



TP de Especificación

Sudoku

21 de Abril de 2017

Algoritmos y Estructuras de Datos I

Grupo 17

Integrante	LU	Correo electrónico
Maqueda, Ignacio	279/14	ignaciomaqueda95@gmail.com
Parral, Guillermo	280/16	guillermoeparral@gmail.com
Quintela, Gonzalo	089/16	gquintela@dc.uba.ar
Sirio, Tomás	440/16	tomassirio@gmail.com



Facultad de Ciencias Exactas y Naturales
Universidad de Buenos Aires

Ciudad Universitaria - (Pabellón I/Planta Baja)

Intendente Güiraldes 2610 - C1428EGA

Ciudad Autónoma de Buenos Aires - Rep. Argentina

Tel/Fax: (+54 +11) 4576-3300

<http://www.exactas.uba.ar>

1. Problemas

```

proc sudoku_esTableroValido (in t: seq⟨seq⟨ℤ⟩⟩, out result: Bool) {
  Pre {True}
  Post {result = esTableroValido(t)}
}

proc sudoku_esCeldaVacía (in t: seq⟨seq⟨ℤ⟩⟩, in f: ℤ, in c: ℤ, out result: Bool) {
  Pre {esTableroValido(t) ∧L 0 ≤ f, c < |t|}
  Post {result = (t[f][c] = 0)}
}

proc sudoku_nroDeCeldasVacías (in t: seq⟨seq⟨ℤ⟩⟩, out result: ℤ) {
  Pre {esTableroValido(t)}
  Post {∑i=0|t|-1 (∑j=0|t|-1 if t[i][j] = 0 then 1 else 0 fi)}
}

proc sudoku_primeraCeldaVacíaFila (in t: seq⟨seq⟨ℤ⟩⟩, out result: ℤ) {
  Pre {esTableroValido(t)}
  Post {if (∃i : ℤ)(0 ≤ i < |t| ∧L filaTieneCeldaVacía(t[i]) ∧L (∀j : ℤ)
    (0 ≤ j < i →L ¬filaTieneCeldaVacía(t[j])))
    then result = i
    else result = -1 fi}
}

proc sudoku_primeraCeldaVacíaColumna (in t: seq⟨seq⟨ℤ⟩⟩, out result: ℤ) {
  Pre {esTableroValido(t)}
  Post {if (∃i : ℤ)(0 ≤ i < |t| ∧L filaTieneCeldaVacía(t[i]) ∧L (∀j : ℤ)(0 ≤ j < i →L ¬filaTieneCeldaVacía(t[j])))
    then result = indicePrimeraCeldaVacíaEnFila(t[i])
    else result = -1 fi}
}

proc sudoku_valorEnCelda (in t: seq⟨seq⟨ℤ⟩⟩, in f: ℤ, in c: ℤ, out result: ℤ) {
  Pre {esTableroValido(t) ∧L 0 ≤ f, c ≤ 8 ∧L t[f][c] ≠ 0}
  Post {result = t[f][c]}
}

proc sudoku_llenarCelda (inout t: seq⟨seq⟨ℤ⟩⟩, in f: ℤ, in c: ℤ, in value: ℤ) {
  Pre {esTableroValido(t) ∧L 0 ≤ f, c ≤ 8 ∧L t[f][c] = 0 ∧L 1 ≤ value ≤ 9 ∧L t = t0}
  Post {t[f][c] = value ∧L (∀i : ℤ)(∀j : ℤ)((0 ≤ i, j < |t| ∧L (i ≠ f ∨ j ≠ c)) →L t[i][j] = t0[i][j])}
}

proc sudoku_vaciarCelda (inout t: seq⟨seq⟨ℤ⟩⟩, in f: ℤ, in c: ℤ) {
  Pre {esTableroValido(t) ∧L 0 ≤ f, c ≤ 8 ∧L t[f][c] ≠ 0 ∧L t = t0}
  Post {t[f][c] = 0 ∧L (∀i : ℤ)(∀j : ℤ)((0 ≤ i, j < |t| ∧L (i ≠ f ∨ j ≠ c)) →L t[i][j] = t0[i][j])}
}

proc sudoku_esTableroParcialmenteResuelto (in t: seq⟨seq⟨ℤ⟩⟩, out result: Bool) {
  Pre {True}
  Post {result = esTableroParcialmenteResuelto(t)}
}

```

```

proc sudoku_esTableroTotalmenteResuelto (in t: seq⟨seq⟨ℤ⟩⟩, out result: Bool) {
  Pre {True}
  Post {result = esTableroTotalmenteResuelto(t)}
}

proc sudoku_esSubTablero (in t0, t1 : seq⟨seq⟨ℤ⟩⟩, out result : Bool){
  Pre {True}
  Post {result = esSubTablero(t0, t1)}
}

proc sudoku_tieneSolucion (in t: seq⟨seq⟨ℤ⟩⟩, out tieneSolucion: Bool) {
  Pre {esTableroValido(t)}
  Post {tieneSolucion = (∃s : seq⟨seq⟨ℤ⟩⟩)(esTableroTotalmenteResuelto(s) ∧L esSubTablero(s, t))}
}

proc sudoku_resolver (inout t: seq⟨seq⟨ℤ⟩⟩, out tieneSolucion: Bool) {
  Pre {esTableroValido(t) ∧L t = t0}
  Post {if (∃s : seq⟨seq⟨ℤ⟩⟩)(esTableroTotalmenteResuelto(s) ∧L esSubTablero(s, t))
    then tieneSolucion = True ∧ t = s
    else tieneSolucion = False fi}
}

proc sudoku_copiarTablero (in src: seq⟨seq⟨ℤ⟩⟩, out target: seq⟨seq⟨ℤ⟩⟩) {
  Pre {esTableroValido(src)}
  Post {esTableroValido(target) ∧L (∀i : ℤ)(∀j : ℤ)(0 ≤ i, j < |src| →L target[i][j] = src[i][j])}
}

```

2. Predicados y Auxiliares generales

```

pred esMatriz (t: seq⟨seq⟨ℤ⟩⟩) {
  (∀i : ℤ)(∀j : ℤ)(0 ≤ i, j < |t| →L |t[i]| = |t[j]|)
}

pred esMatrizCuadrada (t: seq⟨seq⟨ℤ⟩⟩) {
  esMatriz(t) ∧ (cantidadFilas(t) = cantidadColumnas(t))
}

pred esTableroValido (t: seq⟨seq⟨ℤ⟩⟩) {esMatrizCuadrada(t) ∧L |t| = 9 ∧L
  (∀i : ℤ)(∀j : ℤ)(0 ≤ i, j < |t| →L 0 ≤ t[i][j] ≤ 9)}
}

pred filaTieneCeldaVacía (f: seq⟨ℤ⟩) {
  (∃i : ℤ)(0 ≤ i < |f| ∧L f[i] = 0)
}

pred noHayRepetidosEnRegion (t: seq⟨seq⟨ℤ⟩⟩) {(∀i : ℤ)(∀j : ℤ)(∀k : ℤ)(∀l : ℤ)
  (0 ≤ i, j, k, l < 9 ∧L (i div 3 = k div 3) ∧L (j div 3 = l div 3) ∧L (i ≠ k ∨ j ≠ l) →L (t[i][j] = 0 ∨ t[k][l] = 0 ∨ t[i][j] ≠ t[k][l]))}
}

pred noHayRepetidosEnFila (s: seq⟨ℤ⟩) {(∀i : ℤ)(∀j : ℤ)(0 ≤ i, j < |s| ∧L j ≠ i →L (s[i] = 0 ∨ s[j] = 0 ∨ s[i] ≠ s[j]))}
}

pred noHayRepetidosEnColumna (t: seq⟨seq⟨ℤ⟩⟩) {(∀j : ℤ)(0 ≤ j < |t| →L (∀l : ℤ)(∀k : ℤ)
  (0 ≤ l, k < |t| ∧L l ≠ k →L (t[l][j] = 0 ∨ t[k][j] = 0 ∨ t[l][j] ≠ t[k][j])))}
}

```

```

    pred esTableroParcialmenteResuelto (t: seq<seq<ℤ>>) {esTableroValido(t) ∧L (∀i : ℤ)(0 ≤ i < |t| →L
noHayRepetidosEnFila(t[i])) ∧L noHayRepetidosEnColumna(t) ∧L noHayRepetidosEnRegion(t)
}

```

```

    pred esTableroTotalmenteResuelto (t: seq<seq<ℤ>>) {esTableroParcialmenteResuelto(t) ∧L (∀i : ℤ)(0 ≤ i < |t| →L
¬filaTieneCeldaVacía(t[i]))
}

```

```

    pred esSubTablero (t0, t1 : seq<seq<ℤ>>){esTableroValido(t0) ∧L esTableroValido(t1) ∧L (∀i : ℤ)(∀j : ℤ)
((0 ≤ i < |t0| ∧L t0[i][j] ≠ 0) →L t0[i][j] = t1[i][j])
}
    fun cantidadFilas (t: seq<seq<ℤ>>) : ℤ = |t|;

```

```

    fun cantidadColumnas (t: seq<seq<ℤ>>) : ℤ = if cantidadFilas(t) > 0 then |t[0]| else 0 fi;

```

```

    fun indicePrimeraCeldaVacíaEnFila (s: seq<ℤ>) : ℤ = if ((∃i : ℤ)(0 ≤ i < |s| ∧L s[i] = 0 ∧L
(∀j : ℤ)(0 ≤ j < i →L s[j] ≠ 0))) then i else -1 fi;

```

3. Decisiones tomadas