Metodología de síntesis de autómatas para controlar sistemas de navegación autónoma terrestre

A methodology to automaton synthetize in ground autonomous navigation systems control

Jorge Luis Martínez Valencia, Mauricio Holguín Londoño, Germán Andrés Holguín Londoño
Facultad de Ingenierías, Universidad Tecnológica de Pereira, Pereira, Colombia
jolumartinez@utp.edu.co
mau.hol@utp.edu.co
gahol@utp.edu.co

***Resumen*— En este trabajo se plantea una metodología para la síntesis de autómatas en aplicaciones de navegación autónoma terrestre con una tarea global. La metodología propuesta se basa en el diseño de autómatas por gramáticas regulares y permite generar una política de control para la conducción autónoma de un vehículo en un ambiente parcialmente controlado mientras se extrae información del entorno a través de sensores. Además, se plantean los inconvenientes que se presentan al tratar de abordar esta aplicación con enfoques tradicionales que implican la síntesis de autómatas finitos no deterministas. Por último, se presentan los autómatas obtenidos al validar la metodología propuesta a través de simulaciones utilizando el software MATLAB con el Toolbox integrado de realidad virtual (V-Realm Builder).**

***Palabras clave*— Autómatas, gramáticas regulares, navegación autónoma terrestre, explosión combinacional de estados.**

***Abstract*— In this work we propose a methodology for the synthesis of automatons in autonomous ground navigation systems with a global task. Our method is based on the design of automatons using regular grammars allowing for the generation of control policies for autonomous driving in a partially controlled environment where information is extracted using sensors. Furthermore, we showcase the problems that arise when approaching this problem with traditional synthesis of finite non-deterministic automatons. Finally, in the results section, we present the validation of the proposed method with simulations using MATLAB© and the Toolbox for Virtual Reality (V-Realm Builder).**

***Key Word* — Automatons, regular grammars, ground autonomous navigation, combinatorial explosion of states.**

I. INTRODUCCIÓN

La dificultad de las máquinas para la comunicación entre ellas y comprender el entorno humano ha retenido el desarrollo de tecnologías más complejas que puedan conducir tareas autónomas con desenvolvimiento en entornos físicos compartidos con seres humanos. Por ejemplo, la competencia europea SAUC-E (Student Autonomous Underwater Vehicle Challenge-Europe) reta a los académicos a desarrollar vehículos submarinos autónomos (AUV). Dentro de sus aplicaciones más comunes se encuentra la elaboración de cartografía del fondo del mar, la medición de propiedades del agua con fines ambientales, la detección y eliminación de minas submarinas, el reconocimiento del campo de batalla y la protección de puertos marítimos [2]. Por otro lado, estos ambientes dinámicos, plantean desafíos únicos para el desarrollo de técnicas de control y verificación, ya que existen muchos factores a tener en cuenta como pueden ser: la dinámica del ruido del sistema, el acondicionamiento de sensores y transductores, las estrictas condiciones de seguridad que deben asumirse al interactuar con seres humanos, la interacción con usuarios inexpertos y otros. Los modernos sistemas de ingeniería planteados para afrontar este tipo de desafíos poseen un estrecho vínculo entre elementos computacionales y físicos lo que hace que su diseño y verificación sean cada vez más complejos debido al entrelazamiento entre las lógicas de alto nivel y las dinámicas de bajo nivel [3].

Si se analiza un problema de conducción autónoma terrestre urbana como el propuesto en el DARPA Challenge, se tiene que para que los vehículos completen con éxito la carrera, necesitan estar en capacidad de maniobrar de manera autónoma, manejar los cambios en el ambiente o en las condiciones de operación y hacer nuevos planes en respuesta a estos cambios. Por lo tanto, la lógica de alto nivel que rige el comportamiento de los vehículos necesita estar correctamente integrada con el controlador de bajo nivel que regula el hardware físico. La planificación del movimiento y la planificación de tareas son dos problemas fundamentales en la robótica que se han abordado desde diferentes perspectivas. Las técnicas de

Fecha de Recepción: 19 de febrero de 2018
Fecha de Aceptación: 15 de noviembre de 2018

planificación del movimiento Bottom-Up se concentran en crear entradas de control o controladores de lazo cerrado que dirigen un robot de una configuración a otra [4], [5], teniendo en cuenta diferentes dinámicas y restricciones de movimiento. Tales controladores pueden asumir como perfecto el comportamiento del ambiente o recibir la información sobre el entorno a través de sensores [6]. Por otro lado, los enfoques de planificación de tareas Top-Down usualmente se enfocan en encontrar acciones gruesas del robot, típicamente discretas, para lograr tareas más complejas. Tales tareas pueden incluir robots con múltiples objetivos finales [7], ordenación temporal o secuenciación de objetivos [8].

La descomposición jerárquica tradicional de los problemas de desplazamiento en capas de planificación de tareas que se ubican más arriba en la jerarquía que las capas de planificación de movimiento, dio lugar hasta hace poco tiempo a una carencia de enfoques que abordaran el problema de forma integrada [9]. Los avances en sistemas híbridos que acoplan sistemas continuos y discretos han permitido la integración formal de acciones discretas de alto nivel con controladores de bajo nivel [10], lo cual inspiró una variedad de enfoques que traducen tareas discretas de alto nivel a controladores continuos de bajo nivel de una manera verificable y computacionalmente eficiente [11], [12] [13]. En [14], [15] se sostiene que el éxito en el desarrollo de vehículos y robots autónomos en general, requerirá de avances en enfoques formales tanto para la verificación como para la correcta construcción y síntetización de controladores embebidos.

En este trabajo se plantea el desarrollo de una metodología para sintetizar autómatas que garanticen trayectorias continuas para robots móviles en los que las especificaciones de movimiento son dadas como un conjunto de reglas de una gramática regular asegurando que el comportamiento continuo del robot satisface las especificaciones dadas para todos los comportamientos válidos del ambiente. Se consideran particularmente las reacciones específicas en las cuales el comportamiento del robot es diferente dependiendo de la información que este recopila a través de sus sensores en el tiempo de ejecución. Por ejemplo, si el robot encuentra un bloqueo en la vía, este debe reaccionar para garantizar el cumplimiento del objetivo principal. Para abordar el problema de síntesis planteado se construye, a través del uso de lenguajes formales, una gramática regular que permita la síntesis de autómatas que modelen de manera abstracta el sistema físico, el cual normalmente tiene infinitos estados [16], [17] y con base en este modelo abstracto se sintetiza una estrategia de control que satisfaga las especificaciones. Lo anterior desemboca en un diseño jerárquico de dos capas con un conjunto de autómatas que representan los movimientos básicos del vehículo y un secuenciador global que sintetiza el sub-conjunto adecuado de reglas de la gramática regular para responder a los cambios del entorno. Al integrar el autómata obtenido con los controladores continuos, el resultado es un controlador híbrido global que orquesta la composición de los controladores de bajo nivel basándose en la información reunida sobre el entorno.

II. CONTENIDO

A. Preliminares.

1) *Lenguajes:* Para comprender el concepto de lenguaje es necesario definir nociones más elementales como símbolo, alfabeto y palabra. Símbolo es básicamente una representación distinguible de cualquier información en una entidad indivisible; alfabeto es un conjunto no vacío de símbolos. Por ejemplo el alfabeto del idioma español sería $\mathrm{E} = \{\mathrm{a}, \mathrm{b}, \mathrm{c}, \mathrm{d}, \ldots, \mathrm{z}\}$ y palabra es la secuencia o cadena de caracteres construida con los símbolos de un alfabeto. De esta manera, un lenguaje es simplemente un conjunto de palabras [18]. Todos los lenguajes que comparten ciertas propiedades dadas pueden ser clasificados en conjuntos, y ya que los lenguajes en sí son conjuntos de secuencias de símbolos, se infiere que las clases de lenguajes son conjuntos de conjuntos de secuencias de símbolos [19]. N. Chomsky propuso una jerarquía de lenguajes donde las clases más complejas incluyen a las clases más simples [18]. Se distinguen tres clases de lenguajes fundamentales en la teoría de autómatas. En primer lugar están los lenguajes regulares que son todos los lenguajes que se pueden formar a partir de los lenguajes básicos por medio de operaciones básicas como unión, concatenación y la cerradura de Kleene [19]; en segundo lugar están los lenguajes libres de contexto que son lenguajes representados por un conjunto finito de variables donde cada una representa un lenguaje y finalmente los lenguajes recursivamente enumerables que son lenguajes con la capacidad de ser reconocidos por las máquinas de Turing. Este último contiene todos los anteriores.

Los lenguajes regulares son llamados así debido a que sus palabras contienen regularidades o repeticiones de los mismos componentes. Un lenguaje L es regular si y solo si cumple al menos una de las siguientes condiciones: L es finito; dados dos lenguajes regulares R_1 y R_2, $L = R_1 \cup R_2$ o $L = R_1 R_2$ respectivamente; L es la cerradura de Kleene de algún lenguaje regular, $L = R^*$ [20].

2) *Expresiones regulares:* Las expresiones regulares hacen referencia a la definición de un lenguaje en el que cada una de las palabras que lo conforman denota un lenguaje regular tal que si Σ es un alfabeto, el conjunto ER de las expresiones regulares sobre Σ contiene las cadenas en el alfabeto $\Sigma \cup \{"\wedge","+","\cdot","*","(",")","\Phi"\}$ cumpliendo con: "$\wedge$" y "Φ" $\in ER$; si $\sigma \in \Sigma$, entonces $\sigma \in ER$; si $E_1, E_2 \in ER$, entonces $"(" E_1 + E_2 ")" \in ER$, "$(E_1 \cdot E_2)$" $\in ER$, $"(" E_1 ")^*" \in ER$. Esto permite obtener una notación en la que las representaciones de los lenguajes son simplemente texto o palabras de otro lenguaje [19]. La importancia de las expresiones regulares radica en la posibilidad de expresar de forma declarativa las cadenas o palabras que se desean aceptar de determinado lenguaje [18]. De esta manera, el significado de una ER es una función $\mathcal{L}: ER \rightarrow 2^{\{\Sigma^*\}}$ que toma como entrada una expresión regular y entrega como salida un lenguaje definido como sigue:

- $\mathcal{L}\,("\Phi") = \emptyset\,(el\ conjunto\ vacío)$
- $\mathcal{L}\,("\wedge") = \{\varepsilon\}$
- $\mathcal{L}\,(\sigma) = \{\sigma\}, \sigma \in \Sigma$
- $\mathcal{L}\,((R{\cdot}S)) = \mathcal{L}(R)\mathcal{L}(S), R, S \in ER$
- $\mathcal{L}\,("("R{+}S")") = \mathcal{L}(R) \cup \mathcal{L}(S), R, S \in ER$
- $\mathcal{L}("("R")") = \mathcal{L}(R)^*, R \in ER$

3) *Gramáticas formales y regulares:* Una gramática es un conjunto de reglas para formar correctamente las frases de un lenguaje. La representación comúnmente aceptada de una gramática es la debida a N. Chomsky [21], y está basada en las llamadas reglas gramaticales, las cuales son una expresión de la forma $\alpha \to \beta$ en donde α y β son cadenas de símbolos en las cuales aparecen elementos del alfabeto Σ y nuevos elementos llamados *variables*. La aplicación de una regla $\alpha \to \beta$ a una palabra $\mu\alpha\upsilon$ produce la palabra $\mu\beta\upsilon$, por lo que las reglas de una gramática pueden ser vistas como reglas de reemplazo [22].

El interés se centra en las gramáticas regulares cuyas reglas son de la forma $A \to aB$ o bien $A \to a$, donde A y B son variables, y a es un caracter terminal o constante [23]. Para aplicar una gramática se parte de una variable llamada *símbolo inicial*, y se aplican repetidamente las reglas gramaticales hasta que ya no haya variables en la palabra, lo cual indica que la palabra resultante es parte del lenguaje de esa gramática [24]. Formalmente una gramática se define como una cuádrupla (V, Σ, R, S) en donde V es un alfabeto de variables; Σ es un alfabeto de constantes; R, el conjunto de reglas, es un subconjunto finito de $V \times (\Sigma V \cup \Sigma)$ y S, el símbolo inicial que es un elemento de V. Para formalizar la aplicación de una gramática se debe tener en cuenta que una cadena $\mu\, X\, \upsilon$ deriva en un paso una cadena $\mu\, \alpha\, \upsilon$, escrito como $\mu\, X\, \upsilon \Rightarrow \mu\, \alpha\, \upsilon$, si hay una regla $X \to \alpha \in R$ en la gramática. Una palabra $w \in \Sigma^*$ es derivable a partir de $G\ ssi\ S \Rightarrow^* w$ donde $\Rightarrow^*$ denota la cerradura reflexiva y transitiva de $\Rightarrow$. De esta manera, el lenguaje generado por una gramática $G, L(G)$, es igual al conjunto de las palabras derivables a partir de su símbolo inicial.

$$L(G) = \{w \in \Sigma^* | S \Rightarrow^* w\}$$

4. *Autómatas:* Los autómatas finitos corresponden a las máquinas abstractas más simples, comprendiendo como máquina abstracta simple a las abstracciones matemáticas que capturan solamente el aspecto referente a las secuencias de eventos que ocurren, sin tomar en cuenta la forma de la máquina, sus dimensiones, o si efectúa movimientos rectos o curvos. Además, los autómatas finitos se encuentran estrictamente relacionados con los lenguajes regulares [18]. Un autómata de estados finitos M es una quíntupla $(Q, \Sigma, \delta, q_0, F)$, tal que Q es un conjunto finito de estados, Σ es un alfabeto de entrada, $q_0 \in Q$ es el estado inicial, $F \subseteq Q$ es el conjunto de estados finales y δ es la función de transición que proyecta $Q \times \Sigma$ en Q. Es decir, $\delta(q, a)$ es una transición para cada estado q y cada símbolo de entrada a. Debido a que δ es una función y no solo una relación. Para un estado y un símbolo del alfabeto dados, habrá un y solo un estado siguiente, lo que permite establecer que la definición dada corresponde a la de un autómata de estados finitos determinista AFD [22].

$$M = (Q, \Sigma, \delta, q_0, F)$$

Los AFD pueden ser utilizados para reconocer ciertas palabras y diferenciarlas de otras. Para que un AFD reconozca o acepte una palabra, se debe cumplir que todos los caracteres de dicha palabra sean consumidos siguiendo las transiciones y pasando de un estado a otro y que el estado al que se llega al terminar la palabra sea uno de los estados finales del autómata. Dicho esto se puede definir que el lenguaje aceptado por un autómata, o máquina M, es el conjunto de palabras aceptadas por dicha máquina lo que en notación formal sería: una palabra $w \in \Sigma^*$ es aceptada por una máquina $M = (K, \Sigma, \delta, s, F)\ ssi$ existe un estado $q \in F$ tal que $[[s, w]] \vdash_m^* [[q, \varepsilon]]$ garantizando que lo que falta por leer al llegar a un estado final es la palabra vacía ε. $\vdash_m$ denota una relación definida formalmente como $[[q_1, \sigma\, w]] \vdash_m [[q_2, w]]$ para un $\sigma \in \Sigma$ si y solo si existe una transición en M tal que $\delta(q_1, \sigma) = q_2$ siendo σ el caracter que se leyó y $\vdash_m^*$ representa la cerradura reflexiva y transitiva de $\vdash_m$ [18].

En los autómatas finitos no deterministas (AFN), a diferencia de los deterministas, no se sabe exactamente cuál es la transición que se debe llevar a cabo ante un determinado evento e incluso pueden existir varias transiciones entre dos estados. Los AFD son un caso particular de los AFN, por lo que se puede decir que todo AFD es de hecho un AFN [22]. Así mismo existen los autómatas finitos con salidas que no se limitan al hecho de aceptar, o no, una palabra. Su aplicación se enfoca a los sistemas físicos, ya que en estos lo importante es que el sistema sea capaz de responder al entorno dependiendo del estado actual y no solo de la inicial. Existen dos formas de definir los autómatas finitos con salidas, la primera se conoce como autómata de Moore propuesto por E. Moore [25] y la segunda, como autómata de Mealy propuesto por G. Mealy [18]. Un autómata de Moore o máquina de Moore es una séxtupla $M = (K, \Sigma, \Gamma, \delta, \lambda, q_0)$ donde K, Σ y δ son como en los AFD, y q_0 es el estado inicial; además se tiene Γ que es el alfabeto de salida y λ que es una función de K a Γ^*, que obtiene la salida asociada a cada estado [25] y en las máquinas de Mealy la salida depende del estado en que se encuentra el autómata y de la transición que se ejecuta, por lo tanto una máquina de Mealy es una séxtupla $M = (K, \Sigma, \Gamma, \delta, \lambda, q_0)$, en donde todos los componentes son equivalentes a los de las máquinas de Moore con excepción de λ que es una función $\lambda : K \times \Sigma \to \Gamma^*$ lo que indica que se produce una palabra formada por componentes de Γ a partir de un elemento tomado de $K \times \Sigma$ [18].

Por último, es necesario definir el autómata de maniobras, también llamado planificador. Este es el encargado de tomar las decisiones para que el autómata pueda funcionar de manera simple, sin necesidad de programar la totalidad de las reglas. Se encarga de verificar el estado en el que se encuentra el autómata, y dependiendo de la entrada, elige y envía las reglas

o instrucciones que debe utilizar el autómata para lograr un correcto desempeño. Este proceso lo realiza en instantes cortos de tiempo para disminuir el problema de explosión combinacional de estados que se presenta con la variación de los valores de verdad de los estados del autómata. A esta forma de actuar se le llama retroceso de horizonte corto [17], y consiste en que el autómata de maniobras elabora un plan para un espacio de tiempo corto y lo ejecuta constantemente, en lugar de planear una ruta larga que necesita una máquina de estados más grande, además de un procesador más eficiente. Debido a que el entorno cambia dinámicamente, se evalúan las entradas para espacios de tiempo cortos y se genera un autómata que cumpla las características para las entradas evaluadas en ese espacio de tiempo [26]. Este proceso se repite sucesivamente.

B. Formulación del problema.

El objetivo de este trabajo es generar y validar una metodología para sintetizar autómatas para controlar robots móviles que puedan reaccionar evadiendo obstáculos en el camino. Para alcanzar este objetivo, se necesita especificar un *modelo de robot*, un *ambiente parcialmente controlado* y una *tarea global*.

2) *Modelo de robot:* Se asume que el robot móvil opera en un espacio de trabajo poligonal planar P. La estructura física propuesta para el robot con sus respectivos sensores se puede observar en la figura 1 y el movimiento del robot está expresado por:

$$\dot{p}(t) = u(t) \quad p(t) \in P \subseteq R^2 \quad u(t) \in U \subseteq R^2 \qquad (1)$$

Donde $p(t)$ es la posición del robot en el tiempo t y $u(t)$ es la entrada de control. Se asume que el espacio de trabajo P está particionado en un número finito de celdas $P_1, \ldots, P_n$, donde $P = \cup_{\{i=1\}}^{n} P_i$ y $P_i \cap P_j = \emptyset$ si $i \neq j$. Además, se considera cada una de las celdas como un polígono convexo. Al particionar el espacio de trabajo se genera una serie de proposiciones Booleanas $\{r_1, r_2, \ldots, r_n\}$ las cuales son ciertas si el robot se encuentra en la posición P_i.

$$r_i = \begin{Bmatrix} True\ SI\ \ p \in P_i \\ False\ SI\ \neg(p \in P_i) \end{Bmatrix}$$

Debido a que $\{P_i\}$ es una partición de P, solamente un r_i puede ser cierto en cualquier momento.

Adicional a esto, en este caso el robot puede ejecutar en cualquier momento acciones extras como encender una luz. Esto implica que estas acciones también deben estar codificadas como proposiciones atómicas. Para este conjunto de proposiciones $A = \{a_1, a_2, \ldots, a_k\}$ se tiene:

$$r_i = \begin{Bmatrix} True \text{ cuando SI se ejecuta la acción i} \\ False \text{ cuando NO se ejecuta la acción i} \end{Bmatrix}$$

Se define entonces $act(t) \subseteq A$ como el conjunto de acciones que están activas o que son verdad en un tiempo t, es decir: $a_i \in act(t)$ SI a_i es verdad en un tiempo t. Entonces el conjunto Y que define todas las proposiciones asociadas al robot está dado por la unión de los conjuntos que representan las proposiciones de las acciones y las proposiciones de localización en el espacio de trabajo, esto es $Y = P \cup A = \{r_1, r_2, \ldots, r_n, a_1, a_2, \ldots, a_k\}$. Si dos acciones de este conjunto no pueden ser ciertas al mismo tiempo, se debe agregar a las especificaciones del sistema.

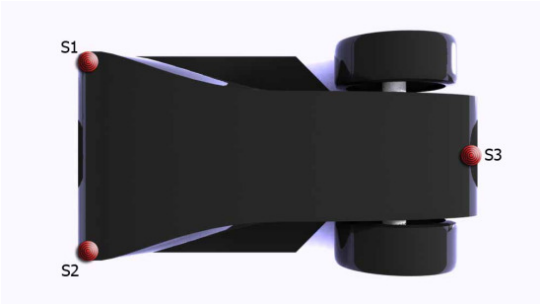

Fig. 1. Estructura propuesta para el desarrollo del problema.

La figura 1 es un modelo de la estructura física del robot propuesto para el desarrollo del problema y en ella se puede apreciar que se propone el uso de 3 sensores por medio de los cuales el robot recibe información del ambiente parcialmente controlado concerniente a los obstáculos que pueden haber, o no, en el camino.

2) *Ambientes permitidos:* El robot interactúa con su entorno a través de sensores, los cuales para este trabajo son asumidos como de salida binaria y también se asume que los sensores siempre van a entregar información confiable y precisa del entorno. Las m variables binarias de sensado $X = \{x_1, x_2, \ldots, x_m\}$ tienen su propia dinámica discreta. Se hace un análisis sobre el posible comportamiento de las variables del sensor para definir una clase de entornos admisibles en los que se desenvuelve el robot. Estas suposiciones del entorno se capturan en el conjunto general de reglas gramaticales, que se define en la sección (II-C), lo cual anula la posibilidad de utilizar entornos definidos arbitrariamente para garantizar que los controladores alcancen las especificaciones deseadas para un entorno permitido.

3) *Especificaciones del sistema:* Para facilitar la definición de las especificaciones deseadas del sistema para el robot, se propone una pista con obstáculos como la que se aprecia en la figura 2, donde los espacios libres están representados por ceros y los espacios ocupados por los obstáculos están representados por unos. La tarea global que debe cumplir el robot lingüísticamente hablando se expresa como "alcance la meta mientras evade los obstáculos presentes en el camino". En este caso, la meta es la línea verde ubicada en la parte inferior de la pista propuesta.

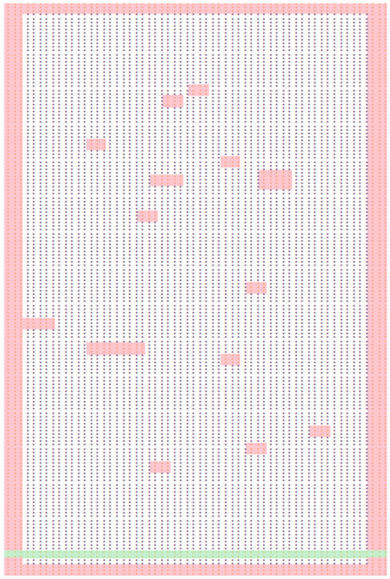

Fig. 2. Pista propuesta para definir las especificaciones del sistema.

Tomando los ítems anteriores se puede describir el problema que se trata en este trabajo como: dado un modelo de robot (1) y una gramática regular que modele un conjunto de ambientes, construir si es posible un controlador para el robot con el cual se alcancen las especificaciones del sistema en un entorno permitido partiendo de cualquier estado inicial.

C. Síntesis del autómata.

1) *Diseño de la gramática regular:* La gramática regular se define formalmente como un cuádruplo $GR = (V, \Sigma, R, S)$, donde:

- V es el conjunto de variables del sistema,
- Σ es el alfabeto de entrada,
- R es el conjunto que contiene todas las reglas de la gramática,
- S es el estado inicial permitido para el sistema

Como conjunto de variables del sistema se define $V = \{V1, V2, P, Gd, Gi, R\}$ donde:

- V1 = Velocidad 1
- V2 = Velocidad 2
- P = Pare
- Gd = Giro a la derecha
- Gi = Giro a la izquierda
- R = Retroceder

Como conjunto de constantes del sistema se define $\Sigma = \{S1, S2, S3, S4, S5, t\}$ donde: $S1$, $S2$ y $S3$ son los sensores del vehículo que se aprecian en la figura 1, $S4 = S1 \wedge S2, S5 = \neg S1 \wedge \neg S2 \wedge \neg S2$ y t es un tiempo de giro dado en segundos que sirve para que el vehículo gire para ubicarse el mismo tiempo que giró para evitar un obstáculo.

Después de definir los conjuntos de variables y constantes del sistema se crea un posible conjunto total de reglas gramaticales para el vehículo:

- $S \rightarrow \epsilon V1$, el vehículo siempre comenzará en su estando inicial S y pasará por medio de una transición vacía (ϵ) a avanzar en $V1$.
- $V1 \rightarrow s5V2$, siempre que los sensores del vehículo no detecten nada pasa de $V1$ a $V2$.
- $V1 \rightarrow s3V2$, Siempre que el vehículo detecte en $s3$ pasa a $V2$.
- $V2 \rightarrow s1Gi$, Si el sensor de la derecha $s1$ se activa, el vehículo empieza a girar a la derecha, sin detenerse.
- $V2 \rightarrow s2Gd$, Si el sensor de la izquierda $s2$ se activa el vehículo empieza a girar a la izquierda, sin detenerse.
- $V2 \rightarrow s4P$, Si se activan los sensores delanteros el vehículo se detiene.
- $V2 \rightarrow s5P$, Si se activan todos los sensores el vehículo se detiene.
- $Gi \rightarrow s1Gi$, Gira a la izquierda mientras $s1$ este sensando.
- $Gi \rightarrow tGi$, Gira a la izquierda para corregir la trayectoria.
- $Gi \rightarrow s2Gd$, Se crea una regla adicional para verificar la trayectoria de giro.
- $Gi \rightarrow s1Gd$, Cuando $s1$ ya no está detectando nada pasa a girar a la derecha para corregir la trayectoria.
- $Gd \rightarrow tGd$, Gira a la derecha para corregir la trayectoria.
- $Gd \rightarrow s1Gi$, Se crea una regla adicional para verificar la trayectoria de giro.
- $Gd \rightarrow s2Gd$, Gira a la derecha mientras $s2$ este sensando.
- $Gd \rightarrow s2Gi$, Cuando $s2$ ya no está detectando nada pasa a girar a la izquierda para corregir la trayectoria.
- $P \rightarrow s4R$, Si los dos sensores de adelante detectan obstáculos, entonces, retroceder.
- $P \rightarrow s5$, Cuando el vehículo esté detenido y todos los sensores están activados, termina el programa.
- $V1 \rightarrow s1$, Estado final.
- $V1 \rightarrow s2$, Estado final.
- $V2 \rightarrow \epsilon$, Estado final.
- $P \rightarrow \epsilon$, Estado final.
- $R \rightarrow \epsilon$, Estado final
- $Gd \rightarrow s2$, Estado final
- $Gd \rightarrow s1$, Estado final
- $Gd \rightarrow s4$, Estado final
- $Gi \rightarrow s2$, Estado final
- $Gi \rightarrow s1$, Estado final
- $Gi \rightarrow s4$, Estado final

2) *Diseño del autómata:* El autómata de maniobras es el encargado de escoger el conjunto de reglas necesarias para el funcionamiento del autómata en cada instante según la entrada y el estado inicial en que se encuentra. El principio para seleccionar las reglas es que solo se verifica el estado actual y la entrada. Para el robot planteado se proponen las combinaciones consignadas en la tabla I.

TABLA I
TABLA DE ESTADOS DEL AUTÓMATA

Entrada sensor	Constantes del sistema	Estado inicial	Estado final	Acción a tomar
000	S5	S	Z	Avanzar
001	S1	V2	Z	Giro a la izquierda
010	S2	V2	Z	Giro a la derecha
011	S4	V2	Z	Retroceder
100	S3	V2	Z	Avanzar
101	$S3 \wedge S1$	V2	Z	Giro a la izquierda
110	$S3 \wedge S2$	V2	Z	Giro a la derecha
111	$\neg S5$	V2	Z	Pare

Partiendo de las combinaciones definidas en la tabla I, se identifica un grupo de cinco acciones (*Avanzar, Giro a la izquierda, Giro a la derecha, Retroceder, Pare*) que cubren los comportamientos posibles del vehículo y se identifican los subconjuntos de las reglas gramaticales correspondientes a cada una de estas acciones para posteriormente sintetizar el autómata y así generar el pseudocódigo para cada acción. A continuación, se describe detalladamente el proceso de diseño, síntesis de autómatas y pseudocódigos de las acciones identificadas de la tabla I.

Avanzar: Al inicio del programa el vehículo siempre comienza en el estado inicial S y pasará a avanzar sin importar la entrada. El subconjunto de reglas gramaticales que describen este comportamiento es $Avanzar = \{(S \rightarrow \epsilon V1), (V1 \rightarrow s5V2), (V1 \rightarrow s3V2), (V1 \rightarrow s1), (V1 \rightarrow s2), \ (V2 \rightarrow \epsilon)\}$. En la figura 3 se puede observar el autómata equivalente al subconjunto anterior de reglas gramaticales.

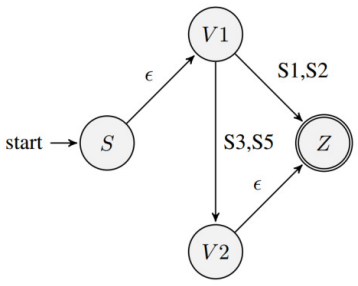

Fig. 3. AFND para la acción "Avanzar".

Con base en la síntesis del autómata que se muestra en la figura 3, se genera el pseudocódigo correspondiente que se muestra en el algoritmo 1.

Algorithm 1 Algoritmo para función "Avanzar"

Require: controlVELOCIDAD, interfazVIRTUAL, matrizDEposiciones
Ensure: posicionCARRO, rotacionCARRO, controlVELOCIDAD, s1, s2, s3, s4, s5, estadoFINAL
1: velocidad1 ← Inicializar
2: velocidad2 ← Inicializar
3: funcion Leer ← s1, s2, s3, s4, s5
4: **if** sensor1, sensor2, sensor5 = 0 **and** estadoFINAL ≠ 1 **or** sensor3 = 1 **then**
5: **if** controlVELOCIDAD **and** rotacionCARRO(angulo)= 0 **then**
6: posicionCARRO(z) ← posicionCARRO(z) + velocidad1
7: controlVELOCIDAD ← controlVELOCIDAD + 1
8: **else if** rotacionCARRO(angulo) > 0 **then**
9: posicionCARRO(x) ← posicionCARRO(x) + 1
10: posicionCARRO(z) ← posicionCARRO(z) + velocidad2
11: **else if** rotacionCARRO(angulo) < 0 **then**
12: posicionCARRO(x) ← posicionCARRO(x) - 1
13: posicionCARRO(z) ← posicionCARRO(z) + velocidad2
14: **else**
15: posicionCARRO(z) ← posicionCARRO(z) + velocidad2
16: **end if**
17: Enviar posición carro a interfazVIRTUAL
18: **end if**

Giro a la izquierda: Si el sensor delantero derecho "s1" detecta un obstáculo en el estado V2, deberá evitar el obstáculo de acuerdo al siguiente sub-conjunto de reglas gramaticales $Giro\ a\ la\ izquierda = \{(V2 \rightarrow s1Gi), (Gi \rightarrow s1Gi), (Gi \rightarrow s1Gd), (Gi \rightarrow s2), (Gi \rightarrow s4), (Gd \rightarrow tGd), \ (Gd \rightarrow s1Gi), (Gd \rightarrow s2), (Gd \rightarrow s4)\}$. En la figura 4 puede observarse el autómata equivalente al subconjunto anterior de reglas gramaticales.

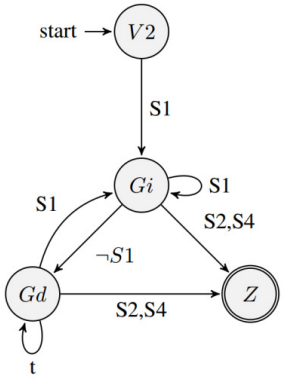

Fig. 4. AFD para la acción "Giro a la izquierda".

Con base en la síntesis del autómata mostrado en la figura 4, se genera el pseudocódigo correspondiente mostrado en el algoritmo 2.

Algorithm 2 Algoritmo para función "Girar a la izquierda"

Require: controlVELOCIDAD, interfazVIRTUAL, matrizDEposiciones
Ensure: posicionCARRO, rotacionCARRO, controlVELOCIDAD, s1, s2 s3, s4, s5, estadoFINAL
Velocidad1 ← Inicializar
Gi ← [0, 1, 0, 0]
Gd ← [0, −1, 0, 0]
funcion Leer ← Leer interfazVIRTUAL, matrizDEposiciones
contGIRO ← 0
contRECTgiro ← 0
if s2 = 0 **and** s1 = 1 **and** estadoFINAL ≠ 1 **then**
  while s2 = 0 y s1 = 1 **do**
    interfazVIRTUAL(traslacion) ← [posicionCARRO]
    funcion Leer ← Leer interfazVIRTUAL, matrizDEposiciones
    while s2, s1, s4 = 0 **and** t < 3 **do**
      interfazVIRTUAL(traslacion) ← [posicionCARRO]
      contGIRO ← contGIRO + 1
      funcion Leer ← Leer interfazVIRTUAL, matrizDEposiciones
    end while
  end while
  while s1, s2 = 0 **and** contRECTgiro ≠ 0 **do**
    interfazVIRTUAL(rotacion) ← rotacionCARRO
    funcion Leer ← Leer interfazVIRTUAL, matrizDEposiciones
  end while
end if

Giro a la derecha: Si el sensor delantero izquierdo "s2" del vehículo detecta un obstáculo en el estado V2, deberá evitar el obstáculo de acuerdo al siguiente sub-conjunto de reglas gramaticales $Giro\ a\ la\ derecha = \{(V2 \rightarrow s2Gd), (Gd \rightarrow s2Gd), (Gd \rightarrow \neg s2Gi), (Gd \rightarrow s1), (Gd \rightarrow s4), (Gi \rightarrow tGi), (Gi \rightarrow s2Gd), (Gi \rightarrow s1), (Gi \rightarrow s4)\}$. En la figura 5 puede observarse el autómata equivalente al subconjunto anterior de reglas gramaticales.

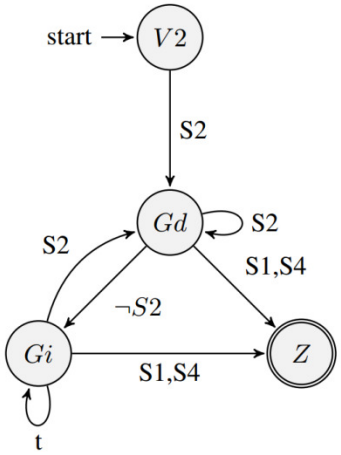

Fig. 5. AFD para la acción "Giro a la derecha".

Con base en la síntesis del autómata mostrado en la figura 5, se genera el pseudocódigo correspondiente mostrado en el algoritmo 3.

Retroceder: Si $(s1 \wedge s2 = True)$, es decir, si los dos sensores delanteros detectan un obstáculo en el estado V2, el vehículo debe retroceder girando siempre a la derecha hasta que $(s1 \wedge s2 = False)$ de acuerdo al siguiente sub-conjunto de reglas gramaticales $Retroceder = \{(V2 \rightarrow s4P), (P \rightarrow s4R), (P \rightarrow \epsilon), (R \rightarrow \epsilon)\}$. En la figura 6 puede observarse el autómata equivalente al subconjunto anterior de reglas.

Algorithm 3 Algoritmo para función "Girar a la derecha"

Require: controlVELOCIDAD, interfazVIRTUAL, matrizDEposiciones
Ensure: posicionCARRO, rotacionCARRO, controlVELOCIDAD, s1, s2 s3, s4, s5, estadoFINAL
Velocidad1 ← Inicializar
Gi ← [0,1,0,0]
Gd ← [0,-1,0,0]
funcion Leer ← Leer interfazVIRTUAL, matrizDEposiciones
contGIRO ← 0
contRECTgiro ← 0
if s2=1 **and** s1=0 **and** estadoFINAL ≠ 1 **then**
  while s2=1 **and** s1=0 **do**
    interfazVIRTUAL(traslacion) ← [posicionCARRO]
    funcion Leer ← Leer interfazVIRTUAL, matrizDEposiciones
    while s2, s1, s4 = 0 **and** t<3 **do**
      interfazVIRTUAL(traslacion) ← [posicionCARRO]
      contGIRO ← contGIRO + 1
      funcion Leer ← Leer interfazVIRTUAL, matrizDEposiciones
    end while
  end while
  while s1, s2 = 0 **and** contRECTgiro ≠ 0 **do**
    interfazVIRTUAL(rotacion) ← rotacionCARRO
    funcion Leer ← Leer interfazVIRTUAL, matrizDEposiciones
  end while
end if

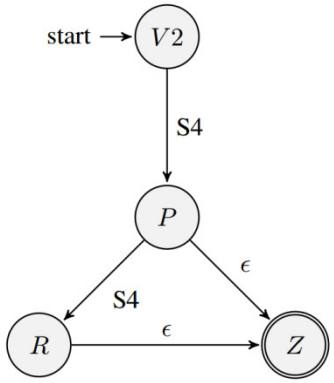

Fig. 6. AFND para la acción "Retroceder".

Con base en la síntesis del autómata que se muestra en la figura 6, se genera el pseudocódigo correspondiente que se muestra en el algoritmo 4.

Pare: A Si $(s1 \wedge s2 \wedge s3 = True)$, es decir, si todos los sensores detectan un obstáculo, el vehículo se detendrá de acuerdo al siguiente sub-conjunto de reglas gramaticales $Pare = \{(V2 \rightarrow \neg s5P), (P \rightarrow \neg s5)\}$. En la figura 7 se puede observar el autómata equivalente al subconjunto anterior de reglas.

Con base en la síntesis del autómata mostrado en la figura 7, se genera el pseudocódigo correspondiente mostrado en el algoritmo 5.

Algorithm 4 Algoritmo para función "Retroceder"

Require: controlVELOCIDAD, interfazVIRTUAL, matrizDEposiciones
Ensure: posicionCARRO, rotacionCARRO, controlVELOCIDAD, s1, s2 s3, s4, s5, estadoFINAL

```
estadoReversa ← Inicializar
Gi ← [0,1,0,0]
Gd ← [0,-1,0,0]
funcion Leer ← Leer interfazVIRTUAL, matrizDEposiciones
contRECTgiro ← 0
if s4=1 and estadoFINAL ≠ 1 then
    interfazVIRTUAL(traslación) ← [posicionCARRO]
    if s4=1 and s3=0 then
        if rotacionCARRO=0 then
            posicionCARRO(x) ← new_posicionCARRO(x)
            interfazVIRTUAL(traslación) ← [posicionCARRO]
        else if rotacionCARRO < 0 then
            posicionCARRO(x) ← new_posicionCARRO(x)
            posicionCARRO(z) ← new_posicionCARRO(z)
            interfazVIRTUAL(traslación) ← [posicionCARRO]
        end if
        if rotacionCARRO=0 then
            rotacionCARRO=Gi
            rotacionCARRO(angulo) ← contRECTgiro
            posicionCARRO(z) ← new_posicionCARRO(z)
            posicionCARRO(x) ← new_posicionCARRO(x)
            interfazVIRTUAL(rotacion) ← rotacionCARRO
            interfazVIRTUAL(traslación) ← [posicionCARRO]
        else if rotacionCARRO(angulo) > 0 then
            rotacionCARRO ← Gi
            rotacionCARRO(angulo) ← contRECTgiro
            posicionCARRO(z) ← new_posicionCARRO(z)
            posicionCARRO(x) ← new_posicionCARRO(x)
            interfazVIRTUAL(rotacion) ← rotacionCARRO
            interfazVIRTUAL(traslación) ← [posicionCARRO]
        end if
        for p=0:2 do
            realizar interfazVIRTUAL(traslación) ← [posicionCARRO]
            interfazVIRTUAL(rotacion) ← rotacionCARRO
        end for
    end if
    funcion Leer ← Leer interfazVIRTUAL, matrizDEposiciones
end if
```

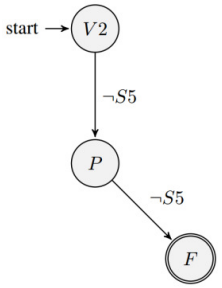

Fig. 7. AFD para la acción "Pare".

Finalmente se diseña un secuenciador global que se encarga de ejecutar cada sub-conjunto de reglas gramaticales de manera independiente para generar así el autómata correspondiente a la configuración que se presente en las entradas del sistema. En la figura 8 se puede observar el autómata de maniobras sintetizado, para el cual:

- I = Estado inicial
- Z = Secuenciador o selección del autómata
- C = Control o ejecución del autómata
- F = Fin del Programa
- In = Entrada
- Or = Entrada diferente, si la entrada cambia regresa a la etapa inicial
- If = Entrada igual, si la entrada es la misma, se debe seguir ejecutando el autómata
- End = Fin del programa

Algorithm 5 Algoritmo para función "Pare"

Require: controlVELOCIDAD, interfazVIRTUAL, matrizDEposiciones
Ensure: posicionCARRO, rotacionCARRO, controlVELOCIDAD, s1, s2 s3, s4, s5, estadoFINAL

```
funcion Leer ← Leer interfazVIRTUAL, matrizDEposiciones
if s5=1 or estadoFINAL=1 then
    interfazVIRTUAL(traslación) ← [posicionCARRO]
end if
```

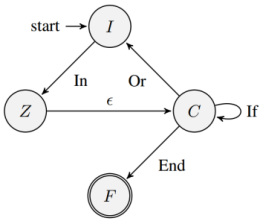

Fig. 8. Autómata de maniobras.

Con base en la síntesis del autómata mostrado en la figura 8, se genera el pseudocódigo correspondiente que se muestra en el algoritmo 6.

Algorithm 6 Algoritmo para función "Secuenciador global"

Require: pista_virtual

```
interfazVirtual ← pista_virtual
posicionOBSTACULO ← interfazVIRTUAL(traslación)
posicionCARRO ← interfazVIRTUAL(traslación)
funcion Leer ← Leer interfazVIRTUAL, matrizDEposiciones
controlVELOCIDAD ← Actualizar
contador ← Actualizar
finalizador ← Actualizar
velocidad1 ← Actualizar
velocidad2 ← Actualizar
Gi ← [0,1,0,0.1]
Gd ← [0,1,0,-0.1]
while finalizador≠1 and contador<=50 do
    while s1, s2, s5 = 0 and estadoFINAL ≠ 1 or s3=1 do
        funcion Avanzar
    end while
    while s1 = 1 and s2 = 0 and estadoFINAL ≠ 1 do
        funcion giroIZQUIERDA
        controlVELOCIDAD ← 0
    end while
    while s1=0 and s2=1 and estadoFINAL ≠ 1 do
        funcion giroDERECHA
        controlVELOCIDAD ← 0
    end while
    while s4=1 and s3=0 and estadoFINAL ≠ 1 do
        funcion Retroceder
        controlVELOCIDAD ← 0
    end while
    while s5=1 or estadoFINAL=1 do
        funcion Parar
        finalizador ← 1
    end while
    contador ← contador + 1
end while
interfazVIRTUAL ← enviar datos interfazVIRTUAL
```

Autómata completo: Otra opción de síntesis para solucionar el problema propuesto consiste en diseñar un único autómata que contenga todas las posibles opciones. Para este

caso, uno de los posibles autómatas no deterministas resultante se muestra en la figura 9.

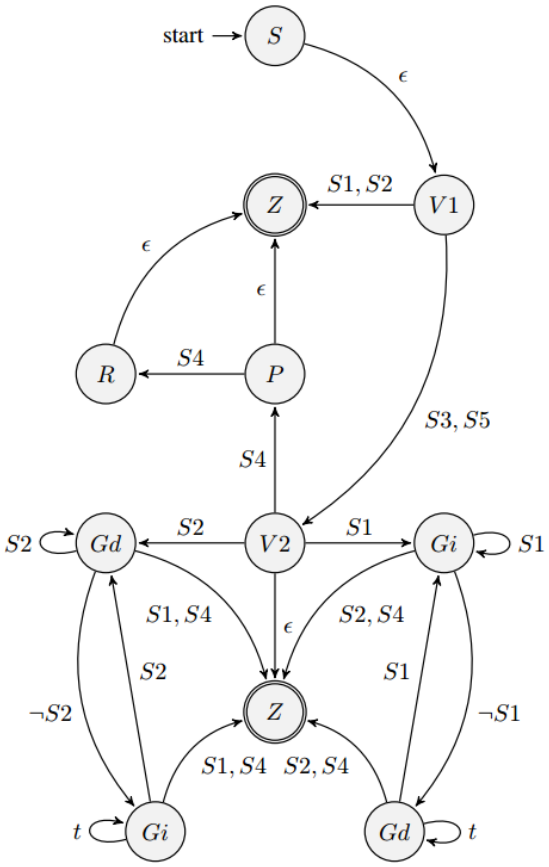

Fig. 9. Síntesis de autómata completo.

2) *Simulaciones y pruebas:* Para validar la metodología propuesta, se ha realizado una serie de simulaciones en el software MATLAB con el Toolbox integrado de realidad virtual V-Realm Builder. En primer lugar, se elaboró la pista en la cual se hicieron las pruebas. En la figura 10 se puede observar que la pista cuenta con una serie de obstáculos que pueden ser ubicados en diferentes posiciones a decisión del programador.

Fig. 10. Pista desarrollada con el Toolbox de Matlab V-Realm Builder.

La información de la ubicación del vehículo en la pista y la información de los sensores del mismo se extrae como una matriz de ceros y unos, de esta manera, se compara constantemente la información de los sensores con la información de los obstáculos en la pista, para saber si el vehículo detecta un obstáculo. Se diseñaron 3 pistas diferentes con obstáculos de diferentes tamaños y en diferentes ubicaciones y se hicieron variaciones en la posición inicial del vehículo en repetidas ocasiones, alcanzando la meta en todos los casos. Además, se cambió la tarea global por una tarea de cobertura, para lo cual, el tiempo de desarrollo y ejecución de la metodología propuesta fue menor al de la metodología tradicional.

III. CONCLUSIONES

En este trabajo se describe un método que permite sintetizar un autómata para que un vehículo autónomo cumpla con un comportamiento especificado por el usuario, expresado con la particularidad de incluir comportamientos reactivos en los cuales el vehículo autónomo depende no solo de las especificaciones de la tarea global dada por el usuario, sino también de la información que captura del entorno a través de sus sensores durante el tiempo de ejecución, mostrando que bajo estos criterios se puede mejorar el desempeño computacional y además se mitiga el problema de explosión combinacional de estados que se presenta al tratar de sintetizar autómatas no deterministas como el obtenido en la figura 9. En comparación con el método de síntesis tradicional en el que se diseña un único autómata partiendo de todo el conjunto que encierra la gramática regular, la metodología propuesta permite flexibilidad en cuanto al cambio de la tarea global, pues en el método tradicional sería necesario volver a sintetizar un autómata completamente diferente y en la metodología propuesta, solo es necesario cambiar la tarea en el secuenciador global. Cuando las opciones de movimiento del vehículo crecen, el proceso de síntesis de este tipo de autómatas implica un costo computacional más alto ya que en cada iteración se debe evaluar todos los posibles estados del sistema, como es el caso de los vehículos omnidireccionales o de los drones que tienen la opción de moverse en cualquier dirección en 3 dimensiones. La metodología propuesta resuelve el problema de síntesis de autómatas para el control de bajo nivel encargado de los movimientos primitivos de robots autónomos, permitiendo así avanzar al campo de la síntesis automática de autómatas para el control en alto nivel de los vehículos en dónde la planificación de movimientos o el cambio de tareas durante la ejecución es una constante.

AGRADECIMIENTOS

Se agradece a la Universidad Tecnológica de Pereira por su apoyo a través de la convocatoria interna para financiación de la Vicerrectoría de Investigaciones, Innovación y Extensión, así como a la Maestría en Ingeniería Eléctrica y al Grupo de Investigación en Gestión de Sistemas Eléctricos, Electrónicos y Automáticos.

REFERENCIAS

[1]. G. B. Enguix and M. D. J. López, "Simposio: Nuevas aplicaciones de la teoría de lenguajes formales a la lingüística."

[2]. H. Kress-Gazit, "Robot challenges: Toward development of verification and synthesis techniques [from the guest editors]," IEEE Robotics & Automation Magazine, vol. 18, no. 3, pp. 22–23, 2011.

[3]. T. Wongpiromsarn, U. Topcu, and R. M. Murray, "Receding horizon temporal logic planning," IEEE Transactions on Automatic Control, vol. 57, no. 11, pp. 2817–2830, 2012.

[4]. H. M. Choset, Principles of robot motion: theory, algorithms, and implementation. MIT press, 2005.

[5]. S. M. LaValle, Planning algorithms. Cambridge university press, 2006.

[6]. H. Choset and J. Burdick, "Sensor-based exploration: The hierarchical generalized voronoi graph," The International Journal of Robotics Research, vol. 19, no. 2, pp. 96–125, 2000.

[7]. R. M. Jensen and M. M. Veloso, "Obdd-based universal planning for synchronized agents in non-deterministic domains," Journal of Artificial Intelligence Research, vol. 13, pp. 189–226, 2000.

[8]. P. Bertoli, A. Cimatti, M. Pistore, M. Roveri, and P. Traverso, "Mbp: a model based planner," in Proc. of the IJCAI'01 Workshop on Planning under Uncertainty and Incomplete Information, 2001.

[9]. H. Kress-Gazit, G. E. Fainekos, and G. J. Pappas, "Temporal-logic-based reactive mission and motion planning," IEEE transactions on robotics, vol. 25, no. 6, pp. 1370–1381, 2009.

[10]. Belta, A. Bicchi, M. Egerstedt, E. Frazzoli, E. Klavins, and G. J. Pappas, "Symbolic planning and control of robot motion [grand challenges of robotics]," IEEE Robotics & Automation Magazine, vol. 14, no. 1, pp. 61–70, 2007.

[11]. M. Kloetzer and C. Belta, "A fully automated framework for control of linear systems from temporal logic specifications," IEEE Transactions on Automatic Control, vol. 53, no. 1, pp. 287–297, 2008.

[12]. J. A. DeCastro, V. Raman, and H. Kress-Gazit, "Dynamics-driven adaptive abstraction for reactive high-level mission and motion planning," in Robotics and Automation (ICRA), 2015 IEEE International Conference on. IEEE, 2015, pp. 369–376.

[13]. S. R. Lindemann and S. M. LaValle, "Computing smooth feedback plans over cylindrical algebraic decompositions." in Robotics: Science and Systems, 2006.

[14]. A. Boteanu, T. Howard, J. Arkin, and H. Kress-Gazit, "A model for verifiable grounding and execution of complex natural language instructions," in Intelligent Robots and Systems (IROS), 2016 IEEE/RSJ International Conference on. IEEE, 2016, pp. 2649–2654.

[15]. H. Kress-Gazit, T. Wongpiromsarn, and U. Topcu, "Mitigating the state explosion problem of temporal logic synthesis," IEEE Robotics & Automation Magazine, 2011.

[16]. V. Raman and H. Kress-Gazit, "Explaining impossible high-level robot behaviors," IEEE Transactions on Robotics, vol. 29, no. 1, pp. 94–104, 2013.

[17]. T. Wongpiromsarn, U. Topcu, and R. M. Murray, "Receding horizon control for temporal logic specifications," in Proceedings of the 13th ACM international conference on Hybrid systems: computation and control. ACM, 2010, pp. 101–110.

[18]. R. F. Brena, Autómatas y lenguajes. Tecnológico de Monterrey, 2003.

[19]. J. E. Hopcroft, R. Motwani, and J. D. Ullman, "Teoría de autómatas, lenguajes y computaci ´ on," Pearson educación, 2007.

[20]. S. Balari Ravera, "Teoría de lenguajes formales," 2014.

[21]. N. Chomsky, "Aspects of a theory of syntax," Cambridge, Mass.: MI'I'. I'rw Language Journal, vol. 53, pp. 334–341, 1965.

[22]. A. A. O. Gutiérrez, G. A. H. Londoño, and M. H. Londoño, Fundamentos teóricos para los autómatas industriales. Universidad Tecnológica de Pereira, 2010.

[23]. M. A. Moreno, J. S. Rodríguez, and M. M. Orga, Teoría de lenguajes, gramáticas y autómatas, 1987.

[24]. C. M. Vide, "Gramáticas formales (y similares) para lingüistas," in Lenguajes naturales y lenguajes formales: actas del X congreso de lenguajes naturales y lenguajes formales:(Sevilla, 26-30 de septiembre de 1994). Promociones y Publicaciones Universitarias, PPU, 1994, pp. 71–92.

[25]. W. Reviewer-Gasarch, "Book review: Finite automata, formal logic, and circuit complexity. by howard straubing.(birkhauser. 1994. xii+ 226pp. isbn 0-8176-3719-2.)," ACM SIGACT News, vol. 25, no. 3, pp. 28–32, 1994.

[26]. T. Wongpiromsarn and R. M. Murray, "Distributed mission and contingency management for the darpa urban challenge," in International Workshop on Intelligent Vehicle Control Systems (IVCS), 2008.

[27]. E. A. Emerson, "Temporal and modal logic." Handbook of Theoretical Computer Science, Volume B: Formal Models and Sematics (B), vol. 995, no.

1072, p. 5, 1990.

[28]. G. E. Fainekos, H. Kress-Gazit, and G. J. Pappas, “Temporal logic motion planning for mobile robots,” in Robotics and Automation, 2005. ICRA 2005. Proceedings of the 2005 IEEE International Conference on. IEEE, 2005, pp. 2020–2025.

[29]. Z. Manna and A. Pnueli, The temporal logic of reactive and concurrent systems: Specification. Springer Science & Business Media, 2012.

[30]. N. Piterman, A. Pnueli, and Y. Sa’ar, “Synthesis of reactive (1) designs,” in International Workshop on Verification, Model Checking, and Abstract Interpretation. Springer, 2006, pp. 364–380.

[31]. A. Pnueli and R. Rosner, “On the synthesis of a reactive module,” in Proceedings of the 16th ACM SIGPLAN-SIGACT symposium on Principles of programming languages. ACM, 1989, pp. 179–190.