

Sommario: 12 Aprile, 2021

- Small21: Definizione e Implementazione
- Definizione: Sintassi e Semantica
- Attività di Oggi: Primi Esercizi
  - Macchina Astratta e AST:  
Rappresentazione dei Programmi nella AM
  - Sintassi Concreta:  
Presentazione dei Programmi  
Una Grammatica Non Ambigua CF

# Small21: Definizione ed Implementazione

## ● Motivazioni e Obiettivi.

- Apprendere la Definizione di un Linguaggio Imperativo:  
**Small21**
- Scrivere Programmi per la Definizione di una AM21:  
**Strutture ed Esecutore di Small21**
- Programmare (intero sviluppo) in un Linguaggio Funzionale:  
**OCaml - Nucleo Funzionale**
- Scrivere programmi, non banali, in (AST) Small21  
**Interfacce<sup>1</sup> ed Espressività<sup>2</sup>**
- Ottenere esecuzioni corrette, secondo la Semantica di Small21  
**Verifica e Correttezza**
- Estendere e/o Modificare Small21 e/o AM21  
**Progetto/Seminario individuale di Fine Corso**

## ● Vincoli.

- Small21 deve essere Imperativo (Valori con Stato) e Prescrittivo (Sequenza)
- Sviluppabile (da noi) in 20 ore (meno 2 ore di oggi)
- Sviluppo collegiale (no suddivisioni)

---

<sup>1</sup> La forma con cui si comunica con AM21: ASTs e (richiesta della) loro Computazione.

<sup>2</sup> Gli algoritmi implementabili in Small21 senza ricorrere ad emulazioni di Nuove Strutture di Controllo.

# Small21: Criteri Generali della Definizione - 1

- **Sintassi Astratta e Concreta.**
  - Programmi: Scriviamo in S.A. ma Presentiamo in S.C.
    - Svantaggi:
    - Vantaggi:
- **Elenchiamo strutture e costrutti.**
  - Minimo (non troppo):
  - Idea:
  - Elenco:
    - **Variabili** (Valori Modificabili)
    - **Valori** (o Dati):
      - Scalari (o Atomici):
      - Strutturati:
    - **Tipi:**
    - **Dichiarazioni:**
    - **Espressioni:**
      - Operatori Dati, Controllo Sequenza e Astrazioni
    - **Comandi:**
      - Operatori Stato, Controllo Sequenza e Astrazioni
    - **Programma:**

# Small21: Struttura e Principali Costrutti

## ● Sintassi Astratta e Concreta.

- I Programmi li scriviamo solo in Sintassi Astratta.
  - Svantaggi: Pesante scrivere programmi in Sintassi Astratta
  - Vantaggi: AM21 non ha bisogno (di Lexer, e Parser) di Front-End
- I Programmi li presentiamo (anche) in Sintassi Concreta.

## ● Elenchiamo Strutture e Costrutti:

- Minimo(o quasi): Indispensabile per definire le funzioni in  $\mathcal{F}$
- Idea: Partiamo dal nucleo del C - estendiamo e generalizziamo.
- Elenco:
  - **Tipi**: `int`, `bool`, `lab`, ..., **Sistema di Tipi** (Programmi Ben Tipati)
  - **Variabili** (valori Modificabili o con Stato)
  - **Label** (etichette per Posizioni in sequenza programma)
  - **Valori**:
    - Scalari (o Atomici): `Interi`, `Booleans`, ...
    - Strutturati: `Array`, ...
  - **Dichiarazioni**: `Variabile`, `Costante`, `Array`, `Procedura-Funzione Ricorsiva`, ...
  - **Espressioni**: Aritmetiche, Relazionali, Logiche, `Invocazione`, `Blocco-Exp`, ...
  - **Comandi**: `Assegn.`, `Cond.`, `Sequenza`, `goto`, `Blocco-Cmd`, `Invocazione`, `Iteratori`, ...
  - **Programma**: `Blocco-Cmd`

# Sintassi Astratta: Una grammatica di alberi AST

```
Type ::= [int] | [bool] | [lab] | [void]
      | [mut] Type | [arr] Type Num | [terr]
      | [abs] TypeSeq
```

```
Dcl ::= [var] Type Ide Exp | [const] Type Ide Exp
      | [array] Type Ide | Dcl [seqD] Dcl
      | [pcd] Type Ide FPars BlockC | [emptyD]
```

```
Exp ::= DExp
      | [num] Num | Exp [+] Exp
      | Exp [==] Exp
      | [true] | [false] | Exp [or] Exp
      | Exp [>] Exp | Exp [<] Exp
      | [apply] Ide APars | [emptyE]
```

```
DExp ::= [val] Ide | DExp [↑] Exp | Ide [↑1] Exp
```

```
Cmd ::= Lab [:] Stm | Stm
```

```
Stm ::= DExp [=] Exp | [ifE] Exp Stm Stm | [goto] Lab
      | Stm [seqC] Stm | BlockC | [emptyS]
      | [call] Ide APars | [return] Exp
```

```
Prog ::= [prog] Ide Block
```

# Sintassi Astratta: Una grammatica di alberi AST - 2

TypeSeq ::= Type [::] TypeSeq | Type

Num                               -- token del lessico degli interi  
Ide                               -- token del lessico degli identificatori  
Lab                               -- token del lessico delle labels di cmd

FPars ::= [fp] PPF Type Ide | [emptyFP]

PPF ::= [value] | [ref]           -- Parameter passing Form

BlockE ::= [blockE] Dcl Exp

BlockC ::= [blockC] Dcl Stm      -- in-line block

Block ::= [block] Dcl CmdSeq

CmdSeq ::= [seqC] Cmd CmdSeq | EC

APars ::= [ap] Exp | [emptyAP]

## Osservazioni

- [mut]: Costruttore per Tipi di valori mutable. Ad es.: [mut][int], [mut]([arr]([int],30)), [arr]([mut][bool],15).
- ↑: Costruttore di termine di accesso a componente array. Ad es.: x ↑ 3 corrisponde, in C, all'espressione x[3] quando x sia una variabile di tipo [mut]([arr]([mut][int],30)).
- SeqTypes: Il tipo più a sinistra indica il tipo immagine i successivi, quando presenti, i tipi degli argomenti della astrazione funzionale, procedurale o di operatore primitivo.
- FPars e APars: Parametri per astrazioni al momento con al più un solo parametro e solo 2 forme di trasmissione.

## Esercizio (1)

*Nel file "Small21.ml" allegato, si estenda la definizione Ocaml del tipo dcl in modo da fornire la Sintassi Astratta delle dichiarazioni del Linguaggio Small21. Allo scopo si riveda quanto fatto, in Ocaml, per la sintassi astratta del Lambda Calcolo.*

## Esercizio (2)

Si calcoli in OCaml lo AST di Small21 corrispondente alla seguente sequenza C:

```
const int x = 7;
int z = x + 3;
int A[7][3][5];
int succ(int n){return n+1;}
```

nell'ipotesi che il significato atteso per gli analoghi costrutti sintattici di Small21, sia lo stesso dato in C. In particolare, nel caso di array, si ricordi che il costrutto "[arr] t k" indica una struttura array di k componenti di uno stesso tipo t.

Per le strutture parzialmente definite si usino i costrutti disponibili al momento dello svolgimento.

## Esercizio (3)

Si proceda come in [1] con gli AST per `exp`, `cmd`, e `stm`. Quindi si calcoli l'AST di `Small121` per la sequenza riportata sotto:

```
loop: if (i==size) goto stop;
sum = sum + A[i];
i = i + 1;
goto loop;
```

sempre nelle ipotesi di esercizio in [2]

Per le strutture parzialmente definite si usino i costrutti disponibili al momento dello svolgimento.

## Esercizio (4 - Uno sguardo alla Macchina Astratta AM21)

*Nell'implementazione che abbiamo fin qui dato, com'è strutturata e dov'è la memoria in cui risiedono i programmi di Small21?*

- a) *Quando AM21 è "spenta" (non è in esecuzione): Struttura e Luogo?*
- b) *Quando AM21 è "accesa": Struttura e Luogo?*