MODELADO Y VERIFICACIÓN MEDIANTE LÓGICA LINEAL TEMPORAL DE UN GRUPO DE DOS ASCENSORES CON SISTEMA DE CONTROL DE DESTINO

Manuel Toscano-Moreno, Alberto Arregui, Anthony Mandow y Alfonso García-Cerezo
Universidad de Málaga, Andalucía Tech, Departamento de Ingeniería de Sistemas y Automática
m.toscano@uma.es

Resumen

Los grupos de ascensores con sistema de preselección de destino persiguen la reducción del tiempo de espera en edificios de mediana y gran altura como hoteles o bloques de oficinas. En este tipo de sistemas, los pasajeros se dirigen al ascensor de acuerdo con el destino que hayan indicado en un panel exterior. El presente trabajo aborda la verificación del control de ascensores con preselección de destino mediante la aplicación de una herramienta software basada en lógica temporal lineal (LTL). En particular, se ha definido tanto el modelo del sistema como la especificaciones LTL mediante la herramienta de código abierto Spin. *Como caso de estudio, se ha implementado el modelo de un grupo de dos ascensores en un edificio de cuatro plantas con paneles de preselección de destino en cada planta. El artículo ofrece resultados preliminares de simulaciones y verificaciones para un conjunto de fórmulas LTL definidas específicamente para este sistema.*

Palabras clave: lógica temporal lineal, verificación, control de grupos de ascensores, sistemas de eventos discretos, sistemas multi-agente

1. INTRODUCCIÓN

Los grupos de ascensores con sistema de control o preselección de destino (*destination dispatch* en la literatura en inglés) persiguen la reducción del tiempo de espera y desplazamiento respecto a los sistemas de control convencionales basados en estrategias colectivas [8]. Los sistemas con control de destino disponen de un panel (dispositivo táctil o botonera) exterior a la cabina, de forma que los pasajeros deben seleccionar la planta destino al solicitar el ascensor (ver figura 1) y son dirigidos al ascensor más apropiado. El interés por el control con preselección ha favorecido la aparición de sistemas comerciales de distintos fabricantes internacionales (e.g., [10][7]).

En general, el control de grupos de ascensores constituye un problema complejo con criterios de optimización conflictivos y sujeto a incertidumbres, por lo que resulta imprescindible un análisis mediante simulación de modelos previo a la implementación real [3]. La lógica temporal lineal (LTL) es un tipo de lógica que se formula mediante un conjunto de operadores Booleanos y operadores modales temporales [1]. La formulación LTL permite especificar tareas de alto nivel para el control de movimiento de sistemas distribuidos, como por ejemplo un conjunto de robots móviles [6]. Asimismo, la LTL permite la codificación de restricciones y regulaciones de seguridad, así como la especificación de relaciones temporales para evaluar una secuencia completa de acciones en un ascensor [5]. La verificación del control de un grupo de dos ascensores con estrategia de control colectivo fue abordado en [2] usando Spin. En este sentido, distintas herramientas como Spin, UPPAAL, y NuSMV, permiten la verificación de sistemas en tiempo real a partir de lógicas temporales [11]. Sin embargo, en el estudio de la bibliografía, no se ha encontrado ningún trabajo que aborde la verificación LTL de sistemas de ascensores con preselección de destino.

(a) (b)

Figura 1: Ejemplo de ascensores con preselección de destino. a) Panel de preselección en el vestíbulo. b) La cabina no dispone de panel de control; los indicadores muestran la planta actual y la de destino.

Este trabajo ofrece resultados preliminares respecto a la verificación de modelos (*model checking*) mediante LTL para un grupo de ascensores con preselección de destino, los cuales son tratados de forma descentralizada. En particular, se

evalúa la herramienta de verificación SPIN [4] para la validación y detección de errores mediante especificaciones LTL. El artículo ofrece ejemplos de verificación de expresiones LTL.

El resto del artículo se organiza de la siguiente manera. A continuación, la sección 2 revisa brevemente la verificación de modelos y la lógica temporal lineal (LTL). La sección 3 define el caso de aplicación, describiendo el sistema de control de destino en un grupo de ascensores. La sección 4 muestra la definición de propiedades LTL para el sistema y resultados de la verificación. Finalmente, se presentan las conclusiones e ideas para desarrollos futuros.

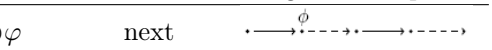
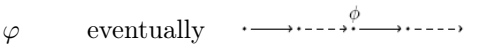
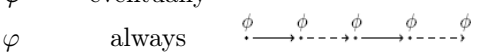
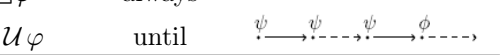
2. VERIFICACIÓN DE ESPECIFICACIONES LTL

La verificación de modelos mediante métodos formales consiste en comprobar, exhaustiva y automáticamente, si un modelo dado satisface un conjunto de especificaciones expresadas mediante fórmulas lógicas. Para resolver este problema algorítmicamente, el modelo del sistema y las especificaciones tienen que estar expresados en un lenguaje preciso, basado en un sistema formal. De esta manera, la herramienta de código abierto SPIN [9] emplea el lenguaje de programación de alto nivel PROMELA para la construcción del modelo y la verificación de especificaciones formuladas en LTL [4].

La verificación de propiedades temporales emplea principalmente dos tipos de lógica modal temporal [1]: la CTL (*Computation Tree Logic)* y la LTL. Mientras la CTL se basa en un modelo temporal con una estructura de árbol, en la que el futuro no está determinado, la LTL no tiene en cuenta esa estructura de árbol y, por tanto, considera una evolución lineal del tiempo. Así, en la verificación formal se comprueba que no exista un camino en el que se comprometa la seguridad del sistema o bien que exista un camino que cumpla con una especificación dada. Existen variaciones de estos tipos de lógicas temporales, como la TCTL (*Timed Computational Tree Logic*) que permite verificar sistemas de tiempo real, o la PLTL (*Probabilistic Computational Tree Logic*) y la PCTL (*Probabilistic Linear Temporal Logic*) que permite determinar la probabilidad de error en el cumplimiento de ciertas propiedades.

La LTL se construye con operadores lógicos como $\neg$ (*not*), $\wedge$ (*and*), $\vee$ (*or*) o $\rightarrow$ (*implies*) y operadores modales temporales. En la tabla 1 se muestran algunos de los operadores modales temporales más utilizados en la LTL. En dicha tabla, el diagrama temporal representa los instantes de tiempo en los

Tabla 1: Operadores modales temporales comúnmente utilizados en LTL. La flecha indica el salto al siguiente (trazo continuo) o a un número indeterminado (trazo discontinuo) de estados.

Expresión	Nombre	Diagrama temporal
$\bigcirc\varphi$	next	
$\Diamond\varphi$	eventually	
$\Box\varphi$	always	
$\psi\,\mathcal{U}\,\varphi$	until	

que las proposiciones lógicas φ o ψ deben cumplirse para que la expresión LTL se satisfaga.

Nótese que la lógica *temporal* especifica una secuencia temporal de eventos, es decir, un evento sucede antes o después que otro, pero no tras un periodo concreto de tiempo. Combinando operadores lógicos y temporales se pueden construir fórmulas como la siguiente:

$$\Box(\psi \vee \varphi) \rightarrow \bigcirc\omega\,, \tag{1}$$

la cual expresa la siguiente propiedad: «siempre que se cumpla ψ o φ implicará que ω se cumpla en el próximo estado».

Mediante fórmulas LTL se pueden expresar diferentes propiedades que deben satisfacer los sistemas: de seguridad ("*safety*"), de vivacidad ("*liveness*") o de equidad ("*fairness*"). Las propiedades de equidad restringen el comportamiento del sistema, obligando a que los procesos se ejecuten de manera razonadamente frecuente. Así, dependiendo del tipo de sistema se deberá imponer cierto nivel de equidad:

- Equidad débil ("*weak fairness*"): Si una proposición p se cumple eventualmente de manera continua, se debe cumplir la proposición q en algún momento. Esta propiedad puede expresarse mediante la siguiente fórmula LTL:

$$\Diamond\Box p \rightarrow \Box\Diamond q\,. \tag{2}$$

- Equidad fuerte ("*strong fairness*"): Si una proposición se cumple infinitamente a menudo (*infinitely often*), se debe cumplir la proposición q en algún momento. Esta propiedad puede expresarse mediante la siguiente fórmula LTL:

$$\Box\Diamond p \rightarrow \Box\Diamond q\,. \tag{3}$$

3. PRESELECCIÓN DE DESTINO

En esta sección se describe la estructura y el comportamiento de un grupo de ascensores con política de peticiones externas.

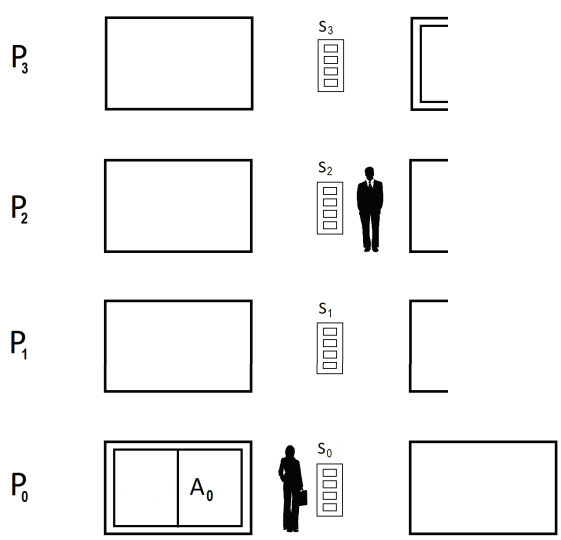


Figura 2: Esquema del grupo de ascensores.

3.1. Grupo de ascensores con control de destino

La figura 2 muestra el esquema del sistema considerado en este trabajo, el cual consta de un grupo de dos ascensores (A_0 y A_1) y cuatro plantas (P_0, P_1, P_2 y P_3). En cada planta existe una panel de control (S_0, S_1, S_2 y S_3) para seleccionar la planta de destino y mostrar el ascensor asignado. Cuando se efectúa la petición, el ascensor asignado acudirá a la planta en la que se encuentra el pasajero para, posteriormente, dirigirse a la planta destino seleccionada.

3.2. Algoritmo de control de destino

El algoritmo básico para el control del grupo de ascensores es el siguiente. Los ascensores esperan detenidos mientras no exista una petición externa por parte de un pasajero. Cuando un pasajero está en una planta P_i y selecciona en el panel de control S_i la planta destino P_j a la que desea ir, uno de los dos ascensores A_k atenderá la petición, pasando por las diferentes plantas hasta llegar a la planta P_i en la que se encuentra el pasajero. Con el ascensor detenido y las puertas abiertas, el pasajero accede a la cabina para ser conducido a la planta destino seleccionada. El ascensor efectuará paradas en los destinos de los distintos pasajeros a bordo de la cabina. Podrán añadirse paradas si durante el trayecto existe una petición de pasajero en la dirección de desplazamiento. Una vez alcanzada una planta destino, el ascensor se detiene y abre sus puertas. Cuando el último pasajero abandona la cabina y el ascensor cierra sus puertas, éste queda disponible para atender peticiones de pasajero que no hayan podido ser atendidas.

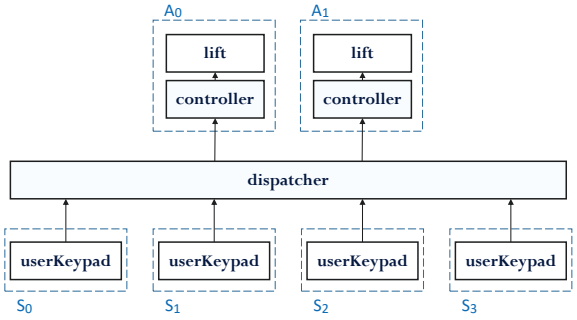


Figura 3: Diagrama de bloques correspondiente al modelado del sistema del grupo de dos ascensores con preselección de destinos.

3.3. Modelado en Promela

La implementación para un grupo de ascensores con preselección de destino constituye una ampliación de los trabajos de [5], donde se modela un único ascensor, y [2] que presenta un grupo de dos ascensores convencionales sin preselección de destino. La aplicación implementada en PROMELA consta de un proceso principal denominado `init` que invoca al resto de procesos. En el lenguaje PROMELA, el modelo se describe mediante un conjunto de procesos paralelos, cuyas instancias se ejecutan asíncronamente. La figura 3 muestra las instancias de todos los procesos del modelo. Cada uno de los procesos da lugar a una máquina de estados, como se ilustra en la figura 4. De esta manera, cada instancia se trata como un autómata independiente ejecutado de manera concurrente.

En concreto, para modelar el sistema de ascensores, se han definido cuatro tipos de procesos (`proctypes`):

- `lift`: modela el cambio de planta de un ascensor y la apertura de puertas. Se instancian dos procesos de este tipo, uno para cada ascensor.
- `userKeypad`: modela el panel de control que utilizará el pasajero para seleccionar como destino una de las cuatro plantas existentes. Se instancian cuatro procesos de este tipo, uno para cada planta.
- `controller`: modela el controlador de un ascensor, considerando las asignaciones establecidas desde el proceso `dispatcher` e indicando a `lift` cambios de planta y apertura o cierre de puertas. Se instancian dos procesos de este tipo, uno para cada ascensor. Cada controlador se considera embebido en cada ascensor, definiendo una arquitectura distribuida.
- `dispatcher`: modela la asignación de as-

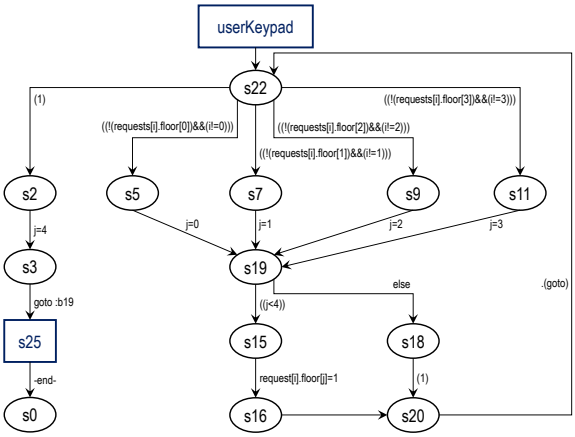


Figura 4: Diagrama de estados resultante de la implementación del proceso `userKeypad` en PROMELA.

censores, considerando las peticiones de `userKeypad` e indicando al correspondiente `controller` el servicio a realizar. Se instancia un único proceso de este tipo. En este proceso, a diferencia de [2], se asigna el ascensor considerando la planta desde donde el pasajero solicita el servicio y la planta destino preseleccionada, contribuyendo así a reducir los tiempos de espera. Para la asignación del ascensor se adopta un sistema de puntos que prioriza aquel ascensor que se encuentre más próximo y comparta planta destino, siempre que las paradas implicadas se encuentren en la misma dirección del movimiento.

4. RESULTADOS DE LA VERIFICACIÓN

4.1. Simulación

En esta sección se describe el ejemplo de simulación para el caso en el que, encontrándose ambos ascensores en la planta P_0 desde el panel S_1 se solicite como destino la planta P_3.

Se ha simulado que «siempre que desde el panel S_1 se solicite como destino la planta P_3, alguno de los dos ascensores debe alcanzar en algún momento dicha planta destino, deteniéndose y abriendo sus puertas». Para la simulación, se construye una fórmula LTL que exprese la negación de dicho comportamiento y así SPIN, en su proceso de verificación, trata de obtener un contraejemplo o traza de ejecución que no la satisfaga y que, por tanto, reproduzca el comportamiento inicialmente deseado. La fórmula LTL que expresa este comportamiento es del tipo:

$$\neg \, \Box(p \rightarrow \Diamond q)\,, \tag{4}$$

cuya implementación en PROMELA corresponde a:

```
@USERKEYPAD   # User request on floor 1 with destination floor 3
@DISPATCHER   # Selected lift 0 for request on floor 1 to floor 3
@LIFT 0       # Lift moving upward from floor 0 to floor 1
@LIFT 0       # Lift stopped in floor 1
@LIFT 0       # Lift doors opening in floor 1
@LIFT 0       # Group of users accessing the lift in floor 1
@LIFT 0       # Lift doors closing in floor 1
@LIFT 0       # Lift moving upward from floor 1 to floor 2
@LIFT 0       # Lift moving upward from floor 2 to floor 3
@LIFT 0       # Lift stopped in floor 3
@LIFT 0       # Lift doors opening in floor 3
@LIFT 0       # Group of users leaving the lift in floor 3
@LIFT 0       # Lift doors closing in floor 3
-------------
```

Figura 5: Resultado de la simulación en SPIN para el caso en el que, encontrándose ambos ascensores en la planta P_0, desde el panel S_1 se solicite como destino la planta P_3. Se indican los estados de los distintos procesos, mostrando la evolución secuencial del sistema (captura de pantalla).

```
#define p ( requests[1].floor[3]==1)
#define q ( ( lift_pos[0]==3   &&
              lift_stopped[0]  &&
              door_opened[0] ) ||
            ( lift_pos[1]==3   &&
              lift_stopped[1]  &&
              door_opened[1] ) )

ltl TP { ![]( p -> <>q ) }
```

Al verificar esta propiedad LTL, SPIN genera un contraejemplo que, tras su simulación guiada (*guided with trail*), muestran los estados por los que transita el sistema hasta alcanzar el estado final del comportamiento deseado (ver figura 5).

4.2. Verificación

Con el objetivo de verificar el correcto comportamiento del grupo de ascensores se han analizado diferentes propiedades expresadas mediante fórmulas LTL. Para este caso de estudio se dividen en dos grupos según su naturaleza: propiedades de seguridad y vivacidad.

Las propiedades de seguridad expresan comportamientos que nunca deben ocurrir. Las verificaciones de este modelo para este grupo de propiedades han requerido un espacio en memoria de 1300 MB (usando la representación compacta *bitstate* de SPIN), implicando 70 millones de transiciones exploradas con una profundidad de búsqueda de 8 millones de pasos, para lo cual se ha requerido un tiempo de cómputo de 3 minutos. Las propiedades de seguridad verificadas han sido las siguientes, para las que se indica su formulación LTL y la correspondiente implementación en PROMELA:

- `S1`: «Cuando no exista ninguna petición de pasajero, ambos ascensores deben permane-

cer detenidos».

$$\Box(p \rightarrow q) \tag{5}$$

```
#define p ( assist[0]==0       &&
            assist[1]==0 )
#define q ( lift_stopped[0]==0 &&
            lift_stopped[1]==0 )

ltl S1 {[] p -> q }
```

- S2: «Ningún ascensor estará en movimiento con sus puertas abiertas».

$$\Box\neg p \tag{6}$$

```
#define p (!lift_stopped[0] &&
            door_opened[0]) ||
          (!lift_stopped[1] &&
            door_opened[1])

ltl S2 {[]!p}
```

- S3: «El ascensor solo se parará antes de llegar a su destino si hay una petición con el mismo destino que la que se está satisfaciendo».

$$\Box\neg p \tag{7}$$

```
#define p  (lift_stopped[0] &&
            door_opened[0]  &&
           ( lift_pos[0]==1 ||
            lift_pos[0]==2 ) &&
     user_destination[0]==3 &&
     !dest_request[2].re[3] &&
     !dest_request[1].re[3] )

ltl S3 {[]!p}
```

Las propiedades de vivacidad expresan comportamientos que deben satisfacerse para asegurar la funcionalidad demandada al sistema. La verificación de estas propiedades requiere menos profundidad de búsqueda, pudiendo ejecutarse de forma exhaustiva.

- V1: «Todas las peticiones deben satisfacerse en algún momento». Se ha definido una propiedad en la cual se modela una petición de un pasajero que se encuentra en la segunda planta.

$$\Box\ p \rightarrow \Diamond q \tag{8}$$

```
#define p level_request[2]
#define q ( door_opened[0] &&
            lift_stopped[0] &&
             lift_pos[0]==2 &&
     user_location[0]==2 ) ||
         (  door_opened[1] &&
            lift_stopped[1] &&
             lift_pos[1]==2 &&
     user_location[1]==2 )

ltl V1 { []( p -> <> q ) }
```

Esta propiedad supone un ejemplo de verificación con detección de errores. En concreto, se identifican errores tanto con *weak fairness* como sin tener en cuenta ninguna restricción de equidad. La herramienta proporciona un contraejemplo en el que un pasajero realiza una petición desde la segunda planta y se queda esperando para siempre.

La verificación ocupa 3187 MB, abarca 15 millones de transiciones con una profundidad de búsqueda de 170000 pasos.

- V2: «Todos los pasajeros deben llegar a su destino en algún momento».

$$\Box\ p \rightarrow \Diamond q \tag{9}$$

```
#define p user_location[0]==2 &&
              lift_pos[0]==2 &&
       user_destination[0]==0
#define q  elevator_opened[0] &&
             lift_stopped[0] &&
              lift_pos[0]==0

ltl V2 { []( p -> <>q ) }
```

Esta especificación tampoco es inicialmente verificada en la implementación, para lo que se obtiene el mismo contraejemplo que en la especificación V1. En este contraejemplo, se producen peticiones en múltiples plantas y el proceso `dispatcher` asigna siempre una planta distinta a la segunda. Para evitar este comportamiento, se ha modificado el proceso `dispatcher` para introducir prioridad dinámica a las peticiones de pasajeros, de modo que el valor asociado al sistema de puntos para la asignación del ascensor, aumente en función del tiempo de espera a ser atendidas. Se han obtenido resultados equivalentes verificando la propiedad para otras plantas y ascensores.

5. CONCLUSIONES

Este trabajo ha presentado resultados preliminares de la aplicación de Lógica Temporal Lineal (LTL) para la verificación de un controlador para un grupo de dos ascensores con preselección de destino. En particular, se ha utilizado el software SPIN para modelar el grupo de ascensores y una estrategia de control definida para el caso de un panel de selección de destinos en cada planta,

donde cada instancia se trata como un autómata independiente ejecutado de manera concurrente. Con esta metodología es posible identificar fallos en el control de acuerdo con las especificaciones formales de la LTL.

La validación LTL resultará fundamental para la síntesis de controladores para grupos de ascensores mediante técnicas de optimización e inteligencia artificial. Asimismo, estos resultados para un sistema distribuido servirán de base para desarrollos futuros en en otros sistemas distribuidos, como los equipos multi-robot, y en especial para aplicaciones de búsqueda y rescate de víctimas en situaciones de catástrofe.

Agradecimientos

Este trabajo ha recibido financiación del proyecto nacional RTI2018-093421-B-I00, la Universidad de Málaga (Andalucía Tech) y la ayuda BES-2016-077022 del Fondo Social Europeo FSE.

English summary

MODELING AND LINEAR TEMPORAL LOGIC VERIFICATION OF A GROUP OF TWO ELEVATORS WITH DESTINATION CONTROL SYSTEM

Abstract

Destination dispatch systems for group control of elevators aim to reduce the waiting and displacement time with respect to conventional control systems in mid and high rise buildings such as hotels or office blocks. In these systems, passengers are directed to cars according to their destinations. The present work addresses verification of the destination control system for an elevator group by using a software tool based on linear temporal logic (LTL). In particular, both the system model and the LTL specifications have been defined using the open source tool Spin. *As a case study, the model of a group of two elevators has been implemented in a four-story building with destination preselection panels in each plant. The article offers preliminary results of simulations and verifications for a set of LTL formulas defined specifically for this system.*

Keywords: linear temporal logic, verification, elevator group control, discrete-event systems, multi-agent systems

Referencias

[1] Belta, C., Yordanov, B. and Gol, E. A.: 2017, *Formal Methods for Discrete-Time Dynamical Systems*, Springer.

[2] Duval, G. and Cattel, T.: 1997, From architecture down to implementation of safe process control applications - the lift case study, *Proceedings of the Hawaii International Conference on System Sciences*, Vol. 1, pp. 24–33.

[3] Fernandez, J. R. and Cortes, P.: 2015, A survey of elevator group control systems for vertical transportation: A look at recent literature, *IEEE Control Systems* **35**(4), 38–55.

[4] Holzmann, G. J.: 1997, The model checker Spin, *IEEE Transactions on Software Engineering* **23**(5), 279–295.

[5] Merz, S. and Navet, N.: 2008, An introduction to model checking, *Modeling and Verification of Real-Time Systems: Formalisms and Software Tools*, chapter 3, pp. 77–110.

[6] Parrilla, L., Mahulea, C. and Kloetzer, M.: 2017, RMTool: Recent enhancements, *IFAC-PapersOnLine* **50**(1), 5824–5830.

[7] Schindler: 2019 (Consultado en junio de 2019), Tecnología Schindler PORT.
URL: *www.schindler.com/es/internet/es/soluciones-de-movilidad/productos/tecnologia/schindler-port.html*

[8] Sorsa, J., Hakonen, H. and Siikonen, M. L.: 2006, Elevator selection with destination control system, *Elevator World* **54**(1), 148–155.

[9] Spin: 2019 (Consultado en junio de 2019), Verifying multi-threaded software with Spin.
URL: *www.spinroot.com*

[10] Thyssenkrupp: 2017 (Consultado en junio de 2019), AGILE destination control - factssheet.
URL: *www.thyssenkruppelevator.com/elevator-products/agile*

[11] Zhongsheng, Q., Xin, L. and Xiaojin, W.: 2018, Modeling distributed real-time elevator system by three model checkers, *International Journal of Online Engineering* **14**(4), 94–110.