INDUZIONE STRUTTURALE:

1. Definisce/Dimostrare una proprietà sugli Elementi Base dell'Insieme ovvero non definiti in funzione di Altri Elementi

2. Nel Passo Assumo la proprietà sugli Elem. dell'Insieme per dimostrare la proprietà Sugli ELEMENTI COMPOSTI, quindi i COMPONENTI IMMEDIATI di ogni REGOLA COSTRUTTIVA

ESEMPI:

$X \subseteq \mathbb{N}$ e per def:

BASE: $0 \in X$

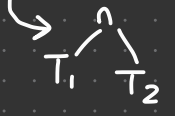
PASSO: se $x \in X$ allora $x+3 \in X$

Si vede che $X = 3\mathbb{N}$

Binary-Trees

BASE: $n \in \mathbb{N}$ è un BT ($n \in BT$) ... $n \subseteq BT$ → Mette una Radice

PASSO: Se T_1 e $T_2 \in BT$ allora $br(n, T_1, T_2) \in BT$



FOGLIE ⇒ Conta il # In un Albero

Inizio a definire:

$foglie(n) = 1$

$foglie(br(n, T_1, T_2)) = foglie(T_1) + foglie(T_2)$

Nodi ⇒ Conta il #Nodi di un BT

$$Nodi(n) = 1$$
$$Nodi(br(n, T_1, T_2)) = Nodi(T_1) + Nodi(T_2) + 1$$

Branch ⇒ Conta #diramazioni di Te BT

$$Branch(n) = 0$$
$$Branch(br(n, T_1, T_2)) = Branch(T_1) + Branch(T_2) + 1$$

Posso dimostrare che $\forall T \in BT.\ foglie(T) = Branch(T) + 1$

BASE: (Un Nodo) $foglie(n) = 1$, $Branch(n) = 0$ } CONFERMATO

PASSO: $br(n, T_1, T_2) \in BT$ con $T_1, T_2 \in BT$

Ip. Induttiva assume proprietà per i Componenti Immediati che Sono T_1, T_2 allora:

- $foglie(T_1) = Branch(T_1) + 1$
- $foglie(T_2) = Branch(T_2) + 1$

Ed Allora:

$$foglie(br(n, T_1, T_2)) = foglie(T_1) + foglie(T_2)$$

APPLICO IPOTESI

$$= (Branch(T_1) + Branch(T_2) + 1) + 1$$

SECONDO LA DEFINIZIONE

$$= Branch(br(n, T_1, T_2)) + 1$$

GRAMMATICHE:

$P \to aa \mid bb \mid bPb \mid aPa$

NON TERINALI Sono le CATEGORIE

Def. di Insieme di Stringhe ST:

- $aa, bb \in ST$
- Se $x \in ST$ Allora $axa, bxb \in ST$

Posso Dimostrare che ST è l'Insieme delle Stringhe Palindrom

$\forall x. \in SP \quad \underbrace{\exists yz \in \{a,b\}^* \quad x = yz \quad e \quad y = reverse(z)}$

ciò Implica che $|x|$ è PARI

BASE: $x = aa, y = a, z = a$ ed ho che $x = yz = aa$ e $y'' = reverse\, z'' = a$

$x = bb, y = b, z = b$ e ho che $x = yz = bb$ e $y'' = reverse(z) = b$

PASSO: $x \in SP$

① axa per Ipotesi $\exists yz.\ x = yz$ e $y = reverse(z)$ e quindi devo dimostrare che $\exists y'z'.\ x' = axa$ e $y' = reverse(z')$

$y' = ay, \ z' = za \Rightarrow x' = axa = ayza = y'z'$

$y' = reverse(z') = reverse(za) \overset{def}{=} a\, reverse(z) = ay = y'$

② Analogo ma REPLACE (a, b)

ESPRESSIONI ARITMETICHE

Grammatica: $E \to n \in \mathbb{N} \mid E+E \mid E \cdot E$

Posso scrivere la Grammatica come Linguaggio Generato, insieme expressions:

- **BASE:** $n \in EXP$ $(n \in \mathbb{N})$
- **PASSO:** se $e_1, e_2 \in EXP$ allora $e_1 + e_2 \in EXP$, $e_1 \cdot e_2 \in EXP$

OP conta il #Operatori di un EXPRESSION

- $op(e) = 0$
- $op(e_1 + e_2) = op(e_1) + op(e_2) + 1 = op(e_1 \cdot e_2)$

VAL conta il #Valori

- $Val(n) = 1$
- $Val(e_1 + e_2) = Val(e_1 \cdot e_2) = Val(e_1) + Val(e_2)$

Dimostriamo $\forall e \in EXP \quad Val(e) = op(e) + 1$

BASE: $n \in EXP \quad val(n) = 1 = 1 + 0 = 1 + op(n)$

PASSO: $e_1, e_2 \in EXP \quad e_1 + e_2$ devo dim che:

$$val(e_1 + e_2) = op(e_1) + op(e_2) + 1$$

Per Ipotesi: $Val(e_1) = op(e_1) + 1$

$Val(e_2) = op(e_2) + 1$

E Quindi:

$$Val(e_1 + e_2) = Val(e_1) + Val(e_2) = op(e_1) + 1 + op(e_2) + 1$$

$$= (op(e_1) + op(e_2) + 1) + 1$$

$$= op(e_1 + e_2) + 1$$

Raggruppo così è più facile NOTARE la definizione data prima

Ho dimostrato ciò che Volevo

COMPOSIZIONALITÀ: (proprietà Importante)

Il Significato di Ogni Costrutto/Programma deve DIPENDERE DIRETTAMENTE (essere funzione) dalle COMPONENTI IMMEDIATE

SISTEMI DI TRANSIZIONE:

Strumento per definire SEMANTICA OPERAZIONALE

- Sono Matematicamente Precisi ⇒ Costrutto formale
- Abbastanza Concisi
- Permettono di Dare SEMANTICA in MODO INDUTTIVO Sulla struttura della Sintassi

Questo Avviene Mediante un insieme di Regole, Metodo Molto

Generale che permette di ASTRARRE (decidere di come descrivere il mio sistema, scendere molto o meno in dettaglio)

È una coppia $\langle T, \rightarrow \rangle$ dove:

- T insieme di Configurazioni/Stati $\gamma \in T$ (descrizione formale dello stato della Macchina)
- $\rightarrow$ Relazione Binaria di Transizione $\subseteq T \times T$

Notazioni Utili: $[(\gamma_1 \gamma_2) \in \rightarrow]$ Viene denotato $\gamma_1 \rightarrow \gamma_2$

$\gamma_0 \rightarrow^* \gamma_n \iff \exists \gamma_1 \dots \gamma_{N-1}$ t.c. $\gamma_0 \rightarrow \gamma_1 \rightarrow \dots \rightarrow \gamma_N$

SISTEMA DI TRANSIZIONE TERMINALE

$\langle T, T, \rightarrow \rangle$ dove $\langle T, \rightarrow \rangle$ è un Sistema di Transizione

$T \subseteq T$ insieme di Configurazioni Terminali t.c.

$\forall \gamma \in T. \nexists \gamma' \in T \quad \gamma \rightarrow \gamma'$

GRAMMATICA $G = \langle V, T, P, S \rangle$

$T = (V \cup T)^*$ quindi qualsiesi sequenza di Terminali e NON

$T \rightarrow P$

$T \rightarrow A \rightarrow \alpha$ ($\alpha \in T^*$)

$$\frac{A \rightarrow \alpha}{\beta A \gamma \rightarrow \beta \alpha \gamma} \equiv (A \rightarrow \alpha) \rightarrow (\beta A \gamma \rightarrow \beta \alpha \gamma)$$

$S \to \alpha \in T^*$

NON DETERMINISMO SISTEMI DI TRANSIZIONI

Posso Avere 2 Transizioni

ad Esempio: $e_1 + e_2$, avendo la prop. COMMUTATIVA ho 2 possibili Valutazioni da fare che però portano Allo Stesso Risultato

E posso Averlo in un linguaggio Determinisco

CATEGORIE SINTATTICHE:

Strumenti Per:

- Manipolare dati → ESPRESSIONI
- Manipolare/Creare Legami → DICHIARAZIONI
- Trasformare Stati/Ambienti → COMANDI

AMBIENTE

Legato alle **dichiarazioni** è l'Insieme dei legami ID-OGGETTI

Procedura ha il Suo AMBIENTE

MEMORIA

Ha un **Andamento Unico**, le **operazioni** fatte sono **IRREVERSIB.** Non Posso Tornare in dietro

Permette di descrivere gli Effetti di una Trasformazione

ESPRESSIONI:

SONO FINITE (DECIDIBILE)

- denotare Valori
- Vengono **Valutate** per Restituire un Valore

∃.e sintatticamente ≠ ma con Stessa Valutazione

- **Sono EQUIVALENTI se denotano lo Stesso Valore**

(sono UGUALI se sintatticamente uguali)

DICHIARAZIONI:

DECIDIBILE

- denotano Richieste di Modifica degli Ambienti
- **Elaborate** per attuare le Trasformazioni dell'Ambiente
- **Sono EQUIVALENTI se Vengono Elaborate Nello Stesso Ambiente**

COMANDI:

NON DECIDIBILI

- Denotano Richieste di Modifica della Memoria
- Vengono **Eseguiti** per Identificare/ottenere la Trasformazione della Memoria Corrispondente
- Programmi **EQUIVALENTI Generano Trasform. Uguali della Memoria**