Reproducció al Laboratori:

Justificació de les operacions

24 de Maig de 2014

Albert Segarra Roca

1. Creixement d'un organisme (estirament)

1.1 L'operació privada `estirar_arbcelula`

1.1.1 Implementació

```
void Organisme::estirar_acel(Arbcel&a, int&idmax) {
    /* PRE: a = A
     *
     * idmax conté l'identificador màxim de totes les cèl·lules d'a
     *
     * a és no buit
     */
    Celula c = a.arrel();
    Acel fe, fd;
    a.fills(fe, fd);
    if (fe.es_buit() and fd.es_buit()) {
        ++idmax;
        Celula f1(idmax, c.es_activa());
        Acel fbuit;
        fe.plantar(f1, fbuit, fbuit);
        ++idmax;
        Celula f2(idmax, c.es_activa());
        fd.plantar(f2, fbuit, fbuit);
    }
    else {
        // fe = FE
        if (not fe.es_buit()) estirar_acel(fe, idmax);
        /* HI1: fe és l'arbre resultant d'afegir dues cèl·lules filles
         * a totes les fulles d'FE, amb valors d'identificadors
         * a partir del següent a idmax, i amb activitat o passivitat
         * heredada de la corresponent fulla mare en FE
         *
         * idmax conté l'identificador màxim de totes les cèl·lules d'a i fe
         */
        // fd = FD
        if (not fd.es_buit()) estirar_acel(fd, idmax);
        /* HI2: fd és l'arbre resultant d'afegir dues cèl·lules filles
         * a totes les fulles d'FD, amb valors d'identificadors
         * a partir del següent a idmax, i amb activitat o passivitat
         * de la corresponent fulla mare en FD
         *
         * idmax conté l'identificador màxim de totes les cèl·lules d'a, fe i
         * fd
         */
    }
    a.plantar(c, fe, fd);
    /* POST: a és el resultat d'afegir dues cèl·lules filles a
     *       totes les fulles d'A, amb valors d'identificadors
     *       a partir del següent al màxim identificador d'A,
     *       assignats correlativament d'esquerra a dreta en l'arbre,
     *       i amb activitat o passivitat heredada de la corresponent
     *       fulla mare en A.
     *
     *       idmax conté l'identificador màxim de totes les cèl·lules d'a */
}
```

1.1.2 Justificació

- ***Instruccions inicials:***
  Inicialment, per la *PRE* sabem que *a* és no buit, per tant respectem la precondició de l'operació *arrel()* i podem accedir a l'arrel d'*a* i guardar-ne el seu valor.
  A més, novament com que per la *PRE* sabem que *a* no és buit, podem obtenir els seus fills en *fe* i *fd*, i estarem respectant la precondició de l'operació *fills* donat que *fe* i *fd* són buits i *a* no és buit.

- ***Cas directe:***
  Per la condició de l'if sabem que tant el fill esquerre com el fill dret d'*A* són buits. Això significa que estem en una fulla, i per a complir la *POST* de l'operació ens cal plantar la cèl·lula esquerra amb l'identificador següent a *idmax* i la cèl·lula dreta amb un identificador més que l'esquerra, donat que han de ser valors correlatius i assignats d'esquerra a dreta segons la *POST*, i amb activitat heredada de la cèl·lula mare, en aquest cas *c*.

  Això ho fem plantant les dues cèl·lules creades amb els paràmetres a dalt descrits en *fe* i *fd* (respectivament esquerra i dreta) amb dos arbres buits com a fill esquerre i fill dret. A més hem d'actualitzar *idmax* amb l'identificador màxim, això ho fem incrementant-lo dos cops, donat que sabem per la PRE que *idmax* ja contenia l'identificador màxim anteriorment, i ara la cèl·lula dreta l'hem plantat amb dues unitats més que l'*idmax* inicial, per tant podem estar segurs de que aquest és el valor correcte per a *idmax*.

  Les crides a les operacions de la classe Arbre són correctes donat que per la condició de l'if sabem que *fe* i *fd* són arbres buits.

  Finalment, per acabar de complir la *POST* només ens cal plantar els subarbres *fe* i *fd* a *a*, perquè aquest contingui les dues cèl·lules fulles, que és el que demana la *POST*.
  Respectem la *PRE* de l'operació *plantar* donat que inicialment hem fet l'operació *fills* amb *a*, i per la *POST* d'aquesta operació *a* ens queda buit, a més *a* no es modifica fins a aquesta última operació *plantar*, per tant sabem que *a* és buit abans de *plantar*.

- ***Cas recursiu:***
  Si considerem la proposició *P = (HI1 ∧ HI2)* aleshores tenim que l'únic que ens cal per acabar de complir la *POST* és plantar *fe* i *fd* a l'arbre *a*. *idmax* té el valor que ha de tenir donat que ara *fe* i *fd* passen a ser part de l'arbre *a*, i per *HI2* idmax ja conté l'identificador màxim de totes les cèl·lules d'*fe, fd* i *a*.

Respectem la *PRE* de l'operació *plantar* donat que inicialment hem fet l'operació *fills* amb *a*, i per la *POST* d'aquesta operació *a* ens queda buit, a més *a* no es modifica fins a aquesta última operació *plantar*, per tant sabem que *a* és buit abans de *plantar*.

- ***Finalització:***
  Funció fita: La distància en nombre de nodes desde l'arrel d'a fins a una fulla d'a compleix els requeriments de la funció fita: és un natural i decreix a cada crida, donat que fem les crides amb els subarbres d'a.

2. Ronda de reproducció

2.1 L'operació pública aplicar_ronda_reproduccio

2.1.1 Implementació

```
void Sistema::aplicar_ronda_reproduccio(Ranking&rank) {
    /* PRE: - Població actual del sistema = POB
   * - POB > 0
   * - Sigui i tal que 1 <= i <= POB l'element i  de la llista
   * d'informació d'emparellaments del paràmetre implícit conté la informació
   * dels emparellaments en rondes anteriors de l'organisme amb id = i */
  int pobaux = pob;
  int i = 0;
  Lemp::iterator it = empar.begin();
  Vbool ocupat(pobaux, false);
  nfills_ur = 0;
  /* INV1:
   * - pobaux = POB
   * - 0 <= i < pobaux, pob < vorg.size()
   * - Els organismes amb identificadors 1...i han intentat escollir
   * a un altre organisme per a reproduir-se (si no havien sigut escollits
   * o no estàn morts)
   * - it apunta a la llista d'informació d'emparellaments de l'organisme
   * amb id = i + 1
   * - Sigui k tal que 0 <= k < pobaux, ocupat[k] indica si l'organisme
   * amb id = k + 1 està ocupat en la ronda
   * - nfills_ur conté el nombre d'organismes que s'han reproduït
   * satisfactòriament en la ronda
   * - La llista amb informació dels emparellaments del paràmetre implícit
   * dels organismes amb identificadors 1...i i els organismes amb els quals
   * s'han emparellat (si s'han emparellat) contenen la informació actualizada
   * dels emparellaments que han tingut en la ronda */
  while (i < pobaux - 1 and pob < vorg.size()) {
    if (not ocupat[i] and not vorg[i].es_mort()) {
      bool incomp = true;
      /* INV2: - Si incomp és cert això implica que l'organisme amb
       * id = (*it).ini és mort o ja s'ha emparellat amb l'organisme
       * amb id = i + 1
       * - Si incomp és fals això implica que l'organisme amb
       * id = (*it).ini + 1 no és mort i no s'ha emparellat encara
       * amb l'organisme amb id = i + 1
       * - No hi ha cap organisme amb id < (*it).ini + 1 que no estigui mort
       * o no s'hagi emparellat amb l'organisme amb id = i + 1
       * i < (*it).ini <= pobaux
       * - Si incomp és fals, l'organisme amb id = i + 1 s'ha emparellat
       * amb un organisme amb id' = (*it).ini+1..pobaux <=> id'
       * és viu, no s'havia emparellat encara amb id i no està ocupat
       * en la ronda */
      while (incomp and (*it).ini < pobaux) {
        int cand = (*it).ini;
        incomp = vorg[cand].es_mort() or (*it).e[cand-i-1];
        if (not incomp) {
          if (not ocupat[cand]) emparellar_organismes(ocupat, rank, i, cand, it);
          else {
            ++cand;
            bool emparellables = false;
            /* INV3: - (*it).ini < cand <= pobaux
             * - emparellables indica si l'organisme amb id = i + 1
             * es pot emparellar amb l'organisme amb id = cand
             * - Si emparellables és cert aleshores s'han emparellat els
             * organismes i+1 i cand
             * - No hi ha cap organisme amb id < (*it).ini + 1 que no estigui mort */
            while (not emparellables and cand < pobaux) {
```

```
            if (not ocupat[cand] and not vorg[cand].es_mort()
                                 and not (*it).e[cand-i-1]) {
              emparellables = true;
              emparellar_organismes(ocupat, rank, i, cand, it);
            }
            ++cand;
          }
          /* POST1: - L'organisme amb id = i + 1 s'ha emparellat o no té cap
           * organisme amb el qual emparellar-se en la ronda. Si s'ha emparellat
           * ocupat[i] retorna cert.
           * Si l'organisme i+1 s'ha emparellat i ha donat lloc a un fill,
           * n_fills_ur s'ha incrementat en una unitat */

        }
      }
      else ++(*it).ini;
    }
    /* POST2: - L'organisme amb id = i + 1 s'ha emparellat o no té cap
     * organisme amb el qual emparellar-se en la ronda. Si s'ha emparellat
     * ocupat[i] retorna cert.
     * - Si l'organisme i+1 s'ha emparellat i ha donat lloc a un fill,
     * n_fills_ur s'ha incrementat en una unitat
     * L'atribut (*it).ini s'ha actualitzat correctament */
  }
  ++i;
  ++it;
 }
}
/* POST: - Si s'ha arribat a la població màxima del sistema durant la ronda,
 * els organismes amb identificadors 1..i han intentat escollir a un altre organisme
 * per reproduir-se (o han estat escollits o són morts).
 * Altrament, els organismes amb identificadors 1...POB-1 han intentat escollir
 * a un altre organisme per reproduir-se (o han estat escollits o són morts)
 * - S'ha actualitzat el nombre de fills nascuts en la ronda
 * - El rànking s'ha actualitzat amb la informació de les reproduccions
 * que hagin tingut fills en la ronda
*/
```

2.1.2 Justificació

Comentaris: Cal tenir molt en compte en aquesta justificació el fet que dins de la funció, i de la classe sistema en general, els organismes es tracten amb el seu identificador decrementat en una unitat (en comptes de numerar-se 1,2,3,4...POB es numeren 0,1,2,3...POB-1. Això fa que en certes condicions de l'invariant ens quedi, per exemple, frases del tipus: Els organismes amb identificadors 1...i s'han emparellat en la ronda. És obvi que en aquest cas no serien 1...i-1 donat que la variable i del bucle conté l'identificador decrementat en una unitat, i alhora de passar a referir-nos a l'identificador real ens queda 1...i-1+1 = 1...i

Bucle 1:

- ***Inicialitzacions:***
  Hem de demostrar que la *PRE* juntament amb les inicialitzacions satisfàn l'*INV1*:
  La primera condició de l'*INV1* la satisfem assignant el valor de la població actual del sistema a *pobaux*, que té el valor requerit per la *PRE*.
  La segona i tercera condicions les satisfem assignant 0 a la variable *i*, donat que no hem tractat cap organisme encara.

La quarta condició també la satisfem, donat que l'organisme amb identificador 1 és el primer de la llista per la *PRE*, i deixem l'iterador *it* apuntant al *begin()* com demana *INV1*.
La cinquena condició es satisfà mitjançant la creació del vector *ocupat* amb totes les seves posicions assignades a fals, donat que en aquest cas cap organisme s'ha emparellat encara, per tant no n'hi ha cap d'ocupat, que és el que indica cada posició del vector. A més mai tindrà mida nul·la o negativa donat que per la *PRE*, *POB > 0* i *pobaux = POB* perquè l'acabem d'assignar.
La sisena condició se satisfà perquè de moment encara no s'ha reproduït cap organisme i per tant *nfills_ur* ha de contenir un 0 com demana l'*INV1*.
L'última condició vé satisfeta per la *PRE*, donat que en aquesta ronda encara no hi ha hagut emparellaments.

- ***Condició de sortida:***
  La primera condició és correcte ja que sortim quan *i == pobaux - 1*. Substituïnt *i* per pobaux *-1* en l *'INV1* s'ens garanteix que es compleix el que es demana en la *POST*, a més hem de respectar el que ens diu la segona condició de l'*INV1*.

  La segona condició també ho és ja que si s'arriba a la població màxima durant la ronda la *INV1* ens garanteix que es compleix el que demana la *POST* en aquest cas.
- **Cos del bucle:**
  Suposant que l'*INV1* és certa en l'iteració *i*, hem de veure que les instruccions del bucle fan que sigui certa l *'INV1* per l'iteració *i+1.* Veiem-ho:
  Per complir la primera condició de l'invariant no ens cal fer res, ja que aquesta no depèn de *i*, (de fet només ens cal no modificar el valor de pobaux, que és el que fem).
  La segona condició es compleix per la condició del bucle.
  Per a complir la tercera condició, ens cal comprovar primr si l'organisme és mort o ja està ocupat en la ronda, ja que en aquest cas l'INV1 només ens demana que incrementem la i, que és el que fem. En el cas de que no sigui mort i no estigui ocupat en la ronda, com que entrem en l'if, el que ens demana l'invariant se satisfà donada la *POST2* del segon bucle i l *'INV1* de l'iteració anterior. Només ens cal incrementar la *i* per a fer certa aquesta condició de l'*INV1*.
  El mateix per la cuarta condició, però aquí el que ens cal, a més d'incrementar *i* és incrementar l'iterador *it* perquè apunti a l'element de la llista corresponent al següent organisme (sabem que apunta a l'actual per l *'INV1*), el que després d'incrementar *i* és el que té per identificador *i+1*.
  La cinquena i sisena condicions se satisfàn suposant que *POST1* sigui cert, per tant no ens cal fer res.
  Per a complir l'última condició, i suposant que *POST1* és cert, només ens cal incrementar la *i* perquè es compleixi el que demana.

- ***Finalització:***

  Funció fita: pobaux-1-i compleix les condicions de la funció fita; és un natural ja que *pobaux > 0* per la *PRE* i *0 <= i < pobaux* per l'*INV1* i decreix a cada iteració donat que incrementem la *i*.

Bucle 2:

- ***Inicialitzacions:***
  Inicialment, la *PRE* ens diu que la llista d'informació dels emparellaments del paràmetre implícit està actualitzada. Sigui *it* l'iterador que apunta a la llista d'informació d'emparellaments de l'organisme amb id *k*, és una propietat invariant de *(*it).ini* definida en la seva corresponent descripció que no hi ha cap organisme amb identificador menor a *(*it).ini + 1* tal que l'organisme *k* s'hi pugui emparellar. Així doncs, inicialment per complir les dues primeres condicions de l'*INV2* hem d'assignar a true la variable *incomp* donat que l'organisme amb *id = (*it).ini* es menor que l'organisme amb *id = (*it).ini + 1*. Amb això que acabem d'exposar automàticament també complim la tercera condició, i com que *incomp* és cert la quarta també donat que no ens diu res sobre que ha de passar quan *incomp* és cert.

- ***Condició de sortida:***
  La primera condició és correcta, ja que sortim quan *incomp* és fals, moment en el qual la quarta condició de l'*INV2* ens indica que complim la *POST2*. L'altra condició també es correcta, ja que si *(*it).ini == pobaux* la tercera condició de l'*INV2* ens diu que no hi ha cap organisme amb el qual l'organisme amb *id = i + 1* es pugui emparellar, per tant complim la *POST2*.
- ***Cos del bucle:***
  Hem de demostrar que, suposant que es compleix l'*INV2* per l'iteració *i*, les instruccions del bucle fan que es compeixi per l'teració *i+1*. Inicialment, per l'*INV2* sabem que no hi ha cap organisme amb identificador menor a *(*it).ini + 1* tal que l'organisme amb *id = i+1* es pugui emparellar. Per tant, només ens falta comprovar-ho per al següent l'organisme, l'*(*it).ini + 1*, que és el que fem. En cas que *incomp* sigui cert, el que ens diu l'invariant és que seguim en la mateixa situació, (primera condició), i per tant l'únic que ens falta per acabar de complir-la (juntament amb la tercera) és incrementar *(*it).ini*, que és el que fem. Si incomp és fals, la quarta condició de l'invariant ens diu que els organismes *(*it).ini + 1* i *i+1* s'han d'emparellar sii es donen 3 condicions. Les dues primeres condicions es compleixen per l'if i l'assignació d *'incomp*, en aquest cas, per tant només ens fa falta comprovar si l'organisme no està *ocupat*, i en cas de que no ho estigui (aleshores satisfarem les tres condicions) emparellar-los, i és que gràcies a la *POST* de l'operació *emparellar_organismes* podem estar segurs que complirem la *POST2* (complim la *PRE* de *emparellar_organismes* ja que *i < cand* per *l'INV2 i*

l'INV3 i *cand < pobaux < pobmax* per *L'INV1 i l'INV2* i *rank* és l'associat al sistema per la *PRE*). Si l'organisme està ocupat, hem de seguir buscant, ja que id' va de *(*it).ini + 1* fins a pobaux, i l'(*it).ini+1 acabem de comprovar que no es pot emparellar. Per tant seguim buscant a partir del següent identificador, l'*(*it).ini+2*, que és del que s'encarrega el segon bucle. Per la seva *POST1* sabem que cumplirem l *'INV2*, ja que la segona condició també serà certa donat que no incrementarem *(*it).ini*.

Tots els accessos a vectors són valids donat que cand = (*it).ini < pobaux per la segona condició del bucle.

- ***Finalització:***

  El bucle finalitza donat que decreix la distància entre *(*it).ini* i *pobaux*, ja que l'incrementem en cada iteració, i si no l'incrementem vol dir que *incomp* és fals i per tant sortim igualment del bucle. *Funció fita: pobaux – (*it).ini* (és un natural donat que per *l'INV2 (*it).ini <= pobaux* i decreix a cada iteració.

Bucle 3:

- ***Inicialitzacions:***
  Inicialment, per l'*INV2* sabem que la variable cand conté el valor d'*(*it).ini*. Per a complir la primera condició de l'*INV3* ens cal incrementar cand. A més, per l'*INV2* sabem que l'organisme amb *id = i+1* no es pot emparellar amb l'organisme amb id = cand (recordem que l'id està decrementat en una unitat), per tant per complir la segona condició s'ha d'assignar emparellables a fals. La tercera la complim ja que no ens diu res si emparellables és fals.
- ***Condició de sortida:***
  Sortim quan emparellables és cert ja que per l'*INV3* això vol dir que els organismes ja s'han emparellat i per tant complim la *POST1*. (Gràcies a la *POST* de l'operació emparellar_organismes podem estar segurs que la complirem) (complim la PRE de emparellar_organismes ja que i < cand per *I'INV2 i I'INV3* i *cand < pobaux < pobmax* per *L'INV1 i I'INV2* i rank és l'associat al sistema per la *PRE*).

  També sortim quan *cand == pobaux* ja que hem de complir la primera condició de *I'INV3*.
- ***Cos del bucle:***
  Inicialment per l'*INV2* i la condició del bucle sabem que *cand < pobaux*, per tant tots els accessos a vectors són correctes (l'últim accés ho és ja que *i < cand* per l'*INV2*.
  Per complir l'*INV3* per l'iteració actual i suposant que és certa per l'anterior, l'únic que ens cal és comprovar si l'organisme *cand+1* no està ocupat, no és mort i no s'ha emparellat amb l'organisme *i+1* (condicions necessàries per l'emparellament que demana la segona condició de l'*INV3*). Això és el que fem, i assignem *emparellables* en conseqüència per complir la segona condició de l'*INV2*. Si *emparellables* és cert, la tercera condició ens demana que emparellem els organismes, que és el que fem. Finalment, l'únic que ens falta per acabar de complir l'*INV2* és incrementar *cand.*
- ***Finalització:***

  El bucle finalitza ja que decreix la distància entre *cand* i *pobaux* en cada iteració. *Funció de fita: pobaux – cand*. És un natural ja que per *I'INV3 cand <= pobaux* i decreix a cada iteració donat que incrementem *cand*.