Un model de decisió pel tractament mèdic a partir de diagnòstics imprecisos

Memòria del Projecte Fi de Carrera d'Enginyeria Informàtica realitzat per:

Marc Valls Ribas

i dirigit per:

Josep Puyol Gruart

Lluís Godó Lacasa

Bellaterra, 31 de Gener de 2007

Universitat Autònoma de Barcelona

etse
Escola Tècnica Superior d'Enginyeria

El sotasignant **Josep Puyol Gruart** *professor de l'Escola Tècnica Superior d'Enginyeria de la UAB,*

CERTIFICA,

***Que el treball a què correspon aquesta memòria ha estat realitzat sota la seva direcció per en* Marc Valls Ribas.**

I per tal que consti signa la present:

Signat: Josep Puyol Gruart
Bellaterra, 31 de Gener de 2007

1 - ÍNDEX

2 - INTRODUCCIÓ

En aquest projecte s'ha treballat sobre un problema que afecta diàriament als metges. L'objectiu és construir un sistema expert que donat un diagnòstic imprecís d'un metge respecte un malalt amb una malaltia infecciosa i els paràmetres mèdics d'aquest pacient, doni com a resultat els possibles tractaments que se li poden subministrar al pacient, ordenats de major a menor eficàcia per combatre la malaltia. Aquest sistema expert resoldrà casos senzills i pretén ser la base d'una possible extensió d'ell mateix per arribar a resoldre casos més complexos. Es programarà en CLOS (Common Lisp Object System), un llenguatge estandaritzat, funcional, interpretat però que també pot ser compilat, i que prové del LISP però s'ha generalitzat per convertir-lo també en un llenguatge Orientat a Objecte.

Com veiem això és una feina molt ambiciosa i atrevida. Si finalment arribés el dia en que s'aconseguís tenir una eina com aquesta que donés uns resultats raonables, seria una revolució en els hospitals. En aquest projecte s'ha treballat per estar més a la vora d'aquest moment, però continuarà sent una utopia, ja que a mesura en que s'ha anat avançant, tot i que el programa anava millorant, donava la sensació d'estar més lluny del final. Com més coneixement s'acumula, més et dones compte que més hauries de saber i més complex és tot. D'aquesta manera vull fer referència a la famosa frase de Sócrates "Només sé que no sé res", perquè tot i que s'han assolit els objectius del projecte, també veurem que aconseguir un programa fiable que es pugui utilitzar en els hospitals és molt complicat, complex i requereix molt de treball.

2.1 - MOTIVACIONS

La revolució tecnològica ens porta, entre altres coses, a fer més còmode la vida. El camp de la intel·ligència artificial també, ajudant a calcular, organitzar i predir diferents punts entre altres. En la medicina també s'està utilitzant, però encara es pot avançar molt, com en molts altres camps. Dir quines malalties pot tenir un pacient, a partir de totes les variables que es mouen no és una feina senzilla. Hi ha moltes variables que hi entren en joc, i per un metge no és una feina fàcil. En molts casos no es pot saber del segur quina malaltia pot tenir concretament, però igualment s'ha de subministrar un tractament al pacient, i això és una tasca difícil, i és el que intentarà fer aquest programa.

Des de fa molts anys que s'ha estat estudiant mètodes. En el capítol de *modelització* veurem una sèrie d'algorismes que s'han utilitzat al llarg del temps, en relació a ajudar als metges a prendre aquest tipus de decisions. Quan en un problema hi entren moltes variables en joc, si a més aquestes hi entren de forma diferent i amb pesos diferents, i amb relacions complexes, treure un resultat és difícil, i per aquests casos per la intel·ligència artificial encara ho és. Des de 1965 s'ha estat investigant, inventant diferents programes informàtics, diferents sistemes experts, que intenten ajudar a la medicina, tant en la decisió com en la predicció. Tot i així aquests programes treuen resultats modestos, i no es fan servir en la medicina tradicional. Només es fan servir programes convencionals que calculen qüestions senzilles. La medicina és una ciència complicada, cada pacient és diferent, i cada cas és un món. Això fa complicat diagnosticar i subministrar medicaments als pacient sempre d'una forma correcte.

Tot el que sigui ajudar a la medicina, i per tant, ajudar a la gent, allargar les nostres vides així com millorar-la, són uns punts que sempre s'intentaran aconseguir, i cada vegada amb resultats més ambiciosos. És per això que aquest projecte té una gran raó de ser, i només serà una petita aportació més a tot el que s'ha fet i a tot el que es farà.

2.1.1 - Motivacions personals

Han sigut moltes les condicions que m'han conduït a fer aquest projecte. Primerament tenia clar que volia fer un projecte relacionat amb la intel·ligència artificial, el camp de la informàtica que més m'ha interessat en el transcurs de la carrera d'Enginyeria Informàtica Superior que amb aquest projecte acabo. Donat que només hi ha dues assignatures que parlen de la intel·ligència artificial en la carrera, més algunes altres d'optatives que en toquen diferents camps, fa que el meu coneixement sobre aquesta ciència sigui limitat, però suficient per veure la immensitat de problemes que es poden resoldre gràcies a ella. La capacitat d'aplicar lleis de la natura a la informàtica o d'aconseguir que un programa pugui aprendre, és fenomenal.

El programa s'implementarà en CLOS (Common Lisp Object System), com hem dit anteriorment, un llenguatge estandaritzat, funcional, interpretat però que també pot ser compilat, i que prové del LISP però s'ha generalitzat per convertir-lo també en un llenguatge Orientat a Objecte. Això em va agradar, ja que és un altre mètode de iniciar-me en la intel·ligència artificial, ja que és conegut que és un llenguatge altament utilitzat en aquest extens camp de la informàtica. Tenia uns coneixements bàsics apresos en l'assignatura de *Intel·ligència Artificial 1* en el llenguatge Common Lisp, i vaig trobar que s'aprenia ràpid a utilitzar, intuïtiu, molt potent, genèric i abstracte, capaç de representar sense gaire esforç qualsevol tipus de dades, sense haver de preocupar-te de molts temes bàsics en altres llenguatges com els tipus de dades o les reserves de memòria. Fins i tot, molts dels programes que per diversió o interès en alguna qüestió construeixo, els havia fet en Common Lisp, gràcies a la rapidesa de programació i senzillesa del llenguatge. No és un llenguatge ràpid en temps d'execució, però això en el nostre cas no té que ser un problema. Em va agradar la idea de programar el model amb CLOS i per tant, haver-ne d'aprendre molt més, i trobar les seves propietats.

I per últim trobo molt interessant treballar en un híbrid entre la informàtica i la medicina. Dos ciències totalment diferenciades, però que en aquest treball troben un nexe d'unió. És una mostra de que tot el coneixement es pot separar en diferents ciències, però finalment trobes que tot és relaciona, sempre trobarem nexes d'unió entre els diferents camps, i vet aquí la riquesa de tot plegat. A més, la informàtica, una ciència nova fruit de la revolució tecnològica del segle XX, pot ajudar a qualsevol ciència, per inversemblant que sembli, ja que sempre podrà calcular, planificar i/o optimitzar qüestions de qualsevol àmbit, i més amb l'ajuda de la Intel·ligència Artificial .

La qüestió del problema, a més, és molt interessant. Trobar un tractament per un pacient amb un diagnòstic, que a més no és concret. És fantàstic! Arribarà el dia que arribarem a l'hospital, el metge ens farà una sèrie de preguntes i proves curtes, entrarà les respostes a l'ordinador, i ens donarà exactament el tractament que necessitem? En tot cas, no serà aquest l'objectiu d'aquest projecte, però tot és un principi, és posar un gra de sorra més. I el que és segur, i sí que és molt viable, és convertir aquest projecte en una utilitat i ajuda per al metge, i això és molt gratificant. A més, m'agrada la idea d'aprendre algunes nocions bàsiques de medicina, o si més no, saber quines són totes les variables que es barregen alhora de decidir un diagnòstic i en acabat el tractament, fins a quin punt és complicat, si es manegen moltes dades i altres questions.

D'aquesta manera vaig decidir embarcar-me en aquest projecte.

2.2 - QUÈ ES PRETÉN RESOLDRE?

Imaginem-nos que arriba un persona a l'hospital amb símptomes de patir una malaltia infecciosa, o sigui una malaltia que estigui provocada per un microorganisme, o més concretament un bacteri (aquestes seran el tipus de malalties que es tracten en aquest projecte). El pacient s'haurà de sotmetre a una sèrie de proves perquè el metge determini quina malaltia té i a partir de l'estat físic i psíquic de la persona i dels símptomes de la malaltia pugui escriure un diagnòstic, on pugui representar tot el coneixement que n'ha pogut extreure, i on redactar el que creu que li pot està passant al pacient, indicant quin o quins bacteris pot tenir. En segon lloc el metge decidirà quin tractament assignar-li al pacient que pugui combatre la malaltia, i per tant curar-lo. Dit així sembla una tasca, si més no, simple. Però no és així. A més, l'èxit del segon pas, dependrà de l'encert del primer, d'haver fet un bon diagnòstic. Però a la vegada això dependrà de totes les proves que se li hagin pogut fer al pacient.

Fer un diagnòstic no és senzill, i el metge es pot trobar molts inconvenients. Normalment se li han de fer un munt de proves al pacient per realitzar un diagnòstic concret i amb un cert grau de certesa, o sigui, una fiabilitat alta del que s'ha diagnosticat sigui realment el que li està passant al pacient. Amb això ens enfrontem a dos seriosos problemes, el temps i els recursos. El temps perquè moltes vegades s'ha d'actuar ràpidament, i el problema dels recursos perquè fer un bon diagnòstic i donar un bon tractament al pacient vol dir també gastar menys recursos, és a dir, gastar menys diners. D'aquesta manera veiem per on van els trets, la qüestió es que treballarem amb diagnòstics no gaire concrets.

El problema de temps es resumeix a que el pacient necessiti rebre un tractament immediatament, que no hi hagi temps per fer-li segons quines proves. En aquest cas hi ha moltes proves que no es podran executar, i per tant els metges manegen menys dades per fer un diagnòstic concret i amb un grau de certesa alt. Bàsicament aquesta és la raó de ser d'aquest projecte. Amb les eines d'avui en dia i amb temps (hores, dies o poder setmanes com a molt) es pot arribar a fer un bon diagnòstic davant de pacients amb malalties infeccioses. Si hi ha un bon diagnòstic, que sigui concret i precís, trobar un

tractament per aquest pacient pot ser força fàcil, i la raó de ser d'aquest projecte no tindria molt sentit. Però en la realitat ens trobem que hi haurà moltes vegades que arribaran pacients a l'hospital que necessitin un tractament immediatament, o sinó els efectes de la malaltia podrien ser irreversibles. I amb poc temps, els diagnòstics poden ser no gaire concrets.

En el cas del problema dels recursos, es resumeix normalment en que les proves que se s'hagi de sotmetre el pacient per trobar un bon diagnòstic són diners, no només el cost en sí de la prova, si no també el temps de més que està el pacient a l'hospital i els metges que s'ocupen de fer la prova. Està clar, que com menys temps passi el pacient a l'hospital i menys feina doni als metges, més barat serà resoldre la malaltia del pacient. Tots sabem el problema que són els diners avui en dia en el nostre món, i és per això que també en els hospitals públics, s'intenten gastar el menor nombre de recursos possibles, i és per això que també contemplem aquest cas. De tota manera, m'agradaria pensar que mai s'utilitzés una eina com aquesta per estalviar diners a no sé que fos el cent per cent fiable.

Donat tot això, ens podem trobar amb diagnòstics imprecisos i incerts, i vet aquí que aquí està la gràcia d'aquest projecte, però mirem primer que es vol dir exactament amb imprecisos i incerts. Amb imprecisos s'entén que el metge pot arribar a assegurar quina malaltia afecta al pacient, però no sap quin bacteri en concret provoca la malaltia, i és que recordem que una mateixa malaltia pot estar provocada per diferents bacteris. Per altra banda, amb incerts es vol dir que el metge pot dubtar entre dues o més malalties com a candidates a causar els problemes al pacient, ja que les dades de que finalment es disposen, concordarien amb un conjunt de possibles malalties. És a dir, si un diagnòstic és incert, normalment encara es manegen més possibles bacteris com a candidats a ser els provocadors del malestar del pacient. Però tot i que un diagnòstic sigui incert, no té perquè ser imprecís. Un metge pot estar dubtant entre dues malalties i només un bacteri per malaltia, i per tant seria un diagnòstic incert però precís. Si dubtés entre dues malalties, i de cada malaltia entre dos o més bacteris, estaríem parlant d'un diagnòstic incert i imprecís. De tota manera això només és conceptual, ja que en aquest projecte alhora d'utilitzar el diagnòstic per resoldre el cas, ens serà igual si és imprecís o incert, o

ambdues coses a la vegada, simplement s'ha d'entendre que hi ha un conjunt de bacteris que poden ser els provocadors de la malaltia.

L'objectiu d'aquest projecte, com s'ha comentat abans, no és resoldre el que s'ha explicat fins ara. El que s'intenta millorar serà la segona part del procés de curar un pacient, quin tractament assignar-li. Un tractament està format per un o més antibiòtics. De tota manera el número d'antibiòtics ha de ser mínim, un, dos o tres com a molt. Així doncs, s'haurà de trobar quins antibiòtics seran els més adequats per subministrar-li al pacient.

El problema que tenim ara, quin tractament escollir pel pacient donat un diagnòstic que a més pot ser incert i/o imprecís, de seguida veiem que no és una feina fàcil. Si més no, pot ser una feina molt farragosa i llarga, ja que es manegen un munt de dades. Per començar de bacteris n'hi ha forces, a prop de cent (poder no tants com la gent pot pensar) i d'antibiòtics també. Per altra banda, si tenim un diagnòstic incert i/o imprecís, hem de trobar un tractament que pugui combatre tots els possibles bacteris candidats a ser el detonant de la malaltia, i aquí el problema es comença a complicar. A més, el pacient, te unes dades pròpies. Que volem dir amb això? Doncs que aquest pacient, pot ser un nadó, un nen o poder un ancià. Pot ser que sigui una dona embarassada, o que tingui una sèrie d'al·lèrgies, o infecció renal, o .... En resum, una sèrie de dades, que s'hauran de tenir en compte, ja que pot ser que el tractament que receptem pel pacient, pugui resoldre la malaltia, però poder en causarà d'altres d'igual o més greus. Al final ens trobem moltes dades, i és difícil decidir quin o quins antibiòtics seran els millors pel nostre pacient.

Així veiem el que es pretén fer, donat un diagnòstic incert i/o imprecís sobre un pacient, trobar el tractament a aplicar-li, assegurant la cura de la malaltia i reduint al mínim els efectes secundaris que aquest tractament li puguin causar.

2.3 - INSPIRACIÓ I OBJECTIUS

El problema que plantegem resoldre, ja ho estant intentant resoldre els metges a diari. I de fet el projecte està inspirat en la manera de com resolen els metges aquests casos, i bàsicament en la *Guide to antimicrobial therapy* [1], una guia que serveix com a referència als metges. En aquesta guia es descriuen totes les malalties d'origen microbial, així com les possibles causes, els possibles tractaments per cada una, les dosis del tractament en quantitat i duració, les possibles reaccions adverses, tracta casos concrets i complicats com que la pacient estigui embarassada, i tot això tenint amb compte totes les característiques importants del pacient, com per exemple l'edat. En realitat són una sèrie de taules que van relacionant totes les variables entre elles.

Un metge sempre pot anar més enllà d'aquesta guia alhora de trobar un tractament, però el nostre programa, tot i que no és l'objectiu d'aquest projecte, i quedaria molt lluny, també pot anar més enllà d'aquesta guia, com en el capítol d*e Treball futur* està explicat. El metge pot ser una persona amb experiència, una ment que pensa, i el llibre li serveix de consulta i de gran ajuda, ja que serà difícil tenir al cap tots els conceptes i relacions que explica la guia. El metge davant d'una malaltia, podrà consultar la guia, veure si en efecte els símptomes que té el malalt són els correctes, quins tractaments ho poden combatre, quina dosi aplicar, i anar amb compte de no perjudicar el pacient segons les seves característiques. Al final però, un metge experimentat sempre tindrà més criteris de decisió, que precisament serà el coneixement i l'experiència que haurà adquirit, i barrejant això i la informació d'aquesta guia (com altres guies que pugui consultar), decidirà un tractament per aplicar al pacient.

Tot i això, aquesta feina no és trivial, la guia està plena de taules. Donat un bacteri pots consultar quins fàrmacs tenen eficàcia, però després s'han de consultar que no provoquin efectes secundaris, quines dosis aplicar, que no interactuint entre ells els tractaments, i tot un seguit de comprovacions. Inversemblantment, aquesta feina es força semblant a intentar resoldre un Sudoku. En els sudokus per posar un número s'han de donar una sèrie de premisses, i normalment aquestes no són trivials i aquestes també s'han de deduir i així fins arribar a premisses trivials. Trobar un tractament donat un

diagnosi imprecís i/o incert a partir d'aquesta guia és seguir el mateix algorisme. I és així perquè abans de decidir que un tractament pot ser bo, hem de consultar moltes taules per arribar a premisses certes, i alhora per arribar a aquestes premisses haurem de consultar altres taules i així fins arribar premisses trivials. Això pot portar temps, a més de que trobar el millor tractament o la millor combinació d'antibiòtics possible és difícil.

Per això el que primer ens plantegem és una qüestió essencial. Representar en un model abstracte tota la informació d'aquesta guia, i implementar un programa en CLOS que donat un diagnòstic incert i/o imprecís i els paràmetres del pacient, ens resolgui el cas receptant un tractament pel nostre pacient. Ara bé, l'objectiu d'aquest projecte és més modest, i serà una simplificació de la informació que posa la guia. Ho podem dividir en tres objectius.

El primer objectiu serà poder representar la següent informació:

- *La relació d'eficàcia que tenen cada antibiòtic en combatre cada agent infecciós.*
- *La relació de sensibilitat que té cada antibiòtic en provocar un efecte secundari.*
- *La relació de sensibilitat dels paràmetres del pacient a ser candidats de rebre un efecte secundari per part del pacient.*

El segon objectiu serà donada tota la informació emmagatzemada, els bacteris que es descriuen al diagnòstic com a possibles candidats a provocar la malaltia i els paràmetres del pacient, donar una sèrie de tractaments adequats que puguin curar el pacient, ordenats per un coeficient d'adequació. Aquest objectiu és molt complex, ja que hi haurà un seguit de regles que s'hauran de complir, i un seguit de condicions que s'hauran de respectar.

El tercer i últim objectiu és el més important. Tant el model abstracte com el model computacional s'haurà de pensar de manera que no només pugui representar tota la informació que hem anomenat en el primer objectiu. Sinó que quan en un futur es

vulgui ampliar el projecte i representar tota la informació de que disposa la guia, i la més possible, sigui fàcil i no s'hagi de modificar res del model, sinó, només ampliar.

2.4 - AGRAÏMENTS

Quan vaig haver de decidir quin projecte fer, no en tenia cap en ment, així que vaig anar veure la relació de projectes oferts que proporciona la titulació d'Enginyeria Informàtica. N'hi havia molts per escollir, però em vaig fixar ràpidament amb els que envoltaven el camp de la intel·ligència artificial. Vaig veure ofert aquest mateix projecte que he acabat fent, però hi havia més gent que el va solicitar. En Josep Puyol i en Lluís Godó, que eren els que proposaven aquest projecte, es van encarregar de que ningú que havia volgut treballar en aquest projecte es quedés sense feina, ja que ens van poder repartir en diferents projectes semblants, jo personalment em vaig quedar amb el que originàriament estava ofert.

Així doncs agraeixo a l'Institut d'Investigació en Intel·ligència Artifical, IIIA, per donar-me l'oportunitat de fer aquest projecte i posar el meu gra de sorra en la recerca que contínuament es fa en aquest centre lligat al Consejo Superior de Investigaciones Científicas CSIC.

He d'agrair també personalment a Sandra Sandri, Lluís Godó i sobretot Josep Puyol per ajudar-me a completar aquest projecte, a explicar-me'l detingudament i fer-me un seguiment adequat i correcte. Hem hagut de quedar moltes vegades, i és d'agrair sempre tenir un forat per prestar l'atenció al meu projecte. Hem anat sempre comentant i revisant totes les fases del projecte, i això sempre ha fet que em sentís recolzat, un punt important quan estàs fent un treball d'aquesta mesura. Un senzill e-mail era suficient perquè ràpidament féssim un forat a les agendes i quedéssim un dia el més aviat possible per discutir els problemes sorgits. Sempre recordaré les múltiples vegades que em vaig dirigir a IIIA per fer una tutoria amb Josep Puyol, entrant per la porta del centre dirigint-me al recepcionista i preguntant per en Josep, i el recepcionista sempre em preguntava: - Perdona, com et deies?. Al final ja rèiem de la seva mala memòria...

Vull agrair també a les persones que tinc més a prop en el meu entorn familiar i d'amistats, que m'han ajudat d'una manera o altre a realitzar aquest projecte. Començaré per la meva família que sempre m'ha donat suport a que estudiés, i m'han donat tota la seva ajuda des de la seva modesta situació, i que sempre que havia d'estar tranquil estudiant o en aquest cas fent aquest treball, m'han deixat el meu espai i la tranquilitat necessària. També agraeixo a tots aquells a qui m'han sabut escoltar, família i amics, quan els hi explicava de que es tractava aquest projecte i com m'anava en el moment, i es que quan faig una cosa que li dedico tantes hores, tinc la estranya necessitat d'explicar el que faig, en que consisteix i saber que n'opina la gent. També agraeixo per aquest fet a Fanny per escoltar-me més que ningú, i donar-me també l'espai, suport i consell per completar satisfactòriament aquest projecte.

3 - MODELITZACIÓ

Els models són la manera com es representa el problema, com enfocar-ho, i donen una definició justa, adequada i formal del cas que s'ocupen. Primerament veurem els models que s'han inventat al llarg de la història i que han ajudat en la decisió a la medicina, d'una manera semblant a com es pretén fer en aquest projecte. És una aproximació a la curta història d'aquest camp i d'una manera resumida, però també es un començament per entendre el nostre cas, que seguidament definim formalment, d'una manera extensa.

3.1 - MÈTODES I MODELS D'AJUT A LA DESICIÓ EN MEDICINA

Hi ha altres mètodes coneguts que tracten de com ajudar als metges a trobar un tractament per un pacient, és a dir, ajuda a la decisió. Aquí comentarem en poques línies només els més importants sistemes experts d'ajuda a la decisió que han existit i que van revolucionar el món de la Intel·ligència Artificial, i que s'assemblen en alguns punts en aquest projecte, i per tant en els que està inspirat. Per més informació remetem al lector a consultar les entrades bibliogràfiques [5], [6], [7], [8], [9] i [10].

Comencem amb el primer sistema expert que es va crear, tot una revolució tecnològica i que va ser un referent per a molts. Dendral va ser el primer sistema expert utilitzat per propòsits reals. Va ser escrit en Lisp, i es caracteritza perquè la seva implementació no separa de forma explícita el coneixement amb el seu raonament i operacions, és a dir, el coneixement pot modificar-se al llarg de la vida del programa, a partir de l'experiència. A partir d'aquest programa es van donar a conèixer molt més la intel·ligència artificial, i va servir d'exemple a molts altres sistemes experts.

A l'any 1965 Edward Feigenbaum va entrar a formar part del departament d'informàtica de la Universitat de Standford. Allà va conèixer Joshua Ledeberg, que volia descobrir quina era l'estructura de les molècules orgàniques complertes. Llavors, aquests dos personatges van començar a treballar amb l'objectiu de resoldre aquesta incògnita i així va néixer el programa Dendral.

L'estructura de les molècules és molt complexa i difícil de determinar. Dendral en primer lloc infereix qualsevol possible restricció sobre la solució basant-se en el coneixement que té i en la seva base de dades. A continuació permet als usuaris afegir qualsevol altre tipus de restricció i finalment genera y comprova una llista de possibles solucions que imprimeix en ordre de preferència.

Així doncs el descobriment de l'estructura global d'un compost exigia buscar en un arbre de possibilitats, tallar tots els camins possibles i finalment treure la solució a partir del coneixement que té el programa a priori en química orgànica i genetista a més de les dades del espectrograma. D'aquí l'origen del nom, Dendral significa arbre en grec.

Abans de Dendral els químics només tenien una manera de resoldre el problema, que era prendre unes hipòtesis rellevants com a possibles solucions, posar-les a prova, comparar-les amb les dades, i mirar quina se semblava més a la solució. Així doncs, durant molts anys, el programa va tenir molt èxit entre químics i biòlegs.

A partir de Dendal, van néixer altres programes que l'intentaven millorar. Els més coneguts han sigut Meta-Dendral i Genoa. Meta-dendral, desenvolupat a principis dels anys 70, és un sistema expert capaç de crear el seu propi sistemes de regles, o sigui, és capaç d'aprendre, però no va tenir molt èxit per culpa de no obtenir un gran èxits en els seus resultats. En canvi Genoa, una versió més actualitzada de Dendral ha tingut bastant èxit comercial, ja que d'una manera més modesta, donava resultats molt bons, tot i que no era capaç de calcular qüestions complexes com Meta-Dendral. Més informació de tot això es pot trobar en les referències d'on s'ha tret aquesta informació [7], [8].

Mentrestant, però després de que s'inventés Dendral, es va crear un nou sistema expert totalment diferent, Mycin, i aquest sí que és un programa que se sembla més al que es pretén fer. La seva funció és aconsellar als metges en la investigació i determinació de diagnòstics en el camp de les malalties infeccioses en la sang. Va ser escrit a principis dels anys 70 per Edgar ShortLiffe i els seus col·laboradors, també en la Universitat de Standford, igualment escrit en Lisp, i com no, estava inspirat en Dendral.

Mycin era un programa que interactuava amb el metge. Primer sol·licitava les dades generals del pacient, sexe, edat, pes, al·lèrgies... i tots els símptomes que tenia. Les preguntes anaven canviant segons les respostes del metge. Quan Mycin tenia prou coneixement per donar un diagnòstic, o simplement se li havien acabat els recursos de preguntes, donava una sèrie de hipòtesis ordenades. A partir d'aquí, si el metge creia que una hipòtesi era impossible l'eliminava, i després tornava a deixar a Mycin que calculés quin tractament s'hauria de seguir per resoldre els diagnòstics que s'havien fet.

Com a ajuda de Mycin va néixer Tieresias, un altre sistema expert que feia d'intèrpret entre Mycin i els metges, alhora d'introduir nous coneixements en la seva base de dades. L'especialista utilitzava Mycin d'una forma normal, y quan aquest cometia un error de diagnosi, Tieresias corregia l'error destruint la metaregla si era falsa o ampliant-la si era el que necessitava. Així doncs aplicava a Mycin aprenentatge supervisat.

Han continuat sortint importants sistemes experts d'ajuda a la decisió, però amb objectius molts més concrets, i que també han tingut molt d'èxit, però que ja no comentarem més perquè no és la intenció d'aquest treball.

3.2 - MODEL ABSTRACTE

En aquest apartat s'explica d'una manera formal i concreta totes les variables que s'utilitzaran per representar el cas que ens ocupa, així com les relacions entre elles, la manera que es calcularan els resultats i tots els coneixements necessaris que es necessiten per començar a pensar com implementar el programa. Fem referència a l'article *Assessing adequacy and risk of drugs in treatments for imprecise clinical diagnosis* [2] ja que el que s'explicarà a continuació està basat en aquest article, segueix el mateix nus de desenvolupament, però està ampliat i amb uns matisos que he decidit canviar.

3.2.1 - Disseny del Model

Els antibiòtics que s'hauran d'aplicar per curar les malalties infeccioses depenen sobretot dels bacteris que la causen. Diferents antibiòtics poden combatre diferents bacteris, i cada antibiòtic pot combatre en diferent grau cada bacteri. Això del grau és important. Durant tot el model estarem parlant de graus, que estaran subjectes a les relacions que hi hagi. Més endavant explicarem què representen. Per altre banda també ens haurem de fixar en les característiques particulars del pacient. Ja que diferents antibiòtics poden provocar adversitats al pacient segons les seves característiques. I altre cop, les adversitats seran en un grau concret. És per això que en el model que nosaltres representem definim 4 grans grups d'objectes.

- **Grup 'A' Microorganismes** (Causal Agents). Són el conjunt de bacteris.
- **Grup 'T' Antibiòtics** (Tractaments). Són els diferents antibiòtics que es poden utilitzar per combatre la malaltia.
- **Grup 'E' Efectes Secundaris** (Side Effects). Són els efectes secundaris que poden produir els antibiòtics.

- **Grup 'P' Paràmetres** (Parameters). Són les diferents característiques del pacient. Cada un dels paràmetres indicarà una característica del pacient com per exemple edat, pes, sexe o al·lèrgies. Com veiem mesclarem dades numèriques (edat, pes), amb dades de ponderació de gravetat (insuficiència renal, que pot ser més o menys aguda) i de dades de si/no (al·lèrgies). Més endavant veurem com podem representar això concretament.

Notar que a partir d'ara si parlem de microorganisme o de bacteri, ens estem referint al mateix. Una vegada definits aquests quatre grans grups, els relacionarem en tres grans representacions, i en aquest cas tres grafs bipartits i dirigits. Seran tres grafs que relacionaran els elements d'un grup, amb els elements d'un altre grup. Hi haurà el graf de sensibilitat farmacològica que relaciona antibiòtics (grup T) i bacteris (grup A), indicant en la relació amb quina efectivitat combaten els antibiòtics als bacteris que estan relacionats. Si un antibiòtic no està relacionat amb un bacteri, significa que aquest antibiòtic no té cap efecte sobre aquest bacteri. Per altre banda hi ha el graf d'efectes secundaris que relaciona antibiòtics (grup T) i efectes secundaris (grup E), on les relacions que hi ha entre aquests grups signifiquen que un antibiòtic és propens a provocar els efectes secundaris que relaciona amb un grau concret. I per últim el graf d'efectes secundaris a evitar, que relaciona efectes secundaris (grup E) amb els paràmetres (grup P), on la relació indica que cada paràmetre propicia en un grau concret que l'efecte secundari sigui propens d'existir, en cas de que algun antibiòtic el pugui provocar. En la *figura 1* veiem un possible graf de sensbilitat farmacològica, on es relacionen antibiòtics i bacteris. *Ceftriaxone*, *cefepime*, *ceftazidime*, *gentamicin* i *aztreonam* són els antibiòtics, mentre que *pneumococcus* i *psedomonas* són els bacteris.

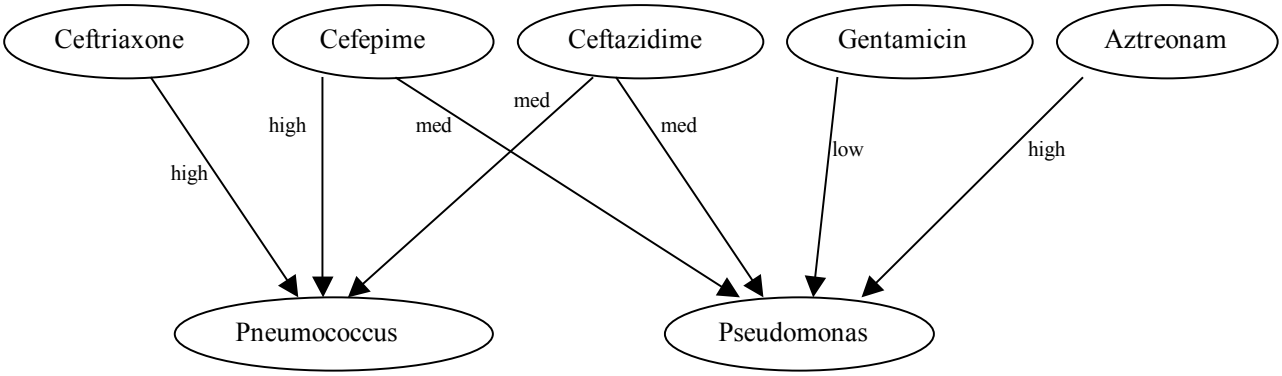


Figura 1: Mostra del graf de sensitivitat farmacològica

Veiem que cada relació porta una etiqueta, que és el grau de la relació, i indica en quin grau l'antibiòtic combat el bacteri.

Formalitzem en la següent definició el que s'ha explicat fins ara.

***Definició 1*:**

Donat un conjunt de medicines T, un conjunt de bacteris A, un conjunt d'efectes secundaris E, un conjunt de paràmetres clínics P i un conjunt de valors V, formalitzem el coneixement en els següent quatre grafs bipartits, dirigits.

- **Graf de sensibilitat farmacològica**. És una relació amb graus entre els objectes T i A, on cada parella $\langle t_i,a_j \rangle \in T \times A$, significa que el antibiòtic t_i combat el bacteri a_j amb un grau, inclòs en la relació, *pharm_sensitivity*$(t_i,a_j) \in V$.
- **Graf d'efectes secundaris**. És una relació amb graus entre els objectes T i E, on cada parella $\langle t_i,e_j \rangle \in T \times E$, significa que el antibiòtic t_i provoca l'efecte secundari e_j amb un grau, inclòs en la relació, *side_effect*$(t_i,e_j) \in V$.
- **Graf d'efectes secundaris a evitar**. És una relació amb graus entre els objectes P i E, on cada parella $\langle p_i,e_j \rangle \in P \times E$, significa que e_j és un efecte secundari a evitar, amb un grau de preferència, inclòs en la relació, *effect_to_avoid*$(p_i,e_j) \in V$, sempre i quan p_i estigui present.

D'aquesta manera es representa tota la informació necessària. S'ha parlat de graus en les relacions dels grafs, i hem vist que aquests es representen en el conjunt V. Aquests graus de les relacions, han de seguir un conjunt de propietats i restriccions. Han d'estar dintre d'un rang definit i es poden ordenar entre ells. El domini dels graus s'ha de definir però pot ser flexible. És a dir, s'ha de poder decidir el domini dels graus i si són valors tipus quantitatius (números) o tipus qualitatius (del tipus que s'ha utilitzat en la *figura 1*), però una vegada s'han definit, tots els graus del model han de ser del mateix tipus. En tot cas els graus han de tenir un element màxim, un element mínim o llindar,

un element null i s'han de poder ordenar entre ells. El domini de V estarà comprés entre l'element null i l'element màxim. L'element mínim o llindar és un valor intermedi. L'element mínim marca el llindar en el qual té sentit d'existència la relació que marca, així doncs no és el valor més baix que pot prendre un valor del conjunt V, ja que aquest és el valor null. Si una relació tingués un grau més baix que l'element mínim, automàticament tindria un grau null, i a la vegada la relació es pot eliminar. Per tant, si una relació possible en qualsevol dels grafs no existeix, és equivalent a que existeixi i el valor del grau de la relació sigui null.

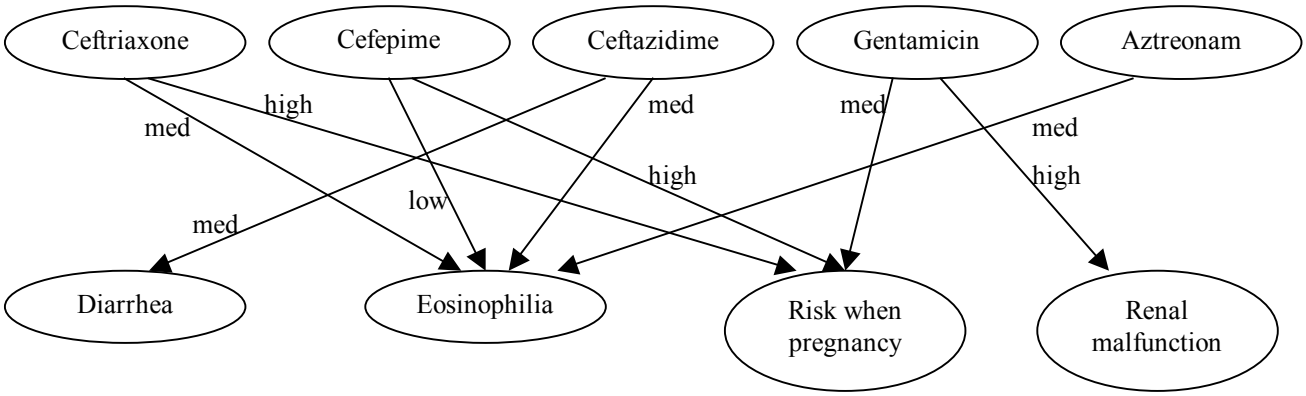


Figura 2: Mostra d'un graf d'efectes secundaris.

La *figura 2* representa un cas simplificat de graf d'efectes secundaris. *Ceftriaxone*, *cefepime*, *ceftazidime*, *gentamicin* i *aztreonam* són els antibiòtics, mentre que *diarrhea*, *eosinophilia*, *risk when pregnancy* i *renal malfunction* són els efectes secundaris. Igual que en la *figura 1*, trobem graus a les relacions. Aquests graus en aquest cas són de tipus qualitatiu, de fet el conjunt de valors que poden prendre els graus en aquest cas és : {null, low, med, high}, on el valor màxim és *high* i el valor mínim és *low*. Entre *cefepime* i *diarrhea* no hi ha cap relació, això és equivalent a que hi hagués una relació amb valor *null*, el valor més baix que pot prendre en aquest cas un conjunt de V, i que per ser més petit que el valor mínim, no té sentit posar la relació. De fet, entendrem que qualsevol valor més petit que el valor mínim, no té sentit quantificar-lo, i sempre serà directament el valor null.

Formalitzem-ho en una altre definició:

***Definició 2*:**

Donat el conjunt de valors V:

- V ha de tenir un rang ordenat definit, i per tant podrem aplicar la funció '$\top$' màxim que ens donarà el $v \in V$ més important i la funció '$\perp$' mínim que ens donarà el $v \in V$ més petit pel qual la relació que identifica cobra sentit d'existència.
- Sempre es podran aplicar les operacions '$>$','$\leq$','$<$','$\geq$' davant de qualsevol parella de $v \in V$ dintre del rang definit.
- Ha d'haver-hi el valor null, que és equivalent a no haver-hi relació. En concepte d'ordre, null és el valor més petit, més que $v \in V$ sent v el valor $\perp$ dintre de V.
- Es podrà aplicar la operació 'neg' (negat) davant qualsevol valor $v \in V$. Aquesta operació donarà el valor complementari al valor original.

En la *figura 3* veiem una mostra d'un graf d'efectes secundaris a evitar. En les relacions es poden veure dades addicionals que anteriorment en els altres grafs no hi eren. Fins ara, els graus de les relacions que hem vist fins ara eren estàtics, és a dir, es defineix el model d'una manera, posant totes aquestes dades i no canviarà més. En canvi, en el graf d'efectes secundaris a evitar, el grau de les relacions pot ser que canviïn segons el cas del pacient. Això és molt important i és el desencadenant d'una part important d'aquest projecte.

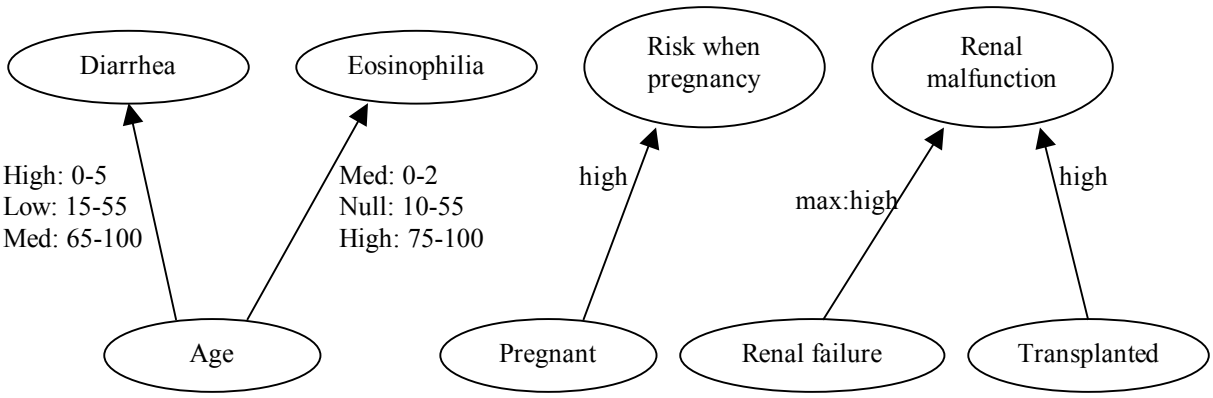


Figura 3: Mostra d'un graf d'efectes secundaris a evitar

Mirant la *figura 3*, *diarrhea*, *eosinophilia*, *risk when pregnancy* i *renal malfunction* són els efectes secundaris, mentre que *age*, *pregnant*, *renal failure* i *transplanted* són paràmetres del pacient. S'ha afegit el paràmetre *age* i les seves dues relacions per completar l'exemple, però no provenen de dades reals.

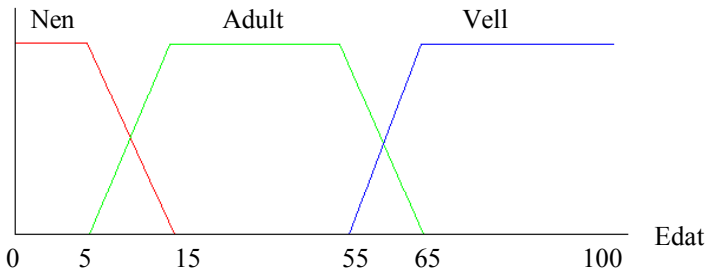
Construir el graf d'efectes secundaris a evitar i aprofitar la informació donada pels paràmetres del pacient és una tasca complicada. Ara passarem a explicar quin tipus de paràmetres ens podem trobar i que motiven a posar unes dades en les relacions. Ha de quedar clar que una vegada s'entrin els paràmetres del pacient, es farà un càlcul i cada relació quedarà exactament amb un grau concret, però depenent de les dades del pacient aquests graus poden canviar.

Hi ha tres tipus de paràmetres. Un tipus són els paràmetres del pacient que venen donats per valors booleans, per exemple, una pacient està o no està embarassada, o té al·lèrgia a la penicil·lina o no en té. En aquests casos la representació és més senzilla. Agafem l'exemple de que una pacient està embarassada, en la *figura 3* el paràmetre *Pregnant*, que pot tenir una o més relacions en el graf d'efectes secundaris a evitar, però que en aquest cas només té una relació, i aquestes relació porta un grau concret. En cas de que efectivament la pacient estigui embarassada, les relacions es mantenen, i en cas de que la pacient no estigui embarassada, automàticament les relacions esmentades passen a tenir un valor null, o el que és el mateix, desapareixen les relacions.

També hi ha paràmetres que un pacient pot tenir més o menys desenvolupats, per exemple un pacient pot tenir diferent gravetat d'insuficiència renal, com en el cas de *la figura 3* en el paràmetre *renal failure*. En aquests casos la relació pre-establerta entre els efectes secundaris i els paràmetres, en el graf d'efectes secundaris a evitar, ha de ser la màxima possible (s'indica aquest tipus de relació en el model amb la paraula clau *max* al davant). Llavors en cas de que el pacient tingués en grau màxim el paràmetre esmentat la relació mantindria el valor posat. En cas de que el paràmetre del pacient no tingués un valor màxim, el grau de la relació pot quedar-se igual o decréixer.

I per últim casos com els que en la *figura 3* representa el paràmetre edat. En aquests tipus les relacions porten unes dades, que indica segons quina edat té el pacient, el grau de la relació. De fet s'està definint els marges de la funció que utilitzarà lògica fuzzy. Com es pot veure, no hi són tots els rangs d'edats. Si el pacient té una edat que no surt en el rang d'edats, es pot calcular el grau seguint la lògica fuzzy, conjunts difusos, i que calcularà a partir dels rangs d'edat més pròxim els seus valors finals per cada cas.

En la relació de la *figura 3* entre *age* i *diarrhea*, tenim que quan el pacient té de 0 a 5 anys (podríem dir un nen), el grau de la relació és *high*, quan té entre 15 i 55 (podríem dir un adult) és *null*, i quan té més de 65 (podríem dir un vell) és *med*. Però, no queda clar quan una persona deixa de ser nen, per ser adult, i quan deixa de ser adult per ser vell. A més, cada relació entre el paràmetre *age* amb altres efectes secundaris, portarà els seus respectius marges, on cada marge representa un grau diferent. De fet, la relació entre *age* i *eosinophilia* de la *figura 5*, té marges totalment diferents com veiem. Per tant, en cada relació pot canviar el concepte de nen, o pot aparèixer el concepte jove o persona gran. El mateix passa amb altres paràmetres com el pes.



Gràfica 1. Conjuts difusos de la categoria per edat d'una persona

En la *gràfica 1* veiem els marges que s'han utilitzat per determinar els graus de la relació entre entre *age* i *diarrhea* de la *figura 3*, i les seves parts difoses. De 0 a 5 anys segur que és un nen i el grau de la relació ha de ser v_i=*high*, entre 5 i 15 anys estem entre nen i adult, i entre un grau de la relació de *high* i *null*. Entre aquest marge d'edat s'haurà de fer un algorisme que determini un grau v_i on: *null* $\leq v_i \leq$ *high*. La lògica dels conjunts difusos ens diu en aquest cas, que entre 5 i 15 anys la persona és una mica

nen i una mica *adult*, agafant unes propietats d'un i altre. Tot i així, pot ser més fàcil com es deia agafar un grau per a la relació intermig entre el grau que marca el ser adult amb el grau que marca ser nen. Llavors entre 15 i 55 anys v_i=*null*, o el que és el mateix, la relació desapareixeria, i entre 55 i 65 anys $null \leq v_i \leq med$. I finalment per més de 65 anys v_i=*med*.

Ara només falta dir que quan es vulgui resoldre un cas i s'introdueixin els paràmetres, aquest valors han de ser del tipus que s'espera. Si per exemple introduïm l'edat, hem de saber que poder s'esperen valors entre 0 i 100, i no es pot posar més de 100 o un valor qualitatiu, ja que amb aquest valor es faran uns càlculs per acabar posant un grau concret en les relacions del graf d'efectes secundaris a evitar que surten del paràmetre age. Així doncs:

***Definició 3*:**

La informació dels paràmetres ha d'estar representada per túples $\{x_j = G_j\}_{i \in I}$ on $(x_j, U_{xj}) \in P_i$ i G_j és qualsevol valor dintre del rang finit representat per U_{xj}.

3.2.2 - Propietats del model

El cas d'un pacient vindrà determinat per dos conjunts. Primer el conjunt de bacteris que són candidats a ser el provocador del transtorn al pacient, i segon els seus paràmetres. Per tant podem definir el següent:

***Definició 4*:**

El problema vindrà representat per la dobleta $C_i = \langle A_i, P_i \rangle$ on:

- $A_i \subseteq A$ representa quin o quins microorganismes pot tenir el pacient.
- $P_i \subseteq P$ conté les dades clíniques disponibles del pacient.

Una solució per la dobleta C_i serà un o més antibiòtics que ho agruparem en el que direm tractament. Aquest tractament tindrà un coeficient d'adequació $v_i \in V$. Al presentar la solució, si hi ha més d'un tractament, els presentarem ordenats pel seu coeficient d'adequació v_i. Aquest coeficient d'adequació vindrà determinat per l'efectivitat en combatre l'efecte de tots els bacteris A_i de la dobleta C_i, conjuntament amb l'efectivitat d'evitar efectes secundaris. Cada tractament T_j serà un conjunt de antibiòtics individuals, per tant $T_j \subseteq T$. Així deixem clara la diferència entre tractament i antibiòtic.

Podem pensar que tot $T_j \subseteq T$, pot ser efectiu en grau $v \in V$ a resoldre el problema. I en efecte, es pot pensar així. De fet, quan no tingui cap efecte, es donarà com a resultat el valor null, que com s'ha dit, serà en tot cas el valor més baix del rang de V. Llavors, la solució seran tots els $T_j \subseteq T$ que no tinguin efecte null? No. Seran solució tots els $T_j \subseteq T$ que tinguin un coeficient d'exit més gran o igual al valor mínim del conjunt de V. Així doncs veiem com aquest valor mínim marca un llindar també en aquest cas.

Anteriorment havíem dit que el tractament T_j combati els bacteris A_i no és l'únic requeriment que ha de complir T_j, també s'haurà de vigilar de no provocar efectes perjudicials al pacient, és a dir, que donades les dades P_i del pacient, no produeixi efectes secundaris en aquest. És important destacar això, ja que es veu clarament, com dos problemes separats, i per això formalitzem la següent definició.

***Definició 5*:**

T_j serà *cobriment* per tot A_i. Quan el tractament T_j sigui efectiu per combatre tots els microorganismes A_i.

T_j serà *segur* per tot P_i, o el que és el mateix, **T_j no és *arriscat* per tot P_i** quan el tractament T_j no produeixi efectes secundaris donades les dades P_i del pacient.

T_j és *adequat* per $C_i = \langle A_i, P_i \rangle$ si T_j és *cobriment* per tot A_i i T_j és *segur* per tot P_i.

Com veiem donem definició als dos problemes per separat. En efectes pràctiques *cobriment* i *segur* seran dues funcions amb els seus respectius paràmetres, que calcularan un coeficient d'adequació $v \in V$, i si aquest és igual o superior al valor mínim de V, les funcions retornaran *cert* i en cas contrari *fals*. Llavors perquè un cas sigui *adequat* per subministrar les funcions *cobriment* i *segur* han de retornar cert.

Ara intentarem donar una definició més exacta de *cobriment*.

- *pharm_sensitivity*(t,a), és el grau de la relació entre l'antibiòtic *t* i el bacteri *a* en que podrem trobar en el graf de sensibilitat farmacològica.
- El conjunt *indicacions*(t) són tots els microorganismes, que donat un antibiòtic *t*, són sensibles a l'efecte de *t*. Amb "sensibles" volem dir que com a mínim l'efecte entre *t* sobre el microorganisme és del mínim de V, o sigui, $\perp$.
- El conjunt *indicacions*(T_i), són tots els microorganismes que són sensibles a l'efecte d'algun antibiòtic t_j dintre del conjunt T_i. Amb "sensibles" volem dir que com a mínim l'efecte entre t_j sobre el microorganisme és del mínim de V, o sigui, $\perp$.
- El conjunt *tractaments*(a) són tots els antibiòtics que fan efecte sobre un bacteri *a*. Amb "efecte" volem dir que com a mínim l'efecte que rep *a*, és del mínim de V, o sigui, $\perp$.
- El conjunt *tractaments*(A_i) són tots els antibiòtics que fan efecte sobre algun dels antibiòtics del conjunt A_i. Amb "efecte" volem dir que com a mínim l'efecte que rep *a*, és del mínim de V, o sigui, $\perp$.

Formalment diem:

***Definició 6*:**

Per a tot $t \in T$, *indicacions*(t) = $\{a \in A \mid \textit{pharm\_sensitivity}(t,a) > \perp\}$.

Per a tot $T_i \subseteq T$, *indicacions*(T_i) = $\bigcup_{t \in Ti}$ *indicacions*(t).

Per a tot $a \in A$, *tractaments*(a) = $\{t \in T \mid$ *pharm_sensitivity*$(t,a) > \perp\}$.

Per a tot $A_i \subseteq A$, *tractaments*$(A_i) = \bigcup_{t \in Tj}$ *tractaments*(t).

Així doncs, estem en condicions de donar una definició acurada de *cobriment*. Direm que un tractament és *cobriment* d'un conjunt de bacteris A_i si aquest conjunt de bacteris està dintre del conjunt d'*indicacions*

Definició 7:

Tenim un conjunt de bacteris A_i, on $A_i \subseteq A$. Llavors un conjunt de antibiòtics T_j, on$T_j \subseteq T$, és *cobriment* de A_i sí i només si $A_i \subseteq$ *indicacions*(T_j). Denotem com a *Cov*(A_i) el conjunt de *cobriments* de A_i.

3.2.3 - Càlculs del model

Bé, recordem que el que estem pretenent és resoldre un cas, que ve representat per la dobleta $C_i = \langle A_i, P_i \rangle$, o sigui, trobar tots els $T_j \subseteq T$, que siguin *adequats* per la dobleta C_i. Doncs el primer pas per resoldre aquest problema és trobar *Cov*(A_i), i ordenar tots els resultats pel seu $v_j \in V$.

Llavors per altra banda, també haurem de calcular amb quin grau $v_j \in V$ un tractament pot provocar o no efectes secundaris, o sigui en quin grau és *segur*. Això ho calcularem de la següent manera. Un antibiòtic t_j provoca un efecte secundari $e_i \in E$, si així s'indica en el graf d'efectes secundaris per una banda, i per l'altre si també hi ha relacions que provenen dels paràmetres del pacient cap a l'esmentat e_i. En aquest cas, l'antibiòtic t_j provoca l'efecte secundari e_i, i el grau concret serà la conjunció dels dos graus de les dues relacions esmentades. S'haurà de tenir en compte que poden haver-hi més d'una

relació en el graf d'efectes secundaris a evitar que provenint de diferents paràmetres vagin a l'efecte secundari e_i. En aquest cas agafarem el màxim valor d'aquests. Ampliant-ho a si un tractament T_j és *segur* donats els paràmetres del pacient, també s'haurà de tenir amb compte que molts antibiòtics poden tenir la relació amb un mateix efecte secundari. Així, que si volem acabar calculant amb quin grau $v \in V$, tenim *risc* de provocar algun efecte secundari amb T_j en la dobleta C_i, haurem de pensar que serà el pitjor cas en que un t_j provoqui un efecte secundari e_i, donats tots els $t \in T_j$, $p \in P_j$ i $e \in E$. Donem però la definició formal de quan T_j és *adequat* per un cas C_i, però abans d'això presentem tres operadors, la conjunció que denotarem com $\otimes$, la disjunció que denotarem com $\oplus$ i el negat que denotarem com *neg*, que només tindran sentit amb valors de V. Els operadors $\otimes$ i $\oplus$ són T-normes, i més endavant donarem les seves propietats. L'operador *neg* és un altre operador que resulta necessari, i també es donen les propietats més endavant.

***Definició 8*:**

$global\_adequacy(T_j \mid C) = pharm\_adequacy(T_j \mid A_i) \otimes contx\_adequacy(T_j \mid P_i)$.

$pharm\_adequacy(T_j \mid A_i) = \otimes_{a \in Ai}\, pharm\_adequacy(T_j \mid a)$;

$pharm\_adequacy(T_j \mid a) = \oplus_{t \in Tj}\, pharm\_sensitivity(t,a)$;

$contx\_adequacy(T_j \mid P_i) = neg(risk(T_j \mid P_i))$

$risk(T_j \mid P_i) = \oplus_{t \in Tj}\ \oplus_{e \in E}\, side\_effect(t,e) \otimes effects\_to\_avoid(e \mid P_i)$

$effects\_to\_avoid(e \mid P_i) = \oplus_{p \in Pi}\, effect\_to\_avoid(p,e)$

Anem a veure pas a pas la definició prèvia, que no és més que tots els passos necessaris per trobar $global\_adequacy(T_j \mid C)$, o el que és el mateix, T_j és adequat pel cas C. Només notar, que totes les funcions descrites, sempre retornen un $v \in V$. Per tant, $global\_adequacy(T_j \mid C)$, ens dirà en quin grau T_j és *adequat* pel cas C, si és null, no serà *adequat*.

Havíem definit *pharm_sensitivity*(t,a), era el grau $v \in V$ en la relació que podem trobar en el graf de sensibilitat farmacològica. Llavors, *pharm_adequacy*(T_j | a), és la disjunció del resultat de totes les *pharm_sensitivity*(t,a), que tenim amb tots els antibiòtics T_j. I si ara calculem la conjunció de tots els *pharm_adequacy*(T_j | a), que podem trobar en el nostre cas, tants com elements A_i, tenim calculat *pharm_adequacy*(T_j | A_i). Si per un cas concret de T_j ens dona un resultat $v \in V$ que no null, o sigui $v \geq \perp$ tenim que T_j és *cobriment* de A_i.

Per altra banda tenim *effect_to_avoid*(p,e), que no és més que $v \in V$ que hi ha en la relació, si existeix, entre $p \in P$ i $e \in E$ que podem veure en el graf d'efectes secundaris a evitar. Ara podem calcular *effects_to_avoid*(e | P_i), que és la conjunció de tots els *effect_to_avoid*(p,e), que podem calcular, que són tots els paràmetres P_i que té el pacient i que poden activar algun efecte secundari. Abans de calcular *risk*(T_j | P_i), que podem definir com el risc de donar el tractament T_j al pacient amb característiques P_i, definim *side_effect*(t,e) com el grau $v \in V$ que hi ha en la relació, si existeix, entre $t \in T$ i $e \in E$ en el graf d'efectes secundaris. Ara per calcular *risk*(T_j | P_i) hem de fer la conjunció entre cada *side_effect*(t,e) i *effects_to_avoid*(e | P_i) que podem trobar, que són tants com la multiplicació de $t \in T_j$ amb $e \in E$. I de tots aquests resultats, hem de fer la disjunció per trobar *risk*(T_j | P_i). Ara ja podem calcular fàcilment *contx_adequacy*(T_j | P_i), que és el negat de *risk*(T_j | P_i). Llavors podem entendre *contx_adequacy*(T_j | P_i), com lo segur que és el tractament T_j en aplicar-lo al pacient amb característiques P_i. Si el resultat és null, direm que T_j és *segur* per tot P_i.

Ara ja podem calcular el resultat final, *global_adequacy*(T_j | C), en quin grau $v \in V$ resol el cas C el tractament T_j, com a disjunció entre els dos resultats que ja hem obtingut, *pharm_adequacy(*T_j | A_i),que seria la definició de cobriment, i *contx_adequacy*(T_j | P_i), que seria la definició de *segur*.

Hem estat fent una sèrie d'operacions amb els operadors $\oplus$ i $\otimes$. És interessant doncs donar unes regles que han de complir aquests operadors. Ho definim formalment directament:

***Definició 9*:** Els operadors $\oplus$ i $\otimes$ han de complir les següents propietats:

Operador $\otimes$:

- Commutativitat: $(a \otimes b) = (b \otimes a)$
- Monotonocitat: $(a \otimes b) \leq T(c \otimes d)$ si $a \leq c$ i $b \leq d$
- Associativitat: $(a \otimes (b \otimes c)) = ((a \otimes b) \otimes c)$
- Element null: $(a \otimes 0) = 0$
- Element identitat: $(a \otimes 1) = 1$
- $(a \otimes b) \leq min(a, b)$

Operador $\oplus$:

- Commutativitat: $(a \oplus b) = (b \oplus a)$
- Monotonocitat: $(a \oplus b) \leq T(c \oplus d)$ si $a \leq c$ i $b \leq d$.
- Associativitat: $(a \oplus (b \oplus c)) = ((a \oplus b) \oplus c)$
- Element null: $(a \oplus 0) = a$
- Element identitat: $(a \oplus 1) = a$.
- $(a \oplus b) \geq max(a, b)$

Finalment, tenim tots els passos, per saber en quin grau un conjunt de antibiòtics T_j, són efectius davant d'un cas C. El que no hem comentat encara, és com trobar tots els T_j que resolen el cas C, i en conseqüència, com trobar més ràpidament el T_j amb un $v \in V$ més gran. I tampoc de si tenim dos T_j, d'igual $v \in V$, quin és millor. Així doncs, abans de discutir tot això, presentem una última definició.

***Definició 10*:** Sent T_j *adequat* per C.

T_j és ***rellevant*** si és un subconjunt de *tractaments*(A_i).

T_j és ***irredundant*** si cap dels seus subconjunts és també *adequat* de A_i, és ***redundant*** en cas contrari.

T_j és ***minimal*** si la seva cardinalitat és la més petita de tots els *adequats* de A_i.

Llavors, tenim que un T_j solució del cas C, hauria de complir que és rellevant i irredundant. Però això no té perquè ser veritat, per exemple, un tractament $T_k \subseteq T_j$ on tant T_k com T_j són *adequats* pel cas C, diríem que T_j que és redundant, i no caldria posar-lo en la llista de solucions. Però també pot ser que T_j tingui un $v_j \in V$ més gran o important que T_k, per tant ens podria interessar llavors triar el tractament T_j, i hauríem fet malament de no incloure'l en la llista d'*adequats*. El que està clar és que si T_j no millorés en $v \in V$ a T_k, llavors segur que no caldria incloure'l, ja que tenim la premissa, de com menys antibiòtic contingui el tractament millor.

Abans de continuar decidint quins T_j posaríem a la llista de solucions i quins no, ja ens donem compte que tot dependrà de saber quin valor real tenen els $v \in V$ que hem marcat. És a dir, si diem que T_j és *adequat*, però té un $v \in V$ mínim, voldrà dir que si li apliquem aquest tractament al pacient, es curarà satisfactòriament? En principi sí, ja que amb un grau mínim és considera bo, però serà millor com més alt sigui el coeficient d'adequació.

El que també està clar, és que és interessant que un T_j sigui minimal, perquè com menys antibiòtics diagnostiquis millor. Normalment haurien de ser un antibiòtic, dos o tres. Però també tornem a estar a les mateixes d'abans, perquè per buscar el T minimal, ens perjudica el coeficient d'adequació $v \in V$, poder no hi guanyem. Per tant, quedarà en criteri de l'usuari a decidir quin T_j aplicar al pacient, fent un balanç entre l'efectivitat del tractament, i el nombre d'antibiòtics que el compon, ja que dependrà això de com s'hagi definit el model, perquè al definir el model, i posar els pesos a les relacions, també s'està definint quin és grau mínim d'un tractament per poder aplicar-li al pacient, que normalment seria el mínim de V, però sobretot, quina és la millora a base de que el coeficient d'adequació sigui més alt.

Per altre banda no és aquí on s'ha d'explicar com calcular el conjunt de solucions, però sí es pot fer esment a una premissa que es pot tenir amb compte. Donat un T_j, ens podem plantejar que volem calcular primer, el seu grau de cobriment, o el seu grau de segur, i això dependrà del temps que trigui l'algorisme a calcular les dues funcions. El

que es pot fer, una vegada s'ha calculat una de les dues funcions, és mirar el seu grau d'efectivitat, perquè si no és prou alt, i no ens interessa, ja no cal calcular l'altre funció, perquè el resultat final serà igual o pitjor. D'aquesta mateixa manera es pot pensar com calcular aquestes funcions, que tindran altres funcions incloses.

4 - IMPLEMENTACIÓ

En aquest capítol s'explica com s'ha implementat el model explicat en el capítol anterior. Per programar-lo s'ha utilitzat el llenguatge CLOS que prové del llenguatge LISP, per això és convenient primerament un apartat en que s'explica les propietats d'aquest llenguatge i el perquè s'ha utilitzat. Seguidament s'explica com s'ha fet la representació del model, amb els consegüents diagrames de classes. En acabat hi ha dos apartats per fer menció en especial als operadors $\oplus$ i $\otimes$ i a la modelització dels paràmetres del pacient, ja que són dos temes en que s'ha aprofundit especialment, i per això es tracta d'una manera especial. Finalment s'explica l'algorisme aplicat per fer els càlculs.

4.1 - LLENGUATGE CLOS

El model s'ha escrit utilitzant el llenguatge CLOS, Common Lisp Object System [1] [12] i [13]. CLOS és un llenguatge estàndard on els seus orígens estan en el llenguatge Lisp amb l'objectiu de convertir Lisp en un llenguatge orientat objecte. Per tant CLOS hereta totes les propietats de Lisp i les augmenta.

Lisp té molt bones propietats.

- És un llenguatge interpretat que també es pot compilar, pensat per programar des d'un terminal amb resposta ràpida, on els programes i les dades es poden modificar fàcilment. Així ens ajuda a fer tests ràpids.
- És uniforme en quan a la forma dels procediments i de les dades. A més, a partir d'aquests es poden crear nous programes i usar-los. Tot això fa que sigui un llenguatge intuitiu i fàcil d'aprendre.

- És capaç de representar conceptes d'alt nivell, qualsevol tipus d'identitat sense preocupar-nos de la seva forma o tamany, sempre pensant en el seu sentit més abstracte i general.
- Al evolucionar positivament, fer-se famós, i ser utilitzat per molts centres de recerca en IA, s'han desenvolupat eines d'edició i depuració que són de gran utilitat sobretot en programes grans. A més, existeix l'estàndard Common Lisp que és el resultat de vint anys d'experiments sobre ell i de refinaments.

Per això s'utilitza Lisp, i en conseqüència CLOS en el camp de la IA, ja que requereix tot això. De tots els llenguatges de programació que s'utilitzen actualment, Lisp és el segon més vell, només superat per Fortran. Lisp té una llarga història, es deriva del llenguatge IPL (Newell, Shaw i Simon (1957)). Les idees que es van seguir per crear-lo venen derivades de la psicologia, especialment de la noció intuitiva de l'associació. Molts dels primers programes foren escrits en IPL però l'any 1958 J. McCarthy inventa Lisp (List Processing), a partir de les idees de processament de llistes junt amb altres idees noves en aquell moment, i va fer que la IA comencés a utilitzar aquest nou llenguatge en comptes de IPL.

4.2 - REPRESENTACIÓ DEL MODEL

Tot seguit s'explica la representació del model que ha sigut orientat a objecte. Hi ha però dues característiques d'aquest model que són delicades d'explicar, és per això que hi ha tres apartats, el primer explica com s'ha representat el model en general, i en els dos següents apartats s'expliquen els dos casos concrets que requereixen una menció especial.

4.2.1 - Model Orientat a Objecte

El primer problema per resoldre el problema, trobar tots els T_j que resolguin un cas C, és de com representar el model computacionalment. Aquesta serà la part més important, ja que tots els càlculs es faran a partir d'aquesta representació. Els requisits d'aquesta representació del model són els següents:

- Representació adequada de la informació.
- Representar de manera més general possible.
- Facilitar l'accés a les dades.
- Deixar permís d'ampliació, de cara als nous requeriments del problema.

El que ens trobem en primer lloc, és un problema de representació de grafs. Així que una representació concreta, senzilla i general, és representar el graf a partir de dos tipus d'objectes, ***nodes*** i ***arestes***. Els nodes representen qualsevol dels quatre objectes que compon el nostre model: antibiòtics (T), agents infecciosos (A), efectes secundaris (E) i paràmetres del pacient (P). Per altra banda les arestes signifiquen la relació que hi ha entre ells, i poden pertànyer al graf de sensibilitat farmacològica, al graf d'efectes secundaris o al graf d'efectes secundaris a evitar. Per això creem, la classe *node* que representa un objecte node, i la classe *edge* que representa un objecte aresta.

Primer ens fixem en quina informació ha de tenir cada node. La classe *node* tindrà un camp que servirà per distingir qualsevol parella d'instàncies de la classe *node*. Aquest camp serà el nom. Així doncs, només ens caldrà saber el nom, per saber tota la informació que la instància d'una classe *node* pugui tenir. Així doncs la classe *node* tindrà un mètode que donant-li un nom, et retorni la instància de la classe *node* que correspongui. A més, la classe *node* tindrà dos camps més: Dues llistes d'instàncies d'arestes, una que indicarà quines arestes estan dirigides cap aquest node, i l'altre indicarà quines arestes surten d'aquest node cap a un altre.

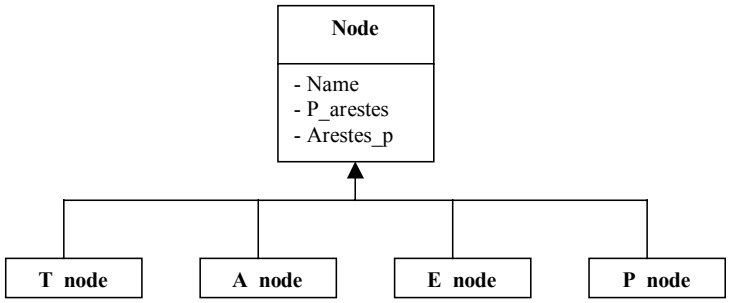


Figura 4. Diagrama de classes que representa els nodes del graf

Seguidament veiem quina forma ha de tenir la classe *edge*. Pensem, que per identificar una instància de la classe *edge* d'una altre haurem de fer servir un camp que sigui l'identificador. Però aviat es veu que no és gens útil. Una classe *edge* ha de contenir dos camps obligatòriament: Un camp conté la instància d'una classe *node* del qual prové l'aresta, i el segon la instància d'una classe *node* al qual es dirigeix l'aresta. I ens adonem llavors que aquest dos camps faran única la instància de la classe *edge*, a partir d'aquest dos camps podrem obtenir tota la informació de l'aresta, és a dir, de la instància de la classe *edge*. Conseqüentment la classe *edge* tindrà un mètode, que donades les instàncies de dos nodes, ens retorna una instància d'*edge* si aquest prové d'un d'aquests nodes i es dirigeix al segon node.

Veiem que tenim el graf muntat. Ens situem a la instància que ens situem sigui una d'aresta o d'un node, podrem anar d'un lloc a un altre qualsevol del graf, recorrent-lo tot si volem. Per això s'han de fer diferents mètodes que ho permetin fer.

Encara ens queden altres qüestions a discutir. Veiem que d'objectes node en tenim de diferents tipus, T, P, E, i A. El primer que hom pensa, és posar un camp en la classe *node* que et descrigui el tipus. Això segurament pel nostre problema, ens serviria, però quan vulguem ampliar el problema, segurament aquesta representació ens portaria molts problemes. La manera més elegant i general de fer-ho, i la que millor resulta, és practicar l'herència. És a dir, creem quatre classes més, la classe T, la classe P, la classe E i la classe A, tots ells han d'heretar de la classe node. D'aquesta manera tots hereten les propietats de la classe *node*, i que ja hem descrit, i a més cada un pot tenir les seves propietats particulars. Per altre banda l'únic que necessitem quan utilitzem un node, és la informació comuna que ja hem descrit de la classe *node*, i el tipus del node, que ens el marcarà el tipus de la classe i per tant veiem que no necessitem donar cap camp a les quatre noves classes que hereten de la classe *node*. Ho veiem això representat a la *Figura 4*.

Un situació semblant afecta a la classe *edge*. Una aresta pot pertànyer als tres tipus de grafs que pretenem representar. D'aquesta manera farem el mateix procediment que anteriorment. Definirem tres classe noves que corresponen als tres tipus de grafs, i tots aquestes tres classes hereten de la classe *edge*. Així en el nostre model mai crearem un classe *edge* general, sempre crearem una de les tres classes que hereten de la classe *edge*, igual que com faríem per crear els nodes. En la figura 5 veiem aquest nou diagrama de classes.

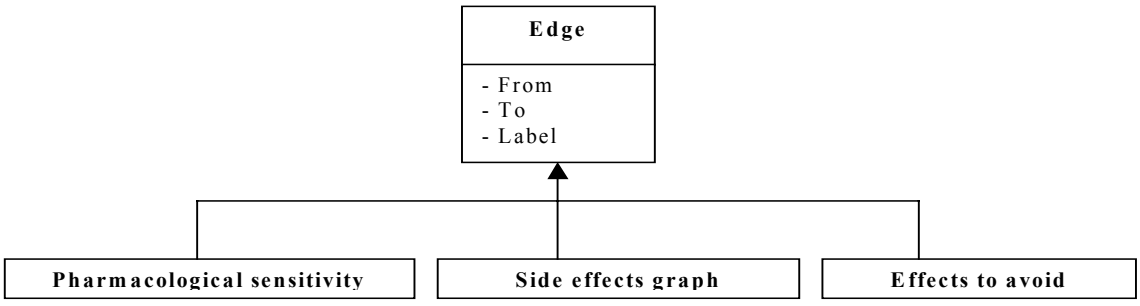


Figura 5. Diagrama de classes que representa les arestes dels grafs

Només queda un detall per configurar, són els graus de les relacions. En principi es pot pensar com un camp de la classe *edge*. Però també hem de definir les particularitats i

característiques que tenien els graus. A més, pot ser que més endavant es necessiti posar més informació en les relacions que no pas els simples graus i les seves característiques. Per això es defineix una etiqueta, una classe *Label*. Així doncs, forçarem que sempre que es vulgui fer operacions entre graus, es passaran a les funcions que facin càlculs entre graus, paràmetres que seran instàncies de la classe *Label*, que a la vegada formaran part d'instàncies de *Edges*. És per això que en la *Figura 5* veiem que hi ha un camp a la classe *Edge* que és label, que contindrà una instància d'una classe *Label*. Per altra banda s'ha pensat que els graus poden ser de dos tipus, de valors discrets o continus. Els discrets poden ser per exemple el conjunt {null, low, med, high}, i els continus poden ser per exemple vi on $0 \leq v_i \leq 100$**.** Els valors discrets seran finits, i els continus infinits. Això requereix operacions diferents. Per això també s'ha pensat en crear dos tipus d'etiquetes, i tornar a utilitzar l'herència, l'etiqueta de valors discrets, i l'etiqueta de valors continus. És per això que es creen dues classes més, *Qualitative_label* i *Quantitavie_label* que hereten de la classe *Label*, com veiem a la figura 6. Els camps que ha de contenir la classe *Label* són el valor que contindrà el grau (camp *value*), el màxim que pot arribar aquest grau (camp *max*), el mínim que pot arribar aquest grau perquè tingui sentit la relació que representa (camp min) i el que representa el valor null (camp *null*). Aquests quatre camps doncs seran comuns per totes les etiquetes, per això es posen en la classe *Label*, com veiem en la *Figura 6*. En la classe *quantitative_label*, heretada de *Label*, no contindrà cap camp més. Sabem que tots els valors possibles estan entre el màxim i el mínim que conté la classe *Label*, i com el conjunt V són números, és immediat saber quins números i quins no formen part del conjunt V. En canvi l'objecte *qualitative_label* haurà de contenir un camp que sigui una llista que contingui tots els valors que pot prendre $v \in V$.

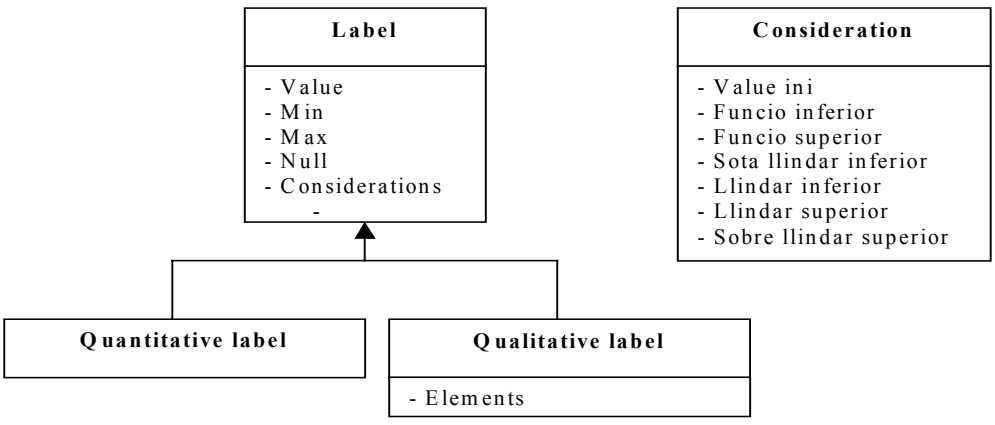


Figura 6. Diagrama de classes que representa les etiquetes de les arestes del graf

Però també s'ha comentat les relacions del graf d'efectes secundaris a evitar pot tenir informació addicional, com vèiem en la *Figura 3*. De fet l'únic cas que necessita d'aquesta informació addicional eren les relacions que sortien de paràmetres com l'edat i el pes. Posat que aquesta informació addicional afecta directament al grau de la relació, posarem aquesta informació en una classe que nombrem Consideration, i que la classe Label en tindrà una o més instàncies a través del camp *considerations*, com veiem a la *Figura 6*. Si al crear una instància de *Label* no es necessita aquesta informació addicional, el camp *considerations* no tindrà cap valor. Si necessita d'aquesta informació addicional, el camp considerations tindrà una instància o més a la classe *Consideration*. De fet la classe *Consideration* marca trossos d'una funció en que s'utilitzarà per la lògica fuzzy, i per posar un valor definitiu al camp *value* de la classe *Label*. La complexitat d'aquesta part, fa que no s'expliquin aquí els camps de la classe *Consideration*, i es remet al lector a l'apartat següent d'aquest capítol, on sí s'explica acuradament.

S'ha de remarcar, que el camp *value* de la classe *Label* és el lloc on es guardarà el grau definitiu de la relació, que en alguns casos s'haurà de calcular fent servir la informació addicional que s'ha comentat, els paràmetres del pacient en cada cas, i la informació estàndard de cada etiqueta.

I per últim s'han creat tres classes més per guardar els paràmetres que s'utilitzaran per acabar de definir les funcions $\oplus$ i $\otimes$ i que més endavant en l'apartat *Les funcions* $\oplus$ i $\otimes$, d'aquest mateix capítol s'explica amb exactitud. Podem veure la seva forma en la *Figura 7*. Una superclasse que conté els paràmetres, i les dues classes heretades, una per cada operador que hereta del pare.

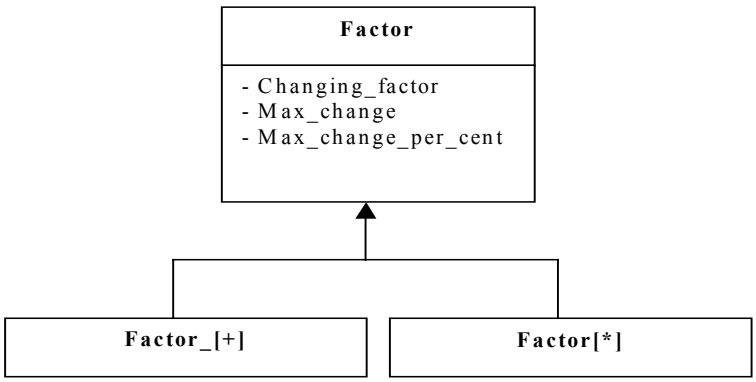


Figura 7. Diagra de classes que representa la informació necessària pels operadors $\oplus$ $\otimes$

D'aquesta manera queda representat tota la informació del model, en un model computacional, d'una manera ordenada, accessible, i senzilla.

4.2.2 - Modelització dels paràmetres del pacient

Introduir els paràmetres dels pacient, així com les relacions del graf d'efectes secundaris a evitar és complex. Se n'havien distingit tres tipus de paràmetres, però tot i així en poden haver-hi més, o d'aquests tipus fer-ne subtipus. De tota manera, aquest model intenta representar i utilitzar de forma correcte aquests tres tipus de paràmetres. Per explicar com s'utilitzaran els paràmetres del pacient, agafarem com a referència la *Figura 3*, una mostra de graf d'efectes secundaris a evitar on surten els tres tipus de paràmetres dels pacients que s'han distingit.

Els tipus de paràmetre més fàcils d'entendre són els de tipus booleà, i el seu funcionament ha de ser el següent. En la *Figura 3*, són de tipus booleà els paràmetres *Pregnant* i *Transplanted*. Si el pacient té un d'aquests paràmetres, ha sofert una transplantació, o és una pacient que està embarassada, les relacions que surten d'aquests paràmetres s'activen, o el que és el mateix, es queden tal i com estan, sense que les etiquetes de la relacions pateixin cap modificació. Pel contrari, si la pacient no està embarassada o no ha sofert cap transplantament, les relacions que surten d'aquests nodes paràmetre s'eliminen, o el que és el mateix, el valor de l'etiqueta de les relacions es posa en valor null. Per tant, en aquest casos, el camp considerations de la classe *Label* romandrà buit.

Un altre tipus de paràmetre és el del cas del paràmetre *Renal Failure*, que veiem a la *Figura 5*. En aquest cas el pacient pot tenir un grau més o alt o més baix de malfunció renal, o no tenir-ne. Estem parlant de la gravetat del paràmetre. En aquest cas s'ha de tenir en compte el següent. Les relacions que pugui haver-hi en el graf d'efectes secundaris a evitar entre algun paràmetre d'aquest tipus i els efectes secundaris han de contenir en el camp *value* de la classe *Label* un valor màxim per defecte. Però què

volem dir amb el valor màxim? Aquest valor màxim és com pot arribar a afectar el paràmetre a l'efecte secundari en el seu pitjor cas. Com veiem en la *Figura 5*, el grau de la relació que hi ha entre el paràmetre *Renal Failure* i l'efecte secundari *Renal Malfunction* és de *high* però podria ser menor si el paràmetre *Renal Failure* no arribés a afectar tant l'efecte secundari *Renal Malfunction* en el pitjor cas. Continuant amb l'exemple, el pacient pot tenir més o menys malfunció renal, llavors en cas de que tingui el màxim grau de malfunció renal, el grau de les relacions que surtin d'aquest node paràmetre es mantindran sense canvis, si el pacient no té malfunció renal, el grau de les relacions que surtin d'aquest paràmetre canviaran a valor null o el que és el mateix, les relacions desapareixeran. En cas de que el pacient tingui un grau determinat de malfunció renal, els graus de les relacions que surten d'aquest paràmetre es decrementaran linealment, segons el grau determinat de malfunció renal del pacient. Estarem modificant el camp *value* de la classes *Label*, que passarà de tenir el valor màxim esmentat que és en el pitjor cas, a un valor adaptat segons la gravetat del paràmetre del pacient. S'ha de tenir en compte que un paràmetre d'aquest tipus pot tenir moltes relacions sortints dirigint-se a diferents efectes secundaris, amb els graus d'aquestes relacions diferents, per això tot aquest algorisme. En aquest cas també, el camp *consideration* de la classe *Label* es quedarà amb un valor nul.

És important dir que aquests tipus de paràmetres, quan s'introdueixin per dir la gravetat en que un pacient té aquest paràmetre, s'haurà de fer donant un valor dintre del conjunt V, o sigui utilitzant la mateixa notació que es dona pels graus de la relació. Llavors, per calcular el valor final del camp *value* de la classe *Label* d'una relació del graf d'efectes secundaris a evitar, es calcula de la següent manera. Les variables que surten són les que té la classe *Label* i ja hem descrit. Només dir que *null* és el valor més baix que pot prendre el conjunt V.

- $Value = value * (1 - ((max - parametre\_value) / (max - null)))$

On *parametre_value* és el grau de gravetat que té el pacient tal paràmetre. Així veiem com el camp *value* de la classe *Label* agafa un valor definitiu després d'haver introduït el valor del paràmetre del pacient. Recordem que això passarà en les relacions dels grafs

d'efectes secundaris a evitar que surten de paràmetres que el pacient pot tenir més o menys desenvolupats.

Per últim presentem el tercer tipus de paràmetre, el més complicat de representar. Ara es pretén explicar com s'intenten calcular els graus sortints de les relacions sortints, i defensar perquè s'ha fet així. Els paràmetres dels quals estem parlant son paràmetres com l'edat o el pes que veiem en la *Figura 3*. En aquest casos les relacions sortints dels paràmetres com l'edat o el pes, han de tenir una funció associada, que segons l'edat o pes del pacient, doni un valor a les etiquetes de les relacions. Per tant en aquest cas el paràmetre del pacient serà un número, i aquest número ha de tenir un valor esperat que es pugui operar en la funció associada que tingui l'etiqueta. Aquesta funció la definirem a trossos i de la següent manera. Cada pujada i baixada de la funció serà una un tros de la funció. La funció en tots els seus trossos tindrà un mínim, i cada pujada i baixada de la funció (o sigui cada tros) començarà i acabarà en el mínim. El mínim de la funció es guarda inicialment al paràmetre *value* de *label*, i cada tros de la funció es guarda en una estructura *consideration* que s'emmagatzema al paràmetre *considerations* de la classe *label*, que és una llista de tots els trossos de la funció. Hi ha però dos casos especials, el primer tros no té perquè començar en el mínim, pot començar en el seu màxim local. Igualment l'últim tros no té perquè acabar en el mínim, pot acabar en el seu màxim local. Abans però de continuar, agafem un exemple per acabar-ho d'explicar millor.

Prenem l'exemple de la *Figura 3*, en el cas de la relació entre *Diarrhea* i *Age*. Com que la funció la definirem a trossos, a la classe *Label* hi ha el paràmetre *Considerations*, una llista de instàncies de l'estructura *Consideration*, que servirà per definir cada tros de la funció. En el nostre exemple veiem que el mínim és *null*, per això el camp *value* de la classe *Label* contindrà *null*. Ara bé, aquest camp serà també el que es guardarà el valor final de l'etiqueta, una vegada fets els càlculs segons el valor del paràmetre del pacient. Per altre banda de 0 a 5 anys la relació ha de tenir un valor *high*. En aquí farem servir la classe *Consideration*, ja que definirem un tros de la funció. Omplim els valors *Value_ini* = *high* (el valor màxim del tros), *Llindar_inferior* = 0 i *Llindar_superior* = 5. Com veiem doncs el camp *value_ini* marca el valor del tros de la funció en el seus valor màxims. I també comprovem que aquest valor màxim està definit per edats de 0 a 5 anys, i aquest marge és el que marquen *Llindar_inferior* i *Llindar_superior*. Si el tros

d'aquesta funció tingués una pujada inicial, el començament de la pujada ho marca *Llindar_sota_inferior*, però en aquest cas com no hi ha pujada aquest camp és igual *Llindar_inferior.* El que si hi ha és una baixada, i aquesta acaba a 15 anys, quan veiem que el valor és *null*, per això *sobre_llindar_superior* és igual a 15. Per aquest tros de la funció ja tenim definida una instància de la classe *Consideration*, excepte els valors *funcio_inferior* i *funcio_superior*. Ho explicarem més tard. Mirem com queda la funció definida per la relació entre *age* i *diarrhea* de la *Figura 3*. Aquesta següent taula representa els valors de les dues classes *Consideration* que s'han definit per representar la funció d'aquesta relació. Per tant aquesta, s'ha definit amb dos trossos.

Dues instàncies de la classe Consideration per definir la relació Age-Diarrhea de la Figura 3

Value_ini	High
Funció_inferior	1
Funció_superior	1
Sota_llindar_inferior	0
Llindar_inferior	0
Llindar_superior	5
Sobre_llindar_superior	15

Value_ini	Med
Funció_inferior	1
Funció_superior	1
Sota_llindar_inferior	55
Llindar_inferior	65
Llindar_superior	100
Sobre_llindar_superior	100

Recordem que el camp *value* de la classe *Label* és *null*, que marca el mínim de la funció. En aquest cas veiem que *Funcio_inferior* i *Funcio_superior* és 1 en ambdós casos. *Funcio_inferior* marca la forma de la funció entre *sota_llindar_inferior* i *llindar_inferior*, mentre que *funcio_superior* marca la forma de la funció entre *llindar_superior* i *sobre_llindar_superior*. Si aquests camps són 1, la funció marca una recta, en canvi, com més gran a 1 és, fa una corba. Així doncs, amb aquests valors de *funció_inferior* i *funcio_superior*, la funció que es defineix té la forma de la *Figura 8.*

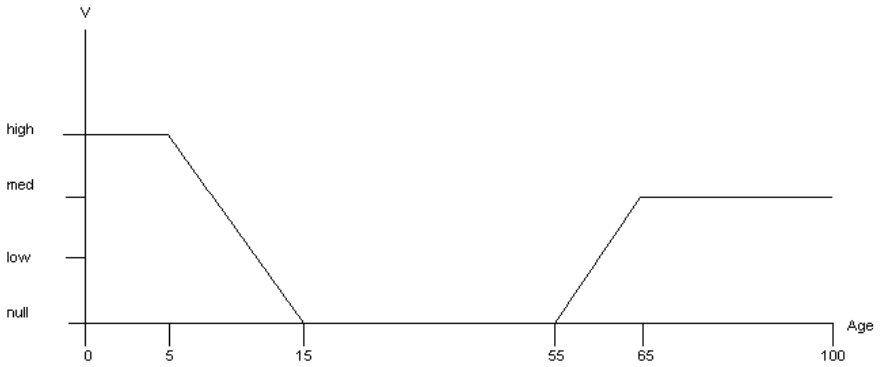


Figura 8. Gràfica que representa el grau de risc per tenir diarrhea respecte l'edat, segons les taules nateriors

En aquest cas, de 0 a 5 anys, de 15 a 55 anys i de 65 a 100 anys el valor que ha de prendre el grau de la relació és clar. Però entre 5 i 15 anys, i 55 a 65 anys és confús. Per això, els paràmetres *funcio_inferior* i *funcio_superior* que marquen la forma de la funció. Posem un altre exemple de la mateixa relació, però canviant els valors de *funcio_inferior* i *funcio_superior*.

Dues instàncies de la classe Consideration per definir la relació Age-Diarrhea de la Figura 3, ara amb valors diferents de funcio_inferior i funcio_superior

Value_ini	High
Funció_inferior	1
Funció_superior	1.75
Sota_llindar_inferior	0
Llindar_inferior	0
Llindar_superior	5
Sobre_llindar_superior	15

Value_ini	Med
Funció_inferior	2.5
Funció_superior	1
Sota_llindar_inferior	55
Llindar_inferior	65
Llindar_superior	100
Sobre_llindar_superior	100

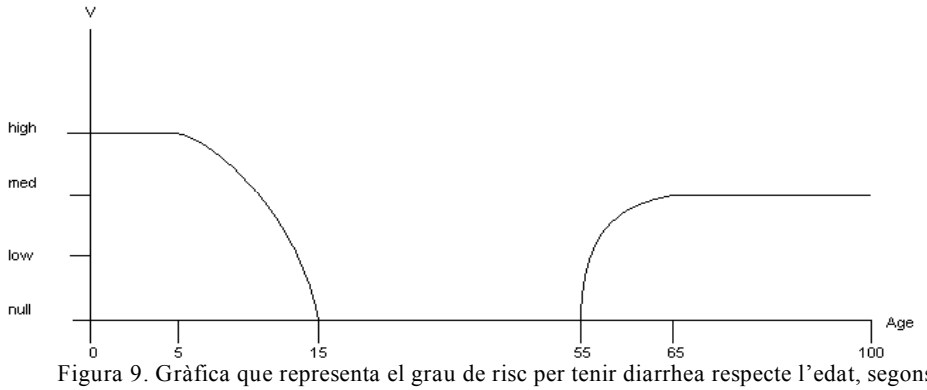


Figura 9. Gràfica que representa el grau de risc per tenir diarrhea respecte l'edat, segons les taules nateriors

Veiem en la *Figura 9* la corba que es forma pel valor *funcio_superior* = 1.75 entre l'edat de 5 i 15 anys. En canvia entre 55 i 65 anys es marca el valor de *funcio_inferior* = 2.5, i la corba és més exagerada.

La fórmula que s'utilitza per calcular els valors imprecisos són les següents:

Cas {*sota_lindar_inferior*, *llindar_inferior*}:

var aux = ((edat – sota_llindar_inferior) / (llindar_inferior – sota_llindar_inferior))

var aux2 = $(aux)^{funcio\_inferior}$

var aux3 = (aux – aux2) + aux

var Valor = ((value_ini – value) * aux3) + value

Cas {*lindar_superior*, *sobre_llindar_superior*}:

var aux = ((llindar_superior - edat) / (sobre_llindar_inferior – sobre_llindar_usperior))

var aux2 = $(aux)^{funcio\_inferior}$

var aux3 = (aux – aux2) + aux

var Valor = ((value_ini – value) * aux3) + value

El que hem fet fins aquí doncs és lògica dels conjunts difusos. Així doncs, donat una edat del pacient, es calcularà d'aquest manera quin valor posar definitivament al camp *value* de la classe *Label*, que representa la relació que hi ha entre el paràmetre *age* i l'efecte secundari *diarrhea*, en la *Figura 3*.

Aquesta manera de representar aquest tipus de paràmetres és adequada, ja que les funcions que hagin de descriure les relacions entre aquests paràmetres i els efectes secundaris no han de ser mai gaire complicades. A part, es recomana que alhora de posar valors a *funció_inferior* i *funció_superior* sempre es posin valors entre 1 i 2. Menys de 1 no té sentit, ja que les corbes que figuren a les gràfiques serien invertides, i no seria útil com es pot comprovar. Posant valors superiors a 2 ens apropem massa a unes pendents molt altes, i en principi no serà mai necessari.

És important entendre bé aquest apartat per poder-lo aplicar correctament, llegint una vegada l'apartat i mirant els exemples hauria de ser suficient, recordar que el valor mínim, (que en l'exemple és null) pot ser un altre valor, i que s'emmagatzema en *value* de la classe *Label*, al mateix lloc on definitivament es guardarà el valor correcte, i entre {*llindar_inferior*, *llindar_superior}* es posaran el valor màxim, i entre {*sota_lindar_inferior*, *llindar_inferior*} i {*llindar_superior*, *sobre_llindar_superior*}, s'aplicaran les funcions per donar un resultat entre el valor mínim i el valor màxim. S'ha pensat de manera que poden haver-hi molts valors màxims (s'omplen tantes estructures com valors màxims) però només un valor mínim comú a cada tros de la funció, però igualment això hauria de ser suficient.

4.2.3 - Els operadors $\oplus$ i $\otimes$

Com hem definit en el model abstracte, sovint, per calcular premisses, i finalment per trobar uns tractaments que resolguin un cas C, utilitzem els operadors $\oplus$ i $\otimes$ per fer operacions entre els graus de les relacions dels grafs, però aquests no s'han definit d'una manera precisa. S'han donat uns paràmetres que han de complir, però no s'ha dit quins són els resultats que han de donar aquests operadors davant de dos operants. Es podria pensar simplement que $\otimes$ correspon a la funció *min* i $\oplus$ correspon funció *max*, i segurament estaria prou bé, però això no és tant clar. De fet, hem vist que $(a \oplus b) \geq max(a\ b)$ i $(a \otimes b) \leq min(a\ b)$, així que el que hem dit, està dintre del permès.

Anem a veure però sí estem en el bon camí dient que $\otimes$ correspon a la funció *min* i $\oplus$ correspon funció *max*. Posem un exemple, intentem calcular *pharm_adequacy*(T_j | a). Per trobar la solució hem de trobar tots els graus de les relacions entre cada un dels antibiòtics $t \in T_j$ amb el bacteri 'a', i finalment aplicar la operació $\oplus$ entre totes elles. D'aquí trobarem un grau resultant que serà la solució a la funció *pharm_adequacy*(T_j | a). Dit d'una altre manera, el que estem calculant és l'eficàcia en que el conjunt d'antibiòtics T_j fa efecte sobre l'agent infecciós 'a'. Posem aquest càlcul en un exemple, i a partir d'aquest pensarem diferents maneres de calcular-ho, i definitivament com es proposa en aquest treball de calcular-ho:

- Tenim el conjunt d'antibiòtics T_j on $0 \leq j \leq 4$.
- Tenim l'agent infecciós a_1.
- El conjunt de valors que poden prendre les etiquetes de les relacions van entre 0 i 10, sent el 0 el valor *null*, i 1 el valor mínim.
- *Pharm_sensitivity(*t_0,a_1) = 5.
- *Pharm_sensitivity(* (t_1,a_1) = 4.
- *Pharm_sensitivity(* (t_2,a_1) = 5.
- *Pharm_sensitivity(* (t_3,a_1) = 1.

- *Pharm_sensitivity(* $(t_4,a_1) = 0$.

Ara volem calcular *pharm_adequacy*($T_j \mid a_1$), que recordem que és el resultat de: $\oplus_{t \in Tj}$ *pharm_sensitivity*(t, a_1).

Es podria pensar que l'operador $\oplus$ pot ser la funció *max*, llavors *pharm_adequacy*($T_j \mid a_1$)=5. Aquest pot ser un resultat raonable, o no. Que l'eficàcia d'un conjunt d'antibiòtics en combatre un agent infecciós sigui el grau màxim de tots ells, pot ser discutible. En el nostre exemple, si només actués t_0, també hi hauria una eficàcia de 5 en combatre l'agent infecciós a_1. Llavors estem dient que per molts més antibiòtics que subministris per combatre a_1, si aquests no tenen una eficàcia major a 5, no fan cap efecte. En el nostre exemple, ni t_1, ni t_3, ni tant sols t_2 que també té un efecte de 5 ajuden a t_0 a tenir un efecte conjunt major. Doncs això no té perquè ser així. Es pot pensar que tots els antibiòtics poden ajudar a pujar el valor d'eficàcia conjunta, d'una manera determinada. És a dir, estem utilitzant la propietat que ens deia que ($a \oplus b$) $\geq$ *max*(*a b*), i discutim que l'operador $\oplus$ doni un resultat una mica més gran que el que dona la funció *max*.

El mateix pot passar amb l'operador $\otimes$. Pensem que calculem *pharm_adequacy*($T_j \mid A_i$) = $\otimes_{a \in Ai}$ *pharm_adequacy*($T_j \mid a$). Això és, calcular l'eficàcia d'un conjunt d'antibiòtics, en combatre un conjunt de bacteris, una vegada es disposa de l'eficàcia del conjunt d'antibiòtics en combatre cada un dels bacteris. Posem el següent exemple:

- Tenim el conjunt d'antibiòtics T_i.
- Tenim els bacteris A_j on $0 \leq j \leq 4$
- El conjunt de valors que poden prendre les etiquetes de les relacions van entre 0 i 10, sent el 0 el valor *null* i 1 el valor mínim.
- *Pharm_adequacy (*T_i,a_0) = 8.
- *Pharm_ adequacy* $(T_i,a_1) = 5$.

- *Pharm_ adequacy* $(T_i,a_2) = 7$.
- *Pharm_ adequacy* $(T_i,a_3) = 5$.
- *Pharm_ adequacy* $(T_i,a_4) = 10$.

Es pot pensar que l'operador $\otimes$ equival a la funció *min*, i per tant en el nostre exemple l'eficàcia del conjunt d'antibiòtics T_i sobre A seria de 5. Un resultat raonable. Però en aquest cas, també només que hi hagués el bacteri a_1 ja tindríem un efecte de combatre'l de 5, per tant encara que combatem tots els altres bacteris, l'efecte total no baixa. El cost és el mateix. No seria raonable pensar que el resultat hauria de ser menor que 5? I si el resultat fos menor, com de menor? Ens trobem en el mateix cas de l'operador $\oplus$, però a la inversa.

Així doncs, hem vist que pot ser molt raonable que l'operador $\oplus$ doni un resultat superior a la funció *max* i que l'operador $\otimes$ doni un resultat inferior a la funció *min*.

És per això que s'ha pensat que el usuari pugui d'alguna manera ajudar a definir aquests operadors, forçant-los a que corresponguin a les funcions *min* i *max*, o si no ho vol així, podent graduar fins a quin punt es vol allunyar del resultat que donen les funcions *min* i *max*. Per això s'han construït les classes *factor*, *factor_[+]* i *factor[*]*. I també uns algorismes que utilitzen les dades d'aquestes classes.

Les classes *factor_[+]* i *factor[*]* deriven de la classe *factor* i d'aquesta manera agafen totes les propietats d'aquesta superclasse. Es crearà una instància de la classe *factor_[+]* per determinar el funcionament de l'operador $\oplus$, i una altre instància de la classe *factor[*]* per determinar el funcionament de l'operador $\otimes$. Per altra banda només la classe *factor* té camps, així que es passa a explicar el significat d'aquests camps:

- *changing_factor*. Conté el valor que determinarà en quina mesura el resultat d'aplicar $\oplus$ entre dos operants s'allunyarà del resultat d'aplicar la funció *max*, i el resultat d'aplicar $\otimes$ entre dos operants s'allunyarà de la funció *min*. Si

changing_factor és el valor 0, llavors . $\oplus$=*max* i $\otimes$=*min*. A mesura que *changing_factor* augmenti al aplicar més alt serà el resultat d'aplicar l'operador $\oplus$ entre dos operadors respecte la funció *max*, i més baix serà el resultat d'aplicar l'operador $\otimes$ respecte la funció *min.*

- *max_change*. Marca un límit. És la màxima diferència que pot haver-hi en aplicar l'operador $\oplus$ respecte aplicar la funció *max*, o la màxima diferència que pot haver-hi en aplicar l'operador $\otimes$ respecte la funció *min*. *Max_change* pot ser utilitzat de dues maneres diferents segons l'algorisme que es segueixi. Pot fer el que s'ha dit en cada operació entre dos operants, o agafar aquest límit com la màxima diferència que pot haver-hi en aplicar l'operador $\oplus$ sobre X operadors aplicar la funció *max*, o en aplicar l'operador $\otimes$ sobre X operadors aplicar la funció *min*. Posem un exemple per explicar aquest segon funcionament. Tenim etiquetes quantitatives. S'ha d'aplicar l'operador $\oplus$ entre X operants. Llavors el resultat d'això no pot superar en *max_change* el resultat que donaria la funció *max* sobre aquest conjunt X d'operants. Notar la diferència d'aplicar *max_change* a cada parella operants o al conjunt total operants.

- *max_change_per_cent*. És el mateix que *max_change*, però ara es refereix a la màxima diferència percentual que pot sofrir el resultat de la funcions $\oplus$ o $\otimes$ respecte les funcions *max* i *min* respectivament. La diferència entre dos valors, pot ser la diferència absoluta (el que mesura *max_change*) o diferència percentual respecte el conjunt de valors que poden prendre els dos valors esmentats (el que mesura *max_change_per_cent*). Els valors també aniran de 0 a 1.

Com veiem, si volem que $\oplus$=*max* i $\otimes$=*min*, només cal quan definim les dues instàncies de les classes *factor_[+]* i *factor[*]*, posem *changing_factor* a 0, i ja està. Ara, es proposen dos funcions diferents *rectifcia_independent* i *rectifica_dependent* que utilitzen aquests paràmetres per a que l'usuari pugui definir a la seva manera les funcions $\oplus$ i $\otimes$.

L'algorisme clau d'aquestes dues funcions es basa en calcular la funció *max* pel cas de la funció $\oplus$, i la funció *min* en el cas de la funció $\otimes$, i en acabat aplicar una rectificació del resultat depenent dels valors de les classes *factor_[+]* i *factor_[*]*. Expliquem exactament aquests dos algorismes, i en que es basa aquestes rectificacions.

Rectifica_independent. En aquest algorisme, la clau serà el significat de *changing_factor*. Aquest tindrà un valor numèric que estarà comprés entre el 0 i la diferència entre el màxim i el mínim del valor que pot adoptar el grau d'una relació. Com sempre, si l'etiqueta és qualitativa, representa que cada pas a un valor superior del grau d'aquesta, és com augmentar 1 en etiquetes quantitatives. *changing_factor* serà el pes que tindrà cada grau alhora de comptar la rectificació del resultat de les funcions $\oplus$ i $\otimes$ després d'haver calculat *max* o *min* respectivament, a més però, les rectificacions també depenen de les posicions que ocupen les etiquetes respecte el màxim en la funció $\oplus$ i el mínim en la funció $\otimes$. Expliquem l'algorisme en la funció $\oplus$ per entendre-ho millor:

- Hem d'aplicar la funció $\oplus$ sobre un conjunt d'etiquetes E1.
- Apliquem la funció *max* sobre el conjunt E1 i tenim una nova etiqueta amb el grau màxim d'aquestes. Del conjunt E1, formem un nou conjunt E2 que serà el mateix conjunt que E1 però traient una etiqueta de grau màxim. És evident que hi haurà una etiqueta o més amb el mateix valor màxim, però en traiem només 1.
- Per cada etiqueta de E2, multipliquem *changing_factor* pel complementari de la distància proporcional de l'etiqueta respecte el màxim. El complementari de la distància proporcional és:

  1– (distància del valor de l'etiqueta al màxim) / (màxim possible valor – mínim possible valor).

  Per tant el càlcul total seria el següent:

  $\oplus$ (E1) = *max(E1)* + $\sum_{k=1}^{n}$ *changing_factor * (*1– *(valor etiqueta màxim – valor etiqueta* $E2_k$*) / (màxim possible valor – mínim possible valor)).*

En el cas de les funcions $\otimes$. E2 serà el mateix conjunt que E1 excepte una etiqueta amb valor mínim, i el càlcul serà el següent:

$\otimes(E1) = min(E1) - \sum_{k=1}^{n} changing\_factor * (1 - (valor\ etiqueta\ màxim - valor\ etiqueta\ E2_k) / (màxim\ possible\ valor - mínim\ possible\ valor))$.

Fem un exemple, fent servir les dades del primer exemple, però només fem servir t_0 i t_1. que tenen un valor de 5 i 4 respectivament. Volem calcular $\oplus$. El primer pas era calcular el màxim que és una etiqueta amb valor 5. Ara del conjunt T_j traiem una etiqueta amb valor màxim, i tenim un nou conjunt T_k, així doncs només ens queda t_1. Ara de t_1 calculem quin pes té sobre el resultat. La distància proporcional és: 1/9, i el seu complementari 8/9. Posant que *changing_factor* és 0.5, el seu pes és de 0.5 * 8/9 = 0.44. Ara hauríem de sumar 0.44 més el 5 inicial. I aquest procediment ho hem de fer per cadascun dels membres de T_k. Si fos en etiquetes qualitatives, primer s'ha d'extrapolar els valors a números enters, fer els càlculs, i després arrodonir per veure a quin valor qualitatiu ens trobem.

Aquest algorisme es diu 'independent' per una raó. Posem que volem calcular la funció $\oplus$, tenim una etiqueta màxim amb valor 'k' , i una altre etiqueta amb valor 'k-2'. Llavors el pes que faci la segona etiqueta sobre la primera, serà sempre el mateix sigui quin sigui el valor de 'k'. Això no és el mateix que passa amb el següent algorisme.

Rectifica_dependent. Aquest algorisme és semblant a l'anterior, però com dèiem en el paràgraf anterior, les rectificacions dependran de si els valors són més grans o més petits. Si estem calculant la funció $\oplus$, i tenim una etiqueta màxim amb valor 'k' , i una altre etiqueta amb valor 'k-2'. Llavors el pes que faci la segona etiqueta sobre la primera, serà més gran com més gran sigui k. Simètricament passa en la funció $\otimes$. Si tenim una etiqueta amb valor 'k', i una altre amb valor 'k+2', com més petit sigui 'k', més gran serà el pes de la segona etiqueta respecte la primera.

De fet l'algorisme calcula el següent per la funció $\oplus$.

$\oplus$ E1 = *max(E1)* + $\sum_{k=1}^{n}$ *position_relative*($E2_k$) *(*changing_factor* * (1– (*valor etiqueta màxim – valor etiqueta* $E2_k$*) / (màxim possible valor – mínim possible valor))*.

E1 i E2 tenen el mateix significat que abans. E1 és el conjunt d'etiquetes inicial, i E2 és el mateix conjunt que E1 però traient l'etiqueta amb valor màxim.

La funció *position_relative*, retorna un número de 0 a 1, i com més a prop del màxim de l'etiqueta estigui el seu paràmetre, un valor més a prop de 1 retornarà, en cas contrari retornarà un valor més proper a 0.

En el cas de calcular la funció $\otimes$, tenim:

$\otimes$ E1 = *min(E1)* + $\sum_{k=1}^{n}$ *position_relative*($E2_k$) *(*changing_factor* * (1– (*valor etiqueta màxim – valor etiqueta* $E2_k$*) / (màxim possible valor – mínim possible valor))*.

E2 en aquest cas és el mateix que el conjunt E1 excepte una etiqueta de valor mínim. Per altra banda *position_relative* també té un valor diferent. Aquesta retorna un número de 0 a 1, i com més a prop del mínim de l'etiqueta estigui el seu paràmetre, un valor més a prop de 1 retornarà, en cas contrari retornarà un valor més proper a 0.

4.3 - ALGORISME APLICAT

Hi ha diferents funcions que han estat desenvolupades per donar una informació que pot interessar a l'usuari, però n'hi ha una que és la més important, que a partir de carregar el model, uns microorganismes a combatre i els paràmetres del pacient, ens retorna ordenats per ordre de coeficient d'adequació tots els tractaments que es poden subministrar al pacient. Aquesta funció és *find_treatment* i té varies parts.

La funció *find_treatment* té tres arguments *microorganisms*, *parameters* i *file*. El funcionament d'aquesta funció és el següent: en primer lloc carrega el model a partir de les dades de *file* que és un fitxer on s'ha hagut d'escriure d'una manera determinada el model. En l'apèndix s'explica com omplir aquest fitxer, que no és més que la representació del model d'una manera concreta perquè el programa ho entengui. En aquest fitxer també hi figura la forma que han de tenir les etiquetes de les relacions. Totes les etiquetes de les relacions seran del mateix tipus, és a dir, totes seran una instància de la classe *qualitative_label* o de la classe *quantitave_label*, i els seus camps *màx*, *min* i *null*, i en el cas de *qualitative_label* el camp *elements* seran sempre els mateixos. D'aquesta manera, sempre podrem accedir a la informació, i quan fer operacions entre etiquetes, també es disposarà de tota la informació. També es carrega la informació que ha de configurar els operadors $\oplus$ i $\otimes$. Es crea una instància de la classe *factor[+]* i s'omplen els paràmetres, i en aquesta instància es guarda la informació que dicta el funcionament de l'operador $\oplus$. També es crea una instància de la classe *factor[*]* i s'omplen els paràmetres, i en aquesta instància es guarda la informació que dicta el funcionament de l'operador $\otimes$. En acabat es creen totes les instàncies de nodes i arestes, que creen els tres grafs que representen el model.

Seguidament la funció *find_treatment* fa una simplificació de l'algorisme deixant només els nodes i les relacions que ens interessen pel cas que s'analitza. És la funció *optimize_graph*. Tot i que es pensa que el temps d'execució no serà un problema d'aquest projecte, no està de més simplificar el model a la hora de trobar solucions, ja que totes les operacions es simplifiquen i fins i tot és més fàcil d'analitzar si es volgués.

La funció *optimize_graph* està formada per tres funcions més. La primera és *optimize_microorganisms* que elimina tots els nodes de tipus microorganismes del model que s'ha carregat que no estan dintre del paràmetre d'entrada de la funció *microorganisms*. És a dir si el model compte amb els microorganismes {a,b,c,d,e,f,g}, i *microorganisms* és el conjunt {b,d}, el model es simplifica als microorganismes {b,d}. Llavors, també s'eliminen totes les arestes del model que tenien en origen o destí alguns dels nodes eliminats. A més, si algun node s'ha quedat sense cap relació entrant o sortint també s'elimina.

La segona funció amb que compta *optimize_graph* és *optimize_parameters* i funciona d'una manera molt semblant i té el mateix objectiu. Tots els paràmetres que hi ha al model i que el pacient no té o no s'ha definit mitjançant el paràmetre d'entrada *parameters*, s'eliminen, eliminant també les relacions que en sortien. Igual que en la funció anterior, si algun node del model es queda sense cap aresta entrant o sortint s'elimina del model.

I per últim, la tercera funció d'*optimize_graph* és *optimize_treatments*. Aquesta elimina tots els antibiòtics que no tenen relació directe en el graf de sensibilitat farmacològica amb algun dels microorganismes que s'han d'atacar en el cas concret, és a dir, antibiòtics que segur que no podran aportar res de bo a combatre els microorganismes que s'han d'eliminar.

Després de simplificar el model la funció general *find_treatment* genera totes les possibles combinacions d'antibiòtics, sabent el màxim nombre d'antibiòtics que pot estar composat un tractament (s'introdueix en el fitxer carregat). Llavors prova cada tractament que s'ha generat quin coeficient d'adequació té, i es guarda en una llista. Es prova mitjançant la funció *global_adequacy*, el cervell de l'algorisme es podria dir, i fa exactament el que s'explica en l'apartat de *Model abstracte* dintre del capítol de *Modelitazació*. Com podem veure, aquest algorisme no és òptim en complexitat temporal, però s'ha pensat que no és important el temps, però si que ho és simplificar el model per si es vol examinar què està passant i per tant fos senzill.

Una vegada es té tots els possibles tractaments que se li poden subministrar al pacient amb la seva efectivitat, es retornen ordenats de major coeficient d'adequació a menor coeficient d'adequació tots aquells que tinguin un coeficient d'adequació major al mínim de V. Recordem que el coeficient d'adequació serà un valor dintre del conjunt V que s'hagi definit.

Per altre banda hi ha altres funcions que podem trobar en el capítol de l'API en l'apèndix d'aquest treball, que ens han d'ajudar a entendre millor el que s'està calculant, o per si es volen fer càlculs més simplificats o concrets. Moltes de les funcions que s'han definit en l'apartat de *Model Abstracte* també s'han definit en el programa, per ser exacutades, i que retornin el que s'espera. Es remet el lector a l'apèndix, per mirar quines són aquestes funcions i com utilitzar-les.

5 - CONJUNTS D'EXEMPLES

En aquest capítol demostrarem el funcionament del programa proposat muntant dos conjunts d'exemples. En el primer conjunt d'exemples es farà servir un model d'exemple que ens trobàvem en la referència [1] i donarem els resultats pels tres casos dels pacients que es donaven. El segon conjunt d'exemples fa servir un model més complet on es reflexen més casos i demostra la capacitat del programa d'una manera més exhaustiva. En aquest segon conjunt d'exemples modificarem paràmetres relatius als càlculs del model, es faran servir valors quantitatius i qualitatius per les etiquetes de les relacions, i veurem les diferències que pot provocar al treure el resultat de tot això. També es modificaran els paràmetres que defineixen els operadors $\oplus$ i $\otimes$ per veure quins efectes té.

L'objectiu d'aquest capítol també és ajudar a qui vulgui utilitzar aquest programa per entendre'l millor, ja que s'intenta que sigui exhaustiu i es trobin tots els casos possibles que hom es pot trobar al utilitzar aquest programa, apart de que també trobarà els exemples de com construir el model a partir de les dades donades. El lector podrà observar els comentaris que es fan, però també serà útil que ell mateix investigui perquè es donen aquests resultats, i que faci servir totes les funcions de l'API per entendre-ho.

En els exemples que veurem ens fixarem amb el conjunt de solucions que es produiran, no només en la millor. De fet en els exemples que anirem veient hi hauran petits canvis puntuals en els paràmetres d'entrada així com en els paràmetres de càlcul del model, i es podrà veure com afecta això al conjunt de solucions.

5.1 - CONJUNT D'EXEMPLES 1. PNEUMONIA

Primer farem l'exemple que es presenta en [1]. De fet els bacteris que s'escriuen són tots provocadors d'una pneumonia, així doncs tots els casos de que parlarem seran pacients amb pneumonia. Per els valors del conjunt V s'han triat valors qualitatius {*null, low,med,high*} amb *null* el valor null. El mínim és *low* i el màxim és *high*. Així, els tres grafs que es relacionen són els següents:

Antibiòtics **Bacteris**	**Amox/Clav**	**Ceftriaxone**	**Cefepime**	**Cefuroxime**	**Ceftazidime**	**Amikacin**	**Gentamicin**	**Aztreonam**
Pneumococcus	High	High	High	Med	Med			
E.Coli	High	High	High	High	High	Low	Med	High
Psudomonas			Med		Med	High	Low	high

Graf de sensibilitat farmacològica

Antibiòtics **Efectes secundaris**	**Amox/Clav**	**Ceftriaxone**	**Cefepime**	**Cefuroxime**	**Ceftazidime**	**Amikacin**	**Gentamicin**	**Aztreonam**
Risk_when_penicillin_allergy	High							
Diarrhea	Med							
Eosinophilia		Med	Low		Med			Med
Anemia				Med				
Risk_when_pregnancy		High	High			Med	Med	
Renal_malfunction						High	High	

Graf d'efectes secundaris

Paràmetres **Efectes secundaris**	**Allergic_to_penicillin**	**Pregnant**	**Light_renal_failure**	**Severe_renal_failure**	**Transplanted**
Risk_when_penicillin_allergy	High				
Diarrhea					
Eosinophilia					
Anemia					
Risk_when_pregnancy		High			
Renal_malfunction			Low	High	High

Graf d'efectes secundaris a evitar

Totes aquestes dades representen el món que coneixem, i s'han de transcriure a un fitxer, que és el *fitxerEntrada1_1.txt* que podem trobar en l'apèndix. Com confeccionar aquest fitxer s'explica també en l'apèndix, així que es remet al lector a llegir aquest capítol per entendre com s'ha d'escriure. Per aquest conjunt d'exemples farem servir sempre aquest fitxer, i s'ha definit al fitxer que els tractaments estaran compostos per com a molt 2 antibiòtics. Per altra banda també s'ha definit que els operadors $\oplus$ i $\otimes$ facin la funció de *max* i *min* respectivament, és a dir, sense que funcionin d'una manera especial.

Una vegada definit el fitxer, presentem tres casos de tres pacients diferents, i veurem quin tractament ens diu el programa que li podem subministrar a cada un d'ells. També animem al lector, si vol investigar més i entendre el que passa amb cada cas, a utilitzar les funcions que es proposen que estan a la API, i que es poden utilitzar amb l'objectiu de trobar més el perquè de cada resultat.

5.1.1 - Cas 1. Pacient embarassada

El diagnòstic d'aquest cas és que la pacient pot tenir qualsevol dels tres bacteris que tenim representats en el model (e.coli, pneumococcus o pseudomonas) i apart tenim que està embarassada. Així doncs s'ha d'executar la funció *Find_treatments* de la següent manera.

- (Find_treatments '(e.coli pneumococcus pseudomonas) '((pregnant yes)) "c:/fitxerEntrada1_1.txt")

El primer paràmetre de la funció és la llista de bacteris que pot tenir la pacient. El segon paràmetre és una llista de llistes, on les llistes interiors són els paràmetres i el valor dels paràmetres que té la pacient. En aquest model que s'ha descrit tots els paràmetres són booleans, i per tant només s'haurà de posar *yes* si el té, o *no* si no el té, però no posar-lo, és com posar *no*. El tercer paràmetre és la ruta del fitxer que ha de carregar.

La sortida d'aquest cas és la següent:

"TRACTAMENT: " AZTREONAM AMOX/CLAV EFICÀCIA: HIGH
"TRACTAMENT: " CEFTAZIDIME EFICÀCIA: MED
"TRACTAMENT: " CEFTAZIDIME CEFUROXIME EFICÀCIA: MED
"TRACTAMENT: " AZTREONAM CEFUROXIME EFICÀCIA: MED
"TRACTAMENT: " CEFTAZIDIME AMOX/CLAV EFICÀCIA: MED
"TRACTAMENT: " AZTREONAM CEFTAZIDIME EFICÀCIA: MED
"TRACTAMENT: " GENTAMICIN AMOX/CLAV EFICÀCIA: LOW
"TRACTAMENT: " AMIKACIN AMOX/CLAV EFICÀCIA: LOW
"TRACTAMENT: " AMIKACIN CEFTAZIDIME EFICÀCIA: LOW
"TRACTAMENT: " GENTAMICIN CEFUROXIME EFICÀCIA: LOW
"TRACTAMENT: " AMIKACIN CEFUROXIME EFICÀCIA: LOW
"TRACTAMENT: " GENTAMICIN CEFTAZIDIME EFICÀCIA: LOW

Com veiem la solució surt ordenada de millor a pitjor, i el millor tractament és la combinació dels antibiòtics *aztreonam* i *amox/clav*, que tenen un coeficient d'adequació màxim per aquesta pacient. Si mirem en el graf de sensibilitat farmacològica, entre aquests dos antibiòtics poden combatre a tots tres bacteris. Per altra banda, en el graf d'efectes secundaris, també veiem que no hi ha cap relació entre aquest dos antibiòtics amb l'efecte secundari *Pregnant*, que és l'únic que pot es pot produir.

Anem a veure per exemple l'antibiòtic *Ceftazidime* perquè té un coeficient d'adequació de *med*. Veiem que en el graf d'efectes secundaris, *ceftazidime* no té cap relació amb *pregnant*, això vol dir que no produirà efectes secundaris. Però en el graf de sensibilitat farmacològica, *ceftazidime* té una relació amb *E.Coli* i *pseudomonas* de high, però amb *pneumococcus* només de *med*. Això fa, que en conjunt, combati els tres bacteris amb un coeficient d'adequació de *med*, i conjuntament amb els efectes secundaris que pot tenir (i que hem vist que no en té), el coeficient d'adequació final d'aquest tractament és *med*.

5.1.2 - Cas 2. Pacient amb insuficiència renal severa

En aquest cas, com l'anterior el diagnòstic d'aquest cas és que el pacient pot tenir qualsevol dels tres bacteris que tenim representats en el model (e.coli, pneumococcus o

pseudomonas) i apart tenim que té una insuficiència renal severa. Així doncs s'ha d'executar la funció *Find_treatments* de la següent manera.

- (Find_treatments '(e.coli pneumococcus pseudomonas) '((sever_renal_failure yes)) "c:/fitxerEntrada1_1.txt")

La sortida del programa és la següent:

"TRACTAMENT: " AZTREONAM CEFTRIAXONE EFICÀCIA: HIGH
"TRACTAMENT: " AZTREONAM AMOX/CLAV EFICÀCIA: HIGH
"TRACTAMENT: " AZTREONAM CEFEPIME EFICÀCIA: HIGH
"TRACTAMENT: " CEFTAZIDIME EFICÀCIA: MED
"TRACTAMENT: " CEFEPIME EFICÀCIA: MED
"TRACTAMENT: " CEFUROXIME CEFEPIME EFICÀCIA: MED
"TRACTAMENT: " CEFTAZIDIME AMOX/CLAV EFICÀCIA: MED
"TRACTAMENT: " CEFTAZIDIME CEFUROXIME EFICÀCIA: MED
"TRACTAMENT: " CEFEPIME AMOX/CLAV EFICÀCIA: MED
"TRACTAMENT: " CEFTAZIDIME CEFTRIAXONE EFICÀCIA: MED
"TRACTAMENT: " CEFEPIME CEFTRIAXONE EFICÀCIA: MED
"TRACTAMENT: " CEFTAZIDIME CEFEPIME EFICÀCIA: MED
"TRACTAMENT: " AZTREONAM CEFUROXIME EFICÀCIA: MED
"TRACTAMENT: " AZTREONAM CEFTAZIDIME EFICÀCIA: MED

Com veiem en aquest cas, hi ha tres possibles tractaments diferents que tenen un coeficient d'adequació màxim. Pel mateix raonament aplicat en el cas anterior, podem examinar que els resultats són correctes.

5.1.3 - Cas 3. Pacient amb insuficiència renal lleu

Tornem a tenir un cas on els possibles bacteris són els tres que tenim al model, i ara l'únic problema que té el pacient és que té una insuficiència renal lleu. Executem:

- (Find_treatments '(e.coli pneumococcus pseudomonas) '((ligth_renal_failure yes)) "c:/fitxerEntrada1_1.txt")

La sortida del programa és la següent:

"TRACTAMENT: " AZTREONAM CEFEPIME EFICÀCIA: HIGH
"TRACTAMENT: " AZTREONAM AMOX/CLAV EFICÀCIA: HIGH
"TRACTAMENT: " AZTREONAM CEFTRIAXONE EFICÀCIA: HIGH
"TRACTAMENT: " CEFTAZIDIME EFICÀCIA: MED
"TRACTAMENT: " CEFEPIME EFICÀCIA: MED
"TRACTAMENT: " AMIKACIN CEFUROXIME EFICÀCIA: MED
"TRACTAMENT: " AZTREONAM CEFUROXIME EFICÀCIA: MED
"TRACTAMENT: " CEFTAZIDIME CEFUROXIME EFICÀCIA: MED
"TRACTAMENT: " AMIKACIN AMOX/CLAV EFICÀCIA: MED
"TRACTAMENT: " CEFUROXIME CEFEPIME EFICÀCIA: MED
"TRACTAMENT: " CEFEPIME CEFTRIAXONE EFICÀCIA: MED
"TRACTAMENT: " GENTAMICIN CEFTAZIDIME EFICÀCIA: MED
"TRACTAMENT: " AMIKACIN CEFEPIME EFICÀCIA: MED
"TRACTAMENT: " AMIKACIN CEFTRIAXONE EFICÀCIA: MED
"TRACTAMENT: " CEFEPIME AMOX/CLAV EFICÀCIA: MED
"TRACTAMENT: " CEFTAZIDIME CEFTRIAXONE EFICÀCIA: MED
"TRACTAMENT: " CEFTAZIDIME AMOX/CLAV EFICÀCIA: MED
"TRACTAMENT: " AMIKACIN CEFTAZIDIME EFICÀCIA: MED
"TRACTAMENT: " GENTAMICIN CEFEPIME EFICÀCIA: MED
"TRACTAMENT: " CEFTAZIDIME CEFEPIME EFICÀCIA: MED
"TRACTAMENT: " AZTREONAM CEFTAZIDIME EFICÀCIA: MED
"TRACTAMENT: " GENTAMICIN CEFUROXIME EFICÀCIA: LOW
"TRACTAMENT: " GENTAMICIN AMOX/CLAV EFICÀCIA: LOW
"TRACTAMENT: " GENTAMICIN CEFTRIAXONE EFICÀCIA: LOW

Com veiem les solucions són semblants a l'anterior, però tenim uns quants tractaments més que donen uns coeficients d'èxit més baixos. Si ens fixem en el graf d'efectes secundaris a evitar, *Ligth_renal_failure* i *Severe_renal_failure* tenen les mateixes relacions, però *Ligth_renal_failure* amb un grau més baix, per això hi ha més tractaments a aplicar-se però que igualment no poden arribar aquests nous tractaments a tenir un coeficient d'adequació màxim.

5.2 - CONJUNT D'EXEMPLES 2

Aquest conjunt d'exemples pretén ser molt més exhaustiu que l'anterior, i demostrar fins on arriba l'eficàcia del model, i també les seves limitacions. Per aquest segon conjunt d'exemples farem servir les dades de la següents taules que representen els tres grafs que defineixen el model. Aquestes dades han estat extretes TERAPI-IA, excepte unes dades del graf d'efectes secundaris a evitar, on es relacionen els paràmetres *age* i *height* amb efectes secundaris, que han sigut inventades, tot i que amb cert sentit, és a dir, que si no són veritat, s'apropen al que és realitat. Aquestes dades s'han afegit per completar el conjunt d'exemples, i fer-lo per tant més atractiu.

	Diarrhea	**Eosinophilia**	**Anemia**	**Risk when pregnancy**	**Renal malfunction**	**Rash**	**Risk penicillin allergy**	**Otoxicity**	**Coombs**
Ampicil·lina	High	High				Low			
Amoxicil·lina	Med	Low				Low			
Amoxi/clav	Med					Low	High		
Amikacina				Med	High			High	
Cefepime	Low	Low		High		Low			High
Cefoxitina	Low	Low			Low				Low
Ceftazidime		Med				Low			Med
Ceftriaxona	Low	Med				Low			
Cefuroxima		High	High						
Gentamicina				Med	High			High	

Graf de sensibilitat farmacològica.

	Anaer.	H inf.	Morax.	Pneum.	Strep.	Staph.	E. Coli	Kleb.	Enterob.	Pseud.
Ampicil·lina				low	high					
Amoxicil·lina				high	High					
Amoxi/clav		High	High	High	High	High	Low	Med		
Amikacina		Med	Med			Med	High	High	High	High
Cefepime		High	High	High	High		High	High		Med
Cefoxitina	Med	High	High	Med	High	Med	High	High		
Ceftazidima		High	High	Med	Hjigh		High	High		Med
Ceftriaxona		High	High	High	High		High	High		
Cefuroxima IV		High	High	Med	High	Med	High	High		
Gentamicina		Med	Med			Med	Med	High	High	Low

Graf de d'efectes secundaris.

	Diarrhea	Eosinophilia	Anemia	Risk when pregnancy	Renal malfunction	Rash	Risk penicillin allergy	Otoxicity	Coombs
Age	0-5 high 15-55 null 65-100 med	0-10 high 15-100 null				0-10 high 15-80 null 85-100 med		0-5 high 15 –100 low	0-5 high 15-100 low
Height	0-40 med 55-150 null		0-50 high 55-150 low						
Allergic-to Pencillin							High		
Pregnant				High					
Transplanted					High				
Renal Failure					Màx: High				

Graf de d'efectes secundaris a evitar.

El graf d'efectes secundaris a evitar té en aquest exemple reflexats en les seves relacions els tres tipus de paràmetres del pacient. Els paràmetres *Allergic-to-Penicillin*, *Pregnant* i *Transplanted*, són del tipus booleà. Així que si el pacient té aquests paràmetres, les relacions que surten d'aquests mantindran els graus que veiem reflexats. El paràmetre *Renal Failure* és del tipus en que un pacient en pot tenim en més gravetat o menys. Ho

veiem reflexat en el grau de les seves relacions amb la paraula clau *max*, que recordem que indica que com en el pitjor cas, la relació ha de tenir el grau que indica, però que segons la gravetat d'aquest paràmetre en el cas de cada pacient, el grau final que han de tenir les relacions que surten d'aquest paràmetre poden canviar. *Age* i *height* són paràmetres numèrics, i que en les relacions que surten d'aquests paràmetres es defineix marges en el que es posa quin grau han de tenir les relacions. Però com no es posen tots els marges, es veu que estem davant un conjunt difús. Depenent de com es defineixi la funció final (que es deixa a l'usuari que ho faci en cada relació per separat)

que doni un grau per cada edat o pes en el nostre cas, i donat evidentment el pes i l'edat de cada pacient, es decidirà un grau concret per cada relació.

Ara presentem un seguit de diferents execucions del programa, tant canviat paràmetres del fitxer d'entrada on es defineixen els operadors $\oplus$ i $\otimes$ i les funcions de les relacions del graf d'efectes secundaris a evitar en cas de paràmetres com *age* i *height*, com canviat els graus de les relacions per valors qualitatius a valor quantitatius, com canviant també el cas de cada pacient, i veurem que passa en cada cas.

5.2.1 - Exemple 1

Per aquest primer exemple els operadors agafen la següent forma: $\oplus$ = *max* i $\otimes$ =*min* és a dir, el camp *changing_factor* de les classes *factor_[+]* i *factor[*]* amb valor 0. Els graus de les relacions seran de tipus qualitatiu con s'indica en les taules anteriors.

Així doncs amb totes aquestes dades podem formar el fitxer d'entrada de dades "fitxerEntrada2_1.txt", que podem trobar a l'Apèndix d'aquest treball. Ens podem fixar com s'han introduït totes les relacions, pot ajudar a entendre exactament com omplir el fitxer. Recomanem especialment observar les relacions del graf d'efectes secundaris a evitar.

5.2.1.1 - Pacient 1

Suposem que el diagnosi imprecís del pacient ens diu que pot tenir el següent conjunt de microorganismes: {e.coli, pneumococcus,morax.} i els paràmetres del pacient són els següents:{Age=20, height=70,pregnant=no, transplanted=yes, allergic-to_penicillin=no, renal_failure=low}. Així doncs, si "fitxerEntrada2_1.txt" el tenim a la ruta "c:/", la comanda que hem d'executar per trobar uns tractmanets adequats a aquest pacient és la següent.

- (Find_treatments '(e.coli pneumococcus morax) '((age 20)(height 70)(transplanted yes)(renal_failure low)) "c:/file.txt")

La sortida és la següent:

"TRACTAMENT: " CEFTRIAXONE EFICÀCIA: HIGH
"TRACTAMENT: " CEFTRIAXONE AMOX/CLAV EFICÀCIA: HIGH
"TRACTAMENT: " CEFTRIAXONE AMPICIL·LINA EFICÀCIA: HIGH
"TRACTAMENT: " CEFTRIAXONE AMOXICIL·LINA EFICÀCIA: HIGH
"TRACTAMENT: " CEFEPIME EFICÀCIA: MED
"TRACTAMENT: " CEFUROXIME EFICÀCIA: MED
"TRACTAMENT: " CEFTAZIDIME EFICÀCIA: MED
"TRACTAMENT: " CEFOXITINA EFICÀCIA: MED
"TRACTAMENT: " CEFUROXIME CEFOXITINA EFICÀCIA: MED
"TRACTAMENT: " CEFTAZIDIME AMOX/CLAV EFICÀCIA: MED
"TRACTAMENT: " CEFEPIME AMOXICIL·LINA EFICÀCIA: MED
"TRACTAMENT: " CEFUROXIME AMOXICIL·LINA EFICÀCIA: MED
"TRACTAMENT: " CEFUROXIME AMOX/CLAV EFICÀCIA: MED
"TRACTAMENT: " CEFOXITINA CEFEPIME EFICÀCIA: MED
"TRACTAMENT: " CEFTAZIDIME CEFOXITINA EFICÀCIA: MED
"TRACTAMENT: " CEFTRIAXONE CEFOXITINA EFICÀCIA: MED
"TRACTAMENT: " CEFOXITINA AMOX/CLAV EFICÀCIA: MED
"TRACTAMENT: " CEFOXITINA AMPICIL·LINA EFICÀCIA: MED
"TRACTAMENT: " CEFOXITINA AMOXICIL·LINA EFICÀCIA: MED
"TRACTAMENT: " CEFTAZIDIME CEFTRIAXONE EFICÀCIA: MED
"TRACTAMENT: " CEFTAZIDIME CEFEPIME EFICÀCIA: MED
"TRACTAMENT: " CEFUROXIME CEFTRIAXONE EFICÀCIA: MED
"TRACTAMENT: " CEFEPIME AMPICIL·LINA EFICÀCIA: MED
"TRACTAMENT: " CEFTAZIDIME AMOXICIL·LINA EFICÀCIA: MED
"TRACTAMENT: " CEFUROXIME AMPICIL·LINA EFICÀCIA: MED
"TRACTAMENT: " CEFTAZIDIME AMPICIL·LINA EFICÀCIA: MED
"TRACTAMENT: " CEFEPIME AMOX/CLAV EFICÀCIA: MED
"TRACTAMENT: " CEFTRIAXONE CEFEPIME EFICÀCIA: MED
"TRACTAMENT: " CEFUROXIME CEFEPIME EFICÀCIA: MED
"TRACTAMENT: " CEFTAZIDIME CEFUROXIME EFICÀCIA: MED
"TRACTAMENT: " AMOX/CLAV EFICÀCIA: LOW
"TRACTAMENT: " AMOX/CLAV AMPICIL·LINA EFICÀCIA: LOW
"TRACTAMENT: " AMOX/CLAV AMOXICIL·LINA EFICÀCIA: LOW

Com veiem, només aplicant Ceftriaxone en tindríem prou per curar el pacient, ja que té un coeficient d'adequació màxim. Si mirem el graf de sensibilitat farmacològica, efectivament l'antibiòtic Ceftriaxone combat amb la màxima efectivitat el conjunt de bacteris {e.coli,pneumococcus,morax}. Per altre banda els seus paràmetres clínics podrien fer efecte, com veiem al graf d'efectes secundaris a evitar al conjunt {*anemia* (des del paràmetre *height*), *renal_malfunction* (des del paràmetre *transplanted*)}, i com veiem l'antibiòtic Ceftriaxone no té relacions amb cap d'aquests membres d'aquest conjunt en el graf d'efectes secundaris.

5.2.1.2 - Pacient 2

Ara fem el mateix cas que l'anterior però per un nen de 4 anys, que com veiem és molt més propens a tenir efectes secundaris. Executem la següent comanda i els coeficients d'èxit seran més pobres:

- Find_treatments '(e.coli pneumococcus morax) '((age 4)(height 20)(transplanted yes)) "c:/file.txt")

```
"TRACTAMENT: " CEFOXITINA   EFICÀCIA: MED
"TRACTAMENT: " AMOX/CLAV   EFICÀCIA: LOW
"TRACTAMENT: " CEFTRIAXONE   EFICÀCIA: LOW
"TRACTAMENT: " CEFTAZIDIME   EFICÀCIA: LOW
"TRACTAMENT: " CEFTAZIDIME AMOXICIL·LINA   EFICÀCIA: LOW
"TRACTAMENT: " CEFTAZIDIME AMOX/CLAV   EFICÀCIA: LOW
"TRACTAMENT: " CEFOXITINA AMOX/CLAV   EFICÀCIA: LOW
"TRACTAMENT: " CEFTRIAXONE CEFOXITINA   EFICÀCIA: LOW
"TRACTAMENT: " CEFOXITINA AMOXICIL·LINA   EFICÀCIA: LOW
"TRACTAMENT: " AMOX/CLAV AMOXICIL·LINA   EFICÀCIA: LOW
"TRACTAMENT: " CEFTRIAXONE AMOX/CLAV   EFICÀCIA: LOW
"TRACTAMENT: " CEFTAZIDIME CEFOXITINA   EFICÀCIA: LOW
"TRACTAMENT: " CEFTRIAXONE AMOXICIL·LINA   EFICÀCIA: LOW
"TRACTAMENT: " CEFTAZIDIME CEFTRIAXONE   EFICÀCIA: LOW
```

Administrar Cefoxitina és el millor tractament en aquest cas. Ara l'antibiòtic Ceftazidime té una eficàcia de *low* degut a l'edat del pacient, que provoca molts efectes secundaris, com l'Eosinophilia i que l'antibiòtic Ceftazidime pot provocar. L'antibiòtic Cefoxitina es lliura, com podem comprovar a les taules, de tot efecte secundari en

aquest cas, però com en la taula de sensibilitat farmacològica veiem que només combat amb una eficàcia de *med* al bacteri *pneumococcus*, per això aquest coeficient d'adequació de *med*.

5.2.1.3 - Pacient 3

Fem un altre exemple, suposem que ara el nen té 13 anys. Amb 13 anys continua sent sensible a molts efectes secundaris, però en quin grau dependrà també de com s'hagi definit la funció en les diferents relacions. Estem en una edat difusa de cara a dir molts dels graus de les relacions del graf d'efectes secundaris a evitar, i és per això que dependrà de com s'hagin definit les funcions d'aquestes relacions. Veiem que en aquest exemple els valors dels camps *funcio_inferior* i *funcio_superior* valien 1.5 o 2 en tots els casos, o sigui de cada tros de cada funció de cada relació del tipus que ho demanava. Executem i les solucions són:

- (Find_treatments '(e.coli pneumococcus morax) '((age 13)(height 50)(transplanted yes)) "c:/file.txt")

"TRACTAMENT: " CEFOXITINA EFICÀCIA: MED
"TRACTAMENT: " CEFOXITINA AMOXICIL·LINA EFICÀCIA: MED
"TRACTAMENT: " CEFOXITINA AMOX/CLAV EFICÀCIA: MED
"TRACTAMENT: " CEFEPIME EFICÀCIA: LOW
"TRACTAMENT: " AMOX/CLAV EFICÀCIA: LOW
"TRACTAMENT: " CEFUROXIME EFICÀCIA: LOW
"TRACTAMENT: " CEFTRIAXONE EFICÀCIA: LOW
"TRACTAMENT: " CEFTAZIDIME EFICÀCIA: LOW
"TRACTAMENT: " CEFTAZIDIME CEFOXITINA EFICÀCIA: LOW
"TRACTAMENT: " CEFEPIME AMOX/CLAV EFICÀCIA: LOW
"TRACTAMENT: " CEFEPIME AMPICIL·LINA EFICÀCIA: LOW
"TRACTAMENT: " CEFUROXIME AMOXICIL·LINA EFICÀCIA: LOW
"TRACTAMENT: " CEFTAZIDIME CEFEPIME EFICÀCIA: LOW
"TRACTAMENT: " CEFTRIAXONE CEFEPIME EFICÀCIA: LOW
"TRACTAMENT: " CEFTAZIDIME AMOXICIL·LINA EFICÀCIA: LOW
"TRACTAMENT: " CEFTRIAXONE AMOXICIL·LINA EFICÀCIA: LOW
"TRACTAMENT: " AMOX/CLAV AMOXICIL·LINA EFICÀCIA: LOW
"TRACTAMENT: " CEFTRIAXONE AMOX/CLAV EFICÀCIA: LOW
"TRACTAMENT: " CEFTAZIDIME AMOX/CLAV EFICÀCIA: LOW
"TRACTAMENT: " CEFTAZIDIME AMPICIL·LINA EFICÀCIA: LOW
"TRACTAMENT: " CEFOXITINA CEFEPIME EFICÀCIA: LOW
"TRACTAMENT: " CEFUROXIME CEFEPIME EFICÀCIA: LOW
"TRACTAMENT: " CEFUROXIME AMOX/CLAV EFICÀCIA: LOW
"TRACTAMENT: " CEFUROXIME AMPICIL·LINA EFICÀCIA: LOW
"TRACTAMENT: " AMOX/CLAV AMPICIL·LINA EFICÀCIA: LOW

"TRACTAMENT: " CEFTRIAXONE CEFOXITINA EFICÀCIA: LOW
"TRACTAMENT: " CEFTAZIDIME CEFTRIAXONE EFICÀCIA: LOW
"TRACTAMENT: " CEFEPIME AMOXICIL·LINA EFICÀCIA: LOW
"TRACTAMENT: " CEFTRIAXONE AMPICIL·LINA EFICÀCIA: LOW
"TRACTAMENT: " CEFUROXIME CEFOXITINA EFICÀCIA: LOW
"TRACTAMENT: " CEFUROXIME CEFTRIAXONE EFICÀCIA: LOW
"TRACTAMENT: " CEFOXITINA AMPICIL·LINA EFICÀCIA: LOW
"TRACTAMENT: " CEFTAZIDIME CEFUROXIME EFICÀCIA: LOW

Hi ha més possibles tractaments que en el cas anterior, però indubtablament que el pacient tingui 13 i no 15 anys (Pacient 1) es nota.

Ara poder ens agradaria saber què està passant exactament. Podem utilitzar les funcions que la API ens posa a la nostra disposició. Hi ha una funció molt útil, que ens dona molta informació de com s'ha entès el model que és *printa_info_arestes*. La podem executar tot seguit d'haver executat *find_treatments* perquè el model no s'ha esborrat i està tal i com l'ha carregat i modificat. Si l'executem ens imprimeix el següent.

from: HEIGHT to: ANEMIA value: HIGH
from: HEIGHT to: DIARRHEA value: LOW
from: AGE to: COOMBS value: MED
from: AGE to: OTOXICITY value: MED
from: AGE to: RASH value: MED
from: AGE to: EOSINOPHILIA value: MED
from: AGE to: DIARRHEA value: LOW
from: TRANSPLANTED to: RENAL_MALFUNCTION value: HIGH
from: GENTAMICIN to: OTOXICITY value: HIGH
from: GENTAMICIN to: RENAL_MALFUNCTION value: HIGH
from: GENTAMICIN to: RISK_WHEN_PREGNANCY value: MED
from: CEFUROXIME to: ANEMIA value: HIGH
from: CEFUROXIME to: EOSINOPHILIA value: HIGH
from: GENTAMICIN to: E.COLI value: MED
from: GENTAMICIN to: MORAX value: MED
from: CEFUROXIME to: E.COLI value: HIGH
from: CEFUROXIME to: PNEUMOCOCCUS value: MED
from: CEFUROXIME to: MORAX value: HIGH
from: CEFTRIAXONE to: E.COLI value: HIGH
from: CEFTRIAXONE to: PNEUMOCOCCUS value: HIGH
from: CEFTRIAXONE to: MORAX value: HIGH
from: CEFTAZIDIME to: E.COLI value: HIGH
from: CEFTAZIDIME to: PNEUMOCOCCUS value: MED
from: CEFTAZIDIME to: MORAX value: HIGH
from: CEFOXITINA to: E.COLI value: HIGH
from: CEFOXITINA to: PNEUMOCOCCUS value: MED
from: CEFOXITINA to: MORAX value: HIGH
from: CEFEPIME to: E.COLI value: HIGH
from: CEFEPIME to: PNEUMOCOCCUS value: HIGH
from: CEFEPIME to: MORAX value: HIGH
from: AMIKACINA to: E.COLI value: HIGH
from: AMIKACINA to: MORAX value: MED
from: AMOX/CLAV to: E.COLI value: LOW

from: AMOX/CLAV to: PNEUMOCOCCUS value: HIGH
from: AMOX/CLAV to: MORAX value: HIGH
from: AMOXICIL·LINA to: PNEUMOCOCCUS value: HIGH
from: AMPICIL·LINA to: PNEUMOCOCCUS value: LOW
from: CEFTAZIDIME to: COOMBS value: MED
from: CEFTAZIDIME to: RASH value: LOW
from: CEFTAZIDIME to: EOSINOPHILIA value: MED
from: CEFUROXIME to: ANEMIA value: MED
from: CEFTRIAXONE to: RISK_WHEN_PREGNANCY value: HIGH
from: CEFTRIAXONE to: EOSINOPHILIA value: MED
from: CEFOXITINA to: COOMBS value: LOW
from: CEFOXITINA to: RENAL_MALFUNCTION value: LOW
from: CEFOXITINA to: EOSINOPHILIA value: LOW
from: CEFOXITINA to: DIARRHEA value: LOW
from: CEFEPIME to: COOMBS value: HIGH
from: CEFEPIME to: RASH value: LOW
from: CEFEPIME to: RISK_WHEN_PREGNANCY value: HIGH
from: CEFEPIME to: DIARRHEA value: LOW
from: CEFEPIME to: EOSINOPHILIA value: LOW
from: AMIKACINA to: OTOXICITY value: HIGH
from: AMIKACINA to: RENAL_MALFUNCTION value: HIGH
from: AMIKACINA to: RISK_WHEN_PREGNANCY value: MED
from: AMOX/CLAV to: RASH value: LOW
from: AMOX/CLAV to: DIARRHEA value: MED
from: AMOX/CLAV to: RISK_PENICILLIN_ALLERGY value: HIGH
from: AMOXICIL·LINA to: RASH value: LOW
from: AMOXICIL·LINA to: DIARRHEA value: MED
from: AMPICIL·LINA to: EOSINOPHILIA value: HIGH
from: AMPICIL·LINA to: DIARRHEA value: HIGH

Aquí veiem totes les relacions que hi ha en el model després d'haver aplicat el cas que ens ocupa i el grau d'aquestes. Primer de tot veiem que en aquí falten moltes relacions. Això és fruit de l'optimització que hi hagut en el model per aquest cas específic. Això ens assegura que totes les relacions que aquí surten imprimides són d'interès pel nostre cas. Per altre banda és interessant observar com han quedat els valors de les etiquetes de les relacions del graf d'efectes secundaris a evitar on participa el paràmetre *age*. Per altra banda pot passar que el model és molt gran, i només ens interessa tenir el valor d'una o dues relacions. Podríem executar la següent comanda, amb la pertinent sortida:

- (Printa_info_arestes (list (find_edge 'age 'rash)))

from: AGE to: RASH value: MED

5.2.2 - Exemple 2

En aquest exemple pretenem fer el mateix que l'anterior però ara les etiquetes qualititatives passaran a ser quantitatives, així doncs fem servir un altre fitxer d'entrada el fitxer "fitxerEntrada2_2.txt" que també podem trobar a l'apèndix. De fet és calcat a l'anterior, però ara les etiquetes de les relacions seran quantitatives amb un rang de {0..3} amb valor null = 0, i mínim=1 i farem les substitucions low=1, med=2, high=3, per passar del fitxer anterior al nou fitxer. Tot lo altre és el mateix. I tot seguit presentem els resultats d'aplicar el mateix cas que l'anterior, el del pacient 3, però ara amb aquestes etiquetes quantitatives:

- (Find_treatments '(e.coli pneumococcus morax) '((age 13)(height 50)(transplanted yes)) "c:/file2.txt")

"TRACTAMENT: " CEFOXITINA EFICÀCIA: 2
"TRACTAMENT: " CEFOXITINA AMOXICIL·LINA EFICÀCIA: 2
"TRACTAMENT: " CEFOXITINA AMOX/CLAV EFICÀCIA: 2
"TRACTAMENT: " CEFEPIME EFICÀCIA: 1.3788854381999833
"TRACTAMENT: " CEFOXITINA CEFEPIME EFICÀCIA: 1.3788854381999833
"TRACTAMENT: " CEFEPIME AMOX/CLAV EFICÀCIA: 1.3788854381999833
"TRACTAMENT: " CEFEPIME AMOXICIL·LINA EFICÀCIA: 1.3788854381999833
"TRACTAMENT: " CEFTAZIDIME EFICÀCIA: 1.3589466384404112
"TRACTAMENT: " CEFTRIAXONE EFICÀCIA: 1.3589466384404112
"TRACTAMENT: " CEFTAZIDIME AMOXICIL·LINA EFICÀCIA: 1.3589466384404112
"TRACTAMENT: " CEFEPIME AMPICIL·LINA EFICÀCIA: 1.3589466384404112
"TRACTAMENT: " CEFTRIAXONE AMPICIL·LINA EFICÀCIA: 1.3589466384404112
"TRACTAMENT: " CEFTAZIDIME CEFEPIME EFICÀCIA: 1.3589466384404112
"TRACTAMENT: " CEFTAZIDIME AMOX/CLAV EFICÀCIA: 1.3589466384404112
"TRACTAMENT: " CEFTAZIDIME AMPICIL·LINA EFICÀCIA: 1.3589466384404112
"TRACTAMENT: " CEFTRIAXONE CEFOXITINA EFICÀCIA: 1.3589466384404112
"TRACTAMENT: " CEFTRIAXONE AMOX/CLAV EFICÀCIA: 1.3589466384404112
"TRACTAMENT: " CEFTRIAXONE CEFEPIME EFICÀCIA: 1.3589466384404112
"TRACTAMENT: " CEFTAZIDIME CEFOXITINA EFICÀCIA: 1.3589466384404112
"TRACTAMENT: " CEFTRIAXONE AMOXICIL·LINA EFICÀCIA: 1.3589466384404112
"TRACTAMENT: " CEFTAZIDIME CEFTRIAXONE EFICÀCIA: 1.3589466384404112
"TRACTAMENT: " CEFOXITINA AMPICIL·LINA EFICÀCIA: 1.3589466384404112
"TRACTAMENT: " AMOX/CLAV EFICÀCIA: 1
"TRACTAMENT: " CEFUROXIME EFICÀCIA: 1
"TRACTAMENT: " CEFUROXIME CEFOXITINA EFICÀCIA: 1
"TRACTAMENT: " CEFUROXIME CEFTRIAXONE EFICÀCIA: 1
"TRACTAMENT: " AMOX/CLAV AMOXICIL·LINA EFICÀCIA: 1
"TRACTAMENT: " CEFUROXIME AMOX/CLAV EFICÀCIA: 1
"TRACTAMENT: " AMOX/CLAV AMPICIL·LINA EFICÀCIA: 1
"TRACTAMENT: " CEFUROXIME AMOXICIL·LINA EFICÀCIA: 1
"TRACTAMENT: " CEFUROXIME AMPICIL·LINA EFICÀCIA: 1
"TRACTAMENT: " CEFUROXIME CEFEPIME EFICÀCIA: 1
"TRACTAMENT: " CEFTAZIDIME CEFUROXIME EFICÀCIA: 1

Comparant amb el cas del pacient 3 de l'exemple anterior, veiem els resultats que abans s'havien arrodonit per donar un resultat dintre del conjunt de valors qualitatius que hi havia, i ara veiem els resultats amb decimals. Evidentment l'aproximació que ens dona l'intèrpret no hauria de ser tant aproximada, en que ens fixem en el primer decimal n'hi ha prou. Amb aquest exemple es volia fer notar la diferència d'utilitzar valors quantitatius o valors qualitatius.

5.2.3 - Exemple 3

Els camps que omplen *fucio_inf* i *funcio_sup* de la taula consideration, o sigui cada tros de la funció que defineixen algunes de les relacions del graf d'efectes secundaris a evitar estaven a 1.5 o 2. Bé, aquests eren uns valors raonables on la funció si va de baixada, baixa menys al principi per acabar baixant més al final, o si va de pujada, puja més al principi i menys al final. Ara canviem això i definim un fitxer on tots valors que omplen els paràmetres *fucio_inf* i *funcio_sup* de la taula consideration valdran 1, o sigui, les funcions que marquin els valors de la relacions tant quan pugui com quan baixi seran de forma lineal. Aquest nou fitxer "fitxerEntrada2_3.txt" manté totes les propietats que guardava el fitxer de l'exemple anterior excepte el canvi esmentat. Ara presentem els resultats del mateix cas del pacient 3. Els resultats són els següents:

- (Find_treatments '(e.coli pneumococcus morax) '((age 13)(height 50)(transplanted yes)) "c:/file3.txt")

"TRACTAMENT: " CEFOXITINA EFICÀCIA: 2
"TRACTAMENT: " CEFOXITINA AMOXICIL·LINA EFICÀCIA: 2
"TRACTAMENT: " CEFOXITINA AMOX/CLAV EFICÀCIA: 2
"TRACTAMENT: " CEFTRIAXONE EFICÀCIA: 9/5
"TRACTAMENT: " CEFTRIAXONE AMOX/CLAV EFICÀCIA: 9/5
"TRACTAMENT: " CEFTRIAXONE AMOXICIL·LINA EFICÀCIA: 9/5
"TRACTAMENT: " CEFOXITINA AMPICIL·LINA EFICÀCIA: 9/5
"TRACTAMENT: " CEFTRIAXONE AMPICIL·LINA EFICÀCIA: 9/5
"TRACTAMENT: " CEFTRIAXONE CEFOXITINA EFICÀCIA: 9/5
"TRACTAMENT: " CEFEPIME EFICÀCIA: 8/5
"TRACTAMENT: " CEFTAZIDIME EFICÀCIA: 8/5
"TRACTAMENT: " CEFEPIME AMPICIL·LINA EFICÀCIA: 8/5
"TRACTAMENT: " CEFTAZIDIME CEFOXITINA EFICÀCIA: 8/5
"TRACTAMENT: " CEFTAZIDIME CEFTRIAXONE EFICÀCIA: 8/5
"TRACTAMENT: " CEFTRIAXONE CEFEPIME EFICÀCIA: 8/5

"TRACTAMENT: " CEFEPIME AMOXICIL·LINA EFICÀCIA: 8/5
"TRACTAMENT: " CEFEPIME AMOX/CLAV EFICÀCIA: 8/5
"TRACTAMENT: " CEFTAZIDIME AMOX/CLAV EFICÀCIA: 8/5
"TRACTAMENT: " CEFTAZIDIME AMOXICIL·LINA EFICÀCIA: 8/5
"TRACTAMENT: " CEFTAZIDIME AMPICIL·LINA EFICÀCIA: 8/5
"TRACTAMENT: " CEFTAZIDIME CEFEPIME EFICÀCIA: 8/5
"TRACTAMENT: " CEFOXITINA CEFEPIME EFICÀCIA: 8/5
"TRACTAMENT: " AMOX/CLAV EFICÀCIA: 1
"TRACTAMENT: " CEFUROXIME EFICÀCIA: 1
"TRACTAMENT: " CEFUROXIME CEFOXITINA EFICÀCIA: 1
"TRACTAMENT: " CEFUROXIME CEFTRIAXONE EFICÀCIA: 1
"TRACTAMENT: " AMOX/CLAV AMOXICIL·LINA EFICÀCIA: 1
"TRACTAMENT: " CEFUROXIME AMOX/CLAV EFICÀCIA: 1
"TRACTAMENT: " AMOX/CLAV AMPICIL·LINA EFICÀCIA: 1
"TRACTAMENT: " CEFUROXIME AMOXICIL·LINA EFICÀCIA: 1
"TRACTAMENT: " CEFUROXIME AMPICIL·LINA EFICÀCIA: 1
"TRACTAMENT: " CEFUROXIME CEFEPIME EFICÀCIA: 1
"TRACTAMENT: " CEFTAZIDIME CEFUROXIME EFICÀCIA: 1

Com veiem els tractaments que estaven entre un coeficient d'adequació de 1 i 2 han millorat sensiblement. Això és degut al canvi que s'ha fet, ja que als 13 anys, dibuixant les gràfiques com s'han dibuixat en aquest cas, té menys efecte el paràmetre *age* a provocar efectes secundaris.

5.2.4 - Exemple 4

Per últim fem un últim exemple amb el fitxer "fitxerEntrada2_4.txt", on modifiquem els paràmetres de les instàncies a les classes *factor_[+]* i *factor[*]* que defineixen els paràmetres dels operadors $\oplus$ i $\otimes$. Posem un petit valor de 0.1 al paràmetre *changing_factor*, tant per l'operador $\oplus$ i $\otimes$. Com a valors restrictius utilitzem els camps *max_change*=1, *max_change_per_cent*=0.20, per les dues funcions. El més restrictiu serà *max_change_per_cent* en aquest cas. Totes les altres dades són com al fitxer "fitxerEntrada2_3.txt". Executem pel mateix cas del pacient 3 que veníem fent últimament.

- (Find_treatments '(e.coli pneumococcus morax) '((age 13)(height 50)(transplanted yes)) "c:/file4.txt")

"TRACTAMENT: " CEFTRIAXONE EFICÀCIA: 1.9028875477274805
"TRACTAMENT: " CEFTRIAXONE CEFOXITINA EFICÀCIA: 1.4433333333333334
"TRACTAMENT: " CEFTAZIDIME CEFTRIAXONE EFICÀCIA: 1.1165280971851854

"TRACTAMENT: " CEFTRIAXONE AMOX/CLAV EFICÀCIA: 1.1019135319096889
"TRACTAMENT: " CEFUROXIME EFICÀCIA: 1
"TRACTAMENT: " CEFTAZIDIME EFICÀCIA: 1
"TRACTAMENT: " CEFOXITINA EFICÀCIA: 1
"TRACTAMENT: " AMOX/CLAV EFICÀCIA: 1
"TRACTAMENT: " CEFTAZIDIME CEFOXITINA EFICÀCIA: 1
"TRACTAMENT: " AMOX/CLAV AMOXICIL·LINA EFICÀCIA: 1
"TRACTAMENT: " CEFTAZIDIME CEFEPIME EFICÀCIA: 1
"TRACTAMENT: " CEFUROXIME CEFOXITINA EFICÀCIA: 1
"TRACTAMENT: " CEFOXITINA AMOXICIL·LINA EFICÀCIA: 1
"TRACTAMENT: " CEFOXITINA AMOX/CLAV EFICÀCIA: 1
"TRACTAMENT: " CEFTRIAXONE AMIKACINA EFICÀCIA: 1
"TRACTAMENT: " CEFUROXIME CEFTRIAXONE EFICÀCIA: 1
"TRACTAMENT: " CEFTRIAXONE AMPICIL·LINA EFICÀCIA: 1
"TRACTAMENT: " CEFTAZIDIME AMOX/CLAV EFICÀCIA: 1
"TRACTAMENT: " CEFOXITINA AMPICIL·LINA EFICÀCIA: 1
"TRACTAMENT: " CEFTRIAXONE AMOXICIL·LINA EFICÀCIA: 1
"TRACTAMENT: " CEFTAZIDIME AMOXICIL·LINA EFICÀCIA: 1
"TRACTAMENT: " CEFTAZIDIME CEFUROXIME EFICÀCIA: 1

Com veiem els resultats en general han baixat. Això és pel següent: Majoritàriament s'han vist efectuades al canvi de resultat operacions amb l'operador $\otimes$ i que calculaven *contx_adequacy*, o sigui els efectes secundaris que pot portar un tractament, i com hem fet que aquest operador doni resultats per sota de la funció *min*, per això els coeficients d'èxit han baixat.

6 - CONCLUSIONS I TREBALL FUTUR

Com en el capítol d'objectius s'ha dit, amb el programa de validació d'un model que s'ha defensat en aquest projecte no podrem atacar casos reals, almenys casos reals amb un pèl de complicació, que en la majoria n'hi hauria. De tota manera, sí que resol casos simplificats, i és la base d'un programa que pot arribar a funcionar en hospitals. Primerament s'explica en aquest capítol totes les variables, casos i complexitats que el model descrit no és capaç d'enfrontar-s'hi, però també es pretén dir si és molt complicat implementar-les i com es pot arribar a fer. També, es criticarà els punts febles d'aquest model i del programa que valida el model. A més a més s'exposaran possibles millores d'aquest mateix model, vistos els errors que té després d'haver fet molts exemples. Finalment s'exposen les conclusions personals que en trec d'aquest projecte.

Hi ha un munt de casos que en el model abstracte que s'ha definit no s'han tingut en compte. Una és que dos antibiòtics poden xocar entre ells, és a dir, hi ha incompatibilitats entre antibiòtics, que poden provocar greus efectes secundaris. En aquest cas la manera de resoldre això no seria molt complicat, tot i que dir això és agoserat ja que sempre poden haver casos concrets que et compliquin molt el model. De tota manera s'hauria de crear un nou tipus de relació, un nova classe que heretés de la classes *Edge*, i que relacionés dos nodes de tipus antibiòtic. La relació hauria d'indicar si aquests dos antibiòtics són compatibles o no. Llavors s'hauria de posar una condició en el model que tingués en compte aquestes incompatibilitats definides d'aquesta forma.

Una qüestió important que no s'ha tractat és el tema de les dosis dels antibiòtics i el temps en aquests s'han de subministrar. En tot cas, la duració no és molt important, ja tindria temps el metge de pensar-ho, però sí que ho és la dosi. Tractar les dosis pot ser molt dificultós. En un cas simplificat, un antibiòtic perquè faci l'efecte que ha de fer s'ha de subministrar en una certa dosi. Doncs bé, s'emmagatzema aquest paràmetre en el model i ja està. Però també pot ser, que com més dosi d'un antibiòtic es subministri, més riscos pot provocar, o si més no, més creixen els riscos que hi havia. També pot ser que a un pacient, segons els paràmetres que tingui (l'edat, per exemple), només se li

pugui subministrar un dosi màxima determinada d'aquest antibiòtic. Si només es tenen en compte aquests casos de les dosis (en poden haver més) es podria realitzar de la següent manera: Les relacions de tipus *pharmacological_sensitivity_graph* hauran de tenir una funció associada, que segons la dosi del tractament que s'apliqui, quin valor ha de tenir dita relació, o sigui, amb quin grau combati l'antibiòtic al bacteri. El mateix ha de passar amb les relacions *side_effects_graph*, que també han de tenir una funció, que segons la dosi de l'antibiòtic indiqui amb quin grau afecta a tal efecte secundari. I per últim s'hauria d'escriure una nova relació entre paràmetres i antibiòtics, en que s'indiqués si el valor del paràmetre té alguna restricció amb la dosi de l'antibiòtic que relaciona. El valor de la relació, dependria doncs de la dosi de l'antibiòtic i del valor del paràmetre. En definitiva aquest és un cas complex. I no tant sols representar-ho com es veu, si no encara més difícil seria implementar tot l'algorisme que tingués en compte aquests paràmetres. La millor manera segurament seria calcular la dosi òptima de cada antibiòtic abans de fer els càlculs.

Un altre tema important són els operadors que hem vist en el model. Bàsicament, els operadors $\oplus$ i $\otimes$. En aquest treball s'ha dedicat una part important en explicar el funcionament que es proposa per aquests operadors. I després de sofisticar-los tant com s'ha fet, s'ha vist que poder no feia tanta falta, apart de que s'ha fet d'una manera molt equivocada. Defenso l'ús de les classes *factor*, *factor[+]* i *factor[*]* per representar uns paràmetres que indiquin el funcionament dels operadors $\oplus$ i $\otimes$. Però no són bones les dues funcions que s'han proposat en aquest projecte *rectifica_dependent* i *rectifica_independent* (que eren dues maneres diferents d'interpretar els operadors $\oplus$ i $\otimes$ i presentaven un algorisme que definia el funcionament d'aquests), i no són bones perquè no compleixen la propietat d'associativitat que han de tenir els operadors $\oplus$ i $\otimes$. Tot i això crec que la idea que defensen aquestes dues funcions és bona, però no s'ha portar a terme d'una manera adequada. És per això que recomano que en un principi s'utilitzin els operadors $\oplus$ i $\otimes$ com si fossin les funcions *max* i *min*, però també animo a qui segueixi aquest projecte, a que estudii les idees originals de les funcions *rectifica_dependent* i *rectifica_independent* i les pugui portar a terme fent altres funcions que sí compleixin totes les propietats $\oplus$ i $\otimes$ i les idees que defensen *rectifica_dependent* i *rectifica_independent* .

Igualment, crec que és millor que l'operador $\oplus$ s'apropi molt a la funció *max*, o que doni un valor sensiblement superior, i que la funció $\otimes$ s'apropi molt a la funció *min*, i que en tot cas doni un valor sensiblement inferior. Per altre banda crec que en alguns casos on s'apliquen els operadors $\oplus$ i $\otimes$, directament s'hauria d'aplicar la funció *max* i *min* respectivament. És a dir, en alguns casos té més sentit que d'altres que dits operadors agafin una forma diferent a les funcions *max* i *min*, que en d'altres. En l'apartat on es defensa que els operadors agafin una forma diferent a les funcions $\oplus$ i $\otimes$, s'explica el perquè, però ara aquí s'explica el perquè a vegades té més sentit que aquests operadors siguin directament la funció *max* i *min*. Quan es fa la operació:

$$global\_adequacy(T_j \mid C) = pharm\_adequacy(T_j \mid A_i) \otimes contx\_adequacy(T_j \mid P_i).$$

seria més adequat fer la operació:

$$\text{global\_adequacy}(T_j \mid C) = \min(\text{pharm\_adequacy}(T_j \mid A_i), \text{contx\_adequacy}(T_j \mid P_i)).$$

pharm_adequacy i *contx_adequacy* són dos càlculs totalment diferents, i estem dient que *global_adequacy* serà el resultat del mínim valor de les dues funcions anteriors. No té perquè baixar més el valor en aquest cas respecte el que donaria la funció *min*, a diferència d'altres casos on s'utilitza l'operador $\otimes$, tal i com s'ha explicat en capítols anteriors.

De tota manera el ideal seria que la funció concreta que desenvolupessin els operadors $\oplus$ i $\otimes$ fos estàtica, que no calgués que l'usuari ho determini. Però per ara s'havia pensat així, ja que si un usuari experimentat que entén perfectament el model utilitzava el programa, pogués donar ell forma a aquests operadors. Tot i així crec que s'han de definir la funció d'aquests operadors concretament, i a més defenso que segons quins operants es facin servir (o sigui, depèn què s'estigui calculant) pot ser que els operadors $\oplus$ i $\otimes$ haguessin d'agafar formes diferents, és per això, que s'hauria d'estudiar cada operació que fa el model per separat, i definir un operador diferent per cada operació.

Representar els paràmetres del pacient i relacionar-los amb els antibiòtics i els efectes secundaris és probablement el més complicat. En aquest projecte s'han tractat tres tipus

de paràmetres, i sempre s'han relacionat directament amb els efectes secundaris. Aquests tres tipus de paràmetres són necessaris que es puguin definir. Per altra banda hi ha altres casos que aquest model no contempla. Hi ha casos que depenen de diferents paràmetres, o sigui, es pot ser propens a un efecte secundari, o no, depenent de diversos paràmetres. És a dir, seria necessari fer relacions entre paràmetres. Tot i així la complexitat que els paràmetres representen no es solucionaria fàcilment. En l'apartat que més s'hauria de treballar és aquest.

Per últim fer esment a un últim aspecte. Hi ha antibiòtics que a vegades reaccionen contra un efecte secundari, i a vegades no, però quan reaccionen ho fan ho amb un efecte determinat. Llavors en el model que hem definit, per curar-nos en salut, hauríem de posar el pitjor cas, o sigui el cas en que reacciona de la manera més contundent. Però hi ha un altre dada que es pot saber, i que en aquest model tampoc hi té lloc encara, i és la probabilitat de que un antibiòtic reaccioni a un efecte secundari. Crec que això seria necessari representar-ho, i finalment les solucions haurien de ser un balança de l'efectivitat de combatre els microorganismes que tenia el pacient, amb la probabilitat de que tingui efectes secundaris o no. Hi ha molts antibiòtics que reaccionen només en l'1% de casos en provocar efectes secundaris, i és bo això tenir-ho en compte, perquè en un cas greu, i si no hi ha més remei, s'hauran de fer servir antibiòtics que puguin provocar efectes secundaris al pacient, i seria bo tenir la probabilitat de que això passi.

A part de tot el que he dit fins ara, vull fer unes recomanacions més a qui vulgui continuar aquest programa que valida el model. Sobretot que es llegeixi bé la memòria, i ho entengui. Mentrestant també es pot mirar el codi, aquest està força comentat, i està escrit d'una manera senzilla i estructurada, sense funcions complicades, i d'una manera genèrica i uniforme. Aleshores la seva feina serà informar-se molt bé de les dades que un metge disposa i de com les maneja. Segurament haurà de treballar amb un metge durant un temps. Crec que aquest projecte és molt ampliable, i pot arribar a bon port.

Vull concloure amb les conclusions personals d'aquest projecte. Dir que me'n agradat moltes parts d'aquest, i estic orgullós del que he fet. He dedicat moltes estones a pensar quines solucions concretes serien les millors pels diferents aspectes del treball. Ho he

trobat molt interessant, i només he trobat a faltar temps per fer-ho millor i representar i portar a terme moltes de les coses que he dit en aquest capítol que s'haurien de fer. He après més sobre el llenguatge CLOS, a representar el coneixement que s'obté, a utilitzar els recursos de la lògica dels conjunt difusos, he après a fer un cosa ben feta i des del principi, passant per tots els passos que requereix fer un projecte, no com molts de les pràctiques que he hagut de fer durant la carrera. També m'ha servit per veure la dificultat de fer un projecte tant ambiciós. De fet, qui continuï aquest treball, encara té molta feina.

7 - BIBLIOGRAFIA

Referències

[1] Lluís Godó, Josep Puyol Gruart, Sandra Sandri i Pilar Barrufet. *Assessing adequacy and risk of drugs in treatments for imprecise clinical diagnoses*. Article originari d'aquest projecte on s'explica el problema a resoldre i el model de càlcul.

[2] *Guide to antimicrobial therapy 2002-03*. Guia d'ajuda a identificar malalties infeccioses i com actuar per combatre-les.

[3] P.Barrufet. *TERÀP-IA: Sistema expert d'ajuda al tractament de pneumonies. PhD thesis,* Universitat Autònoma de Barcelona, 2000. Tesis on es proposa un sistema expert per resoldre casos de pneumonies.

Referències de pàgines web.

[4] http://www.lispworks.com Es pot trobar manuals, història, i programes sobre LISP i tots els llenguatges derivat d'aquest com CLOS.

[5] http://en.wikipedia.org/wiki/Dendral Explica qué es Dendral, que pretén resoldre, com funciona i com va néixer, d'una manera resumida però exacta.

[6]http://www.gratisweb.com/comarsc/inar4.htm Petita aproximació en castellà dels sistemes experts Eliza, Mycin i Dendral.

[7] http://hpslab.stanford.edu/projects/Bioinformatics/Archives/SecondarySite/Lederberg.pdf . Article interessant que explica el naixement de Dendral d'una forma extensa, es situa en la època i en l'equip de persones que el van fabricar. També disposa d'una interessant bibliografia.

[8] http://en.wikipedia.org/wiki/Mycin Explica que és Mycin, i d'una manera curta el seu funcionament.

[9] http://www.aaaipress.org/Classic/Buchanan/buchanan.html Extens treball sobre Mycin, on d'una manera extensa explica el seu naixement, formació, funcionament, aplicacions, i l'èxit que ha tingut.

[10] http://www.aaai.org/AITopics/html/expert.html Es poden trobar els diferents sistemes experts amb interessants enllaços que han revolucionat el món de la IA.

[11] http://en.wikipedia.org/wiki/Triangular_norm#T-conorms . Parla de les T-conormes i T-normes, i per tant dels operadors $\oplus$ i $\otimes$.

[12] http://www.aiai.ed.ac.uk/~jeff/clos-guide.html Tutorial per aprendre a programar en CLOS.

[13] http://en.wikipedia.org/wiki/CLOS Explicació de la història i propietats del llenguatge CLOS.

8 - APÈNDIX

Aquest apèndix està destinat pel lector que vulgui aprofundir en el projecte i estigui interessat en utilitzar el programa que s'ha creat.

En primer lloc trobem el codi del programa, i que fa totes les funcionalitats que s'han explicat. Està degudament comentat, i tot i que un programa escrit en CLOS no és molt agradable de llegir, s'ha seguit un estil determinat i cada funció té una capçalera on s'explica que fa, així com l'explicació de la forma dels paràmetres i el significat.

Els següents capítols estan destinats a com fer servir el programa, recomanacions o com transcriure les dades perquè el programa ho entengui.

En últim lloc presentem els fitxers que s'han utilitzat pels conjunts d'exemples que s'han presentat en aquest treball. Animem a qui vulgui aprofundir i aprendre a utilitzar el programa, que comenci per entendre a seguir pas a pas els exemples que s'han presentat en aquest treball, i veure també com s'han construint els fitxers d'entrada.

8.1 - CODI

```
;Emmagatzemen totes les instàncies de nodes i arestes respectivament.
(defparameter list_of_nodes nil)
(defparameter list_of_edges nil)


;Defineix el nombre maxim de tractaments a subministrar
(defparameter max_treatments 2)

;Serà una instància de la classe factor_[+]
(defparameter fact_[+] nil)
;Serà una instància de la classe factor_[+]
(defparameter fact_[*] nil)



;Definició de totes les classes necessàries per representar els grafs que
;es pretenen dibuixar.

;Classe node. Representa un element que és node de qualsevol graf.
;   name -> Nom del node.
;   p_edges -> Llista d'arestes que estan dirigides en aquest node.
;   edges_p -> Llista d'arestes que surten d'aquest node.
(defclass node ()
        (
                (name
                        :accessor node-name
                        :initform nil
                        :initarg :name)
                (p_edges
                        :accessor node-p_edges
                        :initform nil
                        :initarg :p_edges)
                (edges_p
                        :accessor node-edges_p
                        :initform nil
                        :initarg :edges_p)
        )
)


;Classe T_node. Hereda de la classe node. Representa els Tractament.
(defclass T_node (node)
        ()
)

;Classe A_node. Hereda de la classe node. Representa els Agents infecciosos.
(defclass A_node (node)
        ()
)

;Classe E_node. Hereda de la classe node. Representa els Efectes secundaris.
(defclass E_node (node)
        ()
)

;Classe P_node. Hereda de la classe node. Representa els Paràmetres.
(defclass P_node (node)
        ()
)


;Classe aresta. Representa una relació entre dos nodes qualsevols de qualsevol graf.
;   from -> Node del qual prové l'aresta.
;   to -> Node al qual està dirigida l'aresta.
;   label -> Etiqueta de l'aresta.
(defclass edge ()
        (
                (from
                        :accessor edge-from
                        :initform nil
```

```
                        :initarg :from)

                (to
                        :accessor edge-to
                        :initform nil
                        :initarg :to)

                (label
                        :accessor edge-label
                        :initform nil
                        :initarg :label)
        )
)


;Classe pharmacological_sensitivity_graph. Hereda de la classe edge.
;Representa les arestes del graf de sensitivitat farmacològica.
(defclass pharmacological_sensitivity_graph (edge)
        ()
)

;Classe side_effects_graph. Hereda de la classe edge.
;Representa les arestes del graf d'efectes secundaris.
(defclass side_effects_graph (edge)
        ()
)

;Classe effects_to_avoid_graph. Hereda de la classe edge.
;Representa les arestes d'efectes secundaris a evitar.
(defclass effects_to_avoid_graph (edge)
        ( )
)

;Classe label representa l'etiqueta de qualsevol edge.
;   value -> És el valor que pren l'etiqueta.
;   minim -> És el valor mínim que pot arribar a prendre l'etiqueta.
;   label -> Etiqueta de l'aresta.
(defclass label ()
        (
                (value
                        :accessor label-value
                        :initform nil
                        :initarg :value
                )
                (minim
                        :accessor label-minim
                        :initform 'low
                        :initarg :minim
                )
                (maxim
                        :accessor label-maxim
                        :initform 'high
                        :initarg :maxim
                )
                (null
                        :accessor label-null
                        :initform 'null
                        :initarg :null
                )
                (considerations
                        :accessor label-considerations
                        :initform nil
                        :initarg :consideration
                )
    )
)

(defclass consideration()
 (
                (value_ini
                        :accessor consideration-value_ini
                        :initform nil
                        :initarg :value_ini
                )
                (funcio_inferior
                        :accessor consideration-funcio_inferior
```

```
                        :initform nil
                        :initarg :funcio_inferior
                )
                (funcio_superior
                        :accessor consideration-funcio_superior
                        :initform nil
                        :initarg :funcio_superior
                )
                (sota_llindar_inferior
                        :accessor consideration-sota_llindar_inferior
                        :initform nil
                        :initarg :sota_llindar_inferior
                )
                (llindar_inferior
                        :accessor consideration-llindar_inferior
                        :initform nil
                        :initarg :llindar_inferior
                )
                (llindar_superior
                        :accessor consideration-llindar_superior
                        :initform nil
                        :initarg :llindar_superior
                )
                (sobre_llindar_superior
                        :accessor consideration-sobre_llindar_superior
                        :initform nil
                        :initarg :sobre_llindar_superior
                )
        )
)

;Classe quantitative_label. Hereda de la classe label.
;Representa les etiquetes que tenen un valor quantitatiu.
(defclass quantitative_label (label)
        ()
)

;Classe qualitative_label. Hereda de la classe label.
;Representa les etiquetes que tenen un valor qualitatiu.
;  elements -> Llista de tots els valors ordenats de petit a major que pot
;                   prendre l'etiqueta.
(defclass qualitative_label (label)
        (
                (elements
                        :accessor qualitative_label-elements
                        :initform '(null low med high)
                        :initarg :elements
                )
        )
)


;Estructura factor: Son les dades que s'utilitzaran per completar les funcions _[+] i _[*]
; Parametres:
;       changing_factor: Nombre que representa el canvi proporcional per cada etiqueta.
;       max_change:  Maxim canvi que es pot fer absolut.
;               En el cas d'etiquetes qualitatives, es conten el nombre de posicions.
;       max_change_per_cent_[+]: ;El maxim canvi que es pot fer,
;                       proporcional a la longitud de la llista. (valors entre 0 i 1)
(defclass factor ()
        (
                (changing_factor
                        :accessor factor-changing_factor
                        :initform nil
                        :initarg :changing_factor)
                (max_change
                        :accessor factor-max_change
                        :initform nil
                        :initarg :max_change)
                (max_change_per_cent
                        :accessor factor-max_change_per_cent
                        :initform nil
                        :initarg :max_change_per_cent)
        )
)
```

```
(defclass factor_[+] (factor)
	()
)

(defclass factor_[*] (factor)
	()
)



;Mètode make___label.
;Paràmetres:
;	value -> Valor de la nova etiqueta.
;	instancia -> Etiqueta model, de la qual copiarà tota la informació excepte
;		el valor d'aquesta.
;Retorna: Una etiqueta igual que instancia però amb el valor del paràmetre valor.
(defmethod make__label (value (instancia qualitative_label))
	(make-instance 'qualitative_label :value value
					:minim (label-minim instancia)
					:maxim (label-maxim instancia)
					:null (label-null instancia)
					:elements (qualitative_label-elements instancia)
	)
)



;Mètode make___label.
;Paràmetres:
;	value -> Valor de la nova etiqueta.
;	instancia -> Etiqueta model, de la qual copiarà tota la informació excepte
;		el valor d'aquesta.
;Retorna: Una etiqueta igual que instancia però amb el valor del paràmetre valor.
(defmethod make__label (value (instancia quantitative_label))
	(make-instance 'quantitative_label
					:value value :minim (label-minim instancia)
					:maxim (label-maxim instancia)
					:null (label-null instancia)
	)
)



;Funció node_exists.
;Paràmetres:
;	name -> nom d'un node.
;	l   -> (Opcional) Llista de nodes. Per defecte agafa la llista de nodes general.
;Retorna: Un boleà indicant si existeix un node en la llista que té el nom indicat.
;Donat un nom i una llista de nodes, retorna un boleà indicant si aquest nom està en algun dels nodes de la llista
(defun node_exists (name &optional (l list_of_nodes))
	(cond
		((eq l nil) nil)
		((equal name (node-name (car l))) t)
		(T (node_exists name (cdr l)))
	)
)


;Funció	add_treatments.
;Paràmetres:
;	name -> nom d'un node.
;	p_edges -> (Opcional) Llista d'instàncies d'arestes que van al node.
;	edges_p -> (Opcional) Llista d'instàncies d'arestes que surten del node.
;Fa:	Crea una nova instància de node de tipus tractament i l'afegeix a la llista de nodes.
;Retorna: La instància del node creat.
(defun add_treatments(name &optional (p_edges nil) (edges_p nil))
	(if (eq nil (node_exists name list_of_nodes))
		(setf list_of_nodes
			(cons
				(make-instance 'T_node :name name :p_edges p_edges :edges_p edges_p)
				list_of_nodes
			)
		)
	)
	(find_node name)
```

```
)


;Funció     add_agents.
;Paràmetres:
;              name -> nom d'un node.
;              p_edges -> (Opcional) Llista d'instàncies d'arestes que van al node.
;              edges_p -> (Opcional) Llista d'instàncies d'arestes que surten del node.
;Fa:           Crea una nova instància de node de tipus agents infecciosos
;              i l'afegeix a la llista de nodes.
;Retorna: La instància del node creat.
(defun add_agents(name &optional (p_edges nil) (edges_p nil))
              (if (eq nil (node_exists name list_of_nodes))
                            (setf list_of_nodes
                                          (cons
                                                        (make-instance 'A_node :name name :p_edges p_edges :edges_p edges_p)
                                                        list_of_nodes
                                          )
                            )
              )
              (find_node name)
)


;Funció     add_secundary_effect.
;Paràmetres:
;              name -> nom d'un node.
;              p_edges -> (Opcional) Llista d'instàncies d'arestes que van al node.
;              edges_p -> (Opcional) Llista d'instàncies d'arestes que surten del node.
;Fa:           Crea una nova instància de node de tipus efecte secundari
;              i l'afegeix a la llista de nodes.
;Retorna: La instància del node creat.
(defun add_secundary_effect(name &optional (p_edges nil) (edges_p nil))
              (if (eq nil (node_exists name list_of_nodes))
                            (setf list_of_nodes
                                          (cons
                                                        (make-instance 'E_node :name name :p_edges p_edges :edges_p edges_p)
                                                        list_of_nodes
                                          )
                            )
              )
              (find_node name)
)


;Funció     add_parameter.
;Paràmetres:
;              name -> nom d'un node.
;              p_edges -> (Opcional) Llista d'instàncies d'arestes que van al node.
;              edges_p -> (Opcional) Llista d'instàncies d'arestes que surten del node.
;Fa:           Crea una nova instància de node de tipus paràmetre del pacient
;              i l'afegeix a la llista de nodes.
;Retorna: La instància del node creat.
(defun add_parameter(name &optional (p_edges nil) (edges_p nil))
              (if (eq nil (node_exists name list_of_nodes))
                            (setf list_of_nodes
                                          (cons
                                                        (make-instance 'P_node :name name :p_edges p_edges :edges_p edges_p)
                                                        list_of_nodes
                                          )
                            )
              )
              (find_node name)
)



;Funció find_node.
;Paràmetres:
;              name: Nom d'un node.
;              l: (Opcional) Llista de nodes, per defecte és la llista general de nodes.
;Retorna: Instància de node pertanyent a l
```

```
;               que té com a nom el nom que es passa per paràmetre.
(defun find_node (name &optional (l list_of_nodes))
        (cond
                ((eq l nil) nil)
                ((equal name (node-name (car l))) (car l))
                (T (find_node name (cdr l)))
        )
)



;Funció find_edge.
;Paràmetres:
;       from: Nom d'un node.
;       to: Nom d'un node.
;       l: (Opcional) Llista d'arestes, per defecte és la llista general d'arestes.
;Retorna: Instància d'aresta pertanyent a l que te el nom del node origen és from,
;         i el nom del node destí és to.
(defun find_edge(from to &optional (l list_of_edges))
        (cond
                ((eq l nil) nil)
                ((and (equal from (node-name (edge-from (car l)))) (equal to (node-name (edge-to (car l))))) (car l))
                (T (find_edge from to (cdr l)))
        )
)



;Funció  add_PS_edge.
;Paràmetres:
;       label -> Etiqueta valor.
;       from -> Nom d'un node.
;       to -> Nom d'un node.
;Fa:    Crea una nova instància d'aresta de sensitivitat farmacològica i l'afegeix a
;       a la llista d'arestes general, i en cas         de que els noms de nodes no existissin
;       els crea i els afegeix a la llista de nodes general. També afegeix als nodes from
;       i to l'aresta en els seus paràmetres respectius.
;Retorna: La instància de l'aresta creada.
(defun add_PS_edge (label from to)
        (setf list_of_edges
                (cons
                        (make-instance 'pharmacological_sensitivity_graph :label  label :from (add_treatments from) :to
(add_agents to))
                        list_of_edges
                )
        )
        (setf (node-edges_p (find_node from)) (cons (find_edge from to) (node-edges_p (find_node from))))
        (setf (node-p_edges (find_node to)) (cons (find_edge from to) (node-p_edges (find_node to))))
)


;Funció  add_SE_edge.
;Paràmetres:
;       label -> Etiqueta valor.
;       from -> Nom d'un node.
;       to -> Nom d'un node.
;Fa:    Crea una nova instància d'aresta d'efectes secundaris i l'afegeix a
;       a la llista d'arestes general, i en cas         de que els noms de nodes no existissin
;       els crea i els afegeix a la llista de nodes general. També afegeix als nodes from
;       i to l'aresta en els seus paràmetres respectius.
(defun add_SE_edge (label from to)
        (setf list_of_edges
                (cons
                        (make-instance 'side_effects_graph :label label :from (add_treatments from) :to
(add_secundary_effect to))
                        list_of_edges
                )
        )
        (setf (node-edges_p (find_node from)) (cons (find_edge from to) (node-edges_p (find_node from))))
        (setf (node-p_edges (find_node to)) (cons (find_edge from to) (node-p_edges (find_node to))))
)


;Funció  add_ETA_edge.
;Paràmetres:
;       label -> Etiqueta value.
```

```
;               from -> Nom d'un node.
;               to -> Nom d'un node.
;Fa:            Crea una nova instància d'aresta d'efectes a evitar i l'afegeix a
;               a la llista d'arestes general, i en cas              de que els noms de nodes no existissin
;               els crea i els afegeix a la llista de nodes general. També afegeix als nodes from
;               i to l'aresta en els seus paràmetres respectius.
(defun add_ETA_edge (label from to)
                (setf list_of_edges
                                (cons
                                                (make-instance 'effects_to_avoid_graph :label  label :from (add_parameter from) :to
(add_secundary_effect to))
                                                list_of_edges
                                )
                )
                (setf (node-edges_p (find_node from)) (cons (find_edge from to) (node-edges_p (find_node from))))
                (setf (node-p_edges (find_node to)) (cons (find_edge from to) (node-p_edges (find_node to))))
                (find_edge from to)
)


;Mètodes _>=, _>, _<=, _<.
;Paràmetres:
;               value1 -> Número.
;               value2 -> Número.
;Retorna la operació corresponent del nom de la funció aplicada a dos números.
(defmethod _>= ((value1 number) (value2 number))
                (>= value1 value2)
)

(defmethod _> ((value1 number) (value2 number))
                (> value1 value2)
)

(defmethod _<= ((value1 number) (value2 number))
                (<= value1 value2)
)

(defmethod _< ((value1 number) (value2 number))
                (< value1 value2)
)



;Mètodes _>=, _>, _<=, _<.
;Paràmetres:
;               label1 -> Instància d'etiqueta quantitativa.
;               label2 -> Instància d'etiqueta quantitativa.
;Retorna la operació corresponent del nom de la funció aplicada a les dues etiquetes.
(defmethod _>= ((label1 quantitative_label) (label2 quantitative_label))
                (>= (label-value label1) (label-value label2))
)

(defmethod _> ((label1 quantitative_label) (label2 quantitative_label))
                (> (label-value label1) (label-value label2))
)

(defmethod _<= ((label1 quantitative_label) (label2 quantitative_label))
                (<= (label-value label1) (label-value label2))
)

(defmethod _< ((label1 quantitative_label) (label2 quantitative_label))
                (< (label-value label1) (label-value label2))
)


;Mètodes _>=, _>, _<=, _<.
;Paràmetres:
;               label1 -> Instància d'etiqueta qualitativa.
;               label2 -> Instància d'etiqueta qualitativa.
;Retorna la operació corresponent del nom de la funció aplicada a les dues etiquetes.
(defmethod _>= ((label1 qualitative_label) (label2 qualitative_label))
                (>= (position (label-value label1) (qualitative_label-elements label1)) (position (label-value label2) (qualitative_label-
elements label2)))
)

(defmethod _> ((label1 qualitative_label) (label2 qualitative_label))
```

```
		(> (position (label-value label1) (qualitative_label-elements label1)) (position (label-value label2) (qualitative_label-elements label2)))
)

(defmethod _<= ((label1 qualitative_label) (label2 qualitative_label))
		(<= (position (label-value label1) (qualitative_label-elements label1)) (position (label-value label2) (qualitative_label-elements label2)))
)

(defmethod _< ((label1 qualitative_label) (label2 qualitative_label))
		(< (position (label-value label1) (qualitative_label-elements label1)) (position (label-value label2) (qualitative_label-elements label2)))
)



;Mètode _neg.
;Paràmetres:
;		label1 -> etiqueta qualitativa.
;Retorna: Una etiqueta amb valor negat respecte valor dintre del conjunt
;		que poden prendre les etiquetes qualitatives.
(defmethod _neg ((label1 qualitative_label) &aux list_ resultat)
		(setq list_ (reverse (qualitative_label-elements label1)))
		(dolist (e (qualitative_label-elements label1) resultat)
			(if (eq e (label-value label1))
				(setq resultat (car list_))
				(setq list_ (cdr list_))
			)
		)
		(make__label resultat label1)
)



;Mètode _neg.
;Paràmetres:
;		label1 -> etiqueta quantitativa.
;Retorna: Una etiqueta amb valor negat respecte valor dintre del conjunt
;		que poden prendre les etiquetes quantitatives.
(defmethod _neg ((label1 quantitative_label))
		(make__label (+ (- (label-maxim label1) (label-value label1)) (label-null label1)) label1)
)



;Funció my_label.
;Paràmetres:
;		value -> valor de l'etiqueta
;Retorna: Una etiqueta amb valor el que marqui el paràmetre value.
(defun my_label (value)
		(if (typep (edge-label (car list_of_edges)) 'quantitative_label)
			(progn
				(if (and (<= value (label-maxim (edge-label (car list_of_edges)))) (>= value (label-minim (edge-label (car list_of_edges)))))
					(make__label value (edge-label (car list_of_edges)))
					"El paràmetre no és correcte, no correspon amb els tipus de les etiquetes"
				)
			)
			(progn
				(if (member value (qualitative_label-elements (edge-label (car list_of_edges))))
					(make__label value (edge-label (car list_of_edges)))
					"El paràmetre no és correcte, no correspon amb els tipus de les etiquetes"
				)
			)
		)
)

;Funció the_null.
;Retorna: Una etiqueta amb valor null, del tipus d'etiquetes que s'estan utilitzant.
(defun the_null()
		(make__label (label-null (edge-label(car list_of_edges))) (edge-label (car list_of_edges)))
)



;Funció the_min.
```

```
;Retorna: Una etiqueta amb valor minim, del tipus d'etiquetes que s'estan utilitzant.
(defun the_min()
	(make__label (label-minim (edge-label(car list_of_edges))) (edge-label (car list_of_edges)))
)


;Funció the_max.
;Retorna: Una etiqueta amb valor maxim, del tipus d'etiquetes que s'estan utilitzant.
(defun the_max()
	(make__label (label-maxim (edge-label(car list_of_edges))) (edge-label (car list_of_edges)))
)


;Funció _max.
;Paràmetes:
;		Sèrie de nombres o etiquetes.
;Retorna: El paràmetre amb valor màxim.
(defun _max (a &rest l &aux resultat)
	(setq resultat a)
	(dolist (b l resultat)
		(if (_>= b resultat)
			(setq resultat b)
		)
	)
	resultat
)


;Funció _min.
;Paràmetes:
;		Sèrie de nombres o etiquetes.
;Retorna: El paràmetre amb valor mínim.
(defun _min (a &rest l &aux resultat)
	(setq resultat a)
	(dolist (b l resultat)
		(if (_<= b resultat)
			(setq resultat b)
		)
	)
	resultat
)



;Funció desp_label_right.
;Paràmetres:
;	label -> Etiqueta de tipus qualitatiu.
;	desp -> Nombre de desplaçaments que fara el valor de la etiqueta respecte el rang de possibilitats.
;Retorna: La mateixa etiqueta però amb el seu paràmetre valor, que haurà canviat a pitjor segons indica
;		el parametre desp, el desplaçament que fa.
(defmethod desp_label_right ((label qualitative_label) desp &aux pos)
	(setq pos (position (label-value label) (qualitative_label-elements label)))
	(setq pos (+ pos desp))
	(when (>= pos (length (qualitative_label-elements label)))
		(setq pos (- (length (qualitative_label-elements label)) 1))
	)
	(setf (label-value label) (car (nthcdr pos (qualitative_label-elements label))))
	label
)



;Funció desp_label_left.
;Paràmetres:
;	_label -> Etiqueta de tipus qualitatiu.
;	_desp -> Nombre de desplaçaments que fara el valor de la etiqueta respecte el rang de possibilitats.
;Retorna: La mateixa etiqueta però amb el seu paràmetre valor, que haurà canviat a millor segons indica
;		el parametre desp, el desplaçament que fa.
(defmethod desp_label_left ((label qualitative_label) desp &aux pos)
  (setq pos (position (label-value label) (qualitative_label-elements label)))
  (setq pos (- pos desp))
  (when (<= pos 0)
    (setq pos 0)
  )
  (setf (label-value label) (car (nthcdr pos (qualitative_label-elements label))))
  label
```

```
)



;Mètodes desp_label.
;Paràmetres:
;	label -> Etiqueta de tipus qualitatiu.
;	desp -> Nombre de desplaçaments que fara el valor de la etiqueta respecte el rang de possibilitats.
;	fact -> De tipus factor. Dades per fer l'algorisme.
;Retorna: La mateixa etiqueta però amb el seu paràmetre valor,
;	que haurà canviat a millor o a pitjor segons indica fact, i el desplaçament ho
;	indica el parametre desp.
(defmethod desp_label ((label qualitative_label) desp fact &aux resultat)
		(when (typep fact 'factor_[+])
			(setq resultat (desp_label_right label (round desp)))
		)
		(when (typep fact 'factor_[*])
			(setq resultat (desp_label_left label (round desp)))
		)
		resultat
)



(defmethod desp_label ((label quantitative_label) desp fact &aux pos)
		(when (typep fact 'factor_[+])
			(setq pos (+ (label-value label) desp))
		)
		(when (typep fact 'factor_[*])
			(setq pos (- (label-value label) desp))
		)
		(when (>= pos (label-maxim label))
			(setq pos (label-maxim label))
		)
		(when (<= pos (label-minim label))
			(setq pos (label-minim label))
		)
		(setf (label-value label) pos)
		label
)



;Mètodes pondera_value.
;Paràmetres:
;	label -> Etiqueta de tipus quantitatiu.
;	model_label -> Etiqueta de tipus quantitatiu.
;	fact -> De tipus factor.
;Retorna: el factor de canvi, si el valor de label i model_label son iguals. Com més diferent siguin, es retorna
;	el factor de canvi mes decrementat.
(defmethod pondera_value ((label quantitative_label) (model_label quantitative_label) fact &aux diference)
		(if (eq (label-value label) (label-value model_label))
			(factor-changing_factor fact)
			(progn
				(when (typep fact 'factor_[+])
					(setq diference
						(-
							(label-value model_label)
							(label-value label)
						)
					)
				)
				(when (typep fact 'factor_[*])
					(setq diference
						(-
							(label-value label)
							(label-value model_label)
						)
					)
				)
				(*
					(- 1 (/  diference (+ 1 (- (label-maxim label) (label-minim label)))))
					(factor-changing_factor fact)
				)
			)
```

```
		)
)


(defmethod pondera_value ((label qualitative_label) (model_label qualitative_label) fact &aux diference)
	(if (eq (label-value label) (label-value model_label))
		(factor-changing_factor fact)
		(progn
			(when (typep fact 'factor_[+])
				(setq diference
					(-
						(position (label-value model_label) (qualitative_label-elements
label))
						(position (label-value label) (qualitative_label-elements label))
					)
				)
			)
			(when (typep fact 'factor_[*])
				(setq diference
					(-
						(position (label-value label) (qualitative_label-elements label))
						(position (label-value model_label) (qualitative_label-elements
label))
					)
				)
			)
			(*
				(- 1 (/ diference (- (length (qualitative_label-elements label)) 1)))
				(factor-changing_factor fact)
			)
		)
	)
)


;Funció quit_model_label.
;Paràmetres:
;	l -> Llista de _labels.
;	model_label -> Un _label.
;Retorna: l excepte un dels seus paràmetres que tingui un valor igual al valor de model_label.
(defun quit_model_label (l model_label &aux (r t) (new_l nil))
	(dolist (e l r)
		(if (eq r t)
			(if (eq (label-value e) (label-value model_label))
				(setq r nil)
				(unless (eq (label-value e) (label-null e))
					(setq new_l (cons e new_l))
				)
			)
			(unless (eq (label-value e) (label-null e))
				(setq new_l (cons e new_l))
			)
		)
	)
	new_l
)


;Mètodes max_change_per_cent.
;Parmàtres:
;	_label -> Etiqueta.
;	fact -> De tipus factor. Dades per efectuar l'algorisme.
;Retorna: Depenent la forma de la etiqueta, retorna com pot arribar a transformar-se,
;	segons els parametres de l'estructura factor.
(defmethod max_change_per_cent ((_label quantitative_label) fact &aux len_total)
	(setq len_total (- (label-maxim _label) (label-minim _label)))
	(* (factor-max_change_per_cent fact) len_total)
)


(defmethod max_change_per_cent ((_label qualitative_label) fact &aux len_total)
	(setq len_total (length (qualitative_label-elements _label)))
	(* (factor-max_change_per_cent fact) len_total)
)
```

```
;Mètodes position_relative
;Paràmetres:
;       label -> etiqueta.
;Retorna:  La posició relativa del valor de l'etiqueta respecte totes els valors
;         que pot prendre. El valor que es retorna serà entre 0 i 1.
(defmethod position_relative ((label qualitative_label))
        (/
                (position (label-value label) (qualitative_label-elements label))
                (length (qualitative_label-elements label))
        )
)

(defmethod position_relative ((_label quantitative_label))
        (/
                (- (label-value _label) (label-minim _label))
                (- (label-maxim _label) (label-minim _label))
        )
)


;Funcio rectifica_independent_[+]
;Paràmetres:
;       l -> Llista d'etiquetes
;Retorna: L'algorisme està més explicat a la memòria.
(defun rectifica_independent_[+] (l)
        (rectifica_independent l (apply '_max l) fact_[+])
)


;Funcio rectifica_independent_[+]
;Paràmetres:
;       l -> Llista d'etiquetes
;Retorna: L'algorisme està més explicat a la memòria.
(defun rectifica_independent_[*] (l)
        (rectifica_independent l (apply '_min l) fact_[*])
)


;Funció rectifica_independent
;Paràmetres:
;       l -> Llista d'etiquetes.
;       model_label -> etiqueta.
;       fact -> De tipus factor, té els paràmetres en que es basa l'algorisme.
;Retorna: model_label es una etiqueta que té el maxim valor de totes les etiquetes
;       que hi ha a l. Es retorna una etiqueta amb valor igual o més gran a
;       model_label depenent de les etiquetes de l.
;       L'algorisme està més ben explicat a la memòria.
(defun rectifica_independent (l model_label fact &aux (r 0))
        (setq l (quit_model_label l model_label))
        (dolist (e l r)
                (setq r (+ (pondera_value e model_label fact) r))
        )
        (when (> r (factor-max_change fact))
                (setq r (factor-max_change fact))
        )
        (when (> r (max_change_per_cent model_label fact))
                (setq r (max_change_per_cent model_label fact))
        )
        (if (eq (label-value model_label) (label-null model_label))
                model_label
                (desp_label model_label r fact)
        )
)


;Funcio rectifica_dependent_[+]
;Paràmetres:
;       l -> Llista d'etiquetes
;Retorna: L'algorisme està més explicat a la memòria.
(defun rectifica_dependent_[+] (l)
        (rectifica_dependent l (apply '_max l) fact_[+])
)
```

```
;Funcio rectifica_dependent_[+]
;Paràmetres:
;		l -> Llista d'etiquetes
;Retorna: L'algorisme està més explicat a la memòria.
(defun rectifica_dependent_[*] (l)
		(rectifica_dependent l (apply '_min l) fact_[*])
)



;Funció rectifica_dependent;
;Paràmetres:
;	l -> Llista d'etiquetes.
;	model_label -> etiqueta.
;	fact -> De tipus factor, té els paràmetres en que es basa l'algorisme.
;Retorna: model_label es una etiqueta que té el maxim valor de totes les etiquetes
;	que hi ha a l. Es retorna una etiqueta amb valor igual o més gran a
;	model_label depenent de les etiquetes de l.
;	L'algorisme està més ben explicat a la memòria.
(defun rectifica_dependent (l model_label fact &aux (r 0))
		(setq l (quit_model_label l model_label))
		(dolist (e l r)
			(setq r (* (position_relative e)(+ (pondera_value e model_label fact) r)))
		)
		(when (> r (factor-max_change fact))
			(setq r (factor-max_change fact))
		)
		(when (> r (max_change_per_cent model_label fact))
			(setq r (max_change_per_cent model_label fact))
		)
		(if (eq (label-value model_label) (label-null model_label))
			model_label
			(desp_label model_label r fact)
		)
)



;Funció _[+].
;Paràmetres:
;		Llista de nombres o etiquetes.
;	fact -> Opcional. Estructura fact amb les dades de l'algorisme de rectificar.
;	algorism -> Opcional. Indica quin algorsime aplicar per rectificar el valor.
;Retorna: Una etiqueta amb valor igual o més alt del més alt de l, depenent de
;	l'algorisme de rectificació del valor més alt, i dels paràmetres de fact.
(defun _[+] (l &optional (algorism 'rectifica_independent_[+]))
		(if (eq l nil)
			(the_null)
			(funcall algorism l)
		)
)



;Funció _[*].
;Paràmetres:
;		l -> Llista de nombres o etiquetes.
;	fact -> Opcional. Estructura fact amb les dades de l'algorisme de rectificar.
;	algorism -> Opcional. Indica quin algorsime aplicar per rectificar el valor.
;Retorna: Una etiqueta amb valor igual o més baix del més baix de l, depenent de
;	l'algorisme de rectificació del valor més baix, i dels paràmetres de fact.
(defun _[*] (l &optional (algorism 'rectifica_independent_[*]))
		(if (eq l nil)
			(the_null)
			(funcall algorism l)
		)
)



;Mètode pharm_sensitivity.
;Paràmetres:
;		node_t -> Node de tipus tractament.
;		node_a -> Node de tipus agent infecciós.
```

```
;Retorna: L'eficàcia en que el node tractament actúa sobre el node agent infecciós.
;               És a dir, l'etiqueta de la relació que hi ha entre ells.
(defmethod pharm_sensitivity ((node_t T_node) (node_a A_node) &aux (resultat (the_null)))
        (dolist (e (node-edges_p node_t) resultat)
                (unless (eq (member e (node-p_edges node_a)) nil)
                        (setq resultat (edge-label e))
                )
        )
        resultat
)

;Mètode pharm_sensitivity.
;Paràmetres:
;       node -> Node de tipus tractament.
;       edge -> Aresta de sensitivitat farmacològica.
;Retorna: Si el paràmetre node és el node del qual prové el paràmetre edge, llavors
;         es retorna l'etiqueta de l'aresta,
;         en cas contrari es retorna una etiqueta null.
;Donat un T_node i una PS_edge retorna el valor de l'etiqueta que de l'aresta que hi ha entre ells, si hi és.
(defmethod pharm_sensitivity ((node T_node) (edge pharmacological_sensitivity_graph))
        (if (equal node (edge-from edge))
                (edge-label edge)
                (the_null)
        )
)

;Mètode pharm_sensitivity.
;Paràmetres:
;       node -> Qualsevol.
;       edge -> Qualsevol.
;Retorna: En cas de no entrar en els altres mètodes, entra en aquest i retorna
;         una etiqueta null.
(defmethod pharm_sensitivity (node edge)
        (the_null)
)



;Mètode indication_t.
;Paràmetres:
;       node -> Node de tipus tractament.
;       valu -> (Opcional) Una etiqueta valor, per defecte té un valor mínim.
;Retorna: Una llista de nodes agents infecciosos, els quals tenen una relació amb
;       el node que vé per paràmetre, i aquesta relació té una etiqueta de
;       igual o més valor que valu.
(defmethod indication_t ((node T_node) &optional (valu (the_min)) &aux resultat)
        (dolist (e (node-edges_p node) resultat)
                (when (_>= (pharm_sensitivity node e) valu)
                        (setq resultat (cons (edge-to e) resultat))
                )
        )
)



;Mètode indication_t.
;Paràmetres:
;       nodes -> Llista de nodes de tipus tractament.
;       valu -> (Opcional) Una etiqueta valor, per defecte té un valor mínim.
;Retorna: Una llista de nodes agents infecciosos, els quals tenen una relació amb
;       qualsevol dels node que venen per paràmetre, i aquesta relació té una etiqueta
;       de igual o més valor que valu.
(defmethod indication_t ((nodes cons) &optional (valu (the_min)) &aux resultat resultat2 )
        (dolist (e nodes resultat)
                (let
                        (
                                (indications (indication_t e valu))
                        )
                        (dolist (element indications resultat2)
                                (when (eq (member element resultat2) nil)
                                        (setq resultat2 ( cons element resultat2))
                                )
                        )
                )
        )
        resultat2
```

```
)


;Mètode treatment_a.
;Paràmetres:
;		node -> Node de tipus agent infecciós.
;		valu -> (Opcional) Una etiqueta valor, per defecte té un valor mínim.
;Retorna: Una llista de nodes tractament, els quals tenen una relació amb
;	el node que vé per paràmetre, i aquesta relació té una etiqueta de
;	igual o més valor que valu.
(defmethod treatment_a ((node A_node) &optional (valu (the_min)) &aux resultat)
		(dolist (e (node-p_edges node) resultat)
			(when (_>= (pharm_sensitivity (edge-from e) node ) valu)
				(setq resultat (cons (edge-from e) resultat))
			)
		)
)


;Mètode treatment_a.
;Paràmetres:
;		nodes -> Llista de nodes de tipus agent infecciós.
;		valu -> (Opcional) Una etiqueta valor, per defecte té un valor mínim.
;Retorna: Una llista de nodes tractament, els quals tenen una relació amb
;	qualsevol dels nodes que venen per paràmetre, i aquesta relació té una etiqueta
;	de igual o més valor que valu.
(defmethod treatment_a ((nodes cons) &optional (valu (the_min)) &aux resultat resultat2 )
		(dolist (e nodes resultat)
			(let
				(
					(treatments (treatment_a e valu))
				)
				(dolist (element treatments resultat2)
					(when (eq (member element resultat2) nil)
						(setq resultat2 ( cons element resultat2))
					)
				)
			)
		)
		resultat2
)


;Mètode is_cover.
;Paràmetres:
;		node_t -> Node de tipus tractament.
;		nodes_a -> Llista de nodes de tipus agents infecciosos.
;		valu -> Opcional. Etiqueta, per defecte és la que té valor mínim.
;Retorna: true si node_t combat amb una eficàcia igual o major al que marca l'etiqueta valu, a nodes_a.
(defmethod is_cover ((node_t T_node) (nodes_a cons) &optional (valu (the_min)) &aux (resultat t))
		(dolist (node_a nodes_a resultat)
			(when (_< (pharm_sensitivity node_t node_a) valu)
				(setq resultat nil)
			)
		)
		resultat
)


;Mètode is_cover.
;Paràmetres:
;		nodes_t -> Llista de nodes de tipus tractament.
;		nodes_a -> Llista de nodes de tipus agents infecciosos.
;		valu -> Opcional. Etiqueta, per defecte és la que té valor mínim.
;Retorna: true si entre tots el nodes_t poden combatre tots els nodes_a amb una eficàcia igual o major que
; la que marca l'etiqueta valu.
(defmethod is_cover ((nodes_t cons) (nodes_a cons) &optional (valu (the_min)) &aux resultat resultat2 new_nodes_a)
		(setq new_nodes_a nodes_a)
		(dolist (node_t nodes_t resultat)
			(dolist (node_a nodes_a resultat2)
				(when (_>= (pharm_sensitivity node_t node_a) valu)
					(setq new_nodes_a (substract node_a new_nodes_a))
				)
			)
```

```
		)
		(if (eq new_nodes_a nil)
			t
			nil
		)
)


;Funció global_adequacy.
;Paràmetres:
;	t_nodes_list -> Llista de nodes de tipus tractament.
;	a_nodes_list -> Llista de nodes de tipus agents infecciosos.
;	p_nodes_list -> Llista de nodes de tipus paràmetres del pacient.
;Retorna: Una etiqueta, el valor de la qual compleix la funció global_adequacy,
;	explicada en l'apartat de model abstracte.
(defun global_adequacy (t_nodes_list a_nodes_list p_nodes_list)
		(_[*] (list (pharm_adequacy t_nodes_list a_nodes_list) ( contx_adequacy t_nodes_list p_nodes_list)))
)


;Mètode pharm_adequacy.
;Paràmetres:
;	node_t -> Node de tipus tractament.
;	node_a -> Node de tipus agent infecciós.
;Retorna: Una etiqueta que és la relació que hi ha entre els dos nodes, si no hi ha
;	cap relació es retorna l'etiqueta amb valor null.
(defmethod pharm_adequacy ((node_t T_node) (node_a A_node))
		(pharm_sensitivity node_t node_a)
)



;Mètode pharm_adequacy.
;Paràmetres:
;	t_nodes_list -> Llista de nodes de tipus tractament.
;	node_a -> Node de tipus agent infecciós.
;Retorna: Una etiqueta que és el resultat d'aplicar la funció _[+] de totes les etiquetes
;	que hi ha entre tots els node la llista t_nodes_list i el node_a.
(defmethod pharm_adequacy ((t_nodes_list cons) (node_a A_node) &aux resultat)
		(dolist (e t_nodes_list resultat)
			(unless (eq (pharm_sensitivity e node_a) nil)
				(setq resultat (cons (pharm_sensitivity e node_a) resultat))
			)
		)
		(_[+] resultat)
)



;Mètode pharm_adequacy.
;Paràmetres:
;	t_nodes_list -> Llista de nodes de tipus tractament.
;	a_nodes_list -> Node de tipus agent infecciós.
;Retorna: Una etiqueta que és el resultat d'aplicar la funció _[+] de totes les etiquetes
;	que hi ha entre tots els nodes de les dues llistes que es passen per paràmetre.
(defmethod pharm_adequacy ((t_nodes_list cons) (a_nodes_list cons) &aux resultat)
		(dolist (a a_nodes_list resultat)
			(setq resultat (cons (pharm_adequacy t_nodes_list a) resultat))
		)
	(_[*] resultat)
)



;Funció contx_adequacy.
;Paràmetres:
;	t_nodes_list -> Llista de nodes de tipus tractament.
;	p_nodes_list -> Llista de nodes de tipus parametre.
;Retorna: Una etiqueta que és el negat de la funció risk.
(defun contx_adequacy (t_nodes_list p_nodes_list)
		(_neg (risk t_nodes_list p_nodes_list))
)



;Funció risk.
```

```
;Paràmetres:
;       t_nodes_list -> Llista de nodes de tipus tractament.
;       p_nodes_list -> Llista de nodes de tipus parametre del pacient.
;Retorna: Etiqueta, consulta la funció risk en l'apartat de model abstracte.
(defun risk (t_nodes_list p_nodes_list &aux resultat resultat2)
        (dolist (tt t_nodes_list resultat)
                (dolist (aa (node-edges_p tt) resultat2)
                        (when (eq (type-of aa) 'side_effects_graph)
                                (setq resultat2 (cons (_[*] (list (side_effect tt (edge-to aa)) (effect_to_avoid
(edge-to aa) p_nodes_list))) resultat2))
                        )
                )
                (setq resultat (cons (_[+] resultat2) resultat))
        )
        (_[+] resultat)
)



;Mètode effect_to_avoid.
;Paràmetres:
;       node_e -> Node de tipus efecte secundari.
;       node_p -> Node de tipus paràmetre del pacient.
;Retorna: Etiqueta de la relació que hi ha entre els dos nodes, si no hi hagués relació
;       es retorna una etiqueta amb valor null.
(defmethod effect_to_avoid ((node_e E_node) (node_p P_node) &aux (resultat (the_null)))
        (dolist (e (node-edges_p node_p) resultat)
                (unless (eq (member e (node-p_edges node_e)) nil)
                        (setq resultat (edge-label e))
                )
        )
        resultat
)



;Mètode effect_to_avoid.
;Paràmetres:
;       node_e -> Node de tipus efecte secundari.
;       p_nodes_list -> Llista de nodes de tipus paràmetre del pacient.
;Retorna: Etiqueta resultant d'aplicar la funció _[+] amb les etiquetes de totes les
;       relacions entre tots els nodes de p_nodes_list i node_e.
(defmethod effect_to_avoid ((node_e E_node) (p_nodes_list cons) &aux resultat)
        (dolist (p p_nodes_list resultat)
                (setq resultat (cons (effect_to_avoid node_e p) resultat))
        )
        (_[+] resultat)
)



;Funció side_effect:
;Paràmetres:
;       node_t -> Node de tipus tractament.
;       node_e -> Node de tipus efecte secundari.
;Retorna: Etiqueta de la relació que hi ha entre els dos nodes, si no hi hagués relació
;       es retorna una etiqueta amb valor null.
(defmethod side_effect ((node_t T_node) (node_e E_node) &aux (resultat (the_null)))
        (dolist (e (node-edges_p node_t) resultat)
                (unless (eq (member e (node-p_edges node_e)) nil)
                        (setq resultat (edge-label e))
                )
        )
        resultat
)



;Funció load_data.
;Paràmetres:
;       file -> Instància de fitxer a obrir.
;Fa: Omple totes les dades.
;Retorna -> T.
(defun load_data (file &aux instancia_value auxiliar values_list eta)
 (setq list_of_nodes nil)
```

```
(setq list_of_edges nil)
(with-open-file (s file)
  ; Llegeixo la linea, creo una instància que després utilitzaré
  ; per crear els objectes value
(with-input-from-string (words (read-line s))
        (let (
                (z (read words))
            )
                (cond
                        ((eq z 'quantitative)
                                (setq instancia_value (make-instance 'quantitative_label :value 0 :minim (read words)
:maxim (read words) :null (read words)))
                        )
                        ((eq z 'qualitative)
                                (loop
                                        (setq auxiliar (read words))
                                        (if (eq auxiliar 'final)
                                                (progn
                                                        (setq values_list (reverse values_list))
                                                        (setq instancia_value (make-instance 'qualitative_label
:value nil :minim (cadr values_list) :maxim (car (reverse values_list)) :null (car values_list) :elements values_list))
                                                        (return t)
                                                )
                                                (setq values_list (cons auxiliar values_list))
                                        )
                                )
                        )
                        (t (print "Error"))
                )
        )
)
(with-input-from-string (words (read-line s))
        (setq max_treatments (read words))
        (setq fact_[+] (make-instance 'factor_[+] :changing_factor (read words) :max_change (read words)
:max_change_per_cent (read words)))
        (setq fact_[*] (make-instance 'factor_[*] :changing_factor (read words) :max_change (read words)
:max_change_per_cent (read words)))
)

(do   ((linea (read-line s) (read-line s nil 'eof)))
        ((eq linea 'eof) "File readed correctly.")
        (with-input-from-string (r linea)
                (let (
                        (x (read r))
                        )
                        (cond
                                ((eq x 'PS) (add_PS_edge (make__label (read r) instancia_value) (read r) (read r)))
                                ((eq x 'SE) (add_SE_edge (make__label (read r) instancia_value) (read r) (read r)))
                                ((eq x 'ETA)
                                        (setq eta (add_ETA_edge (make__label (read r) instancia_value) (read r)
(read r)))
                                        (loop
                                                (setq auxiliar (read r))
                                                (when (eq auxiliar 'end)
                                                        (return t)
                                                )
                                                (setf
                                                        (label-considerations (edge-label eta))
                                                        (cons
                                                                (make-instance 'consideration :value_ini
auxiliar
                                                                        :funcio_inferior (read r)
                                                                        :sota_llindar_inferior (read r)
                                                                        :llindar_inferior (read r)
                                                                        :llindar_superior (read r)
                                                                        :sobre_llindar_superior (read r)
                                                                        :funcio_superior (read r)
                                                                )
                                                                (label-considerations (edge-label eta))
                                                        )
                                                )
                                        )
                                )
                                (t "Error")
                        )
                )
```

```
		)
)
)
)



;Funció find_nodes_e.
;Paràmetres:
;	l -> (Opcional) Llista de nodes.
;Retorna : Una llista amb tots els nodes de tipus efecte secundari de la llista l.
(defun find_nodes_e (&optional (l list_of_nodes))
		(cond ((eq l nil) nil)
			((eq (type-of (car l)) 'E_node) (cons (car l) (find_nodes_e (cdr l))))
			(t (find_nodes_e (cdr l)))
		)
)


;Funció find_nodes_p.
;Paràmetres:
;	l -> (Opcional) Llista de nodes.
;Retorna : Una llista amb tots els nodes de tipus paràmetre del pacient de la llista l.
(defun find_nodes_p (&optional (l list_of_nodes))
		(cond ((eq l nil) nil)
			((eq (type-of (car l)) 'P_node) (cons (car l) (find_nodes_p (cdr l))))
			(t (find_nodes_p (cdr l)))
		)
)


;Funció find_nodes_t.
;Paràmetres:
;	l -> (Opcional) Llista de nodes.
;Retorna : Una llista amb tots els nodes de tipus tractament de la llista l.
(defun find_nodes_t (&optional (l list_of_nodes))
		(cond ((eq l nil) nil)
			((eq (type-of (car l)) 'T_node) (cons (car l) (find_nodes_t (cdr l))))
			(t (find_nodes_t (cdr l)))
		)
)



;Funció find_nodes_e.
;Paràmetres:
;	l -> (Opcional) Llista de nodes.
;Retorna : Una llista amb tots els nodes de tipus agent infecciòs de la llista l.
(defun find_nodes_a (&optional (l list_of_nodes))
		(cond ((eq l nil) nil)
			((eq (type-of (car l)) 'A_node) (cons (car l) (find_nodes_a (cdr l))))
			(t (find_nodes_a (cdr l)))
		)
)



(defmethod printa_noms_nodes (argument_buit)
		nil
)


(defmethod printa_noms_nodes ((node node))
		(princ (node-name node))
)


(defmethod printa_noms_nodes ((l cons) &aux (resultat t))
		(dolist (e l resultat)
		(princ (node-name e))
	(princ " ")
		)
)

(defun printa_info_arestes (&optional (l list_of_edges) &aux (resultat t))
```

```
		(dolist (e l resultat)
			(format t "from: ~a to: ~a value: ~a ~%" (node-name (edge-from e)) (node-name (edge-to e)) (label-value
(edge-label e)))
		)
)

(defun printa_info_etiquetes (l &aux (resultat t))
		(print "")
		(dolist (e l resultat)
			(format t " valor: ~a " (label-value e))
		)
)



;Funció find_instances.
;Paràmetres:
;	agents_infecciosos -> Llista de noms de nodes.
;	llista -> Llista de nodes, per defecte la llista general.
;Retorna: Una llista de les instàncies dels nodes indicats pel paràmetre agents_infecciosos
;	 i que estan dins el paràmetre llista
(defun find_instances (agents_infecciosos &optional (list_ list_of_nodes))
		(cond ((eq agents_infecciosos nil) nil)
			(t (cons (find_node (car agents_infecciosos) list_) (find_instances (cdr agents_infecciosos) list_)))
		)
)


;Funció substract.
;Paràmetres:
;	element -> Qualsevol item.
;	list_ -> Llista de items.
;Retorna: La mateixa llista que venia per paràmetre havent extret prèviament els elements
;	que són igual al paràmetre element.
(defun substract (element list_)
		(cond ((eq list_ nil) nil)
			(t
				(if (eq element (car list_))
					(substract element (cdr list_))
					(cons (car list_) (substract element (cdr list_)))
				)
			)
		)
)


;Defun delete_node.
;Paràmetres:
;	 nod -> Node.
;Funcionament: Elimina un node de la llista general de nodes, despres elimina totes
;	les arestes que hi van. De tots els nodes es treuen les arestes que s'han
;	eliminat, i si algun node queda sense cap aresta entrant o sortint,
;	es torna a cridar recursivament la mateixa funció per eliminar aquest node.
;Retorna: t.
(defun delete_node(nod &aux resultat resultat2)
		(print (node-name nod))
		(setq list_of_nodes (substract nod list_of_nodes))
		(dolist (arest list_of_edges resultat)
			(when (or (eq (edge-to arest) nod) (eq (edge-from arest) nod))
				(setq list_of_edges (substract arest list_of_edges))
				(dolist (nod_aux list_of_nodes resultat2)
					(when  (not (eq (member arest (node-p_edges nod_aux)) nil))
						(setf  (node-p_edges nod_aux) (substract arest (node-p_edges nod_aux)))
					)
					(when  (not (eq (member arest (node-edges_p nod_aux)) nil))
						(setf  (node-edges_p nod_aux) (substract arest (node-edges_p nod_aux)))
					)
					(when (and (eq (node-p_edges nod_aux) nil) (eq (node-edges_p nod_aux) nil))
						(delete_node nod_aux)
					)
				)
			)
		)
)
```

```
;Funció optimize_microorganisms.
;Paràmetres:
;          a_nodes_list -> Llista de nodes de tipus agent infecciós.
;Funcionament: Simplifica el graf muntat d'arestes i nodes, on tota la informació està en
;             les variables globals de nodes i arestes. S'eliminen tots els nodes de tipus
;             agent infecciòs que no estiguin també a a_nodes_list.
;Retorna: t.
(defun optimize_microorganisms (a_nodes_list &aux resultat)
            (dolist (e (find_nodes_a) resultat)
                        (when (eq (member e a_nodes_list) nil)
                                    (delete_node e)
                        )
            )
)


;Funció quit_parameters.
;Paràmetres:
;          p_nodes_list -> Llista de nodes de tipus paràmetres.
;Funcionament: Simplifica el graf muntat d'arestes i nodes, on tota la informació està en
;             les variables globals de nodes i arestes. S'eliminen tots els nodes
;             que no estiguin també a p_nodes_list.
;Retorna: t.
(defun quit_parameters (p_nodes_list &aux resultat)
            (dolist (e (find_nodes_p) resultat)
                        (when (eq (member e p_nodes_list) nil)
                                    (delete_node e)
                        )
            )
)

;Funció optimize_parameters.
;Paràmetres:
;             parameters -> Llista de paràmetres de la forma per exemple'((pregnant yes) (edat 25) (renal_malfunction med)...)
;Fa: Actualitza les etiquetes de les arestes que surten dels paràmetres sogons l'argument associat que tenia
;             el paràmetre a la llista, i treu del graf els nodes paràmetre que no estan en aquesta llista. Més ben explicat
;             a la memòria.
(defun optimize_parameters (parameters &aux resultat resultat2 list_parameters)
            (dolist (e parameters resultat)  ;per tots els parametres
                        (setq resultat (node-edges_p (find_node (car e))))
                        (dolist (e2 resultat resultat2)   ;per totes les arestes que surten del parametre
                                    (when (eq (refresh_label (edge-label e2) (cadr e)) t)
                                                (setq list_parameters (cons (car e) list_parameters))
                                    )
                        )
            )
            (quit_parameters (find_instances list_parameters))
)

(defun is_there_any_ps (edges_list)
            (if (eq edges_list nil)
                        nil
                        (if (typep (car edges_list) 'pharmacological_sensitivity_graph)
                                    t
                                    (is_there_any_ps (cdr edges_list))
                        )
            )
)

(defun optimize_treatments (&aux resultat)
            (dolist ( e (find_nodes_t) resultat)
                        ;(print "hola")
                        ;(print (node-name e))
                        (when (not (is_there_any_ps (node-edges_p e)))
                                                (delete_node e)
                        )
            )
)

(defun optimize_graph (microorganisms parameters)
(optimize_microorganisms microorganisms)
(optimize_parameters parameters)
(optimize_treatments)
)
```

```
(defmethod refresh_label ((parametre qualitative_label) ( new_value number) &aux aux1 aux2 resultat)
	(dolist (e (label-considerations parametre) resultat)
		(when (and (>= new_value (consideration-sota_llindar_inferior e)) (<= new_value (consideration-
sobre_llindar_superior e)))
			(when (and (>= new_value (consideration-sota_llindar_inferior e)) (< new_value (consideration-
llindar_inferior e)))
				;cas sota_llindar_inferior ;llindar_inferior
				(setq aux1 (/ (- new_value (consideration-sota_llindar_inferior e)) (- (consideration-
llindar_inferior e) (consideration-sota_llindar_inferior e))))
				(setq aux2 (expt aux1 (consideration-funcio_inferior e)))
				(setq aux2 (+ aux1 (- aux1 aux2)))
				(setq resultat (round (+ (* aux2 (- (position (consideration-value_ini e)
(qualitative_label-elements parametre)) (position (label-value parametre) (qualitative_label-elements parametre)))) (position (label-
value parametre) (qualitative_label-elements parametre)))))
				(setf (label-value parametre) (car (nthcdr resultat (qualitative_label-elements
parametre))))
			)
			(when (and (>= new_value (consideration-llindar_inferior e)) (<= new_value (consideration-
llindar_superior e)))
				;cas llindar_inferior ;llindar_superior
				(setf (label-value parametre) (consideration-value_ini e))
			)
			(when (and (> new_value (consideration-llindar_superior e)) (< new_value (consideration-
sobre_llindar_superior e)))
				;cas llindar_superior ;sobre_llindar_superior
				(setq aux1 (/ (- (consideration-sobre_llindar_superior e) new_value) (- (consideration-
sobre_llindar_superior e) (consideration-llindar_superior e))))
				(setq aux2 (expt aux1 (consideration-funcio_superior e)))
				(setq aux2 (+ aux1 (- aux1 aux2)))
				(setq resultat (round (+ (* aux2 (- (position (consideration-value_ini e)
(qualitative_label-elements parametre)) (position (label-value parametre) (qualitative_label-elements parametre)))) (position (label-
value parametre) (qualitative_label-elements parametre)))))
				(setf (label-value parametre) (car (nthcdr resultat (qualitative_label-elements
parametre))))
			)
		)
	)
	(if (eq (label-value parametre) (label-value (the_null)))
		nil
		t
	)
)

(defmethod refresh_label ((parametre qualitative_label) new_value &aux aux1 aux2)
	(if (eq new_value 'yes)
		t
		(if (eq (member new_value (qualitative_label-elements parametre)) nil)
			nil
			(progn
				(setf aux1 (- (length (qualitative_label-elements parametre)) (position new_value
(qualitative_label-elements parametre))))
				(setf aux2 (- 1 (/ aux1 (length (qualitative_label-elements parametre)))))
				(setf aux2 (round (* (position (label-value parametre) (qualitative_label-elements
parametre)) aux2)))
				(setf (label-value parametre) (car (nthcdr aux2 (qualitative_label-elements parametre))))
				t
			)
		)
	)
)

(defmethod refresh_label ((parametre quantitative_label) (new_value number) &aux aux1 aux2 resultat)
	(if (eq (label-considerations parametre) nil)
		(if (and (>= new_value (label-minim parametre)) (<= new_value (label-maxim parametre)))
			(progn
				(setq aux2 (- 1 (/  (- (label-maxim parametre) new_value) (- (label-maxim parametre)
(label-minim parametre)))))
				(setq aux1 (* (label-value parametre) aux2))
				(setf (label-value parametre) aux1)
				t
			)
			nil
		)
		(progn
			(dolist (e (label-considerations parametre) resultat)
```

```
                                        (when (and (>= new_value (consideration-sota_llindar_inferior e)) (<= new_value
(consideration-sobre_llindar_superior e)))
                                                (when (and (>= new_value (consideration-sota_llindar_inferior e)) (<
new_value (consideration-llindar_inferior e)))
                                                        ;cas sota_llindar_inferior ;llindar_inferior
                                                        (setq aux1 (/ (- new_value (consideration-sota_llindar_inferior e))
(- (consideration-llindar_inferior e) (consideration-sota_llindar_inferior e))))
                                                        (setq aux2 (expt aux1 (consideration-funcio_inferior e)))
                                                        (setq aux2 (+ aux1 (- aux1 aux2)))
                                                        (setq resultat (+ (* aux2 (- (consideration-value_ini e) (label-value
parametre))) (label-value parametre)))
                                                        (setf (label-value parametre) resultat)
                                                )
                                                (when (and (>= new_value (consideration-llindar_inferior e)) (<= new_value
(consideration-llindar_superior e)))
                                                        ;cas llindar_inferior ;llindar_superior
                                                        (setf (label-value parametre) (consideration-value_ini e))
                                                )
                                                (when (and (> new_value (consideration-llindar_superior e)) (< new_value
(consideration-sobre_llindar_superior e)))
                                                        ;cas llindar_superior ;sobre_llindar_superior
                                                        (setq aux1 (/ (- (consideration-sobre_llindar_superior e)
new_value) (- (consideration-sobre_llindar_superior e) (consideration-llindar_superior e))))
                                                        (setq aux2 (expt aux1 (consideration-funcio_superior e)))
                                                        (setq aux2 (+ aux1 (- aux1 aux2)))
                                                        (setq resultat (+ (* aux2 (- (consideration-value_ini e) (label-value
parametre))) (label-value parametre)))
                                                        (setf (label-value parametre) resultat)
                                                )
                                        )
                                )
                                (if (and (>= (label-value parametre) (label-minim parametre)) (<= (label-value parametre) (label-
maxim parametre)))
                                        t
                                        nil
                                )
                        )
                )
)

(defmethod refresh_label ((parametre quantitative_label) new_value)
        (if (eq new_value 'yes)
                t
                nil
        )
)



;Funció afegint.
;Paràmetres:
;       element -> Qualsevol item.
;       list_ -> Llista de la forma  '((a...) (b...) ...) .
;Retorna: llista de la forma ((element a ...) (element b ...) ...).
(defun afegint (element list_)
        (cond ((eq list_ nil) nil)
                (t (cons (cons element (car list_)) (afegint element (cdr list_))))
        )
)



;Funcio gn.
;       list_ -> Llista de items.
;       numero -> Un número.
;Retorna: Una llista de totes les possibles combinacions de tants elements com marca
;         el paràmetre número, de tots els elements de la llista.
(defun gn (list_ numero)
        (cond ((eq (length list_) numero) (list list_))
                ((eq numero 1) (cons (list (car list_)) (gn (cdr list_) numero)))
                (t (append (afegint (car list_) (gn (cdr list_) (- numero 1))) (gn (cdr list_) numero)))
        )
)
```

```
;Funcio generate_possibilites.
;       list_ -> Llista de items.
;       numero -> Un número.
;Retorna: Una llista de totes les possibles combinacions d'un element fins a tants elements
;         com marca el paràmetre número, de tots els elements de la llista.
(defun generate_possibilities (list_ numero)
        (cond ((eq numero 1) (gn list_ numero))
                (t (append (gn list_ numero) (generate_possibilities list_ (- numero 1))))
        )
)



;Defun exchange.
;Paràmetes:
;       i -> Número.
;       j -> Número.
;       list_ -> Llista de items.
;Retorna: La mateixa llista però amb els elements de les posicions i j intercanviats.
(defun exchange (i j list_ &aux aux)
        (when (> i j)
                (setq aux i)
                (setq i j)
                (setq j aux)
        )
        (append
                (nthcdr (- (length list_) (- i 1)) (reverse list_)) ;fins la i-1
                (list (car (nthcdr (- j 1) list_))) ; la j
                (reverse (nthcdr (- (length list_) (- j 1)) (reverse (nthcdr i list_))));entre la i la j
                (list (car (nthcdr (- i 1) list_))) ; la i
                (nthcdr j list_)
        )
)



;Funció order.
;Paràmetres:
;       list_ -> Llista de números o etiquetes.
;Retorna: La mateixa llista ordena de major a menor.
(defun order (list_ &aux ii e1 e2 r1 r2)
        (dotimes (i (- (length list_) 1) r1)
                (setq ii 0)
                (dotimes (j (- (length list_) (+ i 1)) r2)
                        (setq e1 (car (reverse (car (nthcdr ii list_)))))
                        (setq e2 (car (reverse (car (nthcdr (+ ii 1) list_)))))
                        (setq ii (+ 1 ii))
                        (when (or (_< e1 e2) (and (_<= e1 e2) (_>= e1 e2) (> (length (car (car (nthcdr (- ii 1) list_))))
(length (car (car (nthcdr ii list_)))))))
                                (setq list_ (exchange ii (+ ii 1) list_))
                        )
                )
        )
        list_
)


(defun printa_sols (list_ &aux resultat)
        (dolist (element list_ resultat)
                (when (not (eq (label-value (cadr element)) (label-null (cadr element))))
                        (print "TRACTAMENT: ")
                        (printa_noms_nodes (car element))
                        (princ "  EFICACIA: ")
                        (princ (label-value (cadr element)))
                )
        )
)


(defun load_file (file)
        (load_data (merge-pathnames file))
)


;Funcio find_treatments
```

```
;Paràmetres:
;		microorganisms -> Llista de noms d'agents infecciosos.
;		parameters -> Llista de paràmetres del tipus '((age 34) (height 78) (renal_failure med) (transplanted yes)...).
;		file -> String que indica el fitxer.
;Retorna: Una solució per curar el pacient que té aquests agents_infecciosos.
(defun find_treatments (microorganisms parameters file &aux resultat tr_proves solucions)
		(load_data (merge-pathnames file))
		(setq microorganisms (find_instances microorganisms))
		(optimize_graph microorganisms parameters)
		(setq tr_proves (generate_possibilities (find_nodes_t) max_treatments))
		(setq solucions nil)
		(dolist (tr_prova tr_proves resultat)
			(setq solucions (cons (list tr_prova (global_adequacy tr_prova microorganisms (find_nodes_p))) solucions))
		)
		(printa_sols (order solucions))
)
```

8.2 - COM UTILITZAR EL PROGRAMA I RECOMANACIONS

Es pretén en aquest benentès ajudar a qui utilitzi el programa per primeres vegades, a donar un seguit de recomanacions, així com explicar com executar les diferents funcions que estan preparades per ajudar a donar un bon tractament per un pacient.

8.2.1 - Recomanacions prèvies

Primer de tot s'ha de definir el fitxer d'entrada de dades. En aquest fitxer és on es defineix el model i les seves relacions amb els seus graus. Aquesta és una feina farragosa, i s'hauria de fer a consell d'un metge, ja que interpretar i traspassar el coneixement que poden donar les guies d'ajuda [2] no és fàcil, i el consell d'un metge serà important. De la bona definició d'aquest fitxer dependrà l'èxit del programa. S'explica en el capítol posterior com definir aquest fitxer, i remarquem aquí la importància d'entendre bé la nomenclatura d'aquest fitxer com tot ho explicat fins ara per una bona definició. A més en alguns casos, no serà fàcil trobar la correspondència entre les dades que ens facilitin els manuals de medicina i aquest model, és més, a vegades serà impossible, per això tornem a remarcar que aquest projecte està preparat per només casos senzills, però també està preparat per evolucionar i poder un dia tractar problemes reals i complexos.

Hi ha diversos dubtes que poden sorgir alhora de definir el model. Per començar podríem pensar quin conjunt de valors es pot utilitzar per definir les etiquetes de les relacions, els seus graus. Un bon consell és utilitzar un domini el més gran possible sense que s'escapi gaire de la precisió que en realitat pot donar. Així serà més precís el model a donar unes bones solucions. En els exemples que en aquest treball s'han proposat el rang d'aquests valors és molt pobre. Si s'utilitzen valors qualitatius també es perd encara més en precisió, ja que es fan moltes operacions intermèdies que necessiten de l'arrodoniment per donar els valors qualitatius i es va perdent informació. Poder millor donar valors quantitatius (no es perd precisió en càlculs intermedis), però de tota

manera, si el conjunt de valors qualitatius és molt gran, el model no hauria de perdre quasi precisió.

Per altre banda estan els valors que omplen les dues instàncies de les classes *factor[+]* i *factor[*]* que ajuden a donar una definició més acurada pels operadors $\oplus$ i $\otimes$. S'ha d'entendre correctament l'apartat que explica el funcionament d'aquestes funcions en el capítol de *Modelització*. Si el paràmetre *changing_factor* d'aquestes classes és 0, aquestes funcions són com les funcions *min* i *max* respectivament. Si s'omplen aquestes instàncies de les classes *factor[+]* i *factor[*]* , es recomenable fer-ho amb valors molt petits. S'ha de remarcar en aquest apartat que la funció que utilitza per defecte el programa, és la funció *rectifica_independent*. Per canviar aquesta funció, i posar l'altre proposada per aquest treball (*rectifica_dependent*), o qualsevol altre, s'haurà de tocar el codi, és a dir, ho haurà de fer un programador, però igualment està fet d'una manera genèrica i que ajuda a fer aquest canvi. Només s'ha de canviar el nom de la funció que s'ha d'executar en el lloc correcte. Llavors, amb tot això, un valor recomenable per *changing_factor*, seria un valor no més gran que una trentena part del rang del conjunt V. És a dir, si el conjunt V estigués comprés entre el 1 i el 11, el màxim valor recomanble per *changing_factor* seria de 10/30 = 0.3. I també seria convenient posar un màxim valor de *max_change_per_cent* de 0.1.

És important omplir d'una manera adequada els grafs. A la hora de posar els graus en les relacions s'ha de ser més aviat pessimista, o sigui, posar el pitjor que pot passar, i és millor entendre com calcula el programa la solució per poder crear uns bons fitxers d'entrada. Una part complicada pot ser omplir les relacions del graf d'efectes secundaris a evitar, i concretament les relacions on l'origen són paràmetres numèrics com l'edat o el pes, on es defineix una funció en la relació, que donat el paràmetre del pacient, doni un grau per a la relació. S'ha d'entendre correctament l'apartat que explica com omplir aquestes relacions en el capítol *Representació del Model*. Recordar que s'està definint una funció a trossos, on cada tros és una pujada i baixada de la funció. Pensar que el valor d'aquesta funció sempre ha de tenir un valor mínim absolut.

8.2.2 - Utilització del programa

Si veiem la API, veurem quines funcions es poden utilitzar. Evidentment, un anàlisis del codi, podrà portar a utilitzar moltes altres funcions i ampliar les utilitats i en definitiva continuar el projecte, però l'objectiu de l'API és descriure les funcions que es poden utilitzar per trobar tractaments per un pacient, així com investigar una mica sobre el model descrit, sense haver-se mirat el codi i només havent llegit la memòria.

En aquest apartat s'entén que el fitxer d'entrada ja està definit, i només queda utilitzar-lo per diferents casos. Hi ha una funció que ens dona la solució *Find_treatments*, en l'API ens descriu com fer-la servir i també en el capítol d'exemples d'aquesta memòria. Aquesta funció ho fa tot, carrega el model des del fitxer, optimitza el graf amb les dades que només interessen pel determinat cas, fa els càlculs pertinents, i dona tot el conjunt de tractaments possibles ordenats per la seva eficàcia.

Per altra banda hi ha funcions que poden ser molt útils, com *treatment_a*, *indication_t* o *is_cover*, que són qüestions definides en el model, i que poden de ser de gran ajut per entendre el model que s'ha definit en el fitxer. Per utilitzar aquestes funcions s'hauran d'utilitzar però les funcions prèvies descrites en l'API, per convertir per exemple el nom d'un antibiòtic a un node antibiòtic o per convertir el valor d'una etiqueta en una etiqueta amb aquest valor.

S'ha de pensar, que si s'utilitzen funcions que donen pistes sobre el model, es pot fer de dues maneres. Una manera és carregar el model amb la funció *load_model* i executant les funcions desitjades, i podrem consultar el que volguem sobre el model carregat. Una altre manera, és executar la funció *find_treatments* que carrega el model, i el simplifica pel cas que s'ha entrat, així com també s'actualitzen els graus de les relacions del graf d'efectes secundaris a evitar pel cas concret, i després aplicar les funcions desitjades per saber com estar el model havent aplicat aquest cas.

8.3 - CONSTRUCCIÓ FITXER D'ENTRADA DE DADES

En aquest capítol s'explica com escriure el fitxer d'entrada de dades. En aquest fitxer hi ha de figurar l'explicació de tot el model, és a dir, totes les relacions que hi ha entre antibiòtics, efectes secundaris, paràmetres i microorganismes, omplint així els tres grafs. També han d'estar representades la forma de les etiquetes de les relacions, i els graus de les relacions. També s'exposa quina nomenclatura es segueix per omplir les etiquetes, així com els paràmetres que ompliran les dues instàncies a les classes *factor[+]* i *factor[*]* en que es decideix el funcionament dels operadors $\oplus$ i $\otimes$.

La sintaxis que utilitzem per definir les següents línies és la següent.

- Paraules en majúscula són paraules clau. S'han de posar tal qual.
- Paraules en minúscula indiquen el tipus del ítem que va en el determinat lloc.
- $\{\}^+$: repeticions de una vegada o més.
- $\{\}^*$: repeticions de 0 vegades o més.
- A | B : o A, o B.

Passem a explicar la sintaxis del fitxer. Ho farem per parts. Comencem per la primera línia del fitxer. En aquesta es defineix la forma que hauran de tenir les etiquetes de les relacions, és a dir, els graus. Si seran etiquetes quantitatives o qualitatives i els seus marges. S'ha de seguir la següent sintaxis:

- **Sintaxis 1ª Línia:** (QUALITATIVE string { string }$^+$ FINAL) | (QUANTITATIVE number number number)

Tenim dues opcions, que les etiquetes siguin quantitatives (començarem posant la paraula clau *quantitative*), o que les etiquetes tinguin valors discrets representats per noms (començarem per la paraula clau *qualitative*). En cas de voler representar les

etiquetes amb noms, posarem doncs primer la paraula *qualitative* i en seguit una colla de noms. Aquests noms s'entendran ordenadament de valor més baix a valor més gran, sent el primer el valor *null*, el segon el valor mínim i l'últim el valor màxim. En acabat s'ha de posar la paraula clau *final* per indicar que ja s'ha arribat al valor màxim. És a dir, com en l'exemple, si volem representar les etiquetes amb valors {*null low med high*}, sent *null* el valor null, *low* el valor més baix, i *high* el valor més alt, la primera línia del fitxer hauria de ser la següent.

- **Exemple línia 1:** qualitative null low med high final

En cas de voler posar els graus de les relacions, posarem primer la paraula *quantitative*, i llavors tres números, que seran exactament el valor null, el valor mínim (o el llindar en que es considera que una relació tingui sentit) i el valor màxim en aquest ordre. Per un bon funcionament de l'algorisme el valor null ha de ser més petit que el mínim numèricament. Un exemple d'entrada de fitxer en aquest format on els graus de les relacions estiguessin entre 0 i 10 amb el valor mínim o llindar a 1, seria el següent.

- **Exemple línia 1:** quantitative 0 1 10

La segona línia indica dues parts importants que l'algorisme ha de tenir en compte. Primer el número màxim d'antibiòtics que es pot utilitzar per cada tractament, segon els paràmetres de les dues instàncies de les classes *factor[+]* i *factor[*]* que controlen el funcionament dels operadors $\oplus$ i $\otimes$. En tot cas la segona línia té la forma següent. S'utilitzen els parèntesis amb un número a dintre per separar cada paràmetre per després saber-los identificar quan s'expliqui el significat:

- **Sintaxis 2ª Línia:** number(1) number(2) number(3) number(4) number(5) number(6) number(7)

Hem numerat els números per explicar seguidament les condicions que han de complir i que signifiquen.

1. Number1: Ha de ser un número enter entre 1 i el número d'antibiòtics total que es vol que s'arribin a utilitzar per cada tractament. Indica doncs el número d'antibiòtics pot contenir un tractament.
2. Number2: Número real més gran o igual a 0. Serà el corresponent *changing_factor* per a la instància de la classe *factor[+]*.
3. Number3: Número real més gran o igual a 0. Serà el corresponent *max_change* per a la instància de la classe *factor[+]*.
4. Number4: Número real entre 0 i 1. Serà el corresponent *changing_factor_per_cent* per a la instància de la classe *factor[+]*.
5. Number5: Número real més gran o igual a 0. Serà el corresponent *changing_factor* per a la instància de la classe *factor[*]*.
6. Number6: Número real més gran o igual a 0. Serà el corresponent *max_change* per a la instància de la classe *factor[*]*.
7. Number7: Número real entre 0 i 1. Serà el corresponent *changing_factor_per_cent* per a la instància de la classe *factor[*]*.

Així doncs, si volguéssim dos antibiòtics com a màxim per tractament, i que els operadors $\oplus$ i $\otimes$ equivalguessin a les funcions *min* i *max*, un possible segona línia seria la següent:

- **Exemple línia 2:** 2 0 0 0 0 0 0

Si es volgués posar uns sensible canvi als operadors $\oplus$ i $\otimes$ respecte les funcions *min* i *max*, un possible segona línia seria la següent:

- **Exemple línia 2:** 2 0.1 1 0.2 0.1 1 0.2

A partir de la tercera es comença a definir tot el model, on es relacionen els diferents tipus d'objectes i es formen els tres grafs. Serà un seguit de línies, on a cada línia es definirà una relació amb la seva etiqueta corresponent. Ens podem trobar tres tipus de possibles línies, segons els tres tipus de relació que estiguem definint degut als tres tipus de graf. Definirem la sintaxis de cada tipus de línia per separat, però alhora d'escriure

el fitxer no importa cap ordre i fins i tot es poden barrejar. Comencem per definir les relacions del graf de sensibilitat farmacològica, que ha de ser de la següent forma:

- **Sintaxis d'una línia a partir de la 3ª. Possibilitat 1:** PS (string | number) string(2) string(3)

Una relació del graf de sensibilitat farmacològica comença per la paraula clau *ps.* Els següents paràmetres són els següents:

1. (string | number): Valor de l'etiqueta, el grau de la relació. Haurà de concordar amb la definició de les etiquetes que s'ha fet a la primera línia.
2. String2: Nom de l'antibiòtic que es relaciona.
3. String3: Nom del bacteri que es relaciona.

Si es relacionés l'antibiòtic *amikacin* amb el bacteri *pseudomonas* amb un grau de *med*, hauríem d'escriure el següent:

- **Exemple de línia a partir de la 3ª**: PS med amikacin pseudomonas

Una altre relació possible és la que representa el graf d'efectes secundaris, i serà del tipus següent:

- **Sintaxis d'una línia a partir de la 3ª. Possibilitat 2:** SE (string | number) string(2) string(3)

Una relació del graf d'efectes secundaris comença per la paraula clau *se.* Els següents paràmetres són els següents:

1. (string | number): Valor de l'etiqueta, el grau de la relació. Haurà de concordar amb la definició de les etiquetes que s'ha fet a la primera línia.
2. String2: Nom de l'antibiòtic que es relaciona.
3. String3: Nom de l'efecte secundari que es relaciona.

Si relacionéssim l'antibiòtic *amikacin* amb l'efecte secundari *diarrhea* amb un grau de *low*, hauríem d'escriure el següent:

- **Exemple de línia a partir de la 3ª:** SE low amikacin diarrhea

I la última relació és la més complexa, la que relaciona paràmetres del pacient i efectes secundaris. Ha de ser de la següent forma.

- **Sintaxis d'una línia a partir de la 3ª. Possibilitat 3:** ETA (string | number) string(2) string(3) {(string | number)(2) number1 number2 number3 number4 number5 number6}* END

Una relació del graf d'efectes secundaris a evitar comença per la paraula clau *eta.* La part tancada per {}* no serà buida quan es relacionin paràmetres que necessiten a les seves relacions definir funcions per saber el grau final de la relació a partir del paràmetre del pacient. De fet, cada repetició d'aquesta correspon a la definició d'un tros de la funció. Finalment es posa la paraula clau *end* per indicar que s'ha acabat de definir la relació. Els significat dels paràmetres són els següents:

1. (string | number): Valor de l'etiqueta, el grau de la relació. Si estem relacionant un paràmetre booleà, el grau de la relació ha de ser el que és. Si estem relacionant un paràmetre en que el pacient pugui tenir més o menys gravetat d'aquests, s'ha de posar el grau màxim que la relació pugui tenir en el pitjor cas. Si s'està definit una relació on es necessita de posar funcions perquè es relacionen paràmetres com l'edat o el pes,
2. String2: Nom del paràmetre del pacient que es relaciona.
3. String3: Nom de l'efecte secundari que es relaciona.
4. (string | number)(2): Valor màxim que pren el tros de la funció que es defineix, s'omple el camp *value_ini* de la classe *consideration*.
5. Number1: S'omple el camp *funció_inferior* de la taula *consideration*.
6. Number2: S'omple el camp *sota_llindar_inferior* de la taula *consideration*.
7. Number3: S'omple el camp *llindar_inferior* de la taula *consideration*.

8. Number4: S'omple el camp *sobre_llindar_superior* de la taula *consideration*.
9. Number5: S'omple el camp *sobre_llindar_superior* de la taula *consideration*.
10. Number6: S'omple el camp *funció_superior* de la taula *consideration*.

Un exemple de relació, on s'uneix el paràmetre *age*, i l'efecte secundari *diarrhea* amb un efecte de *high* entre 0 i 5 anys, de *null* entre 15 i 55 anys i de *med* entre 65 i 100 anys, amb uns valors a les funcions inferior i superior de 1.5, s'hauria d'escriure al fitxer de la següent forma:

- **Exemple de línia a partir de la 3ª:** ETA null Age Diarrhea high 1.5 0 0 5 15 1.5 med 1.5 55 65 100 100 1.5 end

I així es defineix el fitxer d'entrada, és útil seguir el capítol d'exemples i veure com s'han construït els fitxers d'entrada que s'utilitzen.

8.4 - API

Funcions que es poden utilitzar per carregar les dades des d'un fitxer i a continuació fer proves i utilitzar les dades per treure conclusions i possibles tractaments per a diferents casos de pacients. En negreta el nom de la funció, i a continuació el nom dels paràmetres. En acabat s'explica la forma que han de tenir els paràmetres així com l'objectiu i el resultat de la funció.

- **Load_file** file

  Paràmetres:

  - *File* : De tipus *String*. Ha de ser la ruta d'un fitxer.

  Objectiu: Carrega les dades del fitxer *file*.

  Retorna: Un missatge de que s'han llegit i emmagatzemat les dades correctament.

  Exemple: (load_file "c:/fitxer.txt")

- **Find_node** nom_node &optional l

  Paràmetres:

  - *Nom_node*: De tipus *Symbol*. Representa el nom d'un node.
  - *l*: Llista de nodes. Per defecte és la variable global *list_of_nodes*.

  Objectiu: Troba el node que correspon amb el nom *nom_node* dins de la llista *l* de nodes.

  Retorna: El node que ha trobat.

  Exemple: (find_node 'pneumococcus)

- **Find_edge** from to &optional l

  Paràmetres:

  - *From*: De tipus *Symbol*. Representa el nom d'un node.

- *To*: De tipus *Symbol*. Representa el nom d'un node.

- *l*: Llista d'arestes. Per defecte és la variable global *list_of_edges*.

Objectiu: Troba l'aresta que té com a node origen *From* i com a node destí *To* dins de la llista *l* d'arestes.

Retorna: L'aresta que ha trobat.

Exemple: (find_edge 'amikacina 'e.coli)

- **The_min**

  Objectiu: Crea una etiqueta amb valor mínim (seguint les instruccions del fitxer que s'hagi carregat anteriorment).

  Retorna: L'etiqueta que ha creat.

  Exemple: (the_min)

- **The_max**

  Objectiu: Crea una etiqueta amb valor màxim (seguint les instruccions del fitxer que s'hagi carregat anteriorment).

  Retorna: L'etiqueta que ha creat.

  Exemple: (the_max)

- **The_null**

  Objectiu: Crea una etiqueta amb valor null (seguint les instruccions del fitxer que s'hagi carregat anteriorment).

  Retorna: L'etiqueta que ha creat.

  Exemple: (the_null)

- **My_label** value

  Paràmetres:

- *Value*: De tipus *symbol* per etiquetes qualitatives i de tipus *number* per a etiquetes qualitatives.

Objectiu: Crea una etiqueta amb el valor que se li doni per paràmetre (seguint les instruccions del fitxer que s'hagi carregat anteriorment).

Retorna: L'etiqueta que ha creat. En cas de que el paràmetre no sigui correcte retorna un missatge d'error.

Exemple: (my_label 'low) o (my_label 3.5)

- **Indication_t** nodes &optional label

Paràmetres:

- *Nodes*: Node o llista de nodes del tipus *T_node*.
- *Label*: Una etiqueta, per defecte és la etiqueta amb valor mínim.

Objectiu: Trobar els agents infecciosos, els quals tenen una relació amb *nodes* (que són antibiòtics) que venen per paràmetre, i aquesta relació té una etiqueta de igual o més valor que *label*.

Retorna: La llista d'antibiòtics que s'ha trobat per paràmetre.

Exemple: (indication_t (find_node 'amikacina)) o

(indication_t (list (find_node 'amikacina) (find_node 'cefepime)))

- **Treatment_a** nodes &optional label

Paràmetres:

- *Nodes*: Node o llista de nodes del tipus *A_node*.
- *Label*: Una etiqueta, per defecte és la etiqueta amb valor mínim.

Objectiu: Trobar els antibiòtics, els quals tenen una relació amb *nodes* (que són agents infecciosos) que venen per paràmetre, i aquesta relació té una etiqueta de igual o més valor que *label*.

Retorna: La llista d'antibiòtics que s'ha trobat per paràmetre.

Exemple: (treatment_a (find_node 'pneumococcus)) o

(treatment_a (list (find_node ‘pneumococcus) (find_node ‘e.coli)))

- **Is_cover** nodes_t nodes_a &optional label

  Paràmetres:

  - *Nodes_t*: Node o llista de nodes de tipus T_node.

  - *Nodes_a*: Llista de nodes de tipus A_node.

  - *Label*: Una etiqueta, per defecte és la etiqueta amb valor mínim.

  Objectiu: Trobar si entre tots els antibiòtics *nodes_t* poden combatre els agents infecciosos *nodes_a*, i això vol dir que hi hagi alguna relació entre els antibiòtics amb cadascun dels agents infecciosos amb igual o superior valor a l’etiqueta *l*.

  Retorna: True en cas de complir l’objectiu, Nil en cas de no complir-lo.

  Exemple: (is_cover (find_node ‘amikacina) (list (find_node ‘pneumococcus))) o

  (is_cover (list (find_node ‘amikacina) (find_node ‘cefoxitina)) (list (find_node ‘pneumococcus) (find_node ‘e.coli)))

- **Pharm_adequacy** treatments microorganisms

  Paràmetres:

  - *Treatments*: Llista de nodes de tipus T_node.

  - *Microorganisms*: Llista de nodes de tipus A_node.

  Objectiu: Trobar el coeficient d’adequació en que el tractament que compon tots els antibiòtics de *treatments* és capaç de contrarestar els bacteris *microorganisms*.

  Retorna: Una etiqueta amb el valor del coeficient d’adequació de *Pharm_adequacy* .

  Exemple: (pharm_adequacy (list (find_node ‘amikacina) (find_node ‘ceftazidime) ‘(find_node gentamicin)) (list (find_node ‘pneumococcus) (find_node ‘e.coli)))

- **Contx_adequacy** treatments parameters

  Paràmetres:

  - *Treatments*: Llista de nodes de tipus T_node.

  - *Parameters*: Llista de nodes de tipus P_node.

  Objectiu: Trobar el coeficient d'adequació en que el tractament que compon tots els antibiòtics de *treatments* no provoca efectes secundaris.

  Retorna: Una etiqueta amb el valor del coeficient d'adequació de *Contx_adequacy*

  Exemple: (contx_adequacy (list (find_node 'amikacina) (find_node 'ceftazidime) '(find_node gentamicin)) (list (find_node 'age) (find_node 'pregnant)))

- **Find_treatments** microorganisms parameters file

  Paràmetres:

  - *Microorganisms*: Llista dels noms de nodes de tipus A_node.

  - *Parameters*: Llista de llista de túples, on s'emmagatzema la informació del pacient.

  - *File*: *String* que indica la ruta d'un fitxer.

  Objectiu: Busca un tractament per un pacient, donat el fitxer d'entrada *file*, el conjunt de *microorganisms* (agents infecciosos que ha diagnosticat el metge que pot tenir), i els seus paràmetres donat per *parameters*. Imprimeix tots els conjunts d'antibiòtics possibles ordenats per l'eficàcia en curar el pacient.

  Exemple: (find_treatments '(pneumococcus e.coli) '((age 20) (height 58) (pregnant yes) (renal_failure med)) "c:/fitxer.txt")

- **Printa_noms_nodes** nodes

  Paràmetres:

  - *Nodes*: Node o llista de nodes.

Objectiu: Imprimir els noms dels nodes.

- **Printa_info_arestes** &optional arestes

  Paràmetres:

  - A*restes*: Edge o llista de edges. Per defecte són totes les arestes que hi al model en el moment.

  Objectiu: De cada relació imprimeix el node origen, el node destí i el valor de l'etiqueta de la relació.

8.5 - FITXERS EXEMPLE

Es presenten el conjunt de fitxers d'entrada que completen el capítol d'Exemples, i que s'han utilitzat com a diferents models per fer diversos exemples.

fitxerEntrada1_1.txt

qualitative null low med high final
2 0 0 0 0 0 0
SE high Amox/Clav Risk_penicillin_allergy
SE med Amox/Clav Diarrhea
SE med Ceftriaxone Eosinophilia
SE high Ceftriaxone Risk_when_pregnancy
SE low Cefepime Eosinophilia
SE high Cefepime Risk_when_pregnancy
SE med Cefuroxime Anemia
PS high Amox/Clav Pneumococcus
PS high Ceftriaxone Pneumococcus
PS high Cefepime Pneumococcus
PS med Cefuroxime Pneumococcus
PS med Ceftazidime Pneumococcus
PS high Amox/Clav E.Coli
PS high Ceftriaxone E.Coli
PS high Cefepime E.Coli
PS high Cefuroxime E.Coli
PS high Ceftazidime E.Coli
PS low Amikacin E.Coli
PS med Gentamicin E.Coli
PS high Aztreonam E.Coli
PS med Cefepime Pseudomonas
PS med Ceftazidime Pseudomonas
PS high Amikacin Pseudomonas
PS low Gentamicin Pseudomonas
PS high Aztreonam Pseudomonas
SE med Ceftazidime Eosinophilia
SE med Amikacin Risk_when_pregnancy
SE high Amikacin Renal_malfunction
SE med Gentamicin Risk_when_pregnancy
SE high Gentamicin Renal_malfunction
SE med Aztreonam Eosinophilia
ETA high Allergic-to-Penicillin Risk_penicillin_allergy end
ETA high Pregnant Risk_when_pregnancy end
ETA low Light_renal_failure Renal_malfunction end
ETA high Severe_renal_failure Renal_malfunction end
ETA high Transplanted Renal_malfunction end

fitxerEntrada2_1.txt

qualitative null low med high final
2 0 0 0 0 0 0
SE high Ampicil·lina Diarrhea
SE high Ampicil·lina Eosinophilia
SE low Apicil·lina Rash
SE med Amoxicil·lina Diarrhea

SE low Amoxil·lina Eosinophilia
SE low Amoxicil·lina Rash
SE high Amox/Clav Risk_penicillin_allergy
SE med Amox/Clav Diarrhea
SE low Amox/Clav Rash
SE med Amikacina Risk_when_pregnancy
SE high Amikacina Renal_malfunction
SE high Amikacina Otoxicity
SE low Cefepime Eosinophilia
SE low Cefepime Diarrhea
SE high Cefepime Risk_when_pregnancy
SE low Cefepime Rash
SE high Cefepime Coombs
SE low Cefoxitina Diarrhea
SE low Cefoxitina Eosinophilia
SE low Cefoxitina Renal_malfunction
SE low Cefoxitina Coombs
SE med Ceftriaxone Eosinophilia
SE high Ceftriaxone Risk_when_pregnancy
SE med Cefuroxime Anemia
SE med Ceftazidime Eosinophilia
SE low Ceftazidime Rash
SE med Ceftazidime Coombs
SE low Ceftriaxona Diarrhea
SE med Ceftriaxona Eosinophilia
SE low Ceftriaxona Rash
PS low Ampicil·lina Pneumococcus
PS high Ampicil·lina Strep
PS high Amoxicil·lina Pneumococcus
PS high Amoxicil·lina Strep
PS high Amox/Clav H_inf
PS high Amox/Clav Morax
PS high Amox/Clav Pneumococcus
PS high Amox/Clav Strep
PS high Amox/Clav Staph
PS low Amox/Clav E.coli
PS med Amox/Clav Kleb
PS med Amikacina H_inf
PS med Amikacina Morax
PS med Amikacina Staph
PS high Amikacina E.Coli
PS high Amikacina Kleb
PS high Amikacina Enterob
PS high Amikacina Pseudomonas
PS high Cefepime H_inf
PS high Cefepime Morax
PS high Cefepime Pneumococcus
PS high Cefepime Strep
PS high Cefepime E.Coli
PS high Cefepime Kleb
PS med Cefepime Pseudomonas
PS med Cefoxitina Anaer
PS high Cefoxitina H_inf
PS high Cefoxitina Morax
PS med Cefoxitina Pneumococcus
PS high Cefoxitina Strep
PS med Cefoxitina Staph
PS high Cefoxitina E.coli
PS high Cefoxitina Kleb
PS high Ceftazidime H_inf
PS high Ceftazidime Morax
PS med Ceftazidime Pneumococcus
PS med Ceftazidime Strep
PS high Ceftazidime E.Coli
PS high Ceftazidime Kleb
PS med Ceftazidime Pseudomonas
PS high Ceftriaxone H_inf

PS high Ceftriaxone Morax
PS high Ceftriaxone Pneumococcus
PS high Ceftriaxone Strep
PS high Ceftriaxone E.Coli
PS high Ceftriaxone Kleb
PS high Cefuroxime H_inf
PS high Cefuroxime Morax
PS med Cefuroxime Pneumococcus
PS high Cefuroxime Strep
PS med Cefuroxime Staph
PS high Cefuroxime E.Coli
PS high Cefuroxime Kleb
PS med Gentamicin H_inf
PS med Gentamicin Morax
PS med Gentamicin Staph
PS med Gentamicin E.Coli
PS high Gentamicin Kleb
PS high Gentamicin Enterob
PS low Gentamicin Pseudomonas
SE high Cefuroxime Eosinophilia
SE high Cefuroxime Anemia
SE med Gentamicin Risk_when_pregnancy
SE high Gentamicin Renal_malfunction
SE high Gentamicin Otoxicity
ETA high Allergic-to-Penicillin Risk_penicillin_allergy end
ETA high Pregnant Risk_when_pregnancy end
ETA high Renal_failure Renal_malfunction end
ETA high Transplanted Renal_malfunction end
ETA null Age Diarrhea high 1.5 0 0 5 15 1.5 high 1.5 55 65 100 100 1.5 end
ETA null Age Eosinophilia high 1.5 0 0 10 15 1.5 end
ETA null Age Rash high 1.5 0 0 10 15 1.5 med 1.5 80 85 100 100 1.5 end
ETA low Age Otoxicity high 1.5 0 0 5 15 2 end
ETA low Age Coombs high 1.5 0 0 5 15 1.5 end
ETA null Height Diarrhea med 1.5 0 0 40 55 1.5 end
ETA low Height Anemia high 1.5 0 0 50 55 1.5 end

fitxerEntrada2_2.txt

quantitative 1 3 0
2 0 0 0 0 0 0
SE 3 Ampicil·lina Diarrhea
SE 3 Ampicil·lina Eosinophilia
SE 1 Apicil·lina Rash
SE 2 Amoxicil·lina Diarrhea
SE 1 Amoxil·lina Eosinophilia
SE 1 Amoxicil·lina Rash
SE 3 Amox/Clav Risk_penicillin_allergy
SE 2 Amox/Clav Diarrhea
SE 1 Amox/Clav Rash
SE 2 Amikacina Risk_when_pregnancy
SE 3 Amikacina Renal_malfunction
SE 3 Amikacina Otoxicity
SE 1 Cefepime Eosinophilia
SE 1 Cefepime Diarrhea
SE 3 Cefepime Risk_when_pregnancy
SE 1 Cefepime Rash
SE 3 Cefepime Coombs
SE 1 Cefoxitina Diarrhea
SE 1 Cefoxitina Eosinophilia
SE 1 Cefoxitina Renal_malfunction
SE 1 Cefoxitina Coombs
SE 2 Ceftriaxone Eosinophilia
SE 3 Ceftriaxone Risk_when_pregnancy
SE 2 Cefuroxime Anemia

SE 2 Ceftazidime Eosinophilia
SE 1 Ceftazidime Rash
SE 2 Ceftazidime Coombs
SE 1 Ceftriaxona Diarrhea
SE 2 Ceftriaxona Eosinophilia
SE 1 Ceftriaxona Rash
PS 1 Ampicil·lina Pneumococcus
PS 3 Ampicil·lina Strep
PS 3 Amoxicil·lina Pneumococcus
PS 3 Amoxicil·lina Strep
PS 3 Amox/Clav H_inf
PS 3 Amox/Clav Morax
PS 3 Amox/Clav Pneumococcus
PS 3 Amox/Clav Strep
PS 3 Amox/Clav Staph
PS 1 Amox/Clav E.coli
PS 2 Amox/Clav Kleb
PS 2 Amikacina H_inf
PS 2 Amikacina Morax
PS 2 Amikacina Staph
PS 3 Amikacina E.Coli
PS 3 Amikacina Kleb
PS 3 Amikacina Enterob
PS 3 Amikacina Pseudomonas
PS 3 Cefepime H_inf
PS 3 Cefepime Morax
PS 3 Cefepime Pneumococcus
PS 3 Cefepime Strep
PS 3 Cefepime E.Coli
PS 3 Cefepime Kleb
PS 2 Cefepime Pseudomonas
PS 2 Cefoxitina Anaer
PS 3 Cefoxitina H_inf
PS 3 Cefoxitina Morax
PS 2 Cefoxitina Pneumococcus
PS 3 Cefoxitina Strep
PS 2 Cefoxitina Staph
PS 3 Cefoxitina E.coli
PS 3 Cefoxitina Kleb
PS 3 Ceftazidime H_inf
PS 3 Ceftazidime Morax
PS 2 Ceftazidime Pneumococcus
PS 2 Ceftazidime Strep
PS 3 Ceftazidime E.Coli
PS 3 Ceftazidime Kleb
PS 2 Ceftazidime Pseudomonas
PS 3 Ceftriaxone H_inf
PS 3 Ceftriaxone Morax
PS 3 Ceftriaxone Pneumococcus
PS 3 Ceftriaxone Strep
PS 3 Ceftriaxone E.Coli
PS 3 Ceftriaxone Kleb
PS 3 Cefuroxime H_inf
PS 3 Cefuroxime Morax
PS 2 Cefuroxime Pneumococcus
PS 3 Cefuroxime Strep
PS 2 Cefuroxime Staph
PS 3 Cefuroxime E.Coli
PS 3 Cefuroxime Kleb
PS 2 Gentamicin H_inf
PS 2 Gentamicin Morax
PS 2 Gentamicin Staph
PS 2 Gentamicin E.Coli
PS 3 Gentamicin Kleb
PS 3 Gentamicin Enterob
PS 1 Gentamicin Pseudomonas
SE 3 Cefuroxime Eosinophilia

SE 3 Cefuroxime Anemia
SE 2 Gentamicin Risk_when_pregnancy
SE 3 Gentamicin Renal_malfunction
SE 3 Gentamicin Otoxicity
ETA 3 Allergic-to-Penicillin Risk_penicillin_allergy end
ETA 3 Pregnant Risk_when_pregnancy end
ETA 3 Renal_failure Renal_malfunction end
ETA 3 Transplanted Renal_malfunction end
ETA 0 Age Diarrhea 3 1.5 0 0 5 15 1.5 3 1.5 55 65 100 100 1.5 end
ETA 0 Age Eosinophilia 3 1.5 0 0 10 15 1.5 end
ETA 0 Age Rash 3 1.5 0 0 10 15 1.5 2 1.5 80 85 100 100 1.5 end
ETA 1 Age Otoxicity 3 1.5 0 0 5 15 2 end
ETA 1 Age Coombs 3 1.5 0 0 5 15 1.5 end
ETA 0 Height Diarrhea 2 1.5 0 0 40 55 1.5 end
ETA 1 Height Anemia 3 1.5 0 0 50 55 1.5 end

fitxerEntrada2_3.txt

quantitative 1 3 0
2 0 0 0 0 0 0
SE 3 Ampicil·lina Diarrhea
SE 3 Ampicil·lina Eosinophilia
SE 1 Apicil·lina Rash
SE 2 Amoxicil·lina Diarrhea
SE 1 Amoxil·lina Eosinophilia
SE 1 Amoxicil·lina Rash
SE 3 Amox/Clav Risk_penicillin_allergy
SE 2 Amox/Clav Diarrhea
SE 1 Amox/Clav Rash
SE 2 Amikacina Risk_when_pregnancy
SE 3 Amikacina Renal_malfunction
SE 3 Amikacina Otoxicity
SE 1 Cefepime Eosinophilia
SE 1 Cefepime Diarrhea
SE 3 Cefepime Risk_when_pregnancy
SE 1 Cefepime Rash
SE 3 Cefepime Coombs
SE 1 Cefoxitina Diarrhea
SE 1 Cefoxitina Eosinophilia
SE 1 Cefoxitina Renal_malfunction
SE 1 Cefoxitina Coombs
SE 2 Ceftriaxone Eosinophilia
SE 3 Ceftriaxone Risk_when_pregnancy
SE 2 Cefuroxime Anemia
SE 2 Ceftazidime Eosinophilia
SE 1 Ceftazidime Rash
SE 2 Ceftazidime Coombs
SE 1 Ceftriaxona Diarrhea
SE 2 Ceftriaxona Eosinophilia
SE 1 Ceftriaxona Rash
PS 1 Ampicil·lina Pneumococcus
PS 3 Ampicil·lina Strep
PS 3 Amoxicil·lina Pneumococcus
PS 3 Amoxicil·lina Strep
PS 3 Amox/Clav H_inf
PS 3 Amox/Clav Morax
PS 3 Amox/Clav Pneumococcus
PS 3 Amox/Clav Strep
PS 3 Amox/Clav Staph
PS 1 Amox/Clav E.coli
PS 2 Amox/Clav Kleb
PS 2 Amikacina H_inf
PS 2 Amikacina Morax
PS 2 Amikacina Staph

PS 3 Amikacina E.Coli
PS 3 Amikacina Kleb
PS 3 Amikacina Enterob
PS 3 Amikacina Pseudomonas
PS 3 Cefepime H_inf
PS 3 Cefepime Morax
PS 3 Cefepime Pneumococcus
PS 3 Cefepime Strep
PS 3 Cefepime E.Coli
PS 3 Cefepime Kleb
PS 2 Cefepime Pseudomonas
PS 2 Cefoxitina Anaer
PS 3 Cefoxitina H_inf
PS 3 Cefoxitina Morax
PS 2 Cefoxitina Pneumococcus
PS 3 Cefoxitina Strep
PS 2 Cefoxitina Staph
PS 3 Cefoxitina E.coli
PS 3 Cefoxitina Kleb
PS 3 Ceftazidime H_inf
PS 3 Ceftazidime Morax
PS 2 Ceftazidime Pneumococcus
PS 2 Ceftazidime Strep
PS 3 Ceftazidime E.Coli
PS 3 Ceftazidime Kleb
PS 2 Ceftazidime Pseudomonas
PS 3 Ceftriaxone H_inf
PS 3 Ceftriaxone Morax
PS 3 Ceftriaxone Pneumococcus
PS 3 Ceftriaxone Strep
PS 3 Ceftriaxone E.Coli
PS 3 Ceftriaxone Kleb
PS 3 Cefuroxime H_inf
PS 3 Cefuroxime Morax
PS 2 Cefuroxime Pneumococcus
PS 3 Cefuroxime Strep
PS 2 Cefuroxime Staph
PS 3 Cefuroxime E.Coli
PS 3 Cefuroxime Kleb
PS 2 Gentamicin H_inf
PS 2 Gentamicin Morax
PS 2 Gentamicin Staph
PS 2 Gentamicin E.Coli
PS 3 Gentamicin Kleb
PS 3 Gentamicin Enterob
PS 1 Gentamicin Pseudomonas
SE 3 Cefuroxime Eosinophilia
SE 3 Cefuroxime Anemia
SE 2 Gentamicin Risk_when_pregnancy
SE 3 Gentamicin Renal_malfunction
SE 3 Gentamicin Otoxicity
ETA 3 Allergic-to-Penicillin Risk_penicillin_allergy end
ETA 3 Pregnant Risk_when_pregnancy end
ETA 3 Renal_failure Renal_malfunction end
ETA 3 Transplanted Renal_malfunction end
ETA 0 Age Diarrhea 3 1 0 0 5 15 1 3 1 55 65 100 100 1 end
ETA 0 Age Eosinophilia 3 1 0 0 10 15 1 end
ETA 0 Age Rash 3 1 0 0 10 15 1 2 1 80 85 100 100 1 end
ETA 1 Age Otoxicity 3 1 0 0 5 15 1 end
ETA 1 Age Coombs 3 1 0 0 5 15 1 end
ETA 0 Height Diarrhea 2 1 0 0 40 55 1 end
ETA 1 Height Anemia 3 1 0 0 50 55 1 end

fitxerEntrada2_4.txt

quantitative 1 3 0
2 0.1 1 0.20 0.1 1 0.20
SE 3 Ampicil·lina Diarrhea
SE 3 Ampicil·lina Eosinophilia
SE 1 Apicil·lina Rash
SE 2 Amoxicil·lina Diarrhea
SE 1 Amoxil·lina Eosinophilia
SE 1 Amoxicil·lina Rash
SE 3 Amox/Clav Risk_penicillin_allergy
SE 2 Amox/Clav Diarrhea
SE 1 Amox/Clav Rash
SE 2 Amikacina Risk_when_pregnancy
SE 3 Amikacina Renal_malfunction
SE 3 Amikacina Otoxicity
SE 1 Cefepime Eosinophilia
SE 1 Cefepime Diarrhea
SE 3 Cefepime Risk_when_pregnancy
SE 1 Cefepime Rash
SE 3 Cefepime Coombs
SE 1 Cefoxitina Diarrhea
SE 1 Cefoxitina Eosinophilia
SE 1 Cefoxitina Renal_malfunction
SE 1 Cefoxitina Coombs
SE 2 Ceftriaxone Eosinophilia
SE 3 Ceftriaxone Risk_when_pregnancy
SE 2 Cefuroxime Anemia
SE 2 Ceftazidime Eosinophilia
SE 1 Ceftazidime Rash
SE 2 Ceftazidime Coombs
SE 1 Ceftriaxona Diarrhea
SE 2 Ceftriaxona Eosinophilia
SE 1 Ceftriaxona Rash
PS 1 Ampicil·lina Pneumococcus
PS 3 Ampicil·lina Strep
PS 3 Amoxicil·lina Pneumococcus
PS 3 Amoxicil·lina Strep
PS 3 Amox/Clav H_inf
PS 3 Amox/Clav Morax
PS 3 Amox/Clav Pneumococcus
PS 3 Amox/Clav Strep
PS 3 Amox/Clav Staph
PS 1 Amox/Clav E.coli
PS 2 Amox/Clav Kleb
PS 2 Amikacina H_inf
PS 2 Amikacina Morax
PS 2 Amikacina Staph
PS 3 Amikacina E.Coli
PS 3 Amikacina Kleb
PS 3 Amikacina Enterob
PS 3 Amikacina Pseudomonas
PS 3 Cefepime H_inf
PS 3 Cefepime Morax
PS 3 Cefepime Pneumococcus
PS 3 Cefepime Strep
PS 3 Cefepime E.Coli
PS 3 Cefepime Kleb
PS 2 Cefepime Pseudomonas
PS 2 Cefoxitina Anaer
PS 3 Cefoxitina H_inf
PS 3 Cefoxitina Morax
PS 2 Cefoxitina Pneumococcus
PS 3 Cefoxitina Strep
PS 2 Cefoxitina Staph
PS 3 Cefoxitina E.coli

PS 3 Cefoxitina Kleb
PS 3 Ceftazidime H_inf
PS 3 Ceftazidime Morax
PS 2 Ceftazidime Pneumococcus
PS 2 Ceftazidime Strep
PS 3 Ceftazidime E.Coli
PS 3 Ceftazidime Kleb
PS 2 Ceftazidime Pseudomonas
PS 3 Ceftriaxone H_inf
PS 3 Ceftriaxone Morax
PS 3 Ceftriaxone Pneumococcus
PS 3 Ceftriaxone Strep
PS 3 Ceftriaxone E.Coli
PS 3 Ceftriaxone Kleb
PS 3 Cefuroxime H_inf
PS 3 Cefuroxime Morax
PS 2 Cefuroxime Pneumococcus
PS 3 Cefuroxime Strep
PS 2 Cefuroxime Staph
PS 3 Cefuroxime E.Coli
PS 3 Cefuroxime Kleb
PS 2 Gentamicin H_inf
PS 2 Gentamicin Morax
PS 2 Gentamicin Staph
PS 2 Gentamicin E.Coli
PS 3 Gentamicin Kleb
PS 3 Gentamicin Enterob
PS 1 Gentamicin Pseudomonas
SE 3 Cefuroxime Eosinophilia
SE 3 Cefuroxime Anemia
SE 2 Gentamicin Risk_when_pregnancy
SE 3 Gentamicin Renal_malfunction
SE 3 Gentamicin Otoxicity
ETA 3 Allergic-to-Penicillin Risk_penicillin_allergy end
ETA 3 Pregnant Risk_when_pregnancy end
ETA 3 Renal_failure Renal_malfunction end
ETA 3 Transplanted Renal_malfunction end
ETA 0 Age Diarrhea 3 1 0 0 5 15 1 3 1 55 65 100 100 1 end
ETA 0 Age Eosinophilia 3 1 0 0 10 15 1 end
ETA 0 Age Rash 3 1 0 0 10 15 1 2 1 80 85 100 100 1 end
ETA 1 Age Otoxicity 3 1 0 0 5 15 1 end
ETA 1 Age Coombs 3 1 0 0 5 15 1 end
ETA 0 Height Diarrhea 2 1 0 0 40 55 1 end
ETA 1 Height Anemia 3 1 0 0 50 55 1 end

Jo, Marc Valls Ribas, signo aquest document expressant que en sòc l'autor

És un sistema expert que donat un diagnòstic imprecís d'un metge en un malalt amb una malaltia infecciosa i els paràmetres mèdics d'aquest pacient, retorni com a resultat els possibles tractaments que se li poden subministrar al pacient ordenats per un coeficient d'adequadament, que pugui combatre qualsevol dels possibles bacteris que el metge hagi diagnosticat i vigilant de no provocar efectes secundaris. Aquest sistema expert resoldrà casos senzills i pretén ser la base d'una possible extensió per arribar a resoldre casos més complexos.

Es un sistema experto que dado un diagnóstico impreciso de un médico en un paciente con una enfermedad infecciosa, y los parámetros médicos de este paciente, devuelva como resultado los posibles tratamientos que se le pueden subministrar al paciente ordenados por un coeficiente de éxito, que puedan combatir cualquier de los posibles bacterios que el médico ha diagnosticado, al mismo que tiempo que vigilar de no provocar efectos secundarios. Este sistema experto resuelve casos sencillos y pretende ser la base de una posible extensión para llegar a resolver casos más complejos.

Is an expert system that receives an imprecise diagnosis of a doctor for a patient who has an infectious disease, some medical parameters of him and, finally, returns a list with the possible treatments, sorted by a succes coeficient, to fight all of the bacterias diagnosed for the docotor and trying to no produce any side effect. This expert system solves simple cases but it could be considered the baseline for a future extension to solve harder cases.