Lenguajes formales de especificación

Introducción

- Uso de los métodos formales
  - Especificación
    - Intención de un algoritmo de forma breve y precisa
  - Derivación
    - Deducir formalmente las instrucciones que debe realizar el algoritmo
  - Verificación
    - Verificar su corrección

Introducción

- Objetivo especificación formal
  - Expresar de forma correcta y sin ambigüedades
    - Qué es lo que debe de hacer un algoritmo
    - Bajo que condiciones se puede ejecutar

- Ejemplo
  - *Escribir un algoritmo que realice la división de un entero **a** por otro entero **b**, distinto de cero, devolviendo el cociente entero por defecto en **q** y el resto en* **r**.
  - *{Q≡b≠0}*

    fun *divisionEntera(a,b:* **entero***)* **dev***(q,r:* **entero***)*

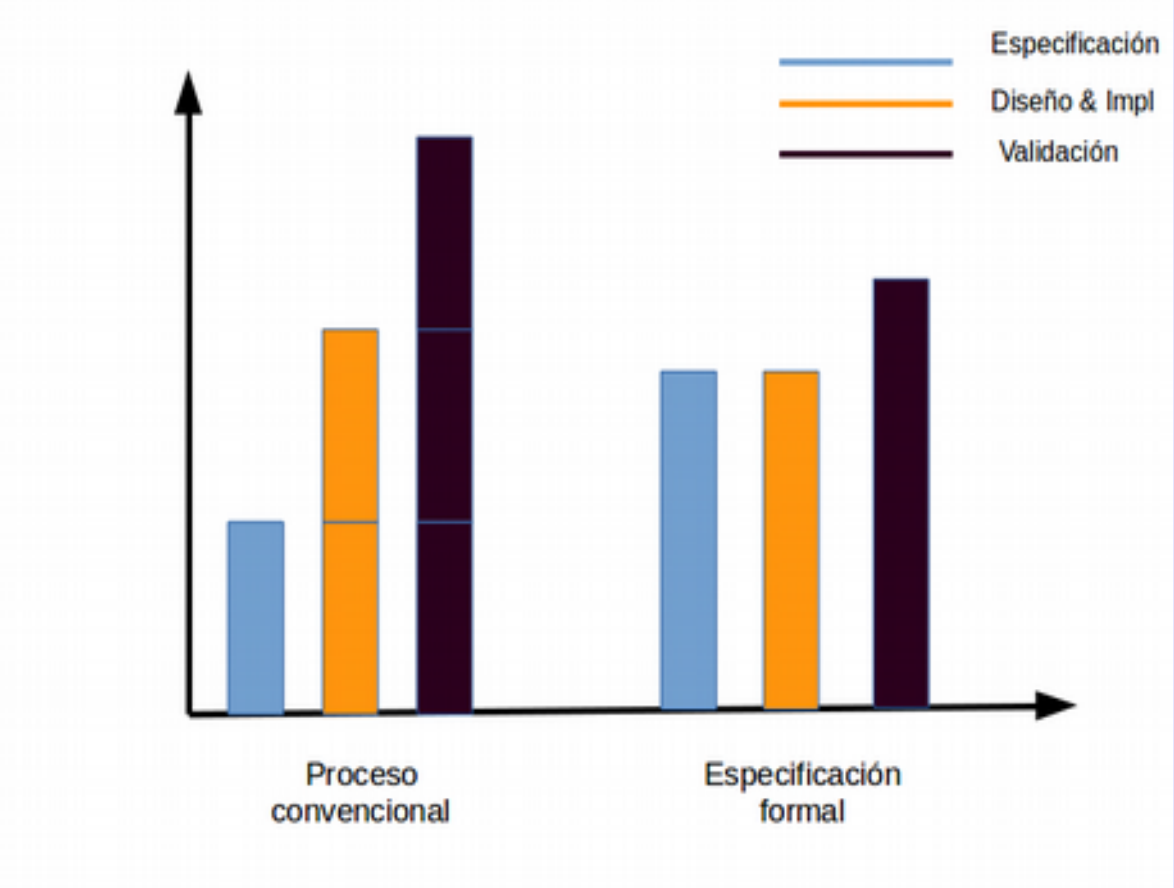
    *{R≡a=b*q+r ∧ r<b ∧r ≥ 0 }*

Introducción

- ¿Por qué desarrollar una especificación formal?
  - Revela errores e incongruencias en la especificación informal de requerimientos
  - Descubrimiento temprano de problemas que serían muy caros de corregir en fases posteriores
- ¿Qué necesita?
  - Vocabulario, sintaxis y semántica formalmente definidos
  - Basarse en conceptos matemáticos cuyas propiedades se comprendan bien

Introducción

- ¿Merece la pena?
  - Desplaza los costes hacia las primeras etapas de desarrollo



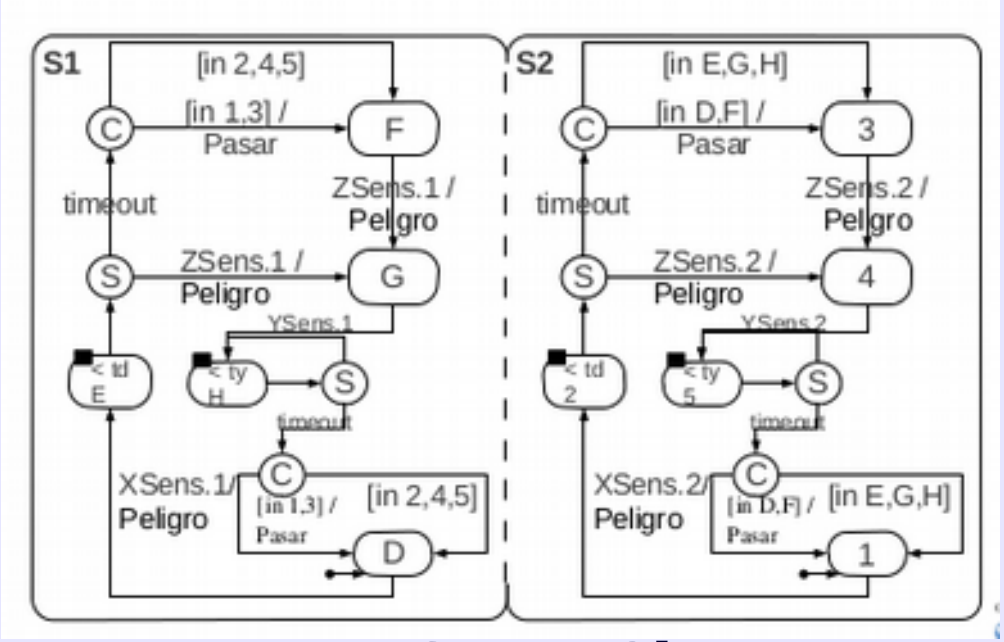
Introducción

- Técnicas de especificación formal
  - Aproximación algebraica o ecuacional
    - Especificar tipos abstractos de datos
    - El sistema se describe en función de las operaciones y sus relaciones
    - Las operaciones independientes del estado
    - La semántica de las operaciones a través de axiomas
  - Aproximación basada en modelos
    - Creación de un modelo del sistema utilizando matemáticas
      - Conjuntos, lógica de primer orden
    - Especificación del sistema en base a sus estados
    - Las operaciones se describen indicando como modifican el estado del sistema

Introducción

- Diferentes lenguajes
  - Especificación requisitos
    - Aproximación algebraica
      - OBJ, Larch
    - Aproximación basada en modelos
      - Z, VDM, B
  - Especificación de sistemas
    - Aproximación algebraico
      - Lotos
    - Aproximación basada en modelos
      - CSP, CSS, Redes de Petri

Introducción



StateChart

$$[NCTA] \qquad SALDO == \mathbb{N}$$

Banco

$$cajas : NCTA \pfun SALDO$$

DepositarOk

$$\Delta Banco$$
$$num? : NCTA$$
$$monto? : \mathbb{Z}$$

$$num? \in \mathrm{dom}\ cajas$$
$$monto? > 0$$
$$cajas' = cajas \oplus \{num? \mapsto cajas\ num? + monto?\}$$

Z

$$BUFFER = long?n + 1 \rightarrow B(n+1, \langle\rangle)$$

$$B(m+1, \langle\rangle) = left?n : \mathbb{N} \rightarrow B(m, \langle n \rangle)$$

$$B(m+1, s^\frown\langle y \rangle) =$$
$$left?n : \mathbb{N} \rightarrow B(m, \langle n \rangle^\frown s^\frown \langle y \rangle)$$
$$\mid right!y \rightarrow B(m+2, s)$$

$$B(0, s^\frown\langle y \rangle) = right!y \rightarrow B(1, s)$$

$$SYSTEM = PRODUCER \parallel BUFFER \parallel CONSUMER$$

CSP

OCL

context Meeting:: checkDate():Bool
post: result = self.participants->collect(meetings) ->forAll(m | m<> self and m.isConfirmed implies (after(self.end,m.start) or after(m.end,self.start)))

context Meeting::isConfirmed ()
post: result= self.checkdate() and self.numConfirmedParticipants > 2

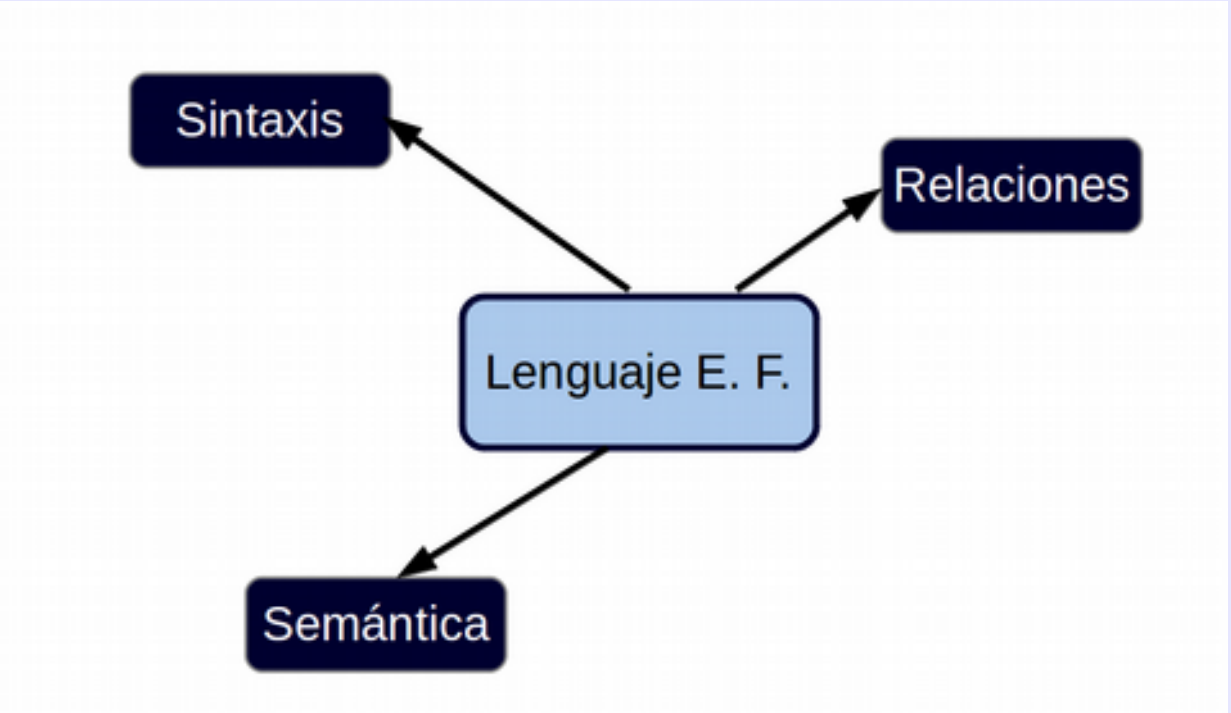
context Meeting :: duration ( ) : Time
post: result = timeDifference (self.end, self.start)

context Person:: numMeeting ( ): Nat
post: result = self.meetings -> size

context Person :: numConfirmedMeeting ( ) : Nat
post: result= self.meetings -> select (isConfirmed) -> size

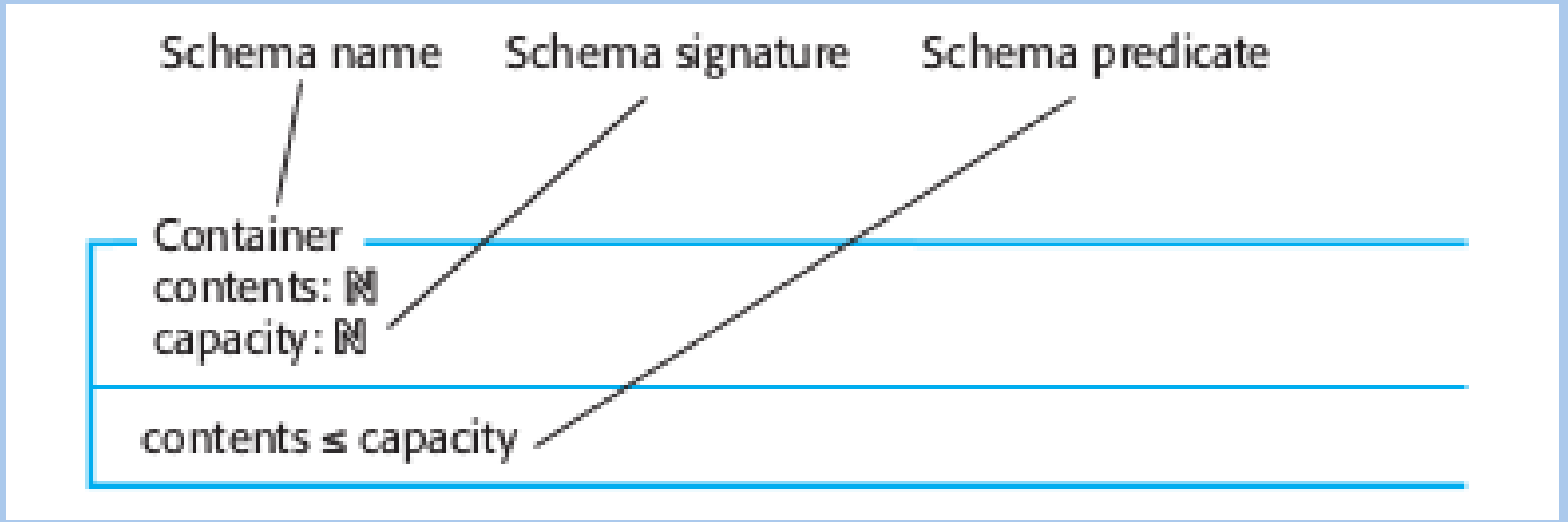
Componentes Lenguaje

- Los lenguajes de especificación formal se usan para escribir la especificación funcional de un programa



Aprox. Basada en Modelos

- Notación Z
  - Combina descripción formal e informal
  - Descripción formal en trozos fáciles de leer y resaltados
    - Esquemas



Aprox. Basada en Modelos

- Ejemplo: Gestor de bloques
  - Esquema que presenta el estado y el invariante

GestorBloques

$usados, libres: \mathbb{P}\ BLOQUES$

$FilaBloques: seq\ \mathbb{P}\ BLOQUES$

$usados \cap libres = \emptyset \land$

$usados \cup libres = TodosBloques \land$

$\forall i: \mathbf{dom}\ ColaBloques \bullet FilaBloques\ i \subseteq usados \land$

$\forall i, j: \mathrm{dom}\ ColaBloques \bullet i \neq j \Rightarrow ColaBloques\ i \cap ColaBloques\ j = \emptyset$

Aprox. Basada en Modelos

- Ejemplo: Gestor de bloques
  - Esquema que describe la operación que elimina un elemento de la cola de bloques

EliminarBloques

Δ *GestorBloques*

$\#GestorBloques > 0,$

$usados' = usados \setminus cabeza\ ColaBloques \land$

$libres' = libres \cup cabeza\ ColaBloques \land$

$ColaBloques' = cola\ ColaBloques$

Aprox. Basada en Modelos

- Ejemplo: Gestor de bloques
  - Esquema que describe la operación añade una colección de bloques al final de la cola

AñadirBloques

Δ GestorBloques

Abloques? : BLOQUES

$Abloques? \subseteq usados$

$FilaBloques' = FilaBloques \frown (Abloques?) \land$

$usados' = usados \land$

$libres' = libres \land$

Especificación algebraica

- Estructura (Somerville)