Índex

1. Introducció, motivacions, propòsit i objectius

La confecció d'horaris de institut es un problema que amaga una alta combinatòria, dificultant-ne molt la seva elaboració manual posat que s'han de prendre moltíssimes decisions a cegues, fent que sigui molt probable cometre errors en la confecció. El HSTT (High School Time Table) consisteix en la solució de forma automàtica de aquest problema d'alta complexitat (NP). Així doncs el problema consisteix en la configuració automàtica de horaris de institut partint de una sèrie de recursos (per exemple: aules, professors, assignatures, grups) i repartir-los de manera que sigui viable i tenint en compte de manera total o parcial les preferències del professorat en quant a horaris, continuïtat, grups, etc. Tot això fa que el problema sigui molt difícil de resoldre, degut al gran nombre de combinacions possibles entre els diferents recursos.

Afegint dificultat al problema, depenguen del país del qual estiguem parlant existeixen una gran diversitat de requisits propis, degut a les característiques pròpies del sistema d'estudis secundaris de cada lloc.

Els objectius d'aquest treball són aprofundir en el problema treballat, l'estat actual d'aquest, estudiar les tècniques més utilitzades per resoldre'l actualment. També es pretén solucionar-lo usant les eines desenvolupades recentment pel grup de recerca de Lògica i Programació. Així s'implementarà un generador d'horaris capaç de resoldre el problema utilitzant aquestes tècniques basades en SMT i SAT.

1.1. Grup de recerca Lògica i Programació

Aquest treball s'emmarca dins del grup de recerca de Lògica i Programació de l'àmbit d'àrea tècnica de la Universitat de Girona.

El grup basa la seva recerca en l'estudi de satisfactibilitat de formules proposicionals booleanes (SAT) i Satisfiability Modulo Theories (SMT) i les seves aplicació per a la resolució de problemes combinatoris com ara problemes de *scheduling* i *planning* arribant a utilitzar amb èxit tècniques innovadores en altres àmbits com pot ser: els problemes de *scheduling* i *planning*.

Durant la elaboració del treball he rebut ajuda i assessorament dels membres del grup, incloent el meu tutor de projecte, el Dr. Josep Suy, qui també en forma part.

2. Estudi de viabilitat

3. Metodologia i Planificació

3.1. Metodologia

La part més important i grossa d'aquest treball és la creació d'un generador automàtic d'horaris utilitzant les eines oferides per el grup de recerca. Primer caldrà estudiar el problema (HSTT) i la seva duresa, per poder ser capaç d'entendre el què estem treballant. Des de aquí es procedirà a la implementació del programa, utilitzant la API SMT creada per el Dr. Jordi Coll, el qual es membre del grup de recerca de Logica i Programació del departament de Informatica, Matematica Aplicada i Estadistica. Aquesta API ens estalviarà la codificació de les restriccions de cardinalitat i les restriccions pseudo-booleanes, apart de oferir-nos una interfície senzilla per poder implementar el model en diferents encodings i múltiples opcions.

3.2. Planificació

Com s'ha dit en l'apartat anterior primer caldrà estudiar el problema HSTT i la seva duresa. Després es procedirà amb el disseny i la implementació del generador. Primer caldrà dissenyar implementar i testejar un *parser* pels fitxers. En aquest pas també cal pensar i implementar en quina estructura es guardaràn aquestes dades i com es transferiràn en el model que es codificarà posteriorment.

Al tenir el *parser* i l'estructura de dades enllestits caldrà començar a estudiar com funciona la API per C++ del yices i posteriorment la API SMT del Dr. Jordi Coll. Això ens permetrà començar a dissenyar, codificar i testejar el model, que és el següent pas del treball. Al tenir enllestit el model, es faràn les proves de rendiment amb diferents límits de optimització i diferents encodings de les restriccions de cardinalitat. Finalment s'implementarà una forma maca i llegible de mostrar els horaris generats.

Amb això el generador es donarà per acabat i es passarà a la confecció de la memòria del treball.

4. Marc de treball i conceptes previs

4.1. Definició del problema

4.2. Format XHSTT

4.3. Cardinality encodings

4.4. Estat de l'art

5. Requisits del sistema

En aquest treball es pretén desenvolupar un programa en C++ que haurà de ser capaç de rebre un fitxer xml en format XHSTT, llegir-lo, codificar-ne el model a yices, resoldre'l i mostrar-ne el resultat de forma llegible. Per fer això requerirem d'una llibreria que ens permeti llegir fitxers XML (en aquest cas s'ha decidit utilitzar pugixml[1]). També requerirem instal·lar el yices[2]. Posat que la API SMT del Dr. Jordi Coll està desenvolupada per a linux, necessitarem que la màquina on s'executi utilitzi de sistema operatiu una distribució de Linux amb un compilador C++.

[1] https://pugixml.org/
[2] https://yices.csl.sri.com/

6. Estudis i decisions

6.1. Programari utilitzat

A continuació s'enumera tot el programari que s'ha utilitzat per la confecció d'aquest treball:

- El solver utilitzat és el Yices versió 2.6.1 degut a que és dels millors que hi ha i el grup de recerca hi te experiència prèvia.
- Per la lectura de les instancies XHSTT s'ha utilitzat la llibreria pugixml versió 1.9-1. S'ha decidit usar aquesta llibreria pel fet de que ja hi tenia experiència prèvia.
- La codificació del generador s'ha fet en C++ 17 i s'ha compilat amb el gcc versió 9.1.0-2 perqué és amb el que es treballa al departament i permet la utilització de la API SMT del Dr. Jordi Coll. Per desenvolupar s'ha utilitzat el IDE qtCreator 4.9.2-3.
- La elaboració d'aquesta memòria s'ha fet amb LaTeX, utilitzant com a IDE el Microsoft Visual Studio Code amb la versió lliure per linux 1.37.1-1, juntament amb l'expansió LaTeX Workshop (versió 5.6.0).

6.2. Maquinari utilitzat

Per efectuar les proves de rendiment s'ha utilitzat un ordinador amb les següents especificacions:

- Processador AMD Ryzen^{TM} 3 1200 a 3.5 GHz amb 4 nuclis físics, 10MB de memòria cau i arquitectura 64 bits.
- 16GB de memòria RAM DDR4 a 3333MT.
- Sistema operatiu Arch Linux 64 bits amb kernel Linux 5.2.9.arch1-1

7. Anàlisis i disseny del sistema

7.1. Anàlisis

7.1.1 Necessitats del sistema

Les necessitats principals del sistema són les següents:

- Necessitem rebre un fitxer de l'usuari.
- Necessitem llegir les dades de un fitxer XHSTT tal i com s'ha explicat anteriorment, per tant requerirem de un *parser* per ferho.
- Necessitarem guardar les dades i les restriccions de alguna manera.
- Necessitarem un model lògic i codificar-lo utilitzant la API SMT del Dr. Jordi Coll.
- Necessitarem processar i guardar les dades de manera que ens faciliti el mostrar-les de forma que es pugui entendre.
- Necessitarem comprovar en la mesura del possible que la instància XHSTT sigui correcta.

7.1.2 Anàlisis de processos

L'usuari cridarà el programa, el programa s'encarregarà de codificar el model pel yices i cridar-lo per resoldre'l, utilitzant la llibreria per C++ pròpia del yices.

7.2. Disseny

7.3. Interfícies d'usuari

El programa funcionarà via consola. Les dades necessàries, com ara el fitxer amb la instància, es passaran per paràmetres, utilitzant el sistema existent en la API SMT del Dr. Jordi Coll.

7.4. Model de dades

El model de dades correspondria al següent diagrama:

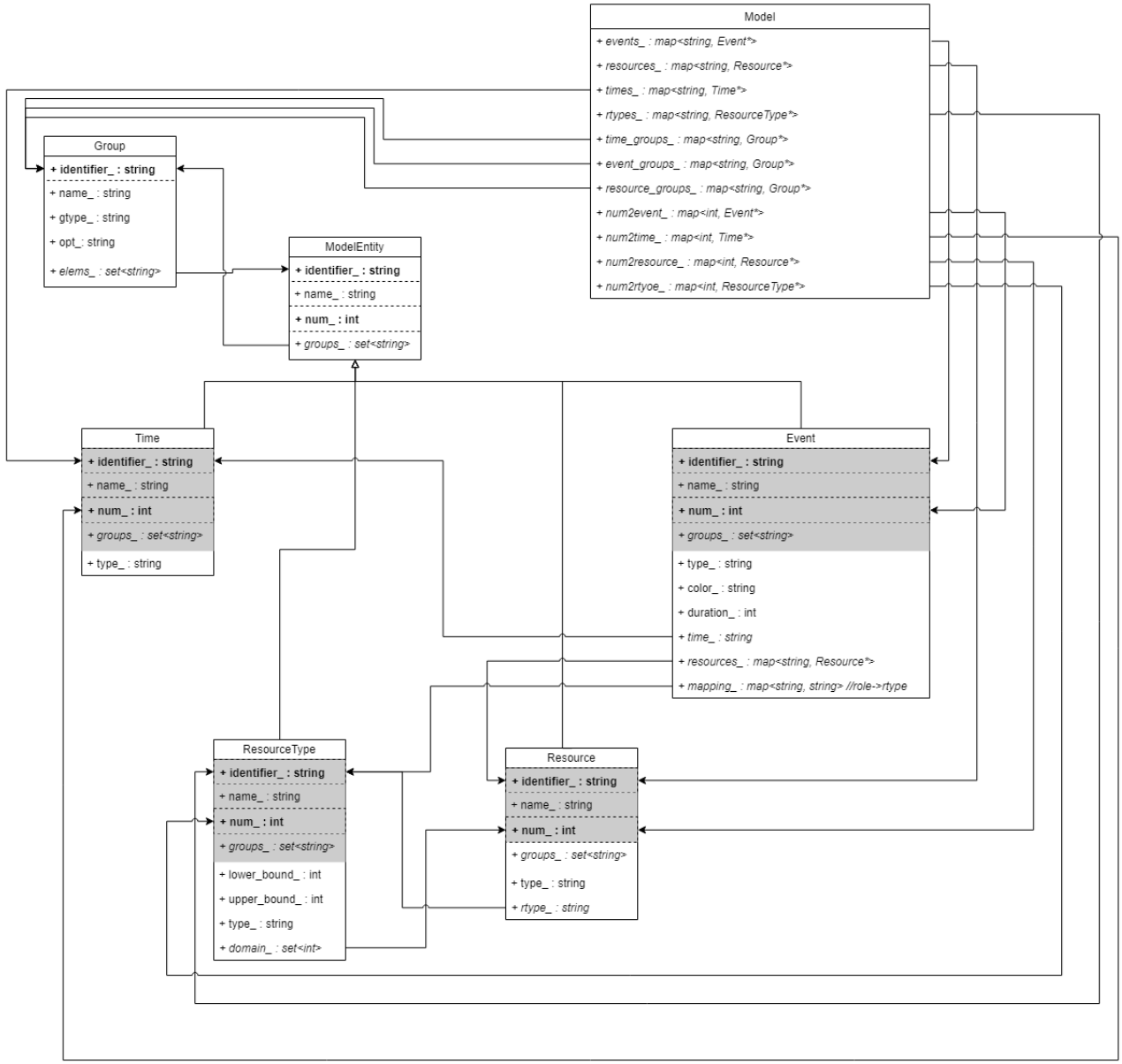

Figura 7.1: Model de Dades

7.5. Model d'objectes

El model d'objectes correspondria al següent diagrama:

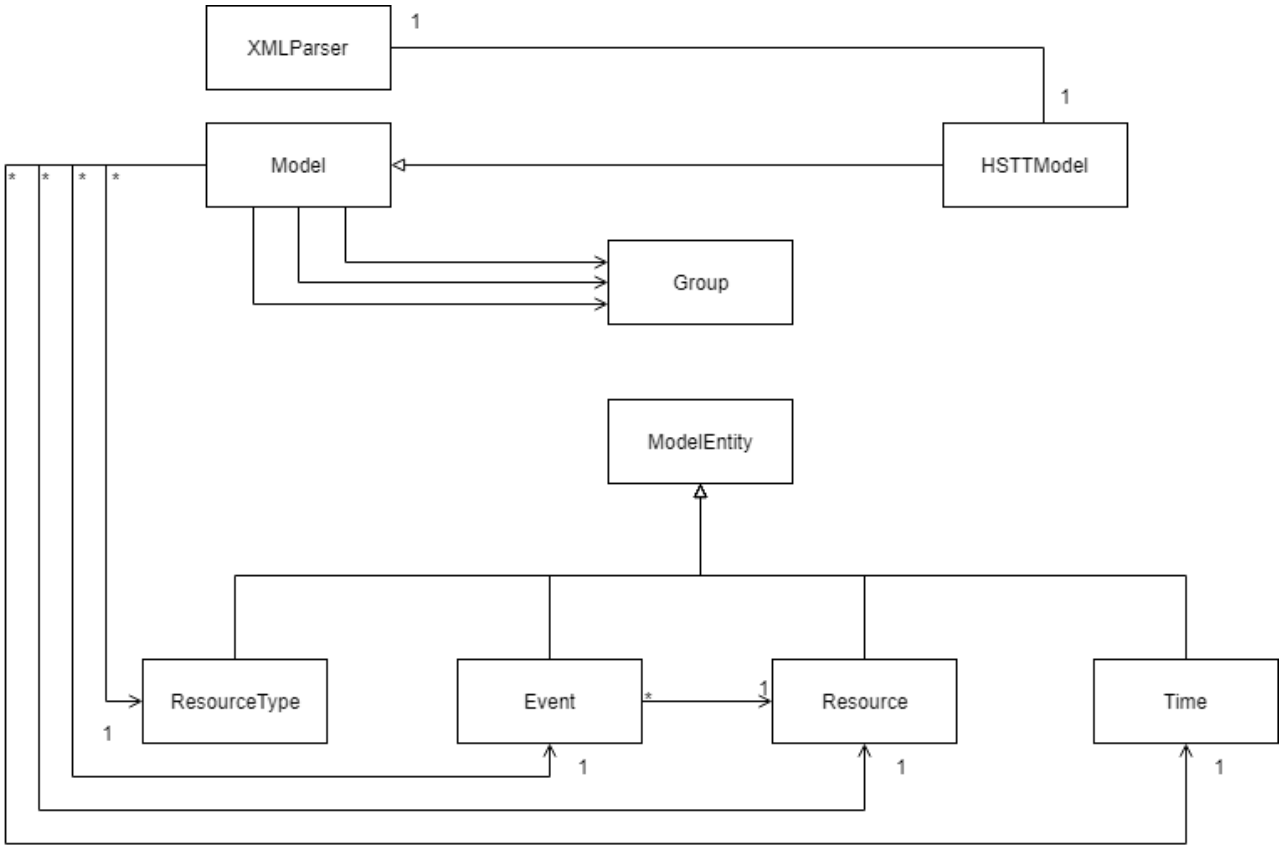

Figura 7.2: Model d'objectes

8. Implementació i proves

9. Implantació i resultats

10. Conclusions

11. Treball futur

12. Bibliografia

13. Manual d'usuari i instal·lació