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

1. Problemas

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

```
proc sudoku_esCeldaVacia (in t: seq⟨seq⟨ℤ⟩⟩, in f: ℤ,in c: ℤ, out result: Bool)  {
       Pre {esTableroV alido(t) ∧_L 0 ≤ f, c < |t|}
       Post {result = (t[f][c] = 0)}
}
```

```
proc sudoku_nroDeCeldasVacias (in t: seq⟨seq⟨ℤ⟩⟩, out result: ℤ)  {
       Pre {esTableroV alido(t)}
       Post {∑_{i=0}^{|t|−1} (∑_{j=0}^{|t|−1} if t[i][j] = 0 then 1 else 0 fi)}
}
```

```
proc sudoku_primeraCeldaVaciaFila (in t: seq⟨seq⟨ℤ⟩⟩, out result: ℤ)  {
       Pre {esTableroV alido(t)}
       Post {if (∃i : ℤ)(0 ≤ i < |t| ∧_L filaTieneCeldaV acia(t[i]) ∧_L (∀j : ℤ)
          (0 ≤ j < i ⟶_L ¬filaTieneCeldaV acia(t[j])))
          then result = i
          else result = −1 fi}
}
```

```
proc sudoku_primeraCeldaVaciaColumna (in t: seq⟨seq⟨ℤ⟩⟩, out result: ℤ)  {
       Pre {esTableroV alido(t)}
       Post {if (∃i : ℤ)(0 ≤ i < |t| ∧_L filaTieneCeldaV acia(t[i]) ∧_L (∀j : ℤ)(0 ≤ j < i ⟶_L ¬filaTieneCeldaV acia(t[j])))
          then result = indicePrimeraCeldaV aciaEnFila(t[i])
          else result = −1 fi}
}
```

```
proc sudoku_valorEnCelda (in t: seq⟨seq⟨ℤ⟩⟩, in f: ℤ, in c: ℤ, out result: ℤ)  {
       Pre {esTableroV alido(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 {esTableroV alido(t) ∧_L 0 ≤ f, c ≤ 8 ∧_L t[f][c] = 0 ∧_L 1 ≤ value ≤ 9 ∧_L t = t_0}
       Post {t[f][c] = value ∧_L (∀i : ℤ)(∀j : ℤ)((0 ≤ i, j < |t| ∧_L (i ≠ f ∨ j ≠ c)) ⟶_L t[i][j] = t_0[i][j])}
}
```

```
proc sudoku_vaciarCelda (inout t: seq⟨seq⟨ℤ⟩⟩, in f: ℤ, in c: ℤ)  {
       Pre {esTableroV alido(t) ∧_L 0 ≤ f, c ≤ 8 ∧_L t[f][c] ≠ 0 ∧_L t = t_0}
       Post {t[f][c] = 0 ∧_L (∀i : ℤ)(∀j : ℤ)((0 ≤ i, j < |t| ∧_L (i ≠ f ∨ j ≠ c)) ⟶_L t[i][j] = t_0[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])}
}
```

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


proc sudoku_esSubTablero (in t_0, t_1 : seq⟨seq⟨Z⟩⟩, out result : Bool){
      Pre {True}
      Post {result = esSubTablero(t_0, t_1)}
}


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 = t_0}
      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]|)
}

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 ;

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

fun indicePrimeraCeldaVaciaEnFila (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 ;

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
¬filaTieneCeldaVacia(t[i]))
}
```

```
    pred esSubTablero (t_0, t_1 : seq⟨seq⟨ℤ⟩⟩){esTableroValido(t_0) ∧_L esTableroValido(t_1) ∧_L (∀i : ℤ)(∀j : ℤ)
((0 ≤ i|t| ∧_L t_0[i][j] ≠ 0) ⟶_L t_0[i][j] = t_1[i][j])
}
```

3. Decisiones tomadas