Langage B ?

Abstract data types and algebraic specifications

Pourquoi :

- pas d'ambiguïté
- formel

=> pour parties critique

- réduit le coût d'implémentation
- m prends du temps, faible adoption

(structures algébriques ex groupes, monoïdes, . . .)

définir quoi, pas comment

contraintes sur résultats

Sémantique

Format :

- title
- "sort" name (abstract type)
- references to other sorts
- natural language description of the operations
- [preconditions]
- operation signatures
- axioms (constraints over operation results)

Axioms : several formalisms :

- Booleans : propositionnal logic, first order logic
- Integers : arithmetics and algebra
- Set theory
- . . .

Constructor operations :

- Primary constructors : create the sort, **not directly defined**
- Secondary constructors : modify the sort, can be defined with ˆ Inspection operations : allows to get informations about the sort

Ex - List :

- Title : LIST(E)
- Sort : List
- References : INTEGER
- Description : Defines a list where elements are added at the end and . . .
- Signatures :

  - Create : -> List
  - Cons : List x E -> List
  - Head : List -> E
  - Length : List -> Integer
  - Tail : List -> List
- Axioms :
  - Head(Create) = Undefined (error empty list)
  - Head(Cons(L, v)) = if L == Create then v else Head(L)
  - Length(Create) = 0
  - Length(Cons(L, v)) = Length(L) + 1
  - Tail(Create) = Create
  - Tail(Cons(L, v)) = if L = Create then Create else Cons(Tail(L), v)

```
Length([3, 7, 8])
Length(Cons([3, 7], 8)) = Length([3, 7]) + 1
 Length(Cons([3], 7)) = Length([3]) + 1
  Length(Cons(Create, 3)) = Length(Create) + 1
   Length(Create) = 0
  1
 2
3
```

Must be sound & complete => no contradictory axioms & enough axioms to describe the operations' semantic

(for each inspection operation (m) write an axiom for each constructor operation (n) => m * n axioms)