UBA – Facultad de Ciencias Exactas y Naturales – Departamento de Computación– Algoritmos y Estructura de Datos I
Recuperatorio Parcial de Especificación– 28/06/2013

LU:
Apellidos:
Nombres:

Aclaraciones: El parcial NO es a libro abierto. Cualquier decisión de interpretación que se tome debe ser aclarada y justificada. Para aprobar se requieren al menos 60 puntos. **Entregar cada ejercicio en hoja separada.**
Importante: Para la resolución del parcial **NO** es necesario ni está permitido el uso de **acum**.

El pueblito de Rastis, es un pequeño pueblito rectangular cuya característica principal es que todas sus calles, o bien se orientan de Norte a Sur, o bien de Este a Oeste. Todas las calles que van de Norte a Sur son paralelas entre sí y perpendiculares a las que van de Este a Oeste. El Dr. Rastri, Intendente de dicho pueblo, decidió implementar un sistema revolucionario de seguridad: instalar cámaras en varios puntos del pueblo. Las cámaras instaladas realmente no son de muy buena calidad y suelen descomponerse periódicamente. Es por ello que se decidió registrar en qué intervalos de tiempo se encuentran activas. También decidió identificar a todos los automóviles del Pueblo (sus Patentes) que pasan por delante de las cámaras. Para modelar dicho escenario, se cuenta con la siguiente representación:

tipo Nombre=String;
tipo Calle=String;
tipo Cruce=(Calle, Calle);
tipo Intervalo=($\mathbb{Z}$, $\mathbb{Z}$);
tipo Patente=Int;

```
tipo Camara {
  observador filmacion (c :Camara) : [Intervalo];
  observador patentes (c :Camara) : [Patente];
  invariante positivos : (∀n ← filmacion(c))
    0 ≤ primero(n) < segundo(n);
  invariante noSeSolapan : (∀i, j ← [0..|filmacion(c)|), i ≠ j)
    ¬seSolapan(filmacion(c)_i, filmacion(c)_j);
  invariante intervalosOrdenados : ordenados(primeros(filmacion(c)));
  invariante patentesOrdenadas : ordenados(patentes(c));
  invariante soloSiHayFilmacion :
    |filmacion(c)| == 0 → |patentes(c)| == 0;
}

tipo Pueblo {
  observador nombre (p: Pueblo) : Nombre;
  observador autos (p: Pueblo) : [Patente];
  observador calles (p: Pueblo) : [Calle];
  observador sonParalelas (p :Pueblo, c1:Calle, c2:Calle) : Bool;
      requiere c1 ∈ calles(p) ∧ c2 ∈ calles(p);
  observador tieneCamara (p :Pueblo, c:Cruce) : Bool;
      requiere prm(c) ∈ calles(p) ∧ sgd(c) ∈ calles(p);
  observador camara (p :Pueblo, c:Cruce) : Camara;
      requiere prm(c) ∈ calles(p) ∧ sgd(c) ∈ calles(p);
      requiere tieneCamara(p, c);
  invariante sinCallesRepetidas : sinRepetidos(calles(p));
  invariante reflexiva : (∀c ← calles(p))sonParalelas(p, c, c);
  invariante simetrica : (∀c1, c2 ← calles(p))
    sonParalelas(p, c1, c2) → sonParalelas(p, c2, c1);
  invariante transitiva : (∀c1, c2, c3 ← calles(p))
    sonParalelas(p, c1, c2) ∧ sonParalelas(p, c2, c3) →
    sonParalelas(p, c1, c3);
  invariante alMenosUnCruce :
    (∃c1, c2 ← calles(p))sonPerpendiculares(p, c1, c2);
  invariante noHay3oMasPerpendicularesEntreSi :
    ¬(∃c1, c2, c3 ← calles(p))sonPerpendiculares(p, c1, c2) ∧
    sonPerpendiculares(p, c2, c3) ∧ sonPerpendiculares(p, c1, c3);
  invariante paraCamaraTieneQueSerCruce : (∀c1, c2 ← calles(p))
    sonParalelas(p, c1, c2) → ¬tieneCamara(p, (c1, c2));
  invariante mismaCamara1 : (∀c1, c2 ← calles(p))
    tieneCamara(p, (c1, c2)) == tieneCamara(p, (c2, c1));
  invariante mismaCamara2 : (∀c1, c2 ← calles(p))
    tieneCamara(p, (c1, c2)) →
    camara(p, (c1, c2)) == camara(p, (c2, c1));
  invariante sinAutosRepetidos : sinRepetidos(autos(p));
  invariante soloAutosDelPueblo : ...;
}
```

```
aux ordenados (l : [T]) : Bool = (∀i, j ← [0..|l|), i < j) l_i < l_j;
aux primeros (l : [(S,T)]) : [S] = [prm(x)|x ← l];
aux seSolapan (n1: Intervalo,n2: Intervalo) : Bool =
  primero(n1) ≤ primero(n2) ≤ segundo(n1) ∨
  primero(n1) ≤ segundo(n2) ≤ segundo(n1) ∨
  primero(n2) ≤ primero(n1) ≤ segundo(n1) ≤ segundo(n2);
aux sinRepetidos (l : [T]) : Bool = (∀i ← [0..|l|)) l_i ∉ l(i..|l|);
aux sonPerpendiculares (p:Pueblo, c1:Calle, c2:Calle) : Bool = ¬sonParalelas(p, c1, c2);
```

Aclaración: Los intervalos son cerrados. Es decir, si filmacion(c) = (1,3), en los momentos 1, 2 y 3, la camara está filmando.

Ejercicio 1. [30 puntos]

a) **[10 p.]** Completar el invariante *soloAutosDelPueblo* del tipo Pueblo, que garantiza que la patentes filmadas por las cámaras sólo pertenecen a autos del pueblo.

b) **[10 p.]** Especificar el aux *camarones*(p : *Pueblo*, a : *Patente*) : [*Cruce*] que devuelve la lista de Cruces en donde fue filmado el auto a. La lista resultante no tiene que tener repetidos. Si el auto fue filmado en el cruce entre las calles A y B, entonces tiene que estar tanto el cruce (A,B) como el (B,A).

c) **[10 p.]** Especificar el aux *autosX*(p : *Pueblo*) : [*Patente*] que devuelve la lista de patentes de los autos del pueblo p que no han sido filmados por ninguna cámara.

Ejercicio 2. [20 puntos] Especificar el `problema registroNacionalDeInfractores (ps: [Pueblo]) = result : [Patente]` . Este problema, dada una lista de pueblos *ps*, devuelve aquellos autos que: a) pertenecen a la vez a TODOS los pueblos de *ps* Y b) que han sido filmados por al menos una cámara (sin importar de qué pueblo). La lista resultante no tiene que tener repetidos.

Ejercicio 3. [20 puntos] Especificar el `problema autoMasFilmado (p: Pueblo) = result : Patente` . Este problema, dado un Pueblo p, devuelve el auto filmado por más cámaras de dicho pueblo.

Ejercicio 4. [30 puntos] Especificar el `problema automatico (p: Pueblo, a:Patente, c:Cruce)` . Un habitante del pueblo adquiere un automóvil cuya patente es a y se desea incorporarlo al pueblo p. Inicialmente el automóvil se encuentra en el cruce c, por lo tanto si en ese cruce hay una cámara y la misma tiene algún momento filmado, el automóvil será incorporado al registro de dicha cámara.