LearnTA: Generación automática de autómatas temporizados mediante el aprendizaje de trazas

Rafael López Gómez[1], María del Mar Gallardo[1], and Laura Panizo[1]

ITIS Software, Andalucía Tech, España
Universidad de Málaga, España
{RafaelLopez, mdgallardo, laurapanizo}@uma.es

Resumen El rápido avance de tecnologías, como la Inteligencia Artificial, está permitiendo el desarrollo de sistemas software muy sofisticados. Para la detección temprana de errores en estos sistemas es usual la construcción de modelos abstractos sobre los que se pueda razonar. Sin embargo, esta tarea de modelado se complica cuando lo único que puede observarse de los sistemas es su interacción con el entorno.
En este trabajo, se presenta *LearnTA*, una herramienta de aprendizaje para la generación automática de modelos de sistemas (Systems Under Learning/SULs) a partir de la observación de su ejecución. Concretamente, la herramienta tiene como objetivo *aprender* sistemas reactivos cuya evolución puede depender del tiempo.
LearnTA emplea un algoritmo de *Automata Learning* con aprendizaje pasivo. *LearnTA* utiliza el comportamiento observado del SUL que se quiere aprender para construir un modelo formal. El comportamiento observado del SUL está constituido por *secuencias finitas* (trazas) de observaciones, cada una de las cuales tiene a lo sumo un *evento* de interacción del sistema con su entorno y su *estado visible* en un *instante de tiempo*. El modelo formal construido por *LearnTA* es un tipo especial de autómata de tiempo real determinista.
Para evaluar *LearnTA*, se han realizado una serie de experimentos en los que los SULs son sistemas sintéticos de diferente tamaño. Asimismo, también se ha realizado una comparación con TAG, otra herramienta perteneciente al estado del arte.

Palabras clave: System Modeling · Passive Automata Learning · System Analysis

1. Introducción

En los últimos años, el desarrollo de distintas tecnologías software ha llevado a construir sistemas cada vez más complejos. La detección de errores en su diseño e implementación no es una tarea trivial sobre todo cuando estos sistemas funcionan como una caja negra, es decir, su implementación es desconocida. Las técnicas de *Automata Learning* (AL) surgen para automatizar la construcción de modelos formales de estos sistemas, que pueden ser reactivos y dependientes del tiempo.

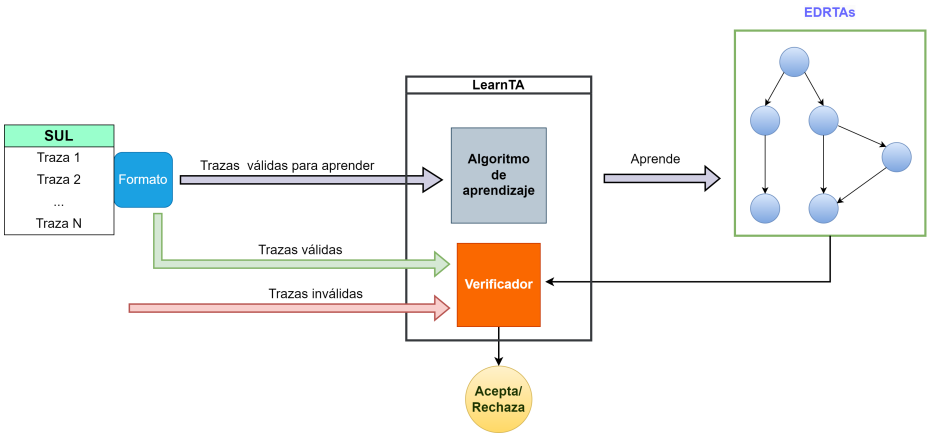

Figura 1: Funcionamiento general de *LearnTA*

Existen dos formas de aplicar las técnicas de AL. Por un lado, los *métodos activos* pueden interactuar con el sistema bajo aprendizaje (System Under Learning/SUL) para determinar si el modelo que se está construyendo es fiel al comportamiento del SUL. Estas técnicas se basan en el algoritmo L^* de Angluin [2]. Por otro lado, los *métodos pasivos* no interaccionan con el SUL sino que construyen el modelo utilizando únicamente observaciones. En este caso, los modelos resultantes producen, en general, un subconjunto del comportamiento del SUL ya que para su construcción solo se ha tenido en cuenta la información observada del SUL que es, normalmente, parcial. Los métodos pasivos se suelen utilizar cuando no es posible interactuar con el SUL.

Uno de los formalismos más utilizados para modelar estos sistemas reactivos y dependientes del tiempo son los *Autómatas Temporizados* (Timed Automata/TA). Los TA incluyen variables de tipo reloj y restricciones sobre estas variables que permiten reflejar la evolución en el tiempo de los SULs. Los Autómatas Deterministas de Tiempo Real (Deterministic Real-Time Automata/DRTA) son una de las subclases de TA más utilizada en los trabajos de AL [8,6,4].

En este trabajo, se presenta la herramienta de aprendizaje *LearnTA*, cuyo núcleo está formado por un algoritmo que usa técnicas de *Passive Automata Learning* (PAL) para modelar el comportamiento de sistemas reactivos. *LearnTA* produce modelos que son una *extensión* de los DRTA (EDRTA). La Figura 1 muestra los principales componentes de *LearnTA* así como el flujo de interacción entre ellos y con el sistema. De izquierda a derecha, aparece el SUL del que se extraen trazas de observaciones cuyo formato debe adaptarse a *LearnTA*. En el centro, se encuentra la herramienta *LearnTA* formada por dos módulos: el algoritmo de aprendizaje y el verificador. El algoritmo de aprendizaje es el componente principal de la herramienta, capaz de generar un EDRTA a partir de secuencias finitas de ejecución, que denominamos *trazas válidas para aprender*. Por otro lado, el verificador se encarga de comprobar que el autómata generado es

capaz de aceptar *trazas válidas* del SUL utilizadas para aprender u otras nuevas. También se puede utilizar para comprobar que rechaza comportamientos que no pertenecen al SUL también denominadas *trazas inválidas*[1].

Las principales aportaciones de *LearnTA* respecto a otras propuestas de la literatura son:

1. *LearnTA* aprende observaciones que, además de marcas de tiempo y eventos, incluyen *atributos* del sistema. Estos atributos representan el estado visible del sistema en un instante de tiempo.
2. El algoritmo de aprendizaje utiliza la noción de *similitud* de estados, que tiene en cuenta los atributos y las restricciones de tiempo. Además, utiliza una técnica de *emparejamiento* (matching) de sub-trazas de observaciones con caminos del autómata para reducir el tamaño del autómata resultante.
3. Los atributos permiten que *LearnTA* aprenda autómatas con dos tipos de aristas: las que representan la ocurrencia de un evento y las que representan únicamente un cambio en los atributos observados. Este último tipo de aristas representan cambios en el sistema debidos a eventos que no se pueden observar en las trazas pero si se observa la respuesta del sistema.

El artículo se organiza de la siguiente manera. La Sección 2 presenta otras propuestas destacadas de *Passive Automata Learning*. En la Sección 3 se describe el algoritmo de aprendizaje, que es el núcleo de la herramienta *LearnTA*. A continuación, la Sección 4 describe los experimentos desarrollados para evaluar *LearnTA*. Estos incluyen una comparación con la herramienta TAG perteneciente al estado del arte. Finalmente, en la Sección 5 se resumen las aportaciones de *LearnTA* y se enumeran los posibles trabajos futuros.

2. Trabajos relacionados

Actualmente, la mayoría de trabajos que siguen las técnicas de *Passive Automata Learning* se enfocan en la inclusión del tiempo en los modelos. En esta línea, podemos encontrar las herramientas TAG [4], RTI+ [7] y TkT [5]. Todas ellas construyen diferentes subclases de TA a partir de un conjunto de trazas observadas del SUL. Por ejemplo, TkT aprende un tipo de TA que tiene relojes locales y un reloj global que nunca se reinicia. Por otro lado, RTI+ y TAG aprenden Probabilistic DRTA, que son TA deterministas con un único reloj que se reinicia en cada arista. Además, las aristas incluyen la probabilidad, en función de las trazas observadas, de que tenga lugar la transición correspondiente. En todos estos trabajos, las aristas de los autómatas siempre están etiquetadas con eventos. En nuestro caso, se permite que una arista no tenga evento debido al uso de los atributos.

TAG, RTI+ y TkT presentan dos fases de aprendizaje similares. En la primera, construyen el llamado "Prefix Tree Automaton" (PTA) en el que cada camino del automata representa una traza observada del SUL. La segunda fase

[1] En el caso de que sea posible generar estas trazas.

que tiene como objetivo reducir el tamaño del PTA, es resuelta por cada algoritmo utilizando una estrategia diferente. RTI+ usa operaciones de mezcla de estados y separación de aristas utilizando pruebas de verosimilitud (likelihood ratio test) como heurística. Por otro lado, TAG y TkT mezclan estados utilizando operaciones basadas en el algoritmo k-Tail [3]. De forma resumida, estos algoritmos mezclan las localizaciones con las mismas secuencias de eventos futuros de longitud k (*k-futuros*). TAG busca combinar las fortalezas de RTI+ y TkT, por eso incorpora además la operación de separación de aristas (como en RTI+). Por su parte, *LearnTA* también realiza dos fases de aprendizaje, aunque son diferentes a los otros algoritmos. En la primera, no se construye un PTA sino que, durante la construcción del autómata, se utilizan operaciones similares a las de mezcla para reducirlo. En la segunda fase, se realizan operaciones de mezcla siguiendo una estrategia basada en los k-futuros que tiene en cuenta las restricciones de tiempo.

3. Funcionamiento de *LearnTA*

Como ya se ha mencionado, el objetivo principal de *LearnTA* es generar un autómata que se comporte como el SUL con respecto al conjunto de trazas observadas. En esta sección, se introducen los Autómata Temporizados, las características específicas del tipo de autómatas que genera el algoritmo *LearnTA* y, también, de las trazas que utiliza para aprender. Finalmente, la sección termina describiendo el algoritmo de aprendizaje.

Autómata Temporizado

Un *autómata temporizado* (TA) [1] es una tupla (Q, q_0, A, C, I, E) donde Q es un conjunto finito de *localizaciones*, $q_0 \in Q$ es la *localización* inicial, A es el conjunto de acciones del sistema y C es el conjunto finito de relojes que representan el paso del tiempo. Denotamos con $B(C)$ el conjunto de restricciones sobre los relojes de C. $I : Q \to B(C)$ es una función que asigna una condición llamada *invariante* (restricción de tiempo) a cada localización. Finalmente, $E \subseteq Q \times A \times B(C) \times 2^C \times Q$ son las aristas etiquetadas de la forma (q, a, g, r, q') donde:

- q y q' son la localización fuente y destino, respectivamente.
- a es una acción de A.
- $g \in B(C)$ son las restricciones de tiempo que constituyen lo que denominamos como la *guarda* de la arista.
- $r \in 2^C$ es el conjunto de relojes que deben ser reiniciados en la transición (sus valores pasa a ser 0).

Adicionalmente, $u : C \to \mathbb{R}^{\geq 0}$ es una función que dado un reloj devuelve su valor (un número real no negativo). El conjunto de todas las valuaciones de los relojes de C se denota como $\mathbb{R}^C$. Dada una restricción $g \in B(C)$ que puede ser una guarda o un invariante, denotamos con $g[u(x)/x]$ a la restricción en la que

todas las apariciones del reloj x se sustituyen por $u(x)$. Si $C = \{x_1, \cdots, x_n\}$, una valuación u satisface una restricción g ($u \in g$) sii[2] y $g[u(x_1)/x_1] \cdots [u(x_n)/x_n]$ es *true*.

Semántica de un autómata temporizado

Dado un TA (Q, q_0, A, C, I, E), su semántica viene dada por un *sistema de transiciones etiquetadas* $\langle S, s_0, \xrightarrow{}\rangle$, donde $S \subseteq Q \times \mathbb{R}^C$ es el conjunto de estados, $s_0 = (q_0, u_0) \in S$ es el estado inicial, y $\xrightarrow{} \subseteq S \times (\mathbb{R}^{\geq 0} \cup A) \times S$ es la relación de transición que modela la evolución del sistema de transiciones de un estado fuente a otro destino. Las transiciones pueden estar etiquetadas por una acción a (transición discreta) o por un número real $d \in \mathbb{R}^{\geq 0}$ (transición continua) que representa el paso del tiempo. La relación de transición $\xrightarrow{}$ se define formalmente como:

- Transición continua: $(q, u) \xrightarrow{d} (q, u + d)$ si y solo si $\forall d', 0 \leq d' \leq d : u + d' \in I(q)$, siendo $u + d$ la valuación que asigna a cada reloj $x \in C$ el valor $u(x) + d$.
- Transición discreta: $(q, u) \xrightarrow{a} (q', u')$ si y solo si $\exists (q, a, g, r, q') \in E$ con $r = \{x_1, \cdots, x_m\}$ tal que $u \in g$, $u' = [r \to 0]u$, y $u' \in I(q')$, donde $[r \to 0]u$ es la valuación que asigna a todos los relojes $x \in C - r$ el mismo valor $u(x)$, mientras que a los relojes de r les asigna el valor 0.

Autómata determinista de tiempo real extendido

El algoritmo de aprendizaje de *LearnTA* construye una versión extendida (EDRTA) de los autómatas deterministas de tiempo real (DRTA) [7], que son la subclase de TA deterministas y con un único reloj que se reinicia después de cada transición. En particular, los EDRTA añaden a las localizaciones atributos observables del sistema. Además, el conjunto de acciones A contiene los eventos de entrada y salida del sistema y una acción especial $\square \in A$ para denotar que no se ha observado ningún evento. A grandes rasgos, las características de los EDRTA son las siguientes:

Localizaciones:

Suponemos que cada observación del sistema hace visible un número n de atributos que son elementos del conjunto $\mathcal{D}$. Asociamos a cada *localización* del automata un vector con los n valores de estos atributos utilizando la función $\nu : Q \to \mathcal{D}^n$. Adicionalmente, asumimos que todos los invariantes de las localizaciones son *true*.

Aristas:

En los EDRTA, la guarda de cada arista es un intervalo de tiempo de la forma $[t_1, t_2]$ con $t_1, t_2 \in \mathbb{N}$ y $t_1 \leq t_2$. Observa que cada intervalo $[t_1, t_2]$ se corresponde con la restricción $t_1 \leq x \wedge x \leq t_2$, siendo x el único reloj del EDRTA. En lo que sigue, utilizamos $ts \in [t_1, t_2]$ para indicar que $u \in [t_1, t_2]$ con $u(x) = ts$.

[2] sii significa "si y solo si", de la misma manera que iff significa "if and only if"

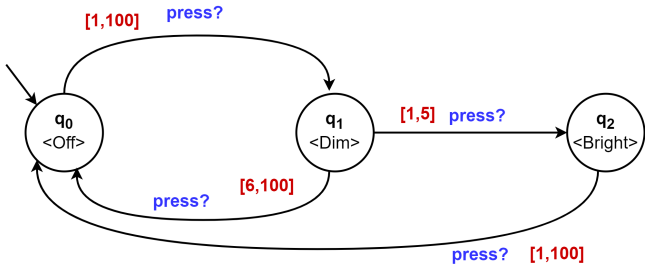

Figura 2: Ejemplo de EDRTA modelando una lámpara

Además, no pueden existir aristas de la forma $(q, \square, [t_1, t_2], q')$[3] con $\nu(q) = \nu(q')$. Esta condición implica que todas las aristas del autómata registran cambios visibles del sistema cuando ocurre un evento de entrada/salida o cuando se observa un cambio en los atributos aunque no se detecte ningún evento ($\square$).

Determinismo:
El algoritmo de aprendizaje asume que el sistema que se está aprendiendo es determinista en el sentido de que una localización no puede tener dos aristas salientes con la misma etiqueta de A y cuyas guardas (intervalos de tiempo) se solapen.

La Figura 2 muestra un EDRTA que modela una lámpara. El autómata tiene tres localizaciones q_0, q_1 y q_2, una acción (evento) *press*? y un atributo observable del sistema, la intensidad, con tres posibles valores $\{Off, Dim, Bright\}$ (apagado, luz tenue y luz brillante). Las aristas están representadas como flechas etiquetadas. Por ejemplo, la arista $(q_1, press?, [1, 5], q_2)$ significa que es posible saltar desde q_1 a q_2 cuando la acción *press*? ocurre entre 1 y 5 unidades de tiempo desde que se llegó a q_1.

Traza de observaciones
Una traza es una secuencia finita de *observaciones* ordenadas por la marca de tiempo en la que se registraron. Cada observación es una 3-tupla (ts, l, v), donde ts es la marca de tiempo; l registra el evento que se ha producido en el sistema (o $\square$ si no se ha observado evento); y v es un vector con el valor observado de los atributos del sistema.

Decimos que un autómata *acepta una traza* de la forma $t = (ts_0, l_0, v_0) \cdot (ts_1, l_1, v_1) \cdots (ts_k, l_k, v_k)$ con $ts_0 = 0, l_0 = \square$ sii existe un camino en el autómata $(q_0, a_1, g_1, q_1) \cdots (q_{k-1}, a_k, g_k, q_k)$ que satisface las siguientes condiciones de aceptación: (1) $v_0 = \nu(q_0)$ y (2) $\forall 0 < i \leq k.\ \nu(q_i) = v_i,\ a_i = l_i, ts_i - ts_{i-1} \in g_i$, es decir, hay una camino en el autómata en el que los atributos de cada localización q_i coinciden con los atributos v_i; y además cada arista (q_{i-1}, a_i, g_i, q_i) es representada por el evento l_i y por el incremento de tiempo entre la observa-

[3] Hemos suprimido el reinicio del reloj porque se realiza en todas las transiciones

Algoritmo 1: Algoritmo de aprendizaje de *LearnTA*

```
1 Algorithm LearnTA(↓ 𝒯 = {t_1, t_2...t_n}, ↓ k, ↑ 𝒜):
2     𝒯' ← CompressTraces(𝒯) ;
3     𝒜' ← LearnFromTraces(𝒯', k) ;
4     𝒜 ← Merge(𝒜', k) ;
5     return 𝒜
```

ción actual y la anterior $ts_i - ts_{i-1}$. En caso contrario, decimos que el autómata *rechaza la traza*.

Por ejemplo, la traza $t = (\square, \langle Off \rangle, 0) \cdot (press?, \langle Dim \rangle, 4) \cdot (\square, \langle Brigth \rangle, 7)$ es aceptada por el EDRTA de la Figura 2 porque el camino $(q_0, press?, [1, 100], q_1) \cdot (q_1, press?, [1, 5], q_2)$ del autómata satisface las condiciones de aceptación. Por otro lado, la traza $t = (\square, \langle Off \rangle, 0) \cdot (press?, \langle Dim \rangle, 20) \cdot (\square, \langle Off \rangle, 300)$ es rechazada por el autómata ya que no existe ningún camino que satisfaga las dos condiciones de aceptación.

3.1. Algoritmo de aprendizaje

En el resto de la sección, utilizamos el autómata Figura 2 como SUL que queremos aprender, y describimos los algoritmos de manera informal, utilizando este ejemplo.

El Algoritmo 1 muestra las principales tareas para la generación del autómata. El algoritmo tiene como entrada un conjunto de trazas observadas $\mathcal{T}$ del SUL y produce un EDRTA $\mathcal{A}$ que acepta todas estas trazas. Además, hace uso de un parámetro $k \in \mathbb{N}^{>0}$, para configurar algunas de las tareas siguiendo la filosofía del algoritmo k-Tail [3].

Compresión de Trazas (*CompressTraces*): El algoritmo comienza con un pre-procesado de las trazas del SUL (línea 2). El objetivo es reducir la longitud de las mismas eliminando observaciones redundantes, es decir, observaciones sucesivas con los mismos atributos y en las que no se ha registrado ningún evento. La Figura 3 muestra un ejemplo de una traza de entrada (izquierda) y de su versión compactada tras el pre-procesado (derecha). Por ejemplo, las observaciones con marca de tiempo 10, 12 y 20 se eliminan ya que no se ha registrado ningún evento y no han cambiado los atributos con respecto a la observación con marca de tiempo 6. Algo similar ocurre con las observaciones con marca de tiempo 1 y 2.

Si $\mathcal{T}$ es el conjunto de trazas de entrada a *LearnTA*, $\mathcal{T}'$ es el conjunto de trazas compactadas que se utilizan en la siguiente tarea del algoritmo.

Aprendizaje de las trazas (*LearnFromTraces*): El objetivo de esta operación es crear de forma incremental un autómata en el que cada observación esté *representada* por alguna localización (entre otras cosas, los atributos de ambas

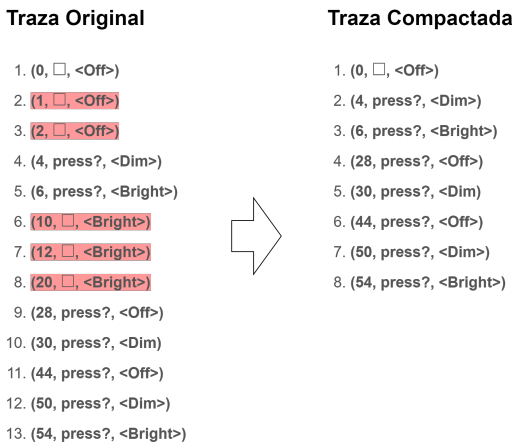

Figura 3: Operación *CompressTraces*

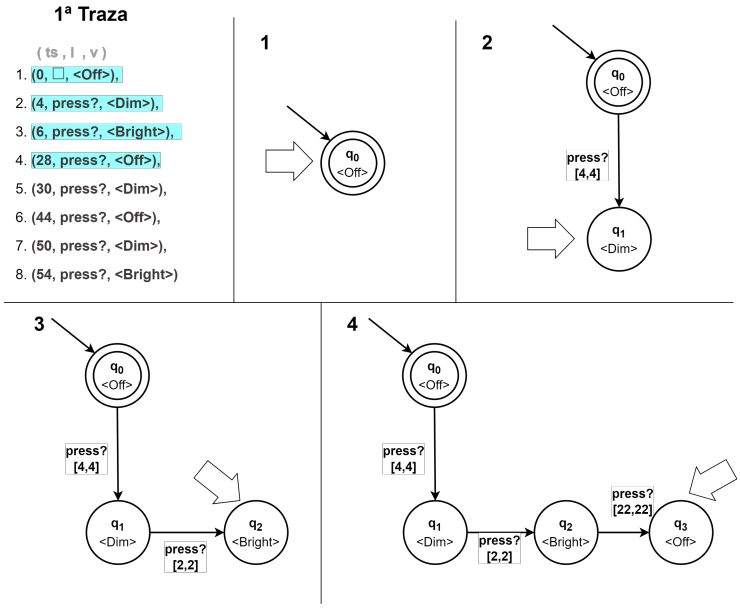

Figura 4: *LearnFromTraces*: Aprendizaje de las observaciones 1-4 de la primera traza

deben coincidir). Para ello, se mantiene siempre una referencia a la localización *distinguida* del autómata que representa la última observación procesada.

LearnFromTraces empieza construyendo un autómata inicial a partir de la primera traza (Figuras 4 y 5) y luego continúa extendiendo el autómata con el resto de las trazas de entrada (Figura 6).

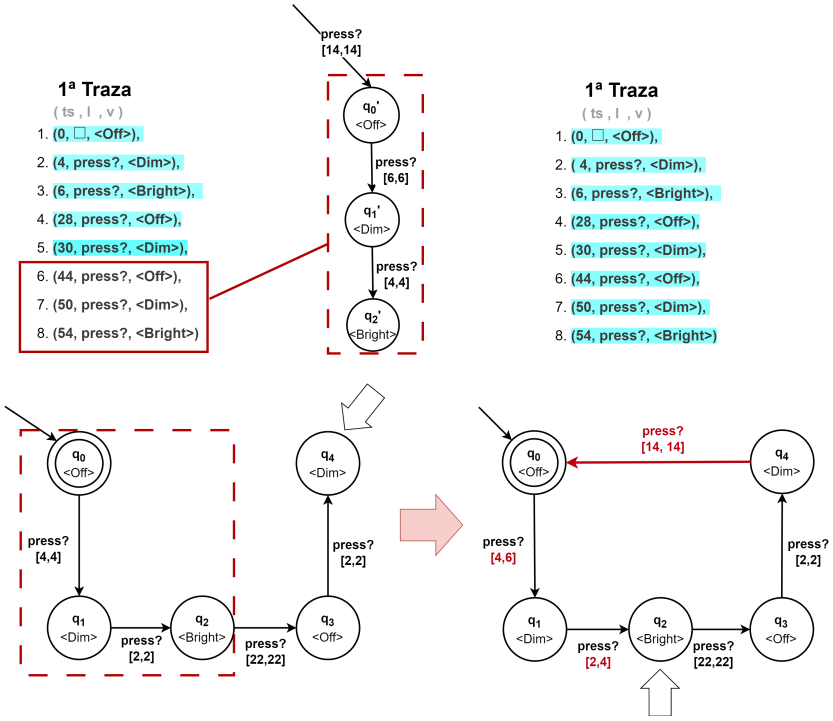

Figura 5: *LearnFromTraces*: Operación *FastMatching* en la primera traza

La primera observación de la primera traza (Figura 4) produce la primera localización del autómata, que se convierte en la localización distinguida. A continuación, se van añadiendo localizaciones y aristas al autómata siguiendo las observaciones de la traza; y se va moviendo la localización distinguida de la forma natural. Las aristas se etiquetan con el evento de la observación y con una guarda que es el incremento de tiempo desde la observación anterior[4].

En la Figura 5 se muestra el efecto de la operación denominada *FastMatching*, que es parte de *LearnFromTraces*. Esta operación se realiza cuando se procesa una nueva observación, pero solo modifica el autómata en algunos casos, como el que se muestra en la Figura 5. En esta figura se asume $k = 2$ y se parte del autómata tras aprender las 5 primeras observaciones de la traza (la localización distinguida es q_4).

La operación *FastMatching* tiene solo en cuenta el evento y los atributos de las observaciones para encontrar ciclos de longitud k en la traza y, de esa forma, reutilizar localizaciones y aristas del autómata ya construido. Por ejemplo, las observaciones 6, 7 y 8 encajan con las aristas $(q_0, press?, \_, q_1)$ y $(q_1, press?, \_, q_2)$, por lo que el algoritmo simplemente añade la arista $(q_4, press?, [14, 14], q_0)$. Además, como el incremento de tiempo de la observación 7 (6 unidades) no

[4] Observa que esto es así porque el reloj se reinicia en cada una de las aristas.

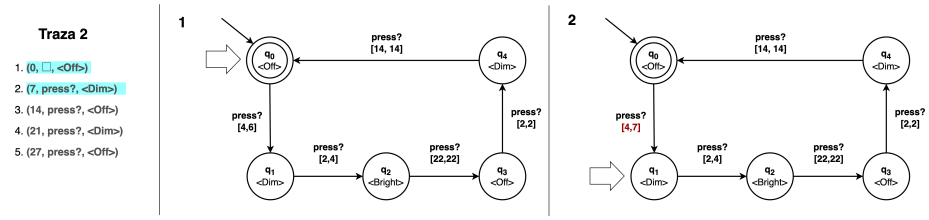

Figura 6: *LearnFromTraces*: Aprendizaje de trazas sucesivas

coincide con la guarda de la arista $(q_0, press?, [4,4], q_1)$, el algoritmo también amplia esa guarda al intervalo $[4,6]$. De forma similar teniendo en cuenta la observación 8, la guarda de la arista $(q_1, press?, [2,2], q_2)$ pasa a ser $[2,4]$. Nótese que si se encuentra un ciclo, se avanza en la traza k observaciones y la localización distinguida pasa a ser la localización que representa a la última observación que forma el ciclo (q_2 en el ejemplo).

La Figura 6 muestra cómo *LearnFromTraces* procesa la segunda traza. Para cada nueva traza el algoritmo siempre parte desde la localización inicial del autómata que asumimos que *representa* la primera observación de una traza. Cuando el algoritmo va a procesar una nueva observación, primero comprueba si ya existe una arista que parta de la localización distinguida etiquetada con el mismo evento de la observación y que lleve a una localización con sus mismos atributos. Ese es el caso de la observación 2 y de la arista $(q_0, press?, [4,6], q_1)$. Como el incremento de tiempo (7) no está incluido en la guarda $[4,6]$, se amplía a $[4,7]$ para contenerlo. Si no existiera dicha arista, se aplicaría la operación *FastMatching* y, en el caso de que no se encuentre ningún ciclo, se añadiría al autómata una nueva localización y una arista. Esta fase seguiría de manera similar hasta procesar todas las trazas pendientes. El autómata final producido en esta fase aparece en el centro de la Figura 7.

Compactación del autómata (*Merge*): La función *Merge* (Algoritmo 1, línea 4) es la encargada de transformar el autómata producido por *LearnFromTraces* en un autómata determinista EDRTA. Como paso previo se realizan operaciones de *mezcla* de localizaciones para reducir el tamaño del autómata. Para ello, se utiliza la noción de localizaciones *similares* inspirada en el algoritmo k-Tail [3]. Dada una localización q del autómata, definimos el conjunto de sus k-futuros como $fut^k(q) = \{(q_0, a_1, g_1, q_1) \cdots (q_{k-1}, a_k, g_k, q_k) |\ q_0 = q\}$, es decir, $fut^k(q)$ es el conjunto de todos los posibles caminos en el autómata de longitud k que comienzan en q. En la Figura 7 se muestran los k-futuros de las localizaciones q_0 y q_3.

Dados dos k-futuros de q_0 y q'_0 $f = (q_0, a_1, g_1, q_1) \cdots (q_{k-1}, a_k, g_k, q_k)$ y $f' = (q'_0, a'_1, g'_1, q'_1) \cdots (q'_{k-1}, a'_k, g'_k, q'_k)$ definimos dos tipos de relación de *similitud*:

1. f y f' son *débilmente-similares* sii $\nu(q_0) = \nu(q'_0)$ y $\forall 0 < i \leq k.\ \nu(q_i) = \nu(q'_i)$, $a_i = a'_i, g_i \cap g'_i \neq \emptyset$. Es decir, dos k-futuros son débilmente-similares si todos

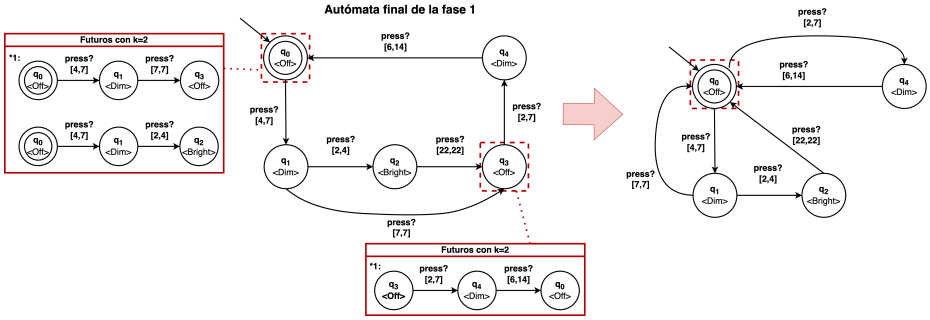

Figura 7: Mezcla de q_0 y q_3

los pares de localizaciones (q_i, q'_i) tienen los mismos atributos, las aristas correspondientes tienen los mismos eventos y sus guardas se solapan.

2. f y f' son *fuertemente-similares* sii $\nu(q_0) = \nu(q'_0)$ y $\forall 0 < i \leq k.\ \nu(q_i) = \nu(q'_i)$, $a_i = a'_i, g_i \subseteq g'_i$ *o* $g'_i \subseteq g_i$. Es decir, dos k-futuros son fuertemente-similares si son débilmente-similares y para cada par de aristas una de las guardas está contenida en la otra.

Usamos estas nociones de similitud para establecer una relación de similitud entre localizaciones (q y q', por ejemplo), teniendo en cuenta sus k-futuros ($fut^k(q)$ y $fut^k(q')$). Consideramos dos casos:

1. Si $fut^k(q)$ y $fut^k(q')$ tienen el mismo número de elementos, entonces para todo k-futuro f de $fut^k(q)$ existe otro f' en $fut^k(q')$ de forma que f y f' son débilmente-similares (y la misma relación a la inversa).
2. Si $fut^k(q)$ tiene menos elementos que $fut^k(q')$, entonces para todo k-futuro f de $fut^k(q)$ existe otro f' en $fut^k(q')$ de forma que f y f' son fuertemente-similares.

Teniendo en cuenta estas definiciones, cuando se encuentran dos localizaciones q y q' similares comienza el proceso de mezcla que consiste en eliminar q' del autómata, y transformar todas las aristas de entrada a q' en aristas de entrada de q y lo mismo para las de salida. Por ejemplo, en la Figura 7, como q_0 y q_3 son similares (por el segundo caso), la localización q_3 desaparece y todas las aristas que llegan o salen de q_3 se reorientan a q_0. De manera similar, la Figura 8 muestra el proceso de mezcla de q_1 y q_4. En este caso, la mezcla produciría aristas con el mismo origen y destino (y el mismo evento).Además, se realiza una mezcla de las aristas actualizando sus guardas.

Cuando no es posible mezclar más localizaciones, se comprueba si el autómata resultante es determinista. Si lo es, el algoritmo finaliza. En otro caso, se intenta resolver el indeterminismo mezclando las aristas y localizaciones implicadas. Si se consigue obtener un autómata determinista, se vuelve a ejecutar la operación de mezcla. En otro caso, el algoritmo termina notificando que el SUL es indeterminista y no es posible generar el autómata. En el ejemplo de

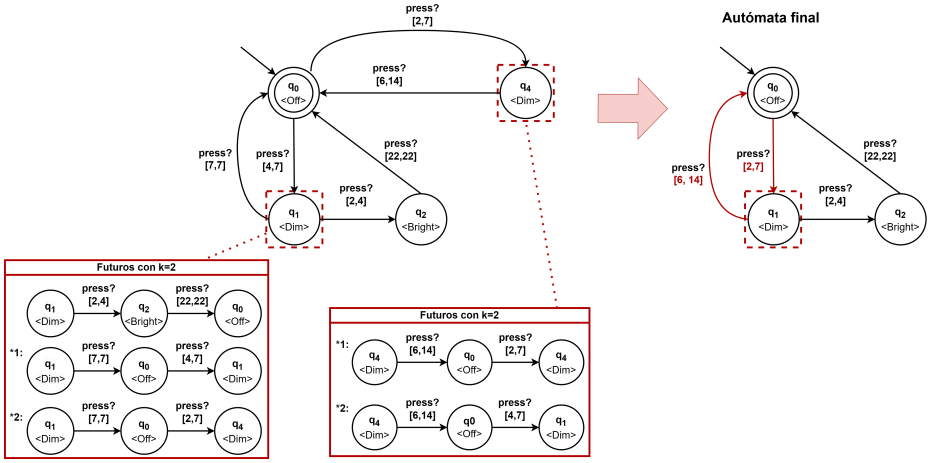

Figura 8: Mezcla de q_1 y q_4

la Figura 8, el autómata resultante ya es determinista por lo que el algoritmo termina.

Finalmente, es interesante señalar que el autómata construido en este ejemplo es *muy parecido* al SUL original de la Figura 2. Las guardas del autómata final reflejan el comportamiento observado y aprendido que, necesariamente, es incompleto.

4. Evaluación

En esta sección se presentan los resultados de la evaluación de *LearnTA*[5] en dos escenarios diferentes. En el primero, se ha analizado la calidad de los autómatas resultantes de aprender trazas extraídas de *SULs sintéticos* usando distintas métricas. En el segundo, además se ha realizado una comparación con la herramienta TAG [4].

Para los experimentos, se ha construido un *Generador de Autómatas Aleatorios* (RAG) configurable que produce EDRTA de diferentes tamaños. Estos autómatas se han utilizado como SULs en los dos escenarios de evaluación. A partir de cada SUL se han creado tres conjuntos de trazas: `LT` son trazas del SUL que se utilizan como entrada del algoritmo de aprendizaje; `TT` son trazas del SUL para ser utilizadas exclusivamente por el verificador; e `IT` son trazas que no pertenecen al SUL y también utilizadas por el verificador. Las trazas de IT se generan recorriendo los autómatas aleatorios, aunque tienen un porcentaje de incluir comportamientos que no pertenecen al SUL. Actualmente estos comportamientos se modelan mediante observaciones con tiempos que están fuera de las guardas del autómata del que se extraen. Las configuraciones de RAG utilizadas durante la evaluación están recogidas en la Tabla 1.

[5] *LearnTA* está disponible en https://gitlab.com/morse-uma/formal-methods/learnta

Parámetro	Escenario Evaluación	Escenario Comparación
Nº localizaciones (n_s)	[50,100]	[4,40]
Nº aristas (n_t)	[$3n_s$,$5n_s$]	[$2n_s$,$4n_s$]
Nº acciones	[5,50]	[4,n_t]
Nº atributos de sistema	[5,50]	-
Min. tiempo en guardas (t_{min})	[2,500]	[2,50]
Max. tiempo en guardas	$[t_{min}+1, t_{min}+500]$	$[t_{min}+1, t_{min}+50]$

Tabla 1: Configuración de RAG para la evaluación

$Precision = \frac{TP}{TP\ +\ FP}$	$Recall = \frac{TP}{TP\ +\ FN}$
$F1 - score = 2 * \frac{Precision * Recall}{Precision + Recall}$	$Accuracy = \frac{TP\ +\ TN}{TP\ +\ TN\ +\ FP\ +\ FN}$
$Acc\ learning = \frac{ALT}{LT}$	$Acc\ test = \frac{ATT}{TT}$
$Acc\ invalid = \frac{AIT}{IT}$	$Time\ elapsed$

Tabla 2: Métricas utilizadas en la evaluación

La Tabla 2 presenta las 8 métricas usadas en la evaluación. En la tabla, `TP` (True Positive) y `FP` (False Positive) representan, respectivamente, el número de trazas válidas/inválidas que acepta el autómata aprendido. `FN` (False Negative) y `TN` (True Negative) son, en este orden, el número de trazas válidas e inválidas que el autómata rechaza. `ALT` (Accepted Learning Traces), `ATT` (Accepted Test Traces) y `AIT` (Accepted Invalid Traces) son el número de trazas de `LT`, `TT` e `IT` aceptadas por el autómata aprendido.

Las métricas de la Tabla 2 dan diferentes medidas estadísticas del comportamiento de *LearnTA*. En este sentido, *Precision* indica el porcentaje de trazas válidas del SUL que el autómata ha aceptado correctamente con respecto al total de trazas que ha aceptado. *Recall*, da una medida ligeramente distinta, ya que proporciona el porcentaje de trazas válidas del SUL que el autómata ha aceptado correctamente con respecto al total de trazas válidas del SUL (`LT` + `TT`). Por otra parte, *F1-score* es una combinación de las dos métricas anteriores. *Accuracy* da el porcentaje de trazas correctamente aceptadas o rechazadas por el autómata con respecto al total de las trazas (válidas e inválidas). Finalmente, Las tres siguientes métricas calculan el porcentaje de trazas correctamente clasificadas de cada conjunto de trazas (`LT`, `TT` e `IT`).

4.1. Evaluación con SULs sintéticos

Para evaluar la calidad de los modelos (EDRTA) producidos por *LearnTA*, se han realizado un total de 1,000 experimentos. En cada uno de ellos se ha producido un SUL sintético configurando RAG con valores de la segunda columna de la Tabla 1. A continuación, para cada SUL, se han creado los tres conjuntos de trazas (`LT`, `TT` e `IT`). Cada conjunto está formado por 750 trazas con 300 observaciones cada una. Posteriormente, se ha ejecutado *LearnTA* para aprender

Prec	Recall	F1-score	Accuracy	Acc learning	Acc valid	Acc invalid	Time (s)
100	99.934	99.967	99.956	100	99.869	100	1.479

Tabla 3: Métricas de *LearnTA* con los SULs sintéticos

un EDRTA utilizando el conjunto de aprendizaje (`LT`). A continuación, se ha utilizado el verificador con los tres conjuntos de trazas y el autómata aprendido para calcular las métricas.

La Tabla 3 contiene la media de las métricas de los 1,000 experimentos. Observa que todas las métricas dan lugar a valores cercanos al 100 %, lo que indica que los autómatas aprendidos se comportan de manera muy similar a los SULs. Además, aunque el algoritmo solo aprende a partir de trazas del SUL, las métricas indican que los autómatas aprendidos son capaces de rechazar trazas con comportamientos que no corresponden al sistema.

4.2. Comparación con TAG

Con respecto a la comparación entre *LearnTA* y TAG [4], hay que tener en cuenta que los autómatas que produce TAG no tienen atributos del sistema en las localizaciones. Por ello, para poder comparar las dos herramientas, se ha asumido que los SULs no tienen atributos observables.

En este escenario, se han realizado 100 experimentos que constan de los siguientes pasos: 1) generación de un SUL sintético con RAG utilizando el rango de valores de la tercera columna de la Tabla 1; 2) creación de los tres conjuntos de trazas, cada uno con 100 trazas de longitud 100; 3) transformación de las trazas al formato de entrada de TAG; 4) ejecución de las dos herramientas con el conjunto de trazas de aprendizaje; 5) cálculo de las métricas para cada uno de los autómatas generados.

La comparación de resultados de estos 100 experimentos se muestra en la Tabla 4. Se observa que *LearnTA* produce autómatas con menos localizaciones y aristas que TAG, aunque esto no influye sustancialmente en los resultados de las métricas realizadas. Aunque TAG tiene una *Precision* ligeramente superior (la diferencia es de menos del 0.1 %), en el resto de las métricas *LearnTA* es algo mejor. Además, es importante resaltar que mientras que TAG tarda alrededor de un minuto por experimento, *LearnTA* solo necesita aproximadamente medio segundo. Otro factor a considerar es que, aunque *LearnTA* está pensado para tener en cuenta los atributos del sistema, el uso de SULs sin atributos no ha afectado de manera significativa a los resultados obtenidos con respecto al anterior bloque de evaluación (alrededor de un 1 % menos por métrica).

5. Conclusiones y Trabajos futuros

En este trabajo se ha presentado *LearnTA*, una herramienta de aprendizaje de modelos enmarcada dentro de las técnicas de *Passive Automata Learning*.

Algoritmo	Prec	Recall	F1-score	Accuracy	Tiempo (s)	Localizaciones	Aristas
LearnTA	99.823	98.695	99.216	99.010	0.671	16	49
TAG	99.918	84.555	91.18	89.65	62.311	91	168

Tabla 4: Comparación de algoritmos

Los modelos resultantes son autómatas deterministas de tiempo real extendidos con atributos del sistema. Además de los atributos, el algoritmo de *LearnTA* se caracteriza por realizar de forma intensiva operaciones que reducen el tamaño de los autómatas sin disminuir en exceso la calidad de los modelos.

Por finalizar, se ha realizado una evaluación de *LearnTA*, analizando la calidad de los autómatas generados con un número significativo experimentos. Para generar los conjuntos de trazas de aprendizaje, de test e inválidas, se han utilizado autómatas generados aleatoriamente que actuaban como SUL. En un caso de estudio real, en el que sistema es una caja negra, es más complejo determinar el conjunto de trazas inválidas. Por último, en la evaluación también se ha comparado *LearnTA* con la herramienta TAG. Los resultados de los experimentos indican que los autómatas construidos por *LearnTA* tienen un comportamiento muy cercano a los SULs originales.

Como líneas futuras de trabajo, se estudiarán distintas técnicas basadas en lenguajes de especificación para mejorar la completitud de los modelos generados. Otra posibilidad es la de estudiar el impacto de incluir ejemplos de trazas negativas en el aprendizaje para el refinamiento de los autómatas resultantes. Por otro lado, también se prevé combinar las técnicas de *Automata Learning* pasivo con técnicas activas y con otras del área de inteligencia artificial. Por ejemplo, la combinación con las técnicas activas, en las que existe un oráculo del sistema, nos permitiría refinar el sistema ya que el oráculo nos permitiría determinar que comportamientos del modelo no corresponden al sistema. Finalmente, se estudiará como utilizar los autómatas aprendidos para llevar a cabo tareas de diagnóstico de errores y reconfiguración automática de sistemas reales.

Agradecimientos Este trabajo ha sido financiado por los proyectos 5G+TACTILE-1 (ref. TSI-063000-2021-11) del Programa UNICO I+D 5G avanzado y 6G, y LearnFDT (ref. PID2022-142181OB-I00) del Plan Estatal de Investigación Científica y Técnica y de Innovación 2021-2023.

Referencias

1. Alur, R., Dill, D.L.: The Theory of Timed Automata. In: Procs. of the Workshop on Real-Time: Theory in Practice, REX. LNCS, vol. 600, pp. 45–73. Springer (1991). https://doi.org/10.1007/BFb0031987
2. Angluin, D.: Learning regular sets from queries and counterexamples. Information and Computation **75**(2), 87–106 (1987). https://doi.org/10.1016/0890-5401(87)90052-6

3. Biermann, A.W., Feldman, J.A.: On the synthesis of finite-state machines from samples of their behavior. IEEE Transactions on Computers **C-21**(6), 592–597 (1972). https://doi.org/10.1109/TC.1972.5009015
4. Cornanguer, L.e.a.: Tag: Learning timed automata from logs. In: 36th AAAI Conf. on Artificial Intelligence. pp. 3949–3958. AAAI Conference on Artificial Intelligence (2022). https://doi.org/10.1609/aaai.v36i4.20311
5. Pastore, F., Micucci, D., Mariani, L.: Timed k-tail: Automatic inference of timed automata. In: 10th IEEE Int. Conf. on Software Testing, Verification and Validation (ICST). pp. 401–411. IEEE Int. Conf. on Software Testing Verification and Validation (2017). https://doi.org/10.1109/ICST.2017.43
6. Verwer, S., de Weerdt, M., Witteveen, C.: An algorithm for learning real-time automata. In: Belgian-Dutch Machine Learning Conference / BeNeLearn. p. 128–135 (2007)
7. Verwer, S., de Weerdt, M., Witteveen, C.: A likelihood-ratio test for identifying probabilistic deterministic real-time automata from positive data. In: Grammatical Inference: Theoretical Results and Applications (ICGI 2010). vol. 6339, pp. 203–216 (2010). https://doi.org/0.1007/978-3-642-15488-1_17
8. Zhang, Y., Lin, Q., Wang, J., Verwer, S.: Car-following behavior model learning using timed automata. IFAC-PapersOnLine **50**(1), 2353–2358 (2017). https://doi.org/10.1016/j.ifacol.2017.08.423, 20th IFAC World Congress