Estudio de Relaciones Entre la Teoría de Autómatas, la Lógica de Segundo Orden y el μ - Cálculo

por

Nicolás Cardozo Álvarez

Una tesis
presentada al departamento
de Matemáticas
como parte de los requisitos
para el grado de
Matemático

Director: Maricarmen Martinez Baldares

Universidad de los Andes
Bogotá, Colombia
Mayo, 2008

Índice general

1. Autómatas. **2**
1.1. Teoría de Autómatas Sobre Objetos Finitos 2
1.1.1. Autómatas Determinísticos 2
1.1.2. Autómatas No Determinísticos 4
1.1.3. Notas Sobre Lenguajes Regulares 7
1.2. Autómatas Sobre Objetos Infinitos 11
1.2.1. ω-Autómatas 11
1.2.2. Autómatas Alternantes 13

2. Autómatas Finitos y Lógica de Segundo Orden **17**
2.1. Lógica Monádica de Segundo Orden 17

3. μ-Cálculo y Autómatas Alternantes **23**
3.1. μ-Cálculo y Autómatas Alternantes 27

Bibliografía **36**

Introducción

En este documento se realizará una aproximación al estudio de lenguajes definidos por medio de, ya sea una lógica, o un autómata; adicionalmente se trabajará sobre las relaciones existentes entre estas dos definiciones. Este trabajo presenta una recopilación de los resultados más característicos de los últimos ocho años, en el desarrollo de estas teorías, así como algunos de los resultados clásicos más importantes, todo esto basado en el trabajo presentado por Igor Walukiewics en [1].

Inicialmente se presentará el estudio de la teoría de lenguajes regulares. Debido a que los autómatas son utilizados comúnmente para el reconocimiento y producción de estos lenguajes, de esencial interés en la computación para definir máquinas de estados, se presentarán los resultados más relevantes de cerradura de autómatas, los cuales serán útiles para el posterior desarrollo de las relaciones con la lógica.

Por otra parte también se puede construir una teoría a partir de una definición lógica de un lenguaje y las propiedades que de ella se desatan. Para cada una de las estructuras de autómatas presentada, se estudia la relación del lenguaje generado por el autómata, con el lenguaje generado por una sentencia de una lógica. Concretamente, relacionaremos el lenguaje generado por una sentencia de la lógica monádica de segundo orden con los autómatas de estado finitos. Después, para realizar la conexión existente de la lógica con los autómatas alternantes se dará una introducción al μ-cálculo, que es una extensión de la lógica temporal.

Finalmente se observará como, por medio de las traducciones presentadas, es posible obtener resultados de la lógica a partir de resultados existentes en la teoría de autómatas y viceversa. Específicamente se presentarán los usos de estas traducciones para mostrar resultados de satisfacibilidad de la lógica monádica de segundo orden, propiedades de cerradura de los autómatas alternantes y el problema del lenguaje vacío.

Capítulo 1

Autómatas.

En este capítulo se introducirán distintas estructuras de autómatas para objetos finitos e infinitos y los lenguajes que se definen para cada una de estas estructuras.
Adicionalmente se presentaran algunos resultados clásicos para estos lenguajes.

1.1. Teoría de Autómatas Sobre Objetos Finitos

Antes de entrar a definir el concepto de autómata daremos una noción estándar de lenguaje, que luego generalizaremos. En su versión más básica un lenguaje es un conjunto de palabras $L \subseteq \Sigma^*$, donde Σ es un alfabeto y Σ^* es el conjuto de todas las cadenas finitas formadas por caracteres de Σ.
Existen distintos tipos de autómatas, los cuales están definidos de acuerdo a su propósito. En esta sección empezaremos introduciendo los autómatas más básicos (autómatas finitos) y luego hablaremos de versiones más sofisticadas de autómatas, asociados también a versiones más sofisiticadas del concepto de lenguaje.

1.1.1. Autómatas Determinísticos

Un autómata de estados finitos, como su nombre lo dice, es esencialmente un grafo finito donde los nodos representan estados dentro de un proceso y los arcos representan cambio de estados causados por una acción (lectura de un caracter).

Definición 1.1.1. Formalmente un autómata de estados finitos se define como una tupla $\mathcal{A} = \langle Q, \Sigma, q^0, F, \delta \rangle$ donde:

1. Q es un conjunto finito y no vacío de estados.
2. Σ es un alfabeto finito.
3. $q^0 \in$ Q es el estado inicial.
4. $F \subseteq Q$ es un conjunto de estados finales.
5. $\delta : (Q \times \Sigma) \longrightarrow Q$ es la función parcial de transiciones.

Si bien δ está definida como una función parcial, puesto que no necesariamente se está asignando valores a cada par $(q, a) \in Q \times \Sigma$, siempre es posible realizar una modificación de la función para que sea función total agregando un nuevo estado $\perp$ y extendiendo δ con $\delta(q, a) = \perp$ siempre que (q, a) no estuviera en el dominio de la función parcial original. El estado $\perp$ juega el papel de un sumidero del autómata. Estas modificaciones no alteran la estructura del autómata ni el lenguaje que acepta.
Para los autómatas determinísticos es posible utilizar más de un estado inicial, pero para los efectos de este documento solo utilizaremos un único estado inicial. Al igual que con la función de transición se puede realizar una modificación del autómata para que solo se utilice un estado inicial, generando un nuevo estado (ahora inicial) y haciendo transiciones vacías a los estados anteriormente iniciales. El resultado de esta transformación es un autómata no determinístico, un tipo de autómata que se introducirá en la sección siguiente [2]. Este autómata no determinístico se puede otra vez convertir en determinístico, como se verá en la proposición 1.1.7

Intuitivamente, una ejecución sobre un autómata es la lectura secuencial de los caracteres de una palabra finita $\omega \in \Sigma^*$. Diremos que una ejecución es exitosa si al final de la lectura nos encontramos en un estado final del autómata. Formalmente definimos una ejecución como sigue. Notaremos a $|\omega|$ como la longitud de la palabra.

Definición 1.1.2. Formalmente, una ejecución[1] de una palabra $\omega = a_0 a_1 \dots a_n \in \Sigma^*$ en $\mathcal{A} = \langle Q, \Sigma, q^0, F, \delta \rangle$ es una secuencia de estados $q_0 q_1 \dots q_{n+1}$ tal que $q_0 = q^0$ y para todo $i \geq 0,\ \delta(q_i, a_i) = q_{i+1}$. La ejecución se dirá exitosa si $q_{n+1} \in F$ es estado final. En este caso diremos que la palabra es reconocida por el autómata.
Una ejecución de $\mathcal{A}$ es una secuencia de estados que es una ejecución de una palabra.
Generalizando lo anterior diremos que q_j es un ω-sucesor de q_i (lo cual denotamos $q_i \overset{w}{\Longrightarrow} q_j$) si existe una secuencia $q_0 q_1 \dots q_{n+1}$ tal que $q^0 = q_i$, $q_{n+1} = q_j$ y para todo $0 < k < j$, se tiene que $\delta(q_{k-1}, a_k) = q_k$.

Definición 1.1.3. Notaremos como $\mathcal{L}(\mathcal{A})$ al lenguaje generado por el autómata $\mathcal{A}$, es decir el conjunto de palabras cuyas ejecuciones son exitosas en el autómata. Si $\omega \in \mathcal{L}(\mathcal{A})$, diremos que ω es reconocida por $\mathcal{A}$.

Ejemplo 1.1.4. Considere el autómata de la figura 1.1 sobre el alfabeto $\Sigma = \{a, b\}$, definido por:

$$\begin{array}{lll} Q = \{0, 1, 2\} & \delta(0, a) = 0 & \delta(0, b) = 1 \\ q^0 = 0 & \delta(1, a) = 1 & \delta(1, b) = 2 \\ F = \{1, 2\} & \delta(2, a) = 2 & \delta(2, b) = 0 \end{array}$$

Éste es un autómata que reconoce las palabras con una cantidad de $b's$ que no es divisible por 3.

[1] Este término también se puede encontrar dentro de la literatura con el nombre de corrida. A lo largo de este texto utilizaremos indistintamente los términos corrida y ejecución.

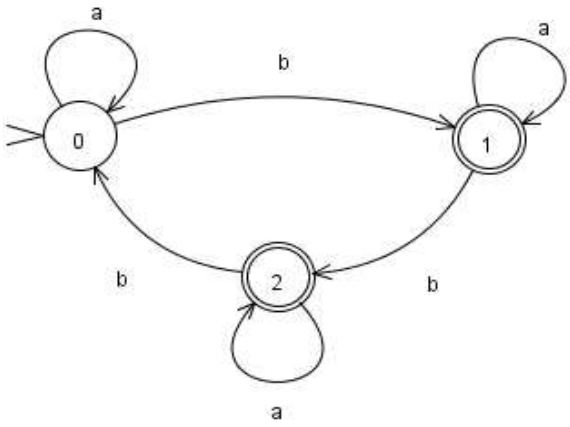


Figura 1.1: Diagrama de estados

Convención del diagrama: En los diagramas de estados de este documento, el estado marcado con $>$ representa el estado inicial del autómata y los estados marcados con un doble círculo representan los estados finales.

En este ejemplo, es posible deducir que palabras que son reconocidas por el autómata únicamente revisando su diagrama de estados; desafortunadamente no para todos los casos tendremos esta facilidad, es por ello que a partir de un autómata dado quisieramos poder encontrar, de manera metódica, el lenguaje generado por el mismo.

1.1.2. Autómatas No Determinísticos

Los autómatas no determinísticos son una extensión de los autómatas determinísticos, que se obtiene permitiendo que las transiciones entre estados no se den necesariamente por una función, sino por una relación. Así, de un mismo estado se puede en principio tener varias opciones para pasar a distintos estados utilizando un mismo símbolo del alfabeto. Además permitiremos transiciones vacías, es decir, será posible cambiar de estado leyendo el caracter "vacio" del alfabeto el cual denotaremos por λ. Debido a estas características, puede haber distintas ejecuciones en el autómata para una misma palabra.

Definición 1.1.5. Un autómata no determinístico es una tupla $\mathcal{A} = \langle Q, \Sigma, q^0, F, \delta \rangle$ donde:

1. Q es un conjunto finito no vacío de estados.
2. Σ es un alfabeto finito.
3. q^0 es el estado inicial[2].
4. $F \subseteq Q$ es el conjunto de estados finales
5. $\delta \subseteq (Q \times \Sigma) \times Q$ es la relación de transición.

[2]Dentro de la literatura se encuentran definiciones de autómatas no determinísticos que utilizan un conjunto de estados iniciales; dado que no se gana ninguna generalidad con esta noción y no facilitarán nuestro trabajo, no se utilizará esta característica en este documento.

Como antes, diremos que un autómata no determinístico acepta una palabra $\omega \in \Sigma^*$ si existe una ejecución sobre ω en $\mathcal{A}$, que termine en un estado final.

Ejemplo 1.1.6. Considere el siguiente autómata $\mathcal{A}$ sobre el alfabeto $\Sigma = \{a, b\}$ definido por:

$$
\begin{array}{ll}
Q = \{0,1,2,3,4,5\} & \delta(0,a) = 0 \quad \delta(5,a) = 5 \\
q^0 = 0 & \delta(0,b) = 1 \quad \delta(3,a) = 3 \\
F = \{5\} & \delta(1,a) = \{2,3\} \quad \delta(2,b) = 4 \\
 & \delta(2,\lambda) = 2 \quad \delta(2,a) = 5 \\
 & \delta(3,b) = 4
\end{array}
$$

A lo largo del documento abusaremos de la notación de función para representar las transiciones definidas por estas relaciones. Por ejemplo en la definición anterior de δ utilizamos la notación $\delta(1,a) = \{2,3\}$ para representar $(1,a,2) \in \delta$ y $(1,a,3) \in \delta$.
En la Figura 1.2 podemos encontrar la representación gráfica del autómata anterior, el cual reconoce las palabras que pertenecen al lenguaje generado por a^*baa^+.
La descripción del lenguaje como a^*baa^+ utiliza un formalismo para expresar lenguajes llamado expresiones regulares[4][3][5]. Aquí, a^* expresa que a puede ocurrir cero o más veces, ab expresa que b ocurre inmediatamente despues de a, $a + b$ expresa que puede ocurrir b ó a. Finalmente, a^+ expresa que a ocurre una o más veces.

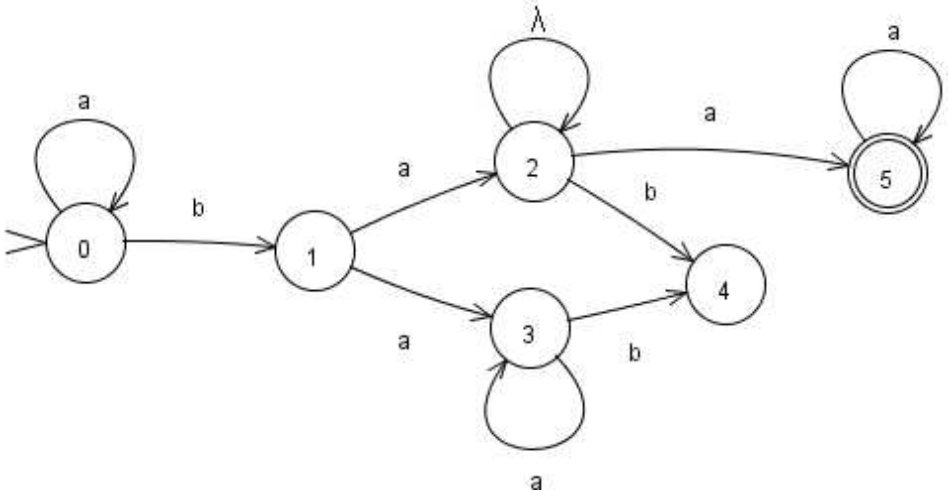


Figura 1.2: Diagrama de estados

Si tomamos $\omega = abaa$ podemos encontrar las siguientes ejecuciones (entre muchas otras que también existen).

$$
\begin{aligned}
r_1 &= 0,0,1,2,5 \\
r_2 &= 0,0,1,3,3 \\
r_3 &= 0,0,1,2,2,2,2,5
\end{aligned}
$$

De aquí vemos que la cadena es aceptada por $\mathcal{A}$, pues las ejecuciones r_1 y r_3 son testigos de esto.
Ahora considere la cadena $\omega = abab$

$$r_1 = 0, 0, 1, 2, 4$$
$$r_2 = 0, 0, 1, 3, 4$$

Esta cadena no es aceptada por el autómata, pues con el segmento inicial *aba* es posible llegar a los estados 2 ó 3, pero no es posible llegar de allí al estado final 5, pues para esto seria necesario continuar con una a que la palabra no contiene. En resumen, para este autómata $\mathcal{A}$ tenemos que $abaa \in \mathcal{L}(\mathcal{A})$ pero $abab \notin \mathcal{L}(\mathcal{A})$.

Es natural pensar que este tipo de autómatas es más general que los determinísticos, pero es posible mostrar que en realidad todo autómata no determinístico es equivalente a uno determinístico (aceptan el mismo lenguaje).

Proposición 1.1.7. *Para todo autómata no determinístico existe un autómata deterministico equivalente.*

Demostración. Dado $\mathcal{A}_{nd} = \langle Q_{nd}, \Sigma, q^{nd}, F_{nd}, \delta_{nd} \rangle$ autómata no determinístico, para cada $\omega \in \Sigma^*$ sea $X_{[\omega]} = \{q \in Q_{nd} | q^{nd} \overset{w}{\Longrightarrow} q\}$. Definimos el autómata determinístico

$$\mathcal{A}_d = \left\langle Q_d, \Sigma, X_{[\lambda]}, F_d, \delta \right\rangle$$

a partir de $\mathcal{A}_{nd}$ así:

1. $X_{[\lambda]} = \{q \in Q_{nd} | q^{nd} \overset{\lambda}{\Longrightarrow} q\}$
2. $Q_d = \mathcal{P}(Q_{nd})$
3. $F_d = \{X \in Q_d | X \cap F_{nd} \neq \phi\}$
4. $\delta(X, s) = X'$ donde $X' = \{q' \in Q_{nd} | q \overset{s}{\Longrightarrow} q' \text{ para algún } q \in X\}$

Mostraremos que $\mathcal{L}(\mathcal{A}_d) = \mathcal{L}(\mathcal{A}_{nd})$.
($\supseteq$)Sea $\omega \in \mathcal{L}(\mathcal{A}_{nd})$. Entonces el estado $X_{[\omega]} \in Q_d$ es final pues $X_{[\omega]} \cap F_{nd} \neq \phi$. Es decir $\exists q \in F_{nd}$ tal que $q^{nd} \overset{\omega}{\Longrightarrow} q$.
Mostraremos por inducción sobre la longitud de la palabra ω que para todo $\omega \in \Sigma^*$ segmento inicial de alguna cadena $v \in \mathcal{L}(\mathcal{A}_{nd})$, $X_{[\lambda]} \overset{\omega}{\Longrightarrow} X_{[\omega]}$.
Caso Base $\omega = \lambda$: De la definición de $X_{[\lambda]}$ tenemos que $X_{[\lambda]} \overset{\lambda}{\Longrightarrow} X_{[\lambda]}$
Hipótesis de inducción: Suponga que para τ segmento inicial de alguna palabra en $\mathcal{L}(\mathcal{A}_{nd})$, se tiene el resultado: $X_{[\lambda]} \overset{\tau}{\Longrightarrow} X_{[\tau]}$.
Tome $\omega := \tau s$ para $s \in \Sigma$ y suponga que ω es segmento inicial de alguna $v \in \mathcal{L}(\mathcal{A}_{nd})$.
Por hipótesis de inducción sabemos que existe $q' \in Q_{nd}$ t.q. $q^0 \overset{\tau}{\Longrightarrow} q'$. Pero como $v \in \mathcal{L}(\mathcal{A}_{nd})$, para al menos uno de esos q' existe $q \in Q_{nd}$ t.q $q' \overset{s}{\longrightarrow} q$ y por definición se tiene que $q \in X_{[\omega]}$.
Utilizando la propiedad 4 tenemos que $\delta(X_{[\tau]}, s) \ni X_{[\omega]}$. Por lo tanto $X_{[\lambda]} \overset{\omega}{\Longrightarrow} X_{[\omega]}$.
($\subseteq$) Sea $\omega \in \mathcal{L}(\mathcal{A}_d)$.
Si $\omega = \lambda$ entonces $X_{[\lambda]} \in F_d$, es decir, existe $q \in Q_{nd}$ t.q. $q^{nd} \overset{\lambda}{\longrightarrow} q$ con $q \in F_{nd}$ y por lo tanto $\omega \in \mathcal{L}(\mathcal{A}_{nd})$. Si $\omega = a_0 a_1 \dots a_n$ con $n \geq 1$, existe una secuencia de estados $X_{[0]} X_{[1]} \dots X_{[n+1]}$ con $X_{[n+1]} \in F_d$ y $X_{[i]} = \delta_d(X_{[i-1]}, a_{i-1})$ para $i = 1, \dots, n+1$.
De la definición de F_d tenemos que existe $q_{n+1} \in X_{[n+1]}$ con $q_{n+1} \in F_{nd}$. ahora como $X_{[n+1]}$ es

a_n-sucesor de $X_{[n]}$ existe $q_n \in X_{[n]}$ tal que $q_n \xrightarrow{a_n} q_{n+1}$. De la misma forma es posible tomar $q_i \in X_{[i]}$ formando una secuencia de estados $q_0 q_1 \dots q_{n+1}$ en Q_{nd} de tal forma que $q_0 \xrightarrow{\omega} q_{n+1}$. □

De la construcción anterior podemos ver que al realizar la conversión a un autómata determinístico el número de estados se aumenta hasta $2^{|Q_{nd}|}$. En general, los autómatas no determinísticos son más fáciles de diseñar, pero realmente no agregan poder de expresión. Por otra parte los algoritmos para decidir si $\omega \in \mathcal{L}(\mathcal{A}_{nd})$ son más complejos que los de $\mathcal{L}(\mathcal{A}_d)$, pues requieren búsquedas y backtracking. Los algoritmos en $\mathcal{A}_d$ son triviales, pero el espacio ocupado por el autómata es mayor.

1.1.3. Notas Sobre Lenguajes Regulares

Diremos que un lenguaje $\mathcal{L}$ es regular si existe un autómata finito que lo reconoce. Note que la condición de determinismo sobre el autómata no afecta en absoluto esta noción de lenguaje a la luz de la Proposición 1.1.7.
Únicamente estudiaremos tres resultados acerca de los lenguajes regulares, que serán utilizados para el desarrollo posterior de algunos resultados.

Lema 1.1.8. *La unión de lenguajes regulares es regular.*

Demostración. Sean $\mathcal{L}_1$ y $\mathcal{L}_2$ los lenguajes regulares reconocidos por autómatas determinísticos $\mathcal{A}_1 = \langle Q_1, \Sigma, F_1, q^1, \delta_1 \rangle$ y $\mathcal{A}_2 = \langle Q_2, \Sigma, q^2, F_2, \delta_2 \rangle$ respectivamente.
Construya el autómata no determinístico $\mathcal{A}_3 = \langle Q_3, \Sigma, q^0, F_3, \delta_3 \rangle$ donde q^0 es un nuevo estado y δ_3 está definida por:

$$\delta_3(q,a) = \begin{cases} \delta_1(q,a) & \text{si } q \in Q_1 \\ \delta_2(q,a) & \text{si } q \in Q_2 \\ q^1 & \text{si } q = q^0 \text{ y } a = \lambda \\ q^2 & \text{si } q = q^0 \text{ y } a = \lambda \end{cases}$$

Con $Q_3 = Q_1 \cup Q_2 \cup \{q^0\}$ y $F_3 = F_1 \cup F_2$. Sin pérdida de generalidad hemos supuesto que $Q_1 \cap Q_2 \neq \phi$; en el caso que ésto no suceda se debe realizar primero un renombramiento de los estados en alguno de los dos conjuntos, antes de hacer la unión. □

Ejemplo 1.1.9. Considere los autómatas $\mathcal{A}_1$ y $\mathcal{A}_2$ que aceptan palabras con exactamente una b y exactamente una a, respectivamente. Estos autómatas etán ilustrados en la Figura 1.3. El autómata de la Figura 1.4 tiene como lenguaje $\mathcal{L}(\mathcal{A}_1) \cup \mathcal{L}(\mathcal{A}_2)$. Note que en este caso como $0, 1, 2$ son estados tanto de $\mathcal{A}_1$ como de $\mathcal{A}_2$, se renombraron primero los estados de $\mathcal{A}_2$ a $4, 5, 6$ antes de realizar la unión.

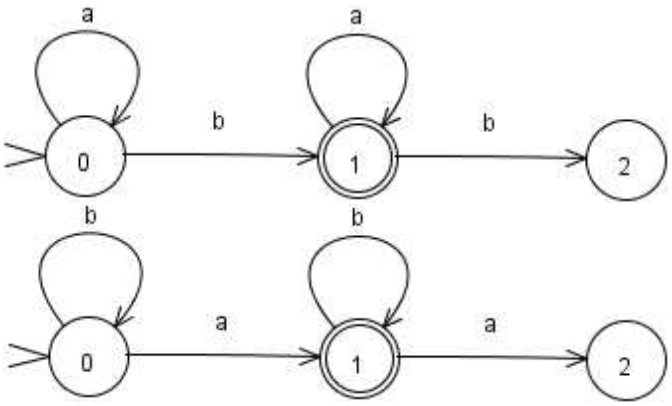


Figura 1.3: Arriba:$\mathcal{A}_1$ Abajo:$\mathcal{A}_2$

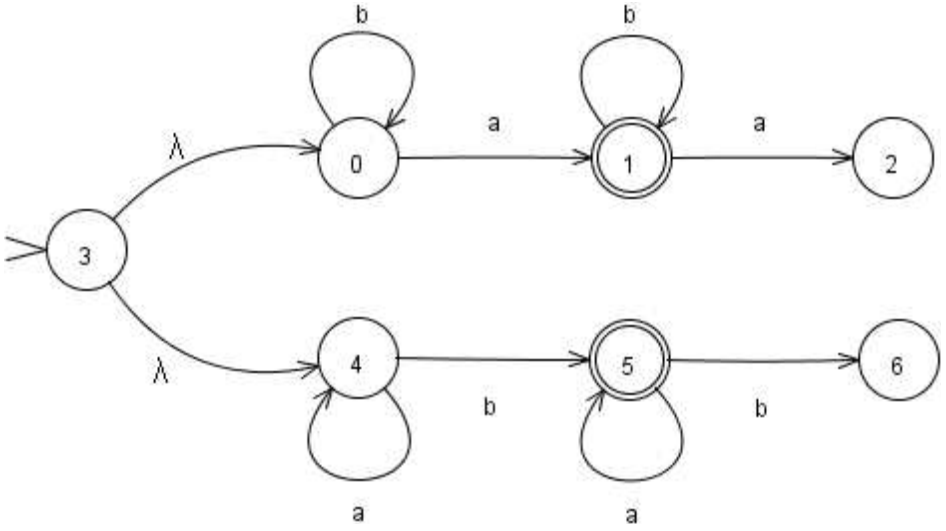


Figura 1.4: $\mathcal{A} = \mathcal{A}_1 \cup \mathcal{A}_2$

Lema 1.1.10. *El complemento de un lenguaje regular es también regular.*

Demostración. Sea $\mathcal{A} = \langle Q, \Sigma, q^0, F, \delta \rangle$ un autómata determinístico y sea $\mathcal{L}(\mathcal{A})$ el lenguaje reconocido por $\mathcal{A}$ [3]. Construiremos un autómata determinístico $\mathcal{A}^c$ que acepta el lenguaje $\Sigma^* \setminus \mathcal{L}(\mathcal{A})$.
Sea $\mathcal{A}^c = \langle Q^c, \Sigma, q^0, F^c, \delta^c \rangle$ donde $Q^c := Q \cup \{\perp\}$ y $F^c = (Q \setminus F) \cup \{\perp\}$. Aquí
$\perp$ se introduce como un nuevo estado de error y

$$\delta^c(q,a) = \begin{cases} \delta(q,a) & \text{si } (q,a) \in Dom(\delta) \\ \perp & \text{si } (q,a) \notin Dom(\delta) \text{ ó } q = \perp \end{cases}$$

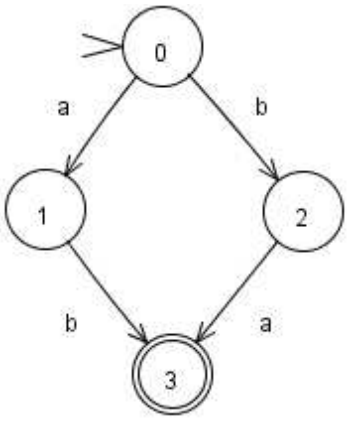
Queremos ver que $\mathcal{L}(\mathcal{A}^c) = \mathcal{L}^c(\mathcal{A})$:
$(\mathcal{L}(\mathcal{A}^c) \subseteq \mathcal{L}^c(\mathcal{A}))$: Sea $\omega \in \mathcal{L}(\mathcal{A}^c)$ con $|\omega| = n$.

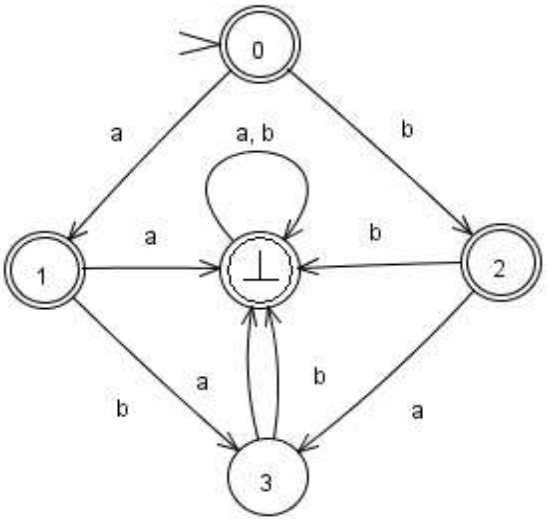
[3] Si $\mathcal{A}$ no es determinístico, es necesario realizar el procedimiento de la proposición 1.1.7

Por definición de $\mathcal{L}(\mathcal{A}^c)$, existe $r = q^0 \xrightarrow{\omega_0} q_1 \ldots \xrightarrow{\omega_n} q_{n+1}$ ejecución de ω sobre $\mathcal{A}^c$ t.q $q_{n+1} \in F^c$.
Por definición de F^c, $q_{n+1} \notin F$. Dado que $\mathcal{A}^c$ es determinístico, no existe una ejecución $r' \neq r$ t.q $r' = \xrightarrow{\omega_0} q_1 \ldots \xrightarrow{\omega_n} q'$ con $q' \in F$ que acepte ω por lo tanto se tiene $\omega \in \mathcal{L}^c(\mathcal{A})$.
$(\mathcal{L}(\mathcal{A}^c) \supseteq \mathcal{L}^c(\mathcal{A}))$: Sea $\omega \in \mathcal{L}^c(\mathcal{A})$ con $|\omega| = n$.
Para toda $r = q^0 \xrightarrow{\omega_0} q_1 \ldots \xrightarrow{\omega_n} q_{n+1}$ ejecución de ω sobre $\mathcal{A}$ tal que $q_{n+1} \notin F$;
por definición de F^c, $q_{n+1} \in F^c$. Ahora, $q^0 q_1 \ldots q_{n+1}$ es una secuencia de estados de $\mathcal{A}^c$. Nuevamente utilizando que $\mathcal{A}^c$ es determinístico, no existe ninguna ejecución $r' = \xrightarrow{\omega_0} q_1 \ldots \xrightarrow{\omega_n} q'$ con $q' \in F$ que acepte ω luego $\omega \in \mathcal{L}(\mathcal{A}^c)$.

□

Ejemplo 1.1.11. El autómata $\mathcal{A}$ que aparece a la izquierda de la figura de abajo reconoce el lenguaje $\mathcal{L}(\mathcal{A}) = ab + ba$. El autómata de la derecha es su complemento, reconociendo el lenguaje $\mathcal{L}(\mathcal{A})^c = \Sigma^* \setminus \{ab, ba\}$





Definición 1.1.12. Dado un alfabeto $\Sigma = \Sigma_1 \times \Sigma_2 \times \cdots \times \Sigma_n$ y una secuencia I de la forma $1 < i_1 < i_2 < \ldots < i_k < n+1$ con $k > 0$, usaremos Σ_I para denotar el alfabeto $\Sigma_{i_1} \times \Sigma_{i_2} \times \cdots \times \Sigma_{i_n}$. Para cada letra $t = (a_1, a_2, \ldots, a_n) \in \Sigma$, definimos $\pi_I(t)$ como el elemento $(a_{i_1}, a_{i_2}, \ldots, a_{i_k})$ de Σ_I. Sea $p = t_1 t_2 \ldots t_s$ un elemento de Σ^*. Así, cada t_i es una tupla de longitud n, es decir, una letra de Σ. Definimos la proyección de la palabra p sobre I como la palabra $\pi_I(p) = \pi_I(t_1)\pi_I(t_2)\ldots\pi_I(t_n)$, que es un elemento de Σ_I^*. Finalmente, dado un lenguaje $\mathcal{L}$ sobre el alfabeto Σ, definimos la proyección de $\mathcal{L}$ sobre I como el lenguaje sobre Σ_I dado por $\pi_i(\mathcal{L}) = \{\pi_I(p) \mid p \in \mathcal{L}\}$.

Lema 1.1.13. *Proyección de lenguajes regulares es regular.*

Demostración. Sea $\mathcal{A} = \langle Q, \Sigma_1 \times \Sigma_2 \times \cdots \times \Sigma_n, q^0, F, \delta\rangle$ un autómata que reconoce un lenguaje $\mathcal{L}(\mathcal{A})$ de palabras de la forma $\vec{a} = \vec{a_1}\ldots\vec{a_n}$ donde cada $\vec{a_i}$ es de la forma $(a_1, a_2, \ldots, a_n)$ con $a_i \in \Sigma_i$. Sea $I = \{i_1, \ldots, i_k\}$ conjunto de indices. Definimos

$$\mathcal{A}_{\pi_I} = \langle Q, \Sigma_I, q^0, F, \delta_{\pi_I}\rangle \text{ con } \delta_{\pi_I}(q, \vec{b}) = \bigcup\{\delta(q, \vec{a}) | \pi_I(\vec{a}) = \vec{b}\}$$

De esta construcción es facil ver que $\pi_I(\mathcal{L}(\mathcal{A})) = \mathcal{L}(\mathcal{A}_{\pi_I})$. □

Ejemplo 1.1.14. Proyección de un autómata $\mathcal{A}_1 = \langle Q, \{0,1\} \times \Sigma, q^0, F, \delta\rangle$ que reconoce el lenguaje $\mathcal{L}(\mathcal{A}_1) = (0,a)(0,b) + (1,a)(1,b)$. Su proyección sobre la segunda componente, $\mathcal{A}_2$ reconoce el lenguaje $\mathcal{L} = ab$

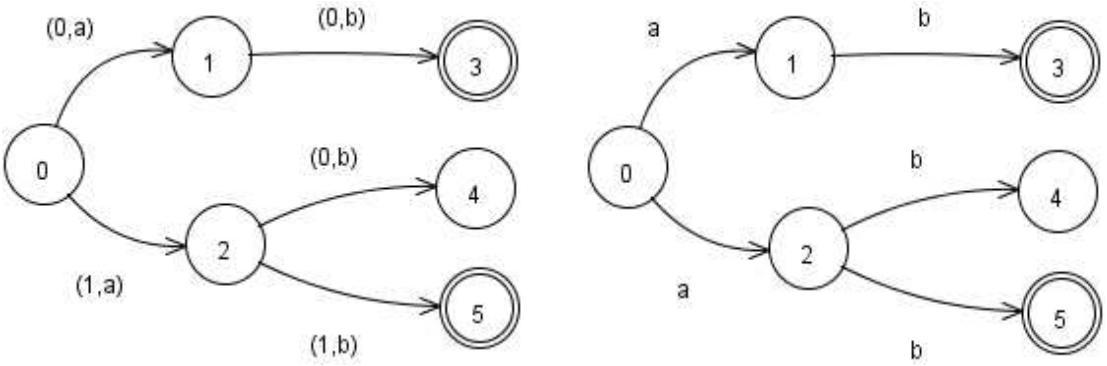


Figura 1.5: izq:$\mathcal{A}_1$ autómata original der: $\mathcal{A}_2$ = proyección de $\mathcal{A}_1$

Observación 1: En el caso en el que existan 2 caminos distintos para una misma secuencia de caracteres de Σ digamos $a_0, \ldots a_n$ con $q_0 \xrightarrow{a_0} q_1 \ldots \xrightarrow{a_n} q_{n+1}$ y $q'_0 \xrightarrow{a_0} q'_1 \ldots \xrightarrow{a_n} q'_{n+1}$ se pueden contraer estos caminos en un único camino (preservando los estados finales). La Figura 1.6 muestra el resultado de realizar este tipo de contracciones en el autómata $\mathcal{A}_2$ de la Figura 1.5.

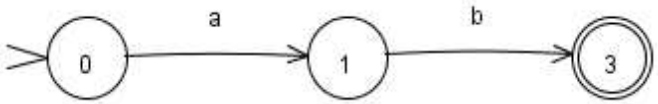


Figura 1.6: $\mathcal{A}$ reconoce el lenguaje $\mathcal{L} = ab$

1.2. Autómatas Sobre Objetos Infinitos

En esta sección generalizamos los autómatas finitos de forma tal que puedan decidir sobre estructuras (palabras o árboles) infinitas. Así mismo extenderemos la noción de lenguaje a lenguajes formados por árboles posiblemente infinitos.

1.2.1. ω-Autómatas

Los ω-autómatas son autómatas de estados finitos con capacidad de reconocer palabras de longitud infinita.

Definición 1.2.1. Un autómata que permita reconocimiento de palabras infinitas tiene la forma $\mathcal{A}_\omega = \langle Q, \Sigma, q^0, \delta, Acc \rangle$, donde Q es un conjunto de estados finitos, $\delta \subseteq Q \times \Sigma \times Q$ y $Acc \subseteq Q^\omega$ es un conjunto de las ejecuciones aceptadas por el autómata. Aquí, Q^ω es el conjunto de secuencias infinitas de estados.

El concepto de ejecución para los ω-autómatas es el mismo introducido anteriormente, la única diferencia es que ahora se utilizan secuencias infinitas de estados.
Note que el autómata puede ser no determinístico, dado que la función de transición puede

especificar varias posibles transiciones para un estado y caracter del alfabeto.

Las cadenas ω aceptadas por el autómata $\mathcal{A}_\omega$ son aquellas para las cuales alguna de sus ejecuciones en $\mathcal{A}_\omega$ esta en Acc.

Existen varias definiciones para el conjunto de aceptación Acc. En este documento utilizaremos la definida por Andrzej Mostowski. Para definir dicha condición, es necesario introducir un concepto auxiliar.

Definición 1.2.2. Dada una ejecución r sobre el autómata $\mathcal{A}_\omega$, denotaremos por $In(r) = \{q \in Q | r(i) = q \text{ para infinitos valores de } i\}$ al conjunto de estados que ocurren infinitas veces dentro de r. Aquí, $r(i)$ es el i-ésimo estado en la secuencia r.

Definición 1.2.3. La condición de Mostowski ésta dada por $Acc = \{r | \text{ mín}\{\Omega(In(r))\} \text{ es par } \}$ donde $\Omega : Q \longrightarrow \mathbb{N}$.

La condición de Mostowski generaliza la que puede verse como la más natural de las condiciones de aceptación para los autómatas con ejecuciones infinitas, que sería fijar un subconjunto $S \subseteq Q^\omega$ de estados y decir que una palabra infinita es aceptada si existe alguna ejecución r de ella en $\mathcal{A}_\omega$ tal que $In(r) \cap S \neq \phi$, lo cual podría pensarse como "'quedarse"' en un "'estado final"'. Este es el caso en que $\Omega : Q \longrightarrow \mathbb{N}$ está dada por la función característica de S.
A este caso particular se le conoce como condición de Büchi. La condición de Mostowski es una noción estrictamente más general que la de Büchi. La idea es que Ω hace dos cosas:

(i) Estratifica los estados de manera que un estado con nivel $\Omega(q)$ bajo es más deseable.

(ii) Particiona los estados Q en "buenos"(si $\Omega(q)$ es par) y "malos"(si $\Omega(q)$ no es par) .

Lo que se pide para que una ejecución de una palabra infinita ω en $\mathcal{A}$ sea testigo de que $\omega \in \mathcal{L}(\mathcal{A})$ es que el nivel de esa ejecución (dado por mín$(\Omega(In(r)))$) sea par.
No profundizaremos en esta idea aquí, simplemente nos interesa la intuición. Encontraremos y profundizaremos en esto en el contexto de autómatas para árboles, que introduciremos en la sección siguiente.

Ejemplo 1.2.4. Considere el autómata $\mathcal{A} = \langle \{0,1,2\}, \{a,b\}, 0, Acc, \delta \rangle$ de la Figura 1.7, que reconoce el lenguaje $\mathcal{L} = a(bba + ba)^*(ba)^\omega$, es decir, el lenguaje de palabras que empiezan en a, continua con una sucesión finita de bba's y/o ba's y termina con una sucesión infinita de ba's

δ se define como: $\delta(0,a) = 1$, $\delta(1,b) = 2$, $\delta(2,a) = 1$, $\delta(2,b) = 0$

La condición de Mostowski esta dada por: $\Omega(0) = 1$, $\Omega(1) = 2$, $\Omega(2) = 2$

Note que para toda ejecución r, si $In(r) \neq \phi$, entonces $\{1,2\} \subseteq In(r)$. Los valores de la condición de Mostowski están indicados en la figura con negrillas por fuera del estado respectivo.

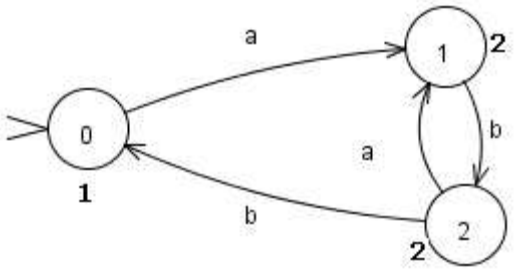


Figura 1.7: Autómata para el lenguaje $\mathcal{L}$

Suponga que $\omega \notin \mathcal{L}$. Si no hay ejecuciones de ω (por ejemplo si $\omega = a^\omega$). Entonces $\omega \notin \mathcal{L}(\mathcal{A})$. Ahora si ω si tiene ejecuciones, es porque ω pertenece al lenguaje $a(ba + bba)^\omega$, con infinitas repeticiones de bba, visitando el nodo 0 infinitas veces; por lo tanto mín$\{\Omega(In(r)))\} = \Omega(0) = 1$, entonces $\omega \notin \mathcal{L}(\mathcal{A})$.

1.2.2. Autómatas Alternantes

En las secciones anteriores introdujimos algunas clases de autómatas no determinísticos, los cuales permiten escoger, de entre varias posibilidades, el camino que se desea tomar para continuar una ejecución, dado el estado actual y el caracter que se recibe como entrada. Entonces es natural pensar en una máquina que sí permita la ejecución paralela de varios caminos posibles en algún punto. Los autómatas alternantes son precisamente artefactos que permiten estas dos clases de comportamiento, extendiendo así la clase de los autómatas no determinísticos.
Adicionalmente también es posible realizar ejecuciones sobre objetos infinitos (en este caso árboles) utilizando un concepto de aceptación similar al de los ω-autómatas.

En esta sección centraremos nuestra atención en el estudio de los autómatas alternantes que en lugar de aceptar palabras, aceptan árboles decorados con caracteres

Definición 1.2.5. Un árbol binario sobre un alfabeto Σ es una estructura de nodos
$\mathcal{T} = \langle \{0,1\}^*, \varepsilon, s_0, s_1, (P_a)_{a\in\Sigma} \rangle$ donde ε es la raíz del arbol, $P_a \subseteq \{0,1\}^*$,
$s_0,\ s_1 : \{0,1\}^* \to \{0,1\}^*$ son las dos funciones sucesor derecho e izquierdo de un nodo respectivamente, y para todo $v \in \{0,1\}^*$, existe una única $a \in \Sigma$ tal que $v \in P_a$.

Así, para cada $a \in \Sigma$ el predicado P_a identifica los nodos del árbol decorados con a.

Definición 1.2.6. Un autómata alternante de Mostowski sobre árboles tiene la forma
$\mathcal{A}_a = \langle Q, Q_\exists, Q_\forall, \Sigma, q^0, \delta, Acc \rangle$ donde Q es conjunto de estados finito y:

- $Q = Q_\exists \cup Q_\forall$ y $Q_\exists \cap Q_\forall = \phi$.
- q^0 es el estado inicial.
- $\delta : Q \times \Sigma \longrightarrow \mathcal{P}(Q \times \{0,1,\lambda\})$, es la función de transición.
- Acc está dado por la condición de Mostowski Ω. (como en la Definifión 1.2.3)

Para un autómata alternante las ejecuciones se llevan a cabo de la siguiente forma. Sea $\mathcal{T}$ el árbol que se está ejecutando. Suponga que nos hemos movido leyendo el árbol desde la raíz hasta un nodo v marcado con a y que antes de leer el caracter a el estado del autómata es q. Entonces, si q es un estado existencial ($q \in Q_\exists$), se escoge una de las transiciones $(q', d) \in \delta(q, a)$ para ser ejecutada. Si q es un estado universal ($q \in Q_\forall$), entonces se ejecutan todas las transiciones $\delta(q, a)$. Es decir, si hay k de estas transiciones, el proceso se ramifica en k ejecuciones del autómata, cada una empezando en el estado q' respectivo y dispuesto a leer el subárbol cuya raíz es el hijo de v indicado por la dirección d. (izquierdo si $d = 0$, derecho si $d = 1$, el mismo si $d = \lambda$).

En resumen, la corrida de un árbol decorado con caracteres se realiza desplazándose por los vértices del árbol y leyendo la coloración de cada uno. A partir de la decoración del vértice se ejecuta la transición de acuerdo a las reglas expuestas anteriormente. Si nos encontramos en el vértice v decorado por a y en el estado q se ejecutará al menos una de las transiciones $\delta(q, a) = (q', d)$ donde q' es el nuevo estado en el autómata y d define la dirección a tomar en el árbol (sucesor izquierdo si $d = 0$, sucesor derecho si $d = 1$ y mantenerse en el vértice si $d = \lambda$).

A continuación daremos la definición formal de la noción de ejecución.

Definición 1.2.7. Sea $\mathcal{T}$ = un árbol binario sobre Σ y $\mathcal{A}$ = un autómata alternante. Una ejecución r de $\mathcal{T}$ en $\mathcal{A}$ es una estructura de árbol de ramificación finita $r = \langle S, padre, f\rangle$ donde:

(i) $S \subseteq \mathbb{N}^*$ es un conjunto tal que para todo $\vec{n} \in S$, existe $k \in \mathbb{N}$ tal que $\vec{n}m \in S$ si y sólo si $m \leq k$. Además, λ, la cadena vacia, es elemento de S.

(ii) $padre \subseteq S \times S$ es la relación $\vec{n}\ padre\ \vec{m} \Leftrightarrow \vec{m} = \vec{n}k$ para algún $k \in \mathbb{N}$

(iii) $f : S \longrightarrow Q \times \{0, 1\}^*$ es la función que lleva el récord de qué nodo del árbol se está procesando, en qué estado.

La función debe satisfacer que si $f(\vec{n}) = (q, t)$ y $t \in P_a$, entonces:

- Si $q \in Q_\exists$, entonces $\vec{n}$ es padre de un único hijo $\vec{n}0$ y además $f(\vec{n}0) = (q', td)$ para algún $(q', d) \in \delta(q, a)$

- Si $q \in Q_\forall$ y $\delta(q, a) = \{(q_0, d_0), \ldots, (q_l, d_l)\}$, entonces los hijos de $\vec{n}$ son $\vec{n}0, \ldots, \vec{n}l$ y $f(\vec{n}i) = (q', td_i)$ para todo $i = 0, \ldots, l$.

Como el lector puede ver, la definición de corrida es intuitivamente sencilla pero técnicamente complicada de manejar y formalizar. Debido a esto, en el futuro apelaremos a una forma más sencilla, en términos de juegos, para describir cuando existe una corrida exitosa. Por ahora, basados en nuestra definición de corrida diremos lo siguiente.

Definición 1.2.8. Una corrida exitosa de $\mathcal{T}$ en $\mathcal{A}$ es una ejecución r tal que cada una de las ramas infinitas $\mathcal{R}$ del árbol r que empieza en la raíz satisface mín$(\Omega(In(\mathcal{R})))$ es par.

Ahora se introducirá la forma más sencilla de determinar cuándo existe una ejecución exitosa en $\mathcal{A}$ en un árbol $\mathcal{T}$.

Definición 1.2.9. Dado un autómata alternante $\mathcal{A}$ y un árbol $\mathcal{T}$, el juego $\mathcal{G}_{\mathcal{A},\sqcup}$ consiste en construir una secuencia de pasos $(q_0,t_0)(q_1,t_1)\ldots$ con $(q_i,t_i)\in Q\times\{0,1\}^*$ así:

Paso 0: Empezar con (q^0,ϵ)

Paso $n+1$: Teniendo $(q_0,t_0)(q_1,t_1)\ldots(q_n,t_n))$, si el turno corresponde al jugador $\exists$ si $q_n\in Q_\exists$. Si no, el turno es para $\forall$. El jugador que tiene el turno escoge (q_{n+1},t_nd) tal que $(q_n,a,(q_{n+1},d))\in\delta$, donde a es el único elemento de Σ tal que $t_n\in P_a$. En caso en que dicha tupla no exista el juego se da por terminado.

Diremos que $\exists$ gana el juego si al final se ha construido una secuencia infinita $(q_0,t_0)(q_1,t_1)\ldots$ tal que la secuencia $q_0q_1\ldots$ esta en el conjunto Acc del autómata.

Ejemplo 1.2.10. Considere el autómata alternante $\mathcal{A}$ en $\Sigma=\{a,b\}$ cuyo lenguaje es el árbol $\mathcal{T}$ con al menos una a.

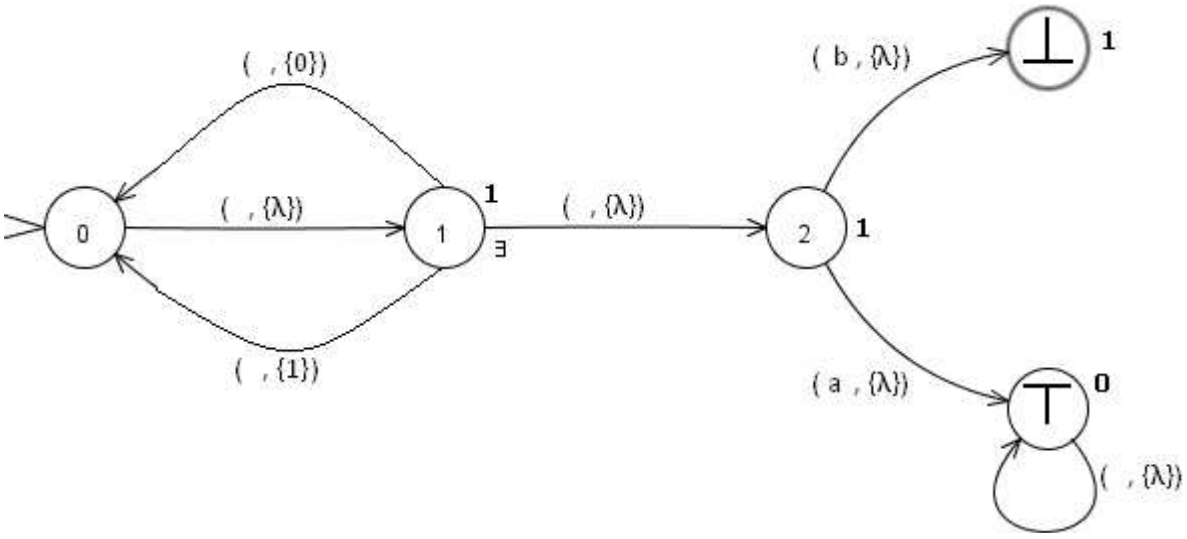


Figura 1.8: Autómata $\mathcal{A}$

Los estados marcados con el símbolo $\exists$ pertenecen al conjunto $Q_\exists$, los estados no marcados con este símbolo están en $Q_\forall$. Las flechas marcadas con $(\ ,\{d\})$ representan en realidad, todas las flechas $(a,d),$ con $a\in\Sigma$.

El conjunto de estados existenciales es:	$Q_\exists=\{1\}$
La condición de Mostowski es:	$\Omega(\top)=0\quad\Omega(q)=1$ si $q\neq\top$

Note que si $\mathcal{T}$ es un árbol de solo b's, entonces es imposible llegar al estado $\top$, luego toda ejecución r debe consistir de ramas infinitas donde $In(r)=\{0,1\}$ y $\text{mín}(\Omega(In(r)))=1$ que es impar. No hay estrategia ganadora para el jugador $\exists$.

Por otra parte si $\mathcal{T}$ tiene una a, la estrategia de $\exists$ es:

- Si el nodo t que se está analizando en el árbol está marcado con a, escoja la transición (a, λ), pasando del estado 1 al 2.
- Si el nodo t no ésta marcado con a y la distancia mínima de t a un nodo marcado con a es k, entonces: si hay un nodo a distancia k marcado con a en el sub-árbol izquierdo de t, escoge la transición $(b, 0)$, donde b es el caracter del nodo t. En caso contrario, escoja la transición $(b, 1)$.

Claramente, en finitos pasos se llega a un nodo marcado con a, pues la distancia a él se decrementa estrictamente cada vez que se pasa por el estado 1. Al encontrar el nodo con a, los siguientes estados de la ejecución son 2 y $\top$, y de éste último no se sale. Como $\Omega(\top) = 0$ es par, la ejecución es exitosa.

Capítulo 2

Autómatas Finitos y Lógica de Segundo Orden

En este capítulo se estudiará la relación existente entre la lógica de segundo orden y la teoría de autómatas finitos. Para ello comenzaremos con una breve explicación del lenguaje de la LMSO que se utilizará y luego se introducirán los modelos en los cuales se interpretan las fórmulas del lenguaje. Estos modelos $\mathcal{M}_\omega$ representan palabras ω. Así usaremos las formulas de LMSO para definir lenguajes $\mathcal{L}(\varphi) = \{\omega | \mathcal{M}_\omega \models \varphi\}$.
En la última parte veremos que los lenguajes definibles con la fórmulas de LMSO(Σ) (fórmulas sobre un alfabeto Σ) son precisamente los lenguajes regulares. Para ello, haremos una traducción en dos sentidos: para cada autómata finito encontramos una fórmula que lo describe y viceversa.

2.1. Lógica Monádica de Segundo Orden

La lógica monádica de segundo orden (LMSO) es una extensión de la lógica de primer orden que permite cuantificar sobre conjuntos de elementos. Al alfabeto de la LPO (Lógica de Primer Orden) agregamos una colección $Var_2 = \{X_1, X_2, \ldots\}$ de variables de segundo orden, donde cada X_i intuitivamente representa un conjunto de elementos.

Definición 2.1.1. Dado un alfabeto Σ, definimos el lenguaje LMSO(Σ) como
$\varphi := \ x_i = x_j \mid P_a(x) \mid X_i(x_j) \mid X_i \subseteq X_j \mid X_i \subseteq P_a \mid Suc(X_i, X_j) \mid \varphi \vee \psi \mid \neg\varphi \mid \exists x\varphi \mid \exists X\varphi$

donde:

- $P_a(x)$ es un predicado de primer orden, para cada $a \in \Sigma$.
- $X_i(x_j)$ es una fórmula atómica expresando que x_j está en el conjunto X_i.
- $X_i \subseteq X_j$ es una fórmula atómica expresando una relación de contenencia entre conjuntos.
- $X_i \subseteq P_a$ es una fórmula atómica expresando que X_i es subconjunto de P_a.

- $Suc(X_i, X_j)$ es una fórmula atómica expresando que X_i, X_j son singletons y el elemento único de X_j es sucesor inmediato del de X_i.
- $\exists X\varphi$ cuantificación sobre las variables de segundo orden.

La utilización de estos lenguaje se justifica debido a que nuestro interés se encuentra en describir propiedades de palabras $\omega \in \Sigma^*$. Para ello damos la siguiente definición.

Definición 2.1.2. Para cada palabra ω de longitud $|\omega| = n + 1$ definimos una estructura de segundo orden, $\mathcal{M}_\omega = \langle dom(w), P_a^\omega \rangle$
donde $dom(\omega) = \{0, \ldots, n\}$ es segmento inicial de los naturales y
$P_a^\omega = \{i \in dom(\omega) | \, \omega(i) = a\}$ es el conjunto de decoraciones de $\mathcal{M}$. En estas estructuras vemos a ω como la función $\omega : \{0, \ldots, n\} \longrightarrow \Sigma$ tal que $\omega(i) = a$ para algún $a \in \Sigma$.

Definición 2.1.3. La semántica de la LMSO sobre una estructura $\mathcal{M}_\omega$ con respecto a una valuación
$V : Var_2 \longrightarrow \mathcal{P}(dom(\omega))$ y $v : Var_1 \longrightarrow dom(\omega)$, es la siguiente:

- $\mathcal{M}_\omega, V, v \models x_i = x_j$ si $v(x_i) = v(x_j)$.
- $\mathcal{M}_\omega, V, v \models P_a(x_i)$ si $v(x_i) \in P_a^\omega$.
- $\mathcal{M}_\omega, V, v \models X_i(x_j)$ si $v(x_j) \in V(X_i)$.
- $\mathcal{M}_\omega, V, v \models \exists x\varphi$ si existe $k \in dom(\omega)$ tal que $\mathcal{M}_\omega, V, v[\frac{x}{k}] \models \varphi$ donde $v[\frac{x}{k}]$ es como v excepto que $v(x) = k$.
- $\mathcal{M}_\omega, V, v \models \exists X\varphi$ si existe $A \subseteq dom(\omega)$ tal que $\mathcal{M}_\omega, V[\frac{X}{A}], v \models \varphi$ donde $V[\frac{X}{A}]$ es como V excepto que $V(X) = A$.
- $\mathcal{M}_\omega, V, v \models \neg\varphi$ si $\mathcal{M}_\omega, V \not\models \varphi$
- $\mathcal{M}_\omega, V, v \models \psi \vee \varphi$ si $\mathcal{M}_\omega, V \models \varphi$ ó $\mathcal{M}_\omega, V \models \psi$
- $\mathcal{M}_\omega, V, v \models X \subseteq Y$ si $V(X) \subseteq V(Y)$.
- $\mathcal{M}_\omega, V, v \models X \subseteq P_a$ si $V(X) \subseteq P_a^\omega$.
- $\mathcal{M}_\omega, V, v \models Succ(X, Y)$ si $V(X)$ y $V(Y)$ son singletons $\{x\}, \{y\}$ respectivamente, y $x + 1 = y$.

Ejemplo 2.1.4. Para la palabra $\omega =$ casa; la estructura correspondiente $\mathcal{M}_\omega = \langle \{0, 1, 2, 3\}, (P_a)_{a \in \Sigma} \rangle$ es tal que $P_a = \{1, 3\}$ $P_c = \{0\}$ $P_s = \{2\}$

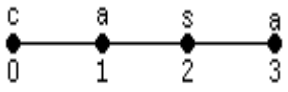


Figura 2.1: Estructura para ω

Definición 2.1.5. Para un modelo $(\mathcal{M}_\omega, V)$ y una sentencia φ diremos que $\omega \in \mathcal{L}(\varphi)$ si y sólo si $\mathcal{M}_\omega, V \models \varphi$.

Definimos el lenguaje de una sentencia φ de $LMSO(\Sigma)$ como $\mathcal{L}(\varphi) = \{\omega \in \Sigma^* | \mathcal{M}_\omega \models \varphi\}$.

A continuación, demostraremos que un lenguaje $\mathcal{L}$ es definible en LMSO si y sólo si $\mathcal{L}$ es regular. Para esto introduciremos el lenguaje restringido $LMSO^-(\Sigma)$ dado por:

$$\varphi := X \subseteq Y \mid X \subseteq P_a \mid Suc(X, Y) \mid \varphi_1 \vee \varphi_2 \mid \neg\varphi \mid \exists X \varphi$$

Para desarrollar más facilmente la teoría se utilizará una traducción de fórmulas de la $LMSO(\Sigma)$ en fórmulas de $LMSO^-(\Sigma)$. La traducción se basará en la existencia de la siguiente fórmula, que expresa la propiedad de ser síngleton.

$$Sing(X) = \exists Y \exists Z \exists W \left[\neg(Y = Z) \wedge \neg(Y = W) \wedge \neg(Z = W) \wedge \forall U \left(U \subseteq X \Rightarrow (U = Y \vee U = Z \vee U = W)\right)\right]$$

donde la fórmula $X = Y$ abrevia: $X \subseteq Y \wedge Y \subseteq X$

Lema 2.1.6. *Toda sentencia de LMSO(Σ) es lógicamente equivalente a una sentencia de LMSO-(Σ) por medio de la traducción* $(\cdot)^t : LMSO(\Sigma) \longrightarrow LMSO^-(\Sigma)$ *dada por:*

$$(x_i = x_j)^t \equiv Sing(X_i) \wedge Sing(X_j) \wedge X_i \subseteq X_j \wedge X_j \subseteq X_i$$

$$(P_a(x_i))^t \equiv Sing(X_i) \wedge X_i \subseteq P_a$$

$$(X_i(x_j))^t \equiv Sing(X_j) \wedge X_i \subseteq X_j$$

$$(X_i \subseteq X_j)^t \equiv X_i \subseteq X_j$$

$$(X_i \subseteq P_a)^t \equiv X_i \subseteq P_a$$

$$(Suc(X_i, X_j))^t \equiv Suc(X_i, X_j)$$

$$(\varphi \vee \psi)^t \equiv (\varphi)^t \vee (\psi)^t$$

$$(\neg\varphi)^t \equiv \neg(\varphi)^t$$

$$(\exists x_i \varphi)^t \equiv \exists X_i (Sing(X_i) \wedge (\varphi)^t)$$

$$(\exists X_i \varphi)^t \equiv \exists X_i (\varphi)^t$$

La demostración de este lema es una inducción rutinaria en la complejidad de fórmulas de $LMSO(\Sigma)$, usando simplemente la Definición 2.1.3. Vale anotar que si queremos ser totalmente cuidadosos, el enunciado del lema deberia pedir renombramiento de variables para evitar conflictos. La idea en la traducción anterior es el reemplazo de x_i por $Sing(X_i)$. Pero si X_i ya aparecía en la fórmula original, seria necesario reemplazar X_i por X_k, donde X_k sea una nueva variable que no aparezca en la fórmula ω.
Ahora si podemos pasar a nuestro resultado.

Teorema 2.1.7. *Un lenguaje de palabras finitas es definible por una sentencia en LMSO$^-$(Σ) si y sólo si es el lenguaje reconocido por algún autómata finito.*

Demostración.
(⇐)Sea $\mathcal{A} = \langle Q, \Sigma, q^0, F, \delta\rangle$ autómata determinístico que reconoce el lenguaje $\mathcal{L}$ con $|Q| = n$. Construiremos una sentencia $\varphi_{\mathcal{A}}$ tal que para toda palabra $\omega \in \Sigma^*$, $\mathcal{M}_\omega \models \varphi_{\mathcal{A}}$ si y solo si $\omega \in \mathcal{L}$. Para esto, es necesario que la fórmula $\varphi_{\mathcal{A}}$ cumpla tres condiciones que describen exactamente el comportamiento del autómata al correr una palabra que es aceptada por él.

Intuitivamente identificaremos cada estado q_i del autómata con la variable X_i. Queremos que $\varphi_{\mathcal{A}}$ sea tal que:

(i) Existen conjuntos $X_1, \ldots, X_n$ que constituyen una partición de $dom(\omega)$. Es decir en cada paso de la lectura de la palabra, el autómata se encuentra en un único estado.

(ii) El primer elemento de $dom(\omega)$ está en X_1 y el último elemento se encuentra en algún X_k tal que $q_k \in F$.

(iii) Para cada elemento en $dom(\omega)$, distinto del último, si el elemento pertenece a X_i y P_a, y su sucesor pertenece a X_j, entonces se cumple $(q_i, a, q_j) \in \delta$.

Definimos entonces

$$\begin{aligned}\varphi_{\mathcal{A}} := \quad & \exists X_1 \ldots \exists X_n \Big[\forall X \Big(Sing(X) \Rightarrow \Big(\textstyle\bigvee_{i=1}^n X \subseteq X_i \wedge \bigwedge_{i\neq j}(X \subseteq X_i \Rightarrow \neg(X \subseteq X_j))\Big)\Big) \\ & \wedge \forall X \left((Sing(X) \wedge \neg\exists Y(Suc(Y, X))) \Rightarrow X \subseteq X_1\right) \\ & \wedge \forall X \Big((Sing(X) \wedge \neg\exists Y(Suc(X, Y))) \Rightarrow \textstyle\bigvee_{q_i \in F} X \subset X_i\Big) \\ & \wedge \forall X \Big(\textstyle\bigvee_{(q_i,a,q_j)\in\delta} \exists Y(Suc(X, Y) \wedge X \subset X_i \wedge X \subseteq P_a) \Rightarrow Y \subseteq X_j\Big)\Big]\end{aligned}$$

(⇒) Sea $\mathcal{L}$ un lenguaje definido por una sentencia φ en LMSO(Σ). Se realizará la construcción de un autómata $\mathcal{A}_\varphi$ que reconozca $\mathcal{L}$.
Probaremos por inducción en fórmulas que para cada fórmula bien formada φ de LMSO$^-$(Σ), con variables libres $X_1, \ldots, X_n$, existe un aútomata $\mathcal{A}_\varphi$ con alfabeto $\Sigma_\varphi = \Sigma \times \mathcal{P}(\{X_1, \ldots, X_n\})$ tal que para toda $\omega \in \Sigma^*$ y valuación $V : Var_2 \longrightarrow \mathcal{P}(\{0, \ldots, |\omega|\})$,

$$\mathcal{M}_\omega, V \models \varphi \text{ si y sólo si } V(\omega) \in \mathcal{A}_\varphi$$

donde $V(\omega)$ es la palabra que se obtiene a partir de ω al reemplazar el caracter ω_k por (ω_k, S) donde $S = \{X_i | k \in V(X_i)\}$.
Para realizar la inducción fijaremos el conjunto de variables de segundo orden $\{X_1, X_2, \ldots, X_n\}$ utilizado por la fórmula φ. Dado este conjunto se puede definir un modelo $(\mathcal{M}, V)$ donde la fórmula es válida. El modelo representa una palabra sobre el alfabeto $\Sigma' = \Sigma \times \mathcal{P}(\{1, \ldots, n\})$. Es decir cada $m \in \mathcal{M}$ se decora con un par (a, S) donde $a \in \Sigma$ y $S = \{i \in \{1, \ldots, n\} | m \in V(X_i)\}$.
Caso Base:

$\varphi := X_i \subseteq X_j$: φ es válida en $(\mathcal{M}, V)$ si $V(X_i) \subseteq V(X_j)$ es decir queremos que el autómata acepte la palabra $\omega = (a_0, S_0) \ldots (a_m, S_m)$ si y solo si cada vez que aparezca X_i en algún S_k entonces X_j aparece en S_k, de lo contrario la palabra no será aceptada. Definimos

$\mathcal{A}_\varphi = \langle \{0,1\}, \Sigma_\varphi, 0, \{0\}, \delta \rangle$ así:

$$\begin{aligned} \delta(0,(a,S')) &= 1 \quad \text{si } X_i \in S' \text{ y } X_j \notin S' \\ \delta(0,(a,S)) &= 0 \quad \text{si } X_i \notin S \text{ ó } X_j \in S \end{aligned}$$

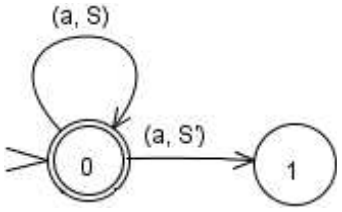


Figura 2.2: Autómata para $X_i \subseteq X_j$

Nota 2.1.8. La transición marcada por (a, S) representa las transiciones en las que $i \in S \wedge j \in S,\ i \notin S \wedge j \in S,\ i \notin S \wedge j \notin S$. Continuaremos utilizando esta notación de múltiples transiciones para poder ofrecer una representación gráfica legible.

$\underline{\varphi := X_i \subseteq P_a}$: φ es válida en $(\mathcal{M}_\omega, V)$ si para cada $X_i \in S$ se tiene que $V(X_i) \subseteq P_a$. De forma similar al caso anterior requerimos que el autómata acepte las palabras $\omega = (s_0, S_0) \dots (s_m, S_m)$ para las cuales si X_i está en algún S_k entonces $s_k = a$.
Definimos $\mathcal{A}_\varphi = \langle \{0,1\}, \Sigma_\varphi, 0, \{0\}, \delta \rangle$ por:

$$\begin{aligned} \delta(0,(b,S) &= 1 \quad \text{si } X_i \in S \text{ y } b \neq a \\ \delta(0,(a,S')) &= 0 \quad \text{de lo contrario} \end{aligned}$$

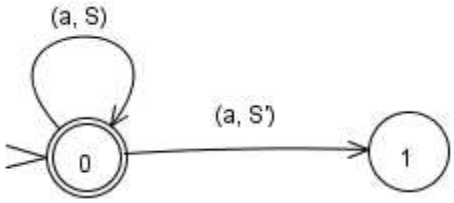


Figura 2.3: Autómata para $P_a \subseteq X_j$

$\underline{\varphi := Suc(X_i, X_j)}$: Definimos un autómata que acepte las palabras para las cuales se lee exactamente una vez un caracter (a, S) con $X_i \in S$ y exactamente una vez un caracter (b, S') con $X_j \in S'$ y además (a, S), (b, S') aparecen consecutivamente. El autómata está representado en la Figura 2.4.

$$\begin{aligned} \delta(0,(a,S)) &= 0 \quad \text{si } X_i \notin S \\ \delta(0,(a,S)) &= 1 \quad \text{si } X_i \in S \\ \delta(1,(b,S')) &= 2 \quad \text{si } X_j \in S' \\ \delta(2,(a,S')) &= 2 \quad \text{si } X_i \notin S',\ X_j \notin S' \\ \delta(2,(a,S')) &= 3 \quad \text{si } X_i \in S' \text{ ó } X_j \in S' \end{aligned}$$

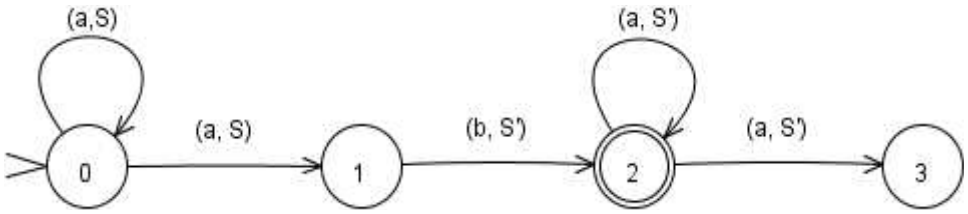


Figura 2.4: Autómata para $Suc(X_i, X_j)$

Hipótesis de Inducción: Suponga que se tiene el resultado para las fórmulas φ_1 y φ_2.

$\varphi := \neg\varphi_1$: Sea $\mathcal{A}_{\varphi_1}$ autómata que acepta φ_1 y $\mathcal{L}_{\varphi_1}$ su lenguaje asociado. Del Lema 1.1.10 sabemos que existe un autómata $\mathcal{A}^c_{\varphi_1}$ que reconoce el lenguaje $\mathcal{L}^c_{\varphi_1}$, precisamente el lenguaje definido por $\neg\varphi_1$.

$\varphi := \varphi_1 \vee \varphi_2$: Sean $\mathcal{A}_{\varphi_1}$ y $\mathcal{A}_{\varphi_2}$ autómatas que reconocen los lenguajes definidos por φ_1 y φ_2 respectivamente. Utilizando el lema 1.1.8 encontramos el autómata $\mathcal{A}_{\varphi_1 \vee \varphi_2}$

$\varphi := \exists X_i.\varphi_1$: Sea $\mathcal{A}_{\varphi_1} = \langle Q, \Sigma \times \mathcal{P}(I), q^0, F, \delta\rangle$ autómata reconociendo el lenguaje $\mathcal{L}_{\varphi_1}$, donde $I = \{X_1, \ldots, X_n\} \cup \{X_i\}$ como en el Lema 1.1.13. Construimos el autómata $\mathcal{A}_{\pi_{I'}}$ para el conjunto de indices $I' = I \setminus \{X_i\}$. De esta forma el lenguaje $\mathcal{L}(\mathcal{A}_\varphi) = \pi_{I'}(\mathcal{L}(\mathcal{A}_{\varphi_1}))$

□

Capítulo 3

μ-Cálculo y Autómatas Alternantes

El μ-cálculo se puede ver como un fragmento de la lógica de segundo orden cuya sintaxis se presenta de forma similar a la de un lenguaje proposicional, pero agregando dos nuevos operadores $\langle - \rangle$ y μ; así mismo podemos ver el μ-cálculo como una extensión de la lógica modal, en particular la lógica temporal lineal (LTL).
Comenzaremos introduciendo la sintaxis del lenguaje del μ-cálculo. El conjunto de fórmulas bien formadas está dado por:

$$\varphi := X_i \mid P_a \mid \neg\varphi \mid \varphi \vee \psi \mid \langle 0 \rangle \varphi \mid \langle 1 \rangle \varphi \mid \mu X.\varphi(X)$$

donde: las X_i son variables de segundo orden.
P_a es un símbolo de predicado para cada $a \in \Sigma$

En cuanto a las fórmulas de la forma $\mu X.\alpha(X)$, se considerarán bien formadas únicamente aquellas positivas en X, es decir, aquellas donde todas las ocurrencias de la variable X están precedidas por una cantidad par de negaciones en el árbol sintáctico de α; de lo contrario, como veremos después, nuestra definición de validez no sería correcta.
Adicionalmente al lenguaje presentado, introducimos también algunas abreviaciones:

$$\nu X.\alpha(X) = \neg\mu X.\neg\alpha(\neg X) \qquad [-]\alpha = \neg\langle - \rangle\neg\alpha \qquad \alpha \wedge \beta = \neg(\neg\alpha \vee \neg\beta)$$

A lo largo del documento se presentarán fórmulas de la forma $\sigma X.\alpha(X)$ donde σ puede representar indistintamenre tanto μ como ν.

Como sucedía con los autómatas alternantes la fórmula φ habla de árboles binarios decorados con caracteres de Σ y su interpretación semántica será un subconjunto de nodos. Se interpretará la sentencia $\mu X.\varphi(X)$ como el menor punto fijo de un operador monótono en $\mathcal{P}(\{0,1\}^*)$ descrito por $\varphi(X)$.

Como se dijo anteriormente, se trabajará con estructuras de árboles binarios t sobre un alfabeto Σ, representadas por $\mathcal{M}_t = \langle \{0,1\}, P_a \rangle$ para $a \in \Sigma$. En lo que resta de este docuemnto usaremos S para denotar el conjunto $\{0,1\}^*$

Definición 3.0.9. Una valuación para el lenguaje del μ-cálculo sobre un modelo $\mathcal{M}_t$ es una función $V : \{X_1, X_2, \ldots\} \longrightarrow \mathcal{P}(S)$, intuitivamente X vale en S si $s \in V(X)$.

Para los modelos $\mathcal{M}_t$ basados en árboles t, la interpretación del cálculo μ está dada por la siguiente extensión de cada valuación V a una valuación $\bar{V} : FBF_\mu \longrightarrow \mathcal{P}(S)$ así:

$$\begin{aligned}
&\bar{V}(X) = V(X) \text{ es el conjunto de nodos del árbol donde vale } X\\
&\bar{V}(P_a) = P_a^{\mathcal{M}_t} \text{ es el conjunto de nodos decorados con } a\\
&\bar{V}(\neg\alpha) = \{0,1\}^* \setminus \bar{V}(\alpha)\\
&\bar{V}(\alpha \vee \beta) = \bar{V}(\alpha) \cup \bar{V}(\beta)\\
&\bar{V}(\langle 0\rangle\alpha) = \{r \in t | r0 \in \bar{V}(\alpha)\}\\
&\bar{V}(\langle 1\rangle\alpha) = \{r \in t | r1 \in \bar{V}(\alpha)\}\\
&\bar{V}(\mu X.\alpha(X)) = \cap\{A \subseteq \{0,1\}^* | \bar{V}_{[\frac{A}{X}]}(\alpha) \subseteq A\}
\end{aligned}$$

donde $\bar{V}_{[\frac{A}{X}]}$ es igual a $\bar{V}$, excepto que $\bar{V}(X) = A$.
Utilizando la definición anterior y la abreviaciones de ν, $\wedge$ y $[0]$ la semántica de estos operadores es la siguiente:

$$\begin{aligned}
\bar{V}(\varphi \wedge \psi) &= \bar{V}(\varphi) \cap \bar{V}(\psi)\\
\bar{V}([0]\varphi) &= \bar{V}(\neg\langle 0\rangle\neg\varphi)\\
&= \{0,1\}^* \setminus \bar{V}(\langle 0\rangle\neg\varphi)\\
&= \{r \in t | r0 \notin \bar{V}(\neg\varphi)\}\\
&= \{r \in t | r0 \in \bar{V}(\varphi)\} = \bar{V}(\langle 0\rangle\varphi)\\
\bar{V}(\nu X.\alpha(X)) &= \bar{V}(\neg\mu X.\neg\alpha(\neg X))\\
&= \{0,1\}^* \setminus \bar{V}(\mu X.\neg\alpha(\neg X))\\
&= \{0,1\}^* \setminus \bigcap\{A \subseteq \{0,1\}^* | \bar{V}_{[\frac{A}{X}]}(\neg\alpha(\neg X)) \subseteq A\}\\
&= \{0,1\}^* \setminus \bigcap\{A \subseteq \{0,1\}^* | \bar{V}_{[\frac{A^c}{X}]}(\neg\alpha(X)) \subseteq A\}\\
&= \{0,1\}^* \setminus \bigcap\{A \subseteq \{0,1\}^* | \bar{V}_{[\frac{A^c}{X}]}(\alpha(X))^c \subseteq A\}\\
&= \{0,1\}^* \setminus \bigcap\{A \subseteq \{0,1\}^* | \bar{V}_{[\frac{A^c}{X}]}(\alpha(X)) \supseteq A^c\}\\
&= \bigcup\{A^c \subseteq \{0,1\}^* | \bar{V}_{[\frac{A^c}{X}]}(\alpha(X)) \supseteq A^c\}\\
&= \bigcup\{A \subseteq \{0,1\}^* | \bar{V}_{[\frac{A}{X}]}(\alpha(X)) \supseteq A\}
\end{aligned}$$

Dada esta semántica, veremos ahora porqué los operadores μ y ν se llaman operadores de punto fijo y porqué hemos pedido que las fórmulas $\alpha(X)$ en fórmulas $\mu X.\alpha(X)$(y por lo tanto en $\nu X.\alpha(X)$) sean positivas para X.

Definición 3.0.10. Para una fórmula $\varphi(X)$ del μ-cálculo que es positiva para X, y para $\langle \mathcal{M}_t, V\rangle$, definimos el operador $F_{\varphi,V} : \mathcal{P}(S) \longrightarrow \mathcal{P}(S)$ por: $F_{\varphi,V}(A) = \bar{V}_{[\frac{A}{X}]}(\varphi(X))$.

Lema 3.0.11. *El operador $F_{\varphi,V}$ de la definición anterior es monótono creciente, es decir, si $A \subseteq B$ entonces $F(A) \subseteq F(B)$.*

Demostración. Procedemos por inducción en la complejidad de $\alpha(X)$. En realidad, se debe probar por inducción de una fórmula $\varphi(X, Y_1, \ldots, Y_k)$ que

(i) Si $\varphi(X, Y_1, \ldots, Y_k)$ es positiva para X, entonces $F_{\varphi,V}$ es monótona creciente

(ii) Si $\varphi(X, Y_1, \ldots, Y_k)$ no es positiva para X, entonces $F_{\varphi,V}$ es monótona decreciente

La inducción no es completamente rutinaria, pero no es dificil; los casos de $\neg$ y μ requieren algún cuidado, no los presentaremos aquí. □

Ahora veremos un caso especial del teorema de Knaster - Tarski

Teorema 3.0.12. ***(Knaster-Tarski)***
Considere el retículo completo $\langle \mathcal{P}(S), \subseteq \rangle$ y sea $f : \mathcal{P}(S) \longrightarrow \mathcal{P}(S)$ una función monótona creciente. Entonces f tiene puntos fijos mínimo y máximo dados por $\bigcup\{A \subseteq S | A \subseteq f(A)\}$ y $\bigcap\{A \subseteq S | A \subseteq f(A)\}$ respectivamente.

Al revisar la definición de $\bar{V}(\mu X.\varphi(X))$, el Lema 3.0.11 y el Teorema de Knaster - Tarski, claramente obtenemos que $\bar{V}(\mu X.\varphi(X))$ es el menor punto fijo del operador $F_{\varphi,V}$. Similarmente, $\bar{V}(\nu X.\varphi(X))$ es el mayor punto fijo de ese operador.

Ahora bien, considere la fórmula $\varphi := \mu X.(P_a \vee \langle 0\rangle X \vee \langle 1\rangle X)$. Una pregunta natural para esta fórmula es ¿Para qué tipo de árboles $\mathcal{M}_t$ se tiene que su raíz ϵ satisface φ, es decir $\bar{V}(\epsilon) \in \bar{V}(\varphi)$ para cualquier valuación V? Para responder ésta pregunta necesitamos que

$$\epsilon \in \bigcap\{A \subseteq S | F_\varphi(A) \subseteq A\}$$

Esta condición es dificil de entender y su cálculo, la intersección para todos los subconjuntos de S, es computacionalmente inviable para árboles infinitos. Sin embargo, es un hecho bien conocido acerca de puntos fijos que si $f : \mathcal{P} \longrightarrow \mathcal{P}$ es monótona creciente, entonces el menor punto fijo se puede obtener como la unión $\bigcup_{\tau \in Ord} f^\tau(\phi)$, donde

- $f^0(\phi) = \phi$
- $f^{\tau+1}(\phi) = f(f^\tau(\phi))$
- $f^\tau(\phi) = \bigcup_{\tau < \tau'} f^\tau(\phi)$ si τ' es ordinal límite[1]

Similarmente el mayor punto fijo se obtiene como $\bigcap_{\tau \in Ord} f^\tau(S)$ comenzando por $f^0(S) = S$.

Ejemplo 3.0.13. Considere la fórmula $\varphi = \mu X.P_a \vee \langle 1\rangle X \vee \langle 0\rangle X$. Calculamos el punto fijo de φ fijando el modelo $\mathcal{M}_t\langle S, P_a^{\mathcal{M}}\rangle$ y la valuación $V : Var_2 \longrightarrow \mathcal{P}(S)$. Calculamos las primeras aproximaciones de $F^\tau_{\varphi,V}(\phi)$

$$\begin{aligned}
F^0_\varphi(\phi) &= \phi \\
F^1_\varphi(\phi) &= P_a \vee \langle 1\rangle\phi \vee \langle 0\rangle\phi \\
&= P_a \\
F^2_\varphi(\phi) &= P_a \vee \langle 1\rangle P_a \vee \langle 0\rangle P_a \\
&= \{v \in t | v \in P_a \vee v1 \in P_a \vee v0 \in P_a\} = B \\
F^3_\varphi(\phi) &= P_a \vee \langle 1\rangle B \vee \langle 0\rangle B \\
&= \{v \in t | v \in P_a \vee v11 \in P_a \vee v00 \in P_a \vee v10 \in P_a \vee v01 \in P_a\} = B_2 \\
F^4_\varphi(\phi) &= \bar{V}_{[\frac{B_2}{X}]} = P_a \cup \{s \in S | s0 \in B\} \cup \{s \in S | s1 \in B_2\} \\
&= P_a \cup \{s \in S | s0 \in P_a \text{ ó } s00 \in P_a \text{ ó } s01 \in P_a\} \cup \{s \in S | s1 \in P_a \text{ ó } s10 \in P_a \text{ ó } s11 \in P_a\} \\
\vdots &= \vdots \\
F^\tau_\varphi(\phi) &= \bigcup_{\tau' < \tau} F^{\tau'}_\varphi(A) \text{ con } \tau > |S|^+
\end{aligned}$$

[1] Para este caso basta con tomar la unión hasta el ordinal mayor al cardinal de S.

En general, podemos probar por inducción en $n > 0$ que la raíz ϵ del árbol t está en $F^n_{\varphi,V}(\phi)$ si y sólo si algún nodo a distancia a lo sumo $n-1$ de la raíz está marcado con el carácter a.

Hemos probado que $\mathcal{L}(\varphi) = \{t | t \text{ árbol y } \mathcal{M}_t \models \varphi\}$ es precisamente $\mathcal{L}(\mathcal{A})$ para el autómata $\mathcal{A}$ del ejemplo 1.2.10

El lector puede comprobar usando las aproximaciones $F^\tau(S)$, que con φ la misma fórmula de antes, $\psi := \nu Y.(\varphi \wedge \langle 0\rangle Y \wedge \langle 1\rangle Y)$ vale en los árboles tales que todo subárbol tiene al menos una a.

Para finalizar este capítulo centraremos nuestra atención para responder la pregunta ¿Siempre es posible, dada una sentencia φ del μ-cálculo, encontrar un autómata alternante $\mathcal{A}$ tal que $\mathcal{L}(\mathcal{A}) = \mathcal{L}(\varphi)$ y viceversa?

Antes de proceder, tenemos la siguiente observación.
Observación 1: Si tomamos los conectivos $\wedge$, ν como primitivos dentro de la sintaxis del cálculo μ, toda fórmula bien formada φ es lógicamente equivalente a una fórmula φ' donde los símbolos de $\neg$ anteceden únicamente a la fórmulas P_a o a las variables X. Si φ es sentencia, entonces las negaciones anteceden sólo a los P_a.

Lema 3.0.14. *Si φ es fbf entonces φ y $\neg\varphi$ son lógicamente equivalentes a algunas fórmulas φ_1 y φ_2 donde las negaciones ocurren únicamente frente a los P_a o a las X_i. Además, si φ es positiva para X libre en φ, entonces φ_1 es positiva para X. Si φ no es positiva para X, entonces φ_1 tampoco lo es, mientras que φ_2 es positiva para X.*

Demostración. La prueba se sigue por inducción en la complejidad de la fórmula φ.
Caso base:

- $\underline{P_a}$: Claramente, $P_a \equiv P_a$ y $\neg P_a \equiv \neg P_a$.
- $\underline{X}$: Similar al caso anterior, $X \equiv X$ y $\neg X \equiv \neg X$.

Hipótesis de inducción: Suponga que $\varphi \equiv \varphi_1$, $\neg\varphi \equiv \varphi_2$, $\psi \equiv \psi_1$ y $\neg\psi \equiv \psi_2$, donde φ_1, φ_2, ψ_1 y ψ_2 son tales que las negaciones aparecen únicamente frente a los P_a o a las X_i.

- $\underline{\varphi \wedge \psi}$: Por hipótesis de inducción para esta fórmula tenemos $\varphi \wedge \psi \equiv \varphi_1 \wedge \psi_1$, así mismo $\neg(\varphi \wedge \psi) \equiv \varphi_2 \vee \psi_2$.
- $\underline{\varphi \vee \psi}$: por hipótesis de inducción para esta fórmula tenemos $\varphi \vee \psi \equiv \varphi_1 \vee \psi_1$, así mismo $\neg(\varphi \vee \psi) \equiv \varphi_2 \wedge \psi_2$
- $\underline{\neg\varphi}$: Por hipótesis de inducción $\neg\varphi \equiv \varphi_2$ y $\neg(\neg\varphi) \equiv \varphi_1$
- $\underline{\mu X.\varphi(X)}$: Por hipótesis de inducción $\mu X.\varphi(X) \equiv \mu X.\varphi_1(X)$. Para la fórmula negada,
  $$\begin{aligned} \neg\mu X.\varphi(X) &\equiv \neg\neg\nu X.\neg\varphi(\neg X) \\ &\equiv \nu X.\neg\varphi(\neg X) \quad \equiv \nu X\varphi_2(\neg X) \end{aligned}$$
  La última equivalencia es correcta debido a que por hipótesis de inducción , si X aparece libre en $\varphi(X)$, entonces como φ debe ser positiva para X entonces $\varphi_2(X)$ es negativa para X, luego $\varphi_2(\neg X)$ es positiva para X.

□

De acuerdo con el Lema 3.0.14, de ahora en adelante tomaremos ν, $\wedge$ como primitivas y usaremos fórmulas con negaciones sólo al frente de P_a's y X_i's.

3.1. μ-Cálculo y Autómatas Alternantes

En esta sección se estudiará la relación existente entre el μ-cálculo y los autómatas alternantes. Comenzaremos por observar cómo se realiza la transición de lenguajes definidos por fórmulas del μ-cálculo a los autómatas alternantes y luego realizaremos la traducción en la otra dirección.

Comenzaremos mostrando cómo se construye un autómata alternante a partir de una fórmula dada φ. Definimos el conjunto $cl(\varphi) = \{\psi | \psi$ es subfórmula de $\varphi\}$ como el conjunto de todas las subfórmulas de φ.

Construimos el autómata $\mathcal{A}_\varphi = \langle Q, \Sigma, q^0, \delta, Acc\rangle$ como sigue:

- $Q = cl(\varphi) \cup \{\bot, \top\}$. Usaremos $\top$ como un estado a partir del cual se acepta todo y $\bot$ como un estado a partir del cual no se acepta nada.
- q^0 es el estado φ. Además la función δ está dada por:

$$
\begin{array}{rcl}
\delta(\top, a) = & \{(\top, \lambda)\} & \\
\delta(\bot, a) = & \phi & \\
\delta(P_a, a) = & \{(\top, \lambda)\} & \\
\delta(P_a, b) = & \{(\bot, \lambda)\} & \text{si } a \neq b \\
\delta(\neg P_a, a) = & \{(\bot, \lambda)\} & \\
\delta(\neg P_a, b) = & \{(\top, \lambda)\} & \text{si } a \neq b \\
\delta(X, a) = & \{(\alpha(X), \lambda)\} & \\
\delta(\alpha_1 \vee \alpha_2, a) = & \{(\alpha_1, \lambda), (\alpha_2, \lambda)\} & \\
\delta(\alpha_1 \wedge \alpha_2, a) = & \{(\alpha_1, \lambda), (\alpha_2, \lambda)\} & \\
\delta(\langle 0\rangle\alpha, a) = & \{(\alpha, 0)\} & \\
\delta(\langle 1\rangle\alpha, a) = & \{(\alpha, 1)\} & \\
\delta(\sigma X.\alpha(X), a) = & \{(\alpha(X), \lambda)\} &
\end{array}
$$

Los estados existenciales $Q_\exists$ estarán conformados por las sub-fórmula de la forma $\varphi_1 \vee \varphi_2$ y los estados universales $Q_\forall$ estarán conformados por todas las otras subformulas de $cl(\varphi)$.
Para definir la condición de aceptación Acc se introducirá una nueva noción.

Definición 3.1.1. Definimos el orden de dependencia para las variables ligadas de φ por. $X <_\varphi Y$ si y solo si X ocurre libre en $\sigma Y.\alpha(Y)$.
Llamaremos profundidad de alternación $adepth(X)$ al máximo número de alternaciones de punto fijo (alternaciones entre μ variables y ν variables en las cadenas $X <_\varphi Z_1 <_\varphi \ldots <_\varphi Y$ que se pueden formar con el orden $<_\varphi$.

Sin pérdida de generalidad, asumiremos siempre que la variable X aparece ligada a lo sumo una vez en las fórmulas φ.

Ejemplo 3.1.2. Considere la fórmula $\varphi = \mu X.(\nu Y.[\langle 1\rangle X \wedge \langle 0\rangle Y])$.
En este caso X ocurre libre en la sub-fórmula $\nu Y.\langle 1\rangle X \wedge \langle 0\rangle Y$ por lo tanto $X <_\varphi Y$. La profundidad de alternación de X $adeph(X)$ es 1. Por su parte el $adepth(Y) = 0$ puesto que Y es un elemento maximal en el orden.

Ejemplo 3.1.3. Considere la fórmula $\varphi = \mu X.[(\nu Y.\langle 0\rangle Y) \wedge \langle 1\rangle X]$.
Para φ vemos que $X \not<_\varphi Y$ y $Y \not<_\varphi X$ por lo tanto, $adepth(X)$ y $adepth(Y)$ son 0.

Ejemplo 3.1.4. Considere la fórmula $\varphi = \nu X.(\nu Y.[\langle 1\rangle X \wedge \langle 0\rangle Y])$.
En este caso, nuevamente X ocurre libre en la sub-fórmula $\nu Y.(\langle 1\rangle X \wedge \langle 0\rangle Y)$ por lo tanto $X <_\varphi Y$, La profundidad de alternación de X, $adeph(X)$, es 0. Puesto que la variable Y está cuantificada por μ nuevamente $adepth(Y) = 0$.

Ahora sí, la condición de Mostowski para el autómata $\mathcal{A}_\varphi$ está dada por:

$$\Omega(\alpha) = \begin{cases} 0 & \text{si } \alpha = \top \\ 2(max_d - adepth(X)) & \text{si } \alpha \text{ es } \nu \text{ variable} \\ 2(max_d - adepth(X)) + 1 & \text{si } \alpha \text{ es } \mu \text{ variable} \\ M & \text{de lo contrario} \end{cases}$$

donde max_d es la máxima profundidad de alternación en $cl(\varphi)$ y M es mayor que el valor máximo de Ω para las variables.

La idea aquí es que toda rama infinita $(q_0, t_0)(q_1, t_1)\ldots$ de una ejecución sobre el árbol $\mathcal{T}$ debe ser tal que, o una de las variables o $\top$ se repite infinitas veces. Cada pareja (q, t) significa que se debe verificar si $t \in \bar{V}(q)$. En teoría de computación μ se asocia con la noción de iteración finita, luego las repeticiones infinitas de variables μ son consideradas “malas".

Introduciremos una serie de definiciones necesarias para la demostración del resultado principal de esta sección.

Nota 3.1.5. Para los modelos del μ-cálculo $\langle \mathcal{M}_t, V\rangle$ definimos la τ-aproximación para la fórmula $\mu X.\alpha(X)$ como $\mu^\tau X.\alpha(X)$, así.

- $\left[\mu^0 X.\alpha(X)\right]_V = \phi$
- $\left[\mu^{\tau+1} X.\alpha(X)\right]_V = [\alpha(X)]_{V\left[\frac{\mu^\tau X.\alpha(X)}{X}\right]}$
- $[\mu^\tau X.\alpha(X)]_V = \bigcup_{\tau<\tau'} [\mu^\tau X.\alpha(X)]_V$ si τ' es ordinal límite

Note que lo único que hicimos aquí fue cambiar la notación para el mismo concepto que se definió en la sección anterior antes del Ejemplo 3.0.13. Esto lo hacemos para adoptar la notación de [1], que es más cómoda para lo que sigue.

Definición 3.1.6. Para una fórmula φ, sea $X_1 <_\varphi X_2 <_\varphi \ldots <_\varphi X_k$ compatible con el orden dado por φ, y sea $\vec{\tau} = (\tau_1, \ldots, \tau_k)$ tupla de ordinales. Para una fórmula $\gamma \in cl(\varphi)$ definimos la sustitución sequencial de aproximaciones así:

$$\langle\gamma\rangle^{\mu}_{\vec{\tau}} = \gamma\left[\frac{\sigma_k X_k.\alpha(X_k)}{X_k}\right]\ldots\left[\frac{\sigma_1 X_1.\alpha(X_1)}{X_1}\right] \text{ donde } \sigma_k = \begin{cases} \nu & \text{si } X_k \text{ es } \nu \text{ variable} \\ \mu^{\tau_k} & \text{si } X_k \text{ es } \mu \text{ variable} \end{cases}$$

$$\langle\gamma\rangle^{\nu}_{\vec{\tau}} = \gamma\left[\frac{\sigma_k X_k.\alpha(X_k)}{X_k}\right]\ldots\left[\frac{\sigma_1 X_1.\alpha(X_1)}{X_1}\right] \text{ donde } \sigma_k = \begin{cases} \nu^{\tau_k} & \text{si } X_k \text{ es } \nu \text{ variable} \\ \mu & \text{si } X_k \text{ es } \mu \text{ variable} \end{cases}$$

Definición 3.1.7. Llamaremos μ-signatura $Sig(\gamma, v)$ en el vertice v para $\gamma \in cl(\varphi)$ al menor $\vec{\tau}$ en el orden lexicográfico de las tuplas de ordinales tal que $v \in \left[\langle\gamma\rangle^{\mu}_{\vec{\tau}}\right]_V$, si $v \in [\gamma]_V$.
De la misma forma, definimos la ν-signatura $Sig^{\nu}(\gamma, v)$ al menor $\vec{\tau}$ en el orden lexicográfico de las tuplas de ordinales tal que $v \notin \left[\langle\gamma\rangle^{\mu}_{\vec{\tau}}\right]_V$, si $v \notin [\gamma]_V$.

Lema 3.1.8. ***(Decrecimiento de la signatura)***
Suponga fijos un árbol $\mathcal{M}_t$, una valuación V y un vértice v. En cada uno de los siguientes casos, la igualdad vale si v satisface la fórmula φ de la parte izquierda $Sig(\varphi, -)$ de la ecuación.

1. $Sig(\top, v) = 0$
2. $Sig(P_a, v) = 1$
3. $Sig(\neg P_a, v) = 1$
4. $Sig(\alpha_1 \wedge \alpha_2, v) = \text{máx}\{Sig(\alpha_1, v), Sig(\alpha_2, v)\}$
5. $Sig(\alpha_1 \vee \alpha_2, v) = Sig(\alpha_1, v)$ *ó* $Sig(\alpha_2, v)$
6. $Sig(\langle 0\rangle\alpha, v) = Sig(\alpha, v0)$
7. $Sig(\langle 1\rangle\alpha, v) = Sig(\alpha, v1)$
8. $Sig(\nu X.\alpha(Y), v) = Sig(\alpha(Y), v)$
9. $Sig(\mu X_i.\alpha(X_i), v) = Sig(\alpha(X_i), v)$ *en las primeras* $i-1$ *posiciones de* $\vec{\tau}$
10. *Si* Y *es* $\nu -$ *variable* $Sig(Y, v) = Sig(\alpha(Y), v)$
11. *Si* X_i *es* μ*-variable* $Sig(X_i, v) > Sig(\alpha(X_i), v)$. *Además, la diferencia se encuentra en la posición* i *de la tupla* $\vec{\tau}$.

Para la ν-signatura se tiene un resultado análogo intercambiando el papel de ν y μ y el papel de $\wedge$ y $\vee$ en las ecuaciones.

Demostración. Los puntos (1)-(7) del lema son claros de la definición de signatura, mostraremos el item (11). Considere la fórmula $\varphi(X_1, \ldots, X_i, \ldots, X_n)$ con variables libres $X_1, \ldots, X_n$ y suponga que la variable X_i es $\mu-$variable cuantificada por la fórmula $\mu X_i.\alpha(X_i)$. La sustitución secuencial de X_i $\langle X_i\rangle^{\mu}_{\vec{\tau}} = X_i\left[\frac{\mu^{\tau_i} X_i.\alpha(X_i)}{X_i}\right] = \mu^{\tau_i} X_i.\alpha(X_i)$, por la definición de $Sig(X, v)$, $\vec{\tau}$ es el mínimo ordinal tal que $v \in [\mu^{\tau_i} X_i.\alpha(X_i)]_V$ si $v \in V(X_i)$.
Por otra parte $Sig(\alpha(X_i), v)$ es el mínimo $\vec{\tau}'$ tal que $v \in \left[\alpha(X_i)\left[\frac{\mu^{\tau_i'} X_i.\alpha(X_i)}{X_i}\right]\right]_V$ si

$v \in \left[\alpha(\mu^{\tau_i'} X_i.\alpha(X_i))\right]_V$. Por la definición 3.1.5 esto es exactamente que $v \in \left[\alpha(\mu^{\tau_i'+1} X_i.\alpha(X_i))\right]_V$. Como $\vec{\tau}$ era el mínimo que cumplía la propiedad, entonces $\vec{\tau} > \vec{\tau}'$, con la diferencia exactamente en la posición i. □

Teorema 3.1.9. *Para todo árbol t, $t \in \mathcal{L}(\mathcal{A})_\varphi$ si y sólo si $\mathcal{M}_t \models \varphi$.*

Demostración. Como se mencionó anteriormente, la noción de aceptación para un autómata alternante se puede presentar en términos de estrategias ganadoras en el juego $\mathcal{G}_{\mathcal{A},t}$ (definición 1.2.9) para el jugador $\exists$.
($\Leftarrow$) Debido que los nodos de éste jugador son los representados por las disyunciones $\alpha_1 \vee \alpha_2$, definimos la estrategia ganadora como: si el jugador $\exists$ se encuentra en una posición $(\alpha_1 \vee \alpha_2, v)$, pasa a la posición

$$\begin{cases} (\alpha_1, v) & \text{si } Sig(\alpha_1 \vee \alpha_2, v) = Sig(\alpha_1, v) \\ (\alpha_2, v) & \text{si } Sig(\alpha_1 \vee \alpha_2, v) = Sig(\alpha_2, v) \end{cases}$$

Suponga por contradicción que existe un juego $(q_0, v_0)(q_1, v_1)\ldots$ tal que $\Omega(In(r)) = p$ impar, donde $r = q_0, q_1, q_2, \ldots$
De la definición de Ω tenemos que existe una μ-variable X_l con $l = \frac{p-1}{2}$
Sea m el paso del juego a partir del cual sólo pueden aparecer los estados que aparecen infinitas veces en el juego.
Utilizando el Lema 3.1.8(11), $Sig(\mu X_l.\alpha(X_l), v) = Sig(\alpha(X_l), v)$ en las primeras $l-1$ posiciones de $\vec{\tau}$, después del paso m de la ejecución.
Cada vez que, dentro del autómata, encontremos el estado identificado por X_l, la $l-$ésima posición en $\vec{\tau}$ decrece debido a que la transición nos lleva al estado identificado con $\alpha(X_l)$ y la signatura de este estado es menor que la signatura del estado anterior (definicion 3.1.7). Dado que esta iteración es infinita, la signatura siempre decrecerá.
Esto lleva a una contradicción con la propiedad del buen ordenamiento del orden lexicográfico de $\vec{\tau}$. Por lo tanto la estrategia es ganadora para el jugador $\exists$.
($\Rightarrow$)Para esta dirección mostraremos que si $\mathcal{M}_t, \varepsilon \not\models \varphi$ entonces $\varphi \notin \mathcal{L}(\mathcal{A}_\varphi)$
Sea δ una estrategia para el jugador $\forall$ definida asi: Si el juego se encuentra en una posición $(\alpha_1 \wedge \alpha_2, v)$ entonces $\forall$ pasa a: $\begin{cases} (\alpha_1, v) & \text{si } Sig^\nu(\alpha_1 \wedge \alpha_2, v) = Sig^\nu(\alpha_1, v) \\ (\alpha_2, v) & \text{en otro caso} \end{cases}$
Suponga por contradicción que existe una jugada r tal que $\Omega(In(r)) = p$ con p par . Entonces existe una ν-variable X_l con $l = \frac{p}{2}$, que se repite infinitas veces, o si no $\top$ se repite infinitas veces. Tome m tal que para las jugadas siguientes solo se tienen los estados que aparecen infinitas veces.
Utilizando el Lema 3.1.8 nuevamente, obtenemos que las primeras $l-1$ posiciones de $\vec{\tau}$ se mantienen constantes; sin embargo cuando encontramos el estado X_l la transición dirige al estado $\alpha(X_l)$ donde la ν-signatura decrece. Nuevamente esto contradice la propiedad del buen orden de la clase de las tuplas $\vec{\tau}$, por lo tanto esa es una estrategia ganadora para el jugador $\forall$. □

Ejemplo 3.1.10. Recuerde la fórmula φ presentada en el Ejemplo 3.1.2:

$$\mu X.(\nu Y.(\langle 1\rangle X \wedge \langle 0\rangle Y)).$$

Construiremos el autómata alternante correspondiente, $\mathcal{A}_\varphi$.
$cl(\varphi) = \{\mu X.(\nu Y.\langle 1\rangle X \wedge \langle 0\rangle Y),\ \nu Y.\langle 1\rangle X \wedge \langle 0\rangle Y,\ \langle 1\rangle X \wedge \langle 0\rangle Y,\ \langle 1\rangle X,\ \langle 0\rangle Y,\ X,\ Y,\ \top,\ \bot\}$ Por facilidad identificaremos cada una de estas fórmulas con un número así: $cl(\varphi) = \{0, 1, 2, 3, 4, 5, 6, 7, 8\}$
$Q_\exists = \phi$
$Q_\forall = \{0, 1, 2, 3, 4, 5, 6, 7, 8\}$
Para el cálculo de la función de Mostowski necesitamos
$max_d = \underset{X \in Var_2(\varphi)}{\text{máx}} \{adepht(X)\} = 1$. $M = 2$

$$\begin{array}{lll} \Omega(0) = 1 & \Omega(1) = 1 & \Omega(2) = 1 \\ \Omega(3) = 1 & \Omega(4) = 1 & \Omega(5) = 1 \\ \Omega(6) = 2 & \Omega(7) = 2 & \Omega(8) = 1 \end{array}$$

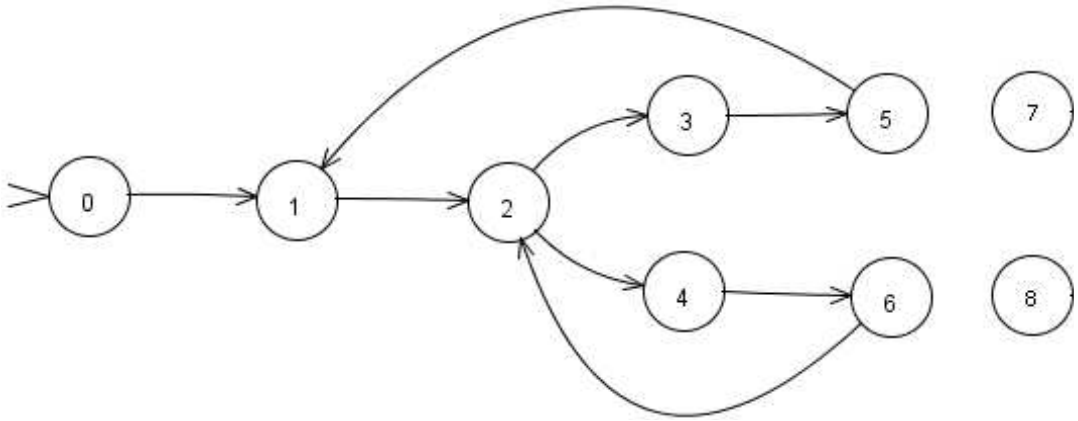


Figura 3.1: Autómata para $\mathcal{A}_\varphi$

Note que en este caso el autómata no acepta ningún árbol puesto que el nodo 2 es un nodo universal. Si $\forall$ toma como estrategia que en el nodo 2 siempre se escoge la transición al nodo 3, se genera una iteración infinita incluyendo el nodo de la μ-variable, luego no existe estrategia ganadora para $\exists$, luego $\mathcal{L}(\mathcal{A}) = \phi$.

Ejemplo 3.1.11. Recuerde que la fórmula $\psi := \nu Y(\phi \wedge \langle 0\rangle Y \wedge \langle 1\rangle Y)$ con
$\varphi := \mu X(P_a \vee \langle 0\rangle X \vee \langle 1\rangle X)$ es tal que $\mathcal{L}(\psi)$ =árboles tales que todo subárbol tiene al menos un nodo marcado con a.
La función de Mostowski esta dada por $\Omega(\top) = 0,\ \Omega(X) = 1,\ \Omega(Y) = 0,$ y $\Omega(q) = 1$ para todos los otros estados q.
Consideramos dos casos de árboles y corridas sobre éstos:

a Considere un árbol que pertenezca a $\mathcal{L}(\psi)$. La estrategia ganadora para $\exists$ en el autómata es, en el estado φ, escoger irse al estado P_a si se está en un nodo marcado con a, irse al estado $\langle 0\rangle X$ si la distancia a un nodo marcado con a decrece, e irse a $\langle 1\rangle X$ en otro caso. Claramente, toda ejecución siguiendo esta estrategia es tal que o ψ es visitado infinitas veces (si $\forall$ nunca escoge irse a φ); la ejecución termina en un loop en $\top$ (si $\forall$ escoge alguna vez φ).

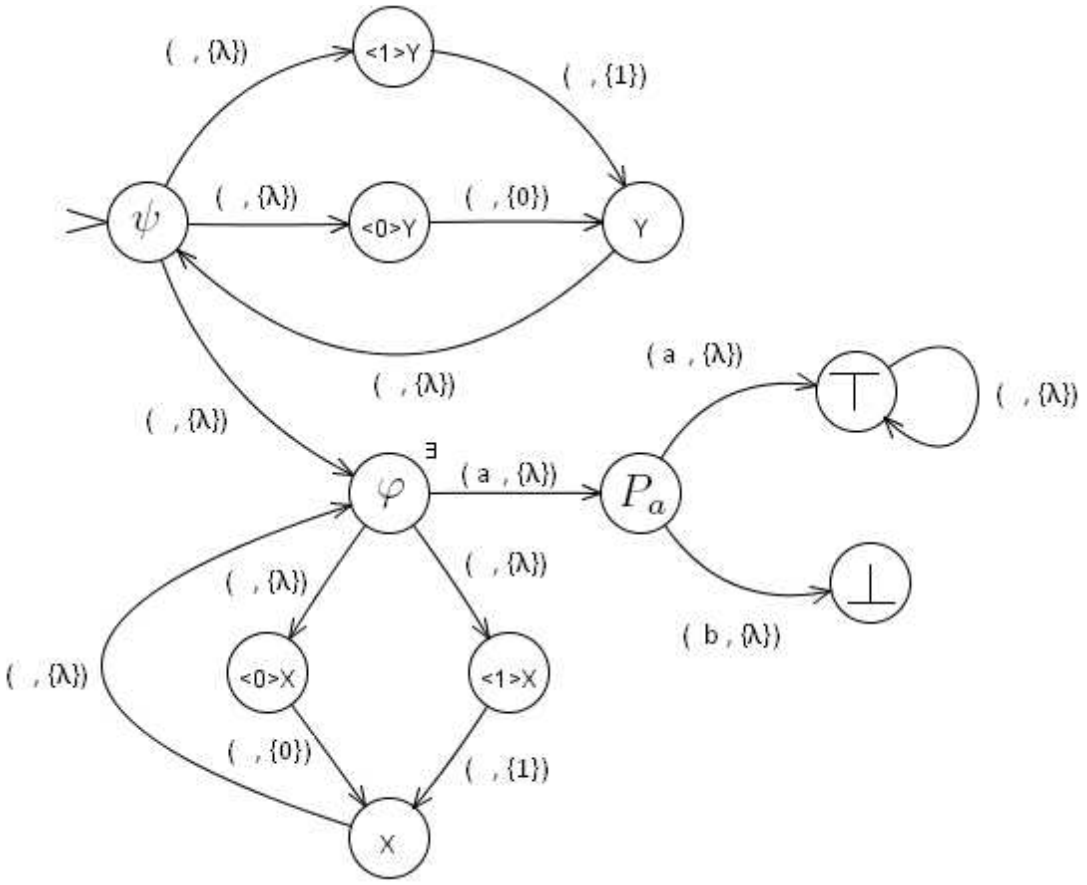


Figura 3.2: Autómata para $\mathcal{A}_\psi$

b Considere el árbol donde todos los nodos debajo 01 (con 01 incluido) están marcados con b. La estrategia ganadora para $\forall$, en este caso en el nodo ψ, empieza escogiendo $\langle 0\rangle Y$, luego en la segunda visita a ψ, escoger $\langle 1\rangle Y$, y en la tercera escoger φ. No importa que haga $\exists$ aquí, $\exists$ pierde.

Para realizar la traducción del lenguaje generado por un autómata alternante al lenguaje generado por una fórmula del μ-cálculo basada en el autómata, se introducirá una sintaxis vectorial para las μ-fórmulas.
En esta ocasión sólo se introducirá la notación necesaria para expresar la fórmula y se realizará un esquema general de la demostración del teorema de traducción.

Dado un entero n, una fórmula vectorial del μ-cálculo, n-aria, es de la forma:

$$\sigma_1 \begin{pmatrix} X_1^1 \\ \vdots \\ X_n^1 \end{pmatrix} \dots \sigma_m \begin{pmatrix} X_1^m \\ \vdots \\ X_n^m \end{pmatrix} \cdot \begin{pmatrix} \varphi_1 \\ \vdots \\ \varphi_n \end{pmatrix}$$

donde, para todo m los $\sigma_1, \dots, \sigma_n$ son operadores de punto fijo, y las fórmulas $\varphi_1, \dots, \varphi_n$ son fórmulas del μ-cálculo tradicional sin operadores de punto fijo.

Definición 3.1.12. Para una estructura $\mathcal{M}_t$ basada en árboles definimos la semántica de las fórmulas vectoriales de aridad n por medio de la valuación $V : FBF_{\mu-\text{vectorial}} \longrightarrow \underbrace{\{0,1\}^* \times \cdots \times \{0,1\}^*}_{n-veces}$
de la siguiente manera: Para una fórmula sin operadores de punto fijo, $V((\varphi_i, \dots, \varphi_n)) = V(\varphi_1) \times \cdots \times V(\varphi_n)$
Para una fórmula con puntos fijos tenemos: $V(\mu\vec{X}.\vec{\varphi}(\vec{X})) = \bigcap \left\{ \vec{S} \subseteq (\{0,1\}^*)^n \,|V_{[\frac{\vec{S}}{\vec{X}}]} \subseteq \vec{S} \right\}$

La notación $(\{0,1\}^*)^n$ es una abreviación de $\underbrace{\{0,1\}^* \times \cdots \times \{0,1\}^*}_{n-veces}$

Lema 3.1.13. *Para toda fórmula $\vec{\alpha}$ del μ-cálculo vectorial, existe una formula φ en el μ-cálculo ordinario tal que para todo modelo $(\mathcal{M}, V)$ se tiene que:* $[(\vec{\alpha}) \downarrow_1]_V^{\mathcal{M}} = [\varphi]_V^{\mathcal{M}}$

Demostración. La prueba de este lema esta basada en la aplicación del lema de Bekič [5] [11]. □

Ejemplo 3.1.14. Considere la fórmula φ del μ-cálculo vectorial de aridad 2 dada por:

$$\mu \begin{pmatrix} X_1 \\ X_2 \end{pmatrix} \cdot \begin{pmatrix} X_1 \vee \langle 0 \rangle X_2 \\ X_1 \wedge P_a \end{pmatrix}$$

Utilizando el lema 3.1.13 tenemos $(\varphi) \downarrow_1 = \mu X_1.(X_1 \vee \langle 0 \rangle(\mu X_2.X_1 \wedge P_a))$.

Para introducir la construcción de la fórmula para un autómata alternante de Mostowski $\mathcal{A}$, se definirán dos funciones que facilitan la expresión de $\varphi_{\mathcal{A}}$.

$$\lceil \delta(q_i, a) \rceil = \begin{cases} \bigvee \{ \lceil (q', d') \rceil | (q', d') \in \delta(q_i, a) \} & \text{si } q_i \in Q_\exists \\ \bigwedge \{ \lceil (q', d') \rceil | (q', d') \in \delta(q_i, a) \} & \text{si } q_i \in Q_\forall \end{cases}$$

$$\text{donde } \lceil \delta(q_j, d) \rceil = \begin{cases} \langle 0 \rangle X_j^j & \text{si } d = 0 \\ \langle 1 \rangle X_j^j & \text{si } d = 1 \\ X_j^j & \text{si } d = \lambda \end{cases}$$

Dado $\mathcal{A} = \langle Q, Q_\exists, Q_\forall, \Sigma, q^0, \delta, Acc\rangle$ y su función de Mostowski Ω, defina $q_1, \ldots, q_n$ un ordenamiento de los estados de $\mathcal{A}$ dado por $\Omega(q_i) \leq \Omega(q_j)$ si $i < j$. La fórmula para $\mathcal{A}$ es:

$$\varphi_\mathcal{A} = \sigma_1 \vec{X_1} \ldots \sigma_n \vec{X_n} \cdot \begin{pmatrix} \bigvee_{a\in\Sigma}(P_a \wedge \lceil\delta(q_1, a)\rceil) \\ \vdots \\ \bigvee_{a\in\Sigma}(P_a \wedge \lceil\delta(q_n, a)\rceil) \end{pmatrix}$$

donde σ_i es μ si $\Omega(q_i)$ es impar, de lo contrario $\sigma_i = \nu$

Teorema 3.1.15. *Para todo árbol t, $t \in \mathcal{L}(\mathcal{A})_\varphi$ si y solo si $\mathcal{M}_t \models (\varphi_\mathcal{A}) \downarrow_1$.*

Demostración. La demostración de este teorema se realiza de forma análoga a la presentada en el teorema 3.1.9, utilizando el lema de decrecimiento de signatura.
Si $\mathcal{M}_t \models (\varphi_\mathcal{A}) \downarrow_1$, la estrategia ganadora para $\exists$ consta en escojer, en una posición (q, v) existencial, para la cual $v \in P_a^\mathcal{M}$, una transición $(q', d') \in \delta(q, a)$ tal que $\mathcal{M}_t \models \lceil\delta(q', d')\rceil$, la cual siempre debe existir. La nueva posición es (q', vd').
Para la otra parte, si $\mathcal{M}_t \not\models (\varphi_\mathcal{A}) \downarrow_1$, entonces la estrategia ganadora para el jugador $\forall$ es escoger en una posición universal (q, v) para la cual $v \in P_a^\mathcal{M}$, una transición $(q', d') \in \delta(q, a)$ tal que $\mathcal{M}_t, v \not\models \lceil\delta(q', d')\rceil$. De nuevo, esa siempre debe existir. □

Nota Final: Aplicaciones

En los capítulos anteriores se expusieron algunas traducciones existentes entre los autómatas, finitos e infinitos alternantes, y las lógicas de segundo orden y μ-cálculo respectivamente. En esta sección mostraremos algunos resultados que se pueden obtener acerca de autómatas a partir de resultados conocidos acerca de la lógica correspondiente, y viceversa, usando las traducciones.

Satisfacibilidad

El problema de satisfacibilidad en LMSO sobre palabras finitas consiste en poder decidir si dada un fórmula φ existe una palabra ω tal que φ vale en el modelo $\mathcal{M}_\omega$. Vía la traducción presentada en el teorema 2.1.7, este problema se puede resolver construyendo el autómata $\mathcal{A}_\varphi$ y contestando a la pregunta de si $\mathcal{L}(\mathcal{A}_\varphi) \neq \phi$. En el caso de los autómatas finitos se sabe siempre cómo hacer esto: se trata sólo de buscar un camino que conecte el estado inicial con uno final. Los modelos generados por estas palabras son precisamente los que valida la fórmula φ.

Propiedades de Cerradura de los Autómatas Alternantes

En el Capítulo 1 fue posible observar algunas propiedades de cerradura de los lenguajes regulares utilizando las propiedades de los autómatas determinísticos de estados finitos, y teniendo en cuenta el resultado del Teorema 1.1.7, se tienen los mismos resultados para los autómatas no determinísticos.

Es natural preguntarse si estas propiedades también se cumplen para los autómatas alternantes. La respuesta a esto es afirmativa, este tipo de autómatas cumple el mismo tipo de propiedades de cerradura que los autómatas de estados finitos, aunque la construcción de la demostración no se puede realizar de forma intuitiva, como pasó en el caso anterior. En este punto se puede utilizar la transición realizada entre el μ-cálculo y éste tipo de estructuras, dado que las fórmulas del μ-cálculo son cerradas bajo unión, intersección y complemento. El método para realizar dicha demostración en el caso de la unión de autómatas alternantes, por ejemplo, es el siguiente:

Dados dos autómatas alternantes $\mathcal{A}_1$ y $\mathcal{A}_2$ utilice el Teorema 3.1.9 para encontrar las fórmulas φ_1 y φ_2 respectivamente. Realice la disjunción de las dos fórmulas $\varphi = \varphi_1 \vee \varphi_2$ y utilice el Teorema 3.1.15 para encontrar el autómata asociado $\mathcal{A}_\varphi$.

Las otras propiedades se demuestran de forma similar.

Bibliografía

[1] Walukiewics, Igor. *Automata and Logic.* Warszawa. Universidad de Warzaw.

[2] Hopcroft, John E. Ullman, Jeffrey D. *Formal Languages And Their Relation To Automata.* Addison-Wesley, Massachusetts, 1969.

[3] Denning, Peter J. Dennis, Jack B. *Machines, Languages, and Computation.* Prentice-Hall, New Jersey, 1978.

[4] Takahashi, Silvia. Notas en Teoría de Lenguajes. 2002.

[5] Schneider, Klaus. *Verification of Reactive Systems.* Springer-Verlag, Berlin, 2004.

[6] Libkin, Leonid. *Elements of Finite Model Theory.* Springer-Verlag, berlin, 2004.

[7] Bradfield, Julian. Stirling, Colin. *Modal Logics and mu-calculi: an introduction.* Edinburgo. Universidad de Edinburgo.

[8] Mukund, Madhavan. *Finite-state Automata on Infinite Inputs.* Madras, 1996. SPIC Mathematical Institute.

[9] Thomas, Wolfgang. *Languages, Automata and Logics.* , 1996.

[10] Vardi, Moshe Y. *Alternating Automata: Checking Truth and Validity for Temporal Logics.* Houston. Rice University.

[11] Kuncak, Victor. Leino, Rustan K. *On Computing the Fixpoint of a Set of Boolean Equations.* Redmond. 2003, Microsoft Corporation.