Correctitud en TADs

Repaso

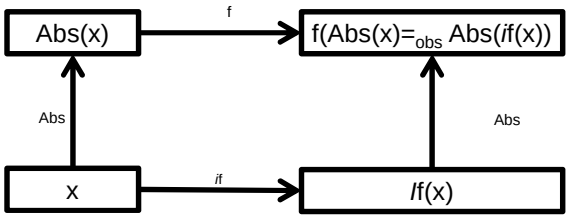
- Un TAD define el qué. Tiene estado y operaciones descriptas mediante pre y post condición, en lógica

- Una implementación define el cómo. Tiene un estado, invariante de representación, función de abstracción y algoritmos para las operaciones

- Un TAD puede tener muchas implementaciones, según los requerimientos y el contexto de uso (por ejemplo de eficiencia)

Verificación

- ¿Podemos demostrar que la implementación de un TAD es correcta respecto de la especificación del TAD? ¡Por supuesto que sí!
- Para toda operación *i*f que implementa una operación del TAD y toda x instancia de representación que cumple el invariante de representación, debemos ver que el siguiente diagrama conmuta:



- Para cada operación, hay que demostrar que:
  - Conserva el invariante
  - El algoritmo respeta la pre y postcondición del TAD
- Spoiler: vamos a tener que viajar entre los mundos de la implementación y el del TAD usando la función de abstracción...

Conservación del invariante - Ejemplo

TAD punto / Operación `rotar`

```
TAD Punto {
  obs x: float
  obs y: float

proc rotar(p: Punto, d: float)
  requiere true
  asegura p.x == auxRho(old(p))*cos(auxTheta(old(p))+d)
  asegura p.y == auxRho(old(p)) * sin(auxTheta(old(p))+d)

aux auxTheta(p: Punto): float {
  if p.x == 0 then pi/2 sign(p.y) else arctan(p.y/p.x)}

aux auxRho(p: Punto): float {
  sqrt(p.x ** 2 + p.y ** 2)}
}
```

```
Impl PuntoImpl {
              var rho: float
              var theta float

       pred InvRep(p': PuntoImpl) {
              0 <= p'.theta < 2*pi
       }

              aux FuncAbs(p': PuntoImpl): Punto {
                            Punto p |
                     p.x == p'.rho * cos(p'.theta) &&
                      p.y == p'.rho * sin(p'.theta)
       }

       proc rotar(p': PuntoImpl, d: float) {
              p'.theta += d;
}
```

Conservación del invariante

```
InvRep(p')

      wp(código, InvRep(p'))

      p'.theta += d;
InvRep(p')
```

Tenemos que demostrar que
`InvRep(p') ==> wp(código, InvRep(p'))`

Conservación del invariante

```
0 <= p'.theta < 2*pi
==> ??

      0 <= p'.theta+d < 2*pi

      p'.theta += d;

0 <= p'.theta < 2*pi
==
InvRep(p')
```

No es verdad que

```
0 <= p'.theta < 2*pi
==>
0 <= p'.theta+d < 2*pi
```

¡Oops! Llegamos a que, como precondición, tiene que suceder que el valor del ángulo **más** el parámetro de entrada esté en rango ¿Qué hacemos?

- ¿Corregimos la especificación?
- ¿Corregimos el invariante?
- ¿Corregimos el algoritmo?

Conservación del invariante

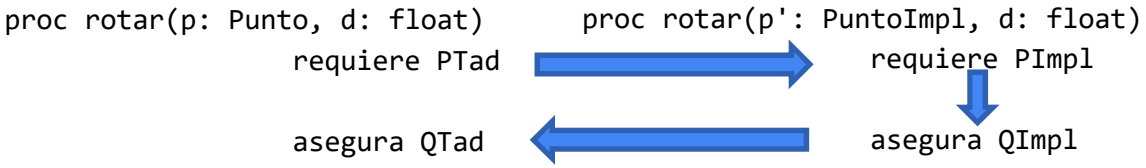
¡El algoritmo!

```
true
      ==
0 <= (p'.theta+d)%(2*pi) < 2*pi
                  p'.theta += d;
0 <= p'.theta%(2*pi) < 2*pi
                  p'.theta %= 2*pi;
0 <= p'.theta < 2*pi
==
InvRep(p')
```

Ahora sí!

`InvRep(p') ==> true`

Correctitud del algoritmo



- Dado un punto cualquiera p y un PuntoImpl p' tales que

```
p == FuncAbs(p')
```

- tenemos que probar que

```
(Ptad(p, d)) ==> Pimpl(p', d) (1)
(InvRep(p') && Qimpl(p', d)) ==> Qtad(p, d) (2)
```

- Lo que, sumado a la tripla de Hoare

```
{InvRep(p') && Pimpl(p', d)} Código del rotarimpl {Qimpl(p', d)} (3)
```

Completa la prueba

- Análogamente, dado un punto cualquiera p y un PuntoImpl p' tales que

```
p == FuncAbs(p')
```

- Tenemos que probar que

```
PTad(p, d) ==> wp(código, Qtad(p, d))
```

Correctitud del algoritmo

- Recordemos

```
Ptad = true
QTad = p.x == auxRho(old(p)) * cos(auxTheta(old(p)+d) && p.y == auxRho(old(p)) * sin(auxTheta(old(p)+d)
Pimpl = true
Qimpl = p'.rho == old(p').rho &&  p'.theta == old(p').theta + d
Código: p'.theta = (p'.theta + d) % (2 * pi)
```

- Entonces

```
(1) (PTad(p) => Pimpl(p', d)) == (True ==> True)

(3) Pimpl {S} Qimpl
wp(p'.theta = (p'.theta + d) % 2pi, Qimpl) =
p'.rho == old(p').rho && (p'.theta + d)%2pi = (old(p').theta + d) % 2pi
¿Pimpl => wp(S, qimpl)?
Sí, pues al inicio p' == old(p')
```

Correctitud del algoritmo

- Falta

```
(2) Qimpl(p') => Qtad(p)

Qimpl = p'.rho == old(p').rho &&  p'.theta == old(p').theta + d
QTad = p.x == auxRho(old(p))*cos(auxTheta(old(p)+d) && p.y == auxRho(old(p))*sin(auxTheta(old(p)+d)

Si p = FuncAbs(p'), entonces vale que:
auxRho(p) == p'.rho
auxTheta(p) == p'.theta

Si reemplazamos en Qtad:
Qtad = p.x == old(p').rho*cos(old(p').theta+d) && p.y == old(p').rho*sin(old(p').theta+d)

Asumimos Qimpl cierto, queremos ver que Qtad es cierto.
Reemplazando en Qtad:
QTad =
p.x == p'.rho*cos(old(p').theta+d) && p.y == p'.rho*sin(old(p').theta+d) ==
p.x == p'.rho*cos(p'.theta) && p.y == p'.rho*sin(p'.theta)
esto vale pues p = FuncAbs(p')
```