío para el practicante. Partiendo por el diseño total de las soluciones y la lectura exhaustiva de artículos científicos, el practicante debió enfrentar la imposibilidad de verificar con el cliente las soluciones parciales.

7.1. Investigación

La fuerte componente de investigación en la práctica representa un fuerte desafío para cualquier persona. En particular si no se quiere solo un resultado teórico, sino más bien funcional y aplicado a un juego. Sumando que el juego está en desarrollo, sin mucho contenido ni información que se pueda usar como base. Por lo mismo, la curva de entrada al problema fue muy alta. No estaba claro qué realmente debíamos buscar, ni en qué exactamente debíamos concentrarnos para crear un prototipo. Así se comenzó a buscar sobre inteligencia artificial en los juegos, luego el foco cambio a narrativas interactivas y modelamiento de jugadores, pero antes de esto también se estaban revisando papers sobre teoría de narración, viendo qué hacia interesante o conflictiva una obra, y por el mismo camino revisando literatura del área sicológica acerca de qué constituye un conflicto o una choque de ideas.

7.2. Poco contacto

La practica se desarrolló presencial en las oficinas de Inria Chile y se tenían reuniones periódicas con el supervisor de investigación, pero ambos estábamos un poco frustrados acerca del compromiso por parte de los clientes, ya que no hubo una sola reunión con ellos durante toda la práctica. Hubo una previa, primera vez para los miembros de Inria también, y una posterior para presentar avances. Pero durante la práctica se tuvo que trabajar bajo supuestos y tratar de crear modelos generales que pusieran moldearse fácilmente a lo que necesitaría eventualmente. Esto incrementa la dificultad ya que no había cómo confirmar que estuviéramos yendo por el camino correcto o si era efectivamente esto lo que querían.

7.3. Inteligencia artificial (IA)

Hubo varios cambios en los enfoques por los cuales se trataba de abordar el problema. Particularmente se sabía que se necesitaría un sistema inteligente que pudiera seleccionar los escenarios y crear nuevos a partir de las interacciones del jugador, pero previo a la práctica el practicante no poseía estudios sobre IA, por lo que resultó todo un desafío. También dentro de la misma área de la IA se pensó usar *machine learning* para crear un modelo capaz de aprender los cambios que el cliente quería, pero nuevamente debido al poco contacto y recursos (en términos de ejemplos de escenarios) se tuvo que ver otras posibilidades que no fueran realmente soluciones adaptativas o que *"aprendieran"*, así se llego finalmente a la planificación automática.

7.4. Reflexión general

La práctica realizada fue una oportunidad increíble para el aprendizaje de nuevas técnicas y áreas de investigación, incrementando al nivel de seguridad de que se quiere seguir un camino en la academia. Muchos de las dificultades habrían desaparecido de estar el cliente presente durante el desarrollo de prototipos y diseños, pero de todas maneras esto sirvió de aprendizaje para ver siempre las cosas de manera general de modo que luego puedan ser aplicadas en un rango mayor de posibilidades. En el caso de hacer nuevamente esta práctica no cambiaría nada, el aprendizaje obtenido y las relaciones formadas cumplen una mayor importancia que las

dificultades y obstáculos en el camino. Tal vez una ayuda durante la investigación habría permitido mayor seguridad las primeras semanas de investigación, o consejos sobre cómo descubrir pronto si un paper servirá.

8. Conclusión

Los resultados del trabajo presentado en este informe fueron considerados satisfactorios, valuables y de gran ayuda para el futuro del proyecto en su momento. Luego de terminada la práctica, las ideas y modelos propuestos fueron implementados siguiendo las reglas que se plantearon en el documento que se dejó en Inria. Además, los papers recopilados y resumidos fueron usados como base para un entregable frente a CORFO mostrando lo existente en la literatura sobre generación automática de narrativas y narrativas interactivas. Por lo tanto, los objetivos técnicos del proyecto se consideran cumplidos y los supervisores manifestaron que estaban muy contentos con los resultados y el trabajo realizado.

En términos de aprendizaje, se destaca la lectura de papers. Si bien el conocimiento adquirido leyéndolos es importante, se considera que aprender a leer documentos técnicos científicos ayuda mucho para el futuro, especialmente con el deseo de seguir el camino de la academia. El aprender a distinguir material que será útil rápidamente, aprender a entender documentos complejos y aprender a buscar material relacionado es uno de los aprendizajes que más se destacan. Otro aprendizaje que se destaca es la introducción al mundo de inteligencia artificial, lo que luego de la práctica se transformó en un profundo interés por el aprendizaje de máquinas y las redes neuronales. También se aprendió que en la industria no todo funciona siempre, por más que uno tenga un equipo destacado, buenos recursos y buena disposición, si el cliente desaparece o no está comprometido con el proyecto (a pesar de que es él el interesado), es muy difícil trabajar y crear una solución que se adecúe a sus necesidades.

En conclusión, me gustó mucho mi práctica. Permitió que aprendiera muchas cosas y me relacionara con gente muy capaz, muy inteligente que ha trabajado en proyectos muy interesantes en el extranjero. Aprendí a leer papers de manera más fluida y practiqué mi inglés escribiendo un documento en este formato que quedó en Inria para usar como referencia en la continuación del proyecto. También esta práctica fue mi punto de partida para comenzar a trabajar en investigación usando inteligencia artificial.

A. Diagrama de relaciones para el modelo generativo

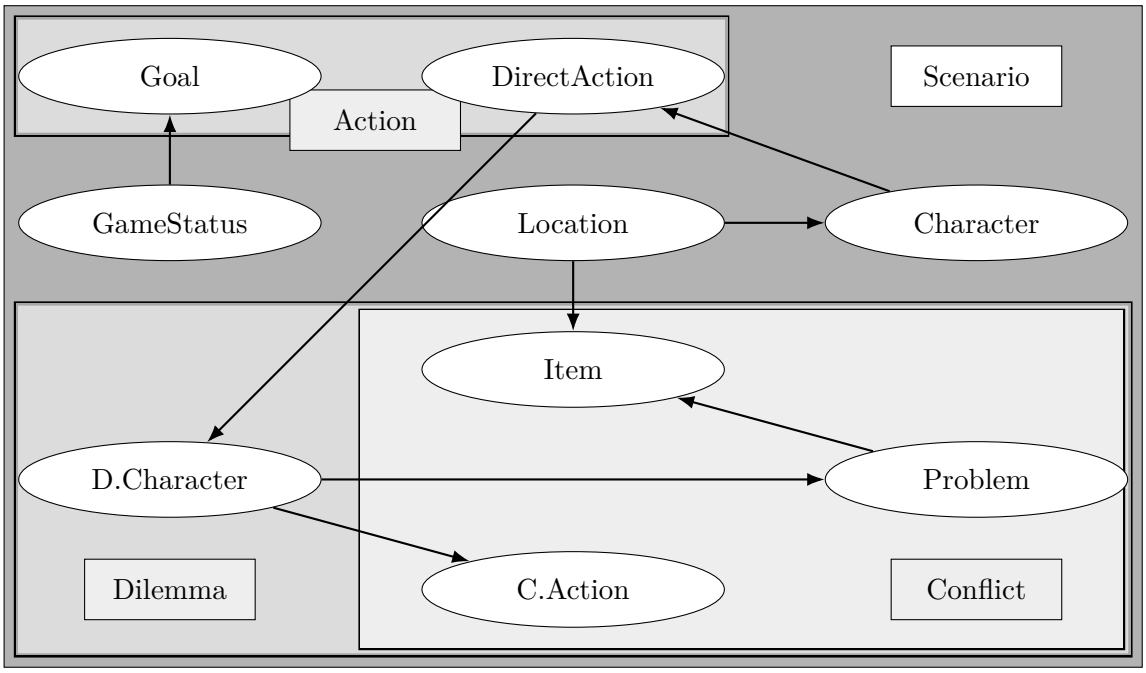

Figura A.1: Diagrama mostrando el primer diseño para el prototipo de la generación aleatoria.

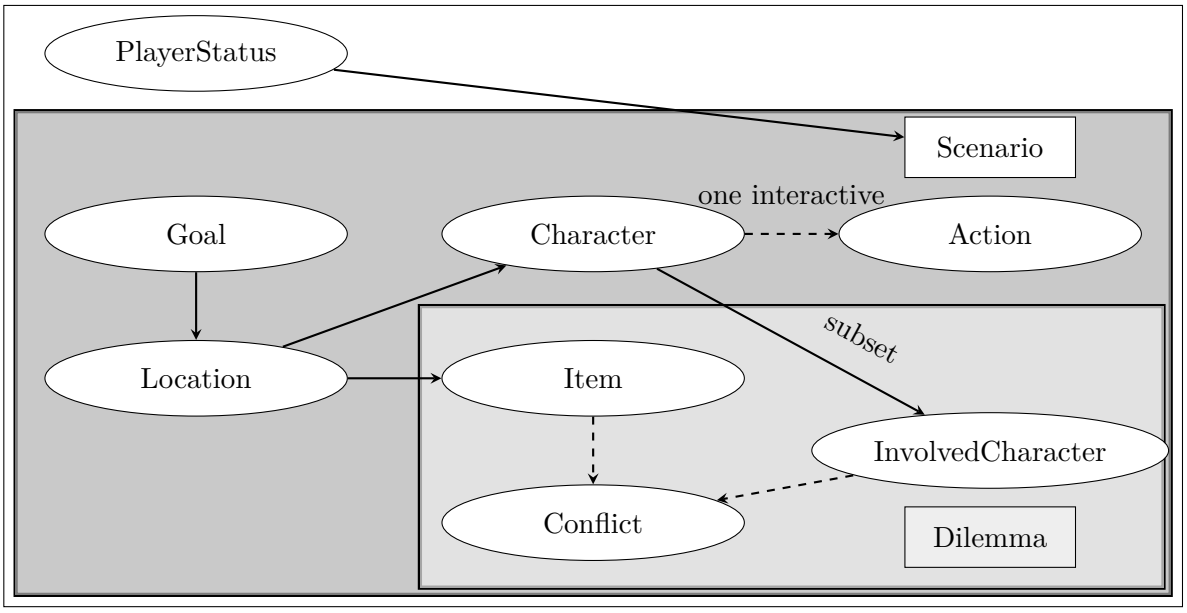

Figura A.2: Diagrama mostrando el diseño final utilizado para el prototipo de la generación aleatoria y almacenamiento de relaciones en el juego.

B. Ejemplo modelo generativo

```
Basic game play.
The player has [0, 0, 0, 0] points at the moment.
The current goal is CrossGate and the player is currently at Gate.
The following characters are currently on the map Kid, BadMan, so you decide to speak to Kid.
Kid begs you to help. He needs Gun for: BadMan, Kid
1- Ignore
2- Listen, say you can't and leave
3- Help
Choose an option:
>3

Your current points are [20, 3, -10, 0]
Choose next location (1), or be it random (2):
>2

The player has [20, 3, -10, 0] points at the moment.
The current goal is CrossGate and the player is currently at Gate.
The following characters are currently on the map Kid, BadMan, so you decide to speak to Kid.
Kid asks you for help. He needs Chocolate, Gun for: self
1- Ignore
2- Listen, say you can't and leave
3- Help
Choose an option:
>1

Your current points are [15, -7, 0, 0]
Choose next location (1), or be it random (2):
>1
Type the location you want to go:
>City

The player has [15, -7, 0, 0] points at the moment.
The current goal is CrossGate and the player is currently at City.
The following characters are currently on the map Kid, so you decide to speak to Kid.
Kid begs you to help. He needs Water for: self
1- Ignore
2- Listen, say you can't and leave
3- Help
Choose an option:
>exit
Game Over.
```

Código 2: Ejemplo del output del modelo generativo para escenarios.

C. Ejemplo modelo interactivo

Código 3: Ejemplo de simulación usando el modelo interactivo. Esta simulación muestra cómo las acciones del jugador afectan su jugabilidad. En el paso 7 el jugador solo tiene una opción para elegir debido a las acciones pasadas que tomó, por lo que solo puede aceptar la petición de la madre, para bien o para mal.
La simulación considera que el jugador le da agua a un anciano, luego habla con un guardia que le pide que le traiga los niños de una madre que se encontraba en el lugar. Basado en las acciones y gustos del personaje, al hablar con la madre y que esta le pidiera un favor, el jugador no puede negarse, ya sea para ayudarla o para llevarle sus hijos al guardia.

```
Initial state:
   Player:
      at(Player, Initial)                  wish(Player, CrossGate)
      holds(Player, Water)                 property(young)
      likes(Player, woman)                 property(man)
      !likes(Player, kid)                  property(poor)
      !likes(Player, officer)              property(weak)
      property(Player, generous=1)         property(Player, strategic=1)
      property(Player, easy-going=1)       property(Player, moral=1)

   OldMan:
      at(OldMan, DumpYard)           property(man)
      holds(OldMan, password)        property(old)
      wish(OldMan, water)            property(poor)

   GateGuard:
      at(GateGuard, FrontGate)                    property(man)
      knows(GateGuard, at(Child1, FrontGate))     property(strong)
      knows(GateGuard, at(Child2, FrontGate))     property(officer)
      knows(GateGuard, at(Mother, FrontGate))     property(rich)
      charAction(GateGuard, Request(genopass))
      wish(GateGuard, Child1)
      wish(GateGuard, Child2)

   Mother:
      at(Mother, FrontGate)                           property(woman)
      wish([Mother, Child1, Child2], CrossGate)       property(poor)
      loves(Mother, Child1)                           property(weak)
      loves(Mother, Child2)
      relation(Mother, motherOf -> Child1)
      relation(Mother, motherOf -> Child2)
      knows(Mother, at(GateGuard, FrontGate))
      knows(Mother, at(Child1, FrontGate))
      knows(Mother, at(Child2, FrontGate))
      knows(Mother, at(strongBox, BackYard))

   Child1:
      at(Child1, FrontGate)                       property(woman)
      wish(Child1, at(Child1, where(Mother)))     property(kid)
      loves(Child1, Mother)                       property(sick)
```

```
41      relations(Child1, childOf -> Mother)        property(weak)
42
43
44    Child2:
45       at(Child2, FrontGate)                      property(man)
46       wish(Child2, at(Child2, where(Mother)))    property(kid)
47       loves(Child2, Mother)                      property(weak)
48       relations(Child2, childOf -> Mother)
49
50 Step 1: Player moves to Dump Yard
51    Player: MOVE(Player, Initial, DumpYard)
52            -> at(Player, DumpYard) and !at(Player, Initial)
53
54 Step 2: Player speaks to the old man and chooses to give him water
55    Player: SPEAK(Player, OldMan)
56            -> charAction(OldMan) (hasWish(OldMan))
57    OldMan: Request(OldMan, Player, water)
58    Options:
59             +give water    (is(Player, generous))
60             +give half     (is(Player, generous) and is(Player, strategic))
61             +don't give    (is(Player, strategic))
62
63 Step 3: Player gives water to the old man, completing his wish so he gives password to the
      player
64    Player: Give(Player, OldMan, water)
65            ->  !holds(Player, water) and knows(OldMan, is(Player, generous))
66                and pointsChange(generous+1=2)
67                and pointsChange(strategic-1=0)
68                    -> !property(Player, strategic)
69    OldMan: accomplish(OldMan, wish(OldMan, water))
70            -> Give(OldMan, Player, password)
71                    -> holds(Player, password) and !holds(OldMan, password)
72
73 Step 4: Player moves to the Front Gate
74    Player: MOVE(DumpYard, FrontGate)
75            -> at(Player, FrontGate) and !at(Player, DumpYard)
76
77 Step 5: Player speaks to the Gate Guard and chooses `I don't have it'
78    Player: SPEAK(Player, GateGuard)
79            -> charAction(GateGuard)
80    GateGuard: Request(GateGuard, Player, genopass) (predefined request)
81    Options:
82             +I don't have it (!holds(Player, genopass))
83             +Attack (is(GateGuard, officer)
84                 and !likes(Player, officer) and !is(Player, strategic))
85             +Leave (is(GateGuard, officer)
86                 and !likes(Player, officer) and is(Player, easy-going))
87
```

```
Step 6: The Gate Guard has a wish on his own (get one of the two children of character
    Mother), so he proceeds to ask the player to help him, because he can accomplish the
    player's wish to cross the gate. Player chooses to accept
   * -> charAction(GateGuard) (hasWish(GateGuard))
   GateGuard: Request(GateGuard, Player, [Child1 or Child2])
   Options:
            +Accept (!likes(Player, kid)
                and is(Child1, kid) and is(Child2, kid))
            +Attack (is(Player, moral)
                and !is(Player, strategic) and is(GateGuard, offices)
                and !likes(Player, officer))
            +Leave (is(Player, moral) and !empty(inventory(Player)))
   * -> subwish(Player, wish(Player, CrossGate), getKid)
       -> pointsChange(moral-1=0)
           -> !property(Player, moral)

Step 7: Player talks to Mother and accepts her request (don't have another option)
   Player: SPEAK(Player, Mother)
           -> charAction(Mother) (hasWish(Mother))
   Mother: Request(Mother, Player, help([Mother, Child1, Child2], CrossGate))
   Options:
            +Accept (is(Player, generous)
                and likes(Player, woman) and is(Mother, woman))
            +Accept (!likes(Player, kid)
                and in(kid, wish(Player) and in(kid, request)))
            +Accept (!is(Player, moral)
                and in(kid, wish(Player) and in(kid, request)))
   * -> INFORM(Mother, Player, at(strongBox, BackYard))
       -> knows(Player, at(strongBox, BackYard)) and fond(Mother, Player)
```