Teorema: Decidibilità ed Enumerazione Ordinata

Teorema Principale

Teorema: Un linguaggio L è decidibile se e solo se esiste un enumeratore che enumera L secondo l'ordinamento standard delle stringhe.

Formalmente: $L \in R \iff \exists E$: E enumera L in ordine lessicografico

Definizioni Preliminari

Enumeratore

Un **enumeratore** E è una TM con un nastro di output che:

- Non ha input
- Stampa stringhe separate da delimitatori sul nastro di output
- L(E) = {w | E stampa w sul nastro di output}

Ordinamento Standard

L'**ordinamento standard** (lessicografico) su Σ^* è definito da:

1. $|w_1| < |w_2| \implies w_1 <_{lex} w_2$ (ordinamento per lunghezza)
2. $|w_1| = |w_2| \implies w_1 <_{lex} w_2$ sse w_1 precede w_2 nell'ordine dizionario

Esempio ($\Sigma = \{0,1\}$): ε, 0, 1, 00, 01, 10, 11, 000, 001, 010, 011, 100, 101, 110, 111, ...

Dimostrazione

Direzione ($\implies$): Decidibile $\implies$ Enumeratore Ordinato

Dato: L decidibile, quindi $\exists$ TM M tale che M decide L
Tesi: $\exists$ enumeratore E che enumera L in ordine lessicografico

Costruzione dell'Enumeratore E

```
Enumeratore E:
1. Genera tutte le stringhe in ordine lessicografico
2. Per ogni stringa w generata:
   a. Simula M(w)
   b. Se M accetta w, stampa w
   c. Continua con la prossima stringa
```

Algoritmo Formale

```
E():
 for n = 0, 1, 2, ... do
  for ogni stringa w ∈ Σ^n in ordine lessicografico do
   simula M(w)
   if M accetta w then
    stampa w
   endif
  endfor
 endfor
```

Correttezza della Costruzione

Lemma 1: E enumera esattamente L
Dimostrazione:

- E stampa w $\iff$ M(w) accetta $\iff$ w $\in$ L (per definizione di M)

Lemma 2: E enumera in ordine lessicografico
Dimostrazione:

- Le stringhe sono generate per lunghezza crescente
- All'interno di ogni lunghezza, sono generate in ordine lessicografico
- M termina sempre (L è decidibile), quindi nessuna stringa blocca l'enumerazione

Lemma 3: E termina la generazione di ogni stringa in tempo finito
Dimostrazione:

- Per ogni w, M(w) termina in tempo finito (M è un decisore)
- Quindi E procede sempre alla stringa successiva

Direzione ($\Longleftarrow$): Enumeratore Ordinato $\Longrightarrow$ Decidibile

Dato: $\exists$ enumeratore E che enumera L in ordine lessicografico
Tesi: L è decidibile

Costruzione del Decisore M

```
Decisore M:
Input: stringa w
1. Simula E finché non stampa una stringa s tale che s ≥lex w
2. If s = w then accetta
3. Else rifiuta
```

Algoritmo Formale

```
M(w):
 repeat
  sia s la prossima stringa stampata da E
  if s = w then
   accetta
  endif
  if s >lex w then
   rifiuta
  endif
 endrepeat
```

Correttezza della Costruzione

Lemma 4: M termina sempre
Dimostrazione:

- E enumera in ordine lessicografico crescente
- Per ogni w, esisterà sempre una stringa $s \geq_{lex} w$ nella sequenza
- Quindi il ciclo termina sempre

Lemma 5: M decide L correttamente
Dimostrazione:

- **Caso $w \in L$**: E stamperà w in posizione corretta, M accetta
- **Caso $w \notin L$**: E non stamperà mai w, ma stamperà una stringa $s >_{lex} w$, M rifiuta

Lemma 6: L'ordinamento garantisce la correttezza
Dimostrazione: Se $w \notin L$ e E enumera $s >_{lex} w$, allora w non può apparire successivamente perché E enumera in ordine crescente.

Conseguenze Teoriche

Corollario 1: Caratterizzazione Alternativa

Corollario: $L \in R \iff$ L è ricorsivamente enumerabile e co-r.e.

Dimostrazione:

- Il nostro teorema fornisce una costruzione diretta
- Un enumeratore ordinato permette sia di verificare appartenenza che non-appartenenza

Corollario 2: Separazione da R.E.

Corollario: Esistono linguaggi r.e. che non sono decidibili

Dimostrazione:

- Alcuni linguaggi r.e. hanno enumeratori che non possono essere resi ordinati
- Esempio: il linguaggio delle TM che terminano su input vuoto

Corollario 3: Ottimalità

Corollario: L'ordinamento è necessario - un enumeratore non ordinato non garantisce decidibilità

Controesempio: K = {⟨M⟩ | M(⟨M⟩) accetta} ha un enumeratore ma non ordinato.

Esempio Applicativo

Linguaggio: L = {ww^R | w ∈ {0,1}*}

Enumeratore Ordinato:

```
E():
  for n = 0, 2, 4, ... do  // solo lunghezze pari
    for ogni w ∈ {0,1}^(n/2) in ordine lex do
      stampa ww^R
    endfor
  endfor
```

Decisore Derivato:

```
M(x):
  if |x| è dispari then rifiuta
  let x = x_1 x_2 ... x_2k
  if x_1 ... x_k = x_k+1 ... x_2k^R then accetta
  else rifiuta
```

Complessità della Costruzione

Analisi Temporale

- **Enumeratore** → **Decisore**: O(2^|w|) nel caso peggiore
- **Decisore** → **Enumeratore**: Tempo polinomiale per stringa generata

Ottimalità della Costruzione

La costruzione è **ottimale** nel senso che:

1. Non esiste una costruzione più efficiente in generale
2. L'ordinamento lessicografico è l'unico ordinamento totale computabile che garantisce la proprietà

Importanza Teorica

Questo teorema stabilisce che:

1. **Decidibilità** ≡ **Enumerabilità ordinata**
2. Fornisce un bridge tra **riconoscimento** ed **enumerazione**
3. Dimostra che l'**ordinamento** è il fattore critico che separa R da R.E.