# Specifica – parte IIIC

Leggere Sez. 5.6.3 Ghezzi et al.

# Specifiche algebriche

- Tradizionalmente, un'*algebra* è composta da un *insieme* e *operazioni* sull'insieme.
- Esempio: insieme N con addizione, sottrazione, moltiplicazione, divisione.
- Le algebre definite su un solo insieme si dicono *omogenee*.

# Specifiche algebriche

- Definiscono un sistema come un'*algebra eterogenea*, cioè come collezione di insiemi diversi su cui sono definite operazioni.
- Adatte alla definizione di tipi di dati astratti

# Algebra per definizione di stringhe

- Vogliamo definire le operazioni:
  - **new**: creazione di nuove stringhe
  - **append**: concatenazione di stringhe
  - **add**: aggiunta di un carattere in coda a una stringa
  - **length**: calcolo della lunghezza di una stringa
  - **isEmpty**: controllo se una stringa è vuota
  - **equal**: controllo se due stringhe sono uguali

# Algebra per definizione di stringhe

- Insiemi che formano l'algebra:
  - String *dominio di interesse*
  - Nat
  - Bool
  - Char
- Ogni insieme si dice *sort (tipo)*
- La collezione di insiemi si dice *segnatura (signature)*.

# Sintassi

- La *sintassi* di un'algebra è definita dalla sua *signature* e dall'elenco delle operazioni, con i rispettivi domini e codomini.
- Larch: famiglia di linguaggi per specifiche formali, formati da:
  - parte comune (Larch Shared Language)
  - parte di interfaccia con linguaggi di programmazione (es. Larch/Pascal)

# Esempio (Larch)

**algebra** StringSpec;
**introduces**
  **sorts** String, Char, Nat, Bool;
  **operations**
    new: () → String;
    append: String, String → String;
    add: String, Char → String;
    length: String → Nat;
    isEmpty: String → Bool;
    equal: String, String → Bool

# Semantica

- Il significato delle operazioni deve essere specificato formalmente.
- La *semantica* di un'algebra è definita da un insieme di *equazioni*
- Rappresentano le *proprietà* essenziali delle operazioni.
- Sono dette anche *assiomi*, in quanto usate per dedurre proprietà delle algebre.

# Esempio

**constrains** new, append, add, length, isEmpty, equal **so that**
String **generated by** [new, add]
for all [s, s1, s2: String; c: Char]
    isEmpty (new ()) = true;
    isEmpty (add (s, c)) = false;
    length (new ()) = 0;
    length (add (s, c)) = length (s) + 1;
    append (s, new ()) = s; (2)
    append (s1, add (s2,c)) = add (append (s1,s2),c); (1)
    equal (new (),new ()) = true;
    equal (new (), add (s, c)) = false;
    equal (add (s, c), new ()) = false;
    equal (add (s1, c), add (s2, c)) = equal (s1,s2);
**end** StringSpec.

# Deduzione di proprietà

append(new(), add(new(),c)) = add(new(), c)

- Dall'assioma (1) con $s_1$=$s_2$=new(),

append (new(), add (new(),c)) = add (append (new(),new()),c); (3)

- Dall'assioma (2), con s = new(),

append(new(), new()) = new()

- E sostituendo in (3) la tesi.

# Deduzione di proprietà

- append(new(),s) = s (4)
- Per induzione sulla lunghezza di s:
  - se s = new(): (4) è vera per l'assioma (2)
  - se (4) è vera per ogni $s_0$ di lunghezza L: ogni $s_1$ di lunghezza L+1 sarà add($s_0$, c) per qualche $s_0$ e c, e

    append(new(),s1) =
    append(new(),add(s0,c)) =
    add(append(new(),s0),c) = add(s0,c) = s1

# Possibili problemi

- Incompletezza
- Sovraspecifica
- Incoerenza
- Ridondanza

# Incompletezza

- Si può dimostrare che

  equal(add(s,’a’),add(s,’b’))=false?

- No. Occorre sostituire l’ultimo assioma con:

equal (add (s1, c1), add (s2, c2)) =

equal (s1,s2) and equalC(c1,c2);

# Sovraspecifica

- Ad es., caso in cui l'ultimo assioma sia:

equal (add (s1, c1), add (s2, c2)) = equal (s1,s2) and equalC(c1,c2) and not equalC(c1,'a');

Afferma che due stringhe sono uguali se non contengono il carattere 'a'.

# Incoerenza

- Si ha quando si riesce a dimostrare true=false
- Ad es. se includessimo l'assioma

  equal(add(s,c),s)=true

# Ridondanza

- Presenza di assiomi non necessari in quanto derivabili
- Ad es. l'assioma:

append(new(),s)=s

To do come esercizio (questo assioma, in realtà, lo abbiamo appena dedotto come proprietà …)

# Specifiche algebriche: perché

- Sono una notazione utile per la specifica della semantica di ADT
- Sono complementari a notazioni come TDN o GDN, o UML
- Specifiche più astratte delle pre/post condizioni e degli invarianti