Algorítmos y Estructuras de Datos II
Trabajo Práctico 1 (Especificación)
Grupo 6

Bayardo, Julián
julian@bayardo.com.ar
850/13

Cuneo, Christian
chriscuneo93@gmail.com
755/13

Gambaccini, Ezequiel
ezequiel.gambaccini@gmail.com
715/13

Lebrero Rial, Ignacio Manuel
ignaciolebrero@gmail.com
751/13

9 de Septiembre del 2014

1. Aclaraciones

Asumimos que las estaciones, junto con las sendas y las restricciones entre ellas no son mutables: una instancia del TAD Ciudad comienza con un mapa definido y no puede ser cambiado. Suponemos también que las estaciones pueden tener sendas que las conecten a sí mismas, mas sólo una senda que la conecte a otra estación definida. Aparte, consideramos que pueden haber estaciones no conexas, y en cuyo caso no son consideradas bloqueantes (aunque los robots en ellas no podrían moverse hacia ningún lado).

Los RUR se suponen asignados al momento de agregar un robot, la unica restricción al respecto es que el RUR efectivamente sea único. No puede existir un robot que no tenga ninguna característica.

Dos restricciones son equivalentes cuando su tabla de verdad es la misma para todo posible conjunto de características.

Cabe destacar que no modelamos el paso del tiempo, por lo que las inspecciones no son automáticas, sino que deben ser causadas deliberadamente por quien maneje el sistema.

2. Sobre el formato

Inicialmente escribimos la especificación utilizando UTF-8 en lugar de LaTeX, pensando que podríamos fácilmente incluir el archivo en nuestro informe, mas esto resultó ser problemático por varias cuestiones. Finalmente, optamos por incluirlo por separado como un archivo en otro pdf. Debido a problemas con esta solución, tuvimos que ajustar el texto para que no pase de 90 columnas; adoptamos como convención que si las precondiciones de una función superan tal tamaño, la guarda se pasa a otra linea y se indenta.

```
TAD caracteristica ES string

TAD Restriccion
  generos restriccion
  exporta AND, OR, NOT, VAR, cumple, FALSE, TRUE
  igualdad observacional
    (∀r₁, r₂ : restriccion) (r₁ =obs r₂ ⟺
      (∀c : conj(caracteristica)) (cumple(c, r₁) =obs cumple(c, r₂)))
  observadores basicos
    cumple: conj(caracteristica) × restriccion → bool
  generadores
    AND: restriccion × restriccion → restriccion
    OR: restriccion × restriccion → restriccion
    NOT: restriccion → restriccion
    VAR: caracteristica → restriccion
  otras operaciones
    TRUE: → restriccion
    FALSE: → restriccion
  axiomas (∀cs : conj(caracteristica)) (∀c1, c2 : restriccion) (∀c : caracteristica)
    cumple(cs, VAR(c)) ≡ c ∈ cs
    cumple(cs, NOT(c1)) ≡ ¬(cumple(cs, c1))
    cumple(cs, AND(c1, c2)) ≡ cumple(cs, c1) ∧ cumple(cs, c2)
    cumple(cs, OR(c1, c2)) ≡ cumple(cs, c1) ∨ cumple(cs, c2)

    TRUE ≡ OR(VAR("dummy"), NOT(VAR("dummy")))

    FALSE ≡ AND(VAR("dummy"), NOT(VAR("dummy")))
Fin TAD

TAD estacion ES string
TAD conexion ES ⟨estacion, restriccion⟩
TAD conexiones ES conj(conexion)

TAD Mapa
  generos mapa
  exporta estaciones, conexiones, nuevo, crearEst, conectar,
          esBloqueante, conectadas, camino
  igualdad observacional
    (∀m₁, m₂ : mapa) (m₁ =obs m₂ ⟺
      estaciones(m₁) =obs estaciones(m₂) ∧ₗ
      (∀e ∈ estaciones(m₁)) (conexiones(m₁, e) =obs conexiones(m₂, e)))
  observadores basicos
    estaciones : mapa → conj(estacion)
    conexiones : mapa m × estacion e → conj(conexion) {e ∈ estaciones(m)}
  generadores
    nuevo    : → mapa
    crearEst : mapa m × estacion a → mapa {¬(a ∈ estaciones(m))}
    conectar : mapa m × estacion a × estacion b × restriccion → mapa
               {a, b ∈ estaciones(m) ∧ₗ ¬conectadas(m, a, b)}
  otras operaciones
    esBloqueante  : mapa m × conj(caracteristica) × estacion e → bool {e ∈ estaciones(m)}
    esBloqueante' : conj(conexion) × conj(caracteristica) → bool
    conectadas    : mapa m × estacion a × estacion b → bool {a, b ∈ estaciones(m)}
    conectadas'   : estacion × conj(conexion) → bool
    camino        : mapa m × estacion a × estacion b → restriccion
      {a, b ∈ estaciones(m) ∧ conectadas(m, a, b)}
    camino'       : estacion e × conj(conexion) cc → restriccion
      {(∃x ∈ cc) (π₁(x) =obs e)}
  axiomas (∀m : mapa) (∀e, a, b : estacion) (∀r : restriccion) (∀cs : conj(caracteristica))
          (∀cc : conj(conexion))
```

```
    conexiones(crearEst(m, e), k) ≡ ϕ
    conexiones(conectar(m, a, b, r), e) ≡
      (if e =obs a then ⟨b, r⟩
      else if e =obs b then ⟨a, r⟩ else ϕ fi
      fi) ∪ conexiones(m, e)

    estaciones(nuevo) ≡ ϕ
    estaciones(crearEst(m, e)) ≡ { e } ∪ estaciones(m)
    estaciones(conectar(m, a, b)) ≡ estaciones(m)

    esBloqueante(m, cs, e) ≡
      if ϕ?(conexiones(m, e)) then False
      else esBloqueante'(conexiones(m, e), cs) fi

    esBloqueante'(cc, cs) ≡
      if ϕ?(cc) then True
      else ¬(cumple(r, π2(dameUno(cc)))) ∧l esBloqueante'(sinUno(cc), cs) fi

    conectadas(m, a, b) ≡ conectadas'(b, conexiones(m, a))

    conectadas'(a, cc) ≡
      if ϕ?(cc) then False
      else (π1(dameUno(cc)) =obs a) vl conectadas'(a, sinUno(cc)) fi

    camino(m, a, b) ≡ camino'(b, conexiones(m, a))

    camino'(a, cc) ≡
      if π1(dameUno(cc)) =obs a then π2(dameUno(cc))
      else caminos'(a, sinUno(cc)) fi
Fin TAD

TAD rur ES nat
TAD robot ES ⟨rur, conj(caracteristica)⟩

TAD Ciudad
  generos ciudad
  exporta mapeo, robots, posicion, #infracciones, #inspecciones, nueva, agregar,
          mover, inspeccion, caracteristicaMasInfractora

  igualdad observacional
  (∀c1, c2 : ciudad) (c1 =obs c2 ⟺
    mapeo(c1) =obs mapeo(c2) ∧ robots(c1) =obs robots(c2) ∧l
    ((∀e ∈ estaciones(mapeo(c))) (#inspecciones(c1, e) =obs #inspecciones(c2, e))) ∧
    ((∀r ∈ robots(c1)) (
      #infracciones(c1, π1(r)) =obs #infracciones(c2, π1(r)) ∧
      posicion(c1, π1(r)) =obs posicion(c2, π1(r)))))

  observadores basicos
    mapeo: ciudad → mapa
    robots: ciudad → conj(robot)
    posicion: ciudad c × rur i → estacion {(∃r ∈ robots(c)) π1(r) =obs i}
    #infracciones: ciudad c × rur i → nat {(∃r ∈ robots(c)) π1(r) =obs i}
    #inspecciones: ciudad c × estacion e → nat {e ∈ estaciones(mapeo(c))}
  generadores
    nueva: mapa → ciudad

    agregar: ciudad c x rur i × conj(caracteristica) cs × estacion e → ciudad
      {¬ϕ?(cs) ∧ ¬((∃r ∈ robots(c)) π1(r) =obs i) ∧ e ∈ estaciones(mapeo(c)) ∧l
      ¬esBloqueante(mapeo(c), cs, e)}

    mover: ciudad c × rur i × estacion e → ciudad
```

```
    {e ∈ estaciones(mapeo(c)) ∧ ((∃r ∈ robots(c)) π₁(r) =obs i) ∧ₗ
    conectadas(mapeo(c), posicion(c, i), e)}

  inspeccion: ciudad c × estacion e → ciudad {e ∈ estaciones(mapeo(c))}
otras operaciones

  masInfracciones: ciudad c × estacion e → conj(robot) {e ∈ estaciones(mapeo(c))}
  robotsEnEstacion: ciudad c × conj(robot) cr × estacion → conj(robot) {cr ⊆ robots(c)}
  maxInfractor: ciudad c × conj(robot) cr × robot r → robot
    {r ∈ robots(c) ∧ cr ⊆ robots(c) ∧ ¬ϕ?(cr)}

  caracteristicaMasInfractora: ciudad c → caracteristica
    {(∃r ∈ robots(c)) (#infracciones(π₁(r)) > 0)}
  infractoras: ciudad c × conj(robot) cr → multiconj(caracteristica) {cr ⊆ robots(c)}
  aMulti: conj(caracteristica) → multiconj(caracteristica)
  maximizar: multiconj(caracteristica) mc → caracteristica {¬ϕ?(mc)}
  agregarN: nat × conj(caracteristica) → multiconj(caracteristica)
axiomas (∀m : mapa) (∀c : ciudad) (∀i : rur) (∀cc : conj(caracteristicas))
        (∀e, e' : estacion) (∀cr : conj(robot)) (∀mc : multiconj(caracteristica))
  mapeo(nueva(m)) ≡ m
  mapeo(agregar(c, i, cc, e)) ≡ mapeo(c)
  mapeo(mover(c, i, e)) ≡ mapeo(c)
  mapeo(inspeccion(c, e)) ≡ mapeo(c)

  robots(nueva) ≡ ϕ
  robots(agregar(c, i, cc, e)) ≡ Ag(⟨i, cc⟩, robots(c))
  robots(mover(c, i, e)) ≡ robots(c)
  robots(inspeccion(c, e)) ≡ robots(c) - masInfracciones(c, e)

  posicion(agregar(c, i, cc, e), r) ≡
    if i =obs r then e
    else posicion(c, r) fi
  posicion(mover(c, i, e), r) ≡
    if i =obs r then e
    else posicion(c, r) fi
  posicion(inspeccion(c, e), r) ≡ posicion(c, r)

  #infracciones(agregar(c, i, cc, e), r) ≡
    if i =obs r then 0
    else #infracciones(c) fi

  #infracciones(inspeccion(c, e), r) ≡ #infracciones(c, r)

  #infracciones(mover(c, i, e), r) ≡
    if r =obs i then
      #infracciones(c, r) + β(¬cumple(camino(mapeo(c), posicion(c, r), e)))
    else #infracciones(c, r) fi

  #inspecciones(nueva, e) ≡ 0
  #inspecciones(agregar(c, i, cc, e'), e) ≡ #inspecciones(c)
  #inspecciones(mover(c, i, e'), e) ≡ #inspecciones(c)
  #inspecciones(inspeccion(c, e'), e) ≡ β(e' =obs e) + #inspecciones(c, e)

  masInfracciones(c, e) ≡
    if ¬ϕ?(robotsEnEstacion(c, robots(c), e)) ∧ₗ
      #infracciones(c, π₁(maxInfractor(c, robotsEnEstacion(c, robots(c), e)))) > 0
    then Ag(maxInfractor(c, robotsEnEstacion(c, robots(c), e)), ϕ)
    else ϕ fi

  maxInfractor(c, cr) ≡
    if ϕ?(sinUno(cr)) vₗ #infracciones(c, π₁(dameUno(cr))) >
```

```
                          #infracciones(c, π₁(maxInfractor(c, sinUno(cr))))
      then dameUno(cr)
      else maxInfractor(c, sinUno(cr)) fi

    robotsEnEstacion(c, cr, e) ≡
      if ϕ?(cr) then ϕ
      else
        if posicion(c, π₁(dameUno(cr))) =obs e then
          Ag(dameUno(cr), robotsEnEstacion(c, sinUno(cr), e))
        else robotsEnEstacion(c, sinUno(cr), e) fi
      fi

    caracteristicaMasInfractora ≡ maximizar(infractoras(c, robots(c)))

    infractoras(c, cr) ≡
      if ϕ?(cr) then ϕ
      else
        infractoras(c, sinUno(cr)) ∪
        if #infracciones(c, dameUno(cr)) > 0 then
            agregarN(#infracciones(c, dameUno(cr)), π₂(dameUno(cr)))
        else ϕ fi
      fi

    aMulti(cs) ≡ if ϕ?(cs) then ϕ else Ag(dameUno(cs), aMulti(sinUno(cs))) fi

    agregarN(i, cs) ≡ if i =obs 0 then ϕ else aMulti(cs) ∪ agregarN(i-1, cs) fi

    maximizar(mc) ≡
      if ϕ?(sinUno(mc)) vₗ #(dameUno(mc), mc) > #(maximizar(sinUno(mc)), mc) then
        dameUno(mc)
      else maximizar(sinUno(mc)) fi
Fin TAD
```