DECISION PROBLEMS, LANGUAGES AND ENCODING SCHEMES

DECISION PROBLEMS

The theory of NP–completeness is designed to be applied to *decision problems* (DP).

DP — either answer YES
— or answer NO

Notation:

Π: decision problem
$\Pi = \{D_\Pi \mid D_\Pi \text{ instance}\}$
$Y_\Pi \subseteq D_\Pi$: subset of YES–instances
$N_\Pi \subseteq D_\Pi$: subset of NO–instances

Specifying decision problems:
- Specifying a generic instance of the problem in terms of components (sets, graphs, etc.).
- Stating a yes – no question in terms of the generic instance.

An instance belongs to D_Π iff it can be obtained from the generic instance by substituting particular objects for all the generic components.

$I \in Y_\Pi$ iff the answer for the particularized question is YES.

<u>Example</u>: Travelling Salesman Problem (TSP)

1. Optimisation problem (OPT)

Given: $C = \{c_1, \ldots, c_m\}$ finite set of cities
$d(c_i, c_j) \in Z^+$ distance, for $\forall c_i, c_j \in C$
Question: Find a tour (ordering) $<c_{(1)}, \ldots, c_{(m)}>$ of C such that

$$\min\left(\sum_{i,j} d\left(c_{(i)}, c_{(j)} \right) \right)$$

2. Decision problem (DP)

Idea: bound $B \in Z^+$

Question: Is there an ordering $<c_{\Pi(1)}, \ldots, c_{\Pi(m)}>$ of C such that

$$\left(\sum_{i=11}^{m-1} d\left(c_{\Pi(i)}, c_{\Pi(i+1)} \right) \right) + d(c_{\Pi(m)}, c_{\Pi(1)}) \leq B$$

<u>Note:</u>

(1) $O(\text{DP}) \leq O(\text{OPT})$
(2) OPT can be transformed into DP.
(3) DP can be expressed using languages, which are suitable objects for computational study from an algorithmic viewpoint.

LANGUAGE

Σ: finite set of symbols, alphabet
Σ^*: set of all finite strings from Σ
L is a language over Σ, if $L \subseteq \Sigma^*$

ENCODING SCHEME

Encoding scheme *e* for a problem Π: a way of describing each instance of Π as a string of symbols over some alphabet Σ.

The language of YES–instances Y_Π:

$L(\Pi,e) = \{x \in \Sigma^*$: Σ is the alphabet used by *e*, and *x* is the encoding under *e* of an instance $I \in Y_\Pi\}$

ENCODING SCHEME

Encoding scheme e maps instances into *structured stings* over alphabet

$$\Psi = \{0, 1, -, [,], (,), ,\}.$$

Structured strings are defined recursively:

(1) The binary representation of an integer k as a string of 0's and 1's (minus sign – if k is negative) is a structured string representing the integer k.

(2) If x is a structured string of an integer k, then $[x]$ is a structured string representing a *name*; e.g. vertex in a graph.

(3) If x_i, $i = \overline{1, m}$ is a structured string representing object X_i, $i = \overline{1, m}$, then $(x_1, \ldots, x_m)$ is a structured string representing sequence $<X_1, \ldots, X_m>$.

ENCODING AN INSTANCE OF A DECISION PROBLEM

- Identify the objects of the instance.
- Represent each object as a structured string.
- Represent the entire instance as a structured string.

Representation of Objects:

- Unstructured elements (vertices, cities, etc.): assign a distinct name to each as constructed by rule (2).
- Set: represent as a sequence $<X_1, \ldots, X_m>$, and take the structured string corresponding to that sequence.
- Graph (V, E): represent as a structured string (x, y) where x is a structured string representing the set V, and y is a structured string representing the set E.
- Rational number $q = \frac{a}{b}$ is represented by a structured string (x, y) where x is a structured string representing an integer a, and y is a structured string representing b.
- Finite function: f: $\{U_1, \ldots, U_m\} \rightarrow W$ is represented by a structured string $((x_1, y_1), \ldots, (x_m, y_m))$ where x_i is a structured string representing object U_i, and y_i is a structured string representing object $f(U_i) \in W$.

Note:

- The encoding is not unique.
- The encoding obeys certain structural constraints.
- Some other encoding scheme could also be used.

REVIEW

OPT

↓ transform

DP

↓ instance

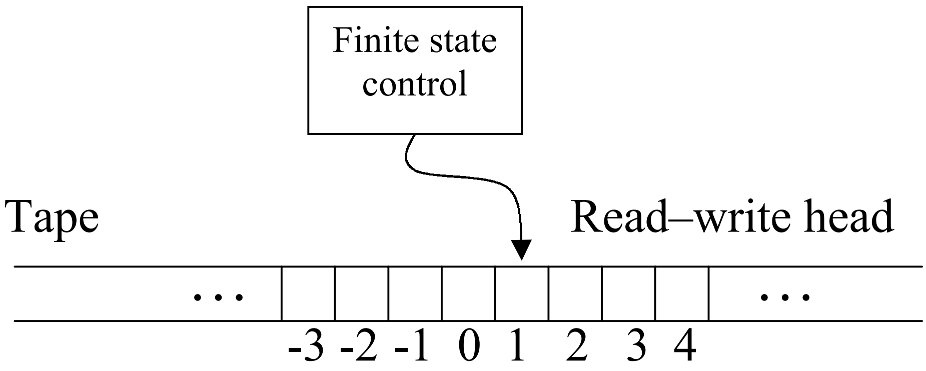
I

↓ encoding scheme *e*

$\boldsymbol{x} \in \boldsymbol{L}(\Pi, \boldsymbol{e})$

<u>Idea</u>: Turing Machine to recognise *x*

DETERMINISTIC TURING MACHINE



A program *M* for DTM specifies the following:

(1) A finite set Γ of *tape symbols*, *input symbols* $\Sigma\subset\Gamma$ and *blank symbol* $b\in\Gamma \setminus \Sigma$.
(2) A finite set Q of *states*; *start state* q_0, and two *halt states* q_Y and q_N.
(3) A *transition function*:

$$\delta : (Q \setminus \{q_Y, q_N\}) \times \Gamma \to Q \times \Gamma \times \{-1,+1\}.$$

A DTM program *M* with alphabet Σ accepts $x\in\Sigma^*$ iff *M* halts in state q_Y.

Language L_M recognised by *M*:

$$L_M = \{x\in\Sigma^*: \text{M accepts } x\}$$

A DTM program *M* solves the decision problem Π under encoding scheme *e* if *M* halts for all input string over its input alphabet and $L_M = L[\Pi, e]$.

PROBLEM

Integer divisibility by four

<u>Instance</u>: a positive integer N.
<u>Question</u>: Is there a positive integer m such that $N = 4m$?

- Encoding scheme: N is represented as a sequence of 0's and 1's.
- N is divisible by four iff the last two digits of its binary representation are 0.

A DTM program that halts if the rightmost two symbols are both 0, DTM recognise the language:

$\{x \in \{0, 1\}^*$: the rightmost two symbols are both $0\}$

$\Gamma = \{0, 1, b\}$
$\Sigma = \{0, 1\}$
$Q = \{q_0, q_1, q_2, q_3, q_Y, q_N\}$

δ	0	1	b
q_0	$(q_0, 0, +1)$	$(q_0, 1, +1)$	$(q_1, b, -1)$
q_1	$(q_2, b, -1)$	$(q_3, b, -1)$	$(q_N, b, -1)$
q_2	$(q_Y, b, -1)$	$(q_N, b, -1)$	$(q_N, b, -1)$
q_3	$(q_N, b, -1)$	$(q_N, b, -1)$	$(q_N, b, -1)$

CLASS P

The reason for introducing the DTM model is to provide a formal counterpart of an algorithm upon which to base further definitions.

Time: time used in the computation of a DTM program M on an input x is the number m of steps occurring until a halt state is entered.

Time complexity T_M: $Z^+ \rightarrow Z^+$ of a program M that halts for all inputs $x \in \Sigma^*$:

$$T_M(n) = \max \{m \mid \exists x \in \Sigma^*, |x| = n, \text{time}(M, x) = m\}$$

M is a polynomial time DTM program if $\exists$ polynomial p such that for $\forall n \in Z^+$, $T_M(n) \leq p(n)$.

The Class P is defined as follows:

$$\text{P} = \{L: \exists \text{ a polynomial time DTM program } M \text{ for which } L = L_M\}$$

or

$$\Pi \in \text{P} \Leftrightarrow L(\Pi, e) \in \text{P}$$