# Bioscena
## Simulació d'un sistema biològic evolutiu amb interacció entre els individuus

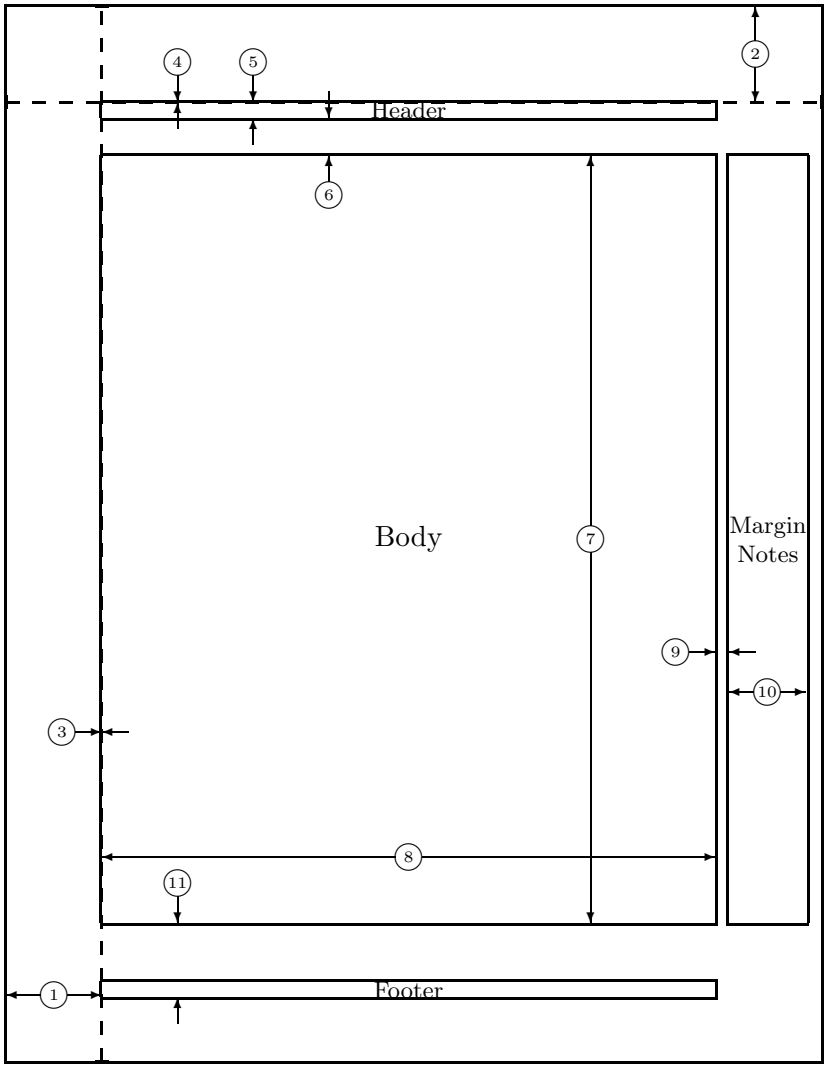
David Garcia
vokimon@jet.es

11 de gener de 2000

# Índex

# Índex de figures

# Índex de taules




| | | | |
|---|---|---|---|
| 1 | one inch + \hoffset | 2 | one inch + \voffset |
| 3 | \oddsidemargin = 0pt | 4 | \topmargin = 0pt |
| 5 | \headheight = 12pt | 6 | \headsep = 28pt |
| 7 | \textheight = 578pt | 8 | \textwidth = 462pt |
| 9 | \marginparsep = 10pt | 10 | \marginparwidth = 59pt |
| 11 | \footskip = 56pt | | \marginparpush = 5pt (not shown) |
| | \hoffset = 0pt | | \voffset = 0pt |
| | \paperwidth = 614pt | | \paperheight = 794pt |

## 0.1 Abstract

Aquest projecte planteja la simulació d'un sistema biològic natural evolutiu amb interacció entre els organismes dins d'un medi de variabilitat controlada. Es tracta de reproduir comportaments naturals o lògics en individuus no cognitius mitjançant el procés evolutiu. Es vol estudiar si el fet d'acostar un algorisme genètic al procés real natural tenint més en compte aspectes biològics i naturals dóna una major adaptabilitat a un medi variable.

La part pràctica consisteix en la implementació d'un prototip que permeti a l'usuari veure les relacions que s'estableixen entre els individuus al llarg del procés evolutiu.

## 0.2 Resum

L'objectiu del present treball de fi de carrera és implementar una eina ampliable d'experimentació pels camps de la biologia i la vida artificial. Aquesta eina simularà un sistema biològic evolutiu amb interacció entre organismes i entre cada organisme i el medi. Ha de permetre a un usuari configurar el sistema, intervindre en la seva dinàmica i oferir eines d'anàlisis per obtenir conclusions.

Tot i que s'intentarà fer un sistema obert que pugui, en el futur, adaptar-se a molts tipus de sistemes, en aquest primer prototip hem implementat els organismes amb un sistema de control que simula els mecanismes de control sobre l'expressió gènica que es donen a la natura.

Per això, primer s'estudiaran els processos naturals (evolutius, etològics i ecològics) prou interessants per introduir-los en el sistema. Es triaran dos tipus de processos: aquells que, d'implementar-los, afegirien realisme al model, i, aquells que s'expera observar en el comportament del biosistema de cara a fer un primer anàlisis.

L'aplicació proveirà eines de configuració i intervenció perque l'usuari pugui controlar la forma en que varia el medi i, així, poder contrastar-ho amb els resultats obtinguts.

També s'implementaran eines d'anàlisis per tal de que es puguin detectar els fenòmens que es considerin interesants en l'estudi previ dels processos naturals.

El sistema ha de ser prou flexible per permetre l'experimentació amb configuracions prou variades. A més, cal donar a l'usuari programador l'espai necessari per modificar algun aspecte concret del model o ampliar les opcions donades, tot modificant el codi font.

# Part I

# Introducció

# Capítol 1

# Introducció al projecte

## 1.1 Àmbit del projecte

Aquest projecte és dins de l'àmbit de la vida artificial (Artificial Life), disciplina que recull coneixements d'informàtica i biologia per recrear fenòmens biològics en un entorn artificial.

Aquesta disciplina va sorgir de cara a la biologia com a camp de proves alternatiu a la vida real, però, les aplicacions van extenent-se, per exemple, aportant noves perspectives a l'analisis, simulació i predicció de sistemes complexos no biològics o a algorismes per la resolució de problemes als sistemes informàtics.

Les aplicacions de la vida artificial tenen un seguit de característiques més o menys comunes. Les principals són:

**Mètode sintètic:** En comptes d'analitzar la vida, sintetitzem artificialment sistemes amb un comportament similar partint de premises que han sigut obtingudes de l'analisis dels sistemes reals.

**Construccio Botton-Up:** Es parteix de unitats petites, definint les interaccions locals, en comptes de partir del comportament global desitjat i anar perfilant com han de ser els components. El comportament global del sistema és un comportament que no estava explícitament

dissenyat.

**Emergència:** És el fet de que apareixin aquests comportaments globals no dissenyats explícitament a partir de l'entramat complex d'interacions simples.

**No lligat als sistemes reals:** Donat el seu caracter sintètic la vida no es limita a la vida coneguda sinó que prova de extreure propietats generals per a qualsevol forma de vida possible.

**Paral·lelisme implícit:** La complexitat dels sistemes vius és deguda, en part, a que els diferents processos es donen en paral·lel. Per fer això en els sistemes de vida artificial, tot i que no sempre sigui possible dedicar un processador a cada procés, sí que caldria fer servir tècniques de temps compartit, que, macroscòpicament, els diferents processos donin la impressió d'executar-se paral·lelament.

## 1.2 Antecedents

Dels treballs que s'han fet sobre vida artificial, cal destacar, com a precedents, per la seva relació amb aquest projecte, els següents:

### 1.2.1 Latent Energy Environment (LEE)

El projecte que aquesta memòria presenta parteix de les experiències que Richard K. Beleg i Filippo Menezer van fer sobre el model Latent Energy Environments (LEE).

LEE és un model que es basa en l'evolució de sistemes cognitius (xarxes neuronals) que interaccionen amb un medi químic de complexitat controlada. La complexitat del medi es controla amb la complexitat d'un model metabòlic: Per obtenir energia útil, a l'organisme no li basta amb introduir els nutrients dins seu, sinó que li cal juntar-los amb altres nutrients que produiran

mitjançant una reacció, un balanç energètic, positiu o negatiu, i uns productes segons s'indica a una taula de reaccions.

Bioscena partirà d'aquesta idea de l'entorn metabòlic de complexitat controlada. La figura 1.1 mostra esquemàticament el sistema metabòlic implementat a LEE. La comparativa es pot fer amb la figura 5.6 que representa el model metabòlic implementat finalment en aquest projecte.

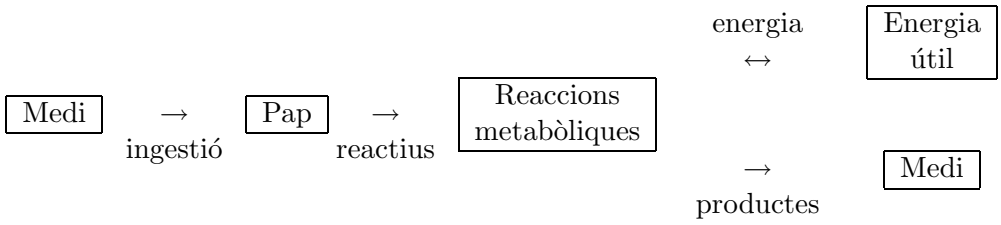



Figura 1.1: Model metabòlic a LEE

La plasticitat, segons Beleg i Menezer [Men94], és la capacitat que té l'organisme de modificar el seu fenotip durant la seva vida. Als LEE aquesta plasticitat els hi dona la xarxa neuronal, un sistema cognitiu. Però, trobem a la natura, que la resta d'organismes que no estan provistos d'un sistema cognitiu d'aprenentatge també presenten una plasticitat equivalent que els hi permet adaptar-se a tot un seguit de canvis.

El que es vol comprovar com a primera experiència amb el sistema és si els sistemes control sobre l'expressió gènica que es donen a natura poden ocupar el paper de les xarxes neuronals en el sentit de donar plasticitat als organismes no cognitius. La funció d'aquest control sobre l'expressió gènica és decidir, en cada moment, quin gens s'expressen i quins no, tot tenint en compte factors del medi i de l'estat intern.

A la natura, aquest control sobre l'expressió gènica, prové d'una certa complexitat en els mecanismes genètics. S'intentaràn implementar els màxims d'aquests mecanismes per tal de determinar l'influència de cadascun sobre la plasticitat de l'organisme, activant uns i desactivant altres.

### 1.2.2 Projecte 'Tierra'

De cara a plantejar com controlar els gens s'ens presenta una altra incògnita prèvia: Als LEE, el codi genètic representa les interconexions i els pesos inicials de les neurones; qué és el que codifiquen els gens al sistema que proposem?

Com a punt de partida per trobar una codificació edient pels gens s'ha pres com a referència el projecte 'Tierra' que encapçala Thomas S. Ray [Ray93]. Thomas S. Ray és un biòleg que va traduir els seus coneixements de genètica i d'ecologia a un medi informàtic on petites porcions de codi competien per la memòria i el temps del processador amb la finalitat de replicar els seus gens. Va obtenir comportaments molt elaborats al llarg de la evolució: parasitisme, simbiosis, curses de braços, comportaments sexuals....

A 'Tierra', el genoma estava composat per un seguit d'instruccions que s'executaven cíclicament. Per modelar el comportament dels organismes de Bioscena, es fan servir, també, ràfegues d'instruccions que formen el genotip de forma semblant a com funcionava 'Tierra'.

Però, a 'Tierra' el propòsit dels fragments de codi era el de replicar-se el més eficientment possible dintre d'un medi que estava composat només pels fragments de codi en competència. Aquí, en canvi, les instruccions hauran d'implementar funcions per interaccionar amb un medi que simularà a un de natural, aliè a la seva estructura interna. Per això, es farà sevir una estructura que farà de pont entre el medi i el codi genètic. D'ara en endavant, aquesta estructura serà la que anomenarem fenotip, tot i tenir present que el fenotip no el forma només aquesta estructura sinó que també hi forma part el comportament global observat i les altres variables d'estat internes de l'organisme. Simplement el diem fenotip perque la seva única funció és, la de fer de fenotip.

El fenotip controla:

- Quin grup d'instruccions s'executa.

- De quina forma ho fan (paràmetres o valors).

Tantmateix, el fenotip és modificat per l'execució d'alguns gens. Aquestes modificacions poden dependre només de la instrucció executada, però, sovint, la modificació també depen de:

1. L'estat de l'entorn de l'organisme
2. L'estat intern de l'organisme


Figura 1.2: Graf de dependències entre el fenotip i els altres elements de l'organisme.

La introducció del fenotip, juntament amb els mecanismes de control de l'expressió génica, marquen una altra diferència amb el model 'Tierra' que té a veure amb el control de fluxe.

A 'Tierra' el genotip és un conjunt d'instruccions que s'executen seqüencialment de forma cíclica. Té un control de fluxe basat en salts a patrons que supleix la debilitat davant de les mutacions que tindria un sistema de salts basats en adreces.

A Bioscena el genotip està composat per diversos conjunts d'instruccions. Un conjunt d'instruccions (gen) necessita que es donin certes condicions al fenotip per poder ser executat.

## 1.3 Objectius del projecte

Els objectius principals del projecte són:

1. Fer un estudi dels processos naturals (evolutius, ecològics i etològics) que es donen a la natura. Per un costat, cal triar els que afegeixirien més realisme i generalitat al model en cas d'implementar-se. Per un altre costat, cal triar aquells fenòmens que es donen en la natura que no cal implementar directament, sinó, que s'espera que puguin emergir. L'esforç

de l'analisis ha d'anar adreçat a modelar aquelles característiques no cognitives de caire general, i sobretot, dels organismes unicel·lulars, tot i que no cal descartar que emergeixin estructures pluricel·lulars o cognitives, com s'ha dit abans.

2. Fer un estudi bibliogràfic dels processos anteriors que ja hagin estat implementats. En quines condicions concretes s'han fet i quins resultats han donat.

3. Dissenyar i implementar el model amb tota la informació recollida, possibilitant, sense dirigir-la, l'aparició dels fenòmens emergents esperats, i l'adaptació dels organismes a variacions que es poden donar durant la vida d'un organisme o al llarg de generacions. El model tindrà diversos elements principals:

    - El biosistema, que coordina la resta d'elements.
    - El biòtop, que és el medi on viuran els organismes. Cal que permeti configuracions molt diverses per donar possibilitats d'experimentació. Per modificar el biòtop farem servir, sobretot, els agents externs, que són modificadors del biòtop dels quals es pot programar el seu efecte al llarg de temps. [1]
    - La comunitat, que controla el conjunt d'organismes presents al medi i la informació que no és intrínseca a ells, com ara la seva posició en el medi i la població a la que pertany. Els organismes proveeixen al medi les accions que volen realitzar quan li són requerides, i un conjunt d'operacions que el medi pot fer sobre ells.

4. Afegir al model les eines d'anàlisis han de poder donar informació útil del que està passant al biosistema: Per això, cal dividir la comunitat en poblacions i analitzar les interaccions entre elles. Cal posar una cura especial en detectar l'aparició de fenòmens emergents. Les eines de configuració i intervenció han de servir per controlar la forma en que canvia aquest

[1] La majoria de la bibliografia de vida artificial anomena agents al que aquí anomenem organismes. Cal que quedi clar que els agents, en aquest projecte, no són els individuus que poblen el món, sinó una eina de configuració.

medi, de forma que, els resultats obtinguts amb les eines d'analisis siguin confrontables i els usuaris puguin extreure conclusions.

A aquests objectius es poden sumar altres objectius adicionals que es cobriran de forma secundària.

1. Els elements del sistema han d'estar dèbilment acoblats per tal de poder, en un futur, intercanviar-los per uns altres de forma individual amb un mínim d'efectes laterals.
2. El sistema ha de poder abocar-se totalment o parcial a disc per tornar-se a restaurar, posteriorment. Això faria possible execucions molt més llargues i obtenir poblacions més evolucionades.
3. Ha de ser possible canviar alguns aspectes de la configuració del biòtop o intervindre en la població amb extraccions, introduccions, clonacions... sobre la marxa.
4. Establir uns procediments de modificació per tal de que l'usuari-programador no necessiti comprendre les interioritats de tot el sistema per fer una modificació puntual.

Els organismes de la simulació han de fer front a un medi variable, amb canvis caòtics o periòdics que es poden produir de dos formes: freqüentment al llarg de la vida d'un organisme o de forma progressiva en el decurs de diverses generacions. Amb la finalitat de que la comunitat tingui capacitat de reaccionar davant de tots aquests canvis, s'incorporen, als mecanismes evolutius que intervenen en la simulació, algunes característiques que es donen en els entorns evolutius naturals i que, clàssicament, no s'incorporen en els entorns evolutius computacionals.

Les característiques naturals que s'estudiarà introduïr a l'entorn evolutiu són:

**Mecanismes d'expressió gènica (transcripció, maduració i traducció):** Útils per implementar les altres característiques

**Regulació sobre l'expressió gènica:** Control dels gens que es transcriuen segons la presència o no d'alguns factors. Ajuda a adaptar-se, sense l'utilització d'un sistema cognitiu, a canvis en el medi tan esporàdics que el procés evolutiu no els soporti.

**Promotor/teminador, zones no codificadores i longitud de cromosoma variable:** Permet solucions obertes no parametritzades (Nombre i longitud variable pels gens)

**Paràmetres del procés evolutiu codificats parcialment al genotip:** Ens permetrà tenir uns paràmetres optimitzats per a cada situació concreta.

**Cariotip multicromosòmic:** Divideix la dotació gènica en subunitats d'alt nivell que permeten traspasar de cop divers material genètic entre organismes i possibilita les mutacions cariotípiques [2]. Amdues coses poden emergir en un creuament.

**Genotips no haplonts i al·lels (Encara he de considerar si implementar-ho):** Mantenen la variabilitat genètica de la descendència augmentant així la capacitat de canvi i adaptació.

Per a la simulació es suposa, per a totes les decisions de disseny on sigui necessari fer-ho, que els organismes són unicel·lulars. Aquesta suposició no vol dir que els resultats siguin aplicables només a aquest tipus d'organisme, però, facilita el disseny, donat que el model d'un organisme pluricel·lular és molt més complexe. A més, hi ha la possibilitat, si el medi fós suficientment ampli, de que l'evolució portés als organismes unicel·lulars a formar estructures cooperatives més grans similars als organismes pluricel·lulars.

## 1.4 Contingut de la memòria

TODO: Quan l'estructura de la memòria estigui més o menys establerta ho poses

[2] Per diferenciant-les de les gèniques o cromosòmiques. Els tipus de mutacions possibles es van comentant més endavant en la memòria.

# Part II

# Part teòrica

# Capítol 2

# Coneixements teòrics sobre biologia

## 2.1 Introducció

Aquest capítol introdueix alguns conceptes sobre procediments que es donen a la natura que necessitem coneixer per, posteriorment, aplicar-los en dos àmbits. Per un costat, necessitem coneixer els procediments genètics que es donen a la natura per adaptar-los als algorismes genètics. Per un altre costat, necessitem coneixements sobre etologia i ecologia per interpretar i analitzar els resultats de la simulació.

A aquest projecte, quan es presenti l'opció d'una nomenclatura o una altra, farà servir la nomenclatura i els conceptes biològics, donat que els usuaris finals seran pretesament biòlegs. Per això, un altre objectiu d'aquest capítol és introduir aquests conceptes i aquesta nomenclatura a tot aquell que no estigui familiaritzat.

## 2.2 Genètica mendeliana

### 2.2.1 Gen, Al·lel, Genotip i Fenotip

Els primers estudis genètics de la història [MENDEL 1866] van pendre una perspectiva externa als individuus per estudiar l'herència als organismes que es reprodueixen sexualment. És a dir,

a partir dels caràcters observables d'individuus de diferents generacions, van intentar deduir els mecanismes o factors que intervenen en l'herència.

Aquest estudis, i les seves ampliacions posteriors, van madurar un seguit de conceptes que han perdurat fins avui en dia. S'expliquen a continuació:

El caràcter observable en el que ens fixem l'anomenem **fenotip**, per exemple, el color dels ulls. Un **gen** és cadascun dels factors hereditaris que controlen aquest fenotip. El normal és que siguin diversos gens els que controlin un fenotip el conjunt dels quals formen el seu **genotip**. Per exemple, imaginem que el color dels ulls el controla un genotip format per dos gens. Generalment, la correspondència entre genotip i fenotip no és directa, perque en el fenotip pot intervindre l'entorn. El conjunt de tots els gens que té un organisme (no pas considerant un sol fenotip) és el **genoma**.

Un **al·lel**, és cadascuna de les alternatives (o "valors") que pot adoptar un gen. Per exemple, imaginem que tenim les alternatives A, B, C y D. Dos al·lels es consideren diferents només si, en algun cas, el fet de tenir un o l'altre afecta al fenotip obtingut. Dos gens poden tenir els mateixos al·lels possibles. Si això passa i, a més, l'efecte d'un és equivalent al de l'altre, diem que són **gens homòlegs**. El genotip és **homozigot** si els seus gens només presenten un al·lel per cada conjunt de gens homòlegs (Raça pura). En canvi, el genotip és **heterozigot** si els seus gens presenten al·lels diversos (Híbrid).

### 2.2.2 Expressió dels al·lels al fenotip

Un individuu homozigòtic presenta el caràcter fenotípic representatiu de l'al·lel. Aquest caràcter s'en diu el fenotip homozigòtic per aquest al·lel.

En la natura, sovint, els genotips homozigòtics estan associats, directament o indirecta, a fenotips negatius. Això és degut a que els individuus homozigòtics donen molt poca variabilitat genètica. Si una població acaba convertint-se en homozigotica, per exemple, per excesiva en-

dogàmia, aquest gen es queda estancat i no dòna variabilitat. Si, per adaptar-se a un canvi en l'entorn cal canviar aquest gen, una població homozigòtica tindrà menys capacitat de resposta perqué dependrà de les mutacions. En els gens que no interesa aquest estancament la pròpia dotació genètica s'autopenalitza. Si una població no castiga la homozigosis, té més probabilitat de tornar-s'en.

A vegades la millor forma de penalitzacio són els gens letals. Un genotip letal és aquella combinació d'al·lels que no es presenten mai donat que els individuus no sobrepassen l'estat embrionari.

Quan el genotip és heterozigot ens podem trobar les següents relacions entre els al·lels dels gens homòlegs, segons la forma d'expressar-se en el fenotip:

- **Dominància - Recessivitat:** Un al·lel (dominant) s'imposa sobre l'altre (recessiu), i, el fenotip resultant és el mateix que el d'un homozigot amb l'al·lel dominant.
- **Herència intermitja:** El fenotip no és l'equivalent a cap dels dos fenotips homozigots sinó que és un fenotip intermig.
- **Codominància:** Es presenten els fenotips dels dos al·lels a la vegada.
- **Superdominància o heterosis:** La presència de l'al·lel recessiu reforça el fenotip de al·lel dominant més que si fós un homozigot.

### 2.2.3 Fenotips de distribució contínua

Alguns caràcters de l'individu, com per exemple l'altura o la inteligència, no presenten unes alternatives fenotípiques tan discontínues sinó que són més contínues.

El caràcter pot dependre de diversos gens **poligen** amb la qual cosa es produeix una distribució normal en el caràcter. Com més gens s'hi impliquin, més graus discontinus hi haurà a la

distribució. Per exemple:

| Número de gens implicats | Distribució del caracter |
|---|---|
| 1 | 1:1 |
| 2 | 1:2:1 |
| 4 | 1:2:3:4:3:2:1 |
| 6 | 1:6:15:20:15:6:1 |

TODO: Pegar un parell de gràfics

Arriba un moment que hi intervenen tants gens que el gradient no és identificable. A més, considerant el fet de que l'entorn afecta al fenotip, és possible que un individuu amb genotip AABBBB sigui fenotipicament més semblant a un homozigot A que un individuu amb genotip AAABBB.

### 2.2.4 Interaccions entre gens no homòlegs

Així com diversos gens homòlegs poden contribuir al fenotip d'un caràcter, gens no homòlegs també poden contribuir-hi, però en aquest cas la influència dels gens no es equivalent (per la definició anterior de gens homòlegs).

Una de les interaccions entre gens no homòlegs és la **epistàsia**: un gen (epistàtic) controla l'activació o desactivació d'un altre (hipostàtic).

### 2.2.5 Mutació gènica

El concepte d'herència estàtica, tal i com el definia Mendel, s'enfrontava amb les noves idees de Darwin sobre l'evolució [DARWIN 1859]. El concepte que tenia Darwin sobre l'evolució és que als individuus es produien petits canvis (mutacions) dels quals la natura escollia els més apropiats per a l'entorn. En el concepte de Mendel sobre l'herència, no es creaven al·lels nous, les generacions posteriors tenen simplement una recombinació dels al·lels de les generacions anteriors. De Vries, un dels redescobridors dels postulats de Mendel, va introduir el concepte de mutació gènica [DEVRIES 1900] que ve a reconciliar els dos corrents i donar una explicació a l'aparició

d'al·lels diferents dintre d'un gen.

Una mutació genica o puntual és l'aparició sobtada d'una nova alternativa (al·lel) per a un gen.

La mutació (tan si és gènica com si és una altra de les que veurem més endavant) és un fenòmen aleatori que es dona amb una determinada freqüència que sol ser molt baixa. La **freqüència de mutació** és una probabilitat que es mesura per a un gen donat, i durant una generació.

La probabilitat de mutació génica és mantè constant, tot i que diferent per a cada gen. Això s'explica per la diferent longitud de gens, o per la quantitat de mutacions que no impliquen un canvi d'al·lel (com es veu a la secció **??**).

Als organismes pluricel·lulars, les mutacions poden ser **somàtiques** o **germinals**. Una mutació somàtica, només afecta a una cel·lula i a totes les que en deriven. Per exemple, els tumors, les pigues... són mutacions somàtiques. Una mutació germinal es produeix a les cèl·lules del teixit que donaran lloc a les gametes. En conseqüència, aquesta mutació afectara, no només a la cèl·lula i a les que en derivin sinó que també a la descendència.

### 2.2.6 Sumari de conceptes aplicables al projecte

L'aproximació mendeliana als mecanismes de l'herència, tot i ser una primera aproximació, ja ens dona alguns conceptes que no són presents a la implementació clàssica dels algorismes genètics.

El primer concepte és el de gens homòlegs. A l'algorisme genètic clàssic, els gens no tènen homòlegs a menys que es considerin així a la funció d'evaluació i, generalment, no es fa. En el cas de tenir homòlegs, caldria resoldre el problema de l'heterozigosi la qual cosa implica un cost computacional superior que és sovint inútil en els problemes que tenen una funció d'avaluació constant. Aquest cost computacional, com a mínim, és el que implica duplicar la informació continguda al cromosoma.

En canvi, als problemes on la funció d'avaluació varia al llarg del temps, com és el cas d'un entorn biològic, és positiu guardar-se aquesta variabilitat en forma d'heterozigosis. Un genotip que no tingui gens homòlegs té els mateixos desavantatges que s'han comentat abans per un genotip homozigot, però, amb el desavantatge afegit de que mai pot arribar a ser heterozigot.

El concepte de mutació gènica, és el concepte de mutació puntual dels algorismes genètics clàssics. El que s'aporta de nou és el concepte de probabilitat de mutació variable segons el gen, i la diferència entre mutació somàtica i germinal de cara a transmetre les mutacions a la descendència. Als apartats següents anirem modificant i enriquint el concepte de mutació a mida que es vagi desgavellant, en aquest capítol, la natura dels gens.

TODO: Cercar Referencia d'algu que faci servir heterozigosis (sense diplonts?)

De cara a l'anàlisis dels resultats, pot ser interesant detectar si hi ha algun mecanisme que afavoreixi els genotips heterozigots i la presència de fenotips continus. També pot interesar detectar si es donen casos d'epistàsia, tot i que, com que els mecanismes d'epistàsia poden ser molt complexos, caldria limitar a alguns casos de baix nivell.

## 2.3 Teoria cromosòmica

### 2.3.1 Els cromosomes i l'herència

Els estudis de Morgan et al. varen demostrar que els gens no eren quelcom independent sinó que estaven en estructures superiors formant els **cromosomes**.

El gens es troben localitzats en un punt concret del cromosoma (**locus**). Quan reconvinem el material genètic de dos progenitors, els gens a locus més propers dins d'un cromosoma tenen molta més probabilitat d'heretar-se conjuntament. Això implica, els gens que controlen cada caràcter no s'hereten de forma tan independent com formula la tercera llei de Mendel. La tercera llei de Mendel diu que els caracters s'hereten de forma independent. Això és veritat, no en els caracters

sinó en els gens i sempre que els gens tinguin un locus suficientment allunyats o en cromosomes diferents.

Anomenem **cariotip** al conjunt de cromosomes que té una espècie (en quant a nombre i morfologia).

Els cromosomes són homòlegs si tènen la mateixa morfologia i contènen gens homòlegs als mateixos *locus*.

Un cariotip és **diplont** si està format per n parelles de cromosomes homòlegs. És a dir, cada cromosoma té un altre d'homòleg, de tal forma que hi ha dos dotacions cromosòmiques homòlogues. Generalment són organismes que es reprodueixen sexualment, i, de cada parella d'homòlegs, cada progenitor n'ha aportat un.

Quan només hi ha una dotació cromosòmica el cariotip es diu que és **haplont**. Si n'hi ha tres dotacions homòlogues, **triplont**, si n'hi ha quatre, **tetraplont** i, si n'hi ha més, **poliplont**.

TODO: Estructura d'un cromosoma: Centròmer. cromàtides...

### 2.3.2 Mitosi i meiosi

TODO: Mitosis

TODO: Gemació vs. Bipartició, Com repartir els recursos entre pare i fill

TODO: Meiosis i la recombinació

### 2.3.3 Reproducció i sexualitat. Cicles biològics

Sexualitat i reproducció són dos procesos amb origen evolutiu i funció diferent. Si bé l'objectiu de la reprodució era obtindre individuus que siguin còpies genètiques dels seus progenitors, l'objectiu de la sexualitat és el de la recombinació genètica per provar generar mes variabilitat genètica.

Alguns organismes unicel·lulars, per exemple, tenen comportaments sexuals no lligats a la reproducció com ara la conjugació que consisteix en l'intercanvi simple de material genètic sense que es produeixi cap nou individuu.

TODO: La conjugacio no es algo mes? No sera altra cosa el que dius?

Dels procesos sexuals en resulta una gama d'individuus diferents molt més amplia del que en resulta amb processos exclusivament asexuals. Això dona més agilitat al procés evolutiu, sobretot de cara a variacions en el medi. Permet que, si un organisme adquireix per mutació una característica positiva, aquesta característica es propagui per la població sense necessitat de que hi hagi una substitució dràstica de la descendència sense mutació per la descendència amb mutació.

Als organismes que es reprodueixen sexualment, a la fecundació, sempre intervenen dos cèl·lules haplonts (gametes), una de cada progenitor, que es junten per formar un zigot diplont. Pot passar que la meiosi es produeixi abans de la fecundació, i que els individuus madurs siguin diplonts (**cicle diplont**) o que la meiosis es produeixi després de la fecundació amb la qual cosa els individuus son haplonts (**cicle haplont**).

Alguns organismes alternen generacions haplonts amb les diplonts (**cicle haplodiplont**. Una generació haplont produeix les gametes que intervenen a la fecundació, formant un zigot que dona un individuu diplont. Aquest crea cèl·lules esporogènees (diplonts) que per meiosi dona lloc a meiospores haplonts de les que es torna a formar un individuu haplont.

TODO: Diagrames dels tres cicles

TODO: Avantatges de cada cicle

### 2.3.4 Mutacions cromosòmiques

El fet de que els gens estiguin dins de l'estrutura cromosòmica dona lloc a altres tipus de mutacions que no són pas les gèniques. Ténen a veure amb la distribució dels gens a dins dels cromosomes, i, així com les mutacions gèniques explicaven l'origen dels diferents genotips, les mutacion cromosòmiques expliquen l'origen dels diferents cariotips per cada espècie.

Les **mutacions cromosòmiques estructurals** són aquelles en les que no intervenen cromosomes integres sino fragments d'aquests.

- Es produeix una **deficiència o delecció** quan un cromosoma en perd un segment. Les causes poden ser una falla en el procés de replica o, en el moment del crossover.

  TODO: Causes de la delecció

  TODO: Efectes gairebé sempre negatius

- La **duplicació** consisteix en el fet de que un segment cromosòmic es dupliqui, generalment, en sèrie.

- La **translocació** és una mutació cromosòmica que consisteix canviar el locus d'una seqüència gènica.

  TODO: DUDA: En el mateix cromosoma o entre homolegs o entre no homolegs??

  TODO: Recíproca o no recíproca

  TODO: DUDA: Jo em pensava que la translocació era com el shift N de AG.

  Si un organisme es reprodueix sexualment, sovint, la translocació causa, en els descendents, duplicació o deficiencia del gen translocat.

- La **inversió** consisteix en el canvi de sentit d'un segment de cromosoma.

De banda de les mutacion estructurals, es donen **mutacions per canvi en el nombre de cromosomes**. La causa del canvi en el nombre de cromosomes pot ser:

- **Fusió cèntrica:** Dos cromosomes no homòlegs es fusionen pel seu centròmer.
- **Escissió cèntrica:** Un cromosoma se escindeix en dos pel seu centròmer.
- **No disyunció en la meiosi:** En la meiosi no es reparteixen per igual les cromàtides de tal forma que una gameta es queda amb més cromàtides del normal i l'altre, amb menys. Es tracta d'una **euploidia** si és un canvi en el nombre de dotacions gèniques. Per exemple, que d'un diploide en sorgeixi un triploide. Pel contrari, si el que hi ha hagut és la falta o la duplicació d'un sol cromosoma, es tracta d'una **aneuploidia**

### 2.3.5 Sumari de conceptes aplicables al projecte

La influència del locus de cada gen en el fet de que dos gens tendeixin a heretar-se junts, és un efecte que, als algorismes genètics clàssics i amb segons quins problemes, resulta negatiu. A la natura és un efecte positiu per que la posició dels gens no es fixa sino que evoluciona juntament amb els individuus i, si dos gens són bons si s'hereten junts, l'evolució tendirà a posar-los junts en el cariotip, i, si fós bo que es recombinin de forma equitativa, l'evolució tendirà a posar-los separats.

A la simulació, de cara a deixar via oberta a diversos comportaments sexuals o a organismes asexuals, farem servir la separació entre els conceptes de sexualitat i reproducció.

TODO: Perque la farem servir?

TODO: Referències a implementacions diplonts dels GA

La introducció d'organismes diplonts o poliplonts suposaria implementar un mecanisme de resolució del fenotip heterozigòtic als gens homòlegs. També sorgirien alguns altres problemes,

relacionats amb el creuament, que es comenten en el següent apartat.

## 2.4 Genètica biomol·lecular

### 2.4.1 Concepte clàssic de gen

Fins ara, hem considerat un gen com quelcom indivisible. Abans de coneixer a fons l'estructura mol·lecular de l'ADN, els biòlegs consideraven el gen com a:

- **Unitat funcional:** De cara a controlar un caràcter.
- **Unitat de reconbinació:** Unitat estructural bàsica e indivisible del cromosoma.
- **Unitat de mutació:** El gen és el que canvia com un tot.

Més tard, van descobrir que els cromosomes estaven formats per seqüències d'ADN. L'estudi mol·lecular de l'ADN va donar una idea més en detall de com s'expressen, com es reconbinen i com muten els gens.

### 2.4.2 Estructura mol·lecular de l'ADN

monocatenario bicatenario circular linial

Model de Watson i Crick (Heleicoidal)

direccional, si és bicatenari cada cadena és complementaria però, la direcció de transcripció és inversa.

Bases al ADN

| **Púriques** | | **Pirimídiniques** |
|---|---|---|
| Adenina | lliga amb | Timina |
| Guanina | lliga amb | Citosina |

Bases al ARN

| **Púriques** | | **Pirimídiniques** |
|---|---|---|
| Adenina | lliga amb | Uracil |
| Guanina | lliga amb | Citosina |

### 2.4.3 Expressió gènica

L'expressió gènica és la forma que tènen els gens, continguts en els cromosomes, per arribar a afectar al caràcter fenotípic que controlen. En general, cada gen contingut als cromosomes està associat a la producció d'una proteina que pot ser enzimatica o estructural. A més, aquesta associació és linial, com ja veurem, això vol dir que

A la natura, l'expressió gènica no es fa directament de l'ADN a les proteines sinó que es fa servir unes cadenes d'ARN com a intermediaries. En conseqüència la expressió gènica es fa en tres passos:

#### Transcripció

La transcripció és el primer pas de l'expressió gènica i l'únic en el que pren part l'ADN. Consisteix en sintetitzar una cadena d'ARN complementària a una subseqüència d'ADN. Es divideix en tres fases:

- **Fase d'iniciació:** Primer, l'enzim transcriptor reconeix una seqüència de nucleòtids, anomenada **regió promotora**, que és la que indica el punt de la cadena on cal començar una transcripció.
- **Fase de allargament de cadena:** Després, a mida que avança en la direcció de transcripció, va enganxant nucleòtids d'ARN complementaris als que es va trobant a la cadena d'ADN.

  TODO: Comentar el tema de la direccio de transcripció.
- **Fase de finalització:** El procés arriba a la seva fi, quan l'enzim arriba a una seqüència d'ADN determinada, **regió terminadora**, que indica la seva fi.

### Processat o maduració

Les cadenes d'ARN$_m$ inmadur, a les cèl·lules eucariotes (amb nucli diferenciat), pateixen tot un seguit de transformacions abans de traduïr-se als ribosomes. Aquest processat no es fa als organismes procariotes (nucli dispers) donat que ARN$_m$ es tradueix directament, abans, i tot, de acabar-se de transcriure.

Per un costat, la seqüència d'ADN que hi ha transcrita al ARN$_m$, no tota conté informació útil. Els **exons** són els segments que codifiquen informació útil, i els **introns** són els segments que no. Una de les transformacions que es fan en aquesta fase és eliminar els introns.

TODO: Lider i trailer

Una altra transformació és afegir al lider i al trailer un cap i una cua respectivament que ajudaran a iniciar i finalitzar la traducció.

Algunes mol·lecules d'ARN (ARN$_t$, ARN$_r$...) que també es transcriuen de l'ADN, pateixen un processat més especialitzat. Aquestes mol·lècules d'ARN no es fan servir per codificar proteines però, com es veurà en la fase següent, tenen un paper importantísim en la síntesis de proteines.

### Traducció

Durant la traducció, l'ARN$_m$ madur s'interpreta per anar enganxant la seqüència d'aminoàcids d'una proteina.

Cada tres nucleòtids d'ARN$_t$ formen un codó. Cada codó té un aminoàcid associat, segons la taula de sota. La concatenació dels aminoàcids segons la seqüència de codons és el que forma la proteina.

Els quatre valors possibles pels tres nucleòtids d'un codó donen $4^3 = 64$ combinacions. Però, a la natura, es donen només 21 aminoàcids. Es dedueix, llavors, que **la codificació és redundant**

TODO: Taula del codi genètic

Taula 2.1: Codi Genètic. Diferents convinacions de codons es corresponen a un sól pèptid.

i produeix **sinònims**. Un parell o tres de codons són **muts** i no tenen traducció. Tot i no tenir un aminoàcid associat, veurem que els codons muts són molt útils.

Aquesta taula és el codi genètic que tradueix els codons a aminoàcids:

L'associació la fan mol·lècules d'ARN$_t$ que tènen a un extrem un anticodó per lligar-se a un codó, i, per l'altre extrem un radical amb el qual es lliguen a l'aminoàcid corresponent. Les mol·lècules d'ARN$_t$ també estan codificades a l'ADN i, per tant, l'associació codó-aminoàcid és també informació genètica. Sorprenentment, el codi genètic és pràcticament universal, donant-se petites variacions, només a l'ADN mitocondrial. La raò és que en els organismes mínimament evolucionats, un canvi en el codi genètic suposaria tants canvis que segurament serien letals, en canvi en els organismes primigenis, seria possible trobar més diversitat de codis. [1]

TODO: AUG (TAC) Sempre el primer, UGA (ACT) (Mut) sempre l'últim

TODO: Reutilització d'ARN$_t$

### 2.4.4 Regulació de l'expressió dels gens

Mutacions i creuaments són els mecanismes d'adaptació al medi que permeten a una població adaptar-se de generació en generació als canvis graduals en el medi. Però, hi ha canvis que són tan freqüents que els ha d'afrontar l'individuu que els pateix i no es pot esperar a que varii el genotip de la seva descendència. Són les respostes que produeix l'individuu a les variacions del medi.

Perque un organisme es pugui adaptar a diverses situacions de l'entorn cal que tingui mecan-

[1] Això dòna una idea de en quin punt de l'evolució, les mitocondries passaren simbiòticament a formar part dels altres organismes

ismes per regular l'expressió dels seus gens segons aquestes situacions. Per exemple, si el medi es torna massa àcid cal generar enzims que ho compensin, però, quan es massa bàsic, la producció d'aquests enzims, no només és un gast inútil d'energies sinó que podria ser contraproduent.

Cal llavors un mecanisme que permeti a l'organisme controlar la seva producció enzimàtica. Aquest control no es podria fer efectiu, si l'$ARN_m$ no tingués una vida molt limitada. La vida de l'$ARN_m$ i la vida de la majoria de proteines enzimàtiques, ve fixada per un compromís entre economia en la seva producció i la capacitat de reacció de l'organisme.

Aquest límit en la vida del $ARN_m$ i de les proteines que en genera, ens permet una regulació basada en el control de la producció d'$ARN_m$. Si es deixa de produir, hi haurà un moment en que les proteines que genera no hi seran presents. A continuació s'explica els factors que intervenen en la síntesis d'$ARN_m$ per a un gen donat.

El gens tenen una **probabilitat transcripció** que depèn de l'afinitat de l'enzim transcriptor amb el seu promotor i de la seva repetició al llarg del genotip. Aquesta probabilitat és inherent al genotip, però, pot ser modificada amb **regulació activa**. Segon l'efecte de la regulació diem que el control que fa és:

- **Control negatiu:** Si es fa mitjançant **agents represors** que impideixen l'unió de l'enzim transcriptor amb el promotor bloquejant la síntesi de $ARN_m$.
- **Control positiu:** Si es fa mitjançant **agents activadors** que es junten amb el promotor per fer-lo més afí amb l'enzim transcriptor.

Els agents represors/activadors són proteines que es sintetitzen també a partir de l'ADN i que actuen o no, segons les condicions d'ambient. Quan aquestes condicions ambientals són principis actius es diu que són:

- **Sistemes inducibles:** Si actuen per la ausencia d'un principi actiu.

- **Sistemes represibles:** Si actuen per la presència d'un principi actiu (correpressor o coactivador).

TODO: No és massa clar que també siguin aquests noms al control positiu

Per exemple, si el gen que volem regular és un catabolitzador d'una sustància A, l'organisme pot fer servir un control negatiu induible, de tal forma que es desactivi quan no hi hagi A, i/o un control positiu represible, perque s'acceleri la producció quan hi hagi A.

### 2.4.5 Mutació i creuament a nivell mol·lecular

Com que amb la genètica mol·lecular hem vist que la unitat de convinació no era el gen sinó el nucleòtid, cal reformular els conceptes de mutació gènica i cromosòmica.

La mutació puntual és causada per un canvi en les bases dels nucleòtids de l'ADN, deguda a l'inestabilitat química del propi ADN o a agents externs.

- **Mutació per transició de bases:** Intercanvi per l'altre base del mateix grup (púriques o pirimírique)
- **Mutació per transversió de bases:** Intercanvi per la base no complementaria de l'altre grup.
- No es donen, de forma natural, les mutacions directes entre bases complementaris.

TODO: Contrastar l'afirmació anterior amb un expert (Pepi)

TODO: Figura amb el quadre de mutacions de base

Tant la transició com la transversió, es donen per la modificació química d'una de les bases (tautomerització) que, en duplicar-se l'ADN, no es lliga amb la seva base complementària sinó amb una altra. Quedant les bases desajustades. Encara cal que passi per uns quants filtres perque la mutació es faci efectiva a la descendència:

- Existeixenn mecanismes de reparació basats en el fet de que, tot i que una base hagi canviat, l'altre pot seguir igual, si no són complementàries, vol dir que hi ha hagut una mutació. L'enzim corrector modifica una de les dues bases per tal de que siguin complementàries, però, pot modificar la mutada o la bona.

- En duplicar-se la cadena, per una divisió cel·lular, si encara no s'ha 'corregit´ la mutació, la meitat de la descendència durà la mutació complerta i l'altra meitat el genoma primitiu.

- Tant si es repara com si es queda la mutació sense reparar, la probabilitat de traspasar-la a un descendent és del 50%.

- La mutació es pot produir a un segment d'ADN no codificant (introns i zones intermitges)

- Els codons sinònims fan que alguns canvis en les bases no impliquin canvi de pèptid.

  TODO: Calcular el tant percent, quan tinguis una estona

- Alguns canvis de pèptids als extrons tampoc no són significatius pel fenotip.

  TODO: Referència a Kimura

- En organismes pluricel·lulars, cal que sigui una mutació al teixit germinal que dona lloc als nous individuus. Si es dona al teixit somàtic, només afecta a les cèl·lules filles al mateix organisme.

TODO: Mutacions no per canvi de base

TODO: Mutacions cromosòmiques a nivell mol·lecular

TODO: Creuament a nivell mol·lecular

### 2.4.6 Sumari de conceptes aplicables al projecte

A nivell mol·lecular, trobem alternatives molt riques al AG clàssic. Algunes de les quals han estat estudiades en la bibliografia.

El gens són de longitud variable. El que ho permet són les seqüències promotora i terminadora que els delimiten. Mayer va experimentar aquest tipus de codificació, mitjaçant promotors i terminadors, en cromosomes de longitud fixa, variant el nombre, posició i longitud dels gens/paràmetres. Raich va trobar ideal una codificació molt semblant (feia servir una longitud en comptes d'una seqüència terminadora) per problemes orientats a disseny que ténen un nombre variable de paràmetres i que requereixen solucions obertes.

No tot l'ADN es transcriu, com a mínim les seqüències promotores i terminadores, no ho fan. A més, tenim els introns que s'eliminen durant la fase de maduració.

TODO: L'unitat de mutació, convinació... és molt més petita que un gen.

TODO: La síntesis és direccional

TODO: Els gens estan limitats per un promotor i un terminador

TODO: El transcriptor cerca els promotors

TODO: No tots els promotors ténen la mateixa probabilitat de sintetitzar-se

TODO: El material transcrit no s'expressa tal qual -¿ Maduració

TODO: Eliminació d'introns, afegits per fer-ho executable

TODO: Codi redundant -¿ Mutacions sense efecte

TODO: Codificació depenent del genoma

TODO: Existeixen mecanismes de regulació activa que afecten a la síntesis per se

TODO: Enzimes correctores

TODO: Algunes mutacions directes no permeses

TODO: Possible codificació de les mutacións bàsiques

TODO: Mutacions a segments no codificants

TODO: No codificadores com a separador de locus pel crossover (3a Mendel)

## 2.5 Genètica de poblacions

## 2.6 Ecologia

# Capítol 3

# Coneixements teòrics sobre vida artificial

## 3.1 Introducció

# Capítol 4

# Coneixements tecnològics

## 4.1 Orientació a objectes

## 4.2 Tecniques de disseny

## 4.3 Llibreria estàndard de C++

# Part III

# Part pràctica

# Capítol 5

# Disseny de l'aplicatiu

## 5.1 Metodologia de disseny

## 5.2 Metodologia d'implementació i estil de programació

### 5.2.1 Registre de canvis

Cada arxiu d'implementació, porta al inici un registre dels canvis (Change Log) que s'anat fent al fitxer. Cada entrada d'aquest registre porta la data, un indicador de l'autor de la modificació i una breu explicació d'una o dos línies, suficient per deduir en què consisteix i a quins llocs afecta.

### 5.2.2 Registre de coses pendents

Per tenir constància de les coses que s'han anat deixant pendents, s'han anat mantenint tres punts de registre de coses a fer (TODO's):

- A peu de codi, és a dir al mateix lloc on cal fer el que quedi pendent. Sempre es tracta d'un comentari que comença amb

  ```
  // TODO:
  ```

  El comentari ha de ser autoexplicatiu per sí mateix, perque no es necessiti veure'l en un

context per entendre'l. D'aquesta forma, es poden extreure totes les modificacions d'aquest tipus que queden pendents executant la comanda:

```
grep -n TODO *.h *.cpp *.c
```

- A l'inici del fitxer d'implementació, a continuació del *Change Log*, es posa els canvis pendents que afectan al mòdul en general que no es puguin localitzar a cap lloc en concret.
- En un fitxer a part anomenat `TODO.txt`, s'han anat recopilant i actualitzant periòdicament els canvis pendents que persisteixen d'entre els anteriors i alguns d'ambit més global o que afecten a mòduls que encara no s'han construït.

El control de les coses pendents resulta molt important per no deixar qüestions deslligades, donada la quantitat de coses que cal tenir presents durant la implementació.

### 5.2.3 Control de versions

Cada cop que es compila, es genera de forma automatitzada una entrada a un log de compilacions amb la data i el número de compilació. Al mateix temps es modifica un arxiu font per tal de que aquest número de compilació i la data estiguin disponibles per al programa.

Això ens permetrà saber, donat un executable, fins a quin punt està actualitzat, i amb els *Change Logs* quines característiques inclou. El registre de compilacions en facilitarà, a més, l'aproximació del temps d'implementació.

### 5.2.4 Fitxers

Tot i que la intenció inicial era mantenir per a cada classe un fitxer de prototipus i un altre d'implementació, l'ús massiu de les classes ha obligat a fusionar algunes classes en el mateix parell de fitxers.

Això sí, només s'ha fussionat en un fitxer classes molt intimament lligades com ara subclasses d'una mateixa classe abstracta factoria en els casos en els que el codi que aportava cada subclasse era molt poc i molt uniforme.

En aquests casos, la classe abstracta factoria té el seu propi fitxer de prototipus de cara a que els seus clients el puguin incloure sense que interfereixi l'existència de les subclasses. La resta s'ha agrupat en un o més.

Generalment el fitxer de la classe abstracta te el nom de la classe en singular i el de les classes derivades en plural. Fent servir un exemple típic, el fitxer `Persona.h` contindria el prototipus de la classe abstracta `CPersona`, i el fitxer `Persones.h` podria contenir les especialitzacions de la classe `CClient` i `CEmpleat` sempre que aquestes classes no afegissin mètodes adicionals al protocol públic de `CPersona`.

### 5.2.5 Criteris de nomenclatura d'identificadors

En molts, casos s'han adoptat alguns criteris que es fan servir en la programació d'Smalltalk.

En general, els identificadors que representen diverses paraules hem adoptat el criteri de fer servir les majúscules per separar-les en comptes del símbol de subratllat com és costum entre alguns programadors de C. Així doncs, farem servir `unIdentificadorLlarg` en comptes de `un_identificador_llarg`.

Els identificadors de les funcions, mètodes de classe (estàtics en nomenclatura C) i objectes globals, els hem començat preferentment per una majúscula. També els noms de les classes i els `namespace`'s.

La primera paraula dels altres identificadors (dades locals o membres, funcions membres no estàtiques...) he adoptat el conveni de començar-la en minúscula.

També he pres alguns convenis estesos en la programació per a windows. Per exemple:

- Preposem una `C` majúscula als identificadors de les classes: `CComunitat`
- Preposem `m_` als identificadors de dades membres no estàtiques: `m_unaVariableMembre`
- Preposem `s_` als identificadors de dades membres estàtiques: `s_unaVariableEstàtica`

### 5.2.6 Proves

Cada classe té una funció membre estàtica anomenada `ProvaClasse` on es deixa tota la bateria de proves unitàries que s'han fet sobre la classe, per, en cas de modificacions, tornar-les a passar.

Els mòduls que no estiguin encapsulats en classes també tindran una funció similar. Generalment per ortogonalitat i per no interferir en l'espai de noms, la funció de proves del mòdul es fica a dins d'un `namespace` sinò hi està ficat ja tot el mòdul.

### 5.2.7 Tipus de comentaris

**Comentaris de mòdul:** Serveixen per explicar qué va al mòdul que encapçalen i si hi ha alguna consideració global que fer en usar-lo o mantenir-lo.

**Comentaris de secció:** Separen visualment les diferents seccions d'un mòdul.

**Comentaris d'encapçalament:** Es troben just després de l'encapçalament d'una funció o mètode i just abans de que s'obrin els claudàtors de l'implementació. Aquests comentaris van adreçats als usuaris de la funció evitant qualsevol menció als detalls d'implementacio. Si cal indicar, precondicions o postcondicions es farà aquí, dedicant una linia a cadascuna que vindrà precedida de les partícules `Pre:` o `Post:`.

**Comentaris de manteniment:** Aquests comentaris es troben al cos de la funció o mètode (entre els claudàtors), parlen de detalls d'implementació i estan adreçats als mantenidors.

## 5.3 Visió global del disseny

### 5.3.1 Disseny modular

L'aplicatiu que es vol dissenyar consta de diversos elements principals, cadascun, amb funcions determinades dintre del sistema.

TODO: Esquemeta del disseny modular global

Figura 5.1: Esquema conceptual del sistema

**Biosistema:** És l'objecte coordinador de la resta d'elements. Les seves funcions són:

1. Multiplexar l'execució concurrent dels diferents organismes.
2. Demanar als organismes les instruccions que volen executar.
3. Realitzar les operacions de modificació i consulta sobre la resta d'elements del sistema, necessàries per executar les instruccions proveïdes per la comunitat d'organismes.
4. Mantenir dintre d'uns mínims la població de la comunitat introduint nous organismes quan aquesta baixa.
5. Accionar els agents externs encarregats de variar el medi al llarg del temps.

**Topologia:** Determina la geometria del medi on viuen els organismes. Les seves funcions són:

1. Associar un identificador a cada posició dins del substrat
2. Establir interconexions entre les parceles de substrat
3. Determinar moviments, direccions, camins... i tota l'operativa que té a veure amb la geometria (topologia) del medi segons aquestes interconexions.

4. Proporcionar l'accés, mitjançant l'identificador de posició, a les propietats del medi en aquesta posició.

**Substrat:** Determina les propietats del medi en una posició donada. Les seves funcions són:

1. Determinar si la posició l'ocupa un organisme i, en cas afirmatiu, quin és l'organisme ocupant.
2. Contenir els nutrients lliures al medi.
3. Altres característiques associades a la localitat que es vulguin afegir més endavant.

**Agents Configuradors:** Determinen l'evolució de certs paràmetres (posició, composició, probabilitat, estacionalitat...) que intervenen en les propietats dels elements del sistema al llarg del temps. Les seves aplicacions són:

1. Afegir o eliminar nutrients lliures dins del medi.
2. Modificar els paràmetres del substrat.
3. Generar expontàneament organismes.

**Comunitat:** Representa al conjunt d'organismes que viuen al biòtop. La comunitat compleix amb les següents funcions.

1. Associar un identificador a cada organisme dintre de la comunitat
2. Afegir-ne o extreure'n organismes.
3. Controlar la informació referent a l'organisme que el relaciona amb el seu entorn, com ara, la posició, el grup reproductiu al que pertany... (Informació externa de l'organisme)
4. Proporcionar l'accés, mitjançant l'identificador d'organisme, tant a la informació externa com al propi organisme.

**Organismes:** Representen als individuus que viuen al biosistema. Contenen la informació genètica i les estructures internes que els fan anar.

1. Oferir instruccions al biosistema del que volen fer.
2. Proporcionar al biosistema accés al fenotip.
3. Proporcionar al biosistema operacions per modificar el seu estat intern.
4. Generar organismes nous.

**Taxonomista:** Reuneix un conjunt d'eines que permeten fer un anàlisis de l'evolució d'un grup reproductiu (població) i de les interaccions amb els altres grups. Aquest seguiment requereix que el taxonomista estigui intimament lligat al funcionament del biosistema.

1. Mantenir informació històrica sobre l'aparició de grups reproductius.
2. Mantenir un cens per edats de la població de cada grup reproductiu i a cada edat.
3. Mantenir un llistat sobre la dieta de cada grup reproductiu.
4. Mantenir un llistat dels agressors de cada grup reproductiu.

## 5.4 Eines i ajudes a la implementació

En aquest apartat s'expliquen algunes eines que s'han implementat per tal d'afavorir l'implementació de la resta del sistema.

### 5.4.1 Dispositius d'entrada i sortida portables

Seguint el paradigma model-vista-controlador el nucli del sistema, el model, hauria de ser independent de l'entorn on executem l'aplicació. Tot i així, a dintre del nucli cal fer algunes operacions d'entrada i sortida, com a mínim per fer les tasques de depurat i els missatges d'error. Per això, ens facilitaria molt les coses que un objecte que tingués un comportament semblant a un `iostream` de C++ però que permeti redireccionar els missatges per gestionar com es visualtitzen depenent de l'entorn destí.

S'ha implementat un objecte `CMissatger` que es comporta de forma molt similar als `iostreams`. Aquest objecte conté una referència a un objecte que pertany a una classe derivada de la clase abstracta `COutputer`. La classe abstracta `COutputer` defineix un protocol molt senzill d'inserció de missatges, per ser controlat per `CMissatger`. Segons la subclasse a que pertanyi l'objecte `COutputer` els missatges insertats es visualitzen d'una forma o d'altra.

Ara mateix estan implementats els seguents `COutputer`'s:

- Consola estàndard
- Control d'edició de MS-Windows
- Caixes de missatges de MS-Windows
- Pop up de la llibreria Curses
- Un scroll limitat de la pantalla fent servir codis ANSI

- Una llista d'strings de la llibreria STL (per imprimir-los en diferit)

En cas de voler una altre dispositiu de sortida, només cal crear el COutputer edient que és una tasca molt senzilla.

### 5.4.2 Funció de compatibilitat de claus

Al sistema resulta molt important una funció que determini, a partir de dos claus, quin els el grau de compatibilitat de les dues.

La compatibilitat entre claus es fara servir, per exemple, per a la identificació d'organismes (amb l'objectiu de cercar preses, col·legues, progenitors...), identificació de nutrients (amb l'objectiu d'ingerir-los, evitar-los, detectar excrecions ajenes, controlar els processos metabòlics interns...) i contesa (mecanismes de depredació i defensa). Donat que aquesta funció és una de les més utilitzades al sistema, ha de ser molt poc costosa.

Cal que la funció no tingui en compte la ponderació dels bits que formen la clau i que els tracti tots de la mateixa forma perque no es converteixi en una optimització numèrica. A mes, és desitjable que aquesta funció permeti nivells de tolerància variables i un cert indeterminisme.

Necessitem tenir en compte llavors tres elements:

- El grau de compatibilitat entre les claus.
- Una tolerància quantificable sobre les variacions entre claus.
- Un element indeterminístic que permeti resultats diferents amb les mateixes entrades.

Si les claus les representem amb dos enters de 32 bits, el grau de coincidència el podrem obtenir fent-ne la o exclusiva bit a bit i complementant el resultat. Al número obtingut l'anomenarem coincidència (C). El nivell de tolerància tambe pot ser un enter (T) que ens vindra donat i l'indeterminisme el pot introduir un altre enter (R) tret d'una funcio pseudo-aleatoria.

**Opció 1**

Els uns del número generat pseudo-aleatoriament (R) es 'filtren' per la coincidència (C) de tal forma ke nomes arribin els uns que estiguin en una posició on no hi havia coincidència entre claus. La tolerància (T) indica el número d'uns que admetem com a màxim per aceptar les claus com a compatibles.

$$ComptaUns(R\& \sim C) < T \tag{5.1}$$

El punt negre d'aquest mètode és l'alt cost de la funció ComptaUns, donat que no és una operacio nativa a la majoria de màquines i cal implementar-la a base de desplaçaments i enmascaraments.

La següent gràfica mostra la probabilitat de que dos claus de 32 bits siguin compatibles segons el bits que tinguin igual i per diferents valors de T. La T pot oscilar entre 0 i 32 tot i que veiem que la distribució no pateix variacions apreciables per valors a partir de 24 o potser abans. Podriem molt be limitar-la entre 0 i 15 sense perdre gaire significat.

TODO: Posar la gràfica compatibilitat1

Figura 5.2: Probabilitat d'encerts segon el nombre d'uns de la coincidència i el nombre d'uns tolerats amb la funció de compatibilitat número 1

La distribució sembla ideal pel que volem: Per valors de poca tolerància, la probabilitat és gairebé nul·la, per toleràncies molt grans ho deixa passar gairebe tot i per a una sèrie de valors intermitjos on es mantenen tres zones:

- Una zona de pas incondicional, per les coincidències més altes.
- Una zona intermitja on la probabilitat de pas depèn de la coincidència.
- Una zona de tall incondicional, per les coincidències més baixes.

### Opció 2

Una altra opció és fer servir la tolerància com una altra màscara. Un bit a un a la tolerància voldria dir que es tolera que aquest bit resulti a un després del filtratge. La condició que determina que dos claus són compatibles quedaria com segueix:

$$(\texttt{R\&} \sim \texttt{C\&} \sim \texttt{T}) == \texttt{0} \tag{5.2}$$

Aquí sí que T agafa tota la franja dels 32 bits. Per fer la gràfica i obtenir un resultat comparable amb l'anterior, s'ha considerat el número de uns a la T en comptes del seu valor.

Figura 5.3: Probabilitat d'encerts segons el nombre d'uns presents a la coincidència i a la tolerància amb la funció de compatibilitat número 2

Observem que hem perdut les zones d'aceptació i rebuig incondicional a les toleràncies intermitges, però, la funció és bastant vàlida pels objectius donat que és una aceptació no deterministica on la probabilitat depèn de la tolerància i de la coincidència, i, a més, hem optimitzat moltissim el cost d'evaluació.

Com a característica afegida, aquesta funció permet, mitjançant la tolerància, un control més acurat de quins bits son els que poden no coincidir. Aquesta peculiaritat pot donar a peu a mecanismes més complexes, que no pas una tolerància cega. A mès, tot i que es té en compte la posició dels bits, no els pondera, com les altres formules provades.

### Altres opcions desestimades

Altres funcions de compatibilitat han estat provades i del tot desestimades pel seu alt cost i/o per la seva poca idoneitat.

Per exemple, es va provar la funció

$$ComptaUns(R\& \sim C\& \sim T1) < (T2\&0x5) \tag{5.3}$$

per sintetitzar en una fòrmula els dos conceptes de tolerància que hem vist, una tolerància que dòna significat a la posició dels uns i una altra que permet tolerar globalment un cert nombre d'uns independentment de la posició.

Degut als pocs bits (32) amb els que juga i a que hi havia dos punts on es tolera, la funció, lluny de donar tot el significat que voliem, dona molt poca variació amb els paràmetres. A més, tornem a tenir el problema de la funció ComptaUns.

$$R >> (T\&0x7) < (C << ((T >> 3)\&0xf)) \tag{5.4}$$

### 5.4.3 Seqüències d'escapament ANSI

De cara a obtenir una sortida rica, però, conservar el caracter portable de l'aplicació, s'ha optat per fer servir seqüències d'escapament ANSI en un terminal de texte. Aquestes seqüències permeten fixar colors, posicionar el cursor, netejar la pantalla... i d'altres operacions en terminals que compleixin aquest estàndard. Això inclou els terminals de Linux i les consoles de MS-DOS i MS-Windows després de instalar el controlador `ANSI.SYS`.

S'ha optat per implementar una biblioteca pròpia per ajudar a insertar aquests codis donat que totes les biblioteques que provades que treballaven amb aquestes seqüències, no eren prou optimes com per l'ús masiu que es fa d'elles al projecte. A més, cap de les provades feia un enfoc compatible amb els `iostream`'s de C++.

Els símbols que defineix la biblioteca de codis ANSI es poden insertar sense problemes en un `iostream` qualsevol (fins i tot a un fitxer obert), o en la classe `CMissatger` implementada.

Generalment la inserció optimitzada simplement inserta una cadena predefinida, o, si hi ha paràmetres, substitueix els caracters que varien d'un a l'altre a una còpia de la cadena original. Es podria optimitzar un xic més del que està no copiant la cadena, però, seria a costa de comprometre la integritat del sistema, si algun dia, l'aplicació ha d'executar-se en multithread.

La biblioteca també implementa la classe `CColor` que implementa operacions amb atributs de color pels caracters. Els objectes CColor en ser inserits en un `stream` inserta la seqüència d'escapament corresponent.

En resum, la biblioteca permet expressions com aquesta:

```
cout
     << clrscr << gotoxy(3,4) << blau.fons(vermell)
     << "Hola Món" << blau.brillant() << "!!!" << endl;
```

que es resolen de forma bastant optima comparant-ho amb altres llibreries de l'estil.

A mode d'exemple posem com hem solucionat cada tipus de seqüència:

```
// Sequencia constant
const char clrscr[] = "\033[2J";



// Seqüència amb parametres de longitud fixa
string color (int fg, int bg)
{
  // Aquesta es la copia necesaria per permetre multithreading
  string ansiseq ("\033[0;30;40m");

  ansiseq[2]=(fg&0x08)?'1':'0';
```

```
  ansiseq[5]='0'+(fg&0x07);
  ansiseq[8]='0'+(bg&0x07);
  return ansiseq;
}

// Seqüència amb parametres de longitud variable
string gotoxy(int col, int lin)
{
  ostrstream str(myBuffer,16);
  str << "\033[" << lin << ';' << col << 'H' << ends;
  return str.str();
}
```

## 5.5 Estructura del medi

### 5.5.1 Visió general

El present apartat descriu el funcionament del medi on viuran els organismes i alguns detalls de disseny i d'implementació. De cara a permetre ampliar fàcilment el model, hem volgut fer un disseny del medi que permeti adaptacions a futures necessitats de modelatge. Per un costat es descriu el model genèric i, per un altre, es descriuen les particularitzacions que s'han implementat per a aquest projecte.

La generalitat del model s'ha volgut limitar al conjunt de biòtops discrets. Això implica dues coses:

- Les posicions dins del biòtop estan quantitzades. No hi ha més que un nombre limitat de posicions a diferència de l'espai continu real.
- Tot i que podem considerar una posició discreta com a representant d'una zona limitada del substrat, les propietats dins d'aquesta zona del substrat són uniformes.

Aquesta limitació juntament amb la discretització del temps és comuna a la majoria de sistemes artificials. L'única forma de limitar aquest efecte és fer que el pas de quantització sigui tan petit que el seu efecte quedi minimitzat.

El nostre model general és un model que consta de posicions discretes amb les seves corresponents propietats i que tènen relacions de veinatge amb les altres posicions segons una topologia.

Així doncs, tenim dos elements que podem modelar independentment.

- **Posicions del substrat:** Elements discrets que indiquen les propietats d'una zona del biòtop.

- **Topologia:** Controla les relacions de veinatge i la identificació de les posicions per part de la resta del sistema.

Combinant aquests dos elements, es pot obtindre un conjunt molt ric de biòtops.

### 5.5.2 Topologies

La topologia determina les relacions de veinatge entre les cel·les. Si les posicions del biòtop fòssin nodes d'un graf, la topologia representaria els vertexs que els uneixen.

Per exemple, podem adaptar la topologia per convertir-la en una topologia 2D, molt vàlida per simular biosistemes terrestres sense estratificar, o podem adaptar-ho a una topologia 3D que és més realista per simular medis fluids, com l'aigua o estratificats com els boscos.

Com que el nombre de cel·les és limitat, el conjunt de cel·les formaran una regió limitada. La topologia ha de determinar quines són les veïnes de les cel·les de les vores. Com a exemple considerem una topologia 2D rectangular. Si féssim que les cel·les limítrofes no tinguin veïnes més enllà dels límits ens trobarem davant d'una regió limitada. Si fem que les cel·les d'un dels costats es conectin amb les cel·les del costat oposat, obtindrem una topologia de superfície cilíndrica. Si fem el mateix amb tots quatre costats obtindrem una topologia de superfície toroidal.

Per a aquest projecte, s'ha implementat una topologia toroidal perque, en principi es vol controlar la complexitat del sistema i, un biòtop amb topologia limitada n'afegiria donat que introdueix situacions a les que hauria de fer front l'organisme, per exemple, quan està en un límit i quan no hi està. En una topologia toroidal no hi ha vores i per tant els organismes no s'hi han d'enfrontar a aquest element de complexitat.

Per facilitar la implementació, hem fet que la veina directa d'una posició a l'estrem dret sigui una posició a l'estrem esquerre però, no pas la que està a la mateixa línia sinó la que està una línia abaix. L'única diferència que introdueix això és que anant sempre a la dreta o a la esquerra,

sense variar la direcció podem recòrrer tot el biòtop.

Cada posició té 8 cel·les veines directes. Un desplaçament a qualsevol d'aquestes 8 cel·les es pot codificar amb 3 bits com indica la figura 5.1

| | | |
|---|---|---|
| 100 | 000 | 001 |
| 101 | Origen | 010 |
| 110 | 111 | 011 |

Taula 5.1: Veïnes directes d'una posició

La concatenació de N desplaçaments bàsics aleatoris tendeix a formar una distribució normal entorn al centre. Com els vectors de desplaçament tenen 32 bits podriem codificar fins a 10 desplaçaments bàsics consecutius en un vector de desplaçament. Però, com ens serà molt útil poder activar i desactivar cada desplaçament bàsic, farem servir un quart bit per a cada bàsic per dir si està habilitat o inhibit el desplaçament, i en un vector hi caben, doncs, 8 desplaçaments bàsics.

| h0 | d0 | h1 | d1 | h2 | d2 | h3 | d3 | h4 | d4 | h5 | d5 | h6 | d6 | h7 | d7 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 1 | 101 | 1 | 101 | 1 | 010 | 1 | 110 | 1 | 110 | 1 | 110 | 1 | 110 | 1 | 110 |

Taula 5.2: Codificació dels desplaçaments al biòtop

Si cal considerar cap ordre en el càlcul dels desplaçaments bàsics, es fa de més significatiu a menys.

La codificació dels desplaçaments bàsics s'ha fet de tal manera que, si invertim bit a bit un vector de desplaçament a excepció dels bits d'habilitació, obtenim un desplaçament invers. És a dir, donada la direcció *desp*, (desp XOR 0x77777777) en dòna la direcció inversa, la qual cosa serà molt útil de cara a afavorir l'aparició de conductes d'evasió.

L'altre funció important de la topologia és assignar a cada cel·la un identificador únic dins de la topologia. La resta del sistema farà servir aquest identificador per referenciar una posició i, en cas de necessitar-ho, demanarà a la topologia quin és el substrat per aquesta posició.

En la present implementació s'han fet servir enters del 0 a N-1 com a identificadors de les posicions, on N es el nombre de cel.les. Aquests identificadors ens permeten fer els desplaçaments de forma molt optima si assignem el números per ordre a les cel·les de cada fila. Només cal afegir (o treure), a l'identificador de la posició origen, un número, que depèn del desplaçament, i ajustar el resultat en cas de sortir-se de límits, per calcular l'identificador de la posició destí.

En resum, una topologia ofereix els següents serveis als seus clients:

- Donar l'identificador de la posició destí a partir de l'identificador d'una posició origen i d'un desplaçament.
- Donar l'identificador d'una posició escollida aleatoriament.
- Determinar si un identificador donat és vàlid dintre de la topologia.
- Accedir al subtrat corresponent a un identificador de posició.
- Donar el desplaçament que apropa una posició d'origen donada a una posició destí per l'itinerari més curt.

### 5.5.3 Substrats

Un substrat particularitza les propietats del medi a una posició. El substrat pot tenir diverses propietats segons la complexitat que desitjem per al medi.

Les dos propietats més importants del substrat són qui ocupa el substrat, i quins nutrients hi han.

#### Ocupants

S'ha restringit l'ocupació de les posicions per part dels organismes a un sól organisme per posició i a una sola posició per organisme. La raó ha estat fer possible referenciar un organisme per

la seva posició. Això simplificarà, a les relacions entre organismes, com referir l'altre organisme. També, com a avantatge adicional, simplifica la interfície amb l'usuari per seleccionar un organisme seleccionant la seva posició.

### Nutrients

En quant els nutrients que hi pot haver en una mateixa posició del substrat, cada posició tè un nombre de nutrients màxim definit. Quan s'afegeixen nutrients per damunt d'aquest nombre, els nutrients més antics desapareixen.

Els nutrients estan diferenciats qualitativament amb un sencer que indica el seu tipus.

La recollida de nutrients, es fa amb un patró de cerca pel tipus de nutrient i una tolerància a nivell de bit segons la funció de compatibilitat estàndard. Com s'explica a l'apartat 5.4.2, aquesta funció retorna cert si

```
((Patro^Clau)&Random&~Tolerancia)==0
```

, de tal forma que un 1 a una posició de la tolerància significa que, encara que no es correspongui aquest bit no es tindrà en compte. La cerca es fa des dels més nous fins els més vells.

### Movilitat

## 5.6 Agents externs

### 5.6.1 Trets generals

Els agents externs són objectes que disparen una acció determinada quan són cridats. La majoria d'accions es produeixen sobre el biòtop, sobre l'estat d'altres agents externs o sobre paràmetres globals de configuració.

El paper dels agents externs a dins del sistema és permetre a l'usuari controlar com evolucionarà l'entorn on es mouran els organismes de cara a obtindre resultats que s'hi puguin contrastar. Per sí mateix, el conjunt d'agents implementats permet configuracions molt complexes, però, a més, ofereix un seguit d'eines molt útils per que l'usuari-programador pugui ampliar aquests agents.

Quan un agent es cridat per realitzar la seva acció, es diu que l'agent ha sigut **accionat**. Quan algú requereix un valor contingut en l'estat de l'agent es diu que l'agent ha sigut **consultat**.

Direm que un agent A té com a **subordinat** a un altre agent B si és A qui acciona a B. Ho representarem així: $A \rightarrow B$. L'estructura de subordinació ha de ser un arbre on els subordinats són els fills d'aquell a qui es subordinen.

Direm que un agent A és **depenent** d'un altre agent B si A, quan és accionat, necessita consultar l'estat de l'agent B. Ho representarem així: A(B). La consulta no ha d'implicar cap modificació ni recàlcul d'estat en l'agent consultat. Això permet que no hi hagi restriccions en l'estructura de dependència i que puguin existir dependències creuades. [1]

Tots els agents duen un nom assocciat que, per defecte, coincideix amb un prefixe i un número de sèrie únic entre tots els agents. Aquest nom es pot canviar per un que sigui més mnemotècnic per a l'usuari.

[1] Tot i així, és important preveure que l'ordre d'accionat entre agents interdependents podria implicar variacions en els resultats.

TODO: El tema dels logs

### 5.6.2 Agents Subordinadors

Els agents subordinadors són agents que, quan són accionats, accionen tot un seguit d'agents subordinats.

#### Agents Múltiples

L'agent múltiple acciona una i només una vegada cadascun dels agents subordinats cada cop que és accionat.

#### Agents Temporitzadors

Els agents temporitzadors són agents multiples que no sempre que reben un accionat el propaguen cap els subordinats. Estableixen dos períodes, un actiu i un altre inactiu. Els accionats només es propagen als subordinats durant el període actiu.

Els períodes es defineixen mitjançant tres paràmetres: El període mínim és el número d'accionats que dura el període com a mínim. Aquest mínim es pot augmentar de forma no determinística el resultat de sumar-li n vegades un número aleatori en l'interval [0,m] (els corxets indiquen que els extrems estan inclosos). $n$ és el número de daus, i $m$ és el valor màxim o magnitud del dau.

D'aquesta especificació es pot deduir algunes dades, pot ser, més intuitives per a l'usuari:

- El valor màxim que pot adoptar el període és el mínim més n*m
- Un sól dau equival a una distribució uniforme entre els límits
- A mesura que incrementem el nombre de daus, la distribució dels períodes s'aproxima a una distribució normal entorn al centre entre el valor màxim i mínim, amb un desviació típica

cada vegada menor.

Per defecte, els paràmetres que introdueixen indeterminisme en els temporitzadors, com són els daus, estan ajustats de manera que el seu efecte sigui nul. Si no es toca res més que els períodes mínims, actuarà de forma determinista. De la mateixa manera, els períodes mínims dels cicles estan ajustats, per defecte, per que sempre s'estigui en un cicle actiu. D'aquesta forma, si no es configura res, l'efecte d'un temporitzador és el d'un agent múltiple ordinari.

Paràmetres per defecte i una execució:

- Cicle Actiu ( minim=1 daus=0 magnitud=0)
- Cicle Inactiu ( minim=0 daus=0 magnitud=0)
- Periode Actual ( actiu )
- Periode Restant (1)

Paràmetres ilustratius:

- Cicle Actiu ( minim=0 daus=2 magnitud=3)
- Cicle Inactiu ( minim=4 daus=0 magnitud=0)
- Periode Actual ( actiu )
- Periode Restant (1)

A continuació hi ha una execució dels paràmetres anteriors. Els guions representen accionats durant el període inactiu i les O's representen accionats durant el període actiu.

```
OO----OOOOO----OO----O----OOOO----O----OOO----OOOO----OOO----OOOO----O----OO----
OOO----OOOO--------OOO----O--------OOO----OOOO----O----OOOO----OOOO--------OOOO-
---OOO----OOO----OOO----OOOOO----O----OOOOOO----OOOO----O----OOOO----OOO----OO--
--OO----OOOO----OOO----OOO----OOOOO----OOO----OOOO----OOO----OO----OO----OO----O
OOOOO----OOO----OOOOO----OOOO----OO----OOOOO----OOOO----OOO----OOO----OOOOO----O
```

### Agents Probabilitzadors

Els agents probabilitzadors són també agents múltiples que controlen si l'accionat es propaga cap els subordinats o no. Però, a diferència dels agents temporitzadors, ho fan mitjançant una llei probabilística. Si es dòna la probabilitat, s'accionen els subordinats, si no es dòna, no s'accionen.

La probabilitat es defineix amb el nombre de vegades que es donaria la probabilitat en un tamany de mostra. Per exemple, podem definir una probabilitat dient que es dòna 3 de cada 14 vegades. 14 és el tamany de mostra i 3 les vegades que es donaria en la mostra.

Els paràmetres estan ajustats per defecte a valors que fan del probabilitzador un agent múltiple ordinari.

Paràmetres ilustratius:

- Probabilitat ( mostra=40 encerts=25)

A continuació hi ha una execució dels paràmetres anteriors. Els guions representen accionats en els quals no s'ha donat la probabilitat i les O's, accionats en els quals sí s'ha donat.

```
OOOO-O-O-O-OO----OOOO-OOOOOOOO-OOO-OOOOOOOOOOO-OOOOOOOOO---O-OOOOOOO-O--OOOO-OO-
OOO-OOO-O-OOOO-OOO-O-OOO-OOOOOOOOOOO-OOOOOOO-OOOOOO--OOO-OOO-OO-O-OOOOO--O---O-
OO--O---OOO-OOOOOOOOO-O-OOO-O-OOO-O--OOOOOOOO-OO-O-OO-OOO-OO-OOOO-OOOOOO-OOO-OOO
--OOOO-OOOO-O-OOO-OO-OO-O--OO-OOOO--O--O-O-OO--OO-O-O-OOO---O-O-OO------OO-OO-O-
OOO-OO-O-OO-OO-OOOOO----OO-OO-----O-OOOOO--O---OOOOOOOOOO-O-OO-O-OOOOO---OOO-OO-
```

### Agents Iteradors

Els agents iteradors són agents múltiples que no limiten els accionats que arriben als seus subordinats, sinó que el que fan és multiplicar els accionats que li arriben.

Més concretament, quan un agent iterador és accionat, els seus subordinats, són accionats un número de vegades que es calcula a partir d'un mínim i uns daus com els que feiem servir per als períodes dels temporitzadors.

Per defecte, la part indeterminística (els daus) no té cap efecte, i la part deterministica (el mínim) està posada a un valor (1) que el fa equivalent a un agent múltiple ordinari.

Paràmetres ilustratius:

- Iteracions ( minim=2 daus=2 magnitud=4)

A continuació hi ha una execució dels paràmetres anteriors. Els parèntesis agrupen els accionaments dels subordinats que es fan sota un mateix accionament de l'iterador.

```
(OOOOOO)(OOOOOOOO)(OOOOOOO)(OOOOO)(OOO)(OOOOOOO)(OO)(OOOOOOO)(OOOOOOO)(OOOOOO)(O
OOOOOOO)(OOOOO)(OOOOOO)(OOO)(OOOOOO)(OOOO)(OOOOOO)(OOOOOO)(OOO)(OOOOOO)
```

Amb dos accions subordinades una execució quedaria com això:

```
(OEOEOEOEOEOEOEOEOEOE)(OEOEOEOEOEOEOEOEOE)(OEOEOEOEOEOEOE)(OEOEOEOE)(OEOEOEOEOEO
EOEOE)(OEOEOEOEOEOEOEOE)(OEOEOEOEOEOEOEOEOE)(OEOE)(OEOEOEOEOEOEOE)(OEOEOEOEOEO
EOE)(OEOEOEOEOEOE)(OEOEOEOEOEOE)(OEOEOEOEOE)(OEOEOEOEOEOEOE)(OEOEOEOEOEOE)(OEOEO
EOEOEOE)(OEOEOEOEOEOE)(OEOEOEOEOE)(OEOEOEOEOE)(OEOEOEOEOE)
```

on les E's representen l'execució de la segona acció subordinada.

### 5.6.3 Agents Posicionadors

Els agents posicionadors controlen una posició en la topologia del biòtop. No tènen subordinats, però, generalment hi ha agents que en depenen del seu valor i segons el tipus de posicionador per recalcular la seva posició fan servir altres agents dels quals depenen.

**Posicionador Bàsic:** No modifica la seva posició si és accionat. A menys que, per configuració, es fixi a una posició concreta, s'inicialitza amb una posició aleatoria vàlida dins de la topologia,

**Posicionador Aleatori:** Cada cop que és accionat pren una posició aleatoria vàlida dintre de la topologia.

**Posicionador Zonal:** Cada cop que és accionat pren una posició aleatoria dintre d'una zona. La zona es defineix per una posició central, determinada per un altre posicionador de qual depèn, i un radi, que no és més que el nombre de desplaçaments aleatoris que es fan a partir d'aquesta posició central per trobar la posició final. Les posicions tendeixen a adoptar una distribució normal en l'entorn de la posició central.

**Posicionador Seqüencial:** Cada cop que és accionat pren la posició del següent posicionador que hi ha en una seqüència de posicionadors. Els agents posicionadors de la seqüència són dependència del posicionador seqüencial.

**Posicionador Direccional (Itinerari):** Cada cop que és accionat pren la posició que en resulta d'aplicar-li un desplaçament a la posició anterior. El desplaçament el determina un agent direccionador que és dependència. Els agents direccionadors s'expliquen al següent apartat.

A continuació es presenten exemples d'execució d'un posicionador aleatori, un de seqüencial i un de zonal aplicats sobre una topologia toroidal.

TODO: Exemples d'execució posicionadors seqüencial, zonal i aleatori

Figura 5.4: Exemples de posicionadors: seqüencial, zonal i aleatori

### 5.6.4 Agents Direccionadors

Els agents direccionadors controlen una direcció per calcular desplaçaments dintre de la topologia del biòtop. La seva utilitat principal radica en controlar la posició d'un posicionador de tipus direccional, però, no es descarten altres aplicacions futures.

**Direccionador Bàsic:** No modifica la seva direcció si és accionat.

**Direccionador Aleatori:** Cada cop que és accionat pren una direcció aleatoria.

**Direccionador Seqüencial:** Cada cop que és accionat pren la direcció del següent direccionador que hi ha en una seqüència de direccionadors. Els agents direccionadors de la seqüència són dependència del direccionador seqüencial.

A continuació es presenten exemples d'execució d'un posicionador direccional, que depèn de diferents tipus de direccionadors.

TODO: Exemple d'execució d'un itinerari amb els diferents direccionadors

Figura 5.5: Exemples de direccionadors (seqüencial, fixe i aleatori) controlant un posicionador direccional

Per obtindre aquests resultats ha calgut fer servir subordinadors per que el posicionador s'accionés més que el direccionador. La topologia de l'exemple és toroidal.

### 5.6.5 Agents Actuadors

Els agents actuadors són els que finalment modifiquen el substrat. Els actuadors depenen d'un agent posicionador que els indica la cel·la que han de modificar.

La majoria dels agents que hem vist fins ara eren molt independents davant de les modificacions en la topologia i en la composició del substrat que es puguin fer més endavant. Les especialitzacions dels actuadors, en canvi, han de dependre per força del substrat i la seva composició, perqué actuen sobre ell. Segueixen sent independents, però, de la topologia.

Aquí a sota, expliquem alguns actuadors vàlids pel substrat implementat en aquest treball.

#### Agents Nutridors

Aquests actuadors depositen nutrients al substrat. Un tipus de nutrient es codifica amb un enter de 32 bits sense signe. El tipus de nutrient que es depositarà s'especifica amb dos nombres enters

de 32 bits sense signe. El primer enter indica el número del tipus bàsic, i el segon indica els bits del tipus bàsic que poden variar aleatoriament.

Per exemple, el parell $\begin{matrix}\text{element bàsic:} & \text{0x0000000000000000} \\ \text{variabilitat:} & \text{0xFFFFFFFF00000000}\end{matrix}$ genera elements que tenen la part baixa igual que l'element bàsic (a zero) i la part alta al atzar.

#### Agents Desnutridors

Els desnutridors són molt semblants als nutridors, però, en comptes de afegir nutrients, en treuen. Es pot treure selectivament cercant un element químic que s'apropi al que s'indica o es pot especificar una tolerància per a certs bits.

### 5.6.6 Arxius de configuració d'agents

#### Motivació i criteris de disseny

De cara a poder passivitzar un biosistema a disc per poder-ho restaurar posteriorment, caldria també poder passivitzar i restaurar l'estat dels seus agents. L'arxiu de configuració d'agents és un arxiu de text que conté l'estat i l'estructura dels agents d'un biosistema, que es pot extreure en un moment donat i restaurar-ho posteriorment.

Els agents a un biosistema, com s'ha dit abans, formen una estructura d'arbre segons les seves relacions de subordinació. Cada arxiu de configuració conté un arbre d'agents subordinats partint d'un agent arrel. Seria possible penjar tota l'estructura d'un arxiu de configuració i subordinar-ho a un agent d'una estructura ja existent a un biosistema. Es podria formar una mena de biblioteca d'arxius amb configuracions comunes que es podrien convinar per muntar ràpidament l'estructura d'agents d'un biosistema.

**Estructura**

Els arxius de configuració d'agents ténen una estructura molt simple: Primer van unes linies de text que determinen, per cada agent, el seu nom i el seu tipus. Un cop definits els noms i els tipus, es configuren els paràmetres per a cada agent.

La definició dels noms i els tipus es fa amb una línia per cada agent sent la primera línia la que defineix l'agent que està en l'arrel de la estructura de subordinació. A cada línia es posa, per ordre i *separats per espais* un signe asterisc, el nom i el tipus.

Quan es carrega un arxiu de configuració d'agents, si és possible a cada agent se li dòna el nom amb el que apareix a l'arxiu, però això no és posible si ja existeix un amb el mateix nom. En aquest cas, se li dòna un nom per defecte i s'hi tradueixen totes les posteriors referències al nom antic.

Els noms poden contenir qualsevol caracter que no es consideri un espai a C (espais, tabuladors, retorns...).

El tipus s'especifica amb un identificador propi de cada tipus d'agent. Dins d'un arxiu de configuració d'agents, es reconeixen els següents tipus:

Es fan servir identificadors jerarquics molt semblants als que es fan servir a UNIX per identificar els directoris. La jerarquia de noms el que especifica aquí és una jerarquia de tipus i subtipus de tal forma que si, per exemple, a un lloc es requereix *Agent/Posicionador*, aquest lloc el pot ocupar tant un *Agent/Posicionador/Direccional* com un *Agent/Posicionador/Zonal*.

Una definició de noms i tipus podria quedar com segueix:

```
* Agent_0000 Agent/Multiple
* Agent_0001 Agent/Multiple/Temporitzador
* Agent_0002 Agent/Direccionador/Aleatori
```

| **Nom de tipus a la memòria** | **Nom del tipus a un fitxer de configuració** |
|---|---|
| Agent Subordinador Multiple | Agent/Multiple |
| Agent Subordinador Temporitzador | Agent/Multiple/Temporitzador |
| Agent Subordinador Iterador | Agent/Multiple/Iterador |
| Agent Subordinador Aleaturitzador | Agent/Multiple/Aleaturitzador |
| Posicionador Fixe | Agent/Posicionador |
| Posicionador Aleatori | Agent/Posicionador/Aleatori |
| Posicionador Zonal | Agent/Posicionador/Zonal |
| Posicionador Direccional (Itinerari) | Agent/Posicionador/Direccional |
| Direccionador Fixe | Agent/Direccionador |
| Direccionador Aleatori | Agent/Direccionador/Aleatori |
| Actuador Nutridor | Agent/Actuador/Nutridor |
| Actuador Desnutridor | Agent/Actuador/Nutridor/Invers |

Taula 5.3: Tipus d'agents implementats al projecte i identificadors associats

```
* Agent_0003 Agent/Posicionador/Direccional
* Agent_0004 Agent/Multiple/Iterador
* Agent_0005 Agent/Actuador/Nutridor
```

Aquí, *Agent_0000* seria l'agent arrel.

Un cop definits els noms i els tipus dels agents, cal configurar els seus paràmetres. Per configurar un agent primer cal posar una línia amb el signe + i el nom de l'agent separats per un espai, i, després, tot un seguit de línies de configuració de paràmetres. Les linies de configuració de paràmetres comencen amb un signe menys i el nom del paràmetre i es segueix amb els valors que necessita el paràmetre per configurar-se, tot separat per espais. Com veurem al següent exemple, és normal que un paràmetre s'especifiqui amb diversos valors separats per espais. La posició dels valors acostuma a ser significativa o sigui que és important mantenir l'ordre.

Per configurar un Nutridor es faria de la següent forma

```
+ Agent_0004
- Posicionador Agent_0003
```

```
- Composicio 31 0
```

Quan un paràmetre necessita com a valor un altre agent, fa servir els seu nom com a referència.

Al següent apartat, es detalla els paràmetres que controlen cada tipus d'agent.

### 5.6.7 Paràmetres configurables per a cada tipus d'agent

El que segueix és una especificació de com es configuren els paràmetres dels tipus d'agent implementats mitjançant el fitxer de configuració. Per fer-ho fem servir la següent estructura:

Per a cada paràmetre de cada tipus d'agent es fa una petita explicació i es detallen, ordenats tal qual han d'aparèixer, els valors que el defineixen.

Els valors dels paràmetres es detallen posant un tipus de dada, dos punts, una petita explicació del valor i, entre parèntesis, les restriccions que s'hi apliquen.

Als agents implementats, els tipus de dada possibles pels valors són:

- **agent:** Nom d'un agent especificat a la definició de noms i tipus
- **uint32:** Senser sense signe codificable en 32 bits i expressat en base decimal
- **id(Alternativa1/Alternativa2...):** Un dels identificadors posats com a alternativa

Després dels paràmetres de cada tipus hi ha un exemple de com quedarien les línies de configuració.

#### MultiAgent (Agent/Multiple)

**Accio:** Determina un agent subordinat. Es repeteix tantes vegades com subordinats tingui.

- agent: agent que es subordina (No ha de ser subordinat de cap altre)

Exemple de configuració texte

```
+ AgentMultiple1:
- Accio Posicionador1
- Accio Posicionador2
- Accio Direccionador1
```

**Temporitzador (Agent/Multiple/Temporitzador)**

**Accio:** Determina un agent subordinat. Es repeteix tantes vegades com subordinats tingui.

- agent: agent que es subordina (No ha de ser subordinat de cap altre)

**CicleActiu:** Determina quan triguen els períodes de temps actius

- uint32: període mínim
- uint32: número de daus
- uint32: magnitud dels daus (Van de zero a la magnitud)

**CicleInactiu:** Determina quan triguen els períodes de temps inactius

- uint32: període mínim
- uint32: número de daus
- uint32: magnitud dels daus (Van de zero a la magnitud)

**AntiAccio:** Agent subordinat especial que s'acciona en el cicle inactiu (Nomes un per temporitzador i es opcional)

- agent: agent que es subordina (No ha de ser subordinat de cap altre)

**CicleActual:** Valors del temporitzador quan es reemprengui la marxa

- id(Actiu/Inactiu): cicle actiu o inactiu
- uint32: període restant del cicle actual

Exemple de configuració texte

```
+ Temporitzador1
- Accio Posicionador3
- CicleActiu 34 2 5
- CicleInactiu 2 4 4
- CicleActual 3 Inactiu
```

**Iterador (Agent/Multiple/Iterador)**

**Accio:** Determina un agent subordinat. Es repeteix tantes vegades com subordinats tingui.

- agent: agent que es subordina (No ha de ser subordinat de cap altre)

**Iteracions:** Determina quantes vegades es repiteixen els subordinats

- uint32: iteracions minimes
- uint32: número de daus
- uint32: magnitud dels daus (Van de zero a la magnitud)

**PreAccio:** Agent subordinat especial que s'executa un sol cop abans de tot (Nomes un per iterador i es opcional)

- agent: agent que es subordina (No ha de ser subordinat de cap altre)

**PostAccio:** Agent subordinat especial que s'executa un sol cop despres de tot (Nomes un per iterador i es opcional)

- agent: agent que es subordina (No ha de ser subordinat de cap altre)

Exemple de configuració

```
+ Iterador3
- Accio Posicionador4
- Accio Actuador2
- Iteracions 20 3 6
```

**Aleaturitzador (Agent/Multiple/Aleaturitzador)**

**Accio:** Determina un agent subordinat. Es repeteix tantes vegades com subordinats tingui.

- agent: agent que es subordina (No ha de ser subordinat de cap altre)

**Probabilitat:** La que hi ha d'accionar els subordinats

- uint32: número d'encerts que segons la probabilitat tenderien a donar-se en la mostra
- uint32: número de intents o mostra

**ReAccio:** Agent subordinat especial que s'acciona si no es dona la probabilitat (Nomes un per temporitzador i es opcional)

- agent: agent que es subordina (No ha de ser subordinat de cap altre)

Exemple de configuració

```
+ Aleaturitzador1
- Accio Posicionador3
- Probabilitat 20 100
```

**Posicionador Fixe (Agent/Posicionador)**

**Posicio:** Posició inicial

- uint32: valor de la posició (Ha d'existir a la topologia)

Exemple de configuració

```
+ Posicionador1
- Posicio 12
```

**Posicionador Aleatori (Agent/Posicionador/Aleatori)**

**Posicio:** Posició inicial

- uint32: valor de la posició (Ha d'existir a la topologia)

Exemple de configuració

```
+ Posicionador2
- Posicio 23
```

**PosicionadorSequencial (Agent/Posicionador/Sequencial)**

**Posicio:** Posició inicial

- uint32: valor de la posició (Ha d'existir a la topologia)

**Sequencia** : Determina una posició de la seqüència. Es repeteix tantes vegades com calgui.

- uint32: valor de la posició (Ha d'existir a la topologia)

**SequenciaActual** :

- uint32: el número de seqüència de la següent posició (Si es passa es pren l'ultim)

Exemple de configuració

```
+ Posicionador3
- Posicio 23
- Sequencia 27
- Sequencia 50
- Sequencia 402
- SequenciaActual 2
```

**PosicionadorZonal (Agent/Posicionador/Zonal)**

**Posicio:** Posicio inicial

- uint32: valor de la posició (Ha d'existir a la topologia)

**Posicionador:** Dona la posició central de la zona

- agent: agent posicionador (dependència)

**Radi:** Nombre de desplacaments que pot fer la posició entorn al centre

- uint32: valor del radi

Exemple de configuració

```
+ Posicionador4
- Posicio 23
- Posicionador Posicionador1
- Radi 3
```

**Itinerari (Agent/Posicionador/Direccional)**

**Posicio:** Posició inicial

- uint32: valor de la posició (Ha d'existir a la topologia)

**Direccionador:** Dona la direcció del desplacament

- agent: agent direccionador (dependència)

**Radi:** Nombre de desplacaments que pot fer la posició respecte a la posició anterior

- uint32: valor del radi

Exemple de configuració

```
+ Posicionador5
- Posicio 23
- Direccionador Direccionador3
- Radi 1
```

**Direccionador (Agent/Direccionador)**

**Direccio:** Direcció inicial

- uint32: valor de la direcció

Exemple de configuració

```
+ Direccionador1
- Direccio 876342
```

**DireccionadorAleatori (Agent/Direccionador/Aleatori)**

**Direccio:** Direcció inicial

- uint32: valor de la direcció

Exemple de configuració

```
+ Direccionador2
- Direccio 23442684
```

**DireccionadorSequencial (Agent/Direccionador/Sequencial)**

**Direccio:** Direcció inicial

- uint32: valor de la direcció

**Sequencia:** Determina una direcció de la seqüència. Es repeteix tantes vegades com calgui.

- uint32: valor de la direcció

**SequenciaActual:** Determina el punt actual de la seqüència

- uint32: el número de seqüència de la següent direcció (Si es passa es pren l'ultim)

Exemple de configuració

```
+ Direccionador3
- Direccio 23
- Sequencia 27
- Sequencia 50
- Sequencia 402
- SequenciaActual 2
```

**Nutridor (Agent/Actuador/Nutridor)**

**Posicionador:** Dona la posició on s'actua

- agent: Agent posicionador (dependència)

**Composicio:** Determina els elements que es depositen

- uint32: element basic
- uint32: variabilitat, a 1 els bits que poden variar

Exemple de configuració

```
+ Actuador1
- Posicionador Posicionador4
- Composicio 13152450903 0
```

**Desnutridor (Agent/Actuador/Nutridor/Invers)**

**Posicionador:** Dona la posició on s'actua

- agent: Agent posicionador (dependència)

**Composicio:** Determina els elements que s'eliminen

- uint32: element basic
- uint32: tolerancia, a 1 els bits que no importa que coincideixin

Exemple de configuració

```
+ Actuador2
- Posicionador Posicionador3
- Composicio 8943742645 768764258
```

### 5.6.8 Exemple complert d'arxiu de configuració d'agents

A continuació es presenta un exemple complert:

```
* Agent_0000 Agent/Multiple
* Agent_0002 Agent/Posicionador/Direccional
* Agent_0005 Agent/Multiple/Iterador
* Agent_0004 Agent/Actuador/Nutridor
* Agent_0003 Agent/Posicionador/Zonal
* Agent_0006 Agent/Multiple/Temporitzador
* Agent_0001 Agent/Direccionador/Aleatori

+ Agent_0002
- Posicio 1271
```

```
- Radi 1
- Direccionador Agent_0001

+ Agent_0004
- Posicionador Agent_0003
- Composicio 31 0

+ Agent_0003
- Posicio 8
- Radi 1
- Posicionador Agent_0002

+ Agent_0005
- Accio Agent_0004
- Accio Agent_0003
- Iteracions 20 0 0

+ Agent_0001
- Direccio 2192479406

+ Agent_0006
- Accio Agent_0001
- CicleActiu 1 0 1
- CicleInactiu 5 0 1
- CicleActual 4 Inactiu
```

```
+ Agent_0000
- Accio Agent_0002
- Accio Agent_0005
- Accio Agent_0006
```

### 5.6.9 Disseny del abocat a disc i de la recuperació

TODO: Agents: Disseny del abocat a disc i de la recuperació

## 5.7 Els organismes

### 5.7.1 Visió externa dels organismes

Externament, sigui quina sigui la estructura interna de l'organisme, ha de proveir a biosistema els següents serveis.

- L'organisme ha d'anar proveint al biosistema de instruccions que l'indiquin les seves accions. Com es generen les instruccions és qüestió dels propis organismes.
- L'organisme proveeix al biosistema un conjunt de registres que formen el seu fenotip. El biosistema pot modificar-los i consultar-los segons ho requereixin les instruccions d'aquest o d'altres organismes.
- L'organisme ha d'implementar unes funcions vitals que modifiquin l'estat intern[2] de l'organisme. El biosistema ha de proporcionar tots els paràmetres que necessitin les funcions vitals, inclosos els que, segurament, provinguin de valors del fenotip.
- També han d'implementar operadors per tal de crear nous organismes a partir d'altres i organismes aleatoris.

### 5.7.2 Model metabòlic dels organismes

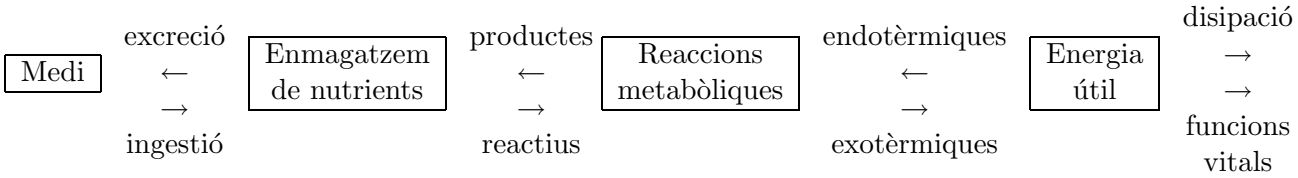



Figura 5.6: Model metabòlic dels organismes a Bioscena

[2]L'estat intern no inclou, per exemple, canvis de posició, donat que són dades que no són intrínseques al propi organisme, sinó que són el resultat de la seva integració en el biosistema. Aquest altre tipus de dades les controla la Comunitat.

Dintre de l'organisme hi ha dos formes de tenir l'energia:

- En forma d'energia útil.
- En forma de nutrients.

No podem passar d'una a l'altre, si no és mitjançant un procés metabòlic. Per un costat, l'energia útil és l'única que pot fer-se servir a la majoria de processos vitals. Per un altre, aquesta energia útil té una caducitat de forma que, quan passa un cert temps de la seva obtenció, es disipa. La figura 5.6 indica els camins que poden fer els nutrients dins de l'organisme.

**Ingestió**

Com s'ha explicat a la secció 5.5, els nutrients tenen un identificador o patró qualitatiu que indica quin tipus de nutrient és. Per seleccionar un nutrient dins d'un grup, només cal oferir una clau o patró de cerca i una tolerància respecte a aquest patró.

Per introduir un nutrient dins del seu cos, un organisme només té que precissar el lloc d'on vol extreure-ho (un substrat o un altre organisme a una posició determinada), si existeix l'element que compleixi aquest requisit i es donen totes les altres condicions necessàries, el nutrient és introduït a l'interior de l'organisme.

Es possible configurar el sistema perque el fet d'engolir un nutrient comporti un guany energètic sense necessitat de processar-ho.

També seria possible configurar un nombre màxim de elements que es mantenen al pap. Això és molt aconsellat si, per les característiques del biosistema els organismes tendeixen a acomular excessius nutrients amb la conseqüent ocupació de memòria. Al igual que al substrat, si en sobren s'eliminen els més antics.

### Metabolisme

Un cop els nutrients estan dins de l'organisme, pot metabolitzar-los, amb l'objectiu d'obtindre energia, un producte resultant o totes dues coses. Els productes resultants es poden fer servir per una altra reacció, per mantenir-ho de reserva, o per excretar-ho.

L'organisme pot fer ús de l'energia que s'obté de les reaccions durant un temps limitat, si no es consumeix dintre d'aquest temps, aquesta es disipa. Es preten que les diferents circumstàncies permetin que hi hagi un equilibri entre els nutrients acomulats i l'energia processada.

Pot ser, una espècie tendeix a acomular masses nutrients. Si això passes, els seus depredadors tenderien a augmentar en nombre d'organismes i en nombre d'espècies.

### Excreció

Tant els nutrients introduïts recentment, com els nutrients que són productes d'una reacció química poden ser excretats per l'organisme. Les oportunitats que donem a l'evolució amb aquest mecanisme són molt diverses.

- Treure a fora els nutrients que ja ens fan més nosa que servei.
- Proporcionar a la descendència nutrients via excreció.
- Detectar als membres d'una espècie o el seu estat per les mol·lecules excretades.
- ...

El caracter obert que poden adoptar les soluccions dels organismes implica que la llista anterior pot no ser tancada, o pot ser els organismes no arribin a cap dels punts anteriors.

Els nutrients excretats s'indiquen amb un element bàsic i una tolerància. Es fa una cerca, amb la funció de compatibilitat, dins dels nutrients que hi ha a l'interior de l'organisme.

TODO: Balanç energètic de l'excreció TODO: Elements configurables de l'excreció

#### Atac i defensa

Un organisme, de banda d'obtindre nutrients del medi, també els pot obtindre d'altres organismes. Per fer-ho, només cal indicar una posició on suposadament hi ha un altre organisme, un element base i una tolerància, i una clau d'atac. Aquesta clau d'atac, en enfrentar-la a la clau de defensa de la victima en resulta la força final de l'atac. Aquesta força indica el nombre de vegades que es provarà d'extreure un nutrient del pap de la victima segons l'element base i la tolerància.

Com es pot deduir, d'un atac es poden obtindre molts nutrients al mateix temps. Això prova de compensar els desavantatges de desenvolupar una conducta depredadora.

TODO: Balanç energètic de l'atac TODO: Elements configurables de l'atac

### 5.7.3 Model genètic dels organismes

#### Trets generals

En aquest projecte, estan diferenciats els conceptes de cariotip i genotip. El cariotip, simplement és un seguit de dades, significatives o no, organitzades en cromosomes. El genotip és la interpretació de la informació útil que s'hi troba al cariotip i està organitzada en gens.

Si els cromosomes que formen el cariotip són unitats estructurals del codi genètic, els gens que formen el genotip en són les seves unitats funcionals. Aquest divisió té les següents justificacions:

1. Permet un model multicromosómic, amb un nombre i longitud de cromosomes variable.
2. Permet implementar, sobre els cromosomes, operadors de mutació i creuament planers.
3. La traducció de cariotip a fenotip, ens permet detectar promotors i terminadors, proporcionar operadors, eliminar introns... facilitant d'aquesta forma l'implementació d'aquests fenòmens.

4. És més semblant al comportament biològic.

**El cariotip i els cromosomes**

Cada organisme conté un nombre variable de **cromosomes** que, en conjunt, formen el **cariotip**. Cada cromosoma està format per una seqüència de bases representada cadascuna amb un bit o un grup de bits.

Tot i que la unitat bàsica del cromosoma és la base (un conjunt reduït de bits), a la implementació, no es considera aquesta unitat més que per a fer algun tipus de mutació puntual. Per a la resta de manipulacions ho farem a nivell de codó, donat que no fa falta arribar a nivell de base i és més òptim accedir-hi. En aquesta representació, un codó coincideix amb una paraula doble (32 bits).

El cromosoma, com a tal, no és una unitat d'informació sinò un medi on estan les dades genètiques. És un medi que pot tenir errors i provocar mutacions. Dins d'un organisme, la tasa de mutació de cada cromosoma és proporcional a la seva longitud.

El cariotip és el responsable de les mutacions germinals (en contraposició a les somàtiques, com es va comentar a la secció **??**), és a dir, les mutacions que passaran a la descendència.

Tot i que la probabilitat de mutació ha de dependre, en gran mesura, dels agens mutagens del medi, els organismes han de poder controlar genèticament la probabilitat de mutació per adaptar-la a la seva situació. En posar en mans de l'evolució la probabilitat de mutar estem confiant en que l'coevolució castigarà els organismes que no mutin en front dels que mutin doncs els primers no podran millorar.

El control sobre la probabilitat de mutació d'un organisme es fa amb una clau que té el propi organisme. Aquesta clau es pot adaptar, en major o menor mesura, al llarg del procés evolutiu, a una d'equivalent que hi ha a cada posició del substrat.

TODO: Cal clarificar més com determinar la probabilitat de mutació.

Un cop es dóna, a un organisme, la probabilitat de mutar, cal decidir com es fa la mutació, es a dir, amb quin operador de mutació es fa. Cada organisme té codificada genèticament una ponderació per a cada operador de mutació.

$$Probabilitat(operador_i) = \frac{ponderacio_i}{\sum_j ponderacio_j} \tag{5.5}$$

Una mutació és cromosòmica si afecta a diversos cromosomes alhora, sent llavors el cariotip el responsable d'executar-la. O bé, si la mutació és gènica o biomol·lecular, afectarà nomes a un sól cromosoma. En aquest cas el cariotip li traspassa la responsabilitat a un cromosoma per que muti. Com decidim en quin cromosoma aplicarem la mutació? Ho farem mitjançant una formula com la de la equació 5.5, pero en comptes de tenir en compte una ponderació codificada genèticament, tindrem en compta la longitud del cromosoma en front de la longitud total del cariotip. Així fem que sigui uniforme la densitat de mutació.

**Els gens i el genotip**

El **genotip** és la traducció del cariotip a elements significatius. Aquests elements significatius s'agrupen en **gens**, que s'interpreten a partir dels codons del cromosoma.

Cada gen té una zona operadora que activa o desactiva el gen segons certa condició que depèn del fenotip o indirectament, mitjançant el fenotip, del món exterior.

La zona estructural (seguint de forma aproximada la nomenclatura de Jacob i Monod) és la que es veu controlada per la zona operadora. En ella estan la informació (instruccions) que s'executaran si la zona operadora ho permet.

En resum, la transcripció del cromosoma en gens constarà de diverses parts:

1. Indentificació de la zona promotora (indica l'inici d'un locus)

2. Indentificació de la zona terminadora corresponent (indica el final d'un locus)

3. Identificació (justament després de la zona promotora) i interpretació de la zona operadora (indicarà quan cal executar el gen traduït).

4. Eliminació d'introns (sequencies de codons no significatius) entre promotora i terminadora

5. Traducció de la zona estructural (la que porta les instruccions que s'executaran amb el gen)

Per raons d'optimització, aquesta transcripció es fa només una vegada durant la vida de l'organisme, encara que, a la natura, la transcripció de l'ADN es fa contínuament. Això no té implicacions massa importants, donat que ens guardem amb el gen el significat de la seva zona operadora que ens serveix per simular el comportament temporal de la transcripció.

**El fenotip**

El que anomenem propiament *fenotip* és un conjunt de 16 registres de 32 bits que té cada organisme. Representen el cos físic de l'organisme.

El fenotip es modifica per acció directa del genotip. Sovint, si es tracta d'operacions sensorials, aquestes modificacions depenen del medi o de l'estat intern. Les operacions també poden ser motores, i en aquest cas el contingut del fenotip és qui afecta al medi i/o a l'estat intern. En resum, tenim els següents grups d'instruccions segons el seu efecte sobre el fenotip, el medi i l'estat intern:

**Instruccions d'operador:** Controlen quines instruccions s'executen segons l'estat del fenotip.

**Instruccions motores:** Modifiquen el medi i/o l'estat intern en funció de paràmetres codificats al fenotip.

**Instruccions sensores:** Introdueixen, dins del fenotip, informació del medi i/o de l'estat intern

de l'organisme. L'execució de instruccions sensores també depenen de paràmetres que es troben codificats al fenotip.

**Instruccions fenotípiques:** Modifiquen el fenotip sense que en depengui d'altre cosa que el fenotip i la instrucció executada.

Per últim, cal remarcar que el fenotip és un dels dos mitjans que tènen els organismes per reconeixer-se, juntament amb la detecció de mol·lècules excretades. Hi ha instruccions sensores el resultat de les quals depén del fenotip de l'organisme que es troba a una posició relativa.

En resum, el fenotip és el mecanisme principal d'interacció que tenen els organismes.

### 5.7.4 Operacions sobre l'estat intern

TODO: Descriure les operacions sobre l'estat intern

### 5.7.5 Conjunt d'instruccions

TODO: Descriure el conjunt d'instruccions (i com s'executen?)

TODO: Posar a algun lloc el tema de les mutacions somàtiques

## 5.8 Eines d'analisis

### 5.8.1 Mecanismes d'especiació

Per que l'usuari pugui extreure una informació útil, cal que poguem agrupar els individus en espècies. És clar que a la nostra comunitat, al igual que a la natura, l'especiació és un fenòmen que cal que emergeixi, L'espècie, no és quelcom intrínsec a tot individu; és a dir, que el concepte d'espècie no estarà implementat al codi genètic o als mecanismes de funcionament comuns dels individus sinó que caldrà observar-ho en el seu comportament.

El concepte clàssic d'espècie considera que dos individus són de la mateixa espècie si són capaços de donar descencència fèrtil.

A la biologia moderna es considera que la diferenciació de les espècies no està tant en la capacitat sinó en el fet mateix de reproduir-se. En això pot influir:

- Compatibilitat genètica
- Compatibilitat d'acoplament
- Compatibilitat geogràfica
- Altres

També, cal adonar-se de que totes aquestes consideracions es refereixen als individus que es reprodueixen sexualment i es creuen. Còm es poden identificar les espècies dels individus que es reprodueixen asexualment? En quin punt es considera que dos descendents d'un mateix individus sòn d'una espècie diferent?

A més, degut a la seva qualitat emergent, l'especiació no estarà sempre ben definida. També pot ser que la selecció a l'hora de reproduir-se tingui en compte altres mecanismes que no pas l'especiació.

Tota aquesta conjuntura ha obligat a deixar de banda el concepte d'espècie cap al concepte, una mica més relaxat, de grup reproductiu que, tot i la relaxació, segueix proporcionant informació útil sobre les diferents poblacions del biosistema. Considerem que un grup reproductiu és un conjunt d'organismes que es creuen entre sí o provenen d'ancestres que s'han creuat entre sí dintre d'un cert període de temps o d'un cert nombre de generacions.

La política que determina els grups reproductius es basa en un marcatge històric.

Cada individu s'associa amb un taxó que no és més que una seqüència de marques amb diferent antiguitat. Les marques es traspasen idèntiques a la descendència via mitosi.

Cada cert temps, es fa una discriminació que consisteix en fondre les dos marques més antigues de cada taxó en una sola i afegir una marca nova que diferenciarà els individus que fins llavors compartien les mateixes marques i que pendrà importància a mida que adquireixi antiguitat.

Cóm es produirà aquesta discriminació? Imaginem que existeixen individus amb els taxons següents:

| | | | | |
|---|---|---|---|---|
| A A A A A A<br>A A A A A A<br>A B A A A A<br>A B B A A A<br>A B B A A A<br>B A A A A A | ⇒ | **A** A A A A<br>**A** A A A A<br>**B** A A A A<br>**B** B A A A<br>**B** B A A A<br>**C** A A A A | ⇒ | A A A A A **A**<br>A A A A A **B**<br>B A A A A **A**<br>B B A A A **A**<br>B B A A A **B**<br>C A A A A **A** |
| Taxons originals dels individuus població abans de la discretització | | Fusió dels dos taxons més antics | | Discriminació dels individus amb les mateixes marques amb una nova: |

D'altra banda, hem de considerar el que passa quan es creuen dos individuus que pertanyen a diferents taxons.

Quan dos individus es creuen, s'asimilen les marques des de la marca més antiga fins a la

primera que els diferencia als dos (marca discriminant). Es a dir, a tots els taxons cal revisar les marques. Pot ser es veu més clar amb un exemple. Considerem el següent conjunt de taxons.

A A A A A A
A A A A A B
A A A A B B
A A A B A B
A A A B B A
A A A B B B
A A B A A A

Si es creuen AAAABB i AAABBA, cal considerar equivalemts les subseqüencies de marques AAAA i AAAB equivalents. Tots els taxons que comencin per AAAB els canviem per AAAA sense oblidar-nos de modificar la següent marca més jove que la discriminant amb l'objectiu de que els taxons asimilats mantinguin el sentit.

A A A A A A $\dashrightarrow$ A A A A A A
A A A A A B $\dashrightarrow$ A A A A A B
A A A A B B $\dashrightarrow$ A A A A B B
A A A B A B $\Rightarrow$ A A A *A* *C* B
A A A B B A $\Rightarrow$ A A A *A* *D* A
A A A B B B $\Rightarrow$ A A A *A* *D* B
A A B A A A $\dashrightarrow$ A A B A A A

De cara a la implementació, he considerat separar la major part del procés associat a la determinació de grups reproductius en un objecte independent anomenat taxonomista. Aquest objecte s'encarrega de mantenir els taxons al dia mitjançant una interfície estreta que manté amb el processador.

A dins de la comunitat es manté una informació mínima: l'identificador del taxó al que pertany cada individu. El tràfec d'informació a l'exterior del objecte taxonomista es basa exclusivament en el pas d'aquests identificadors. La interfície oferida permet al processador:

- Incrementar o decrementar la poblacio assignada a un taxó
- Creuar un parell de taxons
- Generar un nou taxó (Per individuus generats espontàneament)

- Envellir les marques i discriminar la població que comparteix el mateix taxó
- Determinar el grau de parentesc entre dos individuus

Quan hem de discriminar o quan fussionem dos taxons, necessitem que la Comunitat i el Taxonomista cooperin.

Quan cal discriminar la població, la Comunitat demana al Taxonomista, per a cada individu, un nou taxó, basant-se en el taxó antic i el número de individus que en queden sense discriminar d'aquest.

El nou taxo duu les marques X.X.X. ... .X.N on N és el número de queden sense discriminar.

Quan, fruit d'un creuament, es fusionen dos taxons, un dels dos taxons és assimilat per l'altre i, en conseqüència, els individus associats al taxò assimilat, cal associar-los al taxó assimilador.

Per dins, el taxonomista està composat per una llista indexada de taxons (taxonari). Els números d'índex es referencien des de cada individu pertanyent a la Comunitat.

Una alternativa al taxonari hagués sigut una implementació en arbre en comptes de la llista indexada. En cada node hi hauria una marca i a les fulles els identificadors de cada taxó. La implementació en arbre simplifica molt la lògica dels algorismes de discriminació i creuament però complica altres operacions internes que amb la llista indexada són trivials.

La llista indexada permet un accés directe als taxons i, a més, els manté ordenats de tal forma que les cerques de grups de parentesc tenen un cost temporal mínim.

TODO: Revisar tots aquests procediments amb la nova forma de portar la comunitat.

### 5.8.2 Estadístiques de població

### 5.8.3 Estadístiques inter-poblacionals

# Capítol 6

# Manual d'usuari

## 6.1 Instalació de l'entorn

## 6.2 Configuració

## 6.3 Operació normal

## 6.4 Interpretació de la sortida per pantalla

## 6.5 Interpretació dels logs i les dades de sortida

## 6.6 Intervenció

# Capítol 7

# Manual del programador

## 7.1 Programació de noves topologies

Si l'usuari necessités crear un nou tipus de topologia, cal que la faci heretar de CTopologia, que és la classe que defineix el mínim per reservar memoria pel substrat de cada posició. CTopologia també estableix el protocol que han de seguir les subclasses, perqué la resta del sistema l'acepti sense haver de canviar-ho.

El secret està en el fet de que tot el sistema manega identificadors de posicions que són enters sense signe de 32 bits. Tot el significat que poden tenir aquests identificadors el manega la topologia internament.

Quan es deriva de CTopologia, el principal que caldria redefinir, si cal, és:

- Un **constructor** significatiu per a la topologia. Per exemple, en una topologia rectangular és significatiu indicar l'altura i l'amplada. El constructor de CTopologia simplement reserva espai per N casselles. Caldria calcular aquesta N per passar-se-la.
- `t_posicio CTopologia::desplacament (t_posicio origen, t_desplacament desplacament)`: Una funció per averiguar la posició destí en aplicar-li un vector de desplaçament a una posició origen. CTopologia, la defineix de tal forma que el resultat és una posició destí aleatoria.

- `bool CTopologia::esValid(t_posicio id)`: Una funció per saber si un identificador és vàlid. Només cal redefinir-ho si es modifica la correspondència directa entre identificador de posició i index de cassella en l'array de substrats reservada per CTopologia::CTopologia

- `t_posicio CTopologia::posicioAleatoria ()`: Una funció per obtindre aleatòriament una posició vàlida de la topologia. La funció general que no caldria redefinir seria

```
{
    uint32 pos;
    do {pos=rnd.get();} while (!esValid(pos));
    return pos
}
```

  però, CTopologia no fa servir aquest algorisme donat que optimitza agafant un número aleatori entre 0 i N. Aquesta optimització funciona mentre es mantingui la correspondència entre identificador i index abans comentada. Si la subclasse la trenca, es quan cal redefinir la funció.

- `bool CTopologia::unio (t_posicio origen, t_posicio desti, t_desplacament & desp)`: Una funció per calcular el primer d'un conjunt de desplacaments que cal fer per anar de l'origen al destí. Retorna cert si el desplacament és suficient per arribar a la posició destí. CTopologia, la defineix de tal forma que el resultat és un desplaçament aleatori i retorna sempre fals (mai hi arriba).

Pot ser molt ilustratiu, de cara a implementar noves topologies, fixar-se en les ja existents com CTopologiaToroidal.

## 7.2 Programació de nous agents

De cara a afegir nous agents al sistema, s'aconsella seguir els següents passos:

1. Llegir per sobre el codi dels agents ja implementats per assimilar les solucions que s'han donat a problemes que segurament es tornaran a repetir als nous agents. També convé mantenir uniforme l'estil de programació i l'ordre intern dels fitxers per fer-ho més mantenible a tercers. El més pràctic es partir d'una còpia d'un agent que tingui, estructuralment, tot o gran part del que interesa implementar.

2. Escollir la classe d'agent de la que volem heretar l'agent. Generalment voldrem que el nou agent pertanyi a un dels quatre grans grups funcionals d'agents:

   - Subordinadors (CMultiAgent i subclasses) si controla l'accionat d'altres agents
   - Posicionadors (CPosicionador i subclasses) si controla una posició en el biòtop
   - Direccionadors (CDireccionador i subclasses) si controla una direcció
   - Actuadors (CActuadors i subclasses) si modifica el substrat a una posició

   Si no pertany a cap dels quatre grups, caldria plantejar-se heretar de CAgent directament. En aquest cas, convé fer un esforç i fer una classe intermitja que pugui englobar altres agents en el futur. Anomenarem CAgentNou al nou agent afegit i CAgentVell a l'agent del qual heretem.

3. Adaptar el constructor de CAgentNou per que proveeixi els paràmetres del constructor de la superclasse. Posicionadors i direccionadors, per exemple, necessiten una referència a un biòtop en el constructor. A les classes derivades de CPosicionador està implementat com fer-ho.

4. Afegir dins del constructor, la línia.

   `m_tipus+="/ElMeuSubtipus";`

   que afegeix la cadena de subtipus a l'identificador de tipus que hereta de la superclasse.

5. Afegir els nous atributs (variables membre) dels que en depèn l'estat de l'agent i les funcions d'accés als mateixos.

6. Inicialitzar dins del constructor els nous atributs als valors per defecte. Els atributs que siguin dependències amb altres agents, o agents subordinats, es recomana que siguin punters, i no referències, per poder-ho deixar sense especificar al constructor. S'inicialitzen sempre com a punter a NULL. Cal procurar que, si el punter no apunta a un agent vàlid el seu valor sigui NULL i tenir-ho en compte quan hi accedim per evitar accesos il·legals a memòria. Es veu clarament aquesta idea llegint el codi d'alguns agents que ho fan.

7. Afegir dins del destructor, l'alliberament de memòria ocupada pels agents subordinat. Les dependències no s'han de alliberar pas.

8. Redefinir la funció membre `virtual void CAgentNou::operator() (void)` per que faci el que hagi de fer quan l'agent és accionat. Si es tracta d'un actuador, no cal redefinir aquesta sino `virtual void CAgentNou::operator() (CSubstrat & s)` on `s` és el substrat que hem de modificar. [1]

9. Redefinir la funció `virtual void CAgentNou::dump(CMissatger & msg)` per que cridi a la funció corresponent de la superclasse (CAgentVell a l'exemple) i, després, inserti en el CMissatger les noves línies de configuració dels paràmetres que afegeix l'agent:

```
void CAgentNou::dump(CMissatger & msg)
{
```

[1] Veure l'apartat 8.8 que parla del que cal fer si es redefineix el substrat

```
    CAgentVell::dump(msg);
    msg << "- UnParametreNou " << m_valor1 << " " << valor2 << endl;
    msg << "- UnAltreParametreNou " << m_valor3 << endl;
}
```

10. Redefinir la funció `virtual bool CAgentNou::configura(string parametre, istream & valors, t_diccionariAgents & diccionari, CMissatger & errors)` per mirar si el `parametre` és un dels que ha afegit CAgentNou. Si ho és cal parsejar l'istream `valors` en busca dels valors corresponents, reportar els errors que es produeixin pel CMissatger `errors` i retornar cert per dir que el paràmetre era de la classe. Si no ho és, cal cridar a la funció corresponent de la superclasse per que ho pugui interceptar ella. El diccionari serveix per, donat un nom d'agent de l'arxiu, obtindre un punter a l'agent que s'ha creat que, pot ser, té un nom diferent. El diccionari és un `map<string, CAgent*>`, el seu funcionament s'explica a qualsevol manual sobre les Standard Template Libraries de C++. La estructura general de la funció configura quedarà com això:

```
bool CAgentNou::configura(string parametre, istream & valors,
t_diccionariAgents & diccionari, CMissatger & errors)
{
    if (parametre=="UnParametreNou") {
        // Parsing dels valors...
        return true;
    }
    if (parametre=="UnAltreParametreNou") {
        // Parsing dels valors...
        return true;
```

```
    }
    // Li deixem a la superclasse que l'intercepti si vol
    return CAgentVell::configura(parametre, valors, diccionari, errors);
}
```

11. Si cap dels atributs (m_dependencia a l'exemple) és una dependència amb altre agent, cal redefinir la següent funció com segueix:

```
list<CAgent*> CAgentNou::dependencies() {
    list<CAgent*> l=CAgentVell::dependencies();
    if (m_dependencia) l.push_back(m_dependencia);
    return l;
}
```

12. Si cap dels atributs (m_subordinat a l'exemple) és un agent subordinat, cal redefinir la següent funció com segueix:

```
list<CAgent*> CAgentNou::subordinats() {
    list<CAgent*> l=CAgentVell::subordinats();
    if (m_subordinat) l.push_back(m_subordinat);
    return l;
}
```

13. Afegir a l'arxiu `Agent.cpp` un include a `AgentNou.h` i, a la funció estàtica CAgent::CreaAgent(...) una línia com les que ja n'hi ha per cada tipus d'agent, però, per a CAgentNou. Això permet que la funció CAgent::ParsejaArxiu pugui reconeixer el nou tipus als arxius de configuració.

De tots els punts anteriors el que potser és una mica més particularitzat són els atributs i els mètodes d'accés als mateixos, i el mètode d'accionament (o d'actuació en el cas dels actuadors). Per a la resta de coses el més pràctic es fer un cut&paste dels agents ja implementats i retocar el mínim.

## 7.3 Programació de nous substrats

# Part IV

# Conclusions

# Capítol 8

# Vies de futur

## 8.1 Experimentació amb el model presentat

La principal via de futur d'aquest projecte és la seva aplicació. Això vol dir plantejar experiments que es puguin realitzar sobre aquest sistema.

Com a exemple tenim la bateria d'experiments que s'han anat fent a la Universitat de San Diego sobre el sistema LEE.

TODO: Quins experiments

## 8.2 Sistema de control

Es pot adaptar el sistema d'una forma molt directa a altres models d'organisme. Per exemple, es podria implementar fàcilment organismes cognitius basats en diferents tipus de xarxes neuronals, de forma similar a com s'ha fet als sistemes LEE, o en sistemes classificadors, entre d'altres.

Dels resultats d'aquesta primera experiència hem resaltat que potser es milloraria el sistema si s'introdueixen o es modifiquen alguns dels seus elements.

TODO: Modificacions suggerides

## 8.3 Sistema genètic

També és possible afegir altres elements interesants a la part genètica.

**Operadors de mutació:** Els que hi ha implementats potser no són del tot els idonis per un problema donat.

**Evolució dels operadors de mutació** Donada la hetereogeneitat en la distribució dels gens als cromosòmes, pot ser és útil que el tipus d'operador de mutació, o la probabilitat de mutar amb un o altre, també evolucioni amb el mateix cromosoma.

**Sexualitat:** Cal recordar que el model presentat no considera sexualitat, la qual cosa, no permet gaire variabilitat genètica. És trivial introduir mecanismes sexuals no lligats a la reproducció com els dels organismes unicel·lulars. Només caldria considerar l

**Cariotips poliploides:** TODO

## 8.4 Metabolisme

Les reaccions metabòliques s'han implementat d'una forma molt simple si tenim en compte les possibilitats de configuració que donen els sistemes LEE. Els sistemes LEE permeten establir en una taula de reaccions binàries on cada cassella conté una llista de productes i el balanç d'energia per a cada dos possibles reactius.

## 8.5 Interfícies

En aquest projecte, de cara a mantenir la portabilitat, s'ha volgut mantenir una interfície de tipus terminal de texte. Malgrat tot, aquesta interfície probablement no li serà prou amigable a un usuari final, a més, la interfície en mode texte presenta moltes limitacions en la presentació de gràfics i, sobretot, en la quantitat de dades representables i la seva llegibilitat.

Així doncs, es podria implementar una interfície gràfica que facilités la feina als usuaris del sistema.

Aquesta tasca ve facilitada pel fet de que, durant l'elaboració del nucli, s'ha tingut molt present el paradigma model-vista-controlador:

El nucli és molt independent de la interfície. Fins i tot les traces i els missatges de debug que han d'estar implementats a dintre del nucli, estan preparats per no necessitar d'una plataforma concreta. Fan servir una classe adaptadora que pot visualitzar-les en un terminal, en una finestra de la llibreria Curses, en MessageBoxes o controls d'edició de MS-Windows o potencialment d'X-Windows.

Els pocs objectes visuals implementats en mode texte (mapes, gràfics comparatius i gràfics d'evolució temporal) ofereixen un protocol d'accés que podria implementar a sota elements gràfics d'alta resolució.

## 8.6 Eines adicionals

A mida que compliquem el problema, les modificacions dels fitxers de configuració dels agents externs es tornen molt feixugues. De forma no massa complicada es pot reutilitzar el codi del nucli per fer una petita eina que permeti editar l'estructura arboriforme dels agents.

Una altra eina, podria ser una que ens permetés editar els individuus o el seu material genètic. Editar els organismes d'una forma fàcil facilitaria els experiments dels biòlegs i obtenir un organisme primigeni eficient per estalviar temps d'evolució. Proposant com a cultiu primigeni les cepes dominants de diversos experiments editades convenientment per falcilitar la variabilitat genètica.

## 8.7 Funcionalitats

Grabar el biosistema Importar i exportar organismes

## 8.8 Optimitzacions

Hi ha punts del sistema dels qual encara es pot optimitzar el seu comportament:

Per exemple, es podria fer compartir a tots els organismes amb el mateix material genètic un punter a la mateixa estructura de dades. La millora que representa això en temps i en memòria tot i que és molt clara amb mutació o intercanvis sexuals no reproductius, deixaria de ser útil amb intercanvis sexuals lligats a la reproducció donat que la variabilitat genètica seria molt alta.

# Bibliografia


[AML86] Francisco J. Segovia Pèrez Angel Morales Lozano. *Programacion orientada a objetos, aplicaciones con Smalltalk*. Paraninfo, Madrid, 1986.

[AMR97] Jamshid Ghaboussi Anne M. Raich. Autogenesis and redundancy in ga representation. In *Proceedings of the 1997 International Conference on Genetics Algorithms ICGA97*. Michigan State University, 1997.

[Dar59] Charles Darwin. *The Origin of the Especies*. Guttemberg Project, 1859.

[FH91] Thomas Back Frank Hoffmeister. Genetic algortihms and evolution strategies: Similarities and differences technical report sys-91/2. Technical report, Systems Analysis Research Group, LSXI, Dept. of Computer Science, 1991.

[Fra97] Wolfgang Banzhaf; Peter Nordin; Frank D. Francone. Why introns in genetic programming grow exponentially. In *Proceedings of the 1997 International Conference on Genetics Algorithms ICGA97*. Michigan State University, 1997.

[JCGC98] Grupo G9 Jose Carlos González Cristóbal. *Vida artificial: análisis y simulación*. PhD thesis, TODO: Investigar de que escuela es, 1998.

[Kar97] Hillol Kargupta. Relation learning in gene expression: Introns, variable length representation, and all that. In *Proceedings of the 1997 International Conference on Genetics Algorithms ICGA97*. Michigan State University, 1997.

[Lam86] Leslie Lamport. *LaTeX: A Document Preparation System*. Addison-Wesley, 1986.

[Man] G. Mangiarotti. *Del gen al organismo. Biologia General*. Picin, Padova, Italia.

[May97] Helmut A. Mayer. ptga's, genetic algorithms using promoter/terminator sequences evolution of number, size and location of parameters and parts of representation. In *Proceedings of the 1997 International Conference on Genetics Algorithms ICGA97*. Michigan State University, 1997.

[MB93] Filippo Menezer and R. K. Beleg. Latent energy enviroments: A tool for artificial live simulations. Technical Report 93-301, Computer Science and Engineering Department, La Jolla, CA 92093-0114, 1993.

[MB94] Filippo Menezer and R. K. Beleg. Evolving sensors in environments of controled complexity. In R. Brooks and P. Maes, editors, *Artificial Live IV*, La Jolla, CA 92093-0114, 1994. MIT Press.

[MB95] Filippo Menezer and R. K. Beleg. Latent energy enviroments. Technical report, Computer Science and Engineering Department, La Jolla, CA 92093-0114, 1995.

[MB96] Filippo Menezer and R. K. Beleg. Complex environments to complex behaviors. *Adaptive Behavior*, 4(3/4):317–363, 1996.

[MBC95] Filippo Menezer, R. K. Beleg, and Federico Cecconi. Maturation and evolution of imitative learning in artificial organisms. Technical Report 506, Computer Science and Engineering Department, La Jolla, CA 92093-0114, 1995.

[Men94] Filippo Menezer. Changing latent energy enviroments: A case for evolution of plasticity. Technical Report 94-336, Computer Science and Engineering Department, La Jolla, CA 92093-0114, 1994.

[Ray93] Thomas S. Ray. Jugué a ser dios y creé vida en mi computadora. *Epistemología e Informática*, pages 257–267, 1993.

[Ser96] Anselmo Pérez Serrada. *Una Introducción a la Computación Evolutiva*. PhD thesis, TODO: Investigar de que escuela es, Marzo 1996.

[Str] M. Strickberger. *Genética*. Ed. Omega, Barcelona, 3 edition.

[Str86] Bjarne Stroustrup. *El C++, lenguaje de programacion*. Addison-Wesley, 2 edition, 1986.

[TS97] James A. Foster Terence Soule. Comments on the intron/exon distinction as it relates to the genetic programming and biology. In *Proceedings of the 1997 International Conference on Genetics Algorithms ICGA97*. Michigan State University, 1997.

# Índex alfabètic