UNIVERSIDAD POLITECNICA DE VALENCIA

DEPARTAMENTO DE SISTEMAS INFORMATICOS Y COMPUTACION

Sobre algunos rasgos característicos de las gramáticas lineales y su aplicación al estudio de su identificación y su complejidad

Tesis doctoral presentada por D. José M. Sempere Luna.
Dirigida por Dr. D. Pedro García Gómez

Valencia, Octubre de 2002

"*Una obra está acabada cuando no puede ya ser mejorada, aunque se la sepa insuficiente e incompleta. Se está tan exageradamente fatigado de ella que ya no se tiene el valor de añadirle ni una sola coma, aunque fuese indispensable. Lo que decide el grado de acabado de una obra no es en absoluto ninguna exigencia del arte o de la verdad, es la fatiga y, aún más, el asco*"

E.M. CIORAN

Índice General

Índice de Figuras

Capítulo 1

Introducción

El presente trabajo se centra en el área del aprendizaje computacional de lenguajes formales a través de su representación gramatical. El aprendizaje automático es un objetivo perseguido durante mucho tiempo por las comunidades científicas dedicadas a las distintas áreas de la computación. Sus beneficios cuentan, entre otros, la reducción de la fase de programación de sistemas, la resolución no explícita y adaptativa de problemas complejos, y la constante reflexión acerca de la capacidad de resolución de problemas mediante métodos automáticos.

La tesis que aquí se presenta trata acerca del aprendizaje de una clase de lenguajes utilizando un criterio de éxito ya formulado y la resolución de los problemas relacionados con el mismo, entre los que ocupa un papel principal la complejidad descriptiva de las hipótesis formuladas.

En este trabajo nos centraremos en las clases de lenguajes *lineales*. La clase de los lenguajes lineales es de especial interés ya que plantea los problemas característicos que limitan o impiden cierto tipo de aprendizaje: determinismo vs. no determinismo, ambigüedad, equivalencia, etc. Así, nos encontramos en un escenario que nos permite obtener conclusiones válidas para otras clases de lenguajes cuyo interés es, hoy en día, notable.

A lo largo de la presente memoria, trabajaremos bajo el criterio de éxito de *identificación en el límite*. El criterio seleccionado, dentro de la variedad de criterios posibles (identificación estocástica, identificación PAC, minimización de entropía, etc.) ha sido ampliamente aceptado por la comunidad científica desde su formulación a mediados de los años sesenta. Este marco de trabajo no impone restricciones acerca de la presentación de ejemplos para los algoritmos de aprendizaje ni tampoco exige un aprendizaje limitado en el tiempo. Más bien, nos encontramos en un entorno de trabajo similar al que se lleva a cabo

en el aprendizaje por parte de los organismos vivos, especialmente en el aprendizaje del lenguaje por parte de los seres humanos tal y como se ha estudiado en multitud de trabajos por parte de logopedas, lingüistas y psicólogos.

Otro aspecto que se estudia en este trabajo es el de la complejidad descriptiva de la clase de los lenguajes lineales. Nos gustaría destacar el hecho de que la identificación de clases de lenguajes y el análisis de la complejidad de las citadas clases, tanto descriptiva como computacional, no son áreas que, como pudiera parecer, no tengan relación. Sucede más bien todo lo contrario. Cuando se reflexiona acerca de cuáles son las representaciones idóneas para representar clases de lenguajes identificables surge, de forma natural, el análisis descriptivo de dichas clases. Además, un conjunto característico de un lenguaje (si es que lo tuviera) es en sí mismo una descripción del lenguaje junto con el algoritmo de inferencia que lo identifica.

La estructura de esta memoria es como sigue: En el capítulo 2 se introducen los conceptos que, a juicio del autor, son más interesantes para centrar el aprendizaje automático en general y el de lenguajes formales en particular. En el capítulo 3 se proporcionan los conceptos básicos de la teoría de autómatas, gramáticas y lenguajes formales necesarios para formalizar los conceptos con los que se trabajaran posteriormente. En el capítulo 4 se estudia la identificación de la clase de los lenguajes *lineales pares*. Se introducen rasgos característicos de algunas gramáticas y se formulan distintos algoritmos de aprendizaje. En el capítulo 5 se estudia la identificación de los lenguajes lineales. Se introduce la necesidad de utilizar información estructural y se prescinde de la utilización de datos negativos o contraejemplos. Esto producirá la definición de algunas subclases de lenguajes lineales y se resolverá su identificación en el límite. En el capítulo 6 se estudian algunos aspectos de complejidad descriptiva y computacional de los lenguajes lineales. Nos centraremos en la complejidad de Kolmogorov y la complejidad de reverso. Previamente, se habrá definido una clase de representaciones que nos ayudarán en el estudio de la complejidad referido anteriormente. Por último, en el capítulo 7 se establecen las conclusiones generales que se pueden extraer del presente trabajo. Destaquemos el hecho de que en los capítulos 4, 5 y 6 se presentarán las conclusiones relativas a cada uno de ellos y las posibles líneas de investigación que podrán originar trabajos futuros.

Capítulo 2

El problema del aprendizaje computacional

Según el Diccionario de la Lengua Española de la Real Academia en su edición de 1992, *aprender* significa *"Tomar algo en la memoria ..."* y también *"... adquirir el conocimiento de alguna cosa por medio del estudio o de la experiencia"*. Desde el punto de vista computacional, la primera de las dos acepciones es la menos interesante ya que cualquier modelo computacional es capaz de almacenar en memoria información. Es la segunda acepción la que interesa en la presente tesis, es decir, el estudio de la adquisición de conocimiento (información) a partir de experiencia (información). La discusión epistemológica acerca de si el conocimiento se puede adquirir de forma automática y, en caso afirmativo, es un indicio de racionalidad no es objeto de discusión en esta tesis. El planteamiento general en el que se enmarca esta tesis, desde un punto de vista axiomático, es que el aprendizaje computacional es una forma de programación no explícita, por la que un modelo de computación puede llegar a realizar determinadas tareas tomando como entrada información relacionada con las mismas y variando el esquema de resolución de las citadas tareas de forma automática con el fin de obtener mayores prestaciones.

Según Mitchell [49] el problema del aprendizaje computacional se puede definir de la siguiente forma : *" Diremos que un programa aprende de la experiencia* **E** *respecto de una clase de tareas* **T** *y una medida de eficiencia* **P**, *si su eficiencia en las tareas de* **T**, *tal y como las mide* **P**, *se incrementan con la experiencia* **E**". Según esta definición existen tres aspectos distintos que pueden definir un problema de aprendizaje : Primero, la selección de una clase de tareas. Segundo, la definición de una medida de la eficiencia. Por último, el diseño de un protocolo de intercambio de información que defina la

experiencia.

Según el tipo de resolución que se realice para aumentar la eficiencia de un programa, podemos hablar de distintos paradigmas de aprendizaje. Sin embargo, el establecimiento de una taxonomía de los métodos de aprendizaje no resulta fácil y, en definitiva, un mismo método puede ser clasificado de distintas formas en referencia a los distintos aspectos del mismo. No obstante, podemos establecer, a grandes rasgos, algunos paradigmas diferenciadores del aprendizaje como son los siguientes

1. **Aprendizaje por refuerzo**

   En este caso, el objetivo del programa se puede establecer en términos de optimización de una función de ganancia desconocida. Así, mediante *recompensas* o *castigos* que recibe el programa como información de entrada se va ajustando en sucesivas fases la función desconocida. Este tipo de aproximación se suele adoptar en tareas referidas a la robótica, la planificación y, en definitiva, en aquellos sistemas que deban adaptarse a entornos que sufren grandes cambios a lo largo del tiempo. El algoritmo de aprendizaje más conocido bajo esta perspectiva es el conocido como *Aprendizaje Q*.

2. **Aprendizaje por analogía**

   El objetivo consiste en aprender una serie de objetos (conceptos) mediante el establecimiento de medidas de similitud entre los mismos. La representación de los objetos puede ser tanto simbólica como numérica (en espacios n-dimensionales). En el último caso, se pueden establecer técnicas de agrupamiento (*clustering*) que dan lugar a toda una familia de métodos de aprendizaje basados en medidas geométricas o probabilistas.

3. **Aprendizaje conexionista**

   El modelo más conocido bajo esta aproximación son las redes neuronales. Existe una vasta literatura acerca del tema. Fundamentalmente, el principio que guía el aprendizaje conexionista se basa en la especialización de modelos muy simples (neuronas) y la interconexión de esos modelos que puede dar lugar a la aparición de propiedades emergentes en el modelo global. De nuevo, existen vertientes que trabajan con información simbólica así como numérica y aplican técnicas de análisis matemático (como por ejemplo el *descenso por gradiente* en el modelo *perceptrón*).

4. **Aprendizaje deductivo**

   Este es uno de los principios clásicos utilizados en el área de la inteligencia artificial. Básicamente, el aprendizaje deductivo consiste en, a partir de una serie de hechos comprobados y de reglas lógicas, establecer nuevas reglas, por deducción lógica, que aumenten el conocimiento acerca de un fenómeno. Las técnicas que se utilizan son muy variadas, desde los silogismos clásicos más simples hasta técnicas de ramificación y poda o programación dinámica en espacios de búsqueda de reglas lógicas.

5. **Aprendizaje inductivo**

   El principio inductivo se puede enunciar como *la formulación de reglas generales a partir de hechos específicos.* A diferencia del paradigma deductivo, aquí el conocimiento aumenta por observación y descubrimiento y no por la aplicación de reglas lógicas, como sucedía en el paradigma deductivo. Existen multitud de aplicaciones de este principio algunas de las cuales las detallaremos en la siguiente sección.

En cuanto al protocolo de información o el tipo de información disponible para el método de aprendizaje se pueden establecer dos categorías

1. **Información supervisada**

   El sistema recibe la información junto con una declaración explícita de su significado. Así, si el conocimiento se representa mediante lenguajes formales, la información que se recibe toma la forma de cadenas etiquetadas según su pertenencia o no al lenguaje en cuestión. Si la información toma forma de valores numéricos, se pueden añadir, además, etiquetas sobre la significación de ese valor (por ejemplo, en problemas de clasificación añadir la clase a la que pertenece el punto que representa el valor).

2. **Información no supervisada**

   A diferencia del caso anterior la información que recibe el sistema no está clasificada ni tampoco tiene ninguna etiqueta adicional. Es decir, la representación de un objeto o hecho no está especificada como tal. Este tipo de información es típica de los sistemas de aprendizaje por similitud y agrupamiento.

El tercer aspecto a considerar, en cuanto al planteamiento general de los métodos de aprendizaje, se centra en el método de medición de la eficiencia o, dicho de otra forma, el *criterio de éxito* que soporte el método de aprendizaje.

Este aspecto puede variar mucho en función del paradigma de aprendizaje adoptado y de la forma de representar la información (simbólica, numérica, etc). En general podemos adoptar dos criterios que pueden ser aplicados a la mayoría de los casos : la *identificación* y la *aproximación*. El primero establece que el sistema, para concluir su tarea, debe identificar el concepto a aprender de forma exacta mientras que en el segundo, establecida una medida de error, se permite que el sistema aprenda lo suficiente como para quedarse debajo de un umbral de error preestablecido.

Existen otros aspectos a considerar en lo relativo a la eficiencia de un sistema que hacen referencia a aspectos computacionales. Así, independientemente de que un método de aprendizaje aproxime o identifique un concepto, existen ocasiones en las que aspectos como la rapidez con la que el sistema realiza sus acciones (*complejidad temporal*) o la cantidad de información que el sistema necesita para tener éxito son fundamentales a la hora de abordar una tarea concreta.

2.1 Inferencia inductiva

La inferencia inductiva se puede aplicar como paradigma de aprendizaje computacional en numerosas tareas. Existen distintos enfoques acerca de la inferencia inductiva. En la presente tesis seguiremos fundamentalmente, entre otros, a Solomonoff [69], Osherson *et al.* [52] y Angluin y Smith [5]. De igual forma, existen otros trabajos que suponen una excelente revisión de este planteamiento general [12, 53].

Un problema en inferencia inductiva supone la definición de tres aspectos fundamentales (como caso particular del problema genérico del aprendizaje [49]) tal y como detallamos a continuación

1. **Definición de una relación de nombres para conceptos e hipótesis**

   Fundamentalmente, la relación de nombres consiste en una aplicación que permita asignar a los objetos del problema que se pretende resolver formalismos matemáticos de un espacio predeterminado. Este espacio puede tener una serie de propiedades algebraicas (*espacio de versiones*). En el área que nos ocupa podemos tomar como relación de nombre las gramáticas formales, las funciones recursivas o las máquinas abstractas (autómatas). La elección del espacio de hipótesis predetermina el algoritmo de búsqueda o las técnicas de construcción a emplear.

2. **Definición de un protocolo de información**

   En este caso, podemos establecer dos tipos de protocolos : *texto* e *informante*. El texto consiste en proporcionar al metodo de inferencia sólo ejemplos del concepto que debe aprender, mientras que en el informante, se pueden proporcionar ejemplos y contraejemplos. Existe una variedad de formas de representar texto e informante dependiendo de las propiedades que tenga la función que lo genera (si es arbitraria, recursiva, recursiva primitiva, etc). El protocolo de información es importante ya que puede limitar las clases de conceptos que se pueden aprender según el criterio de éxito adoptado. De este último hablaremos a continuación.

3. **Definición de un criterio de éxito**

   El criterio de éxito define cuándo el algoritmo de inferencia ha tenido éxito en el proceso de búsqueda de la hipótesis que da cuenta de los ejemplos (y contraejemplos) proporcionados como fuente de información. Vamos a plantear dos criterios de éxito que en la actualidad son los que gozan de mayor popularidad en la comunidad científica.

   (a) Identificación en el límite

   Este criterio lo propuso Gold en 1967 [24]. Informalmente, el criterio consiste en identificar la hipótesis de forma exacta sin importar el momento de tiempo en el que se consiga. Así, suponiendo que tomamos un espacio de hipótesis consistente en la clase de lenguajes recursivamente enumerables, si el lenguaje a identificar es $\mathcal{L}$ y el algoritmo proporciona una secuencia de posibles hipótesis $\mathcal{H}_1, \mathcal{H}_2, \cdots, \mathcal{H}_n, \cdots$, llegará el momento de tiempo i en el que la hipótesis propuesta por el algoritmo será $\mathcal{H}_i$ que representa a $\mathcal{L}$ y ya no se vuelve a cambiar de hipótesis en momentos posteriores. De esta forma diremos que el algoritmo identifica al lenguaje $\mathcal{L}$ *en el límite*.

   (b) Aproximación probablemente correcta (PAC)

   Propuesto por Valiant en 1984 [74], el criterio asume una distribución de probabilidad sobre los lenguajes a identificar. Así, dado el lenguaje $\mathcal{L}$, se supone que las cadenas de $\mathcal{L}$ tienen una distribución de probabilidad de aparición $\mathcal{P}$. Además se considera un *error de predicción* ϵ y un *parámetro de confianza* δ. El objetivo es que el algoritmo consiga proponer una hipótesis $\mathcal{L}'$ de forma que la acumulación del error sea muy baja con una confianza elevada, es decir

$Pr[\mathcal{P}(\mathcal{L} \triangle \mathcal{L}') \leq \epsilon] \geq 1 - \delta$, donde $(\mathcal{L} \triangle \mathcal{L}')$ denota la diferencia simétrica entre $\mathcal{L}$ y $\mathcal{L}'$. Diversas variantes sobre este esquema general, así como algunos resultados significativos en aproximación pueden verse en [50].

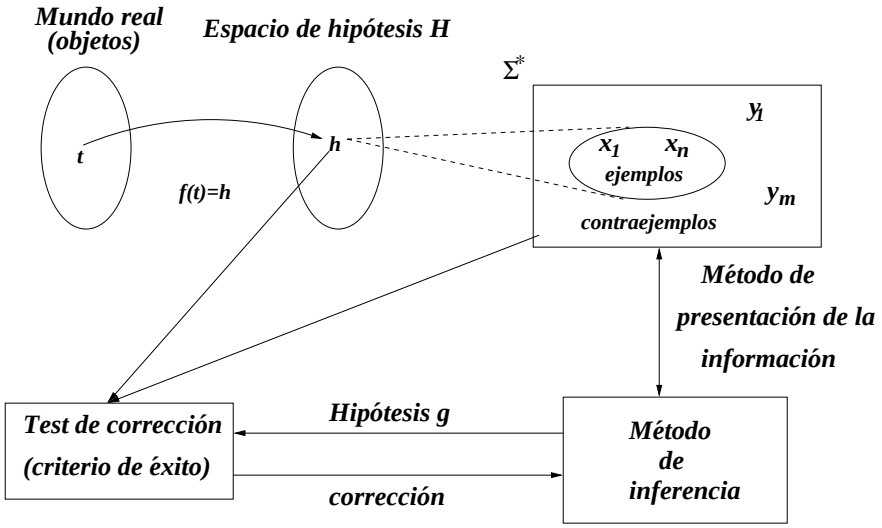


Figura 2.1: Esquema general para la definición de un problema en inferencia inductiva.

En la figura 2.1 podemos ver el esquema genérico de un problema de inferencia inductiva. Así, la función f es la relación de nombre que especifica el espacio de hipótesis $\mathcal{H}$ y Σ es el alfabeto de definición de ejemplos y contraejemplos, teniendo en cuenta que pueden ser no sólo cadenas sino árboles, grafos, etc. El método de presentación de la información especifica si se permiten ejemplos y contraejemplos, en qué orden (si es que existe) se le proporciona al algoritmo o método de inferencia, si se proporcionan de manera incremental, finita o en el límite, etc. Por último, el test de corrección evalúa la hipótesis proporcionada por el método de inferencia a partir de la hipótesis objetivo y de la información proporcionada al método.

2.2 Inferencia gramatical

La inferencia gramatical [19, 48, 64] es un caso particular del paradigma de inferencia inductiva donde las hipótesis y los conceptos se representan mediante

formalismos que provienen de la teoría de lenguajes formales, típicamente gramáticas y autómatas. Así, tomando como marco de referencia la citada teoría, el objetivo del método de aprendizaje es identificar o aproximar un lenguaje definido sobre un alfabeto preestablecido dando como relación de nombre una gramática (enfoque *generador*) o un autómata (enfoque *aceptor*). Una de las grandes ventajas que implica trabajar bajo el paradigma de inferencia gramatical es que cuenta con un gran número de resultados teóricos y prácticos establecidos a lo largo del tiempo por la comunidad de trabajo en informática teórica (*Theoretical Computer Science*). Estos resultados serán de gran ayuda a la hora de establecer clases de lenguajes efectivamente inferibles, propiedades algebraicas de los métodos de aprendizaje, resultados (positivos y negativos) acerca de cual es el límite computacional de lo que se puede llegar a aprender, etc.

Básicamente, los métodos aplicados en inferencia gramatical los podemos agrupar en dos tipos:

1. **Métodos caracterizables**

   Un método diremos que es *caracterizable* si el espacio de hipótesis forma una clase de lenguajes que queda caracterizada por el propio método. Es decir, un lenguaje pertenece a la clase si y sólo si puede ser inferido por el método. Dicho de otra forma, la clase de lenguajes inferidos por el método forma una clase de lenguajes bajo el punto de vista de la teoría de lenguajes formales.

2. **Métodos heurísticos**

   Un método heurístico toma, al igual que en el caso anterior, un espacio de hipótesis que es una clase de lenguajes, pero, a diferencia de los métodos caracterizables, no lo define. Es decir, pueden haber lenguajes de la clase que el método no sea capaz de inferir y, lo que es más importante, no se puede caracterizar la clase de lenguajes inferidos por el método mediante propiedades matemáticas.

Un método de inferencia no lo podemos clasificar como bueno o malo en función de si es caracterizable o heurístico. Por una parte, los métodos caracterizables presentan unas propiedades teóricas más robustas que los métodos heurísticos con lo que es más fácil predecir su comportamiento en función de la información recibida. Por otra parte, un método heurístico puede presentar mayores prestaciones en su aplicación a tareas reales que los métodos caracterizables ya que estos suelen ser más exigentes con el tipo de información

con la que trabajan, que es, por otra parte, lo contrario de lo que sucede en situaciones reales.

Es más interesante, sin embargo, la clasificación de los métodos de inferencia en función del tipo de información recibida. Hablaremos a continuación de algunos métodos y resultados significativos en función del tipo de información recibida.

2.3 Aprendizaje mediante presentación positiva

El aprendizaje a partir de información positiva consiste en, dado un lenguaje $\mathcal{L}$, proporcionar al método únicamente ejemplos del lenguaje, es decir, cadenas $\{x_1, \cdots, x_n, \cdots\}$ donde cada una de ellas pertenece a $\mathcal{L}$. Es lo que llamamos *texto* en el apartado sobre inferencia inductiva. Angluin [3] estableció condiciones bajo las que es posible inferir un lenguaje tomando únicamente texto como información disponible. El resultado de Angluin queda establecido bajo el siguiente teorema

Teorema 2.1.(Angluin [3]) Una familia indexada de lenguajes recursivos no vacíos es inferible a partir de datos positivos si y solo si se satisface que existe un procedimiento efectivo que, a partir de cualquier valor $i \geq 1$, enumera un conjunto de cadenas T_i de forma que

1. T_i es finito
2. $T_i \subseteq L_i$, y
3. $\forall j \geq 1$, si $T_i \subseteq L_j$, entonces L_j no es un subconjunto propio de L_i

□

Los conjuntos T_i se denominan *indicadores* (*telltales*) y permiten distinguir unos lenguajes de otros dentro de la misma familia. Precisamente, la inexistencia de indicadores es lo que puede provocar el efecto de *generalización excesiva* que consiste en inferir un lenguaje que contiene propiamente al lenguaje objetivo.

Trivialmente, los lenguajes finitos pueden ser identificadas en el límite bajo este protocolo. De igual forma, otros métodos han sido propuestos para la identificación de algunas clases de lenguajes. Entre esas clases podemos mencionar los lenguajes reversibles (Angluin [4], Mäkinen [46]), algunas subclases de lenguajes lineales con reversibilidad (Koshiba *et al.* [40]), algunas clases de lenguages incontextuales mediante gramáticas puras (Koshiba *et al.* [39]), los lenguajes *k-explorables en sentido estricto* (García [20],García *et al.*

[23]), los lenguajes regulares distinguibles por símbolos terminales (Radhakrishnan y Nagaraja [56, 57], Fernau [16]) y su correspondiente clase lineal par (Radhakrishnan y Nagaraja [56, 58], Fernau [16]), extensiones de lenguajes k-explorables y distinguibles por terminales (Fernau [17]) y lenguajes con funciones distinguibles (Fernau [18]), algunas familias de lenguajes caracterizadas por explorabilidad local (Ruiz [61]) y, por último, algunas subclases de lenguajes regulares, incontextuales y sensibles al contexto (Emerald, Subramanian y Thomas [13, 14, 15]).

2.4 Aprendizaje mediante presentación completa

El aprendizaje a partir de información completa consiste en, dado un lenguaje $\mathcal{L}$, proporcionar al método ejemplos y contraejemplos del lenguaje, es decir, cadenas $\{x_1, \cdots, x_n, \cdots\}$ donde cada una de ellas pertenece a $\mathcal{L}$ y cadenas $\{y_1, \cdots, y_n, \cdots\}$ donde cada una de ellas no pertenece a $\mathcal{L}$. Es lo que llamamos *informante* al hablar de inferencia inductiva en el apartado 2.1.

Han sido numerosos los algoritmos de inferencia propuestos bajo este protocolo de información. Principalmente, han sido algunas subclases de lenguajes incontextuales las que han sido caracterizadas bajo este protocolo de información. Así, Oncina y García [51] han propuesto un método que identifica en el límite cualquier lenguaje regular a partir de muestra completa. El problema de la identificación de los lenguajes lineales pares ha podido ser reducido al aprendizaje de lenguajes regulares [70, 71]. Más aún, utilizando la misma técnica de reducción que ha aplicado sobre los lenguajes lineales pares, Takada [72] ha establecido una jerarquía de familias de lenguajes identificables por reducción a los lenguajes regulares.

2.5 Aprendizaje estructural

El aprendizaje estructural implica proporcionar al método de inferencia mayor información que la sola pertenencia o no de una cadena a un lenguaje. Así, si tomamos como espacio de hipótesis una clase de gramáticas $\mathcal{G}$ y tomamos como lenguaje objetivo $L = L(G)$ donde $G \in \mathcal{G}$ entonces al método se le proporcionan una serie de descripciones derivativas de cadenas del lenguaje de acuerdo con la gramática G. Estas descripciones toman habitualmente la forma de *esqueletos* o cadenas parentizadas y se definen como las estructuras derivativas que dieron origen a las cadenas en cuestión omitiendo los símbolos auxiliares de las mismas. Entre los resultados que podemos destacar dentro

de esta aproximación podemos citar los aportados por Sakakibara [62, 63], donde se plantea el aprendizaje de gramáticas incontextuales a partir de los esqueletos asociados a los árboles de derivación. Mäkinen [44] plantea un punto de vista distinto al de Sakakibara al utilizar muestra estructural y *lenguajes de Szilard*. Es más, Mäkinen, en otro trabajo distinto [43], propone una clase de gramáticas, *gramáticas invertibles en tipos*, que es suceptible de ser identificada a partir de muestra estructural.

Por su parte, García [21, 22] ha propuesto diversos métodos que trabajan sobre autómatas de árboles y que, por lo tanto, son suceptibles de ser aplicados a partir de información estructural al aprendizaje de lenguajes formales, típicamente lenguajes incontextuales y subclases de los mismos.

Los trabajos anteriormente citados de Radhakrishnan y Nagaraja [56, 57, 58], aún tomando como fuente de información cadenas, representan los datos mediante esqueletos. Sempere y Fos [68] han propuesto un método heurístico que, basándose en los métodos anteriores, puede ser aplicado a los lenguajes lineales.

Finalmente, cabría plantearse si la utilización que han llevado a cabo Takada y Mäkinen de dos conceptos similares como son los *lenguajes de Szilard* y los *conjuntos de control*, puede ser considerada información estructural. Ambos conceptos hacen referencia al lenguaje derivativo de una gramática. Así, el lenguaje de Szilard de una gramática es el conjunto de cadenas formadas por las secuencias, no de símbolos, sino de producciones que dan lugar, en una secuencia de derivación dentro de la gramática, a las cadenas del lenguaje de la misma. El concepto de conjunto de control es similar con la salvedad de que se trata de un subconjunto de cadenas derivativas dentro de todas las posibles (generadas a su vez por una gramática universal). Es obvio que al hacer referencia a secuencias derivativas se hace referencia a una información estructural, pero, en la mayor parte de sus trabajos [40, 42, 43, 45, 70, 71, 72], esa estructura puede deducirse al tomar únicamente cadenas de un lenguaje. Por ejemplo, en los lenguajes regulares y los lineales pares, adoptando las correspondientes formas normales, las estructuras serán siempre las mismas.

2.6 Ubicación de la tesis en el problema del aprendizaje

Una vez realizada una introducción general al aprendizaje automático, pasamos a situar el trabajo que se presenta en esta tesis en relación con el mismo.

La tesis tiene como principal objetivo presentar distintas clases de lengua-

jes formales que coinciden, en algunos casos, con subclases de las presentadas en la jerarquía de Chomsky. En otros casos, las familias presentadas son incomparables con las referidas en la citada jerarquía. Las clases de lenguajes que se presentan se caracterizan gramaticalmente, es decir, los lenguajes presentan ciertas propiedades en referencia al tipo de gramáticas que los generan.

Una vez caracterizadas esas clases, se pasa al estudio de la inferibilidad de los lenguajes dentro del paradigma de *identificación en el límite*. Los métodos que se presentan son *caracterizables* ya que aseguran la identificación de cualquier lenguaje de la clase bajo estudio. Por lo tanto, nos encontramos ante un trabajo en *inferencia gramatical*.

Por último, el protocolo de información que mayoritariamente se utiliza en la presente tesis es el de información estructural. Así, los datos de entrada que se le suministrarán a los algoritmos de inferencia consistirán en esqueletos de las estructuras derivativas que den lugar a cadenas del lenguaje.

Fundamentalmente, en la presente tesis se trabajará sobre clases de lenguajes lineales pares ($\mathcal{ELL}$) y lineales ($\mathcal{LIN}$). En ambos casos se presentarán distintos rasgos diferenciadores que permiten caracterizar nuevas clases de lenguajes. Nos centraremos en rasgos como la *explorabilidad local*, la *reversibilidad* y la *distinguibilidad terminal y estructural*.

Capítulo 3

Lenguajes, gramáticas y autómatas

Procedemos a definir en este capítulo los conceptos formales que utilizaremos en el resto de la memoria sobre la Teoría de Autómatas, Lenguajes y Gramáticas de forma general. Posteriormente, se introducirán en cada capítulo aquellos conceptos y definiciones específicas según sean necesarios.

Los conceptos que aquí se establecen pueden consultarse en cualquier libro básico sobre la teoría de autómatas, gramáticas y lenguajes formales. En nuestro caso hemos seguido fundamentalmente [8, 29, 33, 59].

3.1 Alfabetos y lenguajes

El primer concepto que aparece de forma natural en la teoría de lenguajes formales es el concepto de *alfabeto* que se define como cualquier conjunto no vacío a cuyos elementos se les denomina *símbolos*. A partir de un alfabeto se puede definir *cadena* o *palabra* como cualquier secuencia finita y ordenada de símbolos del alfabeto. En lo sucesivo utilizaremos los símbolos $\Sigma, \Gamma, \Delta, \cdots$ para denotar a los alfabetos y los símbolos $x, y, z, u, \cdots$ para denotar las cadenas. De entre las infinitas cadenas que se pueden definir sobre un alfabeto destacamos la cadena *vacía*, que denotaremos mediante λ que es aquella que no contiene ningún símbolo. La *longitud* de una cadena se define como el número de símbolos que la forman. Dada una cadena x, denotaremos por $|x|$ su longitud. Es evidente que $|\lambda| = 0$ y que $|xa| = |x| + 1$ siendo x una cadena y a el símbolo que la sucede.

Dado el alfabeto Σ, denotaremos por Σ^k al conjunto de todas las cadenas de

longitud k formadas por símbolos de Σ. $\Sigma^{\geq k}$ denotará el conjunto de cadenas de longitud mayor o igual a k y $\Sigma^{\leq k}$ el conjunto de cadenas con longitud menor o igual a k. Denotaremos por Σ^* al conjunto (infinito) de todas las cadenas que se pueden formar a partir de los símbolos de Σ. A Σ^* se le denomina tambien el *lenguaje universal* o la *clausura* de Σ. De igual forma, Σ^+ es el conjunto de todas las posibles cadenas formadas por símbolos de Σ con la excepción de la cadena vacía, es decir $\Sigma^+ = \Sigma^* - \{\lambda\}$. Un *lenguaje* sobre el alfabeto Σ es cualquier conjunto (finito o infinito) de cadenas sobre Σ. Es decir, el lenguaje L se define siempre como $L \subseteq \Sigma^*$. Consideración especial tiene el lenguaje vacío que no contiene ninguna cadena y que denotaremos por $\emptyset$.

Dado un conjunto A, denotaremos por $card(A)$ su cardinalidad, es decir, el número de elementos que A contiene. De igual forma, denotaremos por $\mathcal{P}(A)$ al conjunto potencia de A, es decir el conjunto de todos los subconjuntos posibles de A.

A partir de un lenguaje L y una cadena x sobre Σ, se puede definir el conjunto de los *buenos finales* de x en L como el conjunto $x^{-1}L = \{u \in \Sigma^* : xu \in L\}$. Tambien se le denomina el cociente por la derecha de L respecto de x. De igual forma, el conjunto de los *buenos comienzos* de x en L es el conjunto $Lx^{-1} = \{u \in \Sigma^* : ux \in L\}$. Se le denomina también el cociente por la izquierda de L respecto de x.

El conjunto de *prefijos* de una cadena x lo denotaremos por $pref(x)$, el conjunto de sufijos por $suf(x)$ y el conjunto de segmentos por $seg(x)$. Si además queremos especificar una longitud determinada entonces los denotaremos respectivamente por $pref(x,k)$, $suf(x,k)$ y $seg(x,k)$, siendo k la longitud deseada. Evidentemente, en el caso anterior sólo estarán definidos si $|x| \geq k$. Por ejemplo, si $x = abbaba$ entonces $seg(x,2) = \{ab, bb, ba\}$. La extensión de los anteriores conjuntos para su actuación sobre lenguajes se realiza de forma trivial sobre el conjunto de cadenas que componen el lenguaje en cuestión. Así, dado el lenguaje L, $pref(L,k) = \bigcup_{x \in L} pref(x,k)$, $suf(L,k) = \bigcup_{x \in L} suf(x,k)$ y $seg(L,k) = \bigcup_{x \in L} seg(x,k)$ El *reverso* (o *inverso*) de la cadena x se denotará por x^{inv} y se define como la secuencia de símbolos inversa a la secuencia x. Dadas dos cadenas x e y, su concatenación o producto se define como la secuencia obtenida a partir de la secuencia de x seguida de la de y, lo denotaremos por xy o $x \cdot y$. En el caso de la cadena vacía se cumple que $x\lambda = \lambda x = x$. La potencia n-ésima de una cadena x la denotaremos por x^n y es la secuencia obtenida al concatenar n veces la cadena x con ella misma. Se define $x^0 = \lambda$.

Veamos ahora un concepto que tendrá especial interés en la presente memoria como es el de los símbolos terminales de una cadena. Lo definiremos formalmente como sigue.

Definición 3.1. Definiremos, de forma inductiva, el conjunto de símbolos terminales de la cadena x, denotándolo por $ter(x)$, como sigue

1. $ter(\lambda) = \emptyset$
2. $(\forall y \in \Sigma^*)\ \ (\forall a \in \Sigma)\ \ ter(ya) = ter(y) \cup \{a\}$

□

Por ejemplo, si $x = abba$ entonces $ter(x) = \{a, b\}$. Por otra parte, en relación con los segmentos de una cadena x, se cumple que $seg(x, 1) = ter(x)$.

Se cumple la siguiente propiedad en relación con los segmentos de una cadena

Propiedad 3.1. Sea Σ un alfabeto. Para cualquier valor $k > 0$ y todo par de cadenas $x, y \in \Sigma^{\geq k+1}$ se cumple:

1. Si $seg(x, k) = seg(y, k)$ entonces $seg(x, k-1) = seg(y, k-1)$
2. Si $seg(x, k) \neq seg(y, k)$ entonces $seg(x, k+1) \neq seg(y, k+1)$

Demostración.

Demostraremos cada una de las afirmaciones por separado.

1. Tomemos x con $|x| = m \geq k+1$ e y con $|y| = n \geq k+1$. Tomemos $x = x_1 \cdots x_m$ con $seg(x, k) = \{x_1 \cdots x_k, \cdots, x_{m-k+1} \cdots x_m\}$. De igual forma, $y = y_1 \cdots y_n$ con $seg(y, k) = \{y_1 \cdots y_k, \cdots, y_{n-k+1} \cdots y_n\}$. Partimos de que $seg(x, k) = seg(y, k)$. Tomando ahora cualquier elemento de $seg(x, k-1)$, por ejemplo, $x_{i1} \cdots x_{ik-1}$ podemos observar que procede de un elemento de $seg(x, k)$ que, a su vez, está presente en $seg(y, k)$ y, por lo tanto, el elemento $x_{i1} \cdots x_{ik-1}$ también se encuentra en $seg(y, k-1)$. El razonamiento inverso, partiendo de elementos de $seg(y, k-1)$, también es cierto y, por lo tanto, $seg(x, k-1) = seg(y, k-1)$.

2. Al igual que en el enunciado anterior, tomemos $x = x_1 \cdots x_m$ con $seg(x, k) = \{x_1 \cdots x_k, \cdots, x_{m-k+1} \cdots x_m\}$ e $y = y_1 \cdots y_n$ con $seg(y, k) = \{y_1 \cdots y_k, \cdots, y_{n-k+1} \cdots y_n\}$. En este caso, $seg(x, k) \neq seg(y, k)$. Es obvio que debe existir un elemento en $seg(x, k+1)$, por ejemplo $x_{i1} \cdots x_{ik+1}$ cuyo sufijo o prefijo de longitud k no pertenece a $seg(y, k)$ y, por lo tanto, el citado elemento no pertenece a $seg(y, k+1)$. Así, concluimos que $seg(x, k+1) \neq seg(y, k+1)$.

□

La extensión del conjunto de símbolos terminales sobre lenguajes se realiza de la forma usual. Así, $\forall L \subseteq \Sigma^*\ ter(L) = \bigcup_{x \in L} ter(x)$. El reverso de un lenguaje L es el conjunto obtenido por los reversos de sus cadenas y lo denotaremos por L^{inv}. Dados dos lenguajes L_1 y L_2, su concatenación o producto lo denotaremos por L_1L_2 y se define como el resultado de concatenar las cadenas de L_1 con las de L_2. Dado un lenguaje L, su potencia n-ésima la denotaremos por L^n y es el resultado de concatenar n veces el lenguaje L consigo mismo. Por definición, $L^0 = \{\lambda\}$. A partir de la potencia podemos definir la clausura del lenguaje L, denotándola por L^*, como la unión de todas sus posibles potencias.

3.2 Relaciones

Dado un conjunto A, una relación n-aria definida sobre el conjunto A es un subconjunto del producto $A \times A \times \cdots \times A$ (n veces). Las relaciones que más utilizaremos a lo largo de la presente tesis son las relaciones *binarias*, es decir, definidas sobre $A \times A$. Diremos que la relación binaria R definida sobre A es una *relación binaria de equivalencia* si es *reflexiva*, *simétrica* y *transitiva*.

Dada la relación binaria de equivalencia R definida sobre A, la *clase de equivalencia* del elemento $x \in A$ bajo R es el conjunto formado por todos aquellos elementos que se relacionan con x de acuerdo con R. La clase de equivalencia de x bajo R la denotaremos por $[x]_R$. Es obvio que, al ser la relación R una relación reflexiva, no existe ninguna clase de equivalencia que sea vacía. Al conjunto formado por las clases de equivalencia lo denominaremos *conjunto cociente* y lo denotaremos como A/R.

Dadas dos relaciones $\sim$ y $\approx$ definidas sobre el conjunto A, diremos que $\sim$ *refina* a $\approx$, o que $\approx$ *recubre* a $\sim$, si se cumple el siguiente enunciado

$$(\forall x, y \in A)\ x \sim y \Rightarrow x \approx y$$

Dado el conjunto A y la ley de composición interna μ definida sobre A diremos que la relación binaria de equivalencia $\sim$ definida sobre A es una *congruencia por la derecha* (*por la izquierda*) con respecto a μ si se cumple el siguiente enunciado:

$$(\forall x, y, z \in A)\ x \sim y \Rightarrow x\mu z \sim y\mu z\ (z\mu x \sim z\mu y)$$

A toda relación binaria de equivalencia que es congruencia por la derecha y congruencia por la izquierda se le denomina simplemente *congruencia*.

Una *partición* sobre un conjunto A (distinto del vacío) es una colección de subconjuntos $\{A_i : i \geq 1\}$ que cumplen las tres siguientes condiciones:

1. $\forall i \ A_i \neq \emptyset$
2. $\bigcup_i A_i = A$
3. $\forall i \neq j \ A_i \cap A_j = \emptyset$

A cada uno de los subconjuntos A_i le denominaremos *bloque* de la partición. Al número de bloques que definen una partición le llamaremos *índice*. Una partición diremos que es de *índice finito* si el número de bloques que la definen es finito, en caso contrario es de índice infinito.

Se cumple la siguiente propiedad en lo relativo a las particiones y las relaciones.

Propiedad 3.2. Sea A un conjunto y $\sim$ una relación binaria de equivalencia sobre A. La relación $\sim$ induce un partición sobre el conjunto A de forma que cada clase de equivalencia es un bloque de la partición. □

El enunciado inverso de la propiedad anterior también se cumple. Es decir, podemos establecer una similitud entre relaciones y particiones como conceptos matemáticos equivalentes.

Dada una relación R, la clausura de R con respecto a la propiedad P (reflexividad, transitividad, etc.), es la relación más pequeña que contiene a R y que cumple la propiedad P. A la clausura de una relación se le denomina también cierre.

3.3 Gramáticas

Una gramática es un sistema de reescritura que se define como una tupla de cuatro elementos (Σ, N, P, S), donde Σ y N son alfabetos disjuntos de símbolos terminales y auxiliares respectivamente, $S \in N$ es el *axioma* o símbolo inicial y P es un conjunto finito de producciones de la gramática expresado por los pares (α, β) donde α (el *antecedente*) es una cadena formada por símbolos auxiliares y terminales con al menos un símbolo auxiliar y β (el *consecuente*) es una cadena formada por símbolos auxiliares y terminales que puede ser la cadena vacía. Una producción (α, β) también se puede denotar por $\alpha \to \beta$. Para aquellas producciones que compartan un mismo antecedente se puede establecer la notación siguiente, en vez de escribir $\alpha \to \beta_1$, $\alpha \to \beta_2$, ..., $\alpha \to \beta_n$, escribiremos $\alpha \to \beta_1 \mid \beta_2 \mid \cdots \mid \beta_n$.

Se puede establecer la *relación de derivación directa* entre cadenas formadas por símbolos de $\Sigma \cup N$, de acuerdo con la gramática G, de la siguiente

forma $\alpha\beta\gamma \underset{G}{\Rightarrow} \alpha\beta'\gamma$ si $\beta \in \{\Sigma \cup N\}^* N \{\Sigma \cup N\}^*$, $\alpha, \beta', \gamma \in \{\Sigma \cup N\}^*$ y $\beta \to \beta' \in P$.

La clausura reflexiva y transitiva de la relación $\underset{G}{\Rightarrow}$, la denotaremos por $\overset{*}{\underset{G}{\Rightarrow}}$ y se define como la *relación de derivación* entre cadenas de la gramática. A partir de esta relación se puede definir el lenguaje generado por una gramática $G = (\Sigma, N, P, S)$ como el conjunto $L(G) = \{w \in \Sigma^* : S \overset{*}{\underset{G}{\Rightarrow}} w\}$. De igual forma, para cualquier símbolo auxiliar A de la gramática G se puede definir el conjunto $L(A, G) = \{w \in \Sigma^* : A \overset{*}{\underset{G}{\Rightarrow}} w\}$. Es obvio que $L(S, G) = L(G)$. En algunas ocasiones, cuando se sobreentienda la gramática G, escribiremos indistintamente $L(A, G)$ o $L(A)$. Distinguiremos entre *cadenas* y *formas sentenciales* de la gramática G, de forma que las cadenas se forman únicamente a partir de símbolos de Σ mientras que en las formas sentenciales pueden aparecer tanto símbolos de Σ como de N.

De acuerdo con las formas que pueden adquirir las producciones de las gramáticas, Chomsky propuso una clasificación de las mismas en cuatro grandes familias en lo que es conocido como la *jerarquía de Chomsky*. De mayor a menor restricción en las forma de las producciones, Chomsky propuso las gramáticas regulares (lineales por la derecha o por la izquierda), gramáticas libres de contexto, gramáticas sensibles al contexto y gramáticas no restringidas. Definiremos en primer lugar cada una de los cuatro tipos de gramáticas de acuerdo con la clasificación de Chomsky.

Definición 3.2. Sea $G = (\Sigma, N, P, S)$ una gramática. Diremos que G es lineal por la derecha (por la izquierda) si cada una de las producciones de P toma una de las dos siguientes formas

1. $A \to wB$ $(A \to Bw)$ con $A, B \in N$ y $w \in \Sigma^*$
2. $A \to w$ con $A \in N$ y $w \in \Sigma^*$

A estas gramáticas se les conoce también como gramáticas regulares. □

Definición 3.3. Sea $G = (\Sigma, N, P, S)$ una gramática. Diremos que G es libre de contexto o incontextual si cada una de las producciones de P toma la forma $A \to \alpha$ con $A \in N$ y $\alpha \in (N \cup \Sigma)^*$. □

Definición 3.4. Sea $G = (\Sigma, N, P, S)$ una gramática. Diremos que G es sensible al contexto si cada una de las producciones de P toma la siguiente forma $\alpha A \beta \to \alpha\gamma\beta$ con $\alpha, \beta \in (N \cup \Sigma)^*$, $A \in N$ y $\gamma \in (N \cup \Sigma)^+$. Además se añade la excepción de que el axioma S puede producir λ siempre que S no aparezca en el consecuente de ninguna producción. □

Definición 3.5. Sea G una gramática sin ninguna condición previa sobre sus producciones, entonces G es una gramática no restringida. □

Estos cuatro tipos de gramáticas dan origen a clases de lenguajes formales de acuerdo con el siguiente criterio : un lenguaje L pertenece a una clase C si existe una gramática G perteneciente a C de forma que $L(G) = L$. Si denotamos por $\mathcal{REG}$, $\mathcal{CF}$, $\mathcal{CS}$ y $\mathcal{RE}$ a las familias de lenguajes originadas por las clases de gramáticas enunciadas anteriormente tenemos la siguiente relación entre familias

$$\mathcal{REG} \subset \mathcal{CF} \subset \mathcal{CS} \subset \mathcal{RE}$$

Por supuesto, la clasificación de Chomsky es un marco de estudio general que se ha tomado como referencia para la definición de otras familias de lenguajes que no se ajustan a tal clasificación. A lo largo de la presente tesis se establecerán algunas clases de lenguajes que, en cierta medida, rompen tal clasificación.

Para las gramáticas incontextuales se puede definir un formalismo descriptivo acerca de las derivaciones que se producen en las mismas : los *árboles de derivación*. Así, dada una gramática incontextual $G = (\Sigma, N, P, S)$ se define un árbol de derivación de G de acuerdo con las siguientes reglas

1. Los nodos internos del árbol se etiquetan con símbolos de N
2. Los nodos que forman hojas en el árbol se etiquetan con símbolos de $\Sigma \cup \{\lambda\}$
3. La raíz del árbol se etiqueta con el axioma S
4. Si el nodo interno etiquetado con A tiene hijos (de izquierda a derecha) etiquetados con los símbolos $x_1, x_2, \cdots, x_n$ entonces la gramática G contiene la producción $A \to x_1 x_2 \cdots x_n$
5. Si un nodo está etiquetado con λ entonces el nodo es una hoja y su padre tiene un hijo único que es él

Se definirá un A-árbol como un árbol de derivación cuya raíz está etiquetada con el símbolo $A \in N$. Dada una derivación en la gramática G entonces, siguiendo las anteriores reglas, se puede construir un árbol de derivación que se corresponda con la misma. Dado un árbol de derivación, a la cadena obtenida a partir de las etiquetas de las hojas siguiendo un recorrido de izquierda a derecha se le denomina *resultado* del árbol, en otras ocasiones se le puede denominar también *frontera*. Una gramática incontextual G es *ambigua* si se pueden construir dos o más árboles de derivación distintos cuyos resultados sean la misma cadena del lenguaje $L(G)$. Un lenguaje incontextual es

inherentemente ambiguo si cualquier gramática incontextual que lo genera es ambigua. Al resultado de sustituir las etiquetas de los nodos internos de un árbol de derivación por una única etiqueta ajena a la gramática se le denomina *esqueleto*.

Dadas dos gramáticas G_1 y G_2, diremos que son equivalentes si $L(G_1) = L(G_2)$. Además, si G_1 y G_2 son libres de contexto entonces son estructuralmente equivalentes si el conjunto de árboles de derivación de G_1 es idéntico al conjunto de árboles de derivación de G_2 salvo homomorfismos alfabéticos entre los alfabetos de símbolos auxiliares. Es obvio que dos gramáticas libres de contexto estructuralmente equivalentes son también equivalentes, mientras que lo contrario, no se cumple necesariamente.

3.4 Autómatas

Un autómata es un dispositivo abstracto que, dada una cadena x sobre un alfabeto, tras analizar sus símbolos, en algunas ocasiones puede finalizar el proceso emitiendo una salida que puede ser de *aceptación* o *rechazo* (aceptores) u otra salida simbólica en forma de cadena (transductores). Al igual que sucedía con las gramáticas, existen distintos tipos de autómatas que, según su capacidad de procesamiento, pueden caracterizar distintas clases de lenguajes. Procederemos a definir los autómatas más simples que permiten reconocer las clases de lenguajes objeto de estudio en la presente tesis.

Definición 3.6. Un autómata finito (AF) se define como la tupla $(Q, \Sigma, \delta, I, F)$ donde Q es un conjunto finito de estados, Σ es un alfabeto de símbolos de entrada, $I \subseteq Q$ es un conjunto de estados iniciales, $F \subseteq Q$ es un conjunto de estados de aceptación y $\delta : Q \times \Sigma \to \mathcal{P}(Q)$ es una función de transición entre estados. □

La extensión de la función de transición entre estados sobre cadenas de símbolos la denotaremos también como δ y se define de la siguiente forma

1. $\delta(q, \lambda) = \{q\}$
2. Para todo $a \in \Sigma$, $x \in \Sigma^*$ $\delta(q, xa) = \bigcup_{p \in \delta(q,x)} \delta(p, a)$

De esta forma, el lenguaje que acepta el AF A lo denotamos por $L(A)$ y es el conjunto $\{x \in \Sigma^* : \exists q \in I \wedge \delta(q, x) \cap F \neq \emptyset\}$.

Un resultado bien conocido en la teoría de autómatas y lenguajes es que la clase de lenguajes aceptados por los autómatas finitos es la clase de los lenguajes regulares $\mathcal{REG}$.

En los autómatas finitos deterministas la función de transición δ queda definida como $\delta : Q \times \Sigma \to Q$ e $I = \{q_0\}$. Por otra parte, en los autómatas finitos con transiciones vacías la función δ se define como $\delta : Q \times \Sigma \cup \{\lambda\} \to \mathcal{P}(Q)$. En ambos casos se pueden realizar las extensiones relativas a cadenas de manera similar al caso anterior. También es un resultado clásico de la teoría de autómatas que los autómatas deterministas y con transiciones vacías no alteran la capacidad de procesamiento respecto de los autómatas finitos en general. Es decir, en ambos casos, se sigue caracterizando la clase de los lenguajes regulares.

Definición 3.7. Dado el autómata finito $A = (Q, \Sigma, \delta, I, F)$ su autómata reverso lo denotaremos como A^{inv} y se define como la tupla $(Q, \Sigma, \delta^{inv}, F, I)$, donde

$$(\forall a \in \Sigma \cup \{\lambda\}) \quad (\forall q \in Q) \quad \delta^{inv}(q, a) = \{p : q \in \delta(p, a)\}$$

□

La relación entre los lenguajes definidos por A y A^{inv} es $L(A^{inv}) = [L(A)]^{inv}$.

Dado el autómata finito determinista $A = (Q, \Sigma, \delta, q_0, F)$ y una relación binaria de equivalencia entre sus estados $\sim$, podemos definir el autómata cociente $A/\sim$ como el autómata $A_\sim = (Q/\sim, \Sigma, \delta_\sim, [q_0]_\sim, F/\sim)$, donde se define $\delta_\sim([q]_\sim, a) = [\delta(q, a)]_\sim$. Obsérvese que para que $A/\sim$ esté bien definido entonces debe cumplirse la condición de *unicidad* en la función δ, es decir, $\forall a \in \Sigma \; \forall p, q \in Q$ si $p \sim q$ entonces $\delta(p, a) \sim \delta(q, a)$.

Capítulo 4

Aprendizaje de lenguajes lineales pares

La clase de los lenguajes lineales pares fue definida inicialmente por Amar y Putzolu [1] como una subclase de lenguajes incontextuales ($\mathcal{CF}$). En su trabajo, Amar y Putzolu proporcionaron una caracterización de los lenguajes lineales pares similar a la que realizó Nerode para lenguajes regulares [33]. Desde el punto de vista de la caracterización proporcionada, un lenguaje es lineal par si y sólo si es saturado por una *quasi-congruencia* de índice finito. Informalmente, una quasi-congruencia es similar a una congruencia con la salvedad de que, dadas dos palabras, su equivalencia implica la equivalencia de nuevas palabras obtenidas al incluir segmentos de igual longitud al inicio y final de las palabras originales (contextos de igual longitud).

Algunos trabajos se han centrado en el problema del aprendizaje de lenguajes lineales pares. Por ejemplo, en los trabajos realizados por Radhakrishnan y Nagaraja [56, 58] se utiliza texto finito para llevar a cabo un proceso de inferencia. En el mismo trabajo se muestra cómo se utilizan las cadenas de entrada para obtener gramáticas lineales pares a partir de la información estructural inducida por las cadenas en forma de esqueletos. De igual forma, se establecen aquellos subesqueletos que presentan similitud en términos estructurales y de *distinguibilidad terminal*. El algoritmo de inferencia propuesto por Radhakrishnan y Nagaraja ha sido aplicado, entre otras tareas, para la modelización de un Lenguaje de Descripción de Dibujos (*Picture Description Language*, PDL) para el reconocimiento de objetos simétricos simples. A la clase caracterizada por el algoritmo propuesto en los trabajos citados anteriormente se le conoce como la clase $\mathcal{TDELL}$ (Terminal Distinguishable Even Linear Languages).

Posteriormente, Fernau [16] ha generalizado y redefinido el concepto de distinguibilidad terminal relativo a los lenguajes lineales pares y ha caracterizado y propuesto nuevas clases de lenguajes lineales pares identificables en el límite a partir de muestra positiva.

Otro estudio llevado a cabo en lo referente al aprendizaje de los lenguajes lineales pares es el de Takada [70]. En su trabajo, Takada estableció que cada lenguaje lineal par puede ser generado por una gramática universal junto con un *conjunto de control* que regula la aplicación de sus producciones. Takada demostró que el conjunto de control de cada gramática lineal par es un lenguaje regular. Este resultado permite reducir el problema del aprendizaje de los lenguaje lineales pares al de los lenguajes regulares. Bajo este planteamiento, los datos de entrada son analizados a través de la gramática universal y convertidos en cadenas de reglas a partir de las que se identifica el conjunto de control correspondiente utilizando cualquier algoritmo de identificación de lenguajes regulares. Finalmente, se puede obtener una gramática lineal par a partir del conjunto de control inferido con la propiedad de que genera el mismo lenguaje que se obtiene a partir de la gramática universal y el conjunto de control. Siguiendo en la misma línea, Takada en un trabajo posterior [72] ha aplicado la misma técnica para definir una jerarquía de lenguajes que transcienden a los lenguajes incontextuales y que pueden ser inferidos utilizando la misma técnica que en el primer trabajo citado anteriormente.

En el presente capítulo propondremos una nueva caracterización de los lenguajes lineales pares que permitirá definir una gramática canónica asociada a cada lenguaje lineal par. Más aún, la gramática proporcionada será la mínima de entre todas aquellas que se formulen en una forma normal predeterminada y que generan el lenguaje lineal par bajo estudio. La unicidad de la citada gramática será cierta salvo gramáticas isomorfas en cuanto a los símbolos auxiliares.

De esta forma, el problema del aprendizaje de lenguajes lineales pares se puede abordar, como en el trabajo de Takada, mediante una reducción al aprendizaje de los lenguajes regulares. En este caso, los datos de entrada sufren una transformación inicial y son utilizados como fuente de información para cualquier algoritmo de inferencia de lenguajes regulares. Finalmente, a partir de la hipótesis regular obtenida, se puede aplicar una transformación inversa que proporciona una gramática lineal par consistente con la información de entrada inicial.

De entre aquellos trabajos que se relacionan con el aprendizaje de lenguajes lineales pares cabe citar el de Koshiba, Mäkinen y Takada [40] que han propuesto una subclase de lenguajes lineales pares identificable únicamente

a partir de datos positivos que denominan lenguajes $\mathcal{LRS}(k)$. En este caso, siguiendo ideas similares a las expuestas anteriormente en el trabajo de Takada [70], se puede utilizar una gramática universal y un conjunto de control k-reversible identificable en el límite a partir de datos positivos. Los lenguajes lineales pares $\mathcal{LRS}(k)$ forman una subclase de los lenguajes lineales pares $\mathcal{LR}(k)$ que, tal y como se muestra en el mismo trabajo, no son identificables a partir de texto.

En relación con estas clases de lenguajes, definiremos otro rasgo característico de las gramáticas lineales pares que es el *fuerte determinismo hacia atrás* que dará lugar a las gramáticas lineales pares $\mathcal{SBD}$ que demostraremos que se corresponden con las gramáticas lineales pares $\mathcal{LRS}(0)$ y, por lo tanto, inferibles a partir de datos positivos. Esta última relación entre clases de gramáticas resultará especialmente interesante cuando, en el capítulo siguiente, se analicen las mismas clases de gramáticas en el caso lineal.

4.1 Conceptos básicos acerca de los lenguajes lineales pares

En primer lugar, vamos a definir aquellos conceptos de la teoría de autómatas, gramáticas y lenguajes formales no especificados en el capítulo 3 y que hacen referencia a los lenguajes y gramáticas lineales pares.

Definición 4.1. Una gramática lineal par es una gramática incontextual $G = (N, \Sigma, P, S)$ donde todas las reglas de P toman una de las siguientes formas

1. $A \to xBy$, donde $x, y \in \Sigma^*$, $A, B \in N$ y $|x| = |y|$.
2. $A \to x$, donde $x \in \Sigma^*$, $A \in N$.

□

Un lenguaje L es lineal par si existe una gramática lineal par que lo genera. A la clase de los lenguajes lineales pares la denotaremos por $\mathcal{ELL}$. Se cumple la siguiente relación con respecto a la jerarquía de Chomsky

$$\mathcal{REG} \subset \mathcal{ELL} \subset \mathcal{CF}$$

Dada cualquier gramática lineal par, siempre existe otra equivalente donde cada una de sus producciones puede tomar una de las siguientes formas

1. $A \to aBb$, donde $a, b \in \Sigma$ y $A, B \in N$.

2. $A \to a$, donde $a \in \Sigma \cup \{\lambda\}$ y $A \in N$.

Las anteriores formas en las producciones definen una forma normal en las gramáticas lineales pares. A partir de ahora trabajaremos siempre con gramáticas en forma normal mientras no se especifique lo contrario. Podemos ahora definir el determinismo en las gramáticas lineales pares como sigue

Definición 4.2. Sea $G = (N, \Sigma, P, S)$ una gramática lineal par en forma normal. Diremos que G es determinista si $\forall\ a, b \in \Sigma$; $\forall\ A, B, C \in N$ si $A \to aBb \in P$ y $A \to aCb \in P$ entonces $B = C$.

□

A partir de los sufijos y prefijos de una cadena se pueden establecer las subclases de lenguajes lineales pares $\mathcal{LR}(k)$ y $\mathcal{LRS}(k)$.

Definición 4.3. Sea $G = (N, \Sigma, P, S)$ una gramática lineal par y el valor entero $k \geq 0$. G es $\mathcal{LR}(k)$ si y sólo si $\forall\ u, v, w \in \Sigma^*$, $\forall x \in (\Sigma \cup N)^*$ y $\forall A, B \in N$, si se cumplen las tres siguientes condiciones

1. $S \underset{G}{\overset{*}{\Rightarrow}} uAv \underset{G}{\Rightarrow} uxv$
2. $S \underset{G}{\overset{*}{\Rightarrow}} uBw \underset{G}{\Rightarrow} uxw$
3. $pref(v, k) = pref(w, k)$

entonces $A = B$.

□

En la misma línea, podemos restringir más las condiciones de la definición anterior y definir la siguiente clase de gramáticas

Definición 4.4. Sea $G = (N, \Sigma, P, S)$ una gramática lineal par y el valor entero $k \geq 0$. G es $\mathcal{LRS}(k)$ ($\mathcal{LR}(k)$ en *sentido fuerte*) si y sólo si $\forall$ $u_1, u_2, v_1, v_2 \in \Sigma^*$, $\forall x \in (\Sigma \cup N)^*$ y $\forall A, B \in N$, si se cumplen las cuatro siguientes condiciones

1. $S \underset{G}{\overset{*}{\Rightarrow}} u_1 A v_1 \underset{G}{\Rightarrow} u_1 x v_1$
2. $S \underset{G}{\overset{*}{\Rightarrow}} u_2 B v_2 \underset{G}{\Rightarrow} u_2 x v_2$
3. $pref(v_1, k) = pref(v_2, k)$
4. $suf(u_1, k) = suf(u_2, k)$

entonces $A = B$.

□

Se puede comprobar que las anteriores definiciones, aún siendo distintas de las proporcionadas por Koshiba, Mäkinen y Takada [40], son equivalentes a ellas. En este caso hemos redefinido las clases de gramáticas $\mathcal{LR}(k)$ y $\mathcal{LRS}(k)$ con el objetivo de ser más explícitos en cuanto a la equivalencia con la clase de gramáticas que definiremos a continuación.

Definición 4.5. Sea $G = (N, \Sigma, P, S)$ una gramática lineal par. G es *fuertemente determinista hacia atrás*, de forma abreviada $\mathcal{SBD}$ (*strongly backward deterministic*) sii $\forall w \in \Sigma^*$ y $\forall A, B \in N$, si se cumplen las dos siguientes condiciones

1. $A \overset{*}{\underset{G}{\Rightarrow}} w$, y
2. $B \overset{*}{\underset{G}{\Rightarrow}} w$,

entonces $A = B$.

□

Obsérvese que en la mayoría de los resultados es una condición indispensable que las gramáticas de partida sean *reducidas*, es decir, que no hayan símbolos ni producciones inútiles o, lo que es lo mismo, que cada símbolo aparezca en alguna secuencia de derivación que parta del axioma y que todas las producciones contribuyan a la generación de palabras del lenguaje. Así, a partir de ahora, trabajaremos exclusivamente con gramáticas reducidas. De esta forma, podemos aportar los siguiente resultados que relacionan las tres clases de gramáticas que se han definido anteriormente.

Lema 4.1. (Koshiba *et al.* [40]) Para cada valor $k \geq 0$ se cumple que si G es una gramática lineal par $\mathcal{LRS}(k)$ entonces G es también $\mathcal{LR}(k)$. □

Como aportación al estudio de las gramáticas definidas anteriormente podemos dar el siguiente resultado

Lema 4.2. Sea $G = (N, \Sigma, P, S)$ una gramática lineal par reducida. G es $\mathcal{SBD}$ sii G es $\mathcal{LRS}(0)$.

Demostración.

Demostraremos las condiciones suficientes y necesarias expuestas en el lema para la equivalencia entre las dos clases de gramáticas

En primer lugar tomemos $G = (N, \Sigma, P, S)$ como una gramática lineal par $\mathcal{SBD}$. Se cumplirá que, para cada cadena $w \in \Sigma^*$ si $A \overset{*}{\underset{G}{\Rightarrow}} w$ y $B \overset{*}{\underset{G}{\Rightarrow}} w$ entonces $A = B$. A partir de la última condición podemos observar que si $S \overset{*}{\underset{G}{\Rightarrow}} xAy \underset{G}{\Rightarrow} xzy$ y $S \overset{*}{\underset{G}{\Rightarrow}} uBv \underset{G}{\Rightarrow} uzv$ entonces $A \to z$ y $B \to z$ son producciones de la gramática y, por lo tanto, $A \underset{G}{\Rightarrow} z \overset{*}{\underset{G}{\Rightarrow}} w$ y $B \underset{G}{\Rightarrow} z \overset{*}{\underset{G}{\Rightarrow}} w$ donde $w \in \Sigma^*$ ya que suponemos que G es una gramática reducida y, por lo tanto, no hay símbolos no generativos. Dado que G es $\mathcal{SBD}$ entonces $A = B$ y, por lo tanto, G es $\mathcal{LRS}(0)$. Obsérvese que, en este caso, las condiciones 3 y 4 de la definición 4.4 se cumplen trivialmente ya que $pref(y, 0) = pref(v, 0) = \lambda$ y $suf(x, 0) = suf(u, 0) = \lambda$.

Supongamos ahora que G es una gramática lineal par $\mathcal{LRS}(0)$ reducida. Se cumple, de acuerdo con la definición 4.4 que si $A \underset{G}{\Rightarrow} x$, y $B \underset{G}{\Rightarrow} x$, entonces $A = B$. Obsérvese que, en este caso, al ser G reducida los símbolos A y B son alcanzables desde S y, al ser G $\mathcal{LRS}(0)$ los prefijos y sufijos de longitud cero de cualquier forma sentencial donde aparezcan A o B son irrelevantes al coincidir trivialmente con λ. Supongamos ahora que, en la misma gramática, se cumple que $A \overset{*}{\underset{G}{\Rightarrow}} w$, y $B \overset{*}{\underset{G}{\Rightarrow}} w$ con $w \in \Sigma^*$. Obviamente, las derivaciones anteriores las podemos reescribir como sigue

$$A \underset{G}{\Rightarrow} w_1 A_1 w_n \underset{G}{\Rightarrow} \cdots \underset{G}{\Rightarrow} w_1 \cdots A_i \cdots w_n \underset{G}{\Rightarrow} w_1 \cdots w_{i+1} \cdots w_n = w$$

$$B \underset{G}{\Rightarrow} w_1 B_1 w_n \underset{G}{\Rightarrow} \cdots \underset{G}{\Rightarrow} w_1 \cdots B_i \cdots w_n \underset{G}{\Rightarrow} w_1 \cdots w_{i+1} \cdots w_n = w$$

A partir de las condiciones reformuladas anteriormente de la definición 4.4 es fácil demostrar que, en las anteriores derivaciones $A_i = B_i$ para todo valor i y, en consecuencia, $A = B$. Por lo tanto, G es $\mathcal{SBD}$ tal y como se pretendía demostrar.

□

4.2 Una caracterización de los lenguajes lineales pares.

En primer lugar, como trabajo previo a la resolución del aprendizaje de los lenguajes lineales pares, vamos a proporcionar una caracterización alternativa de los mismos. A continuación estableceremos que, dada cualquier gramática lineal par (en forma normal), podemos encontrar una gramática lineal par determinista equivalente utilizando una transformación sobre las cadenas del lenguaje que genera.

Definición 4.6. Sea Σ un alfabeto y la cadena $x = a_1 \cdots a_{k-1} a_k a_{k+1} \cdots a_n$, donde $\forall i,\ 1 \leq i \leq n,\ i \neq k,\ a_i \in \Sigma$ y $a_k \in \Sigma \cup \{\lambda\}$. Definimos los *extremos agrupados* de x, denotándolo por $\sigma(x)$, a la cadena $[a_1 a_n]...[a_{k-1} a_{k+1}] a_k$. Podemos definir los extremos agrupados de una cadena de forma inductiva como sigue. $\sigma : \Sigma^* \to (\Sigma^2 \cup \Sigma)^*$

1. $(\forall a \in \Sigma \cup \{\lambda\})\ \sigma(a) = a$
2. $(\forall a, b \in \Sigma)\ (\forall x \in \Sigma^*)\ \sigma(axb) = [ab]\sigma(x)$

□

Obsérvese que σ produce un cambio en el alfabeto de las cadenas. Así, Σ^2 se puede considerar como un nuevo alfabeto donde cada símbolo queda representado por los pares de símbolos del alfabeto original.

Podemos extender la definición a lenguajes y proporcionar los extremos agrupados de un lenguaje L como $\sigma(L) = \{\sigma(x) : x \in L\}$.

Además, podemos definir la transformación inversa de σ como sigue

1. $(\forall a \in \Sigma \cup \{\lambda\})\ \sigma^{-1}(a) = a$
2. $(\forall a, b \in \Sigma)\ (\forall x \in \Sigma^*)\ \sigma^{-1}([ab]x) = a\sigma^{-1}(x)b$

En este caso, $\sigma^{-1}(L) = \{\sigma^{-1}(x) : x \in L\}$ y $\sigma^{-1}(\sigma(x)) = x$, por lo que $\sigma^{-1}(\sigma(L)) = L$.

A partir de la anterior definición se puede establecer que la transformación σ obtiene un lenguaje regular a partir de un lenguaje lineal par y que, a partir de este hecho, se puede definir una relación de índice finito que caracteriza a los lenguaje lineales pares.

Teorema 4.1. Si $L \subseteq \Sigma^*$ es un lenguaje lineal par, entonces $\sigma(L)$ es un lenguaje regular.

Demostración.

Sea el lenguaje $L = L(G)$, donde $G = (N, \Sigma, P, S)$ es una gramática lineal par en forma normal. Podemos definir el autómata finito $A = (Q, \Sigma', \delta, q_0, F)$, donde $Q = N \cup \{q_f\}$, $q_f \notin N$, $\Sigma' = \Sigma^2 \cup \Sigma$, $q_0 = S$, $F = \{q_f\}$ y δ se define mediante las siguientes reglas

1. Si $A \to aBb \in P$, entonces $B \in \delta(A, [ab])$.
2. Si $A \to a \in P$, entonces $\delta(A, a) = \{q_f\}$.

Por último, mediante un proceso inductivo se puede demostrar que

$$(\forall A \in N)\ \ (\forall x \in \Sigma^*)\ \ (A \overset{*}{\underset{G}{\Rightarrow}} x \ \text{ sii } \ \delta(A, \sigma(x)) \cap F \neq \oslash).$$

□

Obsérvese que, si se toma un autómata finito como el que se construye en el anterior teorema, se puede obtener una gramática lineal par equivalente mediante el proceso inverso resultando, en este caso, la demostración de la equivalencia trivial. Así, dado un autómata A, la gramática lineal par que se obtiene genera el lenguaje $\sigma^{-1}(L(A))$.

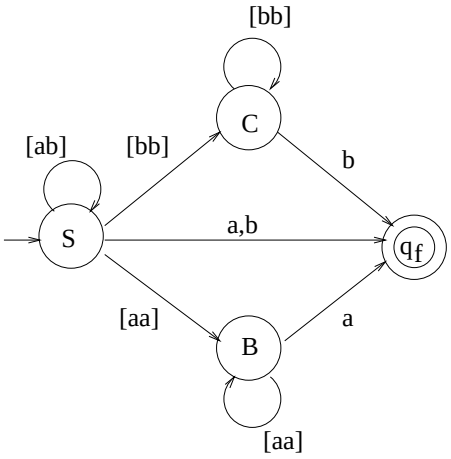
Ejemplo 4.1. Dada la gramática lineal par definida por las producciones

$S \to aSb \mid aBa \mid bCb \mid a \mid b$

$B \to aBa \mid a$

$C \to bCb \mid b$

el autómata que se obtiene de acuerdo a la construcción del teorema 4.1 es el siguiente



□

Una vez mostrada la relación entre un lenguaje lineal par L y su lenguaje regular asociado $\sigma(L)$, podemos establecer cierta correspondencia entre los resultados teóricos en los lenguaje regulares y aquéllos de los lenguajes lineales pares. Así, podemos obtener una relación de índice finito ligada a lenguajes lineales pares que produce resultados similares a los del Teorema de Nerode para lenguajes regulares [33].

En primer lugar, como muestra de la correspondencia citada anteriormente, podemos establecer la equivalencia entre las gramáticas lineales pares no deterministas y aquellas que sí lo son. Lo realizaremos mediante el siguiente teorema.

Teorema 4.2. Dada una gramática lineal par en forma normal $G = (N, \Sigma, P, S)$, existe una gramática lineal par determinista en forma normal G' tal que $L(G) = L(G')$.

Demostración.

Dada G podemos obtener un autómata finito A que acepte $\sigma(L(G))$ tal y como se establece en el teorema 4.1. Utilizando operaciones estándar sobre este autómata [33], podemos obtener un AF determinista A' equivalente a A. Aplicando a A' la transformación inversa podemos obtener una gramática lineal par G' que en este caso será determinista y genera el lenguaje $\sigma^{-1}(\sigma(L(G))) = L(G)$.

□

A partir de cada lenguaje L definido sobre Σ podemos establecer una relación de equivalencia $\equiv_L$ sobre pares de cadenas similar a la de Nerode tal y como lo definimos a continuación.

Definición 4.7. Dado el lenguaje L, diremos que los pares de cadenas (u_1, v_1) y (u_2, v_2) están relacionados bajo L (son *indistinguibles* bajo L) y lo denotamos como $(u_1, v_1) \equiv_L (u_2, v_2)$ si y solo si se cumplen las siguientes condiciones:

1. $|u_1| = |v_1|$
2. $|u_2| = |v_2|$
3. $\forall w \in \Sigma^*\ u_1 w v_1 \in L$ sii $u_2 w v_2 \in L$ o, utilizando una notación alternativa, diremos que $(u_1, v_1)L = (u_2, v_2)L$, donde $(u, v)L = u^{-1}(Lv^{-1}) = (u^{-1}L)v^{-1}$.

□

A partir de la anterior definición, podemos establecer un resultado definitivo que caracteriza a la clase de los lenguaje lineales pares. Lo enunciaremos mediante el siguiente teorema.

Teorema 4.3. $L \subseteq \Sigma^*$ es un lenguaje lineal par sii $\equiv_L$ es de índice finito.

Demostración.

- Condición necesaria.

  Sea el lenguaje $L = L(G)$, donde $G = (N, \Sigma, P, S)$ es una gramática lineal par determinista tal y como se establece en el teorema 4.2. Definimos la relación $\equiv_G$ sobre pares de cadenas como sigue. $(u_1, v_1) \equiv_G (u_2, v_2)$ sii

1. $|u_1| = |v_1|$
2. $|u_2| = |v_2|$
3. $(\forall A \in N)\ S \overset{*}{\underset{G}{\Rightarrow}} u_1 A v_1$ sii $S \overset{*}{\underset{G}{\Rightarrow}} u_2 A v_2$

Obviamente, $\equiv_G$ es de índice finito, ya que tendrá tantas clases de equivalencia como símbolos no terminales tenga la gramática. Demostraremos que si $(u_1, v_1) \equiv_G (u_2, v_2)$, entonces $(u_1, v_1) \equiv_L (u_2, v_2)$, y por lo tanto $\equiv_L$ es de índice finito.

$(u_1, v_1) \equiv_G (u_2, v_2) \Rightarrow \forall w \in \Sigma^*, u_1 w v_1 \in L$ sii $u_2 w v_2 \in L \Rightarrow (u_1, v_1) \equiv_L (u_2, v_2)$.

- Condición suficiente.

  En este caso, supongamos que $\equiv_L$ es de índice finito. Definiremos la gramática $G = (N, \Sigma, P, S)$, donde $N = \{(u, v)L : u, v \in \Sigma^* \text{ y } |u| = |v|\}$, $S = (\lambda, \lambda)L$ y P se define mediante las siguientes reglas

  1. Si $(u, v)L = A$ y $(ua, bv)L = B$, entonces $A \to aBb \in P$.
  2. Si $a \in A \cap (\Sigma \cup \{\lambda\})$, entonces $A \to a \in P$.

  Comprobemos ahora que $L(G) = L$. En primer lugar, se puede demostrar que $(u, v)L = A$ sii $S \overset{*}{\underset{G}{\Rightarrow}} uAv$, mediante un proceso inductivo.

  Una vez demostrado el anterior enunciado, podemos comprobar que $L(G) = L$, mediante una doble inclusión.

  1. $L(G) \subseteq L$

     Tomemos $x \in L(G)$. Se cumple que $S \overset{*}{\underset{G}{\Rightarrow}} uAv \underset{G}{\Rightarrow} uav = x$ con $|u| = |v|$ y $a \in \Sigma \cup \{\lambda\}$, entonces $(u, v)L = A$ y $a \in A \cap (\Sigma \cup \{\lambda\})$, por lo tanto $uav = x \in L$.
  2. $L \subseteq L(G)$

     Tomemos $x = uav \in L$ con $|u| = |v|$ y $a \in \Sigma \cup \{\lambda\}$, entonces $a \in (u, v)L = A$. Es claro que $A \to a \in P$ y $S \overset{*}{\underset{G}{\Rightarrow}} uAv$, por lo que $S \overset{*}{\underset{G}{\Rightarrow}} uAv \underset{G}{\Rightarrow} uav = x \in L(G)$.

  □

Veamos un ejemplo de cómo podemos construir una gramática lineal par a partir de la relación de equivalencia tal y como ha quedado establecido en el anterior teorema.

Ejemplo 4.2. Tomemos el lenguaje $L = aa^*b^*b$. Las clases de equivalencia inducidas por $\equiv_L$ son las siguientes

$(a,a)L = \oslash$	$(a,a)A = a^* = B$	$(a,a)B = a^* = B$	$(a,a)C = \oslash$
$(a,b)L = a^*b^* = A$	$(a,b)A = a^*b^* = A$	$(a,b)B = \oslash$	$(a,b)C = \oslash$
$(b,a)L = \oslash$	$(b,a)A = \oslash$	$(b,a)B = \oslash$	$(b,a)C = \oslash$
$(b,b)L = \oslash$	$(b,b)A = b^* = C$	$(b,b)B = \oslash$	$(b,b)C = b^* = C$

A partir de las anteriores clases de equivalencia se obtiene la siguiente gramática aplicando la construcción del teorema 4.3.

$S \to aAb$	$A \to aBa \mid aAb \mid bCb \mid a \mid b \mid \lambda$
$B \to aBa \mid a \mid \lambda$	$C \to bCb \mid b \mid \lambda$

□

Finalmente, podemos enunciar un resultado relacionado con el tamaño mínimo de las gramáticas lineales pares.

Teorema 4.4. La gramática obtenida según el teorema 4.3 es la gramática lineal par determinista mínima respecto del número de símbolos auxiliares que genera L y es única salvo isomorfismos en los símbolos auxiliares.

Demostración.

Tal y como se ha visto en el teorema 4.3, dada una gramática lineal par G, la relación $\equiv_G$ es un refinamiento sobre $\equiv_L$, por lo que el número de símbolos auxiliares inducidos por $\equiv_G$, para cualquier gramática G, es mayor o igual al número de símbolos inducidos por $\equiv_L$.

□

4.3 Aprendizaje de lenguajes lineales pares.

Una vez presentada la caracterización de los lenguajes lineales pares, el objetivo es aplicarla para obtener un método de aprendizaje de los mismos. Puede comprobarse fácilmente que el aprendizaje de un lenguaje lineal par L puede resolverse mediante el aprendizaje de su lenguaje regular asociado $\sigma(L)$. Así, el problema de aprendizaje de lenguajes lineales pares se reduce al aprendizaje de los lenguajes regulares. La caracterización propuesta en la sección anterior

es diferente de la propuesta por Takada [70] pero permite obtener el mismo resultado desde el punto de vista del aprendizaje.

Un esquema que puede aplicarse para abordar la tarea de aprendizaje en este caso es el que se propone en la figura 4.1.

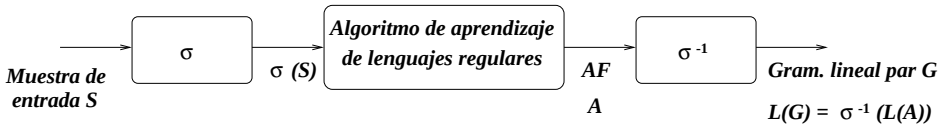


Figura 4.1: Un esquema de aprendizaje de lenguajes lineales pares.

El esquema propuesto es fácil de entender. Dada una muestra del lenguaje lineal par, se le aplica la transformación σ para obtener una muestra de un lenguaje regular. Se puede entonces aplicar sobre la muestra transformada cualquier algoritmo de aprendizaje de lenguajes regulares como el propuesto por Oncina y García [51] que, a partir de la muestra de entrada, obtiene como hipótesis de salida un autómata finito determinista. Por último se aplica la transformación inversa σ^{-1} sobre la hipótesis de salida y se obtiene una gramática lineal par. Obsérvese que en el caso de utilizar el algoritmo de Oncina y García queda garantizada la identificación en el límite de la clase $\mathcal{REG}$. Así, la clase $\mathcal{ELL}$ también es identificable en el límite.

Otra forma de llevar a cabo el proceso de identificación de la clase $\mathcal{ELL}$ sería la adaptación de algoritmos de aprendizaje que trabajen directamente con la muestra sin necesidad de aplicarle ninguna transformación. Así se podrían adaptar algunos algoritmos que se basan en la técnica de agrupamiento de estados como son los expuestos en [51] y [73]. Esta aproximación ya fue expuesta por el autor en [67].

4.4 Explorabilidad local en lenguajes lineales pares

Una vez resuelto el problema genérico del aprendizaje de lenguajes lineales pares, nos proponemos a continuación estudiar algunas clases de lenguajes que, a partir de reducciones similares a las que hemos expuesto, puedan ser inferidas mediante algoritmos de aprendizaje de lenguajes regulares a partir de texto. En concreto, nos centraremos en la *explorabilidad local* como un rasgo distintivo de algunos lenguajes lineales pares que habilitan este tipo de aprendizaje. El análisis que nos proponemos realizar se asemeja en cierto modo al que realizaron Koshiba, Mäkinen y Takada [40] en el que establecieron, como rasgo característico, la *reversibilidad* en los lenguajes lineales pares.

Comenzaremos por definir la clase de lenguajes regulares a los que reduciremos los lenguajes lineales pares que procedemos a estudiar. Posteriormente, estableceremos distintos tipos de explorabilidad que darán lugar a variaciones en las clases de lenguajes lineales pares correspondientes.

Conceptos básicos sobre explorabilidad local

Los lenguajes *localmente explorables* fueron definidos por McNaughton y Papert [47]. En su trabajo, establecieron las características que esta subclase de lenguajes regulares debían cumplir. Proporcionaremos, a continuación, la definición de los mismos

Definición 4.8. Sea el valor entero $k > 0$ y la cadena $x \in \Sigma^*$. Definiremos el *vector de k-explorabilidad* de x como la tupla $v_k(x) = (i_k(x), t_k(x), f_k(x))$, donde

$$i_k(x) = \begin{cases} x & \text{si} \quad |x| < k \\ u : x = uv, |u| = k-1 & \text{si} \quad |x| \geq k \end{cases}$$

$$f_k(x) = \begin{cases} x & \text{si} \quad |x| < k \\ v : x = uv, |v| = k-1 & \text{si} \quad |x| \geq k \end{cases}$$

$$t_k(x) = \{v : x = uvw, u \in \Sigma^*, w \in \Sigma^*, |v| = k\}$$

□

Definición 4.9. Para cada valor entero $k > 0$, definiremos la relación de equivalencia $\equiv_k \subseteq \Sigma^* \times \Sigma^*$ como sigue:

$$(\forall x, y \in \Sigma^*) \quad x \equiv_k y \iff v_k(x) = v_k(y)$$

□

Se ha demostrado en [47] que $\equiv_k$ es una relación de índice finito y que $\equiv_{k+1}$ refina a $\equiv_k$.

Definición 4.10. Diremos que $L \subseteq \Sigma^*$ es k-explorable sii L es la unión de algunas clases de equivalencia de $\equiv_k$. L es localmente explorable si L es k-explorable para algún valor $k > 0$. □

Denotaremos a la clase de lenguajes k-explorables por k-$\mathcal{LT}$ y a la clase de lenguajes localmente explorables por $\mathcal{LT}$. Ha sido demostrado en [47] que $\mathcal{LT} \subset \mathcal{REG}$.

Otra clase de lenguajes que está íntimamente relacionada con $\mathcal{LT}$, es la clase de lenguajes *localmente explorables en sentido estricto*, introducida en [47]. La citada clase de lenguajes la definiremos a continuación.

Definición 4.11. Sea Σ un alfabeto y $Z_k = (\Sigma, I_k, F_k, T_k)$, donde $I_k, F_k \subseteq \Sigma^{\leq k-1}$ y $T_k \subseteq \Sigma^k$. Diremos que L es un lenguaje k-explorable en sentido estricto si podemos definir L mediante la siguiente expresión

$$L \cap \Sigma^{k-1}\Sigma^* = (I_k\Sigma^*) \cap (\Sigma^* F_k) - (\Sigma^* T_k \Sigma^*)$$

Diremos que L es localmente explorable en sentido estricto si es k-explorable en sentido extricto para algún valor entero $k > 0$. □

Denotaremos a la clase de lenguajes k-explorables en sentido estricto por k-$\mathcal{LTSS}$ y a la clase de lenguajes localmente explorables en sentido estricto por $\mathcal{LTSS}$. Ha sido demostrado en [47] que $\mathcal{LTSS} \subset \mathcal{REG}$ y que k-$\mathcal{LT}$ es la clausura booleana de k-$\mathcal{LTSS}$. Obsérvese que las clases k-$\mathcal{LT}$ y k-$\mathcal{LTSS}$ son cerradas bajo la operación de reverso.

Explorabilidad local en lenguajes lineales pares

Una vez que hemos definido la explorabilidad local en relación con los lenguajes regulares, nos proponemos aplicar las reducciones que hemos utilizado de forma genérica en los lenguajes lineales pares, para definir en los mismos algunos tipos de explorabilidad local. En primer lugar, además de utilizar la transformación σ definida anteriormente, emplearemos un morfismo $g : (\Sigma\Sigma) \cup \Sigma \to \Delta$, donde Δ es un alfabeto cuya cardinalidad cumple que $card(\Delta) \geq card(\Sigma)^2 + 2 \cdot card(\Sigma)$. Así, si $\Sigma = \{a_1, a_2, \cdots, a_n\}$ y $\Delta = \{b_1, b_2, \cdots, b_m\}$ entonces g se definirá como sigue

1. $(\forall a_i \in \Sigma)\ g(a_i) = b_i$
2. $(\forall a_i, a_j \in \Sigma)\ g([a_i a_j]) = b_{n+i \cdot n+j}$

La utilización del morfismo g no es estrictamente necesaria. El objetivo que se persigue al introducirlo en este punto es el de la simplificación de los alfabetos empleados en los procesos de inferencia. Así, al utilizar g podemos volver a trabajar con alfabetos que se componen de símbolos simples y no de pares de símbolos como ocurría en el anterior apartado.

Procederemos a continuación a definir la explorabilidad local en los lenguajes lineales pares.

Entrada: Una muestra finita de aprendizaje R
Método:
$S = \emptyset$
$\forall x \in R$
$S = S \cup \{g(\sigma(x))\}$
$\mathcal{H} = A_{\mathcal{LT}}(S)$
Salida: $\sigma^{-1}(g^{-1}(\mathcal{H}))$

Figura 4.2: Un método de inferencia para lenguajes de $\mathcal{LT}_{\mathcal{ELL}}$

Definición 4.12. Sea $L \in \mathcal{ELL}$. Diremos que L es un lenguaje lineal par localmente explorable sii $g(\sigma(L)) \in \mathcal{LT}$. Diremos que L es un lenguaje lineal par localmente explorable en sentido estricto sii $g(\sigma(L)) \in \mathcal{LTSS}$. □

Denotaremos a la clase de los lenguajes lineales pares localmente explorables por $\mathcal{LT}_{\mathcal{ELL}}$ y a la clase de los lenguajes lineales pares localmente explorables en sentido estricto por $\mathcal{LTSS}_{\mathcal{ELL}}$. Si fijamos un valor k concreto entonces tendremos las clases de lenguajes $k - \mathcal{LT}_{\mathcal{ELL}}$ y $k - \mathcal{LTSS}_{\mathcal{ELL}}$. A partir de la definición 4.12, podemos diseñar una técnica de reducción que resuelva el aprendizaje de los lenguajes de $\mathcal{LT}_{\mathcal{ELL}}$ y de $\mathcal{LTSS}_{\mathcal{ELL}}$ aplicando algoritmos de aprendizaje para las clases $\mathcal{LT}$ o $\mathcal{LTSS}$. Así, supongamos que el algoritmo de aprendizaje para $\mathcal{LT}$ es $A_{\mathcal{LT}}$ tal y como describe Ruiz en [61], entonces el método de reducción que podemos emplear se muestra en la figura 4.2

El método para el aprendizaje de lenguajes de $\mathcal{LTSS}_{\mathcal{ELL}}$ es similar al mostrado en la figura 4.2. En este caso, bastaría con emplear un método de aprendizaje para lenguajes de $\mathcal{LTSS}$ como el expuesto en [20]. Suponiendo que denotamos por $A_{\mathcal{LTSS}}$ al citado método, entonces el algoritmo de inferencia es el que se muestra en la figura 4.3.

Veamos a continuación un ejemplo de la reducción propuesta en los métodos de aprendizaje expuestos anteriormente.

Ejemplo 4.3. Supongamos que la muestra R se define mediante el conjunto

$$R = \{10, 10001110, 111000111000, 11100001111000\}$$

entonces $\sigma(R)$ queda definido por el conjunto

$$\{[10], [10][01][01][01], [10][10][10][01][01][01], [10][10][10][01][01][01][01]\}$$

Entrada: Una muestra finita de aprendizaje R
Método:
$S = \emptyset$
$\forall x \in R$
$S = S \cup \{g(\sigma(x))\}$
$\mathcal{H} = A_{\mathcal{LTSS}}(S)$
Salida: $\sigma^{-1}(g^{-1}(\mathcal{H}))$

Figura 4.3: Un método de inferencia para lenguajes de $\mathcal{LTSS}_{\mathcal{ELL}}$

Si aplicamos el morfismo g tal que $g([10]) = a$ y $g([01]) = b$ entonces $g(\sigma(R)) = \{a, abbb, aaabbb, aaabbbb\}$. A continuación, aplicando el algoritmo $A_{\mathcal{LTSS}}(S)$ sobre $g(\sigma(R))$, con $k = 2$, obtenemos una hipótesis $\mathcal{H}$ tal que $\sigma^{-1}(g^{-1}(\mathcal{H}))$ queda definida por la gramática lineal par con las producciones siguientes: $S \to 1A0$; $A \to 1A0 \mid 0B1 \mid \lambda$; $B \to 1A0 \mid 0B1 \mid \lambda$. □

Una definición gramatical de los lenguajes lineales pares localmente explorables

Una vez resuelto de forma efectiva el aprendizaje de los lenguajes lineales pares localmente explorables, podemos aprovechar la reducción aplicada en el mismo para definir las características que deben cumplir las gramáticas lineales pares que generan los citados lenguajes. Formalizaremos dichas características mediante la siguiente propiedad.

Propiedad 4.1. Sea el valor entero $k > 0$ y la gramática lineal par $G = (N, \Sigma, P, S)$ de forma que se cumple la siguiente condición: $(\forall A \in N)$ si $S \overset{*}{\underset{G}{\Rightarrow}} x_1 A y_1$ y $S \overset{*}{\underset{G}{\Rightarrow}} x_2 A y_2$ entonces $(\forall w \in \Sigma^*)\ g(\sigma(x_1 w y_1)) \equiv_k g(\sigma(x_2 w y_2))$. Entonces podemos afirmar que $L(G) \in k - \mathcal{LT}_{\mathcal{ELL}}$.

Demostración.

Supongamos que G cumple el enunciado de la propiedad. Entonces, de forma trivial, las cadenas del lenguaje $L(G)$ pueden obtenerse a partir de $\mathcal{O}(card(N))$ clases de equivalencia de $\equiv_k$ y, por lo tanto, $L(G) \in k - \mathcal{LT}_{\mathcal{ELL}}$.
□

Un resultado similar al anterior puede establecerse para caracterizar las gramáticas lineales pares que generan lenguajes de $\mathcal{LTSS}_{\mathcal{ELL}}$.

4.5 Otras formas de explorabilidad local

En la sección anterior hemos mostrado algunas relaciones entre los lenguajes lineales pares y los lenguajes regulares a partir de las transformaciones σ y g. En esta sección cambiaremos las transformaciones anteriores con el objetivo de definir y caracterizar nuevas clases de lenguajes dentro de la clase $\mathcal{ELL}$ que compartan el rasgo de explorabilidad local de forma diferente a las anteriores.

Cadenas bipartidas

Puede observarse fácilmente que, en cada lenguaje lineal par, todas las cadenas del mismo pueden dividirse en dos mitades de igual longitud y, posiblemente, un símbolo central que se pueden poner en correspondencia con las cadenas que aparecen a izquierda y derecha de cada símbolo auxiliar en las partes derechas de las producciones de las gramáticas que los generan. Tomando esta característica como punto de partida, podemos definir una nueva transformación sobre cadenas que denominaremos `Bipartir` y que dividirá a las mismas en dos mitades de igual longitud y un símbolo central o la cadena vacía.

Formalmente, para cualquier cadena $x \in \Sigma^*$, definiremos $\texttt{Bipartir}(x) = \{x_1, a, x_2^r\}$ con $x = x_1 a x_2$ y $|x_1| = |x_2|$ y $a \in \Sigma \cup \{\lambda\}$.

Podemos extender esta operación sobre un conjunto finito y arbitrario de cadenas S como sigue

$$\texttt{Bipartir}(S) = \{L(S), M(S), R(S)\}$$

donde
$L(S) = \{x_1 \in \Sigma^* : \exists x \in S, \texttt{Bipartir}(x) = \{x_1, a, x_2\}\}$,
$R(S) = \{x_2 \in \Sigma^* : \exists x \in S, \texttt{Bipartir}(x) = \{x_1, a, x_2\}\}$ y
$M(S) = \{a \in \Sigma \cup \{\lambda\} : \exists x \in S, \texttt{Bipartir}(x) = \{x_1, a, x_2\}\}$.

Biparticiones sincronizadas vs. no sincronizadas

A partir de la operación `Bipartir`, definida anteriormente, podemos obtener, mediante algoritmos de inferencia basados en explorabilidad local de lenguajes regulares, hipótesis que llamaremos *constructores* que facilitarán el proceso de aprendizaje de lenguajes lineales pares. Aquí es importante observar que, si en una gramática lineal par arbitraria, eliminamos las cadenas que aparecen

Entrada: Un conjunto finito de aprendizaje S
Método:
$T_L = L(S) \in \texttt{Bipartir}(S)$
$T_R = R(S) \in \texttt{Bipartir}(S)$
$T_M = M(S) \in \texttt{Bipartir}(S)$
$\mathcal{H}_L = A_{\mathcal{LT}}(T_L)$
$\mathcal{H}_R = A_{\mathcal{LT}}(T_R)$
$\mathcal{H}_M = T_M$
Salida: $\{\mathcal{H}_L, \mathcal{H}_M, \mathcal{H}_R\}$

Figura 4.4: Un método de inferencia para obtener constructores para biparticiones sincronizadas y no sincronizadas

a la derecha (o a la izquierda) de un símbolo auxiliar, entonces la gramática resultante es regular. En la figura 4.4 mostramos el método que permite obtener constructores a partir de la operación `Bipartir` y cualquier algoritmo de inferencia para la clase de lenguajes $\mathcal{LT}$.

Veamos a continuación un ejemplo de aplicación del método expuesto en la figura 4.4.

Ejemplo 4.4. Sea el conjunto finito S definido como
$S = \{abbaabab, abbabaababab, abbabbabbaababababab\}$,
entonces $\texttt{Bipartir}(S) = \{L(S), R, (S), M(S)\}$, donde
$L(S) = \{abba, abbaba, abbabbabba\}$,
$R(S) = \{baba, bababa, bababababa\}$ y
$M(S) = \{\lambda\}$.

A continuación aplicaremos un algoritmo de inferencia para la clase $\mathcal{LT}$ que denotamos por $A_{\mathcal{LT}}$, con $k = 2$ y obtenemos los constructores $\mathcal{H}_L$, $\mathcal{H}_R$ y $\mathcal{H}_M$ donde las clases de equivalencia obtenidas por el algoritmo de aprendizaje en cada uno de los constructores quedan definidas por los siguientes conjuntos
$\mathcal{H}_L = \{[\lambda]_{\equiv_2}, [a]_{\equiv_2}, [ab]_{\equiv_2}, [abb]_{\equiv_2}, [abba]_{\equiv_2}, [abbab]_{\equiv_2}\}$
$\mathcal{H}_R = \{[\lambda]_{\equiv_2}, [b]_{\equiv_2}, [ba]_{\equiv_2}, [bab]_{\equiv_2}, [baba]_{\equiv_2}\}$
$\mathcal{H}_M = \{\lambda\}$

□

Una vez obtenidos los constructores a partir de un conjunto finito, podemos definir diferentes subclases de lenguajes lineales pares en función de cómo

relacionemos las clases de equivalencia de los constructores. En este caso, pretendemos combinar las clases de equivalencia de la relación $\equiv_k$ que hemos inferido a partir del algoritmo $A_{\mathcal{LT}}$. Por otra parte, resulta obvio que, los conjuntos obtenidos a partir de la transformación `Bipartir` forman un lenguaje regular localmente explorable.

Las combinaciones entre las clases de equivalencia de $L(S)$ y $R(S)$ que hemos citado las realizaremos mediante *sincronismo* o ausencia de *sincronismo*. Informalmente, el *sincronismo* entre las clases de equivalencia significa que, si tomamos cualquier cadena del conjunto de aprendizaje $x = x_1 a x_2 \in S$, entonces la clase de equivalencia obtenida para x_1 debe combinarse con aquella obtenida para x_2^r. De esta forma, podemos definir la relación $\sim_k$ entre cadenas como sigue: $\forall x, y \in \Sigma^*$ con $x = x_1 a x_2$, $y = y_1 b y_2$, $|x_1| = |x_2|, |y_1| = |y_2|$ y $a, b \in \Sigma \cup \{\lambda\}$

$$x \sim_k y \Longleftrightarrow v_k(x_1) = v_k(y_1) \wedge v_k(x_2) = v_k(y_2)$$

A partir de la relación $\sim_k$ definiremos una clase de lenguajes que tome en cuenta el sincronismo ente las clases de equivalencia.

Definición 4.13. Diremos que el lenguaje $L \subseteq \Sigma^*$ es k-explorable sincronizado bipartido sii es la unión de algunas clases de equivalencia de $\sim_k$. L es localmente explorable sincronizado bipartido si L es k-explorable sincronizado bipartido para algún valor entero $k > 0$. □

Denotaremos a la clase de los lenguajes lineales pares *localmente explorables sincronizados bipartidos* por $\mathcal{LTSHS}_{ELL}$. Fijando un valor entero k tendremos la clase de lenguajes $k - \mathcal{LTSHS}_{ELL}$. Podemos inferir lenguajes de la clase $\mathcal{LTSHS}_{ELL}$ mediante el método expuesto en la figura 4.5.

La identificación en el límite de la clase $\mathcal{LTSHS}_{ELL}$ puede ser fácilmente deducible a partir de la convergencia de las hipótesis $\mathcal{H}_L$ y $\mathcal{H}_R$ utilizando el algoritmo $A_{\mathcal{LT}}$. Obsérvese que el método propuesto en la figura 4.5 es *conservativo*, lo que quiere decir que sólo se combinan aquellas clases de equivalencia de la relación $\equiv_k$ que tienen algún representante en el conjunto de aprendizaje.

Ejemplo 4.5. Tomemos el conjunto de aprendizaje y los constructores definidos en el ejemplo 4.4. La hipótesis obtenida por el método propuesto en la figura 4.5 es la gramática lineal par definida por las siguientes producciones

$A_{00} \to aA_{11}b$ $\quad$ $A_{44} \to bA_{53}b \mid \lambda$
$A_{11} \to bA_{22}a$ $\quad$ $A_{53} \to aA_{44}a \mid bA_{54}a$
$A_{22} \to bA_{33}b$ $\quad$ $A_{54} \to bA_{53}b \mid aA_{43}b$
$A_{33} \to aA_{44}a$ $\quad$ $A_{43} \to bA_{54}a$

□

Entrada: Un conjunto finito de aprendizaje A y constructores $\mathcal{H}_L, \mathcal{H}_M$ y $\mathcal{H}_R$
Método:

Enumerar $\mathcal{H}_L$ y $\mathcal{H}_R$
/* $card(\mathcal{H}_L) = n$ y $card(\mathcal{H}_R) = m$ */
$N = \{A_{ij} : 0 \leq i < n, 0 \leq j < m\}$
$S = A_{00}$
$\forall x \in A$
$\forall u, v, w \in \Sigma^* \; \forall a, b \in \Sigma : x = uawbv$ con $|u| = |v|$
/* $[u]_{\equiv_k}$ es la i-ésima clase de equivalencia en $\mathcal{H}_L$ */
/* $[v^r]_{\equiv_k}$ es la j-ésima clase de equivalencia en $\mathcal{H}_R$ */
/* $[ua]_{\equiv_k}$ es la k-ésima clase de equivalencia en $\mathcal{H}_L$ */
/* $[v^r b]_{\equiv_k}$ es la l-ésima clase de equivalencia en $\mathcal{H}_R$ */
$A_{ij} \rightarrow aA_{kl}b \in P$
si $w \in \mathcal{H}_M$ entonces $A_{kl} \rightarrow w \in P$

Salida: $G = (N, \Sigma, P, S)$

Figura 4.5: Un método de inferencia para lenguajes de $\mathcal{LTSHS}_{ELL}$.

El siguiente resultado muestra las características que cumplen las gramáticas lineales pares que generan los lenguajes de la clase $\mathcal{LTSHS}_{ELL}$.

Propiedad 4.2. Sea el valor entero positivo $k > 0$ y la gramática lineal par $G = (N, \Sigma, P, S)$ de forma que la siguiente condición se cumple: $(\forall A \in N)$ si $S \overset{*}{\underset{G}{\Rightarrow}} x_1 A y_1$ entonces $S \overset{*}{\underset{G}{\Rightarrow}} x_2 A y_2$ sii $x_1 \equiv_k x_2$ y $y_1 \equiv_k y_2$. Entonces, $L(G) \in k - \mathcal{LTSHS}_{\mathcal{ELL}}$.

Demostración.

Supongamos que la gramática G cumple el enunciado de la propiedad. De forma trivial, tomemos dos cadenas distintas de $L(G)$ que puedan ser obtenidas en la gramática mediante los siguientes procesos de derivación

$$S \overset{*}{\underset{G}{\Rightarrow}} x_1 A y_1 \underset{G}{\Rightarrow} x_1 a y_1$$
$$S \overset{*}{\underset{G}{\Rightarrow}} x_2 A y_2 \underset{G}{\Rightarrow} x_2 a y_2$$

Puede demostrarse fácilmente que, de acuerdo con las anteriores derivaciones, $x_1 \equiv_k x_2$ e $y_1 \equiv_k y_2$. Así, las cadenas del lenguaje $L(G)$ pueden obtenerse

con un máximo de $\mathcal{O}(card(N))$ clases de equivalencia de la relación $\sim_k$. En consecuencia, $L(G) \in k - \mathcal{LTSHS}_{\mathcal{ELL}}$

□

La definición de lenguajes *bipartidos no sincronizados* lineales pares es bastante simple. En este caso, basta con no forzar el enlace entre las clases de equivalencia obtenidas por la derecha o la izquierda. Definiremos la relación entre cadenas $\approx_k$ como sigue: $\forall x, y \in \Sigma^*$ con $x = x_1 a x_2$, $y = y_1 b y_2$, $|x_1| = |x_2|, |y_1| = |y_2|$ y $a, b \in \Sigma \cup \{\lambda\}$

$$x \approx_k y \Longleftrightarrow v_k(x_1) = v_k(y_1) \vee v_k(x_2) = v_k(y_2)$$

A continuación definiremos los lenguajes que se pueden formar a partir de la anterior relación.

Definición 4.14. Diremos que $L \subseteq \Sigma^*$ es un lenguaje k-explorable no sincronizado bipartido sii es la unión de algunas clases de equivalencia de la relación $\approx_k$. L es localmente explorable no sincronizado bipartido si es k-explorable no sincronizado bipartido para algún valor $k > 0$. □

Denotaremos a la clase de lenguajes lineales pares *localmente explorables no sincronizados bipartidos* por $\mathcal{LTNSHS}_{ELL}$. Fijando un valor entero positivo k tendremos la clase de lenguajes $k - \mathcal{LTNSHS}_{ELL}$. Podemos inferir lenguajes de la clase $\mathcal{LTNSHS}_{ELL}$ a partir del método expuesto en la figura 4.6.

Al igual que sucedía con la anterior clase de lenguajes, la identificación en el límite de la clase $\mathcal{LTNSHS}_{ELL}$ puede ser fácilmente demostrada a partir de la convergencia de las hipótesis $\mathcal{H}_L$ y $\mathcal{H}_R$ empleando el algoritmo $A_{\mathcal{LT}}$.

Veamos un ejemplo de aplicación del algoritmo mostrado en la figura 4.6.

Ejemplo 4.6. Tomemos el conjunto de aprendizaje y los constructores definidos en el ejemplo 4.4. La hipótesis obtenida por el método de la figura 4.6 queda definida por la gramática con las siguientes producciones

$S \to aA_{1L}a \mid aA_{1L}b \mid aA_{1R}b \mid bA_{1R}b$

$A_{0L} \to aA_{1L}a \mid aA_{1L}b$ $\quad$ $A_{0R} \to aA_{1R}b \mid bA_{1R}b$

$A_{1L} \to bA_{2L}a \mid bA_{2L}b$ $\quad$ $A_{1R} \to aA_{2R}a \mid bA_{2R}a$

$A_{2L} \to bA_{3L}a \mid bA_{3L}b$ $\quad$ $A_{2R} \to aA_{3R}b \mid bA_{3R}b$

$A_{3L} \to aA_{4L}a \mid aA_{4L}b$ $\quad$ $A_{3R} \to aA_{4R}a \mid bA_{4R}a$

$A_{4L} \to \lambda \mid bA_{5L}a \mid bA_{5L}b$ $\quad$ $A_{4R} \to \lambda \mid aA_{3R}b \mid bA_{3R}b$

$A_{5L} \to aA_{4L}a \mid aA_{4L}b \mid bA_{5L}a \mid bA_{5L}b$

□

Entrada: Un conjunto finito de aprendizaje A y constructores $\mathcal{H}_L, \mathcal{H}_M$ y $\mathcal{H}_R$
Método:

Enumerar $\mathcal{H}_L$ y $\mathcal{H}_R$
/* $card(\mathcal{H}_L) = n$ y $card(\mathcal{H}_R) = m$ */
$N = \{A_{iL} : 0 \leq i < n\} \cup \{A_{jR} : 0 \leq j < m\} \cup \{S\}$
$\forall x \in A$
  $\forall u \in \Sigma^* \ \forall a \in \Sigma : x = uawv$ con $|ua| \leq |v|$
  /* $[u]_{\equiv_k}$ es la i-ésima clase de equivalencia de $\mathcal{H}_L$ */
  /* $[ua]_{\equiv_k}$ es la k-ésima clase de equivalencia de $\mathcal{H}_L$ */
    $\forall b \in \Sigma$
      $A_{iL} \rightarrow aA_{kL}b \in P$
      si $w \in \mathcal{H}_M$ entonces $A_{kL} \rightarrow w \in P$
$\forall x \in A$
  $\forall v \in \Sigma^* \ \forall b \in \Sigma : x = uwbv$ con $|bv| \leq |u|$
  /* $[v^r]_{\equiv_k}$ es la i-ésima clase de equivalencia de $\mathcal{H}_R$ */
  /* $[v^r b]_{\equiv_k}$ es la k-ésima clase de equivalencia de $\mathcal{H}_R$ */
    $\forall a \in \Sigma$
      $A_{iR} \rightarrow aA_{kR}b \in P$
      si $w \in \mathcal{H}_M$ entonces $A_{kR} \rightarrow w \in P$
si $A_{0L} \rightarrow \alpha$ entonces $S \rightarrow \alpha \in P$
si $A_{0R} \rightarrow \alpha$ entonces $S \rightarrow \alpha \in P$

Salida: $G = (N, \Sigma, P, S)$

Figura 4.6: Un método para inferir lenguajes de $\mathcal{LTNSHS}_{ELL}$

En cuanto a las características de las gramáticas que generan los lenguajes de la clase $\mathcal{LTNSHS}_{ELL}$, podemos establecer el siguiente resultado.

Propiedad 4.3. Sea el valor entero $k > 0$ y la gramática lineal par $G = (N, \Sigma, P, S)$ que cumple la siguiente condición: $(\forall A \in N)$ si $S \overset{*}{\underset{G}{\Rightarrow}} x_1 A y_1$ entonces $S \overset{*}{\underset{G}{\Rightarrow}} x_2 A y_2$ sii $x_1 \equiv_k x_2$ o $y_1 \equiv_k y_2$. Entonces, $L(G) \in k - \mathcal{LTNSHS}_{\mathcal{ELL}}$.

Demostración.

Supongamos que la gramática G cumple la condición que enuncia la propiedad. Puede comprobarse fácilmente que, si tomamos dos cadenas distintas de $L(G)$

que se puedan obtener mediante los siguiente procesos derivativos en la gramática

$$S \overset{*}{\underset{G}{\Rightarrow}} x_1 A y_1 \underset{G}{\Rightarrow} x_1 a y_1$$
$$S \overset{*}{\underset{G}{\Rightarrow}} x_2 A y_2 \underset{G}{\Rightarrow} x_2 a y_2$$

entonces, de acuerdo con las derivaciones anteriores, $x_1 \equiv_k x_2$ o $y_1 \equiv_k y_2$. Las cadenas del lenguaje $L(G)$ pueden obtenerse como máximo a partir de $\mathcal{O}(card(N))$ clases de equivalencia de la relación $\approx_k$. Por lo tanto, $L(G) \in k - \mathcal{LTNSHS}_{\mathcal{ELL}}$ □

Finalmente, a partir de las propiedades 4.2 y 4.3, podemos deducir el siguiente resultado

Corolario 4.1. $\mathcal{LTNSHS}_{ELL} \subseteq \mathcal{LTSHS}_{ELL}$ □

4.6 Conclusiones y problemas abiertos

En este capítulo hemos estudiado algunas características de las gramáticas lineales pares que posibilitan su identificación mediante una reducción al aprendizaje de lenguajes regulares. En el caso de la explorabilidad local, hemos fijado como lenguajes regulares aquéllos definidos por las clases $\mathcal{LT}$ y $\mathcal{LTSS}$. En ambos casos, las reducciones que se han proporcionado han sido polinómicas lo que implica que los métodos de aprendizaje han resultado eficientes desde el punto de vista de complejidad computacional.

Los futuros trabajos que se pueden derivar a partir de lo expuesto en el presente capítulo los podemos exponer como sigue

- En las formas de explorabilidad local que se han planteado, podemos sustituir las clases de lenguajes regulares por otras subclases como las que se definen en [61]. Este cambio en las clases de lenguajes regulares nos conducen a nuevas definiciones de lenguajes lineales pares tales como *localmente testables por la izquierda, por la derecha, a trozos,* etc.

- A partir de las caracterizaciones algebraicas de los lenguajes localmente testables basadas en la Teoría de semigrupos, se puede establecer un trabajo en paralelo que caracterice las propiedades de las cuasi-congruencias que definen los lenguajes lineales pares.

- Las relaciones entre las clases $\mathcal{LTSHS}_{ELL}$, $\mathcal{LTNSHS}_{ELL}$, $\mathcal{LT}_{ELL}$ y $\mathcal{LTSS}_{ELL}$ no han sido analizadas en profundidad. En concreto existen indicios que nos permiten pensar que la inclusión del corolario 4.1 es, en realidad, una inclusión propia y que $\mathcal{LT}_{ELL} = \mathcal{LTSHS}_{ELL}$.

Capítulo 5

Aprendizaje de lenguajes lineales

La clase de los lenguajes lineales [65] presenta algunas propiedades interesantes desde el punto de vista de la teoría de lenguajes y de la inferencia gramatical. En esta clase se presentan algunos problemas que se repetirán en otras clases superiores y que justifican la utilización de la información estructural como fuente de información para el proceso de inferencia. Los lenguajes lineales forman una familia que está propiamente incluida en la familia de los lenguajes incontextuales y que contienen propiamente a otras familias de lenguajes como son los lineales pares [1] que hemos tratado en el capítulo anterior. Por otra parte, existe una fuerte relación entre los lenguajes lineales y las transducciones racionales [6].

Tal y como hemos indicado anteriormente, en los lenguajes lineales ya aparecen algunos de los problemas característicos de otras clases superiores, en especial de los lenguajes incontextuales, cuya resolución se ha demostrado indecidible o con una complejidad computacional elevada. De entre estos problemas destacaremos los siguientes

1. El problema de la equivalencia [60].

   Dadas dos gramáticas lineales G_1 y G_2 es indecidible establecer si son equivalentes, es decir si $L(G_1) = L(G_2)$.

2. El problema de la ambigüedad [26].

   Dada una gramática lineal mínima[1] es indecidible establecer si es o no

[1] Una gramática lineal mínima se define como aquella en la que el alfabeto de símbolos no terminales contiene un único símbolo.

ambigua.

3. El problema del conjunto característico [10].

   Dada una gramática lineal no es posible calcular un conjunto finito que permita su identificación en tiempo polinómico.

4. El problema de la equivalencia estructural [36].

   Dadas dos gramáticas lineales G_1 y G_2 establecer si son estructuralmente equivalentes es $\mathcal{PSPACE}$ completo. Dos gramáticas lineales son estructuralmente equivalentes si el conjunto de esqueletos asociados a las derivaciones de las palabras de los lenguajes que definen son idénticos.

Los problemas acerca de la equivalencia, la equivalencia estructural y otros como la cobertura de gramáticas y la decidibilidad de la linealidad de los lenguajes formales han sido debidamente estudiados a lo largo del tiempo [25, 26, 27, 35, 36, 37, 54, 60, 66].

El trabajo realizado acerca de la identificación de lenguajes lineales no ha sido muy frecuente en la literatura sobre el tema. Fundamentalmente, la comunidad científica se ha centrado en la inferencia de lenguajes incontextuales que, en consecuencia, incluye la inferencia de lenguajes lineales. Podemos citar, entre otros trabajos, una aportación del autor acerca de la inferencia heurística de cierta subclase de lenguajes lineales [68] y, recientemente, el aprendizaje de lenguajes lineales deterministas a partir de muestra completa [11].

En el presente capítulo abordaremos el aprendizaje de ciertas subclases de lenguajes lineales a partir de información estructural. En concreto nos centraremos en algunos rasgos de las gramáticas lineales como son la *distinguibilidad estructural y terminal*, la *explorabilidad local* y la *reversibilidad*.

5.1 Conceptos básicos acerca de los lenguajes lineales

Al igual que hemos hecho en el capítulo anterior, vamos a proporcionar los conceptos básicos acerca de los lenguajes lineales, de los cuales nos ocuparemos en el presente capítulo. El resto de conceptos que se utilizan han sido definidos previamente en el capítulo 3 de la presente memoria.

Definición 5.1. Sea $G = (\Sigma, N, P, S)$ una gramática. Diremos que G es lineal si cada una de las producciones de P puede tomar una de las dos siguientes formas

1. $A \to xBy \quad A, B \in N \quad x, y \in \Sigma^*$
2. $A \to w \quad A \in N \quad w \in \Sigma^*$

Podemos adoptar la siguiente forma normal para las gramáticas lineales

1. $A \to aBb \quad A, B \in N \quad ab \in \Sigma$
2. $A \to a \quad A \in N \quad a \in \Sigma \cup \{\lambda\}$

□

En general, cuando nos refiramos a gramáticas lineales se entenderán en forma normal y reducidas, es decir, sin producciones unitarias ni símbolos inútiles. Más aún, podemos eliminar las producciones $A \to \lambda$ y trabajar únicamente con lenguajes lineales que no contengan la cadena vacía. Esta es una restricción que no restará generalidad a la mayoría de resultados que se presentarán a lo largo del capítulo.

La clase de los lenguajes lineales la denotaremos por $\mathcal{LIN}$ y es una clase intermedia entre los lenguajes regulares y los libres de contexto. Es decir se cumple la siguiente relación entre familias de la jerarquía de Chomsky

$$\mathcal{REG} \subset \mathcal{ELL} \subset \mathcal{LIN} \subset \mathcal{CF}$$

Los árboles de derivación de las gramáticas lineales presentan una forma similar a los de las gramáticas regulares con la salvedad que los hijos con etiquetas terminales pueden situarse indistintamente a la izquierda o a la derecha de un nodo interno.

Ejemplo 5.1. Sea la gramática lineal definida por las producciones

$$S \to Aa \mid bB \mid a; \quad A \to aA \mid aB \quad B \to b \mid Bb$$

El árbol de derivación para la cadena $aabbba$ se muestra en la figura 5.1 junto con su esqueleto correspondiente

□

En el caso de los lenguajes lineales, la definición de gramáticas $LR(k)$ varía respecto de la utilizada en las gramáticas lineales pares en el capítulo anterior. El motivo fundamental es que, mientras que en las gramáticas lineales pares se puede tener un control a priori sobre la forma en que se derivan las cadenas del lenguaje que consiste en derivar los extremos de la cadena símbolo a símbolo para finalizar en el centro de la misma, en las gramáticas lineales se

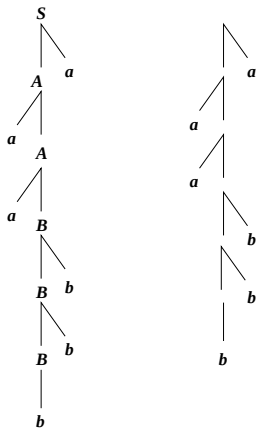


Figura 5.1: Ejemplo de árboles de derivación y esqueletos en gramáticas lineales en forma normal

pueden dar cambios de linealidad a izquierda a linealidad a derecha imposibles de predecir mediante una forma normal. Con todo ello, las relaciones entre gramáticas $LR(k)$ y SBD lineales variará respecto del caso de las gramáticas lineales pares que hemos visto en el capítulo anterior. Vamos en primer lugar a proporcionar la definición correspondiente de gramáticas $LR(k)$ lineales.

Definición 5.2. Sea $G = (N, \Sigma, P, S)$ una gramática lineal. Diremos que G es $LR(k)$ si y sólo si para todo $u, u', v, v' \in \Sigma^*$, $x, x' \in (N \cup \Sigma)^*$ y $A, A' \in N$ si se cumplen las tres siguientes condiciones

1. $S \underset{G}{\overset{*}{\Rightarrow}} uAv \underset{G}{\Rightarrow} uxv$
2. $S \underset{G}{\overset{*}{\Rightarrow}} u'A'v' \underset{G}{\Rightarrow} u'x'v'$
3. $pref(uxv, |ux| + k) = pref(u'x'v', |ux| + k)$

entonces $A = A'$, $x = x'$ y $u = u'$.

□

Por otra parte, la definición de gramáticas lineales SBD coincide con las lineales pares de la definición 4.5. En este caso la relación entre gramáticas lineales SBD y $LR(k)$ no es el mismo que en el de las gramáticas lineales pares tal y como se expresa en el siguiente teorema.

Teorema 5.1. Las clases de gramáticas lineales $LR(k)$ y SBD son incomparables.

Demostración.

Para demostrar la veracidad del enunciado bastará con proporcionar ejemplos de gramáticas SBD que no sean $LR(k)$ para ningún valor de k y viceversa.

Tomemos la gramática definida por las siguientes reglas

$$\begin{array}{llll} S \to aA & D \to Cc \mid Ec & S \to aH & K \to Lc \\ A \to Bg & E \to bF & H \to Ii & L \to Kc \mid Mc \mid Gc \\ B \to Cf & F \to bE \mid bG & I \to bJ & M \to c \\ C \to Dc & G \to d & J \to bI \mid Kh & \end{array}$$

En la anterior gramática pueden obtenerse las siguientes derivaciones

$$S \underset{G}{\overset{*}{\Rightarrow}} ab^s Lc^r hi \underset{G}{\Rightarrow} ab^s Gcc^r hi$$

$$S \underset{G}{\overset{*}{\Rightarrow}} ab^p Fc^l fg \underset{G}{\Rightarrow} ab^p bGc^l fg$$

Tal y como se muestra en las anteriores derivaciones, se puede tomar $u = ab^s$, $v = c^r hi$, $x = Gc$, $u' = ab^p$, $v' = c^l fg$ y $x' = bG$. La anterior gramática es SBD y, para cualquier valor seleccionado de k, podemos encontrar valores para s, p, r y l de forma que $pref(uxv, |ux|+k)$ sea igual a $pref(u'x'v', |ux|+k)$ con $L \neq F$.

Por otra parte, podemos tomar la siguiente gramática $LR(0)$

$$\begin{array}{ll} S \to bA & A \to Bcc \\ B \to bA \mid cC \mid aD & C \to d \\ D \to d & \end{array}$$

Es fácil ver que la anterior gramática no es SBD ya que $C \to d$ y $D \to d$ son producciones de la gramática

□

Por otra parte, el problema de decidir si una gramática lineal es o no SBD es indecidible tal y como se expresa en el siguiente resultado.

Lema 5.1. Sea G una gramática lineal. Es indecidible establecer si G es SBD.

Demostración.

Para demostrar la veracidad del enunciado reduciremos el *Problema de la Correspondencia de Post* (PCP) [55] al problema de establecer si una gramática lineal es SBD.

Tomemos una instancia del PCP definido por las listas $X = \{x_1, x_2, \cdots, x_n\}$ e $Y = \{y_1, y_2, \cdots, y_n\}$, donde $(\forall i\ 1 \leq i \leq n)\ x_i, y_i \in \Sigma^*$. Una solución afirmativa del PCP consiste en una secuencia de índices $j_1, \cdots, j_k$ donde $\forall i\ 1 \leq i \leq k$ $j_i \in \{1, \cdots, n\}$ de forma que se cumple que $x_{j_1} x_{j_2} \cdots x_{j_k} = y_{j_1} y_{j_2} \cdots y_{j_k}$. Por otra parte, una solución negativa indica que la anterior secuencia de índices no existe. Este problema fue demostrado como indecidible [55], es decir no existe ningún algoritmo que resuelva afirmativa o negativamente una instancia arbitraria de PCP.

A partir de las anteriores listas, construimos la gramática lineal $G = (\{S, A, B\}, \Sigma \cup \{1, 2, \cdots, n\}, P, S)$ donde las producciones de P se definen mediante las reglas siguientes

1. $S \to A|B$
2. $\forall i\ 1 \leq i \leq n\ A \to x_i A i | x_i i$
3. $\forall i\ 1 \leq i \leq n\ B \to y_i B i | y_i i$

Se puede demostrar fácilmente que existe una secuencia de índices $j_1, j_2, \cdots, j_k$ de forma que

$x_{j_1} \cdots x_{j_k} = y_{j_1} \cdots y_{j_k}$ sii $A \overset{*}{\Rightarrow} x_{j_1} \cdots x_{j_k} j_k \cdots j_1$ y $B \overset{*}{\Rightarrow} y_{j_1} \cdots y_{j_k} j_k \cdots j_1$ con $x_{j_1} \cdots x_{j_k} = y_{j_1} \cdots y_{j_k}$.

Es decir existe solución en PCP sii G no es SBD.

□

Definición 5.3. Sea G una gramática lineal. Diremos que G presenta *fuerte determinismo estructural hacia atrás* si se cumple que $\forall w \in L(G)$, si $A \overset{*}{\underset{G}{\Rightarrow}} w$ y $B \overset{*}{\underset{G}{\Rightarrow}} w$ entonces $A = B$ y el A-árbol con resultado w es único. □

A las gramáticas con fuerte determinismo estructural hacia atrás las denominaremos gramáticas *SSBD* (*strongly structural backward deterministic*). Se cumple la siguiente propiedad que relaciona las gramáticas SBD y $SSBD$.

Propiedad 5.1. Sea G una gramática lineal. Se cumplen los siguientes enunciados:

1. Si G es $SSBD$ entonces G es SBD.
2. Si G es SBD entonces G no es necesariamente $SSBD$.

Demostración.

La primera parte del enunciado se cumple trivialmente a partir de las definiciones de gramáticas SBD y $SSBD$. Para verificar la segunda parte del enunciado tomaremos la gramática definida por las producciones $S \to aA$; $A \to bB \mid Bc$ y $B \to c \mid b$. La gramática anterior es SBD pero no $SSBD$ ya que las derivaciones $A \Rightarrow bB \Rightarrow bc$ y $A \Rightarrow Bc \Rightarrow bc$ definen árboles de derivación distintos para la cadena bc partiendo del auxiliar A.

□

Por otra parte, al igual que sucedía con las gramáticas SBD, resulta indecidible establecer si una gramática lineal arbitraria es $SSBD$ tal y como se establece en el siguiente resultado.

Propiedad 5.2. Sea $G = (N, \Sigma, P, S)$ una gramática lineal arbitraria. Es indecidible establecer si G es $SSBD$.

Demostración.

A partir de la demostración del lema 5.1, podemos observar que, en la gramática que se obtiene a partir de las listas de la instancia del PCP, establecer si la citada gramática es SBD equivale a establecer si existe más de un S-árbol para alguna cadena del lenguaje. Por lo tanto, se podría resolver el problema PCP a partir de la resolución del enunciado expuesto en la propiedad. □

Obsérvese que, en las anteriores definiciones de gramáticas lineales SBD, $SSBD$ y $LR(k)$ se ha introducido cierto tipo de determinismo. El determinismo en los lenguajes lineales ha sido tratado previamente en trabajos como [2] y [32] donde se aborda el aspecto de complejidad computacional de las tareas de análisis. Existen distintas definiciones de determinismo relativas a los lenguajes lineales y, en el presente trabajo, se tratará con aquellas que se deducen de las definiciones anteriores.

5.2 Distinguibilidad terminal y estructural

El problema de la identificación de los lenguajes lineales presenta múltiples dificultades. Por una parte, la identificación exacta en tiempo polinómico no es posible a partir de un conjunto finito de datos polinómico tal y como se enunció en el problema del conjunto característico [10]. Por otra parte, la identificación en el límite a partir de datos positivos tampoco es posible ya que en la clase de los lenguajes lineales no se cumplen las condiciones de Angluin para la identificación a partir de datos positivos [3]. Ni siquiera la estructura de una gramática lineal puede ser aprendida a partir de ejemplos positivos

estructurales (esqueletos) tal y como se verá en el próximo capítulo. Bajo esta situación, se presentan varias opciones que permiten obtener algoritmos caracterizables de identificación referidos a la clase de los lenguajes lineales. Entre las distintas opciones podemos citar las dos siguientes

1. Identificación en el límite a partir de presentación completa del lenguaje. Este marco de trabajo evita el problema del conjunto característico ya que no se exige una identificación exacta a partir de un conjunto finito de datos. También se evitan las condiciones de Angluin al trabajar con presentación completa.

2. Restricción de la clase de lenguajes obteniendo una subclase de los lenguajes lineales que pueda ser inferida en tiempo polinómico a partir de un conjunto característico.

Es la última aproximación la que adoptaremos en este capítulo. Para ello, definiremos dos propiedades no triviales de las gramáticas lineales que permiten definir una subclase de los lenguajes lineales con el objetivo de plantear algoritmos eficientes de identificación. Las dos características a las que nos referimos son la *distinguibilidad terminal* y el *determinismo estructural*. Informalmente, la distinguibilidad terminal supone que, en una gramática lineal, una situación de indeterminismo se puede resolver mediante la distinción de los símbolos terminales que pueden llegar a derivar los auxiliares involucrados en esa situación. Por otra parte, el determinismo estructural supone la no ambigüedad de las gramáticas lineales ni siquiera de forma parcial, esto es referido a cualquier segmento de una derivación de una cadena del lenguaje.

A partir de la función de símbolos terminales, $ter(\cdot)$, proporcionada en el capítulo 3, podemos formalizar el concepto de *distinguibilidad terminal*. Veamos un ejemplo de conjuntos de símbolos terminales referido a gramáticas lineales.

Ejemplo 5.2. Sea la gramática lineal G definida por las producciones

$$S \to aA \mid bB \mid b;\ \ A \to aA \mid Ba \mid b;\ \ B \to bB \mid b$$

tenemos los siguientes conjuntos

$$ter(L(S,G)) = \{a,b\}$$
$$ter(L(A,G)) = \{a,b\}$$
$$ter(L(B,G)) = \{b\}$$ □

Definición 5.4. Sea $G = (\Sigma, N, P, S)$ una gramática lineal. Diremos que G presenta distinguibilidad terminal si cumple las siguientes condiciones

1. $(\forall A, B, C \in N)$ si $(A \to aB \ \wedge \ A \to aC) \ \vee \ (A \to Ba \ \wedge \ A \to Ca)$ entonces $ter(L(B,G)) \neq ter(L(C,G))$.

2. $(\forall A \in N)\ (\forall x, y \in L(A,G))\ ter(x) = ter(y)$

□

A los lenguajes generados por gramáticas lineales con distinguibilidad terminal los denominaremos *lenguajes lineales distinguibles por símbolos terminales*. La distinguibilidad terminal es una propiedad no trivial de las gramáticas lineales que resulta decidible tal y como se plantea en el siguiente resultado.

Lema 5.2. Sea G una gramática lineal en forma normal. Es decidible establecer si G presenta distinguibilidad terminal.

Demostración.

Partiendo de una gramática $G = (N, \Sigma, P, S)$ lineal en forma normal, para comprobar si presenta distinguibilidad terminal basta con comprobar si cumple las dos condiciones enunciadas en la definición 5.4. Veamos si cumple cada una de esas condiciones por separado.

Para la condición (1) basta con calcular el conjunto de símbolos terminales asociados al lenguaje de cada auxiliar, es decir, calcular $\forall A \in N \ \ ter(L(A,G))$. Este cálculo se reduce al cálculo de símbolos terminales alcanzables desde un auxiliar en una gramática incontextual. Puede consultarse en [33] un algortimo que lo resuelve. Una vez calculados los terminales de cada auxiliar entonces comprobar si se cumple la condición (1) resulta trivial ya que se reduce a comprobar si la intersección de dos conjuntos finitos es distinta o no del vacío.

La condición (2) también se puede comprobar mediante un algoritmo. Para ello supongamos que, dado el auxiliar $A \in N$, definimos $\Sigma_A = ter(L(A)) = \{a_1, \cdots, a_n\}$. Podemos construir expresiones regulares a partir de Σ_A que den cuenta, de forma selectiva, de las cadenas que se pueden formar con subconjuntos de Σ_A. Para formar expresiones regulares vamos a definir en primer lugar para una cadena $x = a_1 \cdots a_n$ las permutaciones de la cadena y lo denotaremos por $perm(x)$. A continuación definiremos la operación $mezcla(L, a_1 \cdots a_n) = La_1La_2L \cdots La_nL$. De esta forma, tomando un subconjunto de símbolos de Σ_A formado por $\{a_1, \cdots, a_p\}$ la expresión regular definida como

$$\sum_{y \in perm(a_1 \cdots a_p)} mezcla(\{a_1, \cdots a_p\}^*, y)$$

denota el conjunto de todas las cadenas formadas por los símbolos $a_1, \cdots, a_p$ donde siempre aparecen todos los símbolos.

Por otra parte, a partir de G podemos definir para todo símbolo auxiliar A la gramática $G_A = (N_A, \Sigma_A, P_A, A)$, donde el conjunto de auxiliares, las producciones y el conjunto de terminales se definen a partir de los símbolos alcanzables desde A y el axioma pasa a ser A. Está demostrado que la intersección entre un lenguaje incontextual y un lenguaje regular proporciona siempre un lenguaje incontextual [33]. Más aún, se puede construir una gramática incontextual que genere el resultado de la intersección tomando una gramática incontextual y una expresión regular. También ha sido demostrado como decidible establecer si el lenguaje generado por una gramática incontextual es o no vacío [33]. A partir de los anteriores resultados el algoritmo de la figura 5.2 comprueba si en una gramática lineal se cumple la condición (2) o no.

□

Entrada : Una gramática lineal $G = (N, \Sigma, P, S)$ en forma normal.

Salida : *Si* si G cumple la condición (2) de distinguibilidad terminal.
No si G no cumple la condición (2) de distinguibilidad terminal.

Método :

(1) $\forall A \in N$ se define $G_A = (N_A, \Sigma_A, P_A, A)$, con $card(\Sigma_A) = n$.

(2) $\forall M_i \in \mathcal{P}(\Sigma_A) : M_i = \{a_1, a_2, \cdots, a_{j_i}\}$ se define

$$r_i = \sum_{y \in perm(a_1 \cdots a_{j_i})} mezcla(\{a_1, \cdots, a_{j_i}\}^*, y)$$

$$L(G_A i) = L(G_A) \cap r_i.$$

(3) Si existe más de un lenguaje $L(G_A i) \neq \emptyset$ el algoritmo da como salida *No*.

(4) En otro caso da como salida *Si*.

finMétodo

Figura 5.2: Método de comprobación de la condición (2) de distinguibilidad terminal en una gramática lineal.

5.3 Definición de los lenguajes $\mathcal{TSDL}$

A partir de las anteriores definiciones de distinguibilidad terminal y fuerte determinismo estructural hacia atrás se puede establecer una nueva subclase de lenguajes lineales. Dado que las dos propiedades no son triviales para los lenguajes lineales, existen algunos lenguajes que las poseen y otros que no. Incluso el estado de no trivialidad de la distinguibilidad terminal en lenguajes regulares también se mantiene, por lo que la nueva clase es incomparable en relación con los lenguajes lineales pares, con los lenguajes regulares e incluso con los lenguajes finitos.

Definición 5.5. Sea G una gramática lineal en forma normal. Diremos que G es *distinguible por símbolos terminales y estructura* (TSDL, *Terminal and Structural Distinguishable Linear*) si G presenta distinguibilidad terminal y fuerte determinismo estructural hacia atrás. □

Un lenguaje es TSDL si existe una gramática TSDL que lo genere. A la clase de los lenguajes TSDL la denotaremos por $\mathcal{TSDL}$.

Podemos dar los siguientes ejemplos de lenguajes TSDL y lenguajes que no lo son.

Ejemplo 5.3. Los siguientes lenguajes son TSDL:

$\{a\}$,
$\{aa^nbb^m : n, m \geq 1\}$,
$\{a^nb^n : n \geq 1\}$ y
$\{a^nb^{2n} : n \geq 1\}$

dado que podemos obtener gramáticas TSDL que los generen.

□

Ejemplo 5.4. Los siguientes lenguajes no son TSDL:

$\{a, b\}$,
$\{aab\} \cup b^*$,
$\{a^nb^n : n \geq 1\} \cup \{b^nc^n : n \geq 1\}$ y
$\{a^nb^{2n} : n \geq 1\} \cup \{b^nc^{2n} : n \geq 1\}$

Obsérvese que proponemos ejemplos de lenguajes finitos, regulares, lineales pares y lineales que no son TSDL. En tales lenguajes existen cadenas x e y de forma que $x, y \in L(S, G)$ para cualquier gramática G y $ter(x) \neq ter(y)$.

□

A la luz de los anteriores ejemplos, podemos establecer que los lenguajes TSDL forman una clase de lenguages que intersecta con las clases de los

lenguajes finitos, regulares, lineales pares y lineales, pero no contienen propiamente a ninguna de ellas. Los ejemplos anteriores son una prueba de que no se cumple la inclusión propia. En la figura 5.3 se muestra la relación entre los lenguajes TSDL y las clases de lenguajes referidas anteriormente.

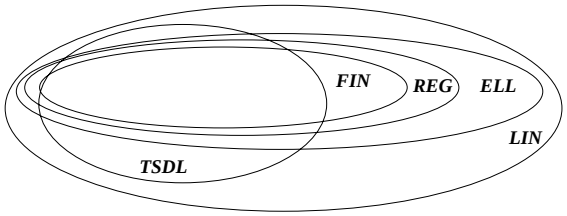


Figura 5.3: La clase $\mathcal{TSDL}$ en relación con otras clases de lenguajes formales.

Una relación de índice finito entre estructuras gramaticales

Una vez definidas las gramáticas TSDL, vamos a plantear una relación entre los nodos internos de los esqueletos que producen las mismas. Esta relación a su vez propiciará la definición de un algoritmo de inferencia de gramáticas $TSDL$ a partir de presentación positiva estructural, esto es esqueletos de cadenas del lenguaje.

En primer lugar introduciremos algunos conceptos referidos a la información estructural que define un nodo interno de un esqueleto de la gramática. Para ello, dado un esqueleto A, denotaremos por $resultado(A)$ al resultado del esqueleto que es la cadena obtenida al recorrer sus hojas e ir encadenando los símbolos de izquierda a derecha.

Definición 5.6. Sea $G = (\Sigma, N, P, S)$ una gramática lineal, $x \in L(G)$ y sk un esqueleto consistente con la gramática de forma que $resultado(sk) = x$. Para cada nodo interno n de sk, la *frontera* de n es el resultado del subesqueleto formado al tomar como raiz a n, lo denotaremos por $frontier(n)$. La *cabeza* de n se define como la subcadena izquierda del resultado de sk hasta llegar al nodo n, la denotaremos por $head(n)$. La *cola* de n se define como la subcadena derecha del resultado de sk a partir de la frontera de n, la denotaremos por $tail(n)$. La *información contextual* de n quedará definida por la tupla $(head(n), frontier(n), tail(n))$. □

En la figura 5.4 se muestra un esqueleto de una gramática lineal, se indica uno de sus nodos internos, n, y se especifica su *información contextual*. Obsérvese que para cualquier esqueleto sk y para cualquiera de sus nodos internos n se cumple que $resultado(sk) = head(n) \cdot frontier(n) \cdot tail(n)$.

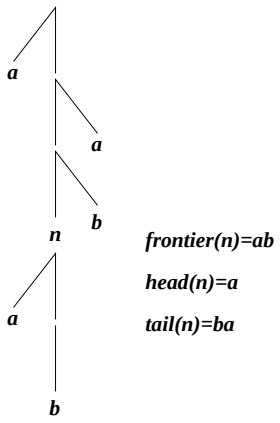


Figura 5.4: Un esqueleto lineal y la información contextual de un nodo interno.

Definición 5.7. Sea $G = (\Sigma, N, P, S)$ una gramática lineal, $x \in L(G)$ y sk un esqueleto consistente con la gramática de forma que $resultado(sk) = x$. Para cada nodo interno n de sk, la *cabeza estructural* de n, denotada por $headstr(n)$, y la *cola estructural* de n, denotada por $tailstr(n)$, se definen como el resultado de incluir el símbolo * en $head(n)$ y $tail(n)$ respectivamente para reemplazar la ausencia de hijos por la derecha o por la izquierda. El símbolo $*$ denota la ausencia de hijo en los nodos anteriores a n en sk. La *información estructural* de n quedará definida por la tupla $(headstr(n), frontier(n), tailstr(n))$. □

En la figura 5.5 se muestra un esqueleto de una gramática lineal, se indica uno de sus nodos internos, n, y se especifica su *información estructural*.

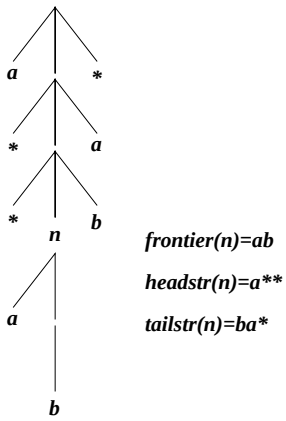


Figura 5.5: Un esqueleto lineal y la información estructural de un nodo interno.

A partir de la definición de información estructural, podemos establecer una relación de índice finito entre los nodos internos de los esqueletos de forma que se tome en consideración la distinguibilidad terminal y el fuerte determinismo estructural hacia atrás. Para ello proponemos la siguiente definición.

Definición 5.8. Sea $G = (\Sigma, N, P, S)$ una gramática TSDL y sea $SK(G)$ el conjunto de esqueletos de G. Se define la relación $\sim$ entre los nodos internos de los esqueletos de la siguiente forma

Si n y m son nodos internos de esqueletos de SK entonces $n \sim m$ si al menos una de las dos siguientes condiciones se cumple

1. $frontier(n) = frontier(m)$
2. $(headstr(n) = headstr(m)) \ \wedge \ (tailstr(n) = tailstr(m)) \ \wedge$ $(ter(frontier(n)) = ter(frontier(m)))$

Denotaremos por $\equiv$ a la clausura reflexiva y transitiva de $\sim$. □

Dada una gramática lineal G y un conjunto de esqueletos SK consistente con la gramática, denotaremos por N_{SK} al conjunto de nodos internos de los esqueletos de SK. A partir de esta definición se obtiene la siguiente propiedad.

Propiedad 5.3. Sea $G = (\Sigma, N, P, S)$ una gramática TSDL y SK el conjunto de esqueletos de la gramática, entonces $\equiv$ es una relación de índice finito y $card(N_{SK}/\equiv) = card(N)$.

Demostración.

Basta con demostrar que para todo par de nodos internos $n_i, n_j \in N_{SK}$ obtenidos a partir del mismo auxiliar de la gramática se cumple que $n_i \equiv n_j$.

Para ello, consideremos las posibles situaciones en que se pueden encontrar n_i y n_j.

1. $n_i \sim n_j$ debido a que se cumple una de las dos condiciones de la definición 5.8. En este caso, de forma evidente, $n_i \equiv n_j$.
2. $n_i \not\sim n_j$. En este caso existirá un nodo interno n_k de forma que $frontier(n_i) = frontier(n_k)$ y $headstr(n_k) = headstr(n_j)$, $tailstr(n_k) = tailstr(n_j)$ y $ter(frontier(n_k)) = ter(frontier(n_j))$. Por lo tanto, $n_i \sim n_k$ y $n_k \sim n_j$ por lo que $n_i \equiv n_j$.

Puesto que los nodos internos de $SK(G)$ son todos inducidos por los auxiliares de la gramática y, dado que existen $card(N)$ símbolos auxiliares entonces existirán en $N_{SK}/\equiv$ un total de $card(N)$ clases de equivalencia.

Obsérvese que $card(N_{SK}/\equiv)$ no puede ser estrictamente menor que $card(N)$. Para que sucediera lo contrario deberían haber dos auxiliares distintos en la gramática cuyos nodos internos en los esqueletos se relacionaran. Supongamos que los citados nodos fueran n_i y n_j. Obviamente, $n_i \not\sim n_j$ ya que la gramática es $TSDL$ y se vulneraría la condición 1 de la definición 5.4. Por otra parte, no puede existir un nodo n_k tal que $n_i \sim n_k$ y $n_k \sim n_j$, ya que ese nodo debería estar asociado al símbolo auxiliar de n_i o de n_j y volveríamos a replicar la argumentación anterior.

□

5.4 Identificación en el límite de lenguajes $\mathcal{TSDL}$

A partir de la definición de la relación $\equiv$ proponemos un algoritmo de inferencia que se basa en el cálculo de las clases de equivalencia de la relación $\equiv$ y la posterior obtención de una gramática TSDL consistente con los datos. El algoritmo queda establecido en la figura 5.6. Daremos un ejemplo de su ejecución, probaremos su convergencia en el límite a la gramática objetivo y, por último, estableceremos la complejidad temporal y espacial del algoritmo.

A continuación proporcionamos el siguiente ejemplo de aplicación del algoritmo.

Ejemplo 5.5. Tomemos el lenguaje $\{a^i b^{2i} : i \geq 1\}$. El conjunto de esqueletos proporcionados al algoritmo como información de entrada se muestra en la figura 5.7 donde aparecen ya enumerados sus nodos internos. La información estructural de los nodos internos se muestra en la Tabla 5.1.

	headstr	*tailstr*	*frontier*	*ter(frontier)*
n_{11}	λ	λ	$aabbbb$	$\{a, b\}$
n_{12}	a	$*$	$abbbb$	$\{a, b\}$
n_{13}	$a*$	$b*$	$abbb$	$\{a, b\}$
n_{14}	$a**$	$bb*$	abb	$\{a, b\}$
n_{15}	$a**a$	$*bb*$	bb	$\{b\}$
n_{16}	$a**a*$	$b*bb*$	b	$\{b\}$
n_{21}	λ	λ	abb	$\{a, b\}$
n_{22}	a	$*$	bb	$\{b\}$
n_{23}	$a*$	$b*$	b	$\{b\}$

Tabla 5.1. Información estructural asociada a cada nodo interno de los esqueletos de entrada.

Las clases de equivalencia inducidas por $\equiv$ son $SK_1 = \{n_{11}, n_{21}, n_{14}\}$, $SK_2 = \{n_{16}, n_{23}\}$, $SK_3 = \{n_{22}, n_{15}\}$, $SK_4 = \{n_{12}\}$ y $SK_5 = \{n_{13}\}$

Entrada : Un conjunto de esqueletos $SK = \{sk_1, \cdots, sk_n\}$ donde cada esqueleto sk_i es lineal en forma normal y no contiene la etiqueta λ.

Salida : Una gramática TSDL G en forma normal donde $(\forall sk_i \in SK)\ \ resultado(sk_i) \in L(G)$.

Método :

(1) Enumerar los nodos internos de cada esqueleto sk_i. El nodo interno n_{ij} es el j-ésimo nodo interno del i-ésimo esqueleto desde la raíz hasta las hojas.

(2) Calcular la *información estructural* de cada nodo interno $(headstr(n_{ij}), frontier(n_{ij}), tailstr(n_{ij}))$.

(3) Calcular la relación $\equiv$ entre los nodos internos.

(4) Enumerar cada clase de equivalencia $SK_1, SK_2 \cdots SK_l$. La clase de equivalencia que contiene las raíces de los esqueletos se etiqueta con S.

(5) Sustituir cada nodo interno por su clase de equivalencia (cada esqueleto es transformado en un árbol de derivación).

(6) Obtener una gramática G a partir de los árboles de derivación. El axioma es S.

finMétodo

Figura 5.6: Método de inferencia de los lenguajes TSDL.

A las anteriores clases les podemos asociar los siguientes símbolos auxiliares $SK_1 = S, SK_4 = A, SK_5 = B, SK_3 = C$ and $SK_2 = D$. Así, la gramática de salida es la siguiente.

$$S \to aA \mid aC;\ \ A \to Bb;\ \ B \to Sb;\ \ C \to Db\ \ D \to b$$

Puede observarse que la gramática obtenida genera el lenguaje de partida $\{a^i b^{2i} : i \geq 1\}$.

□

Una vez comprobado el funcionamiento del algoritmo, vamos a proceder a demostrar su corrección, esto es que converge a la gramática objetivo si se le proporciona suficiente información. En este caso, el protocolo de presentación de la información es *texto estructural*. El objetivo es, por lo tanto, demostrar

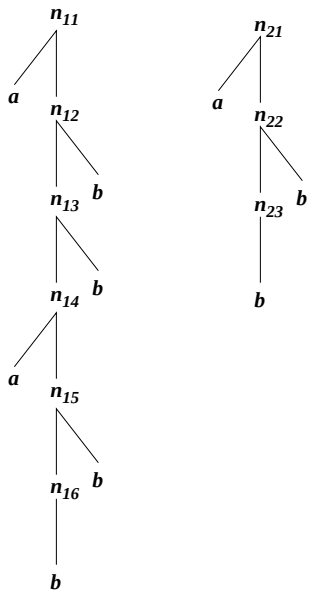


Figura 5.7: Información de entrada y enumeración de los nodos internos.

que el algoritmo propuesto identifica cualquier gramática TSDL en el límite. La demostración la basaremos en la existencia de un conjunto característico, en este caso estructural, que se asocia a cada gramática TSDL. Daremos un algoritmo de construcción del citado conjunto mediante el siguiente lema.

Lema 5.3. Sea G una gramática TSDL. Existe un conjunto finito de esqueletos consistentes con la gramática G tal que si se proporciona como entrada al algoritmo de inferencia propuesto, éste obtiene como salida una gramática G' isomorfa a G.

Demostración.

Sea $G = (N, \Sigma, P, S)$. Para cada símbolo auxiliar $A \in N$, se puede calcular su información estructural mínima como sigue:

- Combinar cada A-producción de la gramática tomando las diferentes derivaciones más cortas en las que A aparece de forma que $S \overset{*}{\underset{G}{\Rightarrow}} \alpha A \beta \overset{*}{\underset{G}{\Rightarrow}} w$ con $w \in \Sigma^*$.

- Construir los árboles de derivación de acuerdo con las derivaciones anteriores.

- Calcular $headstr(A), tailstr(A)$ y $frontier(A)$ en cada árbol tal y como se realiza en los esqueletos.

G es SSBD dado que G es TSDL. Por lo tanto, los esqueletos inducidos por los árboles de derivación construidos anteriormente tienen la propiedad de que si los nodos internos n_i y n_j se etiquetan con diferentes símbolos auxiliares entonces se cumple que

1. $frontier(n_i) \neq frontier(n_j)$, y
2. $headstr(n_i) \neq headstr(n_j) \quad \vee \; tailstr(n_i) \neq tailstr(n_j) \; \vee$ $ter(frontier(n_i)) \neq ter(frontier(n_j))$

Si $S \underset{G}{\Rightarrow} \alpha_1 A_1 \beta_1 \underset{G}{\Rightarrow} \cdots \underset{G}{\Rightarrow} \alpha_n A_n \beta_n \underset{G}{\Rightarrow} \alpha A \beta \overset{*}{\underset{G}{\Rightarrow}} \alpha w \beta$ y $S \underset{G}{\Rightarrow} \alpha_1 A_1 \beta_1 \underset{G}{\Rightarrow} \cdots \underset{G}{\Rightarrow} \alpha_n A_n \beta_n \underset{G}{\Rightarrow} \alpha B \beta \overset{*}{\underset{G}{\Rightarrow}} \alpha w\prime \beta$ son derivaciones con $A \neq B$, entonces $ter(w) \neq ter(w\prime)$ y $A \not\equiv B$.

De esta forma, el algoritmo distingue correctamente dos nodos internos asociados con dos símbolos auxiliares distintos.

Por otra parte, suponiendo que, en el conjunto de esqueletos, dos nodos internos diferentes n_i y n_j, sean etiquetados con el mismo símbolo auxiliar, entonces una de las dos siguientes condiciones debe cumplirse:

1. $frontier(n_i) = frontier(n_j)$. Esta condición se deduce del hecho de que se han seleccionado las derivaciones más cortas para cada símbolo auxiliar.
2. $\exists n_k : n_i \sim n_k \;\wedge\; n_j \sim n_k$, por lo tanto $n_i \equiv n_j$. Este hecho se deduce de haber obtenido las distintas combinaciones de las producciones de cada símbolo auxiliar en, al menos, una de las derivaciones.

A partir de las anteriores condiciones podemos afirmar que el algoritmo relaciona correctamente aquellos nodos internos que denotan el mismo símbolo auxiliar.

Podemos concluir que si los conjuntos de esqueletos definidos para cada símbolo auxiliar se proporcionan como entrada al algoritmo de inferencia propuesto entonces cada símbolo auxiliar de la gramática estará representado y, por lo tanto, el algoritmo los distinguirá correctamente.

□

Como consecuencia de la anterior demostración podemos formalizar la convergencia del algoritmo mediante el siguiente resultado.

Teorema 5.2. El algoritmo propuesto identifica en el límite cualquier lenguaje TSDL si se le proporciona como entrada información estructural.

Demostración.

La demostración es trivial a partir del resultado demostrado anteriormente. En el criterio de identificación en el límite se garantiza la presentación de cada esqueleto posible de la gramática. Por lo tanto, a partir de que se complete el conjunto establecido en el lema 5.3, el algoritmo identificará la gramática generadora de la información. Es decir, el algoritmo proporcionará como salida una gramática isomorfa a la gramática objetivo.

El algoritmo, no cambia de hipótesis al añadir nuevos ejemplos al conjunto definido en el lema 5.3. En ese caso, las clases de equivalencia calculadas no varían ya que se mantiene información estructural equivalente.

□

Para terminar con el estudio de las propiedades del algoritmo vamos a proceder al análisis de su complejidad temporal y espacial. Para ello supongamos que el lenguaje objetivo a identificar se define sobre un alfabeto Σ, con $card(\Sigma) = n$. El tamaño de un conjunto de esqueletos de entrada SK se define como el número de nodos internos que contiene ya que, en las gramáticas lineales en forma normal, el número de nodos internos de un esqueleto coincide con la longitud de su resultado. Por lo tanto, podemos establecer la siguiente expresión que define la talla del conjunto SK

$$|SK| = \sum_{sk_i \in SK} |resultado(sk_i)| = m$$

Tomaremos k como la longitud máxima de los resultados de los esqueletos de entrada.

Podemos establecer la complejidad temporal del algoritmo analizando cada una de sus operaciones

1. **Acción 1**. La enumeración de los nodos internos se realiza leyendo los esqueletos de entrada una única vez. Así, la complejidad temporal en este caso es $\mathcal{O}(m)$.

2. **Acción 2**. Como en el caso anterior, la información concerniente a la información estructural de cada nodo interno correspondiente a $headstr(\cdot)$

y $tailstr(\cdot)$ puede establecerse al leer los esqueletos una sola vez. Posteriormente, el calculo del resultado de cada nodo interno puede establecerse accediendo de nuevo a cada nodo interno. Luego, en este caso, la complejidad temporal se establece como $\mathcal{O}(m)$.

3. **Acción 3**. La relación $\equiv$ se establece mediante la comparación de la información estructural de cada nodo interno. Por lo tanto se realizarán un máximo de $m \times m$ comparaciones. En cada comparación intervienen las cadenas $tailstr(\cdot)$, $headstr(\cdot)$ y $frontier(\cdot)$ que tienen una longitud máxima k. Por último, $ter(frontier(\cdot))$ tiene un máximo de n símbolos por lo que la complejidad total será $\mathcal{O}(m^2(k+n))$.

4. **Acción 4**. Como en el caso 1, la complejidad temporal será $\mathcal{O}(m)$ dado que el número de clases de equivalencia es menor o igual que m.

5. **Acción 5**. De nuevo, como en el caso anterior, la complejidad es $\mathcal{O}(m)$.

6. **Acción 6**. Como en los casos 1, 2, 4 y 5, la complejidad temporal es $\mathcal{O}(m)$.

La complejidad espacial toma en consideración el espacio necesario para almacenar la información estructural de cada nodo interno. Sea l el número de esqueletos de entrada. El espacio necesario para almacenar la información de un esqueleto sk será

$$\sum_{i=1}^{|resultado(sk)|} i$$

Este espacio será en total $\mathcal{O}(k^2)$. Así el espacio necesario para todo el conjunto de esqueletos es $\mathcal{O}(k^2 l)$.

Podemos formalizar el anterior análisis mediante el siguiente resultado.

Teorema 5.3. Sea SK un conjunto de esqueletos proporcionados como entrada al algoritmo de inferencia y definidos sobre un alfabeto con n símbolos. Sea m el tamaño de SK, k el tamaño máximo de las longitudes de los resultados de los esqueletos de SK y l el número de esqueletos en SK. El algoritmo propuesto tiene complejidad temporal $\mathcal{O}(m^2(k+n))$ y complejidad espacial $\mathcal{O}(k^2 l)$. □

5.5 Concatenación de lenguajes $\mathcal{TSDL}$

El algoritmo propuesto para la inferencia de lenguajes TSDL puede ser fácilmente adaptado para trabajar con concatenaciones de estos. De esta forma, se puede ampliar la clases de lenguajes suceptibles de ser inferidos con la anterior técnica. Así, en este apartado, el resultado principal será la identificación de una subclase de lenguajes incontextuales no lineales ya que esta clase no es cerrada bajo concatenación [29].

Los lenguajes definidos a partir de la concatenación de un número finito de lenguajes TSDL forman una clase de lenguajes propiamente incluida en la clase de los lenguajes *metalineales* [29] que, a su vez, se define como el resultado de la aplicación un número finito de veces de las operaciones de concatenación y unión sobre lenguajes lineales. Definiremos esta clase de lenguajes mediante una caracterización gramatical como sigue.

Definición 5.9. Sean $G_1 = (N_1, \Sigma_1, P_1, S_1), \cdots, G_n = (N_n, \Sigma_n, P_n, S_n)$ gramáticas $TSDL$ donde $N_i \cap N_j = \emptyset$ para cada $i \neq j$. Entonces la gramática $G = (N, \Sigma, P, S)$ donde $N = N_1 \cup \cdots \cup N_n \cup \{S\}$, $\Sigma = \Sigma_1 \cup \cdots \cup \Sigma_n$, $P = P_1 \cup \cdots \cup P_n \cup \{S \to S_1 S_2 \cdots S_n\}$ y $S \notin N_1 \cup \cdots \cup N_n$ es CTSDL. □

Un lenguaje L es CTSDL si se puede generar mediante una gramática CTSDL. A la clase de lenguajes CTSDL la denotaremos como $\mathcal{CTSDL}$. La relación entre la clase $\mathcal{CTSDL}$ y otras clases de lenguajes se muestra en la figura 5.8.

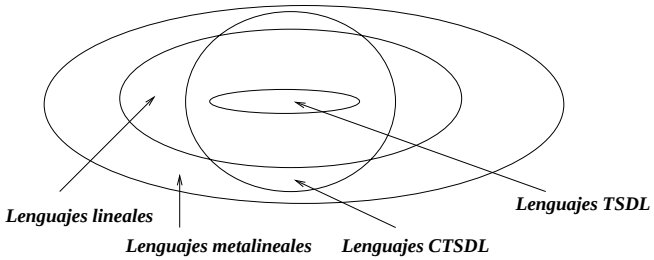


Figura 5.8: La clase $\mathcal{CTSDL}$ en relación con otras clases de lenguajes.

Dado un esqueleto CTSDL sk denotaremos por $aridad(sk)$ el número de hijos de su raiz. Es obvio que en una gramática donde $S \to S_1 S_2 \cdots S_n \in P$, entonces para cualquier esqueleto sk obtenido en la misma se cumple que $aridad(sk) = n$.

Con el fin de proporcionar un algoritmo de inferencia para la clase de los lenguajes CTSDL, podemos reducir la inferencia de los mismos a la inferencia

de los lenguajes TSDL. Es fácil de ver, desde el punto de vista estructural, que, dado un esqueleto CTSDL, los subesqueletos formados a partir de los hijos de la raiz son todos esqueletos TSDL. Así, la idea intuitiva es descomponer cada esqueleto CTSDL en un conjunto de esqueletos TSDL y tratar estos por separado. Posteriormente, se puede realizar la concatenación inicial de los mismos para obtener gramáticas CTSDL. Proporcionamos la siguiente definición que formaliza esta idea.

Definición 5.10. Sea G una gramática CTSDL y sk un esqueleto proporcionado por la gramática G. Definiremos sk^l como el conjunto $\{sk_1, \cdots, sk_n\}$, donde sk_i es el subesqueleto del i-ésimo hijo de la raiz. Se cumple que

$$resultado(sk) = resultado(sk_1) \cdots resultado(sk_n)$$

□

En la figura 5.9, se muestra un ejemplo de un esqueleto sk y su correspondiente conjunto sk^l.

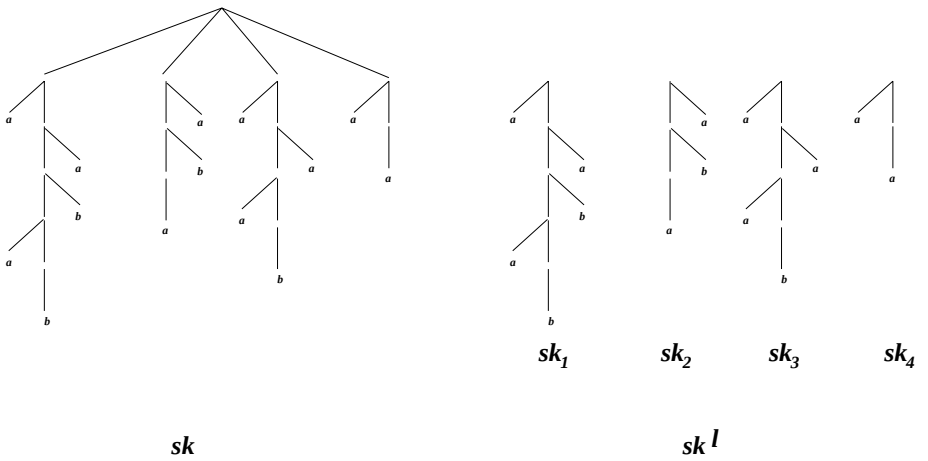


Figura 5.9: Un esqueleto CTSDL sk con su conjunto sk^l.

A partir de la descomposición de esqueletos propuesta, podemos utilizar el algoritmo de inferencia de lenguajes TSDL para plantear una estrategia de inferencia de los lenguajes CTSDL. Así, en la figura 5.10 proponemos un algoritmo de identificación de lenguajes CTSDL.

La convergencia del algoritmo a la gramática objetivo y su complejidad se pueden deducir fácilmente a partir de la reducción propuesta para inferir lenguajes CTSDL utilizando el método de inferencia de lenguajes TSDL.

Entrada : Un conjunto de esqueletos CTSDL $SK = \{sk_1, \cdots, sk_n\}$ donde $(\forall 1 \leq i \leq n)\ aridad(sk_i) = k$. Ningún esqueleto contiene a la etiqueta λ.

Salida : Una gramática CTSDL G, donde $\forall sk_i \in SK \quad resultado(sk_i) \in L(G)$.

Método :

(1) $\forall sk_i \in SK$ obtener el conjunto $sk_i^l = \{sk_{i1}, \cdots, sk_{ik}\}$

(2) $\forall j,\ 1 \leq j \leq k,\ SK(j) = \{sk_{ij} \in sk_i^l \mid 1 \leq i \leq n\}$

(3) $\forall SK(j)$ ejecutar el algoritmo para inferir lenguajes TSDL y obtener una gramática G_j

(4) Construir una gramática $CTSDL\ G$ a partir de las gramáticas $G_1, \cdots, G_k$ tal y como se especifica en la definición 5.9

finMétodo

Figura 5.10: Método de inferencia para la clase $\mathcal{CTSDL}$.

5.6 Reversibilidad en lenguajes lineales

Nos proponemos en esta sección y la siguiente abordar la identificación de nuevas clases de lenguajes aplicando las reducciones y algoritmos que hemos propuesto hasta ahora. En primer lugar veremos cómo a partir de la información estructural que definen las gramáticas lineales podemos dar cuenta de subclases de lenguajes que presentan reversibilidad en el mismo sentido que en los lenguajes regulares. Esta línea de actuación es similar a la que en su momento siguieron Koshiba, Mäkinen y Takada referida a los lenguajes lineales pares [40]. De hecho, el planteamiento que haremos será transformar la información estructural que proporcionan las gramáticas lineales en cadenas de lenguajes lineales pares que se han caracterizado por presentar reversibilidad en el trabajo citado anteriormente.

Conceptos básicos acerca de lenguajes reversibles

En primer lugar vamos a proporcionar conceptos básicos sobre reversibilidad relativa a los lenguajes regulares. Seguiremos fundamentalmente el trabajo de

Angluin [4].

Definición 5.11. Un autómata A diremos que es 0-reversible si y sólo si A y A^{inv} son deterministas. □

A continuación generalizaremos el concepto de k-reversibilidad, tomando las definiciones proporcionadas por Angluin [4]. Dado un autómata $A = (Q, \Sigma, \delta, I, F)$ y un estado $q \in Q$, fijado un valor $k \geq 0$, diremos que la cadena u, con $|u| = k$, es un k-sucesor (k-predecesor) del estado q si $\delta(q, u) \neq \emptyset$ ($\delta^{inv}(q, u^{inv}) \neq \emptyset$). El autómata A diremos que es *determinista con anticipación* k si y sólo si para todo trío de estados q_1, q_2 y q_3 y para todo símbolo a, si $q_1, q_2 \in \delta(q_3, a)$, o $q_1, q_2 \in I$, entonces no existe ninguna cadena que sea un k-sucesor de q_1 y de q_2.

Definición 5.12. Un autómata A diremos que es k-reversible si y sólo si A es determinista y A^{inv} es determinista con anticipación k. □

Un lenguaje L es k-reversible si existe un autómata finito k-reversible que acepta L.

A partir de la definición de buenos finales de una cadena en un lenguaje se puede proporcionar una caracterización de los lenguajes k-reversibles tal y como sigue.

Propiedad 5.4. Un lenguaje L diremos que es k-reversible, para cualquier valor $k \geq 0$, si se cumple que $\forall u_1, u_2, v \in \Sigma^*$ con $|v| = k$ si $(u_1 v)^{-1} L \cap (u_2 v)^{-1} L \neq \emptyset$ entonces $(u_1 v)^{-1} L = (u_2 v)^{-1} L$. □

Denotaremos por $\mathcal{REV}(k)$ a la familia de los lenguajes k-reversibles y por $\mathcal{REV}$ a la familia de lenguajes reversibles en general. Es decir $\mathcal{REV}$ es la unión (infinita) de las familias $\mathcal{REV}(k)$ para cualquier valor $k \geq 0$. Son evidentes las siguientes inclusiones de acuerdo con la jerarquía de Chomsky

$$\forall k \geq 0 \;\; \mathcal{REV}(k) \subset \mathcal{REV} \subset \mathcal{REG}$$

Teorema 5.4. (Anluin [4]) Sea k un entero no negativo ($k \geq 0$). Se cumplen las siguientes afirmaciones:

1. $\mathcal{REV}(k) \subset \mathcal{REV}(k+1)$.
2. $\mathcal{REV}(k)$ es cerrada bajo la intersección dos a dos.
3. $\mathcal{REV}$ no es cerrada ni bajo unión ni complementario.
4. $\mathcal{REV}$ no es cerrada bajo concatenación.

5. $\mathcal{REV}$ no es cerrada bajo clausura de Kleene.

6. $\mathcal{REV}(k)$ es cerrada bajo reverso.

□

Reversibilidad estructural

En primer lugar, estableceremos una reducción de los lenguajes lineales a los lenguajes lineales pares. Obsérvese que, en la definición 5.7, ya se ha introducido el tipo de transformación necesaria para obtener a partir de una gramática lineal otra gramática lineal par asociada a ella que presenta una serie de características que mostraremos más tarde. Formalizaremos este tipo de reducción mediante la siguiente definición.

Definición 5.13. Sea $G = (\Sigma, N, P, S)$ una gramática lineal en forma normal. Definiremos la gramática lineal par *estructuralmente asociada* a G como la gramática $G_{str} = (\Sigma \cup \{*\}, N, P_{str}, S)$, donde $* \notin \Sigma$ y P_{str} se define como sigue

1. Si $A \to aB \in P$ entonces $A \to aB* \in P_{str}$

2. Si $A \to Ba \in P$ entonces $A \to *Ba \in P_{str}$

3. Si $A \to a \in P$ entonces $A \to a \in P_{str}$

□

Obsérvese que existe un homomorfismo $h : \Sigma \cup \{*\} \longrightarrow \Sigma^*$ definido como $\forall a \in \Sigma\ h(a) = a$ y $h(*) = \lambda$ de forma que, de acuerdo con la anterior definición, se cumple que $h(L(G_{str})) = L(G)$. A partir de la gramática G_{str} y del homomorfismo h podemos definir el concepto de reversibilidad estructural en los lenguajes lineales.

Definición 5.14. Sea G una gramática lineal en forma normal y G_{str} su gramática lineal par estructuralmente asociada. Diremos que G es k-reversible estructuralmente sii G_{str} es $LRS(k)$ de acuerdo con la definición 4.4. □

Un lenguaje lineal L diremos que es k-reversible estructuralmente si existe una gramática en formal normal k-reversible estructuralmente que lo genere. A la clase de lenguajes lineales k-reversibles estructuralmente la denotaremos mediante $\mathcal{LIN}_{revstr}(k)$.

Dado que nuestro objetivo es utilizar muestra estructural para formalizar un algoritmo de inferencia, procederemos a establecer la transformación que

Entrada: Un conjunto finito de esqueletos SK
Método:
$S = \emptyset$
$\forall sk \in SK$
$S = S \cup \{cadenastr(sk)\}$
$\mathcal{H} = A_{LRS(k)}(S)$
Salida: $h(\mathcal{H})$

Figura 5.11: Un método de inferencia para lenguajes de $\mathcal{LIN}_{revstr}(k)$

nos permita obtener, a partir de esqueletos de una gramática lineal, cadenas de su gramática lineal par estructuralmente asociada. Para ello utilizaremos la siguiente transformación.

Definición 5.15. Sea sk un esqueleto generado por la gramática lineal en forma normal G. Seleccionemos el único nodo interno de sk que contiene un solo hijo (que será terminal) y que denotaremos por n_c. Definimos $cadenastr(sk) = headstr(n_c) \cdot frontier(n_c) \cdot tailstr(n_c)$. □

Es fácil ver que si sk es generado por una gramática lineal en forma normal G entonces $cadenastr(sk)$ pertenece al lenguaje de la gramática lineal par estructuralmente asociada G_{str}. A modo de ejemplo, si consideramos el esqueleto sk de la figura 5.5 entonces $cadenastr(sk) = a**ab*ba*$.

Podemos ahora establecer un algoritmo de inferencia para los lenguajes de la clase $\mathcal{LIN}_{revstr}(k)$ tomando como fuente de información esqueletos de la gramática objetivo. El algoritmo se basa en la transformación de los esqueletos de entrada en cadenas estructurales y en la posterior obtención de una gramática lineal par asociada a los mismos. Denotaremos por $A_{LRS(k)}$ al algoritmo de inferencia de lenguajes lineales pares establecido en [40]. El esquema de aprendizaje se muestra en la figura 5.11.

En el método planteado, utilizamos la transformación $cadenastr(\cdot)$ y el homomorfismo h definidos anteriormente. Obsérvese que, finalmente, el método proporciona como salida una gramática lineal a partir de la transformación $h(\mathcal{H})$ que consiste en aplicar el homomorfismo h sobre la gramática $\mathcal{H}$ mediante métodos bien conocidos [33].

La identificación en el límite de los lenguajes de la clase $\mathcal{LIN}_{revstr}(k)$ se deduce a partir de la reducción planteada y del resultado de convergencia de

la clase $\mathcal{LRS}(k)$ demostrada en [40]. Formalizaremos este resultado mediante el siguiente teorema.

Teorema 5.5. La clase de lenguajes $\mathcal{LIN}_{revstr}(k)$ es identificable en el límite a partir de texto estructural.

Demostración.

Tal y como se ha formulado el método de inferencia, la identificación en el límite de la clase de lenguajes bajo estudio queda garantizada a partir de los siguientes hechos

1. La clase de lenguajes $\mathcal{LRS}(k)$ es identificable en el límite a partir de texto [40].

2. Si la gramática que actúa como fuente de información es $\mathcal{LIN}_{revstr}(k)$ entonces las cadenas obtenidas mediante $cadenastr(\cdot)$ pertenecen a su gramática lineal par estructuralmente asociada que, a su vez, es $LRS(k)$.

3. La transformación h obtiene la gramática lineal que, originalmente, actuó como fuente de información si la cantidad de información de entrada es suficiente para garantizar la convergencia.

A partir de las observaciones realizadas, trivialmente se comprueba la veracidad del enunciado propuesto.

□

5.7 Explorabilidad local en lenguajes lineales

Al igual que hemos planteado en el anterior capítulo distintas formas de explorabilidad local relativa a los lenguajes lineales pares, en este apartado procederemos a hacer una aproximación similar con respecto a los lenguajes lineales. De esta forma, definiremos algunas subclases de lenguajes lineales que presentan explorabilidad local y cuya identificación en el límite queda garantizada a partir de reducciones a clases de lenguajes inferibles mediante texto.

En este caso, nuestro planteamiento será idéntico al realizado anteriormente con respecto a la reversibilidad. Al utilizar como fuente de información las cadenas estructurales de la gramática a identificar, utilizaremos de nuevo los conceptos ya definidos en la anterior sección.

Explorabilidad estructuralmente local en lenguajes lineales

Partiremos de los conceptos de gramática estructuralmente asociada y del homomorfismo h que hemos definido anteriormente. A partir de ellos podemos definir varias subclases de lenguajes lineales en función de las características de explorabilidad local definidas con respecto a los lenguajes lineales pares en el capítulo 4.

Definición 5.16. Sea G una gramática lineal en forma normal y G_{str} su gramática lineal par estructuralmente asociada. Diremos que $L(G)$ es

1. localmente explorable por estructura sii $L(G_{str}) \in \mathcal{LT}_{\mathcal{ELL}}$ de acuerdo con la definición 4.12.

2. localmente explorable en sentido estricto por estructura sii $L(G_{str}) \in \mathcal{LTSS}_{\mathcal{ELL}}$ de acuerdo con la definición 4.12.

3. localmente explorable sincronizado bipartido por estructura sii $L(G_{str}) \in \mathcal{LTSH}_{\mathcal{ELL}}$ de acuerdo con la definición 4.13.

4. localmente explorable no sincronizado bipartido por estructura sii $L(G_{str}) \in \mathcal{LTNSH}_{\mathcal{ELL}}$ de acuerdo con la definición 4.14.

□

A las cuatro clases de lenguajes definidas anteriormente las denotaremos respectivamente por $\mathcal{LT}_{\mathcal{LINST}}$, $\mathcal{LTSS}_{\mathcal{LINST}}$, $\mathcal{LTSH}_{\mathcal{LINST}}$ y $\mathcal{LTNSH}_{\mathcal{LINST}}$.

Al igual que hemos planteado en la anterior sección, podemos formular algoritmos o esquemas de aprendizaje basándonos en los algoritmos ya planteados sobre lenguajes lineales pares y las transformaciones sobre la información estructural. Así, si denotamos por $A_{LT_{ELL}}$, $A_{LTSS_{ELL}}$, $A_{LTSH_{ELL}}$ y $A_{LTNSH_{ELL}}$, a los algoritmos de aprendizaje especificados en el capítulo 4, podemos formular el esquema de aprendizaje de lenguajes lineales con explorabilidad estructuralmente local. El esquema genérico de aprendizaje lo mostramos en la figura 5.12

Obviamente, la identificación en el límite de los distintos lenguajes lineales con explorabilidad estructuralmente local se deduce a partir de las transformaciones realizadas sobre la información de entrada y la convergencia en el límite de los algoritmos de aprendizaje para lenguajes lineales pares que ha sido demostrada en el capítulo 4. Formalizaremos este resultado mediante el siguiente teorema.

Entrada: Un conjunto finito de esqueletos SK
Método:
$S = \emptyset$
$\forall sk \in SK$
$S = S \cup \{cadenastr(sk)\}$
$\mathcal{H} = A_{LT_{ELL}}$
alternativamente $\mathcal{H} = A_{LTSS_{ELL}}$
alternativamente $\mathcal{H} = A_{LTSHS_{ELL}}$
alternativamente $\mathcal{H} = A_{LTNSHS_{ELL}}$
Salida: $h(\mathcal{H})$

Figura 5.12: Método de inferencia para lenguajes lineales con explorabilidad estructuralmente local

Teorema 5.6. Las clases de lenguajes $\mathcal{LT}_{\mathcal{LIN}\mathcal{ST}}$, $\mathcal{LTSS}_{\mathcal{LIN}\mathcal{ST}}$, $\mathcal{LTSHS}_{\mathcal{LIN}\mathcal{ST}}$ y $\mathcal{LTNSHS}_{\mathcal{LIN}\mathcal{ST}}$ son identificables en el límite a partir de texto estructural.

Demostración.
La demostración es similar a la establecida en el Teorema 5.5 con la salvedad que la garantía de convergencia en el límite se establece a partir de los resultados del capítulo 4.

□

5.8 Conclusiones y problemas abiertos

En el presente capítulo hemos introducido la información estructural como fuente de información para el aprendizaje de lenguajes lineales. La justificación de este protocolo para el intercambio de información se establece a partir de los problemas sobre lenguajes lineales cuya resolución se ha demostrado con una elevada complejidad o inexistente. En el momento de escribir esta memoria el autor no conoce ningún método que permita identificar en el límite a la clase de los lenguajes lineales a partir de muestra completa, exceptuando el método puramente enumerativo y, por lo tanto, exponencial.

Los rasgos que hemos definido para caracterizar las distintas clases de gramáticas objeto de nuestro estudio han sido siempre referidas a la estruc-

tura y se resumen en *fuerte determinismo hacia atrás*, *fuerte determinismo estructural hacia atrás*, *distinguibilidad terminal y estructural*, *reversibilidad estructural* y *explorabilidad estructuralmente local*. En todos los casos se han presentado algoritmos que identifican eficientemente las clases de gramáticas correspondientes.

Podemos establecer unas líneas de actuación que, a juicio del autor, merecen la pena considerarse para futuros trabajos. Las resumimos como sigue.

- Las clases de lenguajes que han sido definidas a lo largo del capítulo no han sido estudiadas. Sus propiedades, relaciones entre sí y caracterizaciones algebraicas deben ser analizadas y formalizadas. De igual forma, se deberían analizar las relaciones con otras clases de lenguajes que han definido distintos autores que se han referido a lo largo de este capítulo.

- La importancia que ha suscitado la información estructural como fuente de información debería impulsar una reflexión más profunda acerca de la *Identificación estructural en el límite*.

- En relación con el punto anterior, todas las clases de lenguajes que se han definido son identificables con información estructural *positiva*. Una cuestión natural que aparece repetidamente en cuanto a la inferencia gramatical es la que hace referencia a la relación entre las muestras negativas y las muestras estructurales y cuándo se pueden sustituir unas por otras. Los lenguajes lineales pueden ser un buen campo de estudio para analizar esta relación en profundidad.

- Al igual que sucedía en el capítulo 4, se deberían estudiar nuevas variantes de explorabilidad local.

Capítulo 6

Complejidad de los lenguajes lineales

En este capítulo nos proponemos analizar otro aspecto de los lenguajes lineales distinto al de su identificación como es el de su *complejidad descriptiva*. La complejidad descriptiva, a diferencia de la complejidad computacional, consiste en el análisis de la cantidad de información necesaria para describir un conjunto de conceptos formales que, en nuestro caso, toma la forma de una clase de lenguajes. Obsérvese que la complejidad computacional es una medida *dinámica* que permite analizar el consumo de recursos necesarios para reconocer o procesar un lenguaje, mientras que la complejidad descriptiva es una medida *estática* que no considera procesos computacionales al margen de la propia descripción de los objetos.

Son muy variadas las propuestas que se han hecho acerca de las medidas descriptivas de algunas clases de lenguajes formales. Por ejemplo, si se trabaja sobre subclases de lenguajes regulares, entonces medidas descriptivas de las mismas pueden ser el tamaño de los autómatas que las caracterizan (como una función de la longitud de las cadenas hacia el número de estados), o el tamaño de los semigrupos que definen las clases bajo estudio.

Es importante diferenciar este tipo de medidas con otra complejidad descriptiva como es la *complejidad de Kolomogorov* [41]. La complejidad de Kolmogorov liga la descripción de los objetos a programas que sean capaces de representarlos. Por lo tanto, se asume que todo objeto que admite una complejidad de Kolmogorov acotada admite también un programa que lo describe. En este capítulo detallaremos sucintamente la complejidad de Kolmogorov de los lenguajes lineales pero, previamente, haremos un análisis en torno a algunas descripciones admisibles para dicha clase de lenguajes.

Además de las anteriores medidas, analizaremos también otra medida de complejidad computacional como es la *complejidad de reverso* [9]. La explicación de por qué este analisis se justificará de nuevo a partir de las descripciones realizadas sobre los lenguajes lineales que nos servirán como punto de reflexión acerca de algunas características exclusivas de esta clase de lenguajes.

6.1 Expresiones regulares

Nuestro punto de partida para analizar la complejidad descriptiva de los lenguajes lineales es proporcionar un mecanismo descriptivo alternativo al que proporcionan las gramáticas lineales, los autómatas de pila o cierto tipo de máquinas de Turing restringidas. En este caso, nos proponemos proporcionar un formalismo similar al de las *expresiones regulares* que describen a los lenguajes regulares. A lo largo del tiempo, han sido numerosos los formalismos similares que se han aplicado para describir ciertas clases de lenguajes. Así, Yokomori [77] propuso una extensión de las expresiones regulares para definir lenguajes de contexto libre para su posterior inferencia inductiva. Hashiguchi y Yoo [30, 78, 31] definieron expresiones regulares para la caracterización de ciertas clases de lenguajes regulares *estrella acotados*. Gruska [28] introdujo el operador de *iteración de símbolos* para obtener cierto tipo de expresiones que caracterizaban los lenguajes incontextuales. Finalmente, podemos citar a Yntema [76] que mediante operadores distintos al de Gruska también obtuvo cierto tipo de expresiones que describían los lenguajes incontextuales.

Bajo nuestra aproximación a los lenguajes lineales, en primer lugar recordaremos algunas definiciones y resultados relacionados con las expresiones regulares que se pueden consultar en [8, 65].

Las expresiones regulares forman un lenguaje donde cada cadena denota a su vez un lenguaje regular que puede coincidir o no con el de otras cadenas. Este tipo de formalismo para expresar los lenguajes regulares sistematizan la definición de los mismos y proporcionan métodos de resolución para los problemas de *análisis* (obtención del lenguaje aceptado por un autómata finito) y *síntesis* (obtención de un autómata finito aceptor de un lenguaje dado).

A continuación, proporcionaremos algunas nociones que se centran básicamente en la relación entre las gramáticas regulares (o los autómatas finitos) y las expresiones regulares.

Definición 6.1. Sea Σ un alfabeto que no contiene los símbolos (y) (paréntesis). Una expresión regular sobre Σ se define de forma inductiva como sigue

1. $\emptyset$ y λ son expresiones regulares,

2. para todo símbolo $a \in \Sigma$, a es una expresión regular,
3. si r es una expresión regular entonces (r) también lo es,
4. sean r y s expresiones regulares, entonces $r + s$, rs y r^* son expresiones regulares.

Sólo son expresiones regulares las definidas en los puntos anteriores.

□

Obsérvese que cualquier expresión regular r definida sobre Σ define un lenguaje que lo denotaremos por $L(r)$ y que queda establecido por las siguientes reglas

1. $L(\emptyset)$ es el lenguaje vacío,
2. $L(\lambda) = \{\lambda\}$,
3. $\forall a \in \Sigma$, $L(a) = \{a\}$,
4. $L((r)) = L(r)$,
5. $L(r + s) = L(r) \cup L(s)$,
6. $L(rs) = L(r)L(s)$,
7. $L(r^*) = (L(r))^*$.

Está demostrado formalmente que una expresión regular denota un lenguaje regular y viceversa. En concreto se refiere al *problema de síntesis* como el problema de encontrar una gramática regular (autómata finito) equivalente a una expresión regular y al *problema de análisis* como el problema inverso, es decir encontrar una expresión regular equivalente a una gramática regular (autómata finito).

A lo largo del tiempo se han proporcionado distintas soluciones al problema de síntesis como los expuestos en [38] y [7] y más recientemente la propuesta de Hromkovič *et al.* [34]. De todas las soluciones expuestas, en este capítulo se tratará el método de las *derivadas formales* [7] con el propósito de adaptarlo a las expresiones lineales que se propondrán más adelante. A continuación se define un concepto básico relacionado con este método.

Queda demostrado por el teorema de Nerode que, dado un lenguaje regular sobre el alfabeto Σ, el número de conjuntos distintos de buenos finales respecto del lenguaje regular, obtenidos sobre todas las cadenas del alfabeto, es un

número finito. De igual forma, a partir de los conjuntos de buenos finales se puede obtener un autómata finito determinista equivalente a la expresión regular tal y como se muestra en [7]. El método es conocido como el de las *derivadas formales*.

Por otra parte, el problema de análisis también ha recibido distintas soluciones, entre ellas las expuestas en [8]. En concreto, existe un método a partir de sistemas de ecuaciones lineales que permite asociar expresiones regulares a las soluciones del sistema y, por lo tanto, al lenguaje de la gramática regular (autómata finito) tal y como puede verse en [8]. Este método, al igual que el planteado para el problema de síntesis también será adaptado para ser aplicado a la extensión que se propondrá sobre las expresiones regulares.

6.2 Expresiones lineales

Partiendo de las definiciones anteriores, en este apartado veremos una extensión de las expresiones regulares para definir los lenguajes generados por las gramáticas lineales. Esta extensión podrá ser aplicada a los lenguajes regulares como caso particular. A lo largo de esta sección se irán proporcionando definiciones y resultados que facilitarán la posterior propuesta de métodos para la resolución de los problemas de síntesis y análisis asociados a los lenguajes lineales.

El planteamiento de este tipo de expresiones implica, entre otras ventajas, las siguientes

1. Se elimina ambigüedad en la definición de los lenguajes lineales (no así en las gramáticas que los generan).

2. Se proporcionan métodos similares para los problemas de síntesis y análisis trasladados a gramáticas lineales.

3. Se explicita la estructura gramatical del lenguaje, es decir, las expresiones no sólo definen las cadenas de un lenguaje sino su forma de derivación en una gramática.

Con un ánimo de simplificación en el formalismo, trabajaremos con gramáticas lineales en forma normal sustituyendo las producciones del tipo $A \to a$ por producciones $A \to aB$ y $A \to \lambda$

Definición 6.2. Sean $\Delta = \{a_1, a_2, \cdots, a_n\}$ y $\Sigma = \{b_1, b_2, \cdots, b_m\}$ dos alfabetos. Se define el alfabeto Δ indexado por Σ y se denota por Δ_Σ como el conjunto $\{a_{1_{b_1}}, a_{1_{b_2}}, \cdots, a_{1_{b_m}}, \cdots, a_{n_{b_1}}, \cdots, a_{n_{b_m}}\}$ □

Tal y como se apuntó anteriormente se propone para cualquier lenguaje lineal la siguiente forma normal en la gramática que lo genera

1. $A \to aB \mid Ba$ donde $A, B \in N$ y $a \in \Sigma$
2. $A \to \lambda$ donde $A \in N$

Podemos ahora introducir el tipo de alfabetos indexados que darán lugar a los lenguajes lineales. En este caso, el conjunto de índices se reduce a sólo dos elementos, Ly R, que definirán respectivamente el tipo de producciones que aparecerán en las gramáticas, esto es respectivamente lineales por la izquierda (R) y lineales por la derecha (L). Veamos en primer lugar, cómo se puede obtener un lenguaje sobre un alfabeto si se proporciona su definición a partir de una indexación sobre el mismo.

Definición 6.3. Sea Σ un alfabeto y $\Delta = \Sigma_{\{L,R\}}$. Dada una cadena x sobre el alfabeto Δ se define la imagen de x en Σ, y se denota por $im_\Sigma(x)$, mediante las siguientes reglas

1. Si $x = \lambda$, entonces $im_\Sigma(\lambda) = \lambda$.
2. Si $x = a_L \cdot w$, con $w \in \Delta^*$, entonces $im_\Sigma(a_L \cdot w) = a \cdot im_\Sigma(w)$
3. Si $x = a_R \cdot w$, con $w \in \Delta^*$, entonces $im_\Sigma(a_R \cdot w) = im_\Sigma(w) \cdot a$

Por extensión, si $L \subseteq \Delta^*$ entonces $im_\Sigma(L) = \{im_\Sigma(x) : x \in L\}$.

□

Pasamos a dar ahora una definición de expresiones regulares extendidas, que denominaremos *expresiones lineales.*

Definición 6.4. Sea Σ un alfabeto y $\Delta = \Sigma_{\{L,R\}}$. Una expresión lineal sobre Δ se define inductivamente a partir de las siguientes reglas.

1. $\emptyset$ y λ son expresiones lineales,
2. para todo símbolo $a \in \Sigma$, a_L y a_R son expresiones lineales,
3. si r es una expresión lineal, entonces (r) también lo es,
4. si r y s son expresiones lineales entonces $r + s$, rs y r^* son expresiones lineales.

Sólo son expresiones lineales aquellas que sigan las reglas anteriormente enunciadas. □

Obsérvese que la definición 6.4 coincide con la definición 6.1. Esto es porque podemos definir cualquier expresión lineal r como una expresión regular sobre $\Sigma^*_{\{L,R\}}$. Como consecuencia lógica de la concordancia de las definiciones, toda expresión lineal r denota un lenguaje regular sobre $\Sigma_{\{L,R\}}$ que se denota por $L(r)$. Por otra parte, toda expresión lineal r denota un lenguaje que, en principio, puede no ser regular. Este lenguaje al que aludimos es el denotado por $im_\Sigma(r)$ y queda definido por las siguiente reglas

1. $im_\Sigma(\emptyset)$ es el lenguaje vacío,
2. $im_\Sigma(\lambda) = \{\lambda\}$,
3. $\forall a \in \Sigma$, $im_\Sigma(a_L) = im_\Sigma(a_R) = \{a\}$,
4. $im_\Sigma((r)) = im_\Sigma(r)$,
5. $im_\Sigma(r + s) = im_\Sigma(r) \cup im_\Sigma(s)$,
6. $im_\Sigma(rs) = \{im_\Sigma(xy) : x \in L(r), y \in L(s)\}$,
7. $im_\Sigma(r^*) = \{\lambda\} \cup im_\Sigma(rr^*)$

Obsérvese que para cualquier expresión lineal r, los lenguajes $L(r)$ e $im_\Sigma(r)$ son diferentes.

Ejemplo 6.1. La expresión lineal $(a_L b_R b_R)^*$ denota el lenguaje lineal no regular definido por el conjunto $\{a^i b^{2i} : i \in \mathbb{N}\}$.

La expresión lineal $(a_L a_R + b_L b_R)^*$ denota el lenguaje lineal no regular $\{ww^r : w \in (a + b)^*\}$.

□

Síntesis de expresiones lineales

En primer lugar, a través del siguiente teorema, podemos establecer la correspondencia entre las expresiones lineales y los lenguajes lineales. Damos solución de esta forma al problema de síntesis de expresiones lineales, es decir, establecemos un procedimiento que, dada una expresión lineal nos permite obtener una gramática lineal equivalente.

Teorema 6.1. Si r es una expresión lineal sobre el alfabeto indexado $\Sigma_{\{L,R\}}$ entonces $im_\Sigma(r)$ es un lenguaje lineal.

Demostración.

La demostración se realizará por inducción sobre el número de operadores que aparecen en la expresión lineal, de forma similar a como se plantea el teorema de Kleene para expresiones regulares [38]. En este caso, se obtendrá una gramática lineal en forma normal para cada expresión

- *Base de inducción*

  Si $r = \emptyset$ entonces la gramática lineal $(\{S\}, \Sigma, \emptyset, S)$ define $im_\Sigma(\emptyset) = \emptyset$.

  Si $r = \lambda$ entonces la gramática lineal $(\{S\}, \Sigma, \{S \to \lambda\}, S)$ define el lenguaje $im_\Sigma(\lambda) = \lambda$.

  $\forall a \in \Sigma$ si $r = a_L$, la gramática lineal se corresponde con la tupla

  $$(\{S, A\}, \{a\}, \{S \to aA; \quad A \to \lambda\}, S).$$

  $\forall a \in \Sigma$ si $r = a_R$, la gramática lineal que se propone es la siguiente

  $$(\{S, A\}, \{a\}, \{S \to Aa; \quad A \to \lambda\}, S).$$

- *Hipótesis de inducción*

  Sea r una expresión lineal que contiene como máximo n operadores de unión, concatenación o clausura con $n \geq 0$, entonces existe una gramática lineal G_r cuyo lenguaje es $im_\Sigma(r)$.

- *Paso de inducción*

  Sea t una expresión lineal que contiene $n + 1$ operadores de unión, concatenación o clausura. Veamos las distintas posibilidades que pueden plantearse en la expresión t

  1. $t = r + s$, siendo r y s expresiones lineales que tienen como máximo n operadores. En este caso, por hipótesis de inducción, existen gramáticas lineales $G_r = (N_r, \Sigma, P_r, S_r)$ y $G_s = (N_s, \Sigma, P_s, S_s)$ que denotan los lenguajes $im_\Sigma(r)$ e $im_\Sigma(s)$ respectivamente. Se puede asumir sin restar generalidad al resultado que $N_r \cap N_s = \emptyset$. En este caso, se puede comprobar fácilmente que la gramática $G_t = (\{S_t\} \cup N_r \cup N_s, \Sigma, P, S_t)$, donde el conjunto P se define de la forma

     $$P = \{S_t \to \alpha : S_r \to \alpha \in P_r\} \cup \{S_t \to \beta : S_s \to \beta \in P_s\} \cup P_r \cup P_s,$$

     genera el lenguaje $im_\Sigma(r) \cup im_\Sigma(s) = im_\Sigma(t)$.

2. $t = rs$, siendo r y s expresiones lineales que tienen como máximo n operadores. Al igual que en caso anterior, por hipótesis de inducción, las gramáticas lineales $G_r = (N_r, \Sigma, P_r, S_r)$ y $G_s = (N_s, \Sigma, P_s, S_s)$ denotan los lenguajes $im_\Sigma(r)$ e $im_\Sigma(s)$ respectivamente. De igual forma, se puede asumir que $N_r \cap N_s = \emptyset$. Se propone la gramática $G_t = (N_r \cup N_s, \Sigma, P'_r \cup P_s, S_r)$ que genera $im_\Sigma(t) = im_\Sigma(rs)$. En la gramática propuesta, P'_r se define por las siguientes reglas
   - Si $A \to aB \in P_r$ entonces $A \to aB \in P'_r$.
   - Si $A \to Ba \in P_r$ entonces $A \to Ba \in P'_r$.
   - Si $A \to \lambda \in P_r$ entonces $\forall \alpha : S_s \to \alpha \in P_s \;\; A \to \alpha \in P'_r$.
3. $t = r^*$, siendo r una expresión lineal que tiene como máximo n operadores. De nuevo, por hipótesis de inducción, la gramática lineal $G_r = (N_r, \Sigma, P_r, S_r)$ denota el lenguaje $im_\Sigma(r)$. Se propone la gramática $G_t = (N_r, \Sigma, P_r \cup \{S_r \to \lambda\} \cup P'_r, S_r)$ para generar $im_\Sigma(t) = im_\Sigma(r^*)$. Aquí, P'_r se define por el conjunto

$$\{A \to \alpha : (A \to \lambda \in P_r) \wedge (S_r \to \alpha \in P_r)\}$$

□

Ejemplo 6.2. A partir de la construcciones propuestas en el anterior teorema, vamos a proporcionar una gramática para el lenguaje denotado por la imagen de la expresión $(a_L b_R b_R)^*$. La gramática se definirá por pasos, según las operaciones que se apliquen en cada momento

1. a_L

   $S \to aA \quad A \to \lambda$

2. b_R

   $S' \to A'b \quad A' \to \lambda$

3. $a_L b_R$

   $S \to aA \quad A \to A'b$

   $A' \to \lambda$

   $S' \to A'b$

   Esta última producción la podemos eliminar por resultar inútil.

4. $a_L b_R b_R$

   $S \to aA \quad A \to A'b$

   $A' \to A''b \quad A'' \to \lambda$

   $S'' \to A''b$

   Al igual que en el caso anterior, esta última producción se elimina por resultar inútil.

5. $(a_L b_R b_R)^*$

   $S \to aA \mid \lambda \quad A \to A'b$

   $A' \to A''b \quad A'' \to \lambda \mid aA$

Fácilmente, se puede comprobar que la última gramática genera el lenguaje $\{a^i b^{2i} : i \in \mathbb{N}\}$, que es la imagen de la expresión lineal propuesta. □

Otro algoritmo de síntesis

Una vez resuelto el problema de síntesis de expresiones lineales, proporcionaremos un método distinto de resolución basado en el método de las derivadas formales [7]. Se procederá, en primer lugar, a dar las reglas de derivación de las expresiones lineales respecto de símbolos. Posteriormente, se extenderán las reglas para la derivación respecto de cadenas y se propondrá un método de construcción de gramáticas lineales a partir de las derivadas calculadas previamente.

Definición 6.5. Sea t una expresión lineal sobre $\Sigma^*_{\{L,R\}}$ y sea $a \in \Sigma$. Se define la derivada de t respecto de a_L, denotándolo por $a_L^{-1}(t)$, mediante las siguientes reglas.

1. $a_L^{-1}(\emptyset) = \emptyset$
2. $a_L^{-1}(\lambda) = \emptyset$
3. $a_L^{-1}(b_L) = \begin{cases} \lambda & \text{si} \quad a = b \\ \emptyset & \text{si} \quad a \neq b \end{cases}$
4. $\forall a, b \in \Sigma \quad a_L^{-1}(b_R) = \emptyset$
5. $a_L^{-1}(r + s) = a_L^{-1}(r) + a_L^{-1}(s)$
6. $a_L^{-1}(rs) = a_L^{-1}(r)s + \delta(r)a_L^{-1}(s)$ donde $\delta(r) = \begin{cases} \lambda & \text{si} \quad \lambda \in r \\ \emptyset & \text{si} \quad \lambda \notin r \end{cases}$

7. $a_L^{-1}(r^*) = a_L^{-1}(r)r^*$

De igual forma se define la derivada de t respecto de a_R, denotándolo por $a_R^{-1}(t)$, mediante las correspondientes reglas

1. $a_R^{-1}(\emptyset) = \emptyset$
2. $a_R^{-1}(\lambda) = \emptyset$
3. $\forall a, b \in \Sigma \quad a_R^{-1}(b_L) = \emptyset$
4. $a_R^{-1}(b_R) = \begin{cases} \lambda & \text{si} \quad a = b \\ \emptyset & \text{si} \quad a \neq b \end{cases}$
5. $a_R^{-1}(r+s) = a_R^{-1}(r) + a_R^{-1}(s)$
6. $a_R^{-1}(rs) = a_R^{-1}(r)s + \delta(r)a_R^{-1}(s)$ donde $\delta(r) = \begin{cases} \lambda & \text{si} \quad \lambda \in r \\ \emptyset & \text{si} \quad \lambda \notin r \end{cases}$
7. $a_R^{-1}(r^*) = a_R^{-1}(r)r^*$

□

A partir de las anteriores reglas, la extensión de las derivadas respecto de cadenas del alfabeto $\Sigma_{\{L,R\}}$ se proporciona mediante la siguiente definición

Definición 6.6. Sea t una expresión lineal sobre $\Sigma_{\{L,R\}}$ y sea $x \in \Sigma^*_{\{L,R\}}$. Se define la derivada de t respecto de x, denotándolo por $x^{-1}(t)$, mediante las siguientes reglas

1. $\lambda^{-1}(t) = t$
2. $(a_L x)^{-1}(t) = x^{-1}(a_L^{-1}(t))$
3. $(a_R x)^{-1}(t) = x^{-1}(a_R^{-1}(t))$

□

A partir del anterior cálculo de derivadas se propone una construcción para la gramática equivalente a una expresión lineal dada. Si t es una expresión lineal sobre el alfabeto $\Sigma_{\{L,R\}}$, entonces denotaremos por $\mathcal{D}(t)$ al conjunto de los conjuntos de buenos finales de las cadenas del alfabeto respecto de la expresión lineal. Este conjunto es finito ya que, en definitiva, t es una expresión regular sobre el alfabeto indexado. La construcción de la gramática lineal equivalente a la expresión se define de la siguiente forma

$G = (N, \Sigma, P, S)$
$N = \mathcal{D}(t)$
$S = \lambda^{-1}(t)$
P se define mediante las siguientes reglas, siendo $x \in \Sigma^*_{\{L,R\}}$

1. $x^{-1}(t) \to a \cdot (a_L x)^{-1}(t)$
2. $x^{-1}(t) \to (a_R x)^{-1}(t) \cdot a$
3. Si $\lambda \in x^{-1}(t)$ entonces $x^{-1}(t) \to \lambda$

Ejemplo 6.3. Dada la expresión regular $(a_L b_R b_R)^*$ se procede a calcular el conjunto de derivadas respecto de las cadenas sobre el alfabeto $\{a, b\}_{\{L,R\}}$

- $\lambda^{-1}((a_L b_R b_R)^*) = (a_L b_R b_R)^*$
- $a_L^{-1}((a_L b_R b_R)^*) = a_L^{-1}(a_L) b_R b_R (a_L b_R b_R)^* = b_R b_R (a_L b_R b_R)^*$
- $b_L^{-1}((a_L b_R b_R)^*) = b_L^{-1}(a_L) b_R b_R (a_L b_R b_R)^* = \emptyset$
- $a_R^{-1}((a_L b_R b_R)^*) = a_R^{-1}(a_L) b_R b_R (a_L b_R b_R)^* = \emptyset$
- $b_R^{-1}((a_L b_R b_R)^*) = b_R^{-1}(a_L) b_R b_R (a_L b_R b_R)^* = \emptyset$
- $(a_L a_L)^{-1}((a_L b_R b_R)^*) = a_L^{-1}(b_R) b_R (a_L b_R b_R)^* = \emptyset$
- $(a_L b_L)^{-1}((a_L b_R b_R)^*) = b_L^{-1}(b_R) b_R (a_L b_R b_R)^* = \emptyset$
- $(a_L a_R)^{-1}((a_L b_R b_R)^*) = a_R^{-1}(b_R) b_R (a_L b_R b_R)^* = \emptyset$
- $(a_L b_R)^{-1}((a_L b_R b_R)^*) = b_R^{-1}(b_R) b_R (a_L b_R b_R)^* = b_R (a_L b_R b_R)^*$
- $(a_L b_R a_L)^{-1}((a_L b_R b_R)^*) = a_L^{-1}(b_R)(a_L b_R b_R)^* = \emptyset$
- $(a_L b_R b_L)^{-1}((a_L b_R b_R)^*) = b_L^{-1}(b_R)(a_L b_R b_R)^* = \emptyset$
- $(a_L b_R a_R)^{-1}((a_L b_R b_R)^*) = a_R^{-1}(b_R)(a_L b_R b_R)^* = \emptyset$
- $(a_L b_R b_R)^{-1}((a_L b_R b_R)^*) = b_R^{-1}(b_R)(a_L b_R b_R)^* = (a_L b_R b_R)^*$

A partir de las anteriores derivadas la gramática equivalente a la expresión es la siguiente tomando los siguientes símbolos auxiliares $S = \lambda^{-1}((a_L b_R b_R)^*)$, $A = a_L^{-1}((a_L b_R b_R)^*)$ y $B = (a_L b_R)^{-1}((a_L b_R b_R)^*)$

$$S \to aA \mid \lambda \quad A \to Bb \quad B \to Sb$$

□

Análisis de expresiones lineales

A continuación se procederá a dar una solución al problema de análisis que es el inverso al enunciado anteriormente. Esto es, dada una gramática lineal encontrar la expresión lineal cuya *imagen* denote el lenguaje de la gramática. El esquema a seguir será la reducción de la gramática lineal a una regular y su posterior resolución con métodos ya conocidos. Para llevar a cabo la reducción necesaria se procede a dar la siguiente definición

Definición 6.7. Sea $G = (\Sigma, N, P, S)$ una gramática lineal en forma normal. Definimos la gramática *extendida regular* de G y lo denotamos por G_{er} como la tupla $(\Sigma_{\{L,R\}}, N, P', S)$, donde P' se define a partir de las siguientes reglas

1. Si $A \to \lambda \in P$ entonces $A \to \lambda \in P'$
2. Si $A \to aB \in P$ entonces $A \to a_L B \in P'$
3. Si $A \to Ba \in P$ entonces $A \to a_R B \in P'$

□

Lema 6.1. Sea G una gramática lineal en forma normal y G_{er} la gramática extendida regular de G, entonces $x \in L(G_{er})$ si y sólo si $im_\Sigma(x) \in L(G)$

Demostración.

En primer lugar, veremos que $x \in L(G_{er})$ implica que $im_\Sigma(x) \in L(G)$. La secuencia de derivación de x en G_{er} tomará la siguiente forma

$$S \underset{G_{er}}{\overset{\alpha_1}{\Rightarrow}} x_1 A_1 \underset{G_{er}}{\overset{\alpha_2}{\Rightarrow}} x_1 x_2 A_2 \cdots \underset{G_{er}}{\overset{\alpha_{n-1}}{\Rightarrow}} x_1 x_2 \cdots x_{n-1} A_{n-1} \underset{G_{er}}{\overset{\alpha_n}{\Rightarrow}} x_1 x_2 \cdots x_n = x$$

donde cada producción α_i toma la forma $A_{i-1} \to x_i A_i$, por lo tanto, eligiendo las producciones de G que originaron cada α_i, que denotaremos por α_i', podemos generar la siguiente derivación en G

$$S \underset{G}{\overset{\alpha_1'}{\Rightarrow}} \gamma_1 A_1 \phi_1 \underset{G}{\overset{\alpha_2'}{\Rightarrow}} \gamma_2 A_2 \phi_2 \cdots \underset{G}{\overset{\alpha_{n-1}}{\Rightarrow}} \gamma_{n-1} A_{n-1} \phi_{n-1} \underset{G}{\overset{\alpha_n'}{\Rightarrow}} \gamma_n \phi_n$$

donde $\gamma_i, \phi_i \in \Sigma^*$ para $i = 1, \cdots, n$, y es fácil comprobar que $\gamma_i \phi_i = im_\Sigma(x_1 \cdots x_i)$ para $i = 1, \cdots, n$. Por lo tanto, podemos concluir que $\gamma_n \phi_n = im_\Sigma(x)$ y que $S \underset{G}{\overset{*}{\Rightarrow}} im_\Sigma(x)$, con lo que $im_\Sigma(x) \in L(G)$ que era el enunciado que queríamos demostrar.

La otra implicación a demostrar, es decir $im_\Sigma(x) \in L(G) \Rightarrow x \in L(G_{er})$ puede realizarse de manera análoga a la anterior.

□

Ejemplo 6.4. Dada la gramática lineal definida por las producciones

$S \to aA \mid bB$
$A \to Aa \mid bB$
$B \to aC \mid Bb$
$C \to \lambda$

la gramática regular extendida se define por las siguientes producciones

$S \to a_L A \mid b_L B$
$A \to a_R A \mid b_L B$
$B \to a_L C \mid b_R B$
$C \to \lambda$

□

Podemos, a partir del concepto de gramática regular extendida, abordar de forma definitiva el problema de análisis. En primer lugar enunciaremos el siguiente lema.

Lema 6.2. Si G es una gramática lineal entonces existe una expresión lineal r tal que $L(G) = im_\Sigma(r)$.

Demostración.

A partir del esquema que se reproduce en la figura 6.1, podemos obtener la expresión lineal de una gramática lineal dada.

Así, tomando como punto de partida G basta con construir su gramática extendida regular G_{er}. Podemos asociar a G_{er} una expresión regular aplicando métodos bien conocidos como los que se muestran en [8]. De esta forma, la expresión regular r para G_{er} denota su lenguaje y, a su vez, es una expresión lineal que denota $im_\Sigma(L(G))$ tal y como se estableció en el lema 6.1.

□

Es claro que, dada una gramática lineal siempre podemos construir una gramática regular extendida asociada a ella. Si aplicamos a la gramática regular extendida el método de análisis basado en sistemas de ecuaciones tal y como se realiza sobre las gramáticas regulares [8], entonces se puede obtener una expresión lineal cuya imagen se corresponde con el lenguaje de la gramática lineal inicial. Veamos a continuación un ejemplo de aplicación de las citadas técnicas.

Ejemplo 6.5. Dada la gramática regular extendida del ejemplo 6.4, su expresión lineal se obtiene mediante el siguiente sistema de ecuaciones

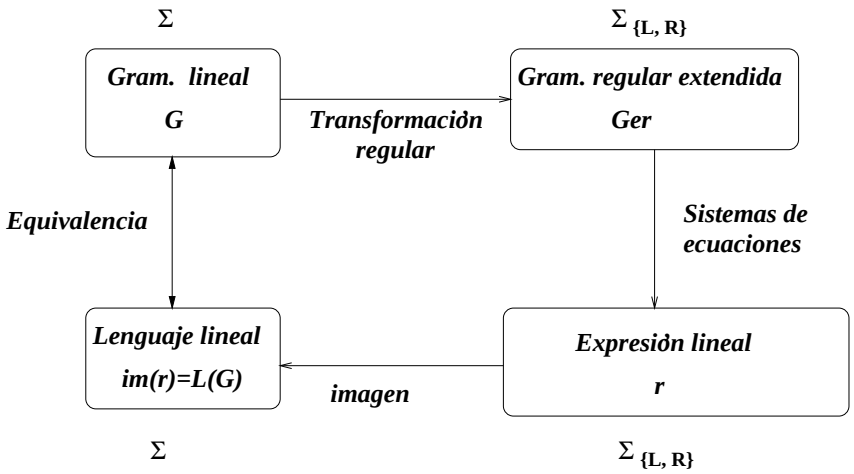


Figura 6.1: Esquema de obtención del lenguaje de una gramática lineal.

$$\left.\begin{array}{l} S = a_L A + b_L B \\ A = a_R A + b_L B \\ B = a_L C + b_R B \\ C = \lambda \end{array}\right\}$$

La expresión lineal asociada a este sistema de ecuaciones se obtiene mediante resolución del sistema anterior de la siguiente forma

$$\left.\begin{array}{l} S = a_L A + b_L B \\ A = a_R A + b_L B \\ B = a_L C + b_R B \\ C = \lambda \end{array}\right\} \Rightarrow C = \lambda$$

$$\left.\begin{array}{l} S = a_L A + b_L B \\ A = a_R A + b_L B \\ B = a_L + b_R B \end{array}\right\} \Rightarrow B = b_R^* a_L$$

$$\left.\begin{array}{l} S = a_L A + b_L b_R^* a_L \\ A = a_R A + b_L b_R^* a_L \end{array}\right\} \Rightarrow A = a_R^* b_L b_R^* a_L$$

$$S = a_L a_R^* b_L b_R^* a_L + b_L b_R^* a_L$$

□

6.3 Equivalencia entre gramáticas lineales

A partir del formalismo de expresiones lineales se pueden abordar problemas tales como el de la equivalencia entre gramáticas lineales [60] o el de la equivalencia estructural entre las mismas [35, 36, 54]. Si bien el primero resulta indecidible tal y como Rozenberg demostró en su momento [60], para el segundo podemos proporcionar un nuevo método que lo resuelve a pesar de no modificar los resultados ya conocidos sobre su complejidad.

Veamos en primer lugar el problema de la equivalencia entre gramáticas que lo relacionaremos con el presente trabajo según el siguiente teorema.

Teorema 6.2. Dadas dos expresiones lineales, r y s, el problema de la equivalencia entre ambas es indecidible. Es decir, no existe un procedimiento efectivo para establecer si $im_\Sigma(r) = im_\Sigma(s)$.

Demostración.

El problema de la equivalencia entre gramáticas lineales es indecidible tal y como demostró Rozenberg [60]. Es fácil reducir el problema de equivalencia de dos gramáticas lineales al de sus respectivas expresiones lineales. De aquí la irresolubilidad del problema expuesto en el teorema.

□

Por otra parte, el problema de la equivalencia estructural entre dos gramáticas lineales G_1 y G_2 consiste en establecer si el conjunto de árboles de derivación de G_1 es idéntico al de G_2 con la excepción de los nombres dados a los nodos internos de los árboles, es decir a los símbolos auxiliares de las gramáticas. Este problema fue demostrado en su momento como $\mathcal{PSPACE}$-completo [35, 36, 54]. Podemos reducir el problema de la equivalencia estructural al de la equivalencia entre gramáticas regulares, demostrado también como $\mathcal{PSPACE}$-completo [35].

Así, dadas dos gramáticas lineales G y G' su equivalencia estructural puede establecerse a partir de la equivalencia entre G_{er} y G'_{er}.

Teorema 6.3. Sean G y G' dos gramáticas lineales en forma normal y G_{er} y G'_{er} sus gramáticas extendidas regulares correspondientes. Entonces, G es estructuralmente equivalente a G' si y sólo si G_{er} es equivalente a G'_{er}

Demostración.

Trivial a partir de la demostración del lema 6.1.

□

Obsérvese que el anterior resultado se limita a gramáticas lineales en forma normal. Podemos extender en cierto sentido el anterior resultado para trabajar

con gramáticas lineales que no presenten una forma normal. Formalizaremos el resultado mediante el siguiente teorema.

Teorema 6.4. Sean G_1 y G_2 gramáticas lineales. Existen gramáticas lineales en forma normal G_1' y G_2' que son equivalentes respectivamente a G_1 y G_2 de foma que se cumplen los dos siguiente enunciados

1. si G_1 es estructuralmente equivalente a G_2 entonces G_1' es estructuralmente equivalente a G_2', y

2. si G_1' es estructuralmente equivalente a G_2' entonces G_1 es equivalente a G_2.

Demostración.

En primer lugar, obtendremos G_1' y G_2' a partir de G_1 y G_2 como sigue: Para cada producción de G_1 (o de G_2) de la forma $A \to a_1 \dots a_n B b_1 b_2 \dots b_m$ obtendremos un conjunto de producciones de la forma $A \to a_1 A_1, \cdots, A_{n-1} \to a_n B_1$ y $B_1 \to B_2 b_m, \cdots, B_m \to B b_1$. Para cada producción de la forma $A \to a_1 \dots a_n$ obtendremos un conjunto de producciones de la forma $A \to a_1 A_1, \cdots, A_{n-1} \to a_n A_n$ y $A_n \to \lambda$. Finalmente, a partir de las producciones $A \to a_1 \dots a_n B$ y $A \to B b_1 \dots b_m$ pueden obtenerse conjuntos de producciones similares a las anteriores.

Podemos analizar, a partir de las gramáticas obtenidas anteriormente, los dos enunciados del teorema.

1. Supongamos que G_1 es estructuralmente equivalente a G_2. Entonces, tal y como hemos obtenido las producciones en G_1' y G_2', estas dos gramáticas también son estructuralmente equivalentes.

2. Si G_1' es estructuralmente equivalente a G_2' entonces, trivialmente G_1 y G_2 son equivalentes. Obsérvese que G_1 y G_2 no son necesariamente estructuralmente equivalentes ya que las transformaciones en las producciones que hemos aplicado no son correspondencias inyectivas. Es decir, puede ser que producciones distintas, proporcionen el mismo conjunto de producciones normalizadas. Por ejemplo, si tenemos $A \to abB$ y $B \to c$ en G_1 y $A \to aB$ y $B \to bc$ en G_2 entonces las estructuras obtenidas en G_1 y G_2 son distintas, no así las que obtenemos en G_1' y G_2'.

□

6.4 Equivalencia entre expresiones lineales

A pesar de los resultados que hemos expuesto acerca de la complejidad e indecidibilidad de los problemas de equivalencia en las expresiones y gramáticas lineales, podemos, sin embargo, establecer algunas transformaciones en estas que nos preserven la equivalencia de las mismas. Así, podemos beneficiarnos de la dualidad que plantean las expresiones lineales (en el sentido de que definen simultáneamente lenguajes regulares y lineales distintos), para proporcionar propiedades de equivalencia entre las mismas.

A partir de estas propiedades podremos profundizar en algunos aspectos de la complejidad de las gramáticas lineales. En primer lugar, definiremos dos tipos de equivalencia que están ampliamente relacionadas.

Definición 6.8. Sean r y s expresiones lineales sobre $\Sigma_{\{L,R\}}$. Diremos que r y s son *regular-equivalentes*, y lo denotaremos por $r \equiv_{REG} s$ sii $L(r) = L(s)$. Diremos que r y s son *lineal-equivalentes*, y lo denotaremos por $r \equiv_{LIN} s$, sii $im_\Sigma(r) = im_\Sigma(s)$. □

A partir de la anterior definición prodemos proporcionar ya un conjunto de propiedades de equivalencia entre expresiones lineales basándonos en las relativas a las expresiones regulares. Esto lo haremos a partir de la siguiente propiedad.

Propiedad 6.1. Sean r y s expresiones lineales sobre $\Sigma_{\{L,R\}}$. Si $r \equiv_{REG} s$ entonces $r \equiv_{LIN} s$. □

Demostración.

Sean r y s expresiones lineales sobre $\Sigma_{\{L,R\}}$ regular-equivalentes. Obviamente, $x \in L(r)$ sii $x \in L(s)$. Supongamos que $x \in im_\Sigma(r)$. Claramente, debe existir $\hat{x} \in L(r)$ de forma que $im_\Sigma(\hat{x}) = x$. Por lo tanto, $\hat{x} \in L(s)$ dado que $r \equiv_{REG} s$ y $x \in im_\Sigma(s)$.

El resultado partiendo de $im_\Sigma(s)$ puede demostrarse de forma similar, por lo que $im_\Sigma(r) = im_\Sigma(s)$ y $r \equiv_{LIN} s$.

Ejemplo 6.6. El resultado inverso al planteado en la Propiedad 6.1 no es cierto. Obsérvese que $a_L b_R \equiv_{LIN} b_R a_L$ mientras que $a_L b_R \not\equiv_{REG} b_R a_L$. □

Una vez establecidas las propiedades de equivalencia entre expresiones regulares como propiedades válidas para las expresiones lineales, nos proponemos establecer dos conjuntos de propiedades de equivalencia que, en general, sólo son válidas para las expresiones lineales. En concreto, nos plantearemos dos tipos de propiedades: propiedades de *permutación* y propiedades de *compresión*. Analizaremos cada una de ellas a continuación.

Permutación

Las *permutaciones* que vamos a analizar en las expresiones lineales intentan producir un orden en los símbolos que aparecen en las mismas. Así, el resultado de aplicar un cierto tipo de permutación en cualquier expresión lineal producirá que, en la misma, aparezcan en primer lugar los símbolos indexados por L y, posteriormente, aquellos indexados por R. Formalizaremos este tipo de permutación mediante la función $\psi : \Sigma^*_{\{L,R\}} \to \Sigma^*_{\{L,R\}}$ que se define como sigue

1. $\psi(\lambda) = \lambda$
2. $(\forall a \in \Sigma)(\forall x \in \Sigma^*_{\{L,R\}})\ \psi(a_R x) = \psi(x)a_R$
3. $(\forall a \in \Sigma)(\forall x \in \Sigma^*_{\{L,R\}})\ \psi(a_L x) = a_L \psi(x)$

Obviamente, si $x \in \Sigma^*_{\{L,R\}}$ entonces $\psi(x) \in \Sigma^*_{\{L\}}\Sigma^*_{\{R\}}$. Así, para cada cadena x, podemos fijar $\psi(x) = x_1 x_2$ donde $x_1 \in \Sigma^*_{\{L\}}$ y $x_2 \in \Sigma^*_{\{R\}}$. A partir de la función ψ podemos definir la transformación $\phi : \Sigma^*_{\{L,R\}} \to \Sigma^*_{\{L,R\}}$ como sigue

$$(\forall x \in \Sigma^*_{\{L,R\}}) \quad \phi(x) = x_1 x_2^{inv} \quad \text{con} \quad \psi(x) = x_1 x_2$$

Podemos establecer propiedades de equivalencia basándonos en la transformación ϕ como sigue

Propiedad 6.2. Sea $x \in \Sigma^*_{\{L,R\}}$, entonces $x \equiv_{LIN} \phi(x)$.

Demostración.

Demostraremos el enunciado mediante un proceso de inducción sobre la longitud de cada posible cadena x. En primer lugar, tomaremos λ como *base de inducción* en la demostración. Claramente, $\phi(\lambda) = \lambda$ y $x \equiv_{LIN} \phi(x)$.

A continuación formulamos una *hipótesis de inducción* que consiste en que para cualquier cadena x de longitud menor o igual que n se cumple que $x \equiv_{LIN} \phi(x)$.

Finalmente, tomaremos una cadena x con longitud $n+1$. Podemos considerar dos posibles casos. En primer lugar, tomemos $x = a_L x\prime$. En este caso, $\phi(x) = a_L \phi(x\prime)$, y $im_\Sigma(\phi(x)) = a \cdot im_\Sigma(\phi(x\prime)) = a \cdot im_\Sigma(x\prime) = im_\Sigma(x)$.

Como segundo caso, tomemos $x = a_R x\prime$. Entonces se cumple que $\phi(x) = x\prime_1 (x\prime_2 a_R)^{inv} = x\prime_1 a_R (x\prime_2)^{inv}$ con $x\prime_1 \in \Sigma^*_L$ y $x\prime_2 \in \Sigma^*_R$. Por lo tanto, $im_\Sigma(\phi(x)) = im_\Sigma(x\prime_1) \cdot im_\Sigma(a_R x\prime_2^{inv}) = im_\Sigma(x\prime_1) im_\Sigma(x\prime_2^{inv}) a = im_\Sigma(\phi(x\prime))a = im_\Sigma(x\prime)a = im_\Sigma(x)$.

□

Una vez demostrada la propiedad anterior, podemos extender la transformación ϕ para que actúe sobre conjuntos de cadenas, esto es lenguajes. Así, si L es un lenguaje sobre $\Sigma_{\{L,R\}}$, entonces $\phi(L) = \{\phi(x) \mid x \in L\}$. Finalmente, podemos definir ϕ sobre expresiones lineales de acuerdo con las siguientes reglas

1. $\phi(\emptyset) = \emptyset$
2. $\phi(\lambda) = \lambda$
3. $\phi(a_R) = a_R$ y $\phi(a_L) = a_L$
4. para cualquier expresión lineal r, $\phi((r)) = \phi(r)$
5. para cualquier par de expresiones lineales r y s, $\phi(r + s) = \phi(r) + \phi(s)$
6. para cualquier par de expresiones lineales r y s, $\phi(rs) = \phi(r)\phi(s)$[1]
7. para cada expresión lineal r, $\phi(r^*) = (\phi(r))^*$

A partir de las anteriores reglas y de la Propiedad 6.2, podemos deducir el siguiente resultado.

Corolario 6.1. Sean r y s expresiones lineales. Entonces se cumplen las siguientes propiedades de equivalencia

1. $r + s \equiv_{LIN} \phi(r) + \phi(s)$
2. $rs \equiv_{LIN} \phi(rs)$
3. $r^* \equiv_{LIN} (\phi(r))^*$

□

Ejemplo 6.7. Sea $r = a_L b_R a_L + (a_R a_L + b_R b_R)^*$. Entonces, r es equivalente a $\phi(r) = a_L a_L b_R + (a_L a_R + b_R b_R)^*$.

Sea $s = a_R a_R b_L^*$. Entonces, s es equivalente a $\phi(s) = b_L^* a_R a_R$.

□

[1]Obsérvese que, para cualquier expresión lineal $t = rs$, el valor de $\phi(rs)$ no es único y depende en la forma de seleccionar las diferentes expresiones r y s que dan lugar a t.

Compresión

La *compresión* es otra transformación sobre expresiones lineales que trata de reducir el número de índices que aparecen en las mismas. De esta forma, el objetivo es obtener una expresión más compacta de forma que aquellos símbolos consecutivos que aparecen con el mismo índice puedan agruparse en una sola cadena con un único índice. Esta forma de agrupación, será especialmente útil cuando se analice la complejidad de Kolmogorov de los lenguajes lineales. Formalizaremos esta transformación mediante la función φ como sigue

$$(\forall x \in \Sigma^*_{\{L,R\}}) \quad \varphi(x) = [im_\Sigma(x_1)]_L\,[im_\Sigma(x_2)]_R \quad \text{con} \quad \phi(x) = x_1x_2$$

Obsérvese que la función φ realiza un cambio de representación de forma que su resultado se expresa mediante cadenas de símbolos con un único índice L o R. En este caso, podemos definir $\{\Sigma^*\}_{\{L\}}$ como el conjunto de cadenas de Σ que tienen un único índice L. De igual forma, $\{\Sigma^*\}_{\{R\}}$ es el conjunto de cadenas de Σ que tienen un único índice R.

Definiremos la función im_Σ para que actúe sobre el resultado de φ como sigue

$$(\forall x \in \Sigma^*_{\{L,R\}}) im_\Sigma(\varphi(x)) = im_\Sigma(x_1)im_\Sigma(x_2) \quad \text{con} \quad \phi(x) = x_1x_2$$

Podemos obtener la siguiente propiedad de equivalencia con respecto a la función φ

Propiedad 6.3. Sea $x \in \Sigma^*_{\{L,R\}}$, entonces $x \equiv_{LIN} \varphi(x)$.

Demostración.

Demostraremos el enunciado tal y como lo hemos hecho en la Propiedad 6.2, es decir mediante un proceso inductivo sobre la longitud de las cadenas x

En primer lugar, como *base de inducción*, tomaremos $x = \lambda$. En este caso, $\varphi(\lambda) = \lambda$ e $im_\Sigma(\lambda) = im_\Sigma(\varphi(\lambda))$. Por lo tanto, $x \equiv_{LIN} \varphi(x)$.

Formularemos como *hipótesis de inducción* que para toda cadena de longitud menor o igual a n se cumple que $x \equiv_{LIN} \varphi(x)$.

Finalmente, tomaremos cadenas x de longitud $n+1$. En primer lugar supongamos que $x = a_Lx\prime$. Entonces $\varphi(a_Lx\prime) = [a \cdot im_\Sigma(x_1)]_L[im_\Sigma(x_2)]_R$. Aquí, $im_\Sigma([a \cdot im_\Sigma(x_1)]_L[im_\Sigma(x_2)]_R) = a \cdot im_\Sigma(x_1)im_\Sigma(x_2) = a \cdot im_\Sigma(x\prime) = im_\Sigma(x)$. La segunda posibilidad que existe es que $x = a_Rx\prime$. Aquí, $\varphi(a_Rx\prime) = [im_\Sigma(x_1)]_L[im_\Sigma(a_Rx_2)]_R = [im_\Sigma(x_1)]_L[im_\Sigma(x_2)\cdot a]_R$. Por lo tanto, $im_\Sigma(\varphi(a_Rx\prime)) = im_\Sigma(x_1)im_\Sigma(x_2) \cdot a = im_\Sigma(a_Rx\prime) = im_\Sigma(x)$.

□

Una vez demostrada la Propiedad 6.3 podemos extender la transformación φ para que actúe sobre conjuntos de cadenas, esto es lenguajes. Así, si tomamos L como un lenguaje definido sobre $\Sigma_{\{L,R\}}$ entonces $\varphi(L) = \{\varphi(x) \mid x \in L\}$. Finalmente, podemos definir φ para que actúe sobre expresiones lineales como sigue

1. $\varphi(\emptyset) = \emptyset$
2. $\varphi(\lambda) = \lambda$
3. $\varphi(a_R) = a_R$ y $\varphi(a_L) = a_L$
4. para cualquier expresión lineal r, $\varphi((r)) = \varphi(r)$
5. para cualquier par de expresiones lineales r y s, $\varphi(r + s) = \varphi(r) + \varphi(s)$
6. para cualquier par de expresiones lineales r y s, $\varphi(rs) = \varphi(r)\varphi(s)$ [2]
7. para cada expresión lineal r, $\varphi(r^*) = (\varphi(r))^*$

A partir de las anteriores reglas y de la Propiedad 6.3, podemos deducir el siguiente resultado.

Corolario 6.2. Sean r y s expresiones lineales. Entonces se cumplen las siguiente propiedades de equivalencia

1. $r + s \equiv_{LIN} \varphi(r) + \varphi(s)$
2. $rs \equiv_{LIN} \varphi(rs)$
3. $r^* \equiv_{LIN} (\varphi(r))^*$

□

Ejemplo 6.8. Sea $r = a_L b_R a_L + (a_R a_L + b_R b_R)^*$. Entonces, r es equivalente a $\varphi(r) = (aa)_L b_R + (a_L a_R + (bb)_R)^*$.

Sea $s = a_L(b_R a_L b_L)^* b_L$. Entonces s es equivalente a $\varphi(s) = a_L((ab)_L b_R)^* b_L$.

□

[2] Obsérvese que, para una expresión lineal dada $t = rs$, el valor de $\varphi(rs)$ no es único y depende de la forma de seleccionar las diferentes expresiones r y s que dan lugar a t.

6.5 Complejidad de reverso de los lenguajes lineales

La complejidad de reverso de un lenguaje, en su versión determinista, se define como el número máximo de cambios de dirección (*reversos*) que realizan las cabezas de cinta de una máquina de Turing determinista durante el procesamiento de una cadena de entrada [9, 75]. Denotaremos a la clase de lenguajes aceptados por máquinas de Turing deterministas con k cintas que realizan como máximo $\mathcal{O}(f(n))$ reversos al procesar entradas de longitud n por $DREVERSAL_k(f)$. Denotaremos por $DREVERSAL(f)$ a la unión de clases de lenguajes $\bigcup_{k\geq 1} DREVERSAL_k(f)$. Podemos proporcionar definiciones similares para máquinas de Turing no deterministas. Así, $NREVERSAL_k(f)$ denota la clase de lenguajes aceptados por máquinas de Turing no deterministas con k cintas que realizan como máximo $\mathcal{O}(f(n))$ reversos al procesar entradas de longitud n. Ha sido demostrado que la clase de lenguajes lineales $\mathcal{LIN}$ se corresponde con la clase de lenguajes $NREVERSAL_1(1)$ [75].

En este apartado, estudiaremos la complejidad de reverso asociada a las expresiones lineales. Obviamente, existe una clara correspondencia entre los cambios de dirección que realiza una máquina de Turing durante su computación ante cadenas de entrada de un lenguaje lineal y los cambios de linealidad que se realizan en un proceso de derivación de cadenas en una gramática lineal que genera el mismo lenguaje.

Con el ánimo de definir el numero de cambios de linealidad de izquierda a derecha (o de derecha a izquierda) que se realiza en un proceso derivativo de una gramática lineal, centraremos nuestra atención en los índices (L o R) que aparecen en los símbolos de la expresión lineal asociada a la gramática. De esta forma, los índices proporcionan suficiente información acerca de la complejidad de reverso de los lenguajes lineales.

Definición 6.9. Sea G una gramática lineal. Definiremos la complejidad de reverso de $L(G)$ como el número máximo de cambios de linealidad (de izquierda a derecha y de derecha a izquierda) que ocurren en la gramática cuando se deriva una cadena $x \in L(G)$. Obviamente, esta complejidad se define como una función que va del conjunto de naturales (la longitud de las cadenas derivadas) al conjunto de naturales (el número de cambios de linealidad). □

Denotaremos por $\mathcal{G}$ a la familia de gramáticas lineales en forma normal. De igual forma, denotaremos por $REVERSAL_{\mathcal{G}}(f)$ a la familia de lenguajes que pueden ser generados por gramáticas lineales de $\mathcal{G}$ con complejidad de reverso $\mathcal{O}(f(n))$.

Dado que el lenguaje de cualquier gramática lineal en forma normal puede describirse mediante una expresión lineal, podemos deducir la complejidad de reverso de un lenguaje lineal a partir del número de índices L y R no consecutivos que aparecen en su correspondiente expresión lineal.

Dado que trabajaremos con las expresiones lineales para analizar la complejidad de reverso, definiremos el *lenguaje de reverso* de una expresión lineal como el conjunto de cadenas formadas por los índices de las cadenas que pertenecen a la expresión. Lo formalizaremos mediante el conjunto $im_{\{L,R\}}(r)$ que se calcula mediante las siguientes reglas:

1. $im_{\{L,R\}}(\emptyset)$ es el lenguaje vacío
2. $im_{\{L,R\}}(\lambda) = \{\lambda\}$
3. $im_{\{L,R\}}(a_L) = L$
4. $im_{\{L,R\}}(a_R) = R$
5. $im_{\{L,R\}}((r)) = im_{\{L,R\}}(r)$
6. $im_{\{L,R\}}(r + s) = im_{\{L,R\}}(r) \cup im_{\{L,R\}}(s)$
7. $im_{\{L,R\}}(rs) = \{im_{\{L,R\}}(xy) : x \in L(r), y \in L(s)\}$
8. $im_{\{L,R\}}(r^*) = (im_{\{L,R\}}(r))^*$.

Dada una expresión lineal, a cada una de las cadenas de su lenguaje de reverso la denominaremos *cadena de reverso*. Obsérvese que, en cualquier proceso de derivación de una cadena en una gramática lineal en forma normal, aparece asociada una cadena de reverso que describe el número de cambios de linealidad que ocurren durante la derivación de la cadena en cuestión.

Ejemplo 6.9. Damos los siguientes ejemplos de lenguajes de reverso asociados a expresiones lineales:

1. Sea $r = (a_L b_R b_R)^*$, entonces $im_{\{L,R\}}(r) = (LRR)^*$.
2. Sea $s = (a_L a_R + b_L b_R)^*$, entonces $im_{\{L,R\}}(s) = (LR)^*$.
3. Sea $t = a_L b_R a_L + (a_R a_L + b_R b_R)^*$, entonces $im_{\{L,R\}}(t) = LRL + (RL + RR)^*$.

□

Un primer resultado que podemos obtener en cuanto a los lenguajes de reverso es el que se muestra a continuación.

Teorema 6.5. Sea r una expresión lineal sobre $\Sigma_{\{L,R\}}$. Entonces, $im_{\{L,R\}}(r)$ es regular.

Demostración.

Construiremos una gramática regular (lineal por la derecha) para cada una de las posibles expresiones lineales.

1. $r = \emptyset$

   La gramática regular $(\{S\}, \{L, R\}, \emptyset, S)$ genera trivialmente el conjunto vacío.

2. $r = \lambda$

   La gramática $(\{S\}, \{L, R\}, \{S \to \lambda\}, S)$ genera trivialmente el lenguaje $\{\lambda\}$.

3. $r = a_L$

   La gramática $(\{S\}, \{L, R\}, \{S \to L\}, S)$ genera $\{L\}$.

4. $r = a_R$

   La gramática $(\{S\}, \{L, R\}, \{S \to R\}, S)$ genera $\{R\}$.

5. $r = s + t$

   Supongamos que las gramáticas regulares $G_s = (N_s, \{L, R\}, P_s, S_s)$ y $G_t = (N_t, \{L, R\}, P_t, S_t)$ existen y que $L(G_s) = im_{\{L,R\}}(s)$ y $L(G_t) = im_{\{L,R\}}(t)$ con $N_s \cap N_t = \emptyset$. La gramática $(N_s \cup N_t \cup \{S_r\}, \{L, R\}, P_s \cup P_t \cup \{S_r \to S_s \mid S_t\}, S_r)$ genera el lenguaje de reverso $im_{\{L,R\}}(r)$.

6. $r = st$

   Como en el caso anterior, supongamos que las gramáticas regulares $G_s = (N_s, \{L, R\}, P_s, S_s)$ y $G_t = (N_t, \{L, R\}, P_t, S_t)$ generan $im_{\{L,R\}}(s)$ y $im_{\{L,R\}}(t)$ respectivamente, con $N_s \cap N_t = \emptyset$. La gramática $(N_s \cup N_t, \{L, R\}, P'_s \cup P_t, S_s)$ donde P'_s está definida por las reglas

   (a) $(\forall A, B \in N_s)\ (\forall a \in \{L, R, \lambda\})\ A \to aB \in P_s \Rightarrow A \to aB \in P'_s$

   (b) $(\forall A \in N_s)\ (\forall a \in \{L, R, \lambda\})\ A \to a \in P_s \Rightarrow A \to aS_t \in P'_s$

   genera el lenguaje de reverso $im_{\{L,R\}}(r)$.

7. $r = s^*$

   Supongamos que la gramática regular $G_s = (N_s, \{L, R\}, P_s, S_s)$ genera $im_{\{L,R\}}(s)$. Entonces, la gramática $G_r = (N_s \cup \{S_r\}, \{L, R\}, P_s \cup \{S_r \to S_s \mid \lambda\} \cup P'_s, S_r)$ donde P'_s se define mediante la regla

   $(\forall A \in N_s)\ (\forall a \in \{L, R, \lambda\})\ A \to a \in P_s \Rightarrow A \to aS_r \in P'_s$

   genera el lenguaje de reverso $im_{\{L,R\}}(r)$.

□

Una vez establecida la regularidad del lenguaje de reverso de cualquier expresión lineal podemos transformar cualquiera de ellas, de acuerdo con la función φ anteriormente definida, para obtener una descripción más compacta de los lenguajes lineales. Nuestra intención es evitar expresiones donde aparezcan de forma secuencial índices L o índices R dado que únicamente implica un cambio de linealidad la alternancia de índices L y R. Podemos encontrar para cada gramática lineal en forma normal G otra gramática lineal $\hat{G}$ de forma que si r es la expresión lineal para G entonces $\varphi(r)$ es la expresión lineal para $\hat{G}$. Así, denotaremos por $\hat{\mathcal{G}}$ a la clase de gramáticas $\hat{G}$ que cumplen la anterior característica.

Basándonos en la anterior clase de gramáticas, proporcionaremos el siguiente resultado acerca de la complejidad de reverso de los lenguajes lineales.

Teorema 6.6. $\mathcal{LIN} \subseteq REVERSAL_{\hat{\mathcal{G}}}(n)$.

Demostración.

La demostración del enunciado es fácilmente deducible a partir de las propiedades de equivalencia de compresión y del Teorema 6.5. Obsérvese que el lenguaje de reverso de cualquier expresión lineal es a su vez regular. Por lo tanto, cualquier cadena de reverso puede ser analizada en tiempo lineal y el número de reversos es en consecuencia lineal. De aquí que la función de complejidad sea $f(n) = n$.

□

Obviamente, todos los lenguajes lineales presentan una complejidad de reverso que está acotada superiormente de forma lineal, tal y como se expresa en el Teorema 6.6. Sin embargo, subclases de los lenguajes lineales, como por ejemplo los lenguajes regulares presentan una complejidad de reverso acotada por una constante. Es trivial comprobar que para cada lenguaje regular existe una expresión lineal donde no existen índices L (si la gramática es lineal por la izquierda) o índices R (si la gramática es lineal por la derecha) y, en consecuencia, no hace falta ningún cambio de linealidad para la derivación de

las cadenas del lenguaje. El lenguaje de reverso que presentan los lenguajes regulares está incluido en L^* o R^*.

Aceleración de la complejidad de reverso

Una vez establecido el resultado general para la complejidad de reverso, nos proponemos a continuación obtener un resultado de aceleración para la misma similar a los que se han planteado para otras medidas de complejidad como es el tiempo y similar a los resultados de compresibilidad que se han planteado para el espacio. Aquí debemos entender la aceleración como una reducción en el número de cambios de linealidad necesarios para derivar las cadenas de cualquier lenguaje lineal de acuerdo con una gramática lineal dada. Así, nos planteamos demostrar que, para cada gramática lineal, siempre podemos obtener otra gramática lineal equivalente de forma que el número de reversos de la gramática equivalente reduce en un factor constante el número de reversos de la gramática original. Esto lo estableceremos mediante el siguiente teorema.

Teorema 6.7. Sea L un lenguaje lineal. Para cada constante c tal que $0 < c \leq 1$, $L \in REVERSAL_{\hat{\mathcal{G}}}(c \cdot n)$.

Demostración.

En primer lugar supondremos que L es infinito ya que, si L fuera finito entonces fácilmente podríamos construir una gramática regular que lo genere y que, en consecuencia, tuviera una complejidad de reverso constante.

Por lo tanto, supongamos que L es infinito y que r es una expresión lineal que describe a L. Podemos obtener una expresión lineal equivalente a r a partir de la transformación $\varphi(r)$. Obsérvese que, en cualquier expresión lineal t^* podemos aplicar la propiedad de equivalencia $t^* \equiv_{REG} (\lambda + t + tt + \cdots + t^{k-1})(t^k)^*$. De esta forma, tomaremos cualquier subexpresión s que aparezca en $\varphi(r)$ y que esté afectada por el operador de clausura y le aplicaremos la anterior propiedad con el valor $k = \lceil 1/c \rceil$. Dado que $s \equiv_{LIN} (\lambda + s + ss + \cdots + s^{k-1})(s^k)^*$, entonces s es equivalente a $(\varphi(\lambda) + \varphi(s) + \cdots + \varphi(s^{k-1}))\varphi((s^k)^*)$. A continuación observemos la expresión $\varphi((s^k)^*)$. Podemos aplicar la propiedad de equivalencia $\varphi((s^k)^*) \equiv_{LIN} (\varphi(s^k))^*$. Es fácil comprobar que en $(\varphi(s^k))^*$ sólo es necesario un cambio de linealidad cada k símbolos. En consecuencia, la complejidad de reverso se reduce k veces, es decir, aumenta c veces. Por lo tanto, si aplicamos la anterior propiedad de equivalencia a todas aquellas subexpresiones que aparezcan en $\varphi(r)$ y que estén afectadas por el operador de clausura, entonces podemos reducir el número de cambios de linealidad de sus cadenas. Las demás subexpresiones que aparecen en $\varphi(r)$, y que no están afectadas por el operador de clausura, pueden transformarse de forma trivial

en otras expresiones equivalentes que reducen en la misma medida el número de reversos. □

Ejemplo 6.10. Sea $r = (a_L b_R)^*$. Podemos reducir el número de reversos a la mitad si obtenemos la expresión equivalente $(\lambda + a_L b_R)((aa)_L(bb)_R)^*$. De igual forma, se puede reducir el número de reversos a un tercio del de la expresión r al obtener la expresión $(\lambda + a_L b_R + (aa)_L(bb)_R)((aaa)_L(bbb)_R)^*$ □

Un problema de identificación asociado a la complejidad de reverso

Al analizar la complejidad de reverso de los lenguajes lineales hemos definido los lenguajes de reverso asociados a los mismos. El lenguaje de reverso de un lenguaje lineal no sólo nos proporciona una herramienta útil para analizar la complejidad de reverso sino que, además, describe de forma precisa la información puramente estructural que alberga la gramática asociada al mismo. Por ejemplo, en el caso de los lenguajes regulares, el lenguaje de reverso, al trabajar con gramáticas regulares, se limita a subconjuntos triviales de L^* (si trabajamos con gramáticas lineales por la derecha) o de R^* (si las gramáticas son lineales por la izquierda). En el caso de gramáticas lineales pares, el lenguaje de reverso vuelve a definirse trivialmente a partir de subconjuntos de $(LR)^*$ (si trabajamos con gramáticas en forma normal). Sin embargo, en el caso de los lenguajes lineales, los lenguajes de reverso asociados ya no se definen como conjuntos triviales. La variabilidad que aparece en las gramáticas lineales a la hora de ordenar la generación de los símbolos durante una derivación no es única y, por lo tanto, pueden aparecer multitud de lenguajes de reverso que formen una clase de lenguajes regulares a analizar.

Este problema suscita nuestro interés ya que, si podemos catalogar los lenguajes de reverso como alguna clase de lenguajes ya conocida, entonces podemos deducir nuevas formas normales para las gramáticas lineales que tienen asociadas. Más aún, si la clase de lenguajes de reverso pudiera ser inferida de forma eficiente, entonces se abriría una nueva aproximación a la identificación de lenguajes lineales a partir de cadenas. Podríamos deducir la estructura subyacente de las cadenas a partir del lenguaje de reverso y, posteriormente, podríamos resolver el problema genérico de la identificación mediante reducciones a clases ya conocidas de forma similar a como hizo Sakakibara mediante información estructural [62, 63]. Analizaremos estos aspectos a continuación.

Denotaremos a la clase de los lenguajes de reverso por $\mathcal{LIN}_{RV}$. Podemos aportar el siguiente resultado acerca de la inferibilidad de $\mathcal{LIN}_{RV}$.

Teorema 6.8. La familia de lenguajes de reverso, $\mathcal{LIN}_{RV}$ no es identificable únicamente a partir de datos positivos.

Demostración.

Definamos el siguiente conjunto de lenguajes de reverso

$S_1 = LR(LR)^*$

$S_2 = LR(LR)^*LR$

$S_3 = LR(LR)^*LR(LR)^*LR$

$S_i = S_{i-1}(LR)^*LR$

$S_* = (LR)^*$

No existe ningún indicador finito posible para S_*, ya que ese indicador lo sería también para algún lenguaje S_i que, a su vez, estaría incluido en S_* rompiendo la *Condición 3* expuesta por Angluin [3] como necesaria para la inferencia a partir de datos positivos y enunciada en el Teorema 2.1.

□

Obsérvese que el anterior teorema establece que los lenguajes de reverso no son identificables a partir de información positiva. Es decir, no podemos inferir los cambios de linealidad de una gramática lineal basándonos únicamente en ejemplos de cómo se producen esos cambios.

6.6 Complejidad de Kolmogorov de los lenguajes lineales

La complejidad de Kolmogorov [41] mide la cantidad de información necesaria para describir un objeto. Habitualmente, los objetos a los que nos referimos son generadores de secuencias infinitas de información (cadenas infinitas *aleatorias* o no). En nuestro caso, nos interesamos por la complejidad de Kolmogorov de los lenguajes lineales, es decir el número de *bits* necesarios para describirlos. Es evidente que la complejidad de Kolmogorov de un lenguaje lineal puede relacionarse con la complejidad de una gramática que lo genere. Dado que existen infinitas gramáticas que generan el mismo lenguaje y que los tamaños de esas gramáticas pueden variar, la complejidad de Kolmogorov de un lenguaje expresada a través de la complejidad de una gramática lineal que lo genere es una cota superior de la complejidad real del lenguaje.

Otro punto de interés al medir la complejidad de Kolmogorov de los lenguajes lineales es que cada uno de ellos puede considerarse una cadena infinita compuesta por las cadenas del lenguaje separadas por símbolos predefinidos. Es decir, si $L = \{x_1, x_2, \cdots\}$, entonces L representa la cadena (infinita) $< L >= x_1\#x_2\#\cdots$ donde el símbolo # actúa como separador. Bajo este

punto de vista, los lenguajes lineales codifican cadenas no aleatorias ya que, como es bien conocido, las cadenas aleatorias no pueden comprimirse y las cadenas de los lenguajes lineales sí, debido a que pueden describirse mediante la expresión lineal correspondiente.

Para abordar este aspecto, proporcionaremos, en primer lugar, la notación y definiciones que pueden consultarse en [41]. Posteriormente, proporcionaremos una cota superior para la complejidad de Kolmogorov de los lenguajes lineales.

Dado un *programa* p, denotaremos su longitud mediante $lg(p)$ y denotaremos su salida mediante $S(p)$. Dada una cadena finita x, denotaremos mediante $n(x)$ su orden en una enumeración predefinida.

Definición 6.10. Sea Σ un alfabeto y x una secuencia (posiblemente infinita) de símbolos de Σ. La complejidad de Kolmogorov de x se define como $\mathcal{K}(x) = min\{lg(p) : S(p) = n(x)\}$. Es decir, es el tamaño del programa mínimo que escribe (da como salida) x. □

Podemos modificar la anterior definición para su actuación sobre lenguajes lineales como sigue.

Definición 6.11. Sea L un lenguaje lineal. La complejidad de Kolmogorov de L se define como $\mathcal{K}(L) = min\{lg(p) : S(p) = n(r) \wedge im_{\Sigma}(r) = L\}$. Es decir, es el tamaño del programa mínimo que escribe una expresión lineal r tal que r describe a L. □

Obsérvese que, para describir cualquier expresión lineal, debemos especificar: (1) Un conjunto de cadenas pertenecientes a un alfabeto Σ (los *generadores*), (2) Un conjunto de índices que definan los cambios de linealidad L y R, y (3) Una secuencia de operadores de clausura, concatenación y unión que aparecen en la expresión. Por lo tanto, cualquier expresión lineal r se puede definir como $\eta_m(x_{1y_1}x_{2y_2}\cdots x_{my_m})$, donde $x_1, \cdots, x_m$ son los generadores, $y = y_1y_2\cdots y_m \in \{L, R\}^*$ son los índices para los cambios de linealidad y η es un operador que define el orden de actuación de los operadores dentro de la expresión. A partir de esta definición, podemos proporcionar la siguiente cota superior para cualquier lenguaje lineal $\widehat{L}$ que admita una expresión lineal r

$$\mathcal{K}(\widehat{L}) \leq \mathcal{K}(r) \leq \mathcal{K}(x_1) + \cdots + \mathcal{K}(x_m) + \mathcal{K}(y) + \mathcal{K}(\eta_m(x_{1y_1}\cdots x_{my_m})) + \mathcal{O}(1)$$

con $im_{\Sigma}(r) = \widehat{L}$ y $r = \eta_m(x_{1y_1}\cdots x_{my_m})$. Podemos deducir una nueva cota para η_m dado que la información relevante que aporta η_m es la posición donde

actúa cada operador y el operador que actúa en cada posición. Por lo tanto, podemos establecer la siguiente cota superior para η_m

$$\mathcal{K}(\eta_m(x_{1y_1} \cdots x_{my_m})) \leq log(x_1) + \cdots + log(x_m) + \mathcal{O}(1)$$

Podemos aplicar las propiedades de equivalencia de compresión y permutación para establecer nuevas cotas. Lo formalizaremos mediante el siguiente resultado.

Teorema 6.9. Sea L un lenguaje lineal definido sobre Σ y r una expresión lineal tal que $L = im_\Sigma(r)$. Entonces $\mathcal{K}(L) \leq \mathcal{K}(\varphi(r)) \leq \mathcal{K}(r)$.

Demostración.

Obviamente, la primera desigualdad se cumple dado que podemos describir L a partir de $\varphi(r)$. La segunda desigualdad se puede deducir a partir de las siguiente afirmaciones:

1. El número de índices para los cambios de linealidad que aparecen en $\varphi(r)$ es menor o igual que el de los que aparecen en r.
2. El número de operadores de concatenación que actúan en r es mayor o igual que el número de operadores de concatenación que actúan en $\varphi(r)$. Por lo tanto, la complejidad del término η puede reducirse.

□

Finalmente, podemos establecer una relación entre la complejidad de reverso y la complejidad de Kolmogorov. Veremos que a medida que la complejidad de reverso decrece (obteniendo gramáticas *aceleradas*) la complejidad de Kolmogorov aumenta (cuesta más describir las citadas gramáticas). Lo formalizaremos mediante el siguiente resultado.

Teorema 6.10. Sea L un lenguaje lineal sobre Σ y r una expresión lineal tal que $im_\Sigma(r) = L$. Sea c una constante tal que $0 < c \leq 1$ y r_c una expresión lineal para L de forma que la complejidad de reverso de L de acuerdo con r decrece $\lceil 1/c \rceil$ veces al utilizar r_c. Entonces $\mathcal{K}(r) \leq \mathcal{K}(r_c)$.

Demostración.

Sea r_c una expresión lineal para L obtenida a partir de r aplicando el Teorema 6.7. Obviamente, podemos observar que r_c tiene un número de generadores mayor o igual que r. Además, r_c tiene un mayor número de operadores de unión y concatenación que r. Por lo tanto, se puede deducir que la cantidad de información necesaria para describir r_c es mayor que la necesaria para describir r.

□

6.7 Conclusiones y problemas abiertos

Las expresiones lineales introducidas en el presente capítulo se han mostrado como un formalismo adecuado para el estudio de algunas características de los lenguajes lineales y, de forma más directa, de las gramáticas lineales. Este tipo de expresiones nos permiten establecer reducciones de los lenguajes lineales a los lenguajes regulares y, por lo tanto, aprovechar algunos resultados ya establecidos sobre los mismos.

Además, hemos iniciado el estudio de la complejidad descriptiva de los lenguajes lineales de forma natural aprovechando las expresiones lineales definidas en este capítulo. No sólo hemos acotado la complejidad de Kolmogorov de los lenguajes lineales sino que, además, hemos podido comprobar algunas relaciones entre medidas de complejidad como son la complejidad de Kolmogorov y la complejidad de reverso.

A la luz de lo ya estudiado, podemos establecer algunas líneas de investigación que profundicen y amplíen los resultados sobre complejidad descriptiva de los lenguajes formales. Detallamos a continuación algunos de esos futuros trabajos.

- Las expresiones lineales definidas en este capítulo representan un primer paso a dar en cuanto a la definición de clases de lenguajes con el mismo formalismo. Así, tomando en cuenta la estructura de las gramáticas, podemos definir nuevas clases de expresiones que se asemejen a las ya estudiadas y que caractericen clases de lenguajes superiores como los lenguajes incontextuales o los sensibles al contexto.

- Otro trabajo que reclama nuestra atención es la del estudio de las distintas clases de lenguajes lineales definidas en los capítulos 4 y 5 mediante expresiones lineales. Así, podemos redefinir en esos términos los lenguajes reversibles, localmente testables y distinguibles por símbolos terminales y estructura.

- Hemos estudiado la relación entre la complejidad descriptiva y la complejidad de reverso. Podemos estudiar otras relaciones entre otras medidas de complejidad tomando, como nexo de enlace, la complejidad de Kolmogorov. De esta forma, podríamos estudiar relaciones entre la complejidad de espacio, tiempo y de reverso y la complejidad de descripción en los lenguajes lineales.

- Finalmente, como un objetivo que nos vuelve al área del aprendizaje automático, debemos estudiar la obtención de nuevas formas normales

en los lenguajes lineales a través de las expresiones lineales. Este estudio parece ser crítico a la hora de plantear algoritmos de aprendizaje de lenguajes lineales que prescinda de la información estructural empleada en el capítulo 5. En este sentido, las propiedades de permutación y compresión ya estudiadas son un primer paso.

Capítulo 7

Conclusiones

A lo largo de la presente tesis se han estudiado algunos problemas relativos al aprendizaje de lenguajes lineales y a su complejidad descriptiva. En ambas áreas de estudio se presenta un nexo discursivo común como es el relativo al estudio de las gramáticas formales. Habitualmente, en cuanto al aprendizaje de lenguajes formales, se hace especial énfasis en la identificación de cualquier gramática que genere el lenguaje objetivo. Aquí, sin embargo, hemos dado el papel principal a la identificación de gramáticas estructuralmente equivalentes a las que generan el lenguaje objetivo. Una de las conclusiones que podemos establecer como válida es que, cuando los problemas de la clase de lenguajes bajo estudio, empiezan a mostrarse con alta complejidad o sencillamente irresolubles, el discurso del aprendizaje algorítmico y, más en concreto, el de la inferencia gramatical, debe cambiar. Se debe cambiar el protocolo de información y contar con información estructural y, por lo tanto, se deben reformular las condiciones generales del aprendizaje para llegar hacia lo que podríamos denominar la *Identificación estructural en el límite.*

Lejos de cerrar el estudio de algunos problemas, la presente tesis ha tenido como objetivo el estudio de los mismos con un nuevo enfoque. La importancia que se le ha dado a la información estructural como fuente de datos válida ha sido escasa a lo largo del tiempo. Sin embargo, creemos que este tipo de información resulta imprescindible si queremos estudiar el aprendizaje de conceptos cada vez más complejos. Por otra parte, desde un punto de vista práctico, el esfuerzo que se hace en algunas tareas a la hora de representar los objetos, como en el Reconocimiento Automático de Formas, la Predicción Automática de Series Formales y la Inteligencia Artificial en general, no se ve comprometido al introducir estructura que, por otra parte, nos permite establecer relaciones más complejas en los conceptos a representar.

Otro aspecto fundamental que se enfatiza en la presente tesis es la concerniente al estudio formal de los lenguajes y las gramáticas. En este trabajo resulta imprescindible apelar a aquellos resultados que previamente han sido estudiados en el área de la Teoría de autómatas, gramáticas y lenguajes formales. Más aún, en múltiples ocasiones hemos obtenido resultados más cercanos a esta teoría que a la del aprendizaje computacional o algorítmico. Este trasvase de resultados resulta especialmente enriquecedor ya que no sólo permite profundizar en los problemas bajo estudio sino que los enfoca bajo una perspectiva que en algunos casos nos muestra aspectos nuevos que antes no habían sido considerados.

Por último, nos gustaría resaltar que la clase de los lenguajes lineales es un laboratorio de pruebas ideal para entender las dificultades algorítmicas del aprendizaje automático. En esta clase podemos obtener resultados y conclusiones válidas para clases de lenguajes más amplias. Obsérvese que, en el caso de los lenguajes regulares, los problemas de equivalencia, equivalencia estructural, ambigüedad, determinismo vs. no determinismo y complejidad descriptiva o de reverso no nos permiten obtener conclusiones para otras clases superiores, dado que, los problemas que suscitan son en algunos casos inexistentes o con una dificultad limitada.

Bibliografía

[1] V. Amar, G. Putzolu On a Family of Linear Grammars. *Information and Control* 7, pp 283-291. 1964.

[2] S. Andrei, M. Kudlek *Bidirectional Parsing for Linear Languages.* Preproceedings of the Fourth International Conference on Developments in Language Theory (DLT 99). W. Thomas (Ed.) pp 331-344. 1999.

[3] D. Angluin Inductive Inference of Formal Languages from Positive Data. *Information and Control* 45, pp 117-135. 1980.

[4] D. Angluin Inference of Reversible Languages. *Journal of the Association for Computing Machinery* Vol 29 No 3, pp 741-765. July 1982.

[5] D. Angluin, C. Smith Inductive Inference : Theory and Methods. *Computing Surveys* 15, No. 3 pp 237-269. 1983.

[6] J. Berstel *Transductions and Context-Free Languages.* Ed. Teubner Stuttgart. 1979.

[7] J. Brozozowski Derivatives of Regular Expressions. *Journal of the Association for Computing Machinery* Vol. 11, No. 4, 481-494. 1964.

[8] J. Carroll, D. Long *Theory of Finite Automata.* Ed. Prentice-Hall. 1989.

[9] J. Chen, C. Yap Reversal Complexity. *SIAM Journal on Computing* Vol. 20, No. 4, 622-638. 1991.

[10] C. de La Higuera. Characteristic Sets for Polynomial Grammatical Inference. *Machine Learning* 27, 125-138. 1997.

[11] C. de la Higuera, J. Oncina. *Inferring Deterministic Linear Languages.* (aceptado en la conferencia COLT 2002).

[12] D. de Jongh, M. Kanazawa Learnability Theory. *7th ESSLLI, Barcelona.* 1995.

[13] J.D. Emerald, K.G. Subramanian, D.G. Thomas *Learning code regular and code linear languages.* Proceedings of the Third International Colloquium on Grammatical Inference ICGI-96. (L. Miclet, C. de la Higuera, eds). LNAI Vol. 1147, pp 211-221. Springer. 1996.

[14] J.D. Emerald, K.G. Subramanian, D.G. Thomas *Learning a subclass of context-free Languages.* Proceedings of the 4th International Colloquium ICGI-98. (V. Honavar, G. Slutzki, eds). LNAI Vol. 1433, pp 223-243. 1998.

[15] J.D. Emerald, K.G. Subramanian, D.G. Thomas *Inferring Subclasses of contextual languages.* Proceedings of the 5th International Colloquium ICGI 2000. (A. Oliveira, ed.)LNAI Vol. 1891, pp 65-74. 2000.

[16] H. Fernau *Learning of terminal distinguishable languages.* Technical Report WSI–99–23. Universität Tübingen (Germany), Wilhelm-Schickard-Institut für Informatik, 1999.

[17] H. Fernau *k-gram extensions of terminal distinguishable languages.* Proceedings of the International Conference on Pattern Recognition ICPR 2000, Vol. 2 pp 125-128. IEEE Press. 2000.

[18] H. Fernau *Identification of function distinguishable languages.* Proceedings of the 11th International Conference on Algorithmic Learning Theory ALT 2000. (H. Arimura, S. Jain, A. Sharma, eds.). LNCS Vol. 1968 pp 116-130. Springer-Verlag 2000.

[19] K. Fu, T. Booth Grammatical Inference : Introduction and Survey, Part I and II. *IEEE Transactions on Pattern Recognition and Machine Intelligence* Vol PAMI-8, No.3, pp 343-375. 1986.

[20] P. García *Explorabilidad Local en Inferencia Inductiva de Lenguajes Regulares y Aplicaciones.* Ph.D. Thesis. Departamento de Sistemas Informáticos y Computación. Universidad Politécnica de Valencia. 1988.

[21] P. García Learning k-Testable Tree Sets from positive data. *Technical Report* DSIC-II/46/93. Departamento de Sistemas Informáticos y Computación. Universidad Politécnica de Valencia. 1993.

[22] P. García, J. Oncina Inference of Recognizable tree sets. *Technical Report* DSIC-II/47/93. Departamento de Sistemas Informáticos y Computación. Universidad Politécnica de Valencia. 1993.

[23] P. García, E. Vidal, J. Oncina Learning Locally Testable Languages in the Strict Sense. Proceedings of the First International Workshop on Algorithmic Learning Theory. pp 325-338. 1990.

[24] E.M. Gold Language Identification in the Limit. *Information and Control* 10, pp 447-474. 1967.

[25] J. Gray, M. Harrison On the Covering and Reduction Problems for Context-Free Grammars. *Journal of the ACM*, Vol. 19, No. 4. pp 675-698. 1972.

[26] S. Greibach The Undecidability of the Ambiguity Problem for Minimal Linear Grammars. *Information and Control* 6, pp 119-125. 1963.

[27] S. Greibach The Unsolvability of the Recognition of Linear Context-Free Languages. *Journal of the ACM* 13:4, pp 582-587. 1966.

[28] J. Gruska A Characterization of Context-free languages. *Journal of Computer and System Sciences* 5, 353-364. 1971.

[29] M. Harrison *Introduction to Formal Language Theory.* Addison-Wesley Publishing Company. 1978.

[30] K. Hashiguchi, H. Yoo. Extended regular expressions of star degree at most two. *Theoretical Computer Science* 76, 272-284. 1990.

[31] K. Hashiguchi The Infinite 2-Star Height Hierarchy of Extended Regular Languages of Star Degree at Most Two. *Information and Computation* 114, 237-246. 1994.

[32] M. Holzer, K. Lange *On the Complexities of Linear LL(1) and LR(1) Grammars.* Proceedings of the 9th International Conference on Fundamentals of Computation Theory (FCT 93). LNCS 710. pp 299-308. Springer. 1993.

[33] J. Hopcroft, J. Ullman *Introduction to Automata Theory, Languages and Computation.* Addison-Wesley Publishing Company. 1979.

[34] J. Hromkovič, S. Seibert and T. Wilke Translating Regular Expressions into Small ϵ-Free Nondeterministic Finite Automata. *14th Annual Symposium on Theoretical Aspects of Computer Science (STACS'97).* R. Reischuk and M. Morvan (Eds.). LNCS Vol. 1200, pp 55-66. Springer. 1997.

[35] H. Hunt III, D. Rosenkrantz, T. Szymanski On the Equivalence, Containment, and Covering Problems for Regular and Context-Free Languages. *Journal of Computer and System Sciences* 12 pp 222-268. 1976.

[36] H. Hunt III, D. Rosenkrantz, T. Szymanski The Covering Problem for Linear Context-Free Grammars. *Theoretical Computer Science* 2, pp 261-382. 1976.

[37] H. Hunt, D. Rosenkrantz Efficient Algorithms for Structural Similarity of Grammars. *Proceedings of the Seventh Annual ACM Symposium in Principles of Programming Languages.* pp 213-219. 1980.

[38] S. C. Kleene Representation of Events in Nerve Nets and Finite Automata. pp 3-41 in *Automata Studies*, C.E. Shannon and J. McCarthy (Eds.). Princeton University Press. 1956.

[39] T. Koshiba, E. Mäkinen, Y. Takada Inferring pure context-free languages from positive data. *Acta Cybernetica* 14 pp 469-477. 2000.

[40] T. Koshiba, E. Mäkinen, Y. Takada Learning deterministic even linear languages from positive examples. *Theoretical Computer Science* 185 pp 63-97. 1997.

[41] M. Li, P. Vitányi *An Introduction to Kolmogorov Complexity and its Applications.* Springer-Verlag. 1993.

[42] E. Mäkinen The grammatical inference problem for the Szilard languages of Linear Grammars. *Information Processing Letters* 36, pp 203-206. 1990.

[43] E. Mäkinen On the structural grammatical inference problem for some classes of context-free grammars. *Information Processing Letters* 42 pp 1-5. 1992.

[44] E. Mäkinen Remarks on the structural grammatical inference problem for context-free grammars. *Information Processing Letters* 44 pp 125-127. 1992.

[45] E. Mäkinen A Note on the Grammatical Inference Problem for Even Linear Languages. Technical Report A-1994-9. Department of Computer Science. University of Tampere. 1994.

[46] E. Mäkinen *On inferring zero-reversible languages.* Acta Cybernetica 14, pp 479-484. 2000 Science. University of Tampere. 1998.

[47] R. McNaughton, S. Papert *Counter-free automata.* MIT Press. 1971.

[48] L. Miclet Grammatical Inference. In *Syntactic and Structural Pattern Recognition. Theory and Applications* Series in Computer Science Vol. 7. World Scientific. 1990.

[49] T. Mitchell *Machine Learning.* Ed. McGraw Hill. 1997.

[50] B. Natarajan *Machine Learning. A Theoretical Approach.* Morgan Kaufmann Publisher Inc. 1991.

[51] J. Oncina, P. García Inferring Regular Languages in Polynomial Updated Time. In *Series in Machine Perception and Artificial Intelligence (1) : Pattern Recognition and Image Analysis.* World Scientific. 1992.

[52] D. Osherson, M. Stob, S. Weinstein *Systems That Learn.* Cambridge, MA : MIT Press. 1986.

[53] D. Osherson, S. Weinstein, D. de Jongh, E. Martin Formal Learning Theory. Research Report LP-94-08. ILLC. Universiteit van Amsterdam. 1994.

[54] M. Paull, S. Unger Structural Equivalence of Context-Free Grammars. *Journal of Computer and Systems Sciences* 2, pp 427-463. 1968.

[55] E. Post Recursive Unsolvability of a Problem of Thue. In *The Undecidable.* M. Davis (ed.). Raven Press Books. 1965.

[56] V. Radhakrishnan Grammatical Inference from Positive Data : An Effective Integrated Approach. *Ph.D. Thesis.* Department of Computer Science and Engineering. IIT Bombay. 1987.

[57] V. Radhakrishnan, G. Nagaraja Inference of Regular Grammars via Skeletons. *IEEE Trans. on Systems, Man and Cybernetics*, 17 No. 6, pp 982-992. 1987.

[58] V. Radhakrishnan, G. Nagaraja Inference of Even Linear Grammars and its Application to Picture Description Language. *Pattern Recognition* 21, No. 1. pp 55-62. 1988

[59] G. Révész *Introduction to Formal Languages.* McGraw-Hill Book Company. 1983.

[60] G. Rozenberg Direct Proofs of the Undecidability of the Equivalence Problem for Sentential Forms of Linear Context-Free Grammars and the Equivalence Problem for 0L Systems. *Information Processing Letters* 1, pp 233-235. 1972.

[61] J. Ruiz Familias de Lenguajes Explorables : Inferencia Inductiva y Caracterización Algebraica. *Tesis doctoral*. Departamento de Sistemas Informáticos y Computación. Universidad Politécnica de Valencia. 1997.

[62] Y. Sakakibara Learning context-free grammars from structural data in polynomial time. *Theoretical Computer Science* 76 pp 232-242. 1990.

[63] Y. Sakakibara Efficient Learning of Context-Free Grammars from Positive Structural Examples. *Informacion and Computation* 97 pp 23-60. 1992.

[64] Y. Sakakibara Recent advances of grammatical inference. *Theoretical Computer Science* 185 pp 15-45. 1997.

[65] A. Salomaa *Formal Languages*. Academic Press. 1973.

[66] K. Salomaa, S. Yu EDT0L Structural Equivalence is Decidable. *Proceedings of DTMCS'96* Springer. pp 363-375. 1996.

[67] J. Sempere Un nuevo algoritmo de aprendizaje de lenguajes regulares a partir de presentación completa ordenada lexicográficamente. *Proyecto de Licenciatura*. Facultad de Informática. Universidad Politécnica de Valencia. 1992.

[68] J. Sempere, A. Fos *Learning Linear Grammars from Structural Information Lecture Notes in Artificial Intelligence. Proceedings of the Third International Colloquium, ICGI-96*. Springer-Verlag. pp 126-133. 1996.

[69] R. Solomonoff A formal Theory of Inductive Inference (Part I and II). *Information and Control* 7, pp 1-22, pp 224-254. 1964.

[70] Y. Takada Grammatical Inference of Even Linear Languages based on Control Sets. *Information Processing Letters* 28, No. 4, pp 193-199. 1988.

[71] Y. Takada Algorithmic Learning Theory of Formal Languages and its Applications. Research Report IIAS-RR-93-6E. FUJITSU. 1993.

[72] Y. Takada A Hierarchy of Language Families Learnable by Regular Languages Learners. *Lecture Notes in Artificial Intelligence. Proceedings of the Second International Colloquium on Grammatical Inference and Applications, ICGI-94*. Springer-Verlag. pp 16-24. 1994.

[73] B. Trakhtenbrot, Y. Barzdin *Finite Automata: Behavior and Synthesis*. North Holland Publishing Company. 1973.

[74] L. Valiant A Theory of the Learnable. *Communications of the ACM* 27(11), pp 1134-1142. 1984.

[75] K. Wagner, G. Wechsung *Computational Complexity*. D. Reidel Publishing Company. 1986.

[76] M.K. Yntema Cap Expressions for Context-Free Languages. *Information and Control* 18, 311-318. 1971.

[77] T. Yokomori Inductive Inference of Context-free Languages Based on Context-free Expressions. *International J. Computer Math.*, Vol 24, pp 115-140. 1988

[78] H. Yoo, K. Hashiguchi Extended automata-like regular expressions of star degree at most (2,1). *Theoretical Computer Science* 88, 351-363. 1991.