



UNIVERSIDAD NACIONAL DE SANTIAGO DEL ESTERO
Facultad de Ciencias Exactas y Tecnologías
Licenciatura en Sistemas de Información



PROGRAMACIÓN II

EQUIPO CÁTEDRA

MSc. Ing. Margarita María Álvarez
Lic. Paola Budán

2011

OBJETIVOS

- Identificar los conceptos constructivos de la Teoría de Algoritmos.
- Habilidad para formalizar problemas.
- Capacidad para determinar la eficiencia de los algoritmos
- Habilidad para generar y reconocer gramáticas y lenguajes de distinto tipo.
- Determinar las equivalencias entre los lenguajes formales, las gramáticas formales, las expresiones regulares y las máquinas reconocedoras
- Capacidad para construir compiladores/intérpretes sencillos.

PRINCIPALES TEMAS A ABORDAR



Teoría de la Computabilidad.

Formalización, representación y solución de problemas.
Función computable. Complejidad y eficiencia de algoritmos.
Clasificación de problemas.



Teoría de los Lenguajes Formales.

Definición y conceptos. Gramáticas de estructura de frase.
Jerarquía de los lenguajes y gramáticas.
Características de las gramáticas.



Teoría de Automatas.

Automatas finitos. Automata de Pila. Máquina de Turing.
Relaciones entre automatas y lenguajes.



Diseño de Compiladores.

Analizador lexicográfico. Analizador Sintáctico. Analizador Semántico.
Generación de código intermedio. Optimización de código intermedio.
Generación de código objeto.

REQUISITOS PARA PROMOCIONAR O REGULARIZAR LA ASIGNATURA

PARA PROMOCIONAR

- Aprobar los talleres 1, 2 y 3
- Aprobar la Actividad de Formación Experimental
- Obtener una calificación ≥ 70 puntos en cada Taller (1,2 y 3 y el taller de lenguaje C)
- Obtener una calificación ≥ 70 puntos en cada parcial
- Coloquio final sobre temas a consignar

PARA REGULARIZAR

- Aprobar los dos parciales o sus instancias recuperatorias
- Presentar y aprobar la Actividad de Formación Experimental





**TEORÍA DE LA
COMPUTABILIDAD**

TEORÍA DE LA COMPUTABILIDAD

Según T. Kuhn en distintos momentos de la evolución de una ciencia toman relevancia las Teorías Nucleares.

Si se quiere llegar a una comprensión de la Informática, o más precisamente de la Ciencia de la Computación, es necesario abordar su Núcleo Teórico, que es la **Teoría de la Computabilidad**.

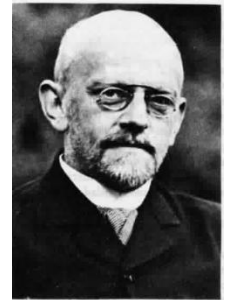
TEORÍA DE LA COMPUTABILIDAD. ORÍGENES (I)

El desarrollo formal de la teoría de la computación se originó hace casi setenta años, a partir de los trabajos de, entre otros, **Hilbert, Gödel, Church, Turing y Kleene.**

¿Existirá un *Algoritmo Universal* capaz de resolver cualquier problema matemático, filosófico, o en general de carácter intelectual?

David Hilbert formula el **Entscheidungsproblem** o **problema de la decisión** que trata de descubrir un método general para decidir si una fórmula lógica es verdadera o falsa.

La meta de Hilbert era crear un sistema matemático formal "completo" y "consistente", en el que todas las aseveraciones pudieran plantearse con precisión. Su idea era encontrar un algoritmo que determinara la verdad o falsedad de cualquier proposición en el sistema formal.



David Hilbert
(1862-1943)

TEORÍA DE LA COMPUTABILIDAD. ORÍGENES (II)



Kurt Gödel
(1906-1978)

En 1931, el matemático austriaco Kurt Gödel publica su famoso **teorema de incompletitud** que establece toda formulación axiomática consistente en la teoría de números contiene proposiciones indecidibles".

Esto acaba con las esperanzas de los matemáticos de crear un *sistema completo y consistente* donde fuera posible demostrar o negar cualquier teorema.



Alan Turing
(1912-1954)

Alan Turing, 1937, publicó un trabajo sobre números calculables que puede considerarse en parte como el origen de la Informática Teórica.(Sus primeras publicaciones científicas aparecen cuando aún no había cumplido los 25 años).

Introdujo la **máquina de Turing** (MT) como un modelo matemático abstracto que permitió formalizar el concepto de algoritmo.

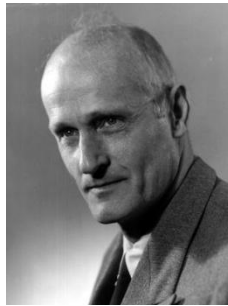
La noción Turing-computabilidad puede considerarse como la base de **la programación imperativa**.

TEORÍA DE LA COMPUTABILIDAD. ORÍGENES (III)



Alonzo Church
(1903-1995)

Su obra más conocida es el desarrollo del **cálculo lambda**, y su trabajo de 1936 que muestra la existencia de problemas indecidibles. La tesis de Alonzo Church fue, básicamente, conjeturar que cualquier función computable podía escribirse en términos de ciertos operadores aritméticos (más precisamente, el cero, el sucesor de un número, la composición, la recursión primitiva y la minimización). Las funciones escritas con tales operadores recibieron el nombre de **funciones recursivas generales**.



Stephen Kleene
(1909 - 1994)

Kleene, 1938, presenta la teoría de las funciones μ - recursivas, que basan su mecanismo de cómputo en la composición de funciones y no en la transición entre estados (programación funcional). Se especializó en el estudio de las funciones recursivas y la teoría de los autómatas.

TEORÍA DE LA COMPUTABILIDAD- CUESTIONES (I)

La **Teoría de la Computabilidad** se ocupa de construir un formalismo matemático para razonar sobre la existencia o no existencia de ***algoritmos efectivos*** para ***problemas*** particulares. Los resultados que se prueben dentro de esta teoría deben ser aplicables a todas las arquitecturas de ordenadores, independientemente de sus parámetros, como pueden ser la velocidad del procesador o el tamaño de la memoria.

TEORÍA DE LA COMPUTABILIDAD- CUESTIONES (II)

PRINCIPALES CUESTIONES:

¿Qué entendemos por algoritmo efectivo y por problema?

¿Es posible diseñar un algoritmo (eficiente) que resuelva el problema?

¿Qué problemas se pueden resolver con algoritmos (programas de computadora)?

LA TEORÍA DE LA COMPUTABILIDAD PUEDE CONSIDERARSE COMO UN SISTEMA DE ADVERTENCIA TEMPRANA

TEORÍA DE LA COMPUTABILIDAD- CUESTIONES (III)

TEORÍA DE LA COMPUTABILIDAD

Formalización de los conceptos:

- Problema
- Algoritmos

TEORÍA DE LA COMPLEJIDAD COMPUTACIONAL

- Eficiencia de algoritmos en tiempo y espacio
- Notaciones asintóticas
- Eficiencia en tiempo de instancias de problemas
- Cálculo de tiempo de ejecución
- Clases de complejidad (P, NP, coNP, etc)

CONCEPTO DE ALGORITMO



Resolver un problema computacional (o mejor, una clase de problemas) significa: encontrar una **máquina de Turing**, o un **algoritmo de Markov** o una **función parcial recursiva** que calcule o reconozca las soluciones.

PROCEDIMIENTOS Y ALGORITMOS

Un **algoritmo** debe satisfacer las siguientes condiciones:

1. Consiste en un **conjunto finito de instrucciones** simples y precisas, que son descritas por un conjunto finito de símbolos.
2. Siempre producirá el **resultado** (la respuesta al problema) en un número finito de pasos.
3. Hay un **agente computacional** que lleva a cabo las instrucciones (**guardar, recabar** y **realizar** los pasos de una computación). Este requerimiento es satisfecho tanto por las computadoras como por los seres humanos, puesto que ambos tienen memoria.
- 4 El agente computacional debe realizar las instrucciones por medio de **pasos discretos**.
- 5 El agente computacional debe llevar a cabo las instrucciones **determinísticamente o mecánicamente**.

CONCEPTO DE PROBLEMA – PROBLEMAS DE DECISIÓN

La Teoría de la Computabilidad y la Teoría de la Complejidad Computacional requieren una definición formal de los conceptos: **PROBLEMA Y ALGORITMO**

La mayor parte de los problemas en teoría de la complejidad tienen que ver con los **problemas de decisión**, que corresponden a poder dar una respuesta positiva o negativa a un problema dado.

Así la pregunta:

¿Qué problemas pueden ser resueltos por computadoras?

es equivalente a:

¿Qué problemas de decisión pueden resolverse?

Un **problema de decisión** es una función cuyo resultado es 0 ó 1 (**falso** o **verdadero**).
Por lo tanto, todas las funciones

$$f : \Sigma^* \rightarrow \{0,1\}$$

son **problemas de decisión**

CONCEPTO DE PROBLEMA - FORMALISMOS BÁSICOS (I)

Un **problema** es una cuádrupla $\mathbf{P} = \langle \mathbf{D}, \mathbf{R}, \mathbf{q}, \mathbf{I} \rangle$ donde:

D: dominio de Datos

R: dominio de Resultados

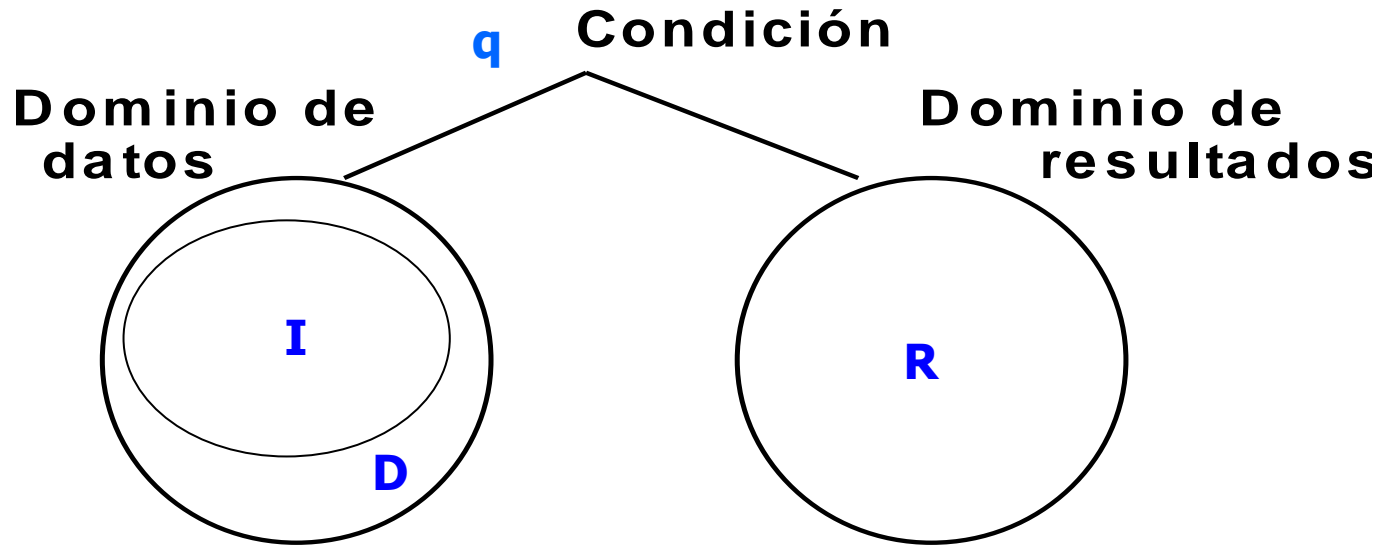
D y **R** subconjuntos de **U** (Universo del discurso)

q: relación binaria de **D** en **R**

$\mathbf{q} \subseteq \mathbf{D} \times \mathbf{R}$ es la especificación o condición del problema.

I: conjunto de instancias de interés. $\mathbf{I} \subseteq \mathbf{D}$

CONCEPTO DE PROBLEMA - FORMALISMOS BÁSICOS (II)



Un elemento $d \in D$ y un elemento $r \in R$ están en relación q si y sólo si r es un **resultado esperado** o **aceptable** para ese problema. Es decir, un elemento $r \in R$ es un **resultado admisible** para un dato $d \in D$ exactamente en el caso en que $(d,r) \in q$.

EJEMPLO

Un **número pandigital** es aquel que tiene los nueve dígitos del 1 al 9 y ninguno se repite.

Por ejemplo: 435271698

Determinar si un número es pandigital.

$$\mathbf{D} = \mathbf{N}$$

$$\mathbf{I} = \{ x / x \in \mathbf{N} \wedge x = x_1 x_2 \dots x_n \wedge n = 9 \}$$

$$\mathbf{R} = \{ y / y = \text{"sí"} \vee y = \text{"no"} \}$$

$$\mathbf{q} = \{ (x, y) \in \mathbf{I} \times \mathbf{R} / y = \text{"sí"} \Leftrightarrow \forall x : x_i \neq x_j \ i=1, \dots, n-1 \ y \ j=i+1, \dots, n \vee y = \text{"no"} \Leftrightarrow \exists x : x_i = x_j \ i=1, \dots, n-1 \ y \ j=i+1, \dots, n \}$$

Además tiene la propiedad curiosa que al dividirlo por 2 da otro número pandigital.

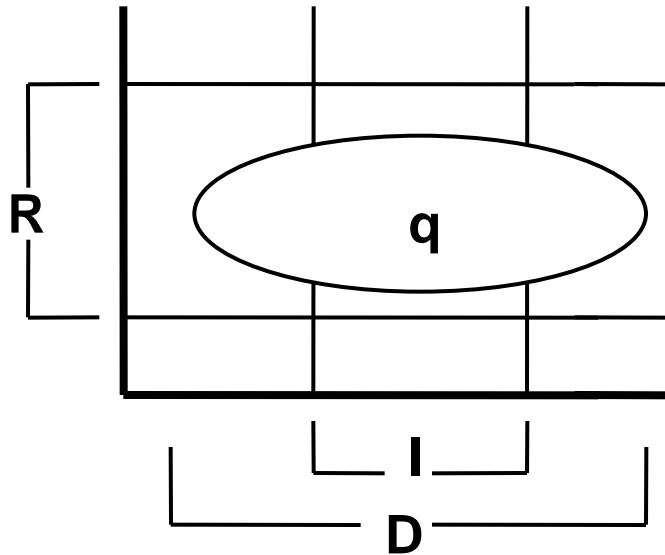
Por ejemplo: 435271698...217635849

REPRESENTACIÓN DE PROBLEMAS

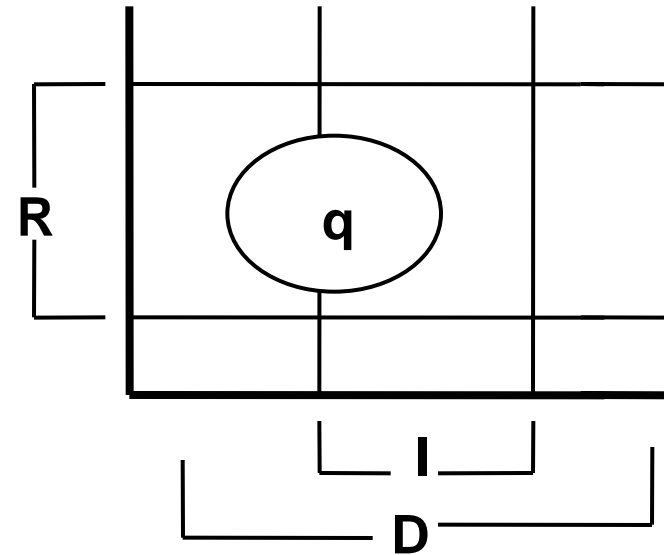
Un **problema es viable**, si y sólo si, para cada dato $d \in \mathbf{I}$ existe por lo menos un elemento r perteneciente a \mathbf{R} tal que el par (d,r) pertenezca a la condición q . Es decir, q debe estar definida para todo el conjunto de instancias de interés.

La **condición de viabilidad** de la forma:

$$(\forall d): (d \in \mathbf{I} \rightarrow (\exists r) (r \in \mathbf{R} \wedge q(d,r)))$$



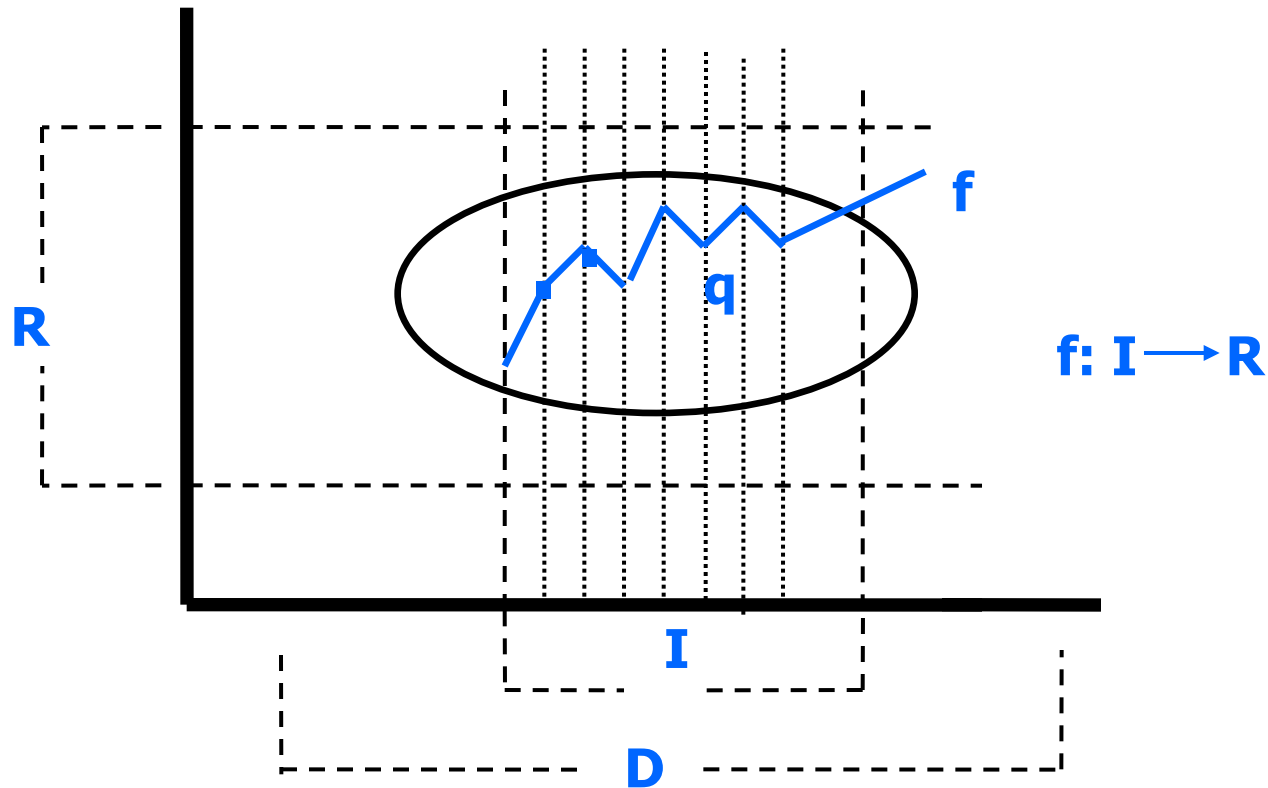
PROBLEMA VIABLE



PROBLEMA NO VIABLE

En esta representación q será una región cuya proyección sobre las abscisas deberá contener a \mathbf{I} , si el problema es viable.

SOLUCIÓN DE PROBLEMAS



Una **solución** es una función f de I en R que satisface la condición q .

Es decir: $\forall d: [d \in I \Rightarrow (d, f(d)) \in q]$

CONCEPTOS FUNDAMENTALES DE LA TEORÍA

➤ **Computabilidad:** una función es **computable**, si y sólo si, existe un algoritmo que para cualquier d que pertenezca al conjunto I calcula el valor de $f(d)$. Es decir, una función es computable, si y sólo si, existe un algoritmo que proporciona para cualquier entrada el valor de la función.

➤ **Enumerabilidad:** Un conjunto D de elementos con una propiedad dada es **enumerable**, si y sólo si, existe un algoritmo (función computable) que determine si el conjunto está vacío o bien enumere todos los miembros del conjunto.

➤ **Decidibilidad:** Un conjunto D es **decidible** (determinable), si y sólo si, existe un algoritmo que pueda determinar si un elemento dado pertenece o no al conjunto

Generabilidad: Un conjunto M de palabras sobre un alfabeto A se denomina *generable* si existe un sistema de reglas tal que una palabra W sea deducible, con la ayuda de las reglas del sistema, y si y sólo si pertenece a M .
Un conjunto M de palabras sobre un alfabeto U es generable si y sólo si M es enumerable.

GÖDELIZACIÓN

La **numeración de Gödel** consiste en una codificación que convierte a los argumentos múltiples así como a los de otro tipo a enteros no negativos.

➤ Varios enteros no negativos x_1, x_2, \dots, x_n , obtenidos mediante una primera codificación se pueden representar por un entero simple Z :

$$Z = 2^{x_1} \cdot 3^{x_2} \cdot 5^{x_3} \cdot \dots \cdot p^{x_n}$$

donde $2, 3, \dots, p$ son los n primeros **números primos**.

➤ El número original x_1, x_2, \dots, x_n se puede recuperar del entero Z ya que cada entero tiene una descomposición única en primos.

Dado un alfabeto finito U que consta de N elementos y sea W una palabra o símbolo perteneciente a U ; W puede asociarse biunívocamente con un entero no negativo $G(W)$.

Esta aplicación, G , se denomina Gödelización y $G(W)$ es el número de Gödel (con respecto a G) de la palabra W .

PROPIEDADES O REQUISITOS DE LA GODELIZACIÓN

1. Si $\mathbf{W1} \neq \mathbf{W2}$, entonces $\mathbf{G(W1)} \neq \mathbf{G(W2)}$
2. Para cualquier palabra \mathbf{W} , se puede computar en un número finito de pasos, mediante un algoritmo, el número natural correspondiente $\mathbf{G(W)}$.
3. Para cualquier número natural \mathbf{n} se puede establecer en un número finito de pasos, si \mathbf{n} es el número de Gödel de una palabra \mathbf{W} sobre \mathbf{U} .
4. Si \mathbf{n} es el número de Gödel de una palabra \mathbf{W} sobre \mathbf{U} esta palabra \mathbf{W} (unívocamente determinada según 1) debe poder ser hallada en un número finito de pasos.

EJEMPLO

$U = \{ W / W \text{ es una proposición lógica} \}$

CODIFICACIÓN 

$p : 1$
 $q : 2$
 $\wedge : 3$
 $\vee : 4$
 $\sim : 5$
 $(: 6$
 $) : 7$
 $\Rightarrow : 8$

$W = (p \wedge q) \Rightarrow \sim p \vee q$

GODELIZACIÓN 

$G(W) = \text{Número Natural } n = 2^6 \cdot 3^1 \cdot 5^3 \cdot 7^2 \cdot 11^7 \cdot 13^8 \cdot 17^5 \cdot 19^1 \cdot 23^4 \cdot 29^2$

$n = 64 \cdot 3 \cdot 125 \cdot 49 \cdot 19487171 \cdot 815730721 \cdot 1419857 \cdot 19 \cdot 279841 \cdot 841 = \dots$