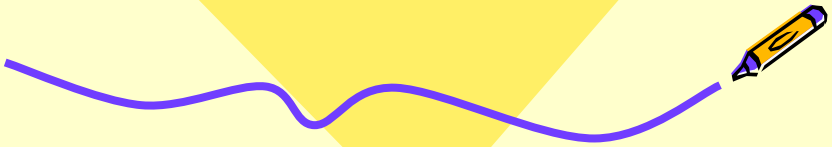
Fondamenti dei linguaggi di programmazione

Aniello Murano
Università degli Studi di Napoli
"Federico II"

Riassunto delle lezioni precedenti

- Prima Lezione: Introduzione e motivazioni del corso; Sintassi e semantica di ARITHM
- Seconda lezione: Sintassi e semantica operazionale del linguaggio imperativo IMP
- Terza lezione:

Tecniche di prova per induzione



Introduzione

- Le dimostrazioni di proprietà degli elementi di un insieme (e dunque anche dei programmi) si basano spesso sull'uso di un metodo di prova chiamato ***induzione***
- Le forme di induzione più comunemente usate sono ***l'induzione matematica*** e ***l'induzione strutturale***
- Induzione matematica e strutturale sono casi particolari di un potente metodo di prova chiamato ***induzione ben fondata***
- In questa lezione vedremo in dettaglio:
  - Induzione matematica (la più semplice)
  - Induzione strutturale (la più usata nel contesto della semantica)
  - Induzione ben fondata (la più generale)
  - Induzione sulle derivazioni (utile per i comandi)

Uso dell'induzione

- Per utilizzare l'induzione come metodo di prova su un determinato insieme occorre
- Definire l'insieme utilizzando delle regole
  - Per esempio, come abbiamo fatto con gli insiemi Aexp, Bexp, Com nella sintassi astratta di IMP
- Definire una relazione sugli elementi dell'insieme utilizzando regole
  - Per esempio, come abbiamo fatto per la relazione di derivazione per IMP
- Definire una funzione sugli elementi dell'insieme

Induzione matematica

- Una proprietà P(n) è vera per tutti i numeri naturali se
  - P(0) è vera
  - Se P(m) vera implica P(m+1) vera per tutti i numeri naturali m
- Oppure, usando le regole di inferenza

$$\frac{P(0) \qquad \forall\, m \in \omega\,.\, P(m) \Rightarrow P(m+1)}{\forall\, n \in \omega\,.\, P(n)}$$

- Terminologia:
  - P(0) è il ***caso base***
  - P(m) è ***l'ipotesi induttiva***
  - $\forall\, m \in \omega\,.\, P(m) \Rightarrow P(m+1)$ è il ***passo induttivo***

Primo esempio

◆Theorem:

$2^0 + 2^1 + \ldots + 2^n = 2^{n+1} - 1$

◆Proof: By induction on n

◆Base case (n = 0)

$2^0 = 1 = 2^{1+1} - 1$

◆Inductive case (P(n) ⇒ P(n + 1))

$$\begin{aligned} 2^0 + 2^1 + \ldots + 2^{n+1} &= (2^0 + 2^1 + \ldots + 2^n) + 2^{n+1} \\ &= (2^{n+1} - 1) + 2^{n+1} \qquad \text{IH} \\ &= 2 \cdot 2^{n+1} - 1 \\ &= 2^{n+2} - 1 \end{aligned}$$

P(n + 1)

Secondo esempio

- Si ricordino le regole di valutazione per i comandi Com di IMP
- Si provi che se $\sigma(x) \leq 6$ allora
  ‹while $x \leq 5$ do x:=x+1, σ› → σ[x:=6]
- Al fine di utilizzare una induzione matematica, conviene riformulare l'enunciato "parametrizzato" sui numeri naturali nel modo seguente
  - Sia W = while $x \leq 5$ do x:=x+1
  - Let $\sigma_i = \sigma[x:=6-i]$
  - Teorema: $\forall i \in \mathbb{N}$, ‹W,σ_i› → σ_0
- Così formulato, il teorema può essere provato per induzione matematica sull'indice i.

Secondo esempio (caso base)

- Caso Base: i = 0 or $\langle W, \sigma_0 \rangle \to \sigma_0$
- Per provare una valutazione, è necessario costruire l'albero di derivazione :

$$\dfrac{\dfrac{\dfrac{\sigma_0(x) = 6}{\langle x, \sigma_0 \rangle \to 6} \quad \dfrac{}{\langle 5, \sigma_0 \rangle \to 5}}{\langle x \leq 5, \sigma_0 \rangle \to \text{false}}}{\langle \text{while } x \leq 5 \text{ do } x := x + 1, \sigma_0 \rangle \to \sigma_0}$$

- Questo completa il caso base

Secondo esempio (passo induttivo)

- Dobbiamo provare che $\forall\, i \in \mathbb{N}.\ \langle W, \sigma_i \rangle \to \sigma_0 \Rightarrow \langle W, \sigma_{i+1} \rangle \to \sigma_0$
- L'inizio della prova è semplice:
  - Si sceglie un arbitrario $i \in \mathbb{N}$
  - Si assume che $\langle W, \sigma_i \rangle \to \sigma_0$
  - Si prova che $\langle W, \sigma_{i+1} \rangle \to \sigma_0$
  - Per quest'ultimo dobbiamo costruire un albero di derivazione.

$$\dfrac{\dfrac{\dfrac{}{\langle x, \sigma_{i+1} \rangle \to 5\text{-}i \leq 5}}{\langle x \leq 5, \sigma_{i+1} \rangle \to \text{true}} \quad \dfrac{\dfrac{\langle x{+}1, \sigma_{i+1} \rangle \to 6\text{-}i}{\langle x := x{+}1, \sigma_{i+1} \rangle \to \sigma_i} \quad \langle W, \sigma_i \rangle \to \sigma_0}{\langle x := x{+}1;\ W, \sigma_{i+1} \rangle \to \sigma_0}}{\langle \text{while } x \leq 5 \text{ do } x := x + 1, \sigma_{i+1} \rangle \to \sigma_0}$$

Osservazione

- Una dimostrazione formale è più potente dell'esecuzione del codice e della successiva valutazione del risultato (Perchè?)
- Noi abbiamo provato terminazione e correttezza. La combinazione di queste due proprietà è chiamata ***correttezza totale.***
- L'induzione matematica è efficiente quando proviamo proprietà di numeri naturali.
- Ma nell'analisi dei linguaggi di programmazione molto spesso proviamo proprietà di wpressioni, comandi, dati in input ai programmi, ecc.
- Abbiamo bisogno di un principio di induzione più forte.

Induzione Strutturale

- Induzione basata sulla struttura dei termini

"Se la base dell'induzione è vera e se, per un certo termine s, P(r) vale per tutti gli immediati sottotermini r di s, allora P(n) vale per tutti gli elementi n dell'insieme"

Induzione strutturale: Osservazioni

- Questo metodo di prova è chiamato induzione strutturale perché le prove sono guidate dalla struttura delle espressioni
- In una prova, devono essere considerati molti casi, dovuti alle varie forme delle espressioni:
  - Espressioni atomiche(senza sottoespressioni) sono tutti i case base
  - Le espressioni composte sono i casi induttivi
- L'induzione strutturale è la forma più utile di induzione nello studio dei linguaggi di programmazione

Induzione Strutturale su Aexp

- Una proprietà P(a) è vera per ogni a in Aexp se
  - Per ogni numero naturale m, P(m) vale
  - Per ogni locazione X, P(X) vale
  - Per ogni a_0 e a_1, se $P(a_0)$ e $P(a_1)$ valgono, allora vale $P(a_0 + a_1)$
  - Per ogni a_0 e a_1, se $P(a_0)$ e $P(a_1)$ valgono, allora vale $P(a_0 - a_1)$
  - Per ogni a_0 e a_1, se $P(a_0)$ e $P(a_1)$ valgono, allora vale $P(a_0 \times a_1)$

Esempio sulla struttura delle espressioni

- Let
  - L(e) be the number of literals and variable occurrences in e
  - OP(e) be the number of operators in e
- Prove that $\forall e \in Aexp.\ L(e) = OP(e) + 1$
- By induction on the structure of e
  - Case e = n. L(e) = 1 and OP(e) = 0
  - Case e = x. L(e) = 1 and OP(e) = 0
  - Case e = $e_1 + e_2$.
    - ✓ $L(e) = L(e_1) + L(e_2)$ and $OP(e) = OP(e_1) + OP(e_2) + 1$
    - ✓ By induction hypothesis $L(e_1) = OP(e_1) + 1$ and $L(e_2) = OP(e_2) + 1$
    - ✓ Thus $L(e) = OP(e) + 1$
  - Case e = $e_1 * e_2$: same as the case for +.

$\rightsquigarrow_{Aexp}$ is Deterministic

- Structural induction using induction hypothesis:
  $P(a) = \forall \sigma,m,m'.\ \langle a, \sigma\rangle \rightsquigarrow m \ \&\ \langle a, \sigma\rangle \rightsquigarrow m' \Rightarrow m=m'$
- $a \equiv n$
  There is only one rule for evaluating numbers: $n = m = m'$
- $a \equiv X$
  There is only one rule for evaluating locations: $\sigma(X) = m = m'$
- $a \equiv a_0 + a_1$ [similar for $(a_0 - a_1)$ and $(a_0 \times a_1)$]
  If $\langle a, \sigma\rangle \rightsquigarrow m,m'$ then by the + rule there must exist m_0, m_1 where
  $\langle a_0, \sigma\rangle \rightsquigarrow m_0$ and $\langle a_1, \sigma\rangle \rightsquigarrow m_1$ where $m=m_0+m_1$
  and also exist m'_0, m'_1 where
  $\langle a_0, \sigma\rangle \rightsquigarrow m'_0$ and $\langle a_1, \sigma\rangle \rightsquigarrow m'_1$ where $m=m'_0+m'_1$
  By the induction hypothesis applied to a_0 and a_1 we have
  $m_0 = m'_0$ and $m_1=m'_1$. Therefore $m = m_0+m_1 = m'_0+m'_1 = m'$.

$\rightarrow_{Aexp}$ è totale

- Dimostriamo per induzione strutturale che la valutazione di espressioni aritmetiche termina sempre. Cioè, vogliamo provare che $\rightarrow_{Aexp}$ è totale (si ricordi che σ è totale per definizione).
- La precedente proprietà può essere formalmente scritta come

$$P(a)= \forall\sigma, \exists m. \langle a,\sigma\rangle \rightarrow m$$

- La prova per induzione tiene conto dei vari casi
- $\mathbf{a \equiv n}$: L'esistenza è dimostrata prendendo m=n
- $\mathbf{a \equiv X}$: Siccome σ è una funzione totale, $\sigma(X)$ esiste sempre e quindi prendiamo $m=\sigma(X)$
- $\mathbf{a \equiv (a_0 + a_1)}$: Per ipotesi induttiva, esistono m_0 e m_1 tale che $(a_0,m_0) \rightarrow m_0$ e $(a_1,m_1) \rightarrow m_1$. Dunque basta prendere $m=m_0+m_1$
- Un ragionamento simile al caso precedente può essere utilizzato per $\mathbf{(a_0 - a_1)}$ e $\mathbf{(a_0 \times a_1)}$

Induzione ben fondata

- Induzione Matematica e Strutturale sono casi particolari di un più generale principio di prova chiamato ***induzione ben fondata***
- L'induzione strutturale è efficiente perché il processo di scomporre un'espressione in sottoespressione non può procedere all'infinito.
- Estendiamo questo concetto all'induzione ben fondata introducendo il concetto di ***relazione ben fondata***

Relazione ben fondata

- Una relazione binaria $\prec$ su un insieme *A* è ***ben fondata*** se
  1. non esistono catene discendenti infinite $\ldots \prec a_i \prec \ldots a_1 \prec a_0$,

     oppure,
  2. se ogni sottoinsieme Q di *A* ha un elemento minimale, cioè un elemento m di Q tale che $\forall\, b \prec m.\ b \notin Q$ (**Provate che 1 ≡ 2** soluzione su Winskel)
- Se $a \prec b$ si dice che a è un predecessore di b.
- Si noti che $\prec$ è irriflessiva, (cioè, $a \prec a$ non vale per nessun a).
- Esempi:
  - Sui numeri naturali la relazione "**maggiore**" è ben formata, ma "**maggiore uguale**" non lo è perché ammette catene infinite
  - Su Bexp, la relazione "$\prec$" **sottoespressione** è ben formata, per esempio $a \leq 5 \prec (a \leq 5) \wedge (b \leq 4)$

Principio di Induzione ben fondata

- ***Principio di induzione:*** Se P(x) è vera, assumendo che P(y) è vera per ogni $y \prec x$ allora P(a) è vera per ogni a in *A*
- Esempio 1: Si consideri $\prec$ come la relazione **successore n $\prec$ m sse m=n+1** sugli interi non negativi. Il principio di induzione ben fondata coincide con l'induzione matematica
  - Dominio sintattico: **numeri naturali**
  - Relazione ben fondata: **Essere successore di**
- Esempio 2: Si consideri $\prec$ come la relazione tra espressioni aritmetiche tale che **a $\prec$ b sse a è una immediata sotto-espressione di b**, il principio di induzione ben fondata coincide con l'induzione strutturale
  - Dominio sintattico: ***Aexp***
  - Relazione ben fondata: **Essere sottoespressione di**

Uso dell'induzione ben fondata

- L'induzione ben fondata è il più importante principio per la dimostrazione della terminazione dei programmi.
- Solitamente, in un programma le incertezze riguardo la terminazione nascono a causa dei cicli o della ricorsione.
- Se si dimostra che l'esecuzione di un ciclo o la ricorsione in un programma decrementa un valore in un insieme ben fondato, alla fine il programma deve terminare
- Esercizio da svolgere:
  - Si consideri il seguente programma di Euclide per il MCD scritto in IMP

    $E \equiv$ While $\neg(M=N)$ do if $M \leq N$ then N:=N-M else M:=M-N
  - Si provi per induzione ben fondata che per ogni stato σ

    $\sigma(M) \geq 1 \;\&\; \sigma(N) \geq 1 \Rightarrow \exists\, \sigma' \,.\, \langle E,\sigma\rangle \rightarrow \sigma'$

Intermezzo

- L'induzione matematica è utile per provare proprietà sui numeri.
  - Una proprietà P(n) è vera per tutti i numeri naturali n se P(0) è vera (base) e se, per qualsiasi m, P(m) vera implica P(m+1) vera.
  - Con questo metodo abbiamo provato che $\forall e \in Aexp.\ L(e) = OP(e) + 1$
- L'induzione strutturale è una tecnica utile per provare proprietà di insiemi ***sintattici***, usando una prova che segue dalla struttura della grammatica che definisce l'insieme stesso.
  - Una proprietà P(x) è vera per tutti gli elementi x dell'insieme se è vera per gli oggetti elementari (che non hanno sottotermini) dell'insieme e se, per un certo termine s, P(r) vale per tutti gli immediati sottotermini r di s.
  - Con questo metodo abbiamo provato che $\rightarrow_{Aexp}$ è deterministica
- E per le proprietà semantiche…

Induzione sulle derivazioni

- A volte, l'induzione strutturale non è sufficiente per provare proprietà della semantica operazionale (ad esempio quando non esiste una relazione ben fondata su sottoespressioni)
  - Per esempio, con la sola induzione strutturale non è possibile provare che $\rightarrow_{COM}$ è deterministica
- Il motivo è che l'induzione strutturale si basa sulla struttura dei termini, ma esistono termini la cui semantica dipende da termini dello stesso tipo
  - Si pensi per esempio alla semantica del comando **while**
- In questi casi è utile ragionare per induzione sulla struttura delle ***derivazioni*** invece che sulla struttura delle regole (in pratica se non utilizzo l'informazione aggiuntiva della derivazione, i metodi di prova precedente falliscono)
- Una sottoderivazione di una derivazione è una derivazione utilizzata all'interno dell'albero di derivazione.

Esempio di derivazione

- Dato un termine di un linguaggio il suo significato dipende dalle regole di derivazione semantiche
- Per esempio, per valutare il termine y:=y+(3*(5+4)), si costruisce un albero di derivazione utilizzando le regole di valutazione per i numeri, la somma, la moltiplicazione ed infine per l'assegnamento di variabili che restituisce la sua valutazione (il nuovo stato).
- Per valutare l'assegnamento dunque si considera la regola dell'assegnamento su una **istanza** della regola.
- Si ricorda che una istanza di una regola è formata da un insieme di **premesse** (x_1,... x_n tutte istanziate) e una **conclusione (y)**. Per cui una istanza è solitamente indicata con una coppia ($\{x_1,...x_n\}/y$).
- Per esempio, sia $\sigma_0(x)=0$ per ogni locazione x e $\sigma=\sigma_0[27/x]$, una istanza della regola di assegnazione è

$$\frac{\langle y+(3*(5+4)),\sigma_0\rangle \rightarrow 27}{\langle y:=y+(3*(5+4)),\sigma_0\rangle \rightarrow \sigma}$$

Derivazione di un termine

- Data una istanza di una regola ($\{x_1,\ldots x_n\}/y$), una **derivazione d** di y è una coppia ($\{d_1,\ldots d_n\}/y$) tale che d_i è una derivazione di x_i ($d_i \Vdash x_i$) per ogni i.
- In pratica, una derivazione **d** è ottenuta dall'albero di derivazione guardando solo ai primi due livelli.
- Chiaramente, per un assioma ($\emptyset/y$), l'istanza dell'assioma coincide con la derivazione. Formalmente $d=(\emptyset/y) \Vdash y$
- Talvolta si omette "d" e si usa $\Vdash y$ per indicare che esiste una derivazione di un termine y
  - $\Vdash \langle c,\sigma\rangle \rightarrow \sigma'$ significa che $\langle c,\sigma\rangle \rightarrow \sigma'$ è derivabile nella semantica operazionale

$\rightsquigarrow_{Com}$ is Deterministic

- Theorem 3.11
  $P(c) = \forall \sigma_0,\sigma,\sigma'.\ \langle c, \sigma_0\rangle \rightsquigarrow \sigma \ \&\ \langle c, \sigma_0\rangle \rightsquigarrow \sigma' \Rightarrow \sigma=\sigma'$
- Derivation d of execution
  $d \Vdash \langle c, \sigma\rangle \rightsquigarrow \sigma'$
  - $\langle c, \sigma_0\rangle$ terminates with state σ
- Induction hypothesis
  $P(\mathbf{d}) = \forall \sigma_0,\sigma,\sigma'.\ \mathbf{d} \Vdash \langle c, \sigma_0\rangle \rightsquigarrow \sigma \ \&\ \langle c, \sigma_0\rangle \rightsquigarrow \sigma' \Rightarrow \sigma=\sigma'$
- Define $d' \prec d$ if d' is a sub-derivation of d

$$\frac{\forall d' \prec d.\ P(d') \Rightarrow P(d)}{\forall d.\ P(d)}$$

$\rightarrow_{Com}$ è deterministica

- Teorema

  $P(c) = \forall\, \sigma_0, \sigma_1, \sigma', \langle c, \sigma_0 \rangle \rightarrow \sigma \;\&\; \langle c, \sigma_0 \rangle \rightarrow \sigma'$ then $\sigma=\sigma'$
- Base della prova: mostrare l'asserto è valido per gli assiomi.
- Passo induttivo: mostrare che se l'asserto è valido per le sottoderivazioni, allora è valido per la derivazione finale.
- Dunque, dobbiamo provare una proprietà di un comando sulle derivazioni.

$\rightsquigarrow_{Com}$ is Deterministic

◆Given

$$P(d) = \forall \sigma_0,\sigma,\sigma'. d \Vdash \langle c, \sigma_0\rangle \rightsquigarrow \sigma \;\&\; \langle c, \sigma_0\rangle \rightsquigarrow \sigma' \Rightarrow \sigma=\sigma'$$

◆Show

$$\forall d' \prec d.\ P(d') \Rightarrow P(d)$$

◆Show by cases that

$$\frac{d \Vdash \langle c, \sigma_0\rangle \rightsquigarrow \sigma \quad \forall d' \prec d.\ P(d') \quad \exists d_1 .\ d_1 \Vdash \langle c, \sigma_0\rangle \rightsquigarrow \sigma'}{\sigma=\sigma'}$$

Ho bisogno di questa informazione aggiuntiva per provare che Com è det.!!!

$\rightsquigarrow_{Com}$ is Deterministic

- $c \equiv$ **skip**
  - $d = d_1 = \langle skip, \sigma_0 \rangle \rightsquigarrow \sigma_0$
- $c \equiv X := a$

$$d = \frac{\vdots}{\langle a, \sigma_0 \rangle \rightsquigarrow n \quad \langle X := a, \sigma_0 \rangle \rightsquigarrow \sigma_0[n/X]} \qquad d_1 = \frac{\vdots}{\langle a, \sigma_0 \rangle \rightsquigarrow n_1 \quad \langle X := a, \sigma_0 \rangle \rightsquigarrow \sigma_0[n_1/X]}$$

  - Since $n=n_1$ we have $\sigma = \sigma_0[n/X] = \sigma_0[n_1/X]$

Abbiamo già provato che la valutazione di Aexp è deterministica

$\rightsquigarrow_{Com}$ is Deterministic

- $c \equiv c_0; c_1$

$$d^0 = \frac{\vdots}{\langle c_0, \sigma_0 \rangle \rightsquigarrow \sigma_1} \qquad d^1 = \frac{\vdots}{\langle c_1, \sigma_1 \rangle \rightsquigarrow \sigma} \qquad d = \frac{d^0 \quad d^1}{\langle c_0; c_1, \sigma_0 \rangle \rightsquigarrow \sigma}$$

$$d_1 = \frac{\vdots \quad \vdots}{\langle c_0, \sigma_0 \rangle \rightsquigarrow \sigma_1' \quad \langle c_1, \sigma_1' \rangle \rightsquigarrow \sigma'}{\langle c_0; c_1, \sigma_0 \rangle \rightsquigarrow \sigma'}$$

- By induction hypothesis $P(d^0)$ and $P(d^1)$
  So, $\sigma_1=\sigma_1'$, and $\sigma=\sigma'$
- Similar for **if** and **while**