Esercizio 1 - Indecidibile

2. **(12 punti)** I *grawlix* sono sequenze di simboli senza senso che sostituiscono le parolacce nei fumetti.

Un linguaggio è *volgare* se contiene almeno un grawlix. Considera il problema di determinare se il linguaggio di una TM è volgare.

(a) Formula questo problema come un linguaggio $GROSS_{TM}$.

(b) Dimostra che il linguaggio $GROSS_{TM}$ è indecidibile.

Risoluzione Formale

(a) Formulazione del linguaggio GROSS_TM

Definizione: Un *grawlix* è una stringa di simboli che rappresenta una parolaccia nei fumetti (es. "#$@!", "£*#×+!").

Linguaggio GROSS_TM:

```
GROSS_TM = {⟨M⟩ | M è una macchina di Turing e L(M) contiene almeno un grawlix}
```

(b) Dimostrazione dell'indecidibilità di GROSS_TM

Teorema: GROSS_TM è indecidibile.

Dimostrazione: Mostreremo che ATM ≤m GROSS_TM tramite riduzione many-to-one.

Costruzione della riduzione: Dato ⟨M,w⟩ come input per ATM, costruiamo la seguente macchina M':

```
M' = "Su input x:
    1. Ignora x
    2. Simula M su input w
    3. Se M accetta w, accetta x
    4. Se M rifiuta w, rifiuta x"
```

Analisi della riduzione:

- **Caso 1:** Se ⟨M,w⟩ ∈ ATM (M accetta w) → M' accetta ogni stringa, quindi L(M') = Σ* → L(M') contiene tutti i possibili grawlix → ⟨M'⟩ ∈ GROSS_TM
- **Caso 2:** Se ⟨M,w⟩ ∉ ATM (M non accetta w) → M' non accetta mai, quindi L(M') = ∅ → L(M') non contiene alcun grawlix → ⟨M'⟩ ∉ GROSS_TM

Conclusione: ⟨M,w⟩ ∈ ATM ⟺ ⟨M'⟩ ∈ GROSS_TM

Poiché ATM è indecidibile e abbiamo una riduzione computabile da ATM a GROSS_TM, ne segue che GROSS_TM è indecidibile. □

Scaletta Risolutiva

Schema Generale per Problemi di Indecidibilità su Proprietà di Linguaggi

1. **Identificazione della proprietà semantica:**
   - Verificare se la proprietà riguarda il contenuto/comportamento del linguaggio riconosciuto
   - Distinguere da proprietà sintattiche (struttura della macchina)
2. **Scelta della tecnica di dimostrazione:**
   - **Riduzione da ATM:** Per proprietà non-triviali dei linguaggi
   - **Teorema di Rice:** Se la proprietà è semantica e non-triviale
   - **Riduzione dal Halting Problem:** Per proprietà temporali
3. **Costruzione della macchina ausiliaria:**
   - Progettare M' che dipende dal risultato di M su w
   - M' deve soddisfare la proprietà ⟺ M accetta w
4. **Verifica della correttezza:**
   - Dimostrare entrambe le direzioni dell'equivalenza
   - Assicurarsi che la riduzione sia computabile

Esercizio 2 - Indecidibile

4. **(9 punti)** Considera il seguente problema: data una TM M a nastro semi-infinito, determinare se esiste un input w su cui M sposta la testina a sinistra partendo dalla cella numero 2023 (ossia se in qualche momento durante la computazione la testina si muove dalla cella 2023 alla cella 2022).

   (a) Formula questo problema come un linguaggio 2023_{TM}.

   (b) Dimostra che il linguaggio 2023_{TM} è indecidibile.

(a) Formulazione del linguaggio 2023_TM

Definizione:

```
2023_TM = {⟨M⟩ | M è una TM a nastro semi-infinito e
           esiste un input w tale che durante la computazione
           di M su w la testina si sposta dalla cella 2023 alla cella 2022}
```

(b) Dimostrazione dell'indecidibilità di 2023_TM

Teorema: 2023_TM è indecidibile.

Dimostrazione: Mostreremo che A_TM ≤_m 2023_TM.

Costruzione della riduzione

Data la coppia ⟨M,w⟩ come input per A_TM, costruiamo la seguente macchina M':

```
M' = "Su input x:
    1. Simula M su input w
    2. Se M accetta w:
       a. Sposta la testina alla cella 2023
       b. Esegui una mossa L (sinistra) → testina va in cella 2022
       c. Accetta x
    3. Se M rifiuta w:
       a. Rifiuta x senza mai raggiungere la cella 2023"
```

Analisi dei casi

Caso 1: ⟨M,w⟩ ∈ A_TM (M accetta w)

```
⇒ M' simula M su w e la simulazione termina con accettazione
⇒ M' esegue il punto 2: sposta testina da cella 2023 a cella 2022
⇒ Esiste un input (qualsiasi input a M') per cui si verifica lo spostamento
richiesto
⇒ ⟨M'⟩ ∈ 2023_TM
```

Caso 2: ⟨M,w⟩ ∉ A_TM (M non accetta w)

Sottocaso 2a: M rifiuta w

```
⇒ M' simula M su w e la simulazione termina con rifiuto
⇒ M' esegue il punto 3: rifiuta senza raggiungere mai la cella 2023
⇒ Per nessun input la testina si sposta da cella 2023 a cella 2022
⇒ ⟨M'⟩ ∉ 2023_TM
```

Sottocaso 2b: M va in loop su w

```
⇒ M' simula M su w e la simulazione non termina mai
⇒ M' non raggiunge mai il punto 2 né il punto 3
⇒ La testina non arriva mai alla cella 2023
⇒ ⟨M'⟩ ∉ 2023_TM
```

Verifica della correttezza

Equivalenza bidirezionale:

```
⟨M,w⟩ ∈ A_TM ⟺ ⟨M'⟩ ∈ 2023_TM
```

Computabilità della riduzione:

- La costruzione di M' è algoritmica e deterministica
- Non richiede la risoluzione di A_TM
- La riduzione è computabile in tempo polinomiale

Conclusione

Poiché A_TM è indecidibile e abbiamo mostrato una riduzione computabile A_TM ≤_m 2023_TM, segue che **2023_TM è indecidibile**. □

Note tecniche

La dimostrazione sfrutta il fatto che il movimento specifico della testina (dalla cella 2023 alla 2022) può essere controllato condizionalmente rispetto all'accettazione di M su w, creando una corrispondenza diretta tra il problema dell'accettazione e la proprietà geometrica richiesta sul nastro.

Esercizio 3 - Indecidibili

2. **(12 punti)** Considera il linguaggio

$$\text{EVEN-HALTS} = \{\langle M \rangle \mid \text{per ogni numero naturale } n \text{ pari, } M \text{ termina la computazione su } n\}.$$

Dimostra che EVEN-HALTS è indecidibile.

Risoluzione Formale

Teorema: EVEN-HALTS è indecidibile

Dimostrazione: Mostreremo che HALT_TM ≤_m EVEN-HALTS.

Costruzione della riduzione

Data la coppia ⟨M,w⟩ come input per HALT_TM, costruiamo la seguente macchina M':

```
M' = "Su input x:
    1. Converti x in formato numerico
    2. Se x non è un numero naturale pari, rifiuta
    3. Se x è un numero naturale pari, simula M su input w
    4. Se M termina (accetta o rifiuta w), termina accettando x
    5. Se M va in loop su w, va in loop"
```

Analisi dei casi

Caso 1: ⟨M,w⟩ ∈ HALT_TM (M termina su w)

```
⇒ Per ogni numero pari n: M' simula M su w al passo 3
⇒ La simulazione termina (perché M termina su w)
⇒ M' termina accettando n al passo 4
⇒ M' termina su tutti i numeri pari
⇒ ⟨M'⟩ ∈ EVEN-HALTS
```

Caso 2: ⟨M,w⟩ ∉ HALT_TM (M non termina su w)

```
⇒ Per ogni numero pari n: M' simula M su w al passo 3
⇒ La simulazione non termina mai (perché M non termina su w)
⇒ M' va in loop al passo 5 per ogni numero pari n
⇒ M' non termina su nessun numero pari
⇒ ⟨M'⟩ ∉ EVEN-HALTS
```

Verifica della correttezza

Equivalenza bidirezionale:

```
⟨M,w⟩ ∈ HALT_TM ⟺ ⟨M'⟩ ∈ EVEN-HALTS
```

Computabilità della riduzione:

- La costruzione di M' è algoritmica e deterministica
- Richiede solo di combinare il codice di M con controlli sui numeri pari
- Non richiede la risoluzione di HALT_TM

- La riduzione è computabile in tempo polinomiale

Conclusione

Poiché HALT_TM è indecidibile e abbiamo mostrato una riduzione computabile HALT_TM ≤_m EVEN-HALTS, segue che **EVEN-HALTS è indecidibile**. □

Note tecniche

La dimostrazione sfrutta il fatto che possiamo condizionare il comportamento di M' su tutti i numeri pari rispetto alla terminazione di M su un singolo input w, creando una corrispondenza diretta tra il problema del halting e la proprietà universale richiesta da EVEN-HALTS.

Esercizio 4 - NP-Hard

> **3. (12 punti)** In una delle storie delle Mille e una notte, Alì Babà, mentre viaggiava con il suo asino, trovò la grotta in cui i 40 ladroni avevano nascosto il loro bottino. Come cittadino rispettoso della legge, denunciò il fatto alla polizia, ma solo dopo aver tenuto il più possibile per sé. Il problema è che c'è troppo bottino e l'asino non può portarlo tutto: c'è un limite M al peso che l'asino può trasportare. Supponiamo che ognuno degli N oggetti rubati abbia un prezzo $P[i]$ e un peso $W[i]$. Alì Babà può caricare sull'asino un numero sufficiente di oggetti in modo che il prezzo totale sia almeno L?
>
> Formalmente, possiamo rappresentare il problema che Alì Babà deve risolvere con il linguaggio
>
> $$ALIBABA = \left\{\langle N, P, W, M, L\rangle \;\middle|\; \text{esiste } B \subseteq \{1, \dots, N\} \text{ tale che } \sum_{j \in B} W[j] \leq M \text{ e } \sum_{j \in B} P[j] \geq L\right\}.$$
>
> (a) Dimostra che $ALIBABA$ è un problema NP.
>
> (b) Sappiamo che il linguaggio
>
> $$SUBSET\text{-}SUM = \left\{\langle S, t\rangle \;\middle|\; S \text{ insieme di naturali, ed esiste } S' \subseteq S \text{ tale che } \sum_{x \in S'} x = t\right\}$$
>
> è NP-completo. Dimostra che $ALIBABA$ è NP-hard, usando $SUBSET\text{-}SUM$ come problema NP-hard di riferimento.

(a) Dimostrazione che ALIBABA ∈ NP

Teorema: ALIBABA è un problema in NP.

Dimostrazione: Definiamo un algoritmo di verifica polinomiale:

Certificato: Un sottoinsieme B ⊆ {1,...,N}

Algoritmo di verifica V:

```
V(⟨N,P,W,M,L⟩, B):
1. Verifica che B ⊆ {1,...,N}
```

```
2. Calcola peso_totale = Σ W[j] per j ∈ B
3. Calcola prezzo_totale = Σ P[j] per j ∈ B
4. Accetta sse peso_totale ≤ M e prezzo_totale ≥ L
```

Correttezza:

- Se ⟨N,P,W,M,L⟩ ∈ ALIBABA, allora esiste B che soddisfa i vincoli, e V(input,B) = accetta
- Se V(input,B) = accetta per qualche B, allora B testimonia che input ∈ ALIBABA

Complessità: O(|B|) = O(N), quindi polinomiale.

Pertanto **ALIBABA** ∈ **NP**. □

(b) Dimostrazione che ALIBABA è NP-hard

Teorema: ALIBABA è NP-hard.

Dimostrazione: Mostreremo che SUBSET-SUM ≤_p ALIBABA.

Costruzione della riduzione

Data un'istanza ⟨S,t⟩ di SUBSET-SUM dove $S = \{s_1,...,s_n\}$ e target t, costruiamo l'istanza f(S,t) di ALIBABA:

```
f(S,t) = ⟨N,P,W,M,L⟩ dove:
- N = n (stesso numero di elementi)
- Per ogni i = 1,...,n:
  * P[i] = sᵢ (prezzo = valore elemento)
  * W[i] = sᵢ (peso = valore elemento)
- M = t (limite peso = target)
- L = t (soglia prezzo = target)
```

Analisi della correttezza

Lemma: ⟨S,t⟩ ∈ SUBSET-SUM ⟺ f(S,t) ∈ ALIBABA

Dimostrazione del Lemma:

(⟹) Supponiamo ⟨S,t⟩ ∈ SUBSET-SUM.

```
⇒ ∃ S' ⊆ S tale che Σ(S') = t
⇒ Sia B = {i | sᵢ ∈ S'} ⊆ {1,...,n}
⇒ Σ W[j] = Σ sⱼ = Σ(S') = t ≤ M
   j∈B    j∈B
⇒ Σ P[j] = Σ sⱼ = Σ(S') = t ≥ L
```

```
    j∈B     j∈B
⇒ B soddisfa entrambi i vincoli di ALIBABA
⇒ f(S,t) ∈ ALIBABA
```

(⟸) Supponiamo f(S,t) ∈ ALIBABA.

```
⇒ ∃ B ⊆ {1,...,n} tale che:
  • Σ W[j] ≤ M = t
    j∈B
  • Σ P[j] ≥ L = t
    j∈B
⇒ Σ sⱼ ≤ t e Σ sⱼ ≥ t
   j∈B        j∈B
⇒ Σ sⱼ = t (uguaglianza necessaria)
   j∈B
⇒ Sia S' = {sⱼ | j ∈ B} ⊆ S
⇒ Σ(S') = t
⇒ ⟨S,t⟩ ∈ SUBSET-SUM
```

Verifica della polinomialità

La riduzione f è computabile in tempo O(n), quindi polinomiale.

Conclusione

Poiché SUBSET-SUM è NP-completo e abbiamo mostrato SUBSET-SUM ≤_p ALIBABA tramite una riduzione polinomiale, segue che **ALIBABA è NP-hard**. □
