Universidad Salesiana de Bolivia

Ingeniería de Sistemas

DOSSIER

SIS-426 LENGUAJE FORMALES Y COMPILADORES

8vo. Semestre

Lic. Elva Acarapi

La Paz - Bolivia

2007

DOSSIER
Lenguajes Formales y Compiladores

1. Objetivo del Dossier:

El objetivo del presente documento es desarrollar un recurso pedagógico que permita recuperar la memoria educativa durante el proceso de enseñanza aprendizaje para apoyar el logro de los resultados de aprendizaje esperados en la materia de Lenguajes Formales y Compiladores.

2. Programa Puntual

1. Introducción y Preliminares matemáticos
2. Alfabetos y Lenguajes
3. Lenguajes y Expresiones Regulares
4. Máquinas de Estados Finitos
5. Gramáticas Formales
6. Aplicación de Lenguajes Formales
7. Compiladores

3. Documentos Desarrollados

3.1 Texto de Consulta

Material seleccionado y sintetizado de las unidades didácticas definidas en el programa, cuyo objetivo es reforzar el contenido teórico de la materia.(Anexo 1)

3.2 Prácticas

Enunciados de ejercicios y prácticas de temas a desarrollar durante el curso (Anexo 2).

3.2 Presentaciones

Diapositivas elaboradas para la presentación de unidades seleccionadas (Anexo 3).

4. Propuestas

A continuación se detalla 2 propuestas de estrategias de aprendizaje que serán implementadas en el Curso para la asimilación e integración de los nuevos conocimientos a las estructuras cognitivas existentes de los alumnos relacionados al contenido temático de la materia. Estas estrategias consideran los lineamientos del Aprendizaje Cooperativo a través de la formación de GACs, los fundamentos del estilo salesiano de educar y los recursos de las nuevas tecnologías educacionales.

4.1. 1ra. ESTRATEGIA:

PRESENTACION DEL CONTENIDO TEORICO DEL TEMA

JUSTIFICACION

Esta estrategia será utilizada para la presentación de tópicos con caracterización teórica. En el contexto de la asignatura “Lenguajes Formales y Compiladores”. El contenido teórico se refiere a aquellos tópicos que pueden ser desarrollados en el aula, es decir no se requiere que el alumno implemente programas en el Laboratorio de Computación.

1ra. Etapa: PRESENTACION

Actividades:

- **De Entrada,** Saludar e incluir en el saludo un galanteo al curso para crear un ambiente agradable y establecer la comunicación con los estudiantes.
- **De motivación,** Mostrar y describir algunas aplicaciones del tema en el mundo real para despertar el interés de los estudiantes con relación al tema. Es posible utilizar gráficos y esquemas que describan visualmente dichas aplicaciones.
- **De diagnóstico de saberes previos,** Realizar un sondeo aleatorio entre los estudiantes para determinar sus conocimientos previos y su nivel de asimilación para detectar sus necesidades y dificultades de aprendizaje.

2da. Etapa: DESARROLLO

Actividades:

- **De Participación,** exposición del nuevo tópico por parte del docente con participación del alumno para sentar los fundamentos conceptuales. En esta actividad se sugiere hacer uso de presentaciones en diapositivas en computadora con apoyo del Datadisplay o en transparencias con retroproyector.
- **De ejercitación,** desarrollar ejemplos utilizando la información recientemente expuesta para operacionalizar la teoría.
- **De Práctica,** plantear un problema de aplicación a los estudiantes para concretar los objetivos de la clase, y evaluar los resultados de aprendizaje. En esta actividad se aplica el trabajo por parejas.
- **De andamiaje,** facilitar información al estudiante durante la resolución del problema y apoyar su aprendizaje en su Zona de desarrollo próximo.
- **De Retroalimentación,** evaluar el proceso de resolución y la solución encontrada para verificar si se pudo asimilar la información y convertirla en conocimientos integrados a sus estructuras cognitivas.

3ra. Etapa: INTEGRACION

Actividades:

- **De evaluación,** evaluar la solución del problema encontrada por cada pareja y elaborar un informe escrito del mismo.
- **De Socialización de conclusiones,** compartir entre todo el curso distintas soluciones para establecer las mejores opciones.

HERRAMIENTAS Y RECURSOS

- Pizarra y Tizas
- Material impreso: Resumen, Esquemas, gráficos
- Diapositivas
- Datadisplay

4.2 2da. ESTRATEGIA:

UTILIZACION DE LOS RECURSOS TELEMATICOS PARA APOYAR A LAS CLASES PRESENCIALES

JUSTIFICACION

Esta estrategia será utilizada como herramienta de apoyo a las clases presenciales del Curso, es decir que permite complementar las clases en la USB por medio de herramientas de comunicación basadas en redes telemáticas.

1ra. Etapa: PRESENTACION

Actividades:

- Realizar una presentación y entrenamiento de las herramientas que se utilizarán como complemento a las clases presenciales del curso, como la creación y manipulación de correo electrónico, comunicación vía Chat, Página Web de la Materia.
- Establecer los objetivos y normas de uso de cada una de las herramientas que se utilizarán, como la creación de cada alumno de su cuenta de e-mail, horario y frecuencia de uso del Chat y la información difundida en la Página de la Materia.

2da. Etapa: DESARROLLO

Actividades:

- Cada alumno deberá crear una cuenta de e-mail a la cual el docente podrá enviar mensajes e información relacionada al curso tales como prácticas, cronograma de actividades, requisitos para las clases presenciales, calificaciones y documentos adicionales.
- El docente debe poseer una cuenta de e-mail a la cual los alumnos se dirigirán para realizar consultas complementarias a las clases presenciales, para hacer llegar ciertos tipos de prácticas, informes, reportes y programas.
- Se deben formar grupos de trabajo de 5 alumnos para la realización de prácticas y proyectos definidos para el Tema de estudio. Estos grupos trabajarán bajo la metodología de los GAC y se mantendrán durante todo el semestre rotando los roles para cada nuevo trabajo que el grupo deberá encarar.

- Se realizarán Chat cada 2 semanas de acuerdo a un cronograma establecido previamente con cada grupo, estos chat permitirán que los alumnos puedan realizar sus consultas en línea e interactivamente,ya que las clases presenciales solo se realizan 2 veces a la semana y la comunicación vía e-mail no es directa ni muy interactiva.
- Cada grupo puede establecer su cronograma para sus respectivos Chat, que permiten salvar dificultades de reuniones presenciales.
- Cada semana la página de la Materia publicará el contenido temático planificado para esa semana, incluyendo ejemplos que los alumnos pueden bajar antes de ingresar a las clases presenciales, y prácticas que deben ser resueltas para las siguientes clases.

3ra. Etapa: INTEGRACION

Actividades:

- Al finalizar cada Tema cada grupo deberá elaborar un informe de los productos obtenidos con el trabajo en GACs. Estos informes se deben intercambiar entre todos los grupos y elaborar un reporte general del Tema estudiado.
- Los mejores trabajos deberán ser publicados en la página Web de la Materia, con sus respectivos links para descargar los programas y proyectos.

HERRAMIENTAS Y RECURSOS

- Computadoras con acceso a Internet
- Una cuenta de e-mail
- Gerenciadores de Chat

4.4. JUSTIFICACION

Esta propuesta permitirá adquirir nuevas competencias en los alumnos a nivel cognitivo desde un punto de vista técnico, considerando el enfoque de la materia; además de desarrollar competencias afectivas y morales contribuyendo a la formación de profesionales íntegros.

5. Alcances de la propuesta? Límites

Esta propuesta será implementada siempre y cuando se tenga a disposición:

- Equipamiento necesario (Computadoras, Data-Display)
- Acceso a Tecnología Internet
- Tiempo disponible en cada clase y durante el semestre

A través de la implementación de esta propuesta se pretende aplicar el uso de Grupos de Aprendizaje Cooperativos (GAC), el uso de nuevas tecnologías y el estilo salesiano para la formación de profesionales íntegros.

6. Procesos Didácticos Metodologías didácticas

Considerando el carácter eminentemente práctico y técnico de la asignatura se utilizaran estrategias y actividades fundamentadas en el Aprendizaje basado en Problemas o educación problematizadora, el aprendizaje participativo y el

Aprendizaje cooperativo, todo enmarcado en los lineamientos del Sistema Preventivo de Don Bosco al estilo salesiano.

El aprendizaje basado en problemas se fundamenta en los siguientes lineamientos:

P1: Una persona sólo conoce algo cuando lo transforma y ella misma se transforma durante la cognición.

P2: Implica la participación activa y un aprendizaje interactivo entre alumnos y profesores en la solución de problemas.

P3: El aprendizaje está encaminado a la asimilación de conocimientos y modos de actividad mediante la percepción de las explicaciones del docente en las condiciones de una situación problémica, el análisis independiente de esta situación, la formulación de problemas y su solución mediante el planteamiento de suposiciones e hipótesis, su fundamentación y demostración, así como la verificación del grado de correlación de las soluciones.

Las materias relacionadas con el uso de la computadora, son adecuadas para utilizar este enfoque de situaciones problémicas. Esto conduce a plantear constantemente problemas que les permita comprender mejor a través del descubrimiento de nuevos conocimientos para hallar la solución.

Aprendizaje Interactivo – participativo

La comunicación entre el docente y alumno es determinante para la evaluación del proceso formativo y para la retroalimentación que permite al docente establecer las dificultades del aprendizaje y definir logros de aprendizaje.

Se organizan actividades que faciliten la participación activa de los alumnos, para este fin es posible utilizar técnicas de trabajo en grupo y de aprendizaje cooperativo:

- Por parejas
- Debate
- Laboratorio
- Grupos de Aprendizaje Cooperativo
- Debates y análisis vía Foros de discusión
- Comunicación vía chat y vía e-mail

7. Cronograma semestral

Cronograma de Ejecución	UNIDADES Y CONTENDIDO ANALÍTICO	Porcentaje Avanzado	MEDIOS Y TÉCNICAS UTILIZADOS
6 – FEB - 07	Presentación e introducción a la Asignatura - Explicar sistema de evaluación y desarrollo de actividades		Pizarra – Diálogo Participativo
7 – FEB - 07	UNIDAD I : INTRODUCCION Y PRELIMINARES MATEMATICOS Introducción- Lógica proposicional	3%	Pizarra – Internet (E-mail , Chat) Resolución de Problemas – Aprendizaje Participativo
13 – FEB - 07	UNIDAD I Continuación Teoría de Conjuntos	5%	Pizarra - Material Impreso Aprendizaje Cooperativo – Aprendizaje Participativo e Interactivo
14 – FEB - 07	UNIDAD I Continuación Funciones UNIDAD II: LENGUAJES Y ALBETOS Alfabeto Cadena Lenguaje	8%	Pizarra - Material Impreso Aprendizaje Cooperativo – Aprendizaje Participativo e Interactivo
21 – FEB - 07	UNIDAD II Continuación Manejo de caracteres y cadenas	11%	Pizarra - Página Web de la Materia Aprendizaje Cooperativo – Aprendizaje Participativo e Interactivo
27 – FEB - 07	UNIDAD I Continuación Operaciones con cadenas – Operaciones con lenguajes	14%	Pizarra - Página Web de la Materia Aprendizaje Cooperativo – Aprendizaje Participativo e Interactivo
28 – FEB - 07	UNIDAD II: LENGUAJES Y EXPRESIONES REGULARES Lenguajes Regulares – Lenguajes sobre alfabetos	19%	Pizarra Aprendizaje Cooperativo – Aprendizaje Participativo e Interactivo
6 – MAR - 07	UNIDAD II Continuación Lenguajes regulares -Propiedades	22%	Pizarra Aprendizaje Cooperativo – Aprendizaje Participativo e Interactivo
7 – MAR – 07	**PRIMERA EVALUACIÓN PARCIAL**		

Cronograma de Ejecución	UNIDADES Y CONTENDIDO ANALÍTICO	Porcentaje Avanzado	MEDIOS Y TÉCNICAS UTILIZADAS
13– MAR - 07	UNIDAD III Continuación Expresiones Regulares	25%	Pizarra - Material Impreso Aprendizaje Cooperativo – Aprendizaje Participativo e Interactivos
14- MAR - 07	UNIDAD III Continuación Propiedades de Expresiones Regulares	28%	Pizarra - Material Impreso Aprendizaje Cooperativo – Aprendizaje Participativo e Interactivo- Resolución de Problemas
20-MAR - 07	UNIDAD IV MAQUINAS DE ESTADOS FINITOS Introducción a los autómatas finitos	31%	Pizarra - Material Impreso Aprendizaje Cooperativo – Aprendizaje Participativo e Interactivo
21– MAR - 07	UNIDAD IV Continuación Autómatas finitos deterministas	34%	Pizarra - Material Impreso Aprendizaje Cooperativo – Aprendizaje Participativo e Interactivo
27– MAR - 07	UNIDAD IV Continuación Ejemplos de ADF	37%	Pizarra - Página Web de la Materia Aprendizaje Cooperativo – Aprendizaje Participativo e Interactivo
28– MAR - 07	UNIDAD IV Continuación Autómatas Finitos no deterministas.	40%	Pizarra - Página Web de la Materia Aprendizaje Cooperativo – Aprendizaje Participativo e Interactivo
3– ABR - 07	UNIDAD IV Continuación Ejemplos de AFND	43%	Pizarra - Página Web de la Materia Aprendizaje Cooperativo – Aprendizaje Participativo e Interactivo – Aprendizaje Basado en Problemas
4– ABR - 07	UNIDAD III Continuación Equivalencia entre AFD y AFND	46%	Pizarra - Página Web de la Materia Aprendizaje Cooperativo – Aprendizaje Participativo e Interactivo– Aprendizaje Basado en Problemas
10– ABR - 07	UNIDAD IV Continuación Ejercicios de aplicación	49%	Power Point – Computadora – Pizarra Aprendizaje Cooperativo – Aprendizaje Interactivo
11 – ABR - 07	**SEGUNDA EVALUACIÓN PARCIAL**		

Cronograma de Ejecución	UNIDADES Y CONTENDIDO ANALÍTICO	Porcentaje Avanzado	MEDIOS Y TÉCNICAS UTILIZADAS
17– ABR - 07	UNIDAD V GRAMATICAS FORMALES Gramáticas Regulares y Lenguajes regulares	51%	Power Point – Computadora -Página Web de la Materia Grupos de Aprendizaje Cooperativo – Aprendizaje Basado en Problemas
18– ABR - 07	UNIDAD V Continuación GR y Autómatas- Jerarquía de Gramáticas	54%	Power Point – Computadora -Página Web de la Materia Grupos de Aprendizaje Cooperativo – Aprendizaje Basado en Problemas
24– ABR - 07	UNIDAD V Continuación Ejercicios GR	57%	Power Point – Computadora -Página Web de la Materia Grupos de Aprendizaje Cooperativo – Aprendizaje Basado en Problemas
25– ABR - 07	UNIDAD V Continuación Gramáticas Independientes de contexto	60%	Power Point – Computadora -Página Web de la Materia - Internet (foro) Grupos de Aprendizaje Cooperativo – Aprendizaje Basado en Problemas
2– MAY - 07	UNIDAD V Continuación GIC y Autómatas de pila	63%	Power Point – Computadora -Página Web de la Materia Grupos de Aprendizaje Cooperativo – Aprendizaje Basado en Problemas
8– MAY - 07	UNIDAD V Continuación Ejemplos de autómatas de pila	66%	Power Point – Computadora -Página Web de la Materia Grupos de Aprendizaje Cooperativo – Aprendizaje Basado en Problemas
9– MAY - 07	UNIDAD VI APLICACIÓN DE LENGUAJES FORMALES Lenguajes Formales y Lenguajes de Programación	69%	Power Point – Computadora -Página Web de la Materia Grupos de Aprendizaje Cooperativo – Aprendizaje Basado en Problemas
15– MAY - 07	UNIDAD VI Continuación Características de los Lenguajes de Programación	72%	Power Point – Computadora -Página Web de la Materia Grupos de Aprendizaje Cooperativo – Aprendizaje Basado en Problemas
16-MAY 07	**EVALUACION FINAL**		

Cronograma de Ejecución	UNIDADES Y CONTENDIDO ANALÍTICO	Porcentaje Avanzado	MEDIOS Y TÉCNICAS UTILIZADAS
22– MAY – 07	UNIDAD VI Continuación Ejemplos de definiciones Sintácticas. Diagramas Sintácticos	75%	Power Point – Computadora -Página Web de la Materia Grupos de Aprendizaje Cooperativo – Aprendizaje Basado en Problemas
23– MAY – 07	UNIDAD VI Continuación Definiciones Semánticas	78%	Power Point – Computadora -Página Web de la Materia Grupos de Aprendizaje Cooperativo – Aprendizaje Basado en Problemas
29– MAY - 07	UNIDAD VII COMPILADORES Compiladores e Intérpretes	81%	Power Point – Computadora –Internet (foro) Grupos de Aprendizaje Cooperativo – Aprendizaje Basado en Problemas
30– MAY - 07	UNIDAD VII Continuación Fases de Compilación	85%	Power Point – Computadora –Internet (foro) Grupos de Aprendizaje Cooperativo – Aprendizaje Basado en Problemas
5– JUN – 07	UNIDAD VII Continuación Analizador Léxico	89%	Power Point – Computadora -Página Web de la Materia Grupos de Aprendizaje Cooperativo – Aprendizaje Basado en Problemas
6– JUN– 07	UNIDAD VII Continuación Analizador Sintáctico	94%	Power Point – Computadora -Página Web de la Materia Grupos de Aprendizaje Cooperativo – Aprendizaje Basado en Problemas
12– JUN - 07	UNIDAD VII Continuación Optimización y Generación de código	96%	Power Point – Computadora –Internet (foro) Grupos de Aprendizaje Cooperativo – Aprendizaje Basado en Problemas
13– JUN - 07	UNIDAD VII Continuación Aplicaciones	100%	Power Point – Computadora –Internet (foro) Grupos de Aprendizaje Cooperativo – Aprendizaje Basado en Problemas
20-27-MAY- 07	**EVALUACION FINAL**		

8. Evaluación

El seguimiento y evaluación de los alumnos comprende dos fases:

- Cualitativa
- Cuantitativa, y

tres modalidades:

- Diagnóstica
- Formativa (competencias sociales y afectivas)
- Sumativa

Las actividades en aula se desarrollarán en forma: **individual y grupal** .

CRITERIOS	**PROCESOS**	**Puntaje**	**CRONOGRAMA**
Cantidad y Calidad de aprendizaje	1er. Evaluación 1er. Parcial 15% Otros * 10%	25 puntos	5-12 de Marzo
Cantidad y Calidad de aprendizaje	2do. Evaluación 2do. Parcial 15% Otros * 10%	25 puntos	9-16 de Abril
Cantidad y Calidad de aprendizaje	3ra. Evaluación 3er. Parcial 15% Otros * 10%	25 puntos	14-21 de Mayo
Control y Evaluación Sumativa	Examen Final	25 puntos	20-27 de Junio

***** Control y evaluación formativa; Precisión, orden y calidad en la elaboración de trabajos, prácticas y participación en clases; Calidad y compromiso en los Grupos de Aprendizaje Cooperativo; Desarrollo de un Trabajo Final de Curso: Proyecto de Software

9. Bibliografía

- Kelley, Dean - "Teoría de autómatas y Lenguajes Formales" – España - Prentice Hall – 1995
- Aho, Sethi & Ullman – "Compiladores: Principios, técnicas y Herramientas" - Estados Unidos - Addison Wesley Iberoamericana, 1990.
- Facultad de Informática Madrid – "Informática Teórica 2da. Parte"
- Pyster – " Diseño y Construcción de Compiladores"
- Mak, Ronald – "Writing Compilers & Interpreters"

Anexo 1

Texto de Consulta

Unidad 1

Introducción y Preliminares Matemáticos

Teoría elemental de conjuntos

CONJUNTOS

Un conjunto es una lista, colección o clase de objetos bien definidos, objetos que, como se verá en los ejemplos, pueden ser cualesquiera: números, personas, letras, ríos, etc. Estos objetos se llaman elementos o miembros del conjunto.

Ejemplos particulares de conjuntos.

Los números 1, 3, 7 y 10.

Las vocales del alfabeto: a, e, i. o, u.

Las personas que habitan la Tierra.

Los estudiantes Tomas, Ricardo y Enrique.

Los países Inglaterra, Francia y Dinamarca.

Nótese que los conjuntos de los ejemplos impares vienen definidos, o sea presentados, enumerando de hecho sus elementos, y que los conjuntos de los ejemplos pares se definen enunciando propiedades, o sea reglas, que deciden si un objeto particular es, o no, elemento del conjunto.

NOTACIÓN

Es usual denotar los conjuntos por letras mayúsculas

A, B, X, Y, ...

Los elementos de los conjuntos se representan por letras minúsculas

a, b, x, y, ...

Al definir un conjunto por la efectiva enumeración de sus elementos, por ejemplo. el A, que consiste en los números 1, 3, 7 y 10, se escribe

$$A = \{1, 3, 7, 10\}$$

separando los elementos por comas, y encerrándolos entre llaves. Esta es la llamada forma tabular de un conjunto. Pero si se define un conjunto enunciando propiedades que deben tener sus elementos como, por ejemplo, el B, conjunto de todos los números país, entonces se emplea una letra, Por lo general x, para representar un elemento cualquiera y se escribe

$$B = \{x \square\ x \text{ es par}\}$$

lo que se lee «B es el conjunto de los números x tales que x es par». Se dice que ésta es la forma de definición por comprensión o constructiva de un conjunto. Téngase en cuenta que la barra vertical se lee «tales que».

Para aclarar el empleo de la anterior notación, se escriben de nuevo los conjuntos de los ejemplos anteriores:

A1 = {1, 3, 7, 10}.

A2 = {a, e, i, o, u}.

A3 = {x□ x es un apersona que habita en la Tierra}.

A4 = {Tomás Ricardo, Enrique}.

A5 = {x□ x es estudiante y x está ausente de la escuela}.

A6 = {Inglaterra, Francia, Dinamarca}.

Si un objeto x es elemento de un conjunto A, es decir, si A contiene a x como uno de sus elementos, se escribe

x □ A

que se puede leer también «x pertenece a A» o «x está en A». Es costumbre en los escritos matemáticos poner una línea vertical «□ » u oblicua «/» tachando un símbolo para indicar lo opuesto o la negación del significado del símbolo.

Ejemplo:

Si A = {a, e, i, o, u}, entonces a □ A, b /□ A, e □ A, f /□ A.

CONJUNTOS FINITO E INFINITOS

Intuitivamente, un conjunto es finito si consta de un cierto número de elementos distintos, es decir, si al contar los diferentes elementos del conjunto el proceso de contar puede acabar. Si no, el conjunto es infinito

Ejemplos:

Si M es el conjunto de los días de la semana, entonces M es finito.

Si N = {2, 4, 6, 8, ... } N es infinito.

IGUALDAD DE CONJUNTOS

El conjunto A es igual al conjunto B si ambos tienen los mismos elementos, es decir, si cada elemento que pertenece a A pertenece también a B y si cada elemento que pertenece a B pertenece también a A. Se denota la igualdad de los conjuntos A y B por

A = B

Ejemplos:

Sean A = {1, 2, 3, 4} y B = {3, 1, 4, 2} Entonces A = B, es decir, {1, 2, 3, 4} = {3,1, 4, 2},

pues cada uno de los elementos 1, 2, 3 y 4 de A pertenece a B y cada uno de los elementos 3, 1. 4 y 2 de B pertenece a A.

Sean C = {5, 6, 5, 7} y D ={7, 5, 7, 6}. Entonces C = D. No importa que los números se repitan siempre y cuando sean los mismos.

CONJUNTO VACIO

Conjunto que carece de elementos. Este conjunto se suele llamar conjunto nulo. Aquí diremos de un conjunto semejante que es vacío y se le denotará por el símbolo □ .

Ejemplos:

Si *A* es el conjunto de personas vivientes mayores de 200 años, *A* es vacío según las estadísticas conocidas.

SUBCONJUNTOS

Si todo elemento de un conjunto A es también elemento de un conjunto B, entonces se dice que A es un subconjunto de B. Más claro: A es un subconjunto de B si x □ A implica x □ B. Se denota esta relación escribiendo

A □ B

que también se puede leer «A está contenido en B».

Ejemplo:

El conjunto C = {1, 3, 5} es un subconjunto del D = {5, 4, 3, 2, 1}, ya que todo número 1, 3 y 5 de C pertenece también a D.

El conjunto E = {2, 4, 6} es un subconjunto del F = {6, 2. 4}, pues cada número 2, 4 y 6 que pertenece a E pertenece también a F. Obsérvese en particular que E = F. De la misma manera se puede mostrar que todo conjunto es subconjunto de sí mismo.

Con la anterior definición de subconjunto se puede dar de otra manera la definición de la igualdad de dos conjuntos:

Definición: Dos conjuntos A y B son iguales, A = B, si, y solo si, A □ B y B□ A. Si A es un subconjunto de B, se puede escribir también

B □ A

que se lee «B es un superconjunto de A» o «B contiene a A». Y se escribe, además,

A/□ B o B/□ A

si A no es subconjunto de *B.* Para concluir, se tiene:

El conjunto vacío □ se considera subconjunto de todo conjunto.

Si A no es subconjunto de B, es decir, si, A /□ B, entonces hay por lo menos un elemento de A. que no es elemento de B.

COMPARABILIDAD

Dos conjuntos A y B se dicen comparables si

A □ B o B □ A

Esto es, si uno de los conjuntos es subconjunto del otro. En cambio, dos conjuntos A y B se dicen no comparables si

A □ B o B □ A

Nótese que si A no es comparable con B, entonces hay en A un elemento que no está en B y hay también en B un elemento que no está en A.

Ejemplo:

Sean A = {a, b} y B = {a, b, c}. Entonces A es comparable con B, pues A es un subconjunto de B.

CONJUNTOS DE CONJUNTOS

Para evitar decir «conjuntos de conjuntos», se suele decir «familia de conjuntos o «clase de conjuntos». En tales casos y para evitar confusiones, se emplean letras inglesas

□ , □ , ...

para designar familias o conjuntos, ya que las mayúsculas denotan sus elementos.

Ejemplo:

En geometría es corriente hablar de «familias de rectas» o «familias de curvas», pues rectas y curvas ya son ellas mismas conjuntos de puntos.

En teoría es posible que un conjunto tenga entre sus elementos algunos que sean a su vez conjuntos y otros que no lo sean. Pero en las aplicaciones de la teoría de conjuntos este caso se presenta rara vez.

Ejemplo:

Sea A = {2, {1, 3}, 4, {2, 5}}. A no es pues, una familia de conjuntos; algunos elementos de A son conjuntos y otros no.

CONJUNTO UNIVERSAL

En toda aplicación de la teoría de conjuntos todos los conjuntos que se consideran serán muy probablemente subconjuntos de un mismo conjunto dado. Este conjunto se llamará conjunto universal o universo del discurso y se denotará por U.

Ejemplo:

En geometría plana el conjunto universal es el de todos los puntos del plano.

En los estudios sobre población humana el conjunto universal es el de todas las gentes del mundo.

CONJUNTO POTENCIA

La f'amilia de todos los subconjuntos de un conjunto S se llama conjunto potencia de S. Se le designa por

$$2^S$$

Ejemplo:

Si M = {a, b}, entonces

2^M = {{a, b}, {a}, {b}, □ }

Si un conjunto S es finito, digamos que S tenga n elementos, entonces el conjunto potencia de S tendrá 2^n elementos, como se puede demostrar. Esta es una razón para llamar conjunto de potencia de S la clase de los subconjuntos de S y para denotarla por 2^S.

Nociones sobre Funciones

Definición de función:

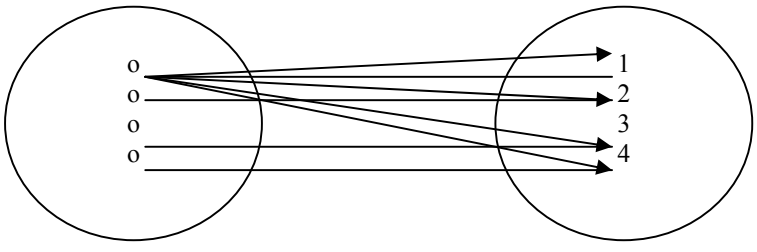
Una relación del conjunto A con conjunto B es un subconjunto de producto cartesiano AxB entonces

$R \subseteq A \times B$ y $(a,b) \in R$ se dice que está relacionado con b bajo la relación R

EJEMPLO

$$\mathbf{R \subset NxN \wedge (x,y) \in R \Leftrightarrow X \leq y}$$

R = {(1,1), (1,2),(1,3),(5,8),.....}



F es una funcion ó aplicación de A en B si solo si F es una relacion entre A y B talque todo elemento de A tiene único correspondiente en B.

F es una funcion de A en B si solo si :
$F \subset A \times B$ satisface las siguientes condiciones:

i).- $\forall\ X \in A, \exists\ y \in B/(x , y) \in F$

ii).- $(x , y) \in F \ \wedge\ (x, z) \in F \Rightarrow y = z$

Sea F : $A \rightarrow B$, donde $X \in A$
a).- Qué es F, A, B

F = es una función
A = es el dominio de f
B = es el codominio

DOMINIO.- es el conjunto de los elementos de A que admite imagen en B.
$Df = \{X \in A / (X, Y) \in F\}$
CODOMINIO.- Es el conjunto de elementos de B que admite una antecedente en A como:

$Cof = \{y \in B / (x,y) \in f\}$

b).- ¿ f = f(x)? es igual o no. Esto no es igual.

f = {(1,1,), (2,4), (3,9), (4,16),.....}

Obtener la cardinalidad de los siguientes conjuntos
La cardinalidad de un conjunto se refiere a el numero de elementos de conjunto.

a) $A = \{\} \Rightarrow |A| = 0$
b) $B = \{a, b, c, d\} \Rightarrow |B| = 4$
c) $C = \{a1, a2. a3.....a^k\} \Rightarrow |C| = k$

Cuando es cierta la siguiente igualdad?
$|A \cup B| = |A| + |B|$

Resp = cuando A y B son conjuntos disjuntos, es decir $|A \cap B| = 0$

Unidad II

Alfabetos y Lenguajes

CADENA:

Una cadena o palabra está formada por la concatenación de caracteres.
La cadena mínima o nula se llama λ.

Ejemplo:

Dado el siguiente alfabeto: Σ = {a,b}

λ, a, b, ab, aaa, bbb, abba $\in \sum$

Nota:

a) el orden de las palabras es importante de esta manera ab es diferente de ba
b) Por lo general las palabras se representan con las últimas letras del alfabeto, como t, u, v, w, x, y, z

Ejemplo:

x=abc z = dac

la concatenación será xz = abcdac

Ejemplo:

λ x = xλ = x = abc

LONGITUD DE UNA CADENA la longitud de una cadena x, se indica con |x| y es el número de caracteres que la complementan

Ejemplo:

|λ| = 0 , |a| = 1, |abcde| = 5

POTENCIA DE UNA CADENA se forma concatenando una palabra con sigo misma tantas veces como indica el exponente

LENGUAJE: Un lenguaje está formado por los siguientes elementos:

- Un diccionario: que indica los significados de la palabra del lenguaje
- Un conjunto de reglas para describir las sentencias válidas del lenguaje, este conjunto de reglas forma la gramática del lenguaje (sintaxis y semántica)

Un lenguaje es un conjunto de cadenas de caracteres, además:

- Una “cadena” del lenguaje es una secuencia ordenada de símbolos.
- Un “símbolo” es un ítem elemental del vocabulario del lenguaje que se emplea para formar las cadenas del lenguaje, que se llaman sus sentencias
- Un “alfabeto” o vocabulario, es el conjunto de todos los símbolos que forman las sentencias del lenguaje

FORMAS DE REPRESENTACION DE UN LENGUAJE

Dependiendo de la complejidad del lenguaje se utilizan dos formas:

a) “la enumeración” de todas las cadenas de los símbolos que constituyen el lenguaje
b) “Descripción algebraica” utilizando la forma abreviada de que a^n es la concatenación de n símbolos.
c) “Conjunto con una propiedad”, la propiedad servirá para decidir si una tira pertenece o no al lenguaje.

La operación básica para formar las cadenas del lenguaje en cuestión es la “concatenación”

Unidad III

Lenguajes y Expresiones Regulares

Un **lenguaje regular** es un tipo de lenguaje formal que satisface las siguientes propiedades:

Puede ser reconocido por:

- un autómata finito determinista
- un autómata finito no determinista
- un autómata finito alterno
- una máquina de Turing de solo lectura

Es generado por:

- una gramática regular
- una gramática de prefijos

Es descrito por:

- una expresión regular

Definición. Sea **V** un alfabeto cualquiera. Una *expresión regular* sobre **V** se define recursivamente, como sigue:

1. La expresión regular ε representa al lenguaje {ε}
2. La expresión regular **a**, donde a ϵ $V_{T,}$ representa al lenguaje {a}
3. Si **r** y **s** son expresiones regulares tales que:

   L(r) es el lenguaje representado por **r** y
   L(r) es el lenguaje representado por **r**

entonces:

a) (r) | (s) es una expresión regular que representa al lenguaje:

L(r) U L(s)

b) (r)(s) es una expresión regular que representa al lenguaje:

$$L(r)L(s)$$

c) $(r)^*$ es una expresión regular que representa al lenguaje:

$$(L(r))^*$$

Unidad IV

Máquinas de Estados Finitos

Definición. Un *autómata finito determinista* (AFD) es un sistema de la forma:

$$(E, V_T, f, e_0, F)$$

donde:

E es un conjunto finito no-vacío de elementos llamados *estados*

V_T es un alfabeto llamado *alfabeto de entrada*

f es una función de un subconjunto de E x V_T en E llamada *función de transición*

e_0 es un estado especial llamado *estado inicial*

F es un subconjunto de E cuyos elementos se nombran *estados finales*

Definición. Dado un AFD (E, V_T, f, e_0, F) llamaremos *configuración* a cualquier par ordenado de la forma:

$$(e , x)$$

tal que e ϵ E y x ϵ V^*_T

Si e = e_0 diremos que la configuración es *inicial*

Si e ϵ F y x = ε diremos que la configuración es *final*

Definición. Dado un AFD (E, V_T, f, e_0, F) y dos configuraciones de la forma:

$$(e , x) \quad y \quad (e' , y)$$

diremos que existe un *movimiento* de (e , x) a (e' , y) y lo denotaremos por:

$$(e , x) \text{ |--- } (e' , y)$$

si x = ay, a ϵ V_T y f(e,a) = e'

Notaciones. Dadas dos configuraciones c y c' denotaremos por:

a) c |---^+ c' si c = c_1 |--- c_2 |--- ..... |--- c_n = c' con n ≥ 1

b) c |---^* c' si c |---^+ c' ó c = c'

Definición. Dado un AFD A = (E, V_T, f, e_0, F) definimos como el *lenguaje aceptado* por **A** al conjunto siguiente:

$$L(A) = \{x \mid x \epsilon V^*_T \text{ y } (e_0 , x) \text{ |---}^* (e , \varepsilon) \text{ para algún e } \epsilon F \}$$

Definición. Un *autómata finito no-determinista* (AFN) es un sistema de la forma:

$$(E, V_T, f, e_0, F)$$

donde:

E es un conjunto finito no-vacío de elementos llamados *estados*

V_T es un alfabeto llamado *alfabeto de entrada*

f es una función de E x (V_T U {ε}) en P(E) llamada *función de transición*

e_0 es un estado especial llamado *estado inicial*

F es un subconjunto de E cuyos elementos se nombran *estados finales*

Definición. Dado un AFN (E, V_T, f, e_0, F) y dos configuraciones de la forma:

(e , x) y (e' , y)

diremos que existe un *movimiento* de (e , x) a (e' , y) y lo denotaremos por:

(e , x) |--- (e' , y)

si x = ay, a ϵ (V_T U {ε}) y e'ϵ f(e,a)

Autómatas de Pila

Definición. Un *autómata de pila* (AP) es un sistema de la forma:

(E, V_T, Γ, f, e_0, Z_0, F)

donde:

E es un conjunto finito no-vacío de elementos llamados *estados*

V_T es un alfabeto llamado *alfabeto de entrada*

Γ es un alfabeto llamado *alfabeto de pila*

f es una función de E x (V_T U {ε}) x Γ en los subconjuntos finitos de E x Γ^*

e_0 es un estado especial llamado *estado inicial*

Z_0 es un símbolo especial de Γ llamado *símbolo inicial de pila*

F es un subconjunto de E llamado *conjunto de estados finales*

Definición. Dado un AP (E, V_T, Γ, f, e_0, Z_0, F) llamaremos *configuración* a cualquier triplo ordenado de la forma:

(e , x , α)

tal que e ϵ E, x ϵ V^*_T y α ϵ Γ^*

Si e = e_0 y α = Z_0 diremos que la configuración es *inicial*

Si e ϵ F y x = ε diremos que la configuración es *final*

Definición. Dado un AP (E, V_T, Γ, f, e_0, Z_0, F) y dos configuraciones de la forma:

(e , x , α) y (e' , y , α')

diremos que existe un *movimiento* de (e , x , α) a (e' , y , β) y lo denotaremos por:

(e , x , α) |--- (e' , y , α')

si x = ay, a ϵ (V_T U {ε}), α = Zδ, Z ϵ Γ, α' = βδ, δ, β ϵ Γ^* y f(e,a,Z) contiene a (e', β)

Notaciones. Dadas dos configuraciones c y c' denotaremos por:

a) $c \vdash^{+} c'$ si $c = c_1 \vdash c_2 \vdash \ldots\ldots \vdash c_n = c'$ con $n \geq 1$

b) $c \vdash^{*} c'$ si $c \vdash^{+} c'$ ó $c = c'$

Definición. Dado un AP $A = (E, V_T, \Gamma, f, e_0, Z_0, F)$ definimos como el *lenguaje aceptado* por A al conjunto siguiente:

$$L(A) = \{x \mid x \in V^*_T \text{ y } (e_0, x, Z_0) \vdash^{*} (e, \varepsilon, \alpha) \text{ para algún } e \in F \text{ y } \alpha \in \Gamma^*\}$$

Definición. Un AP $P = (E, V_T, \Gamma, f, e_0, Z_0, F)$ se denomina *determinista* (APD) si para todo e ϵ E y todo Z ϵ Γ se cumple al menos alguna de las dos condiciones siguientes:

a) f(e,a,Z) contiene a lo sumo un elemento para todo a ϵ V_T y $f(e,\varepsilon,Z) = \emptyset$ ó

b) $f(e,a,Z) = \emptyset$ para todo a ϵ V_T y f(e,ε,Z) contiene a lo sumo un elemento

En caso contrario, P se denomina *no-determinista*

Notación. Puede observarse que en un APD, cuando $f(e,a,Z) \neq \emptyset$ para cualquier a ϵ $(V_T \cup \{\varepsilon\})$, entonces f(e,a,Z) contiene solamente un elemento; luego, en este caso, denotaremos el valor de la función de transición del autómata por f(e,a,Z) = (e',α) en lugar de f(e,a,Z) = {(e',α)}

TEOREMA DE TRANSFORMACIÓN AFN A AFD

Para todo AFN N existe algún AFD tal que L(N)=L(D)

Un AFN con transiciones ε puede ser convertido en un AFN sin transiciones ε, eliminando las transiciones vacías, sin alterar el comportamiento del autómata. Para hacer esto, es necesario comprender que las deltas manejadas tienen una diferencia cuando se trata de la cerradura al vacío, ya que en el AFN sin ε la cerradura al vacío de un estado es solamente el mismo estado.

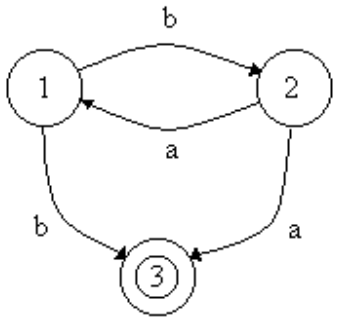
LEMA:

Para cada $x,y \in \Sigma$ y $A \subseteq K$, $\Delta(A,xy) = \Delta(\Delta(A,x),y)$

El lema anterior nos dice que es posible separar las cadenas en una operación de transición de un Autómata Finito. Esta separación nos ayudará a simplificar el rastreo de la cadena general.

CONVERTIR UN DIAGRAMA NO DETERMINISTA EN UNO DETERMINISTA

El diagrama del siguiente autómata para el alfabeto S ={a, b}. Como podemos ver, no es determinista pues desde el estado 1 salen dos arcos rotulados con b y del estado 2 salen dos arcos etiquetados con a.



<>

Para convertir el diagrama no determinista en uno que lo sea vamos ha realizar los siguientes pasos:

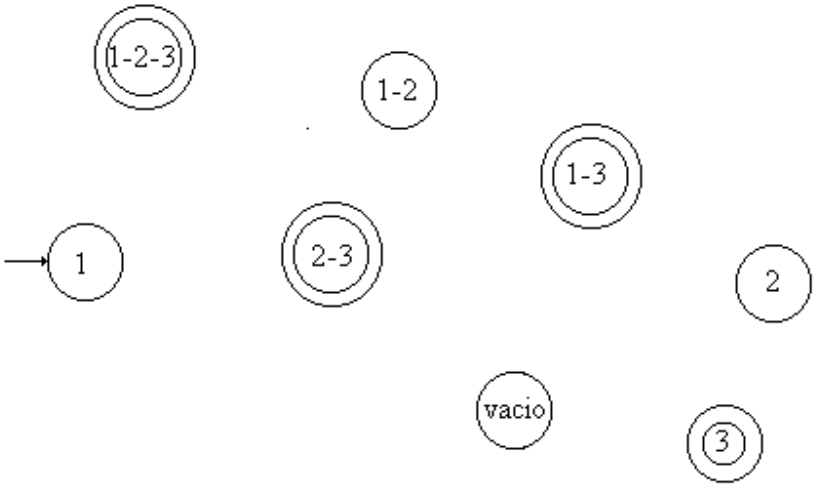
- S'=P(S) Conjunto de todos los subconjuntos de S (recordar que el conjunto potencia se encuentra incluido el conjunto vacío, que será el estado de captación global) Como tenemos tres estados, el conjunto potencia P(S) = { l, 1, 2, 3, 1-2, 1-3, 2-3, 1-2-3 }

- i'= {i} (mismo estado inicial)

- En nuestro caso seguirá siendo el estado 1.

- F' es la colección de subconjuntos de S (estados de S') que contienen, por lo menos, un estado de F (cada uno de los estados de S' dentro de los cuales hay al menos un estado de aceptación de M).

- En nuestro caso serán todos los subconjuntos que tengan el estado 3, ya que este es el único estado de aceptación del diagrama original; luego F'= { 3, 1-3, 2-3, 1-2-3 }

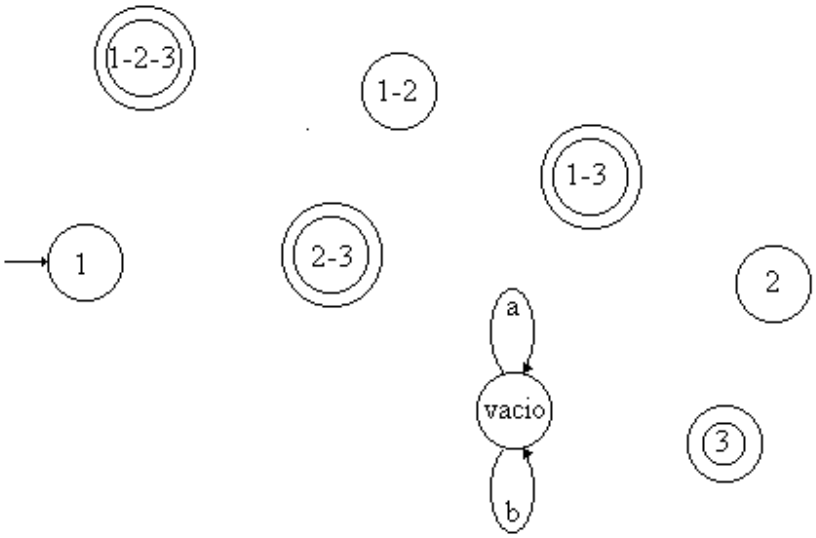


- d es la función de S' x S a S'; Para cada símbolo del alfabeto y estado s' de S', d (s',x) es el estado de S' compuesto por los estados de S a los que es posible llegar desde todos los estados s de s' siguiendo un arco con etiqueta x. Como d es una función, M' es finito determinista.

  En nuestro caso, En cada estado del conjunto potencia solo va a salir un arco por cada símbolo, siendo el destino, el estado de S' que tenga todos los estados a los que fuera en el diagrama inicial: para ello:

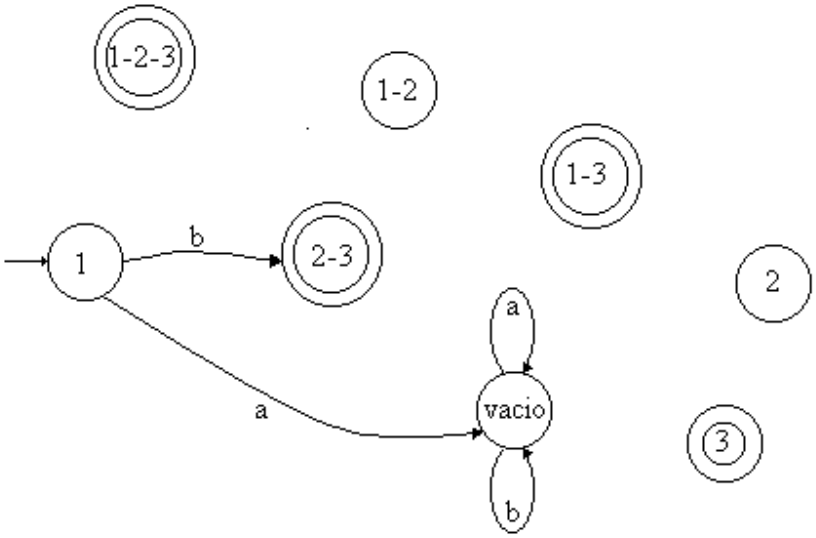
- VACÍO.-

Es el estado de captación global, por lo tanto se le dibujan tantos arcos que salen e inciden en el estado, como símbolos del alfabeto haya, con los cuales se rotulan. Además, en este estado, van a incidir todas aquellas transiciones que no existían para algún símbolo en algún estado original.



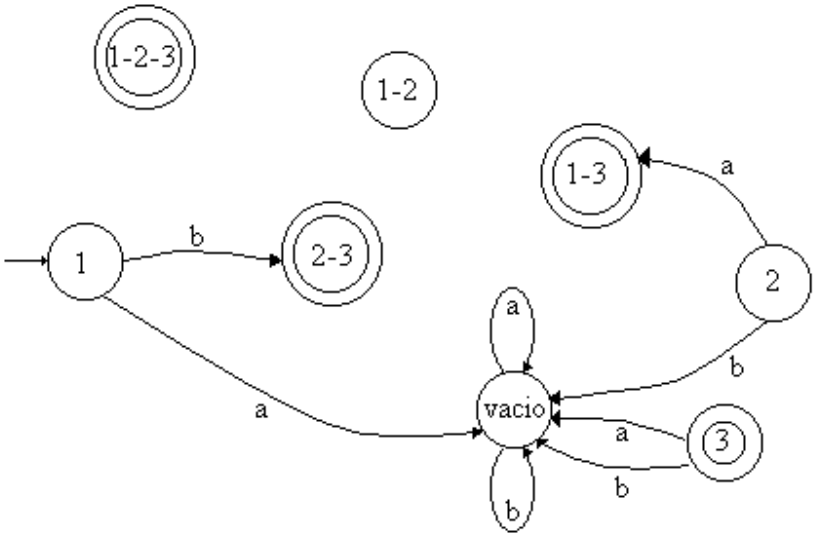
- ESTADO 1.-

Con la etiqueta a no hay transición en el original, por lo tanto el arco se dibuja hacia el estado vacío con la etiqueta b salen dos arcos, uno hacia el estado 2 y otro al estado 3, por lo tanto el arco se dibuja al estado 2-3



- ESTADO 2.- Con la etiqueta b no hay transición en el original, por lo tanto el arco se dibuja hacia el estado vacío; con la etiqueta a salen dos arcos, uno hacia el estado 1 y otro al estado 3, por lo tanto el arco se dibuja al estado 1-3.

  \+ Estado 3.- Con ninguna de las dos etiquetas hay transición en el original, por lo tanto se dibujan sendos arcos hacia el estado vacío.

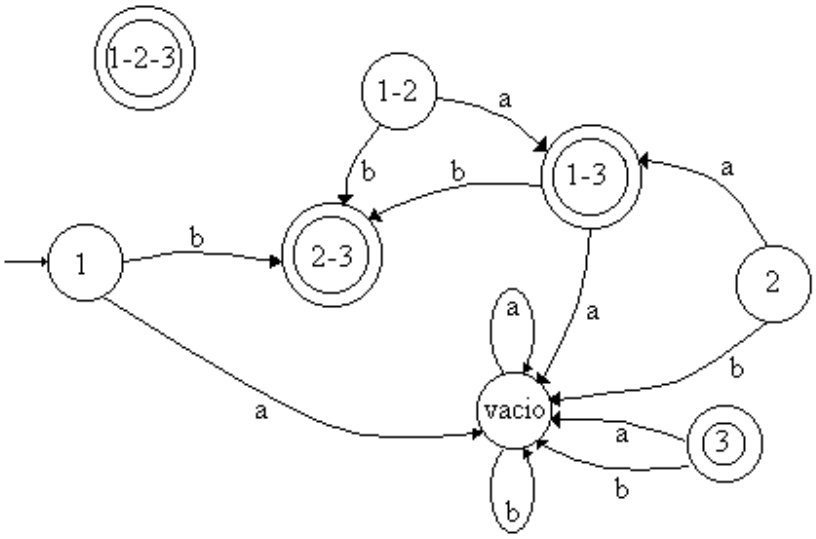


- ESTADO 1-2.-

Con la etiqueta a hay transición desde el estado 2 original al 1 y 3 original, por lo tanto el arco se dibuja hacia el estado 1-3; con la etiqueta b salen dos arcos desde el estado 1 original, uno hacia el estado 2 y otro al estado 3, por lo tanto el arco se dibuja al estado 2-3.

- ESTADO 1-3.-

Con la etiqueta a no hay transición desde ninguno de los dos estados originales, por lo tanto el arco se dibuja hacia el estado vacío; con la etiqueta b salen dos arcos desde el estado 1 original, uno hacia el estado 2 y otro al estado 3, por lo tanto el arco se dibuja al estado 2-3.

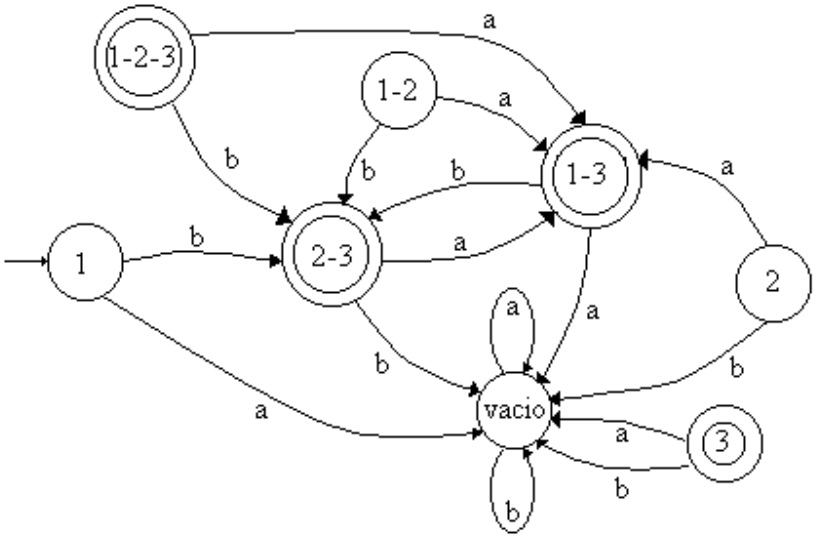


- ESTADO 2-3.-

Con la etiqueta a hay transición desde el estado 2 original al 1 y 3 original, por lo tanto el arco se dibuja hacia el estado 1-3; con la etiqueta b no sale ningún arco en ninguno de los dos estados originales, por lo tanto el arco se dibuja al estado vacío.

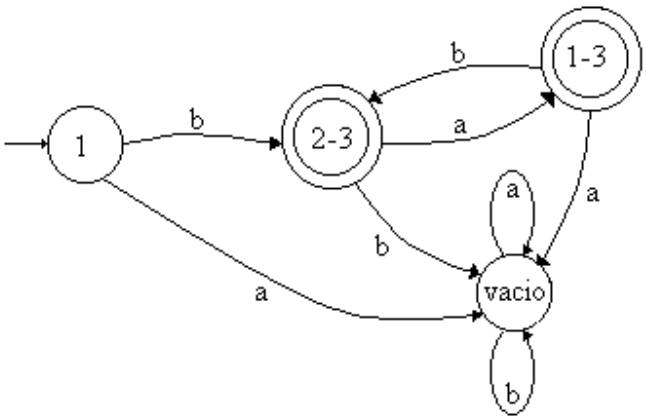
- Estado 1-2-3.-

Con la etiqueta a hay transición desde el estado 2 original al 1 y 3 original, por lo tanto el arco se dibuja hacia el estado 1-3; con la etiqueta b salen dos arcos desde el estado 1 original, uno hacia el estado 2 y otro al estado 3, por lo tanto el arco se dibuja al estado 2-3.



- Una vez que hemos terminado todos los pasos, podremos eliminar aquellos estados que sean superfluos al diagrama que acabamos de obtener.

  En nuestro caso particular podemos eliminar los estados 2, 3, 1-2 y 1-2-3, quedando el definitivo autómata finito determinista.



Unidad V

Gramáticas Formales

Una gramática esta formada por la cuádrupla:

(N, T, P, S)

donde:

N: vocabulario no terminal de símbolos que nosotros introducimos como elementos auxiliares para la definición de la gramática y que no figura en las sentencias del lenguaje

T: vocabulario terminal, todas las sentencias del lenguaje definidas por esta gramática están formadas por los símbolos o caracteres terminales

P: es un conjunto de reglas de derivación de las cadenas de la forma:

Cadena 1 $\rightarrow$ Cadena 2

S: símbolo inicial, denominado también símbolo distinguido o axioma

Una gramática describe de forma natural la estructura jerárquica de muchas construcciones de los lenguajes de programación.
De una gramática se derivan cadenas comenzando con el símbolo inicial y reemplazando repetidamente un no terminal por el lado derecho de una producción para ese no terminal. Las cadenas de componentes léxicos derivadas del símbolo inicial forman el lenguaje que define la gramática

Nota.

- En una producción los elementos léxicos, como palabras clave se denominan *componentes léxicos*
- Se dice que una producción es para un no terminal si el no terminal aparece en el lado izquierdo de la producción

CONSIDERACIONES DE NOTACION

1. Los elementos T o “vocabulario terminal” pueden ser:
2. letras minúsculas del comienzo del abecedario
   a) operadores
   b) caracteres especiales
   c) dígitos numéricos
   d) palabras reservadas de un lenguaje de programación
3. Los elementos de N o “vocabulario no terminal” pueden ser:

a) letras mayúsculas del comienzo del alfabeto
b) nombres en minúscula
c) También se acepta la notación BNF para escribir el nombre de la metanoción entre paréntesis angulares.
d) El axioma o símbolo inicial

4. El alfabeto $\Sigma =$ NUT comprende terminales y no terminales
5. Las "cadenas" de caracteres que tienen símbolos terminales y no terminales ($\in$(NUT)*) se escriben con letras minúsculas del alfabeto griego. Por ejemplo:

$$A \rightarrow \alpha$$

6. Se indica la cadena nula con λ

CLASIFICACION

Según Chomsky tenemos las siguientes clasificaciones:

Tipo 0: o gramática con estructura de frase, esta gramática se caracteriza por que las producciones presentan la forma:

$$\alpha \rightarrow \beta \text{ donde } \alpha \in (NUT)^+ \quad \beta \in (NUT)^*$$

es decir que la parte izquierda de una producción (regla) puede ser una tira de símbolos cualesquiera de N y T pero la parte derecha puede ser además nula.

Tipo 1: o gramática sensible al contexto. Donde P presenta la forma:

$$\alpha A \beta \rightarrow \alpha \gamma \beta \text{ donde } A \in N, \alpha \text{ y } \beta \in (NUT)^*$$
$$\gamma \in (NUT)^+$$

esta gramática se denomina sensible al contexto por que A cambia por gamma, pero solo en el contexto formado por alfa . . . beta

Tipo2: o gramática libre de contexto. Donde la forma de P es:

$$A \rightarrow \alpha \text{ donde } A \in N \text{ y } \alpha \in (NUT)^*$$

Tipo 3: o gramáticas regulares. Donde P presenta una de las dos formas siguientes:

$$A \rightarrow aB$$
$$A \rightarrow a$$

En estos tipos de gramáticas, cada una de ellas es más restringida que la anterior, es decir comprende un número menor de lenguajes. La inclusión de una gramática en otra es propia, o sea hay como mínimo un lenguaje que pertenece a una de ellas y no a su restringida.

GRAMATICAS LIBRES DE CONTEXTO

DERIVACION

En G, se dice que la tira alfa "produce directamente" la tira beta, lo que se representa $\alpha \Rightarrow \beta$ si se puede escribir:

$$\alpha = \delta\ A\ \mu \qquad \beta = \delta\ \gamma\ \mu$$

para alguna cadena *delta, mu* (que pueden ser nulas) y además existe una regla de P que sea $A \rightarrow \gamma$

Aplicando repetidamente la regla de derivación directa:

$$\alpha = \gamma_0 \Rightarrow \gamma_1 \Rightarrow \gamma_2 \Rightarrow \gamma_3 \Rightarrow \ldots \Rightarrow \gamma_n = \beta \quad \text{para } n>0$$

lo cual se representa abreviadamente:

a) $\alpha \Rightarrow^{+} \beta$
b) $\alpha \Rightarrow^{*} \beta$

GRAMATICAS Y LENGUAJES

Lenguaje definido por una gramática

L(G) representa un lenguaje L definido por una gramática G, que es un conjunto de cadenas (palabras) de símbolos terminales (sentencias) y que se pueden derivar partiendo de S, empleando las reglas de P.

$$L(G) = \{ x / (S \Rightarrow^{*} x) \text{ and } (x \in T^{*}\}$$

Formas sentenciales

$$D(G) = \{ \alpha \mid (S \Rightarrow^{*} \alpha) \text{ and } \alpha \in (NUT)^{*}\}$$

Derivación izquierda

Derivación realizada sustituyendo en la forma sentencial dada, la metanoción, de "más a la izquierda" por alguna de sus partes derechas que la definen

Derivación izquierda

Cuando las derivaciones que se realizan en la forma sentencial son siempre en la metanoción de "más a la derecha"

Unidad VI

Aplicación de lenguajes Formales a Lenguajes de Programación

Lenguaje.

Es un conjunto de oraciones basadas en una gramática, que a su vez se basa en un alfabeto; un lenguaje es generalmente infinito y se forma con distintas combinaciones de palabras. Es necesario que estas combinaciones sean bien formadas, es decir, correctas basadas en una sintaxis y una semántica propia del lenguaje. Un lenguaje generalmente nos sirve para poder expresar pensamientos ideas y de esta forma poder comunicarnos con las demás personas.

Clasificación de los lenguajes

Los lenguajes pueden ser naturales o formales.

Lenguajes naturales. Son aquellos lenguajes que se han desarrollado y organizado gracias a la experiencia humana; generalmente usamos los lenguajes naturales para comunicarnos con los demás, para analizar distintas situaciones y razonar sobre las mismas, tienen un gran poder expresivo y por consiguiente son polisemánticos, es decir, una misma expresión puede tener distintos significados dependiendo del contexto.

Características de los lenguajes naturales

- son espontáneos, se han desarrollado progresivamente, enriqueciendose gracias a la experiencia humana;
- Son polisemánticos gracias a la riqueza de componentes y a la variedad de sus expresiones.
- No pueden ser formalizados completamente, precisamente por ser polisemánticos.

Lenguajes formales

Son lenguajes que el hombre ha desarrollado de manera objetiva para expresar y almacenar conocimiento científico. Cada palabra y oración está perfectamente definido porque tiene un solo significado que no depende del contexto.

Los lenguajes formales pueden ser usados en cualquier área de investigación para describir de manera precisa y concisa hechos y acontecimientos libres de toda ambigüedad.

Características de los lenguajes formales.

- Se desarrollan basados en una teoría previamente establecida.
- Tienen pocos componentes semánticos que pueden ser incrementados de acuerdo con la teoría que se desea formalizar.
- No producen oraciones ambiguas gracias a que no son poli semánticos.
- Los números son muy importantes en todo lenguaje formal.
- Se pueden formalizar completamente.

Lenguaje de programación

Es una combinación de un lenguaje natural y de un lenguaje formal, para poder escribir programas que pueden ser entendidos y ejecutados por un computador. Está compuesto por dos elementos importantes.

Sintaxis. Cada sentencia de un programa debe ser escrito correctamente.

Semántica. Cada sentencia de un programa debe tener un significado correcto y único.

Generaciones de los lenguajes de programación.

Desde la aparición de las computadoras, los lenguajes de programación han ido evolucionando, distinguiéndose cinco generaciones de lenguajes de programación.

Primera generación. Son conocidos como los lenguajes máquina basados en el sistema de numeración binario, en el que se utiliza solamente unos y ceros para comunicarse con la computadora, programar en esta generación era muy complicado, porque no se podían expresar programas complejos. Actualmente son muy poco utilizados, muy necesarios para la programación de chips.

Segunda generación. En esta generación surgen los lenguajes ensambladores, se basan en el uso de acrónimos, que se resumen de varias palabras en una sola. Por ejemplo la sentencia ADC que significa "ADd with Carry", en español "sumar con acarreo".

Tercera generación. En esta generación surgen los llamados lenguajes de alto nivel, los más representativos son: el C, Pascal, Basic, Cobol, Fortran. Se asemejan más a lenguaje humano, usando palabras completas del inglés para escribir programas. Por ejemplo writeln("Hola Mundo"), que significa imprimir en una línea nueva "Hola Mundo".

Cuarta generación. En esta generación surgen los lenguajes visuales, que cuentan con asistentes llamados Wizards para facilitar el diseño y la implementación de los programas. Se asemejan mucho más lenguaje humano, utilizando incluso frases de lenguaje natural, los más representativos son:Visual Basic, Visual C, Visual Java.

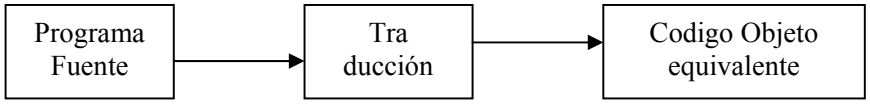
Quinta generación. En esta generación los programadores ya no programan, ya no se preocupan sobre cómo están implementados los programas; simplemente se dedican a ingresar datos y hacer consultas y el computador responde a los mismos.

Unidad VII

Compiladores

1. LOS TRADUCTORES.

La computadora sólo puede ejecutar instrucciones escritas en un lenguaje formado por secuencias de ceros y unos, al que normalmente se le denomina lenguaje máquina. Por ello, cualquier lenguaje de programación que no sea lenguaje máquina necesitará un proceso de traducción. Este proceso lo llevan a cabo las computadoras.

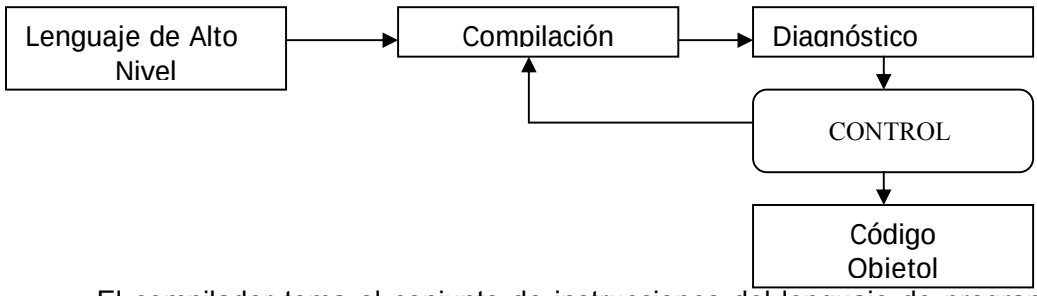


Es una máquina teórica que tiene como entrada un texto escrito en un lenguaje L1 y como salida un texto escrito en un lenguaje L2. Habitualmente se denomina a L1 lenguaje fuente y a L2 lenguaje objeto.

Traductores de lenguaje natural: Serían los que tradujeran un lenguaje natural en otro (por ejemplo, español a inglés). Esto en la actualidad no se ha conseguido debido fundamentalmente a la ambigüedad del lenguaje natural. Los mayores logros en la materia siempre trabajan con un subconjunto del lenguaje natural, limitando las construcciones sintácticas válidas y/o el vocabulario. Este tema se aborda generalmente mediante técnicas de inteligencia artificial.

a) COMPILADORES. Los compiladores traducen las sentencias o instrucciones del lenguaje de programación y las convierten en un conjunto de instrucciones en lenguaje máquina directamente ejecutables de la computadora.

El proceso de traducción con compilador no se realiza simultáneamente, como en el caso de los interpretes, sino que se hace en un proceso aparte. No hay diálogo con el programador durante la programación, ni es posible ir probando partes del programa.



El compilador toma el conjunto de instrucciones del lenguaje de programación, que se denomina **programa fuente**, como datos de entrada y las convierte en instrucciones de lenguaje generalmente máquina, cuyo conjunto pasa a denominarse **programa objeto**. Como parte importante de este proceso de traducción, el compilador informa al usuario la presencia de errores.

De esta manera, cuando la CPU ejecuta el programa ya no necesita un traductor, porque ella misma ejecuta directamente las instrucciones del programa ya traducido. Además de un compilador, se pueden necesitar otros programas para crear un programa objeto ejecutable.

b) INTERPRETES. El trabajo de un interprete, como su nombre indica, es el de traducir las instrucciones de los lenguajes de programación al lenguaje máquina, de tal manera que la CPU puede ejecutarlas.

> Tanto el programa usuario como el programa interprete se encuentran en memoria; la traducción es simultánea y se produce de forma dialogada con el programador; es posible ir probando progresivamente porciones del programa.

El intérprete es un traductor que realiza la operación de compilación paso a paso. Para cada sentencia que compone el texto de entrada, se realiza una traducción, ejecuta dicha sentencia y vuelve a iniciar el proceso con la sentencia siguiente. La principal ventaja del proceso de compilación frente al de interpretación es que los programas se ejecutan mucho más rápidamente una vez compilados; por el contrario, es más cómodo desarrollar un programa mediante un intérprete que mediante un compilador puesto que en el intérprete las fases de edición y ejecución están más integradas. La depuración de los programas suele ser más fácil en los intérpretes que en los compiladores puesto que el código fuente está presente durante la ejecución. Estas ventajas pueden incorporarse al compilador mediante la utilización de entornos de desarrollo y depuradores simbólicos en tiempo de ejecución.

El usuario generalmente no llega a notar la diferencia entre compilador e intérprete

2. FUNCIONES DE UN COMPILADOR

Un compilador básicamente cumple dos funciones:

a) Establecer la validez de la cadena de entrada dentro de la estructura gramática del lenguaje
b) Proporcionar una cadena de salida escritas bajo la estructura gramatical del lenguaje objeto

Una construcción bajo la estructura gramatical se conoce como programa. Un programa se construye con varias sentencias. Una sentencia se construye en base de unidades sintácticas llamadas **tokens o componentes léxicos.**

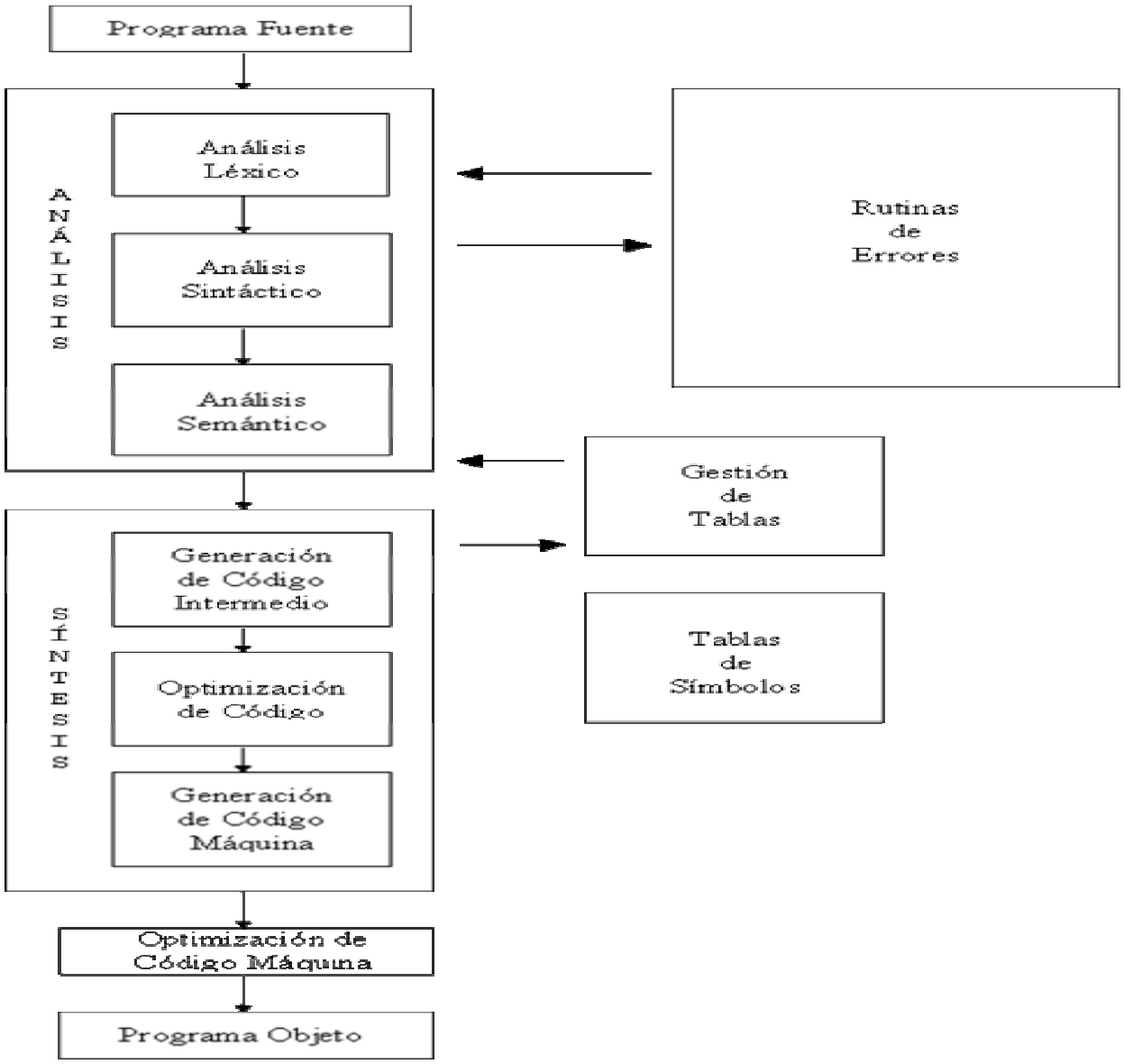
Normalmente cualquier compilador incluye sentencias de preprocesamiento para hacer más ligero el proceso de compilación.

- Para un lenguaje no ambiguo se tiene un autómata determinístico
- Para un lenguaje ambiguo se tiene un autómata determinístico

3. ESTRUCTURA DE UN COMPILADOR

Un *compilador* es un “programa”, en el que pueden distinguirse dos subprogramas o fases principales: una fase de *análisis*, en la cuál se lee el programa fuente y se estudia la estructura y el significado del mismo; y otra fase de *síntesis*, en la que se genera el programa objeto.

En un compilador pueden distinguirse, además, algunas estructuras de datos comunes, la más importante de las cuales es la tabla de símbolos, junto con las funciones de gestión de ésta y de los demás elementos del compilador, y de una serie de rutinas auxiliares para detección de errores.



Las funciones de estos módulos son las siguientes:

a) Analizador lexicográfico: Las principales funciones que realiza son:

- Identificar los símbolos.
- Eliminar los blancos, caracteres de fin de línea, etc...
- Eliminar los comentarios que acompañan al fuente.
- Crear unos símbolos intermedios llamados tokens.
- Avisar de los errores que detecte.

Ejemplo: A partir de la sentencia en PASCAL siguiente

nuevo := viejo + RAZON*2

se genera un código simplificado para el análisis sintáctico posterior, por ejemplo:

<id1> <:=> <id2> <+> <id3> <*> <ent>

Nota: Cada elemento encerrado entre <> representa un único *token*. Las abreviaturas id y ent significan identificador y entero, respectivamente.

b) Analizador sintáctico: Comprueba que las sentencias que componen el texto fuente son correctas en el lenguaje, creando una representación interna que corresponde a la sentencia analizada. De esta manera se garantiza que sólo serán procesadas las sentencias que pertenezcan al lenguaje fuente. Durante el análisis sintáctico, así como en las demás etapas, se van mostrando los errores que se encuentran.

Ejemplo: El esquema de la sentencia anterior corresponde al de una sentencia de asignación del lenguaje Pascal. Estas sentencias son de la forma:

<id> <:=> <EXPRESION>

y la parte que se denomina <EXPRESION> es de la forma:

- <id> <+> <EXPRESION> o bien
- <id> <*> <EXPRESION> o bien
- <real>

La estructura de la sentencia queda, por tanto, de manifiesto mediante el siguiente esquema:

<id1> < :=> <EXPRESION> <id2> <+> <EXPRESION>

c) Análisis semántico: Se ocupa de analizar si la sentencia tiene algún significado. Se pueden encontrar sentencias que son sintácticamente correctas pero que no se pueden ejecutar porque carecen de sentido. En general, el análisis semántico se hace a la par que el análisis sintáctico introduciendo en éste unas rutinas semánticas.

Ejemplo: En la sentencia que se ha analizado existe una variable entera. Sin embargo, las operaciones se realizan entre identificadores reales, por lo que hay dos alternativas: o emitir

un mensaje de error "Discordancia de tipos", o realizar una conversión automática al tipo superior, mediante una función auxiliar inttoreal.

<id1> <:=> <EXPRESION> <id2> <+> <EXPRESION>

d) Generador de código intermedio: El código intermedio es un código abstracto independiente de la máquina para la que se generará el código objeto. El código intermedio ha de cumplir dos requisitos importantes: ser fácil de producir a partir del análisis sintáctico, y ser fácil de traducir al lenguaje objeto. Esta fase puede no existir si se genera directamente código máquina, pero suele ser conveniente emplearla.

Ejemplo: Consideremos, por ejemplo, un código intermedio de tercetos, llamado así porque en cada una de sus instrucciones aparecen como máximo tres operandos. La sentencia traducida a este código intermedio quedaría :

temp1 := inttoreal (2)temp2 := id3 * temp1temp3 := id2 + temp2id1 := temp3

e) Optimizador de código: A partir de todo lo anterior crea un nuevo código más compacto y eficiente, eliminando por ejemplo sentencias que no se ejecutan nunca, simplificando expresiones aritméticas, etc... La profundidad con que se realiza esta optimización varía mucho de unos compiladores a otros. En el peor de los casos esta fase se suprime.

Ejemplo: Siguiendo con el ejemplo anterior, es posible evitar la función inttoreal mediante el cambio de 2 por 2.0, obviando además una de las operaciones anteriores. El código optimizado queda como sigue :

temp1 := id3 * 2.0id1 := id2 + temp1

Generador de código: A partir de los análisis anteriores y de las tablas que estos análisis van creando durante su ejecución produce un código o lenguaje objeto que es directamente ejecutable por la máquina. Es la fase final del compilador. Las instrucciones del código intermedio se traducen una a una en código máquina reubicable.

Nota: Cada instrucción de código intermedio puede dar lugar a más de una de código máquina.

Ejemplo: El código anterior traducido a ensamblador DLX quedaría:

LW R1,id3MUL R1,R1,2LW R2,id2ADD R2,R2,R1SW id1,R2

en donde id1, id2 y id3 representan las posiciones de memoria en las que se hallan almacenadas estas variables; R1 y R2 son los registros de la máquina; y las instrucciones LW, SW, MUL y ADD representan las operaciones de colocar un valor de memoria en un registro, colocar un valor de un registro en memoria, multiplicar en punto flotante y sumar, respectivamente.

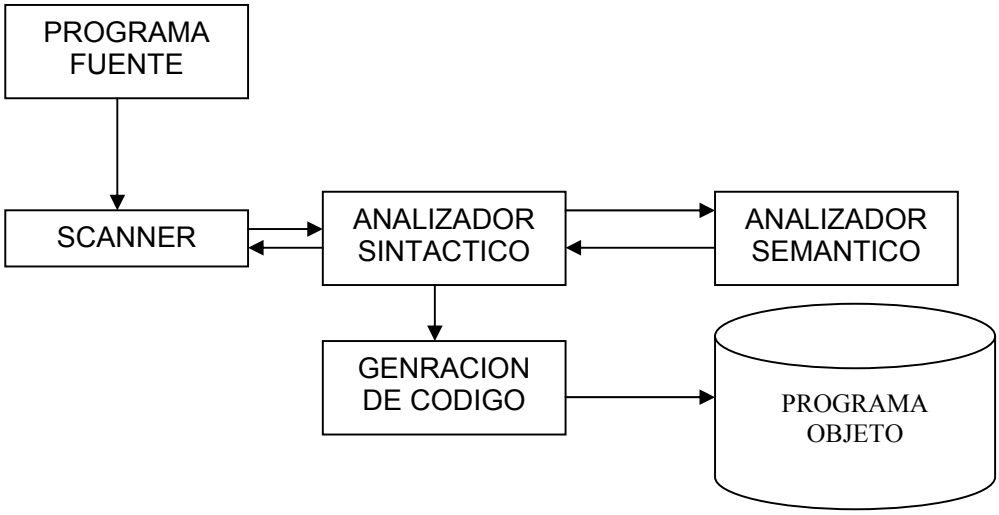
f) La tabla de símbolos: Es el medio de almacenamiento de toda la información referente a las variables y objetos en general del programa que se está compilando.

Ejemplo: Hemos visto que en ciertos momentos del proceso de compilación debemos hacer uso de cierta información referente a los identificadores o los números que aparecen en nuestra sentencia, como son su tipo, su posición de almacenamiento en memoria, etc... Esta información es la que se almacena en la tabla de símbolos.

g) Rutinas de errores: Están incluidas en cada uno de los procesos de compilación (análisis lexicográfico, sintáctico y semántico), y se encargan de informar de los errores que encuentran en texto fuente.

Ejemplo: El analizador semántico podría emitir un error (o al menos un aviso) cuando detectase una diferencia en los tipos de una operación.

4. FORMA INTERNA DEL PROGRAMA FUENTE



Es la representación interna del programa que dependerá en gran parte de cómo se va ha manejar después este programa.

El proceso de preparación para la generación de código es el encargado de manipular y cambiar la forma interna asignando memoria a todas las variables para el tiempo de ejecución.

El proceso de generación de código es la parte mas detallada y pesada del compilador.

Anexo 2
Prácticas

Práctica Inicial
Lenguajes Formales y Compiladores

RESOLVER LOS SIGUIENTES EJERCICIOS

1. Sea la relación $R \subset N \times Z^{-}$ (negativos) y $(a,b) \in R \Longleftrightarrow a>b$
   R={ , , , , ...} (MOSTRAR 4 elementos POR EXTENSION)
2. Definir formalmente la noción de "función".
3. Si $h : N \rightarrow N^{-0}$ (menos 0) y $h(x)=2x+1$. Es h una función? (Si, No-Justificar respuesta)

4. Si $h : R \rightarrow R$, $g : N \rightarrow R$, $h(x)=2x$ y $g(x)=(2x+1)/2$

Obtener si es posible:
a) $[h \circ g](x)=$
b) $[g \circ h](x)=$

Práctica 1
Lenguajes Formales y Compiladores

1. Cuantificar y expresar como función proposicional la siguiente proposición:
"Existen enteros que no son pares "

2. Verificar si la siguiente proposición es una contradicción o una tautología
(p→ q) ^ (p ^ -q)

3. Si f : Z → R y f(x) = 2/(x - 1). Porque f no es una función?
Recordar:"f es una función o aplicación de A en B si y solo si f es una relación entre A y B tal que **todo** elemento de A tiene un único correspondiente en B".

4. Sea ∑={a}. Si w es una cadena de ∑*

a) Es verdad que para todo número natural **n** hay alguna cadena **w** para la cual |w| = n, n Є N^0 (naturales incluído el 0)?

b) Si se cumple: |w| = n, n Є N^0 ¿w es única?

Realizar el análisis y la justificación de su respuesta.

5. Sea w = 10101 definida sobre ∑={0,1}

Hallar : a**)** todos los prefijos
b) todas las subcadenas

6. Sea w = 11001 definida sobre ∑={0,1}

Hallar : a**)** $\mathbf{w'}$ =
b) $\mathbf{w^3}$ =
c**)** $\mathbf{w^0}$ =
d) **|w| =**

7. Sean los lenguajes A={a,b} y B={bb,aa} sobre ∑={a,b}

Hallar:

a) **A.B**

b**)** $\mathbf{A^2}$

c) $\sum^2$

8. Si A= {λ, 1}es un lenguaje sobre ∑={1}

a. Obtener $\mathbf{A^n}$ para n=0,1,2

b. Cuantos elementos tiene $\mathbf{A^n}$**,** para cualquier n Є N^0

Anexo 3
Presentaciones