

*Facultad
de
Ciencias*

**IDENTIFICACIÓN DE ACTIVIDADES HUMANAS
UTILIZANDO AUTÓMATAS FINITOS CON
PESOS**
(Human activity recognition using weighted finite
automata)

Trabajo de Fin de Grado
para acceder al

GRADO EN INGENIERÍA INFORMÁTICA

Autor: Miguel Martín Turrión

Directora: Cristina Tîrnăucă

Junio – 2025

Resumen

El presente trabajo aborda la problemática de la identificación de actividades humanas en entornos ubicuos, caracterizados por la recopilación de datos provenientes de múltiples fuentes heterogéneas. El enfoque adoptado se basa en el aprendizaje de patrones de comportamiento, así como de los patrones sensoriales asociados a distintas actividades, mediante la construcción de autómatas finitos ponderados y expresiones regulares, respectivamente. A partir de estos modelos, es posible predecir secuencias de actividades a partir de los datos recopilados, utilizando las expresiones regulares como herramienta de inferencia. El estudio se apoya en un conjunto de datos recogidos durante diez días en un entorno inteligente, donde un único habitante fue monitorizado mediante sensores binarios, un suelo inteligente, balizas BLE y un reloj inteligente, registrando 24 clases de actividades distintas.

Palabras clave: autómatas finitos ponderados, expresiones regulares, aprendizajes de patrones, reconocimiento de actividades humanas.

Abstract

This work addresses the problem of identifying human activities in ubiquitous environments, which are characterized by the collection of data from multiple heterogeneous sources. The approach adopted is based on learning behavioral patterns as well as the sensory patterns associated with different activities, through the construction of weighted finite automata and regular expressions, respectively. Based on these models, it is possible to predict sequences of activities from the collected data, using regular expressions as an inference tool. The study relies on a dataset gathered over ten days in a smart environment, where a single inhabitant was monitored using binary sensors, a smart floor, BLE beacons, and a smartwatch, recording 24 distinct activity classes.

Keywords: weighted finite automata, regular expressions, pattern learning, human activity recognition.

Índice general

1. Introducción	1
2. Descripción del dataset	2
2.1. Formato del dataset	4
2.1.1. Contenido de la carpeta <code>data</code>	4
2.1.2. Contenido de la carpeta <code>layout</code>	5
2.1.3. Contenido de la carpeta <code>pictures</code>	6
2.1.4. Descripción del archivo <code>results.csv</code>	7
2.1.5. Descripción del archivo <code>time-slots training.csv</code>	8
2.2. Formato de tablas de valores	8
2.2.1. Descripción de los archivos <code>acceleration</code>	8
2.2.2. Descripción de los archivos <code>floor</code>	9
2.2.3. Descripción de los archivos <code>proximity</code>	9
2.2.4. Descripción de los archivos <code>sensors</code>	10
2.2.5. Descripción de los archivos <code>activity</code>	10
2.3. Análisis y justificación de las fuentes de datos	11
3. Fundamentos teóricos	13
3.1. Autómatas finitos	13
3.2. Aceptadores de árbol de prefijos	14
3.3. Inferencia de autómatas finitos deterministas a partir de muestras etiquetadas	16
3.4. Expresiones regulares y su relación con los autómatas finitos	19
3.5. Búsqueda de patrones con expresiones regulares	21
4. Implementación	23
4.1. Construcción de autómatas finitos a partir de secuencias de actividades . .	23
4.2. Síntesis de autómatas finitos mínimos a partir de secuencias de sensores . .	25
4.3. Aprendizaje de AFD a partir de ejemplos positivos y negativos	27
4.3.1. Ejemplo ilustrativo de la aplicación del algoritmo RPNI	27
4.3.2. Casos problemáticos durante la fusión de nodos	29
4.4. Conversión de un AFD a una expresión regular	30
4.5. Predicción de secuencias de actividades mediante sensores	31
5. Resultados	33
6. Conclusiones y líneas futuras	37
Bibliografía	38

Índice de figuras

2.1.	Estructura de la carpeta Data .	4
2.2.	Distribución de sensores binarios.	5
2.3.	Distribución de sensores de proximidad.	5
2.4.	Distribución de sensores de suelo.	6
2.5.	Sensores utilizados en el sistema.	7
2.6.	Vista del interior del laboratorio.	7
2.7.	Fragmento de la tabla results .	8
2.8.	Fragmento de la tabla time-slot training .	8
3.1.	Ejemplos para $PTA(S_+)$ (Izquierda) y $APTA(S_+, S_-)$ (Derecha).	15
3.2.	PTA para (S_+) .	17
3.3.	Evolución del autómata tras combinar (q_0) y (q_1) .	18
3.4.	Autómata resultante de combinar $\{(q_0, q_1, q_3)\}$ y $\{(q_5, q_6)\}$.	19
4.1.	Autómata de actividades realizadas el 31 de octubre de 2017 (mañana).	23
4.2.	Autómata que modela la secuencia de actividades realizadas durante la mañana.	24
4.3.	Autómata que modela la secuencia de actividades realizadas durante el mediodía.	24
4.4.	Autómata que modela la secuencia de actividades realizadas durante la tarde.	25
4.5.	Variabilidad en las secuencias de activación de sensores asociadas a la ejecución de la actividad Act15 (Sacar la basura).	26
4.6.	PTA que representa la estructura secuencial de la actividad Act15 (Sacar la basura).	26
4.7.	Evolución del autómata correspondiente a la actividad Act15 .	28
4.8.	Progresión del modelo de autómata correspondiente a la actividad Act15 .	29
5.1.	Fragmento de la tabla time-slot training con las actividades predecidas.	33
5.2.	Transformación de la representación temporal de actividades.	34

Índice de cuadros

2.1. Actividades realizadas y su frecuencia.	3
2.2. Coordenadas de los sensores binarios.	6
2.3. Fragmento de los valores de aceleración realizados en la mañana del 2017-10-31.	9
2.4. Fragmento de los valores recogidos por el suelo en la mañana del 2017-10-31.	9
2.5. Fragmento de los valores recogidos por los sensores de proximidad en la mañana del 2017-10-31.	10
2.6. Fragmento de los valores recogidos por los sensores binarios en la mañana del 2017-10-31.	10
2.7. Registro de actividades realizadas por el participante durante la mañana del 2017-10-31.	11
5.1. Resultados obtenidos al comparar con los datos de entrenamiento y test.	35
5.2. Expresiones regulares obtenidas de la publicación [4].	35
5.3. Resultados obtenidos con expresiones regulares externas sobre los datos.	36

Capítulo 1

Introducción

En la actualidad, un número creciente de aplicaciones en la informática ubicua, la inteligencia ambiental y la asistencia personalizada requieren la capacidad de interpretar el comportamiento humano en entornos reales. Esta interpretación se basa en la obtención, procesamiento y análisis de información relacionada con las actividades que las personas realizan de manera cotidiana, con el fin de ofrecer servicios más adaptativos, eficientes y centrados en el usuario. Comprender el contexto de las acciones humanas no solo permite mejorar la interacción con los sistemas inteligentes, sino también anticipar necesidades y brindar soporte en áreas tan diversas como la salud, la domótica o la industria.

Con el propósito de fomentar la investigación y el desarrollo de metodologías avanzadas en este campo, surge la **UCAmI Cup**, una competición organizada en el marco del congreso internacional UCAmI (*Ubiquitous Computing and Ambient Intelligence*). Esta iniciativa brinda a los investigadores la oportunidad de aplicar sus herramientas, técnicas y modelos sobre un conjunto de datos públicos específicos del reconocimiento de actividades humanas. Además, al establecer un entorno común de evaluación, permite comparar de forma objetiva y equitativa los resultados obtenidos por distintos enfoques, fomentando así la colaboración científica y el avance conjunto en esta disciplina.

En este contexto, el presente Trabajo de Fin de Grado (TFG) tiene como principal objetivo replicar y analizar en profundidad el trabajo presentado por Sergio Salomón y Cristina Tîrnăuică [4], ganadores de la edición del 2018 del concurso, quienes lograron alcanzar una precisión del 90.65 % en el reconocimiento de actividades humanas a partir del conjunto de datos proporcionado por la organización [1]. A partir de dicha base, se explorarán posibles mejoras en la metodología implementada con el fin de incrementar la precisión obtenida sobre el problema abordado.

El documento se estructura de la siguiente manera: el **Capítulo 2** presenta una descripción detallada del conjunto de datos utilizado, así como de los criterios adoptados para su selección y preparación. El **Capítulo 3** expone los fundamentos teóricos necesarios para comprender las técnicas empleadas en el desarrollo del sistema de reconocimiento. A continuación, el **Capítulo 4** detalla el proceso de implementación, describiendo paso a paso las distintas fases que componen el enfoque propuesto. Finalmente, el **Capítulo 5** recoge las conclusiones derivadas del trabajo realizado y plantea posibles líneas de investigación futura que permitan continuar mejorando los resultados alcanzados.

El código generado se encuentra disponible en el repositorio remoto <https://github.com/mmt883/TFG>.

Capítulo 2

Formato, organización interna y criterios de selección de las fuentes de información

El dataset del laboratorio inteligente UJAml se compone de cuatro fuentes de datos diferenciadas extraídas durante un periodo de 10 días (en el cual un único habitante realizó un total de 246 acciones, distribuidas en 24 tipos diferentes de actividades):

- Secuencias de datos generadas por 30 sensores binarios. Estos sensores pueden ser de movimiento, presión o tipo apertura-cierre.
- Información de proximidad basada en la señal *Received Signal Strength Indicator* (RSSI) entre un reloj inteligente, portado por el usuario, y 15 balizas *Bluetooth Low Energy* (BLE) desplegadas en distintas ubicaciones del laboratorio.
- Datos de acelerometría registrados por el reloj inteligente, los cuales proporcionan información sobre la dinámica del movimiento del usuario.
- Un sistema de suelo inteligente compuesto por 40 módulos que aportan información espacial.

Cada fuente de datos está asociada a un identificador único, y todas las muestras registradas contienen marcas temporales (*timestamp*) que indican la fecha y hora exactas de su registro.

Los datos recogidos están divididos en dos subconjuntos diferenciados:

- 1º **Conjunto de entrenamiento** (*Training*): este subconjunto está compuesto por datos recogidos a lo largo de siete días, incluyendo un total de 169 acciones, cada una de ellas acompañada de su correspondiente etiqueta de actividad.
- 2º **Conjunto de *test***): este subconjunto corresponde a los datos recolectados durante un periodo de tres días y está compuesto por 77 acciones no etiquetadas, utilizadas para la evaluación de modelos (sin conocimiento previo de la clase asociada).

El habitante que realizó las actividades es un varón de 24 años, estudiante de la Universidad de Jaén. Durante la recolección de datos, el participante llevaba el reloj

inteligente LG Urbane Model en su mano derecha. Para optimizar el consumo energético, no se han registrado datos durante los períodos en los que el habitante permanecía en la cama o fuera del entorno del laboratorio inteligente UJAml.

En la Tabla 2.1 están descritas las 24 actividades incluidas en el dataset, junto con sus identificadores y la frecuencia correspondiente en el conjunto de entrenamiento.

Tabla 2.1: Listado de actividades realizadas por los participantes y su frecuencia de aparición durante el estudio.

Identificador	Nombre de la actividad	Frecuencia
Act01	Tomar medicación	7
Act02	Preparar desayuno	7
Act03	Preparar comida	6
Act04	Preparar cena	7
Act05	Desayunar	7
Act06	Comer	6
Act07	Cenar	7
Act08	Comer un aperitivo	5
Act09	Ver la television	6
Act10	Entrar en el laboratorio	12
Act11	Jugar a videojuegos	1
Act12	Relajarse en el sofa	1
Act13	Salir del laboratorio	9
Act14	Recibir visita en el laboratorio	1
Act15	Sacar la basura	11
Act16	Lavarse las manos	6
Act17	Lavarse los dientes	21
Act18	Usar el baño	10
Act19	Lavar los platos	2
Act20	Poner la lavadora	6
Act21	Trabajar en el escritorio	2
Act22	Cambiarse de ropa	15
Act23	Irse a la cama	7
Act24	Levantarse de la cama	7

La forma de registrar las actividades realizadas por el habitante ha sido mediante etiquetas NFC y un teléfono móvil. Esto tiene gran importancia debido a que indica la hora de inicio y fin de cada actividad.

2.1. Formato del dataset

En este apartado se explican los contenidos de la carpeta base del conjunto de datos, que está compuesta por tres carpetas (**Data**, **Layout** y **Pictures**) y dos ficheros (**results.csv** y **time-slots training.csv**).

2.1.1. Contenido de la carpeta data

Esta carpeta contiene un conjunto de diez carpetas (tres carpetas para tests y siete carpetas para el entrenamiento de los modelos). El título de cada una de estas carpetas tiene el formato YYYY-MM-DD, donde YYYY indica el año, MM indica el mes y DD indica el día.

Cada una de estas carpetas contiene tres subcarpetas, modificando el formato de título anterior añadiendo una letra (es decir, YYYY-MM-DD-X), donde X toma valores en función del momento del día: A para la mañana, B para el mediodía y C para la tarde.

Se puede observar la estructura del sistema de ficheros en la Figura 2.1. Los detalles sobre el contenido de estos ficheros se presentan en la sección 2.2.

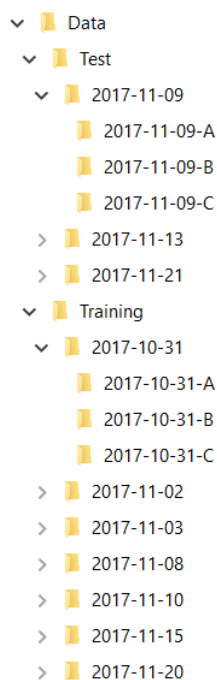


Figura 2.1: Representación de la estructura interna de la carpeta **Data**, que contiene los archivos fundamentales del conjunto de datos utilizado en el estudio.

Las subcarpetas contienen los siguientes ficheros: **YYYY-MM-DD-X-acceleration.csv**, que registra los valores de aceleración observada; **YYYY-MM-DD-X-floor.csv**, con los datos capturados por los sensores del suelo; **YYYY-MM-DD-X-proximity.csv**, que recoge la información de los sensores BLE; y **YYYY-MM-DD-X-sensors.csv**, que incluye medidas relacionadas con los sensores binarios.

En el caso de los datos de entrenamiento, se incluye el fichero **YYYY-MM-DD-X-activity.csv**, que recoge las fechas de inicio y fin, además de las actividades realizadas durante el día.

2.1.2. Contenido de la carpeta layout

La carpeta **Layout** contiene cuatro imágenes y un fichero Word, tal como se describe a continuación.

- La imagen **sensors.png** muestra la distribución de los sensores binarios en un mapa del laboratorio (véase la Figura 2.2).

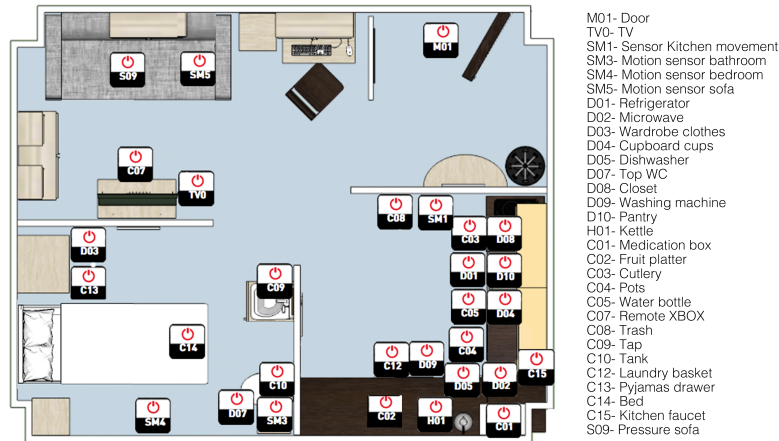


Figura 2.2: Distribución espacial de los sensores binarios instalados en el entorno, utilizada para la detección y el reconocimiento de actividades.

- La imagen **proximity.png** muestra la distribución de los sensores de proximidad en un mapa del laboratorio (véase la Figura 2.3).

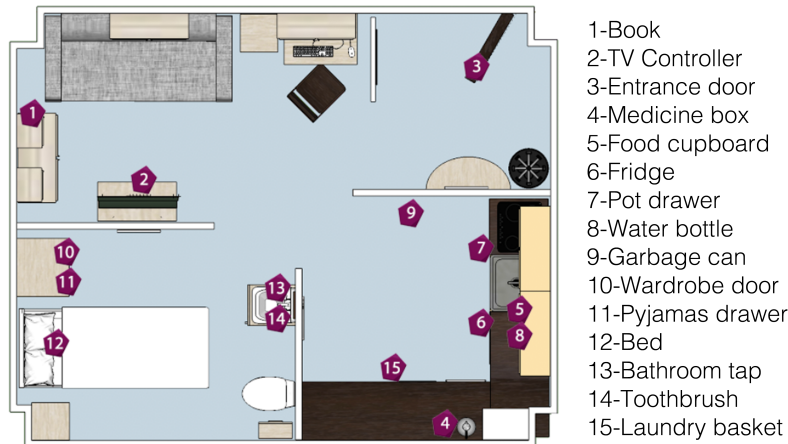


Figura 2.3: Distribución espacial de los sensores de proximidad empleados en el entorno experimental para la detección de presencia y la inferencia de actividades.

- Las imágenes **floor.png** y **floor-modules.png** contienen la localización e identificación de cada sensor de suelo, respectivamente, como se puede observar en la Figura 2.4.

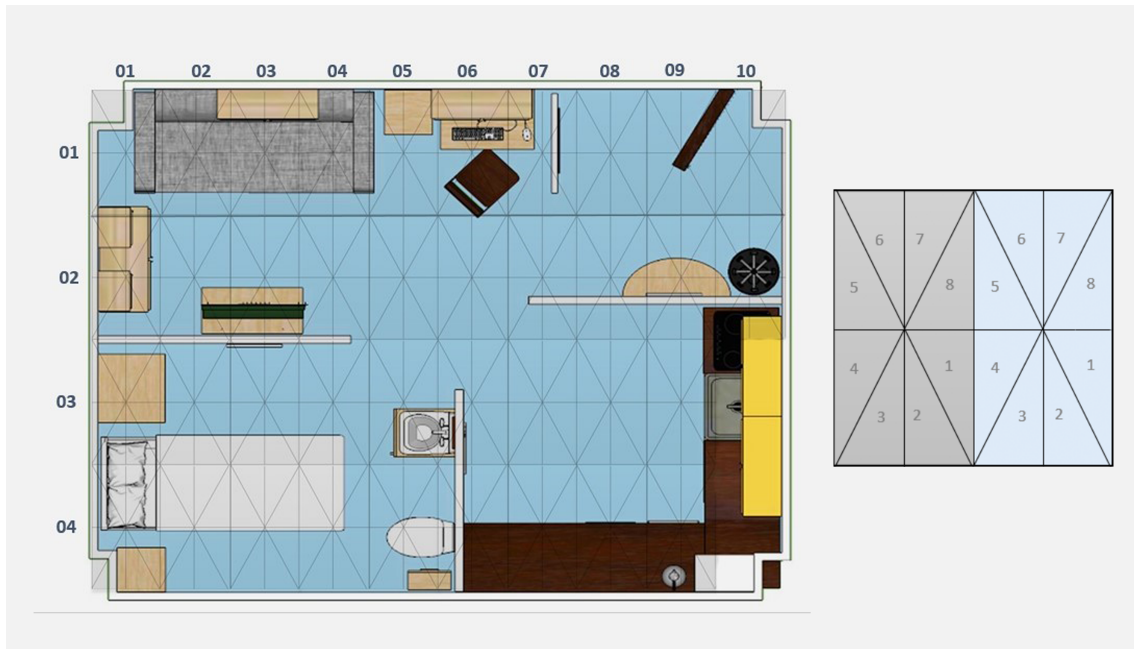


Figura 2.4: Distribución espacial de los sensores de suelo instalados en el entorno experimental, utilizados para la detección de pisadas y el seguimiento de la actividad de los participantes.

- El archivo `Coordinates.docx` contiene las coordenadas X e Y de cada sensor sobre el largo y ancho del laboratorio. A continuación se muestra parte del contenido de ese fichero en la Tabla 2.2.

Tabla 2.2: Coordenadas espaciales de los sensores binarios instalados en el entorno, incluyendo su identificador, nombre descriptivo y posición en los ejes X e Y del plano del escenario experimental.

ID SENSOR	Nombre	X	Y
M01	Puerta	450	460
TV0	Televisión	119	252
D01	Nevera	510	144
D02	Microondas	480	37
D03	Armario de ropa	59	169
SM1	Sensor de movimiento Cocina	580	260

2.1.3. Contenido de la carpeta `pictures`

La carpeta `Pictures` contiene:

- Una carpeta llamada `Binary Sensors`, que contiene imágenes de todos los sensores binarios distribuidos a lo largo del laboratorio, similares al que muestra la Figura 2.5a.

- Una carpeta llamada **BLE sensors**, que contiene imágenes de todos los sensores BLE ubicados en el laboratorio, similares al que muestra la Figura 2.5b.



(a) Sensor binario con identificador C02, empleado para detectar eventos discretos en el entorno.



(b) Sensor BLE instalado en la puerta de entrada, utilizado para la detección de proximidad mediante señales de baja energía.

Figura 2.5: Ejemplos de sensores utilizados en el sistema de reconocimiento de actividades, incluyendo sensores binarios y sensores BLE.

- Una carpeta llamada **SmartLab**, que contiene imágenes del laboratorio por dentro, como se puede ver en la Figura 2.6.



Figura 2.6: Vista del interior del laboratorio donde se llevó a cabo la recolección de datos, mostrando el entorno físico en el que se desarrollaron las actividades monitorizadas.

2.1.4. Descripción del archivo **results.csv** y su función en la codificación temporal de actividades

El archivo **results.csv** contiene una tabla que debe completarse con las actividades realizadas en intervalos de 30 segundos.

Para ello, se debe asignar una columna a cada una de las actividades presentes en los distintos archivos de datos (por ejemplo, **2017-11-09-A-activity.csv**), indicando con los valores **TRUE** o **FALSE** si la actividad correspondiente se está llevando a cabo en el instante de tiempo especificado (véase la Figura 2.7).

2017-11-09-A-activity.csv									
Timestamp	ActXX	ActXX	ActXX	ActXX	ActXX	ActXX	ActXX	ActXX	ActXX
09/11/2017 12:02									
09/11/2017 12:03									
09/11/2017 12:03									
09/11/2017 12:04									
09/11/2017 12:04									
09/11/2017 12:05									
09/11/2017 12:05									
09/11/2017 12:06									

Figura 2.7: Fragmento representativo de la tabla **results**, que muestra la codificación temporal de las actividades registradas durante el experimento.

2.1.5. Descripción del archivo **time-slots training.csv** y su uso en la segmentación de datos para entrenamiento

El archivo **time-slots training.csv** funciona de manera similar al anterior mencionado, ya que incluye una tabla rellena con la actividad que se realiza en cada intervalo de 30 segundos. Contiene, para cada archivo de datos, una secuencia de las actividades dadas y el valor **TRUE** o **FALSE** de cada una de ellas, en función de si se está realizando la actividad en el momento indicado, como se puede observar en la Figura 2.8.

2017-10-31-A-activity.csv								
Timestamp	Act24	Act18	Act16	Act02	Act05	Act17	Act22	Act13
31/10/2017 11:11:30	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE
31/10/2017 11:12:00	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE
31/10/2017 11:12:30	TRUE	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE
31/10/2017 11:13:00	TRUE	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE
31/10/2017 11:13:30	TRUE	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE
31/10/2017 11:14:00	TRUE	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE
31/10/2017 11:14:30	TRUE	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE
31/10/2017 11:15:00	TRUE	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE

Figura 2.8: Fragmento ilustrativo de la tabla **time-slot training**, que contiene los intervalos temporales utilizados para la segmentación de datos durante la fase de entrenamiento del modelo de reconocimiento de actividades.

2.2. Formato de tablas de valores

En esta sección se describen los ficheros disponibles en la carpeta **Data**.

2.2.1. Descripción de los archivos **acceleration**

Estos archivos contienen los datos recopilados por el reloj inteligente del habitante, incluidos en una tabla cuyas columnas son **TIMESTAMP**, **X**, **Y** y **Z**, donde se van actualizando los valores cada dos décimas de segundo aproximadamente. Un pequeño ejemplo de esto se puede ver en la Tabla 2.3.

Tabla 2.3: Fragmento de los valores de aceleración realizados en la mañana del 2017-10-31.

TIMESTAMP	X	Y	Z
2017/10/31 11:11:22.807	-14.513.855	-944.754	-0.8767853
2017/10/31 11:11:22.826	-14.359.131	-96.284.485	-0.9148712
2017/10/31 11:11:22.846	-13.978.271	-9.915.283	-0.97676086
2017/10/31 11:11:22.867	-12.347.717	-10.162.842	-0.8089447

2.2.2. Descripción de los archivos floor

Los archivos en cuestión contienen los datos recopilados por el sistema de suelo inteligente del laboratorio, organizados en una tabla cuyas columnas son: **TIMESTAMP**, **DEVICE** y **CAPACITANCE**. En dicha tabla se registran, respectivamente, la fecha y hora en las que se detecta una modificación en el valor de un sensor, el identificador del sensor determinado por su fila y columna, y los valores correspondientes a cada uno de los ocho módulos que lo componen, separados por comas. Un ejemplo ilustrativo de este formato se presenta en la Tabla 2.4.

Tabla 2.4: Fragmento de los valores recogidos por el suelo en la mañana del 2017-10-31.

TIMESTAMP	DEVICE	CAPACITANCE
31/10/2017 11:12:46	1,09	0.00, 0.00, -1.00, -1.00, 0.00, -3.00, -1.00, 0.00
31/10/2017 11:12:47	1,09	0.00, 0.00, -1.00, -1.00, 0.00, -1.00, -1.00, 0.00
31/10/2017 11:12:47	1,09	0.00, 0.00, -2.00, -1.00, 0.00, -3.00, -1.00, 0.00
31/10/2017 11:13:36	1,07	0.00, -1.00, -2.00, 1.00, 0.00, 0.00, 11.00, -2.00

2.2.3. Descripción de los archivos proximity

Estos archivos contienen los datos recopilados por los sensores BLE incluidos en una tabla cuyas columnas son **TIMESTAMP**, **ID**, **OBJECT** y **RSSI**. En esta tabla los valores se actualizan aproximadamente cada cinco segundos para cada sensor que detecta actividad dentro de su alcance.

Se recoge la fecha y hora en la que se almacena la información del sensor, el ID único de ese sensor y el nombre asignado. Por último, la distancia entre el habitante y el sensor se representa mediante RSSI, es decir, cuanto mayor sea el valor, más cerca se encuentra.

Un pequeño ejemplo de esto se puede ver en la Tabla 2.5

Tabla 2.5: Fragmento de los valores recogidos por los sensores de proximidad en la mañana del 2017-10-31.

TIMESTAMP	ID	OBJECT	RSSI
2017/10/31 11:11:26.831	8108b9e0bcd42be1	WARDROBE DOOR	-95
2017/10/31 11:11:31.967	8108b9e0bcd42be1	WARDROBE DOOR	-95
2017/10/31 11:11:42.796	472c18626db5d102	BED	-68
2017/10/31 11:11:47.966	472c18626db5d102	BED	-84

2.2.4. Descripción de los archivos *sensors*

Estos archivos contienen los datos recopilados por los sensores binarios repartidos por el laboratorio inteligente, que se incluyen en una tabla cuyas columnas son **TIMESTAMP**, **OBJECT**, **STATE** y **HABITANT**.

En dicha tabla se registran la fecha y hora en las que se produce una modificación en un sensor, el identificador del sensor correspondiente, y su estado actual (el cual puede variar según el tipo de sensor). Finalmente, se incluye una columna adicional que indica el habitante que activa el sensor. En este caso particular, esta última información resulta irrelevante, ya que todas las pruebas han sido realizadas por el mismo individuo.

Un pequeño ejemplo de esto se puede ver en la Tabla 2.6.

Tabla 2.6: Fragmento de los valores recogidos por los sensores binarios en la mañana del 2017-10-31.

TIMESTAMP	OBJECT	STATE	HABITANT
2017/10/31 11:08:55.0	SM4	Movement	Mario
2017/10/31 11:09:11.0	SM4	No movement	Mario
2017/10/31 11:09:31.0	C14	Pressure	Mario
2017/10/31 11:09:31.0	SM4	Movement	Mario

2.2.5. Descripción de los archivos *activity*

Estos archivos solo se encuentran en los días destinados al entrenamiento. Su principal función es indicar el inicio y final de cada actividad realizada en ese período.

Para ello contienen una tabla con columnas **DATE BEGIN**, **DATE END**, **ACTIVITY** y **HABITANT**. Debido al mismo motivo que el explicado en el apartado anterior, saber el habitante es irrelevante.

Un pequeño ejemplo se puede observar en la Tabla 2.7.

Tabla 2.7: Registro de actividades realizadas por el participante durante la mañana del 2017-10-31.

DATE BEGIN	DATE END	ACTIVITY	HABITANT
2017/10/31 11:12:38.0	2017/10/31 11:15:25.0	Act24	Mario
2017/10/31 11:15:51.0	2017/10/31 11:16:37.0	Act18	Mario
2017/10/31 11:17:04.0	2017/10/31 11:18:06.0	Act16	Mario
2017/10/31 11:18:36.0	2017/10/31 11:21:54.0	Act02	Mario

2.3. Análisis, evaluación y justificación de la selección de fuentes de datos para el reconocimiento de actividades

En el desarrollo del presente trabajo se ha llevado a cabo un análisis detallado de los distintos tipos de datos disponibles, con el objetivo de seleccionar aquellos que resulten más adecuados para la tarea de reconocimiento de actividades. Esta selección se ha fundamentado tanto en criterios de relevancia informativa como en la viabilidad de su procesamiento y su fiabilidad.

En primer lugar, se han descartado los datos de aceleración. A pesar de su potencial para captar patrones de movimiento, se ha considerado que presentan una complejidad considerable en su tratamiento. Asimismo, se ha estimado que su utilidad para la identificación de actividades específicas dentro del entorno doméstico es limitada, por lo que se ha prescindido de ellos.

En segundo lugar, se han evaluado los datos proporcionados por los sensores de proximidad. No obstante, se ha constatado que estos presentan una elevada sensibilidad al ruido, lo que se traduce en mediciones inestables y poco confiables. Esta falta de precisión compromete su utilidad práctica y puede inducir errores significativos en la interpretación del comportamiento del usuario, motivo por el cual también han sido excluidos del conjunto final de datos considerados.

Por otro lado, se han analizado los datos obtenidos a través del suelo inteligente. Aunque en principio se esperaba que estos pudieran ofrecer una información precisa sobre la localización y desplazamiento del usuario, se han detectado múltiples inconsistencias en su funcionamiento. Entre los errores observados, se encuentran situaciones anómalas tales como transiciones imposibles entre habitaciones en intervalos de tiempo especialmente cortos (por ejemplo, el paso de una habitación a la cocina en un segundo), o la activación de sensores incompatibles con la ubicación real del usuario (como la apertura del cajón de los pijamas mientras el sujeto se encuentra supuestamente en la cocina). Estas deficiencias comprometen seriamente la fiabilidad del sistema, razón por la cual también se ha optado por no considerar estos datos en el análisis.

En consecuencia, el estudio se ha centrado exclusivamente en el uso de datos de los sensores binarios, los cuales indican el estado de activación de distintos sensores distribuidos en el entorno. Este tipo de datos ha demostrado ser el más informativo y el más sencillo de manipular. Su estructura simple permite una representación clara del com-

portamiento del usuario y facilita la implementación de modelos de reconocimiento de actividades sin requerir transformaciones complejas ni suposiciones adicionales sobre el entorno. Por todo lo anterior, los sensores binarios han sido considerados como la fuente de información principal para el desarrollo del sistema propuesto.

Capítulo 3

Fundamentos teóricos del modelado con autómatas y expresiones regulares

En este capítulo se introducen los fundamentos teóricos esenciales para el tratamiento formal del problema, mediante la definición rigurosa de conceptos como los autómatas finitos y las expresiones regulares.

3.1. Autómatas finitos

En el ámbito de la modelación de procesos y actividades, los autómatas finitos constituyen una herramienta fundamental para describir de manera formal el comportamiento de sistemas dinámicos. En particular, los autómatas finitos con pesos permiten incorporar información probabilística a las transiciones entre estados, enriqueciendo así su capacidad de representación. Para establecer las bases conceptuales necesarias, a continuación se presenta una definición formal de estos sistemas.

Definición 1. Un **autómata finito indeterminista** se representa mediante la quintupla $A := (Q, \Sigma, \delta, q_0, F)$, donde:

- Q es un conjunto finito cuyos elementos, denominados **estados**, conforman el espacio de estados,
- Σ es un conjunto finito que representa el **alfabeto**,
- q_0 es un elemento de Q que se denomina **estado inicial**,
- F es un subconjunto de Q , cuyos elementos se denominan **estados finales** o **aceptadores**,
- $\delta : Q \times (\Sigma \cup \{\lambda\}) \rightarrow \mathcal{P}(Q)$ es una función parcial que se denomina **función de transición**, donde $\mathcal{P}(Q)$ representa el conjunto de las partes de Q .

Cuando la función de transición δ toma valores en Q en lugar de $\mathcal{P}(Q)$, es decir, cuando a cada par (estado, símbolo) $\in Q \times \Sigma$ se le asocia un único estado en Q , el autómata se considera **determinista** [2].

Un concepto central asociado a los autómatas finitos es el de **lenguaje reconocido**. Dado un autómata $A = (Q, \Sigma, \delta, q_0, F)$, se define su lenguaje reconocido $L(A) \subseteq \Sigma^*$ como el conjunto de todas las cadenas que, comenzando en el estado inicial q_0 , permiten alcanzar un estado final $f \in F$ mediante una secuencia de transiciones válidas definidas por δ . Este lenguaje formal constituye una herramienta esencial para caracterizar el comportamiento observable del sistema representado y para establecer equivalencias o diferencias entre distintos modelos automáticos.

Un conjunto finito S_+ se denomina **muestra positiva** para el lenguaje L si $S_+ \subseteq L$. De manera análoga, una **muestra negativa** para el lenguaje L es un conjunto finito S_- tal que $S_- \subseteq \Sigma^* \setminus L$. Un conjunto de datos completamente etiquetado incluye todas las cadenas de ejemplo hasta una longitud dada.

Decimos que un autómata es **consistente** con una muestra si acepta todos los ejemplos positivos y rechaza todos los ejemplos negativos. Un conjunto se dice **estructuralmente completo** con respecto a un autómata finito determinista A si cubre cada transición de A y utiliza cada estado final de A .

En el presente trabajo, se adoptará el enfoque de autómatas indeterministas para representar patrones de comportamiento: los nodos representarán distintas actividades y las transiciones se modelarán mediante λ -transiciones, es decir, no se requerirá la lectura de ningún símbolo para realizar el cambio de estado. En su lugar, cada arco tendrá asociado un peso que corresponderá a la probabilidad de dicha transición, la cual será calculada como la razón entre el número de veces que se ha observado dicha transición y el total de transiciones salientes desde el nodo origen. Por otro lado, los autómatas deterministas se utilizarán de manera implícita a la hora de reconocer patrones en los sensores de movimiento.

3.2. Aceptadores de árbol de prefijos

Dado un conjunto S_+ no vacío, sea $PTA(S_+)$ el **aceptador de árbol de prefijos** (*prefix tree acceptor* en inglés, o PTA) para S_+ . $PTA(S_+)$ es un autómata finito determinista que contiene un camino desde el estado inicial hasta un estado de aceptación para cada cadena en S_+ , considerando prefijos comunes. Claramente, $L(PTA(S_+)) = S_+$.

Más formalmente, $PTA(S_+) = (Q, \Sigma, \delta, q_0, F)$, donde:

- $Q = Pr(S_+)$,
- Σ es el alfabeto de S_+ ,
- $\delta(u, a) = ua$, para todo $u, ua \in Q$,
- $q_0 = \lambda$
- $F = S_+$

Un **aceptador de árbol de prefijos aumentado** (*augmented prefix tree acceptor* en inglés, o APTA) con respecto a S_+ y S_- ($(S_+ \cup S_-) \neq \emptyset$), denotado como $APTA(S_+, S_-)$, se define como una 6-tupla $(Q, \Sigma, \delta, q_0, F_+, F_-)$ donde:

- $Q = Pr(S_+ \cup S_-)$,

- Σ es el alfabeto de $S_+ \cup S_-$,
- $\delta(u, a) = ua$, para todo $u, ua \in Q$,
- $q_0 = \lambda$
- $F_+ = S_+$,
- $F_- = S_-$

Cabe destacar que los conjuntos $Pr(S_+)$ y $Pr(S_+ \cup S_-)$ representan el conjunto de todos los prefijos (incluyendo la cadena vacía λ) de las secuencias contenidas en S_+ y en $S_+ \cup S_-$, respectivamente. Formalmente, si $S \subseteq \Sigma^*$, entonces:

$$Pr(S) := \{u \in \Sigma^* \mid \exists v \in \Sigma^* \text{ tal que } uv \in S\}$$

En el contexto del autómata $PTA(S_+)$, el conjunto de estados $Q = Pr(S_+)$ asegura que cada posible prefijo intermedio necesario para alcanzar las cadenas de entrenamiento positivas esté explícitamente representado en la estructura del autómata. De manera análoga, en el caso de $APTA(S_+, S_-)$, el conjunto de estados $Q = Pr(S_+ \cup S_-)$ incluye los prefijos tanto de secuencias aceptadas como rechazadas, permitiendo así capturar rutas compartidas entre ejemplos positivos y negativos. Esto resulta fundamental para tareas de inferencia gramatical y minimización, ya que posibilita detectar puntos de decisión donde las trayectorias de aceptación y rechazo divergen dentro del árbol.

Ejemplo 1. Dados los conjuntos $S_+ = \{0, 1, 010, 011\}$ y $S_- = \{01, 11\}$, el $PTA(S_+)$ y el $APTA(S_+, S_-)$ se ilustran en la Figura 3.1.

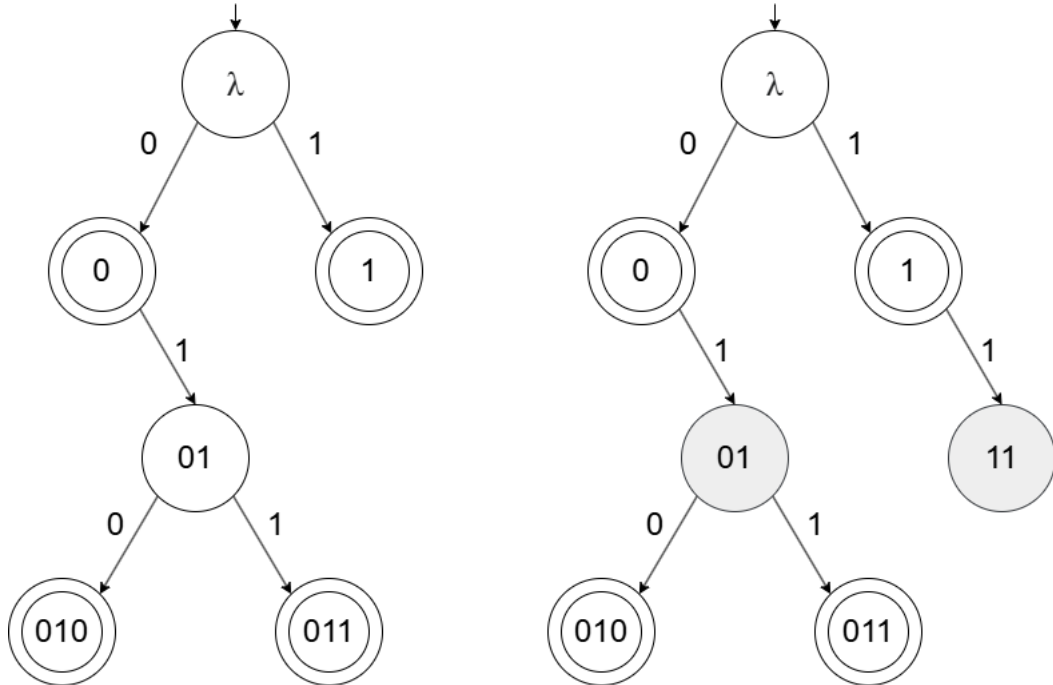


Figura 3.1: Representación gráfica del árbol de prefijos (PTA) construido a partir del conjunto positivo S_+ (izquierda), y del árbol de prefijos aumentado (APTA) generado a partir de los conjuntos S_+ y S_- (derecha).

3.3. Inferencia de autómatas finitos deterministas a partir de muestras etiquetadas

En el contexto del modelado de actividades a partir de secuencias observadas, es fundamental obtener representaciones compactas y generalizables de los patrones de comportamiento. Para ello, se recurre a la inferencia de autómatas finitos deterministas (AFD) a partir de ejemplos positivos y negativos, con el objetivo de minimizar las estructuras que describen cada actividad sin perder capacidad de discriminación. En esta sección se presenta el algoritmo *Regular Positive and Negative Inference* (RPNI).

El algoritmo RPNI [3] es un método clásico para la inferencia de AFD a partir de muestras positivas y negativas. Su principio fundamental se basa en la idea de que, con suficiente evidencia en forma de ejemplos etiquetados, eventualmente se observará un conjunto característico que permita identificar de forma unívoca el lenguaje objetivo.

RPNI es un algoritmo **optimista**: en cada iteración compara estados candidatos para fusión. Una fusión solo se acepta si no contradice la muestra negativa. Este enfoque permite una construcción eficiente del AFD consistente con los datos observados, aunque errores tempranos en fusiones pueden llevar a inconsistencias posteriores.

Sea Σ un alfabeto finito. Dado un conjunto de cadenas positivas $S_+ \subseteq \Sigma^*$ y un conjunto de cadenas negativas $S_- \subseteq \Sigma^*$, el objetivo del algoritmo es construir un AFD mínimo $A_F = (Q, \Sigma, \delta, q_0, F)$ tal que:

- $\forall w \in S_+ : A_F$ acepta w (es decir, $w \in L(A_F)$),
- $\forall w \in S_- : A_F$ no acepta w (es decir, $w \notin L(A_F)$).

Entrada: Conjunto S_+ de ejemplos positivos y conjunto S_- de ejemplos negativos

Salida: un AFD que acepta exactamente el lenguaje compatible con S_+ y rechaza S_-

Algorithm 1 Algoritmo RPNI

```

1:  $A := PTA(S_+)$ 
2:  $K := \{q_0\}$ 
3:  $Fr := \{\delta(q_0, a) \mid a \in \Sigma^*\}$ 
4: while  $Fr \neq \emptyset$  do
5:   elegir  $q$  de  $Fr$ 
6:   if  $\exists p \in K$  tal que  $L(dmerge(A, p, q)) \cap S_- = \emptyset$  then
7:      $A := dmerge(A, p, q)$ 
8:   else
9:      $K := K \cup \{q\}$ 
10:  end if
11:   $Fr := \{\delta(q, a) \mid q \in K\} - K$ 
12: end while

```

En el Algoritmo 1, **dmerge** actúa de la siguiente forma:

- Intenta fusionar p con q si:
 - ambos nodos pertenecen al mismo conjunto de ejemplos (es decir, ambos Positivos o ambos Negativos).

- la fusión de sus descendientes en el árbol no introduce inconsistencias.
- La fusión de dos nodos p y q consiste en:
 - eliminar p del autómata y redirigir sus transiciones entrantes hacia q y sus transiciones salientes desde q .
 - iterativamente combinar nodos descendientes que tengan transiciones con la misma etiqueta (homogeneización).
- Si tras la fusión el conjunto de cadenas negativas s_- sigue sin ser aceptado por el autómata, la fusión se considera válida; en caso contrario, se descarta.

A continuación se presenta un ejemplo ilustrativo del funcionamiento del algoritmo RPNI. Consideramos el alfabeto binario $\Sigma = \{0, 1\}$ y los siguientes conjuntos de ejemplos:

- Conjunto positivo $S_+ = \{\lambda, 0, 00, 10, 010\}$
- Conjunto negativo $S_- = \{01\}$

Paso 1: Construcción del PTA

El algoritmo inicia construyendo el árbol de prefijos a partir de S_+ en el caso del algoritmo RPNI. El PTA generado se puede observar en la Figura 3.2. Para facilitar su referencia, se denotan los estados del autómata con la letra q seguida de un subíndice i , de modo que $q_0 = \lambda$, $q_1 = 0$, $q_2 = 1$, $q_3 = 00$, $q_4 = 01$, $q_5 = 10$ y $q_6 = 010$. Los estados aceptores son aquellos en los que terminan las cadenas positivas: q_0 , q_1 , q_3 , q_5 , q_6 .

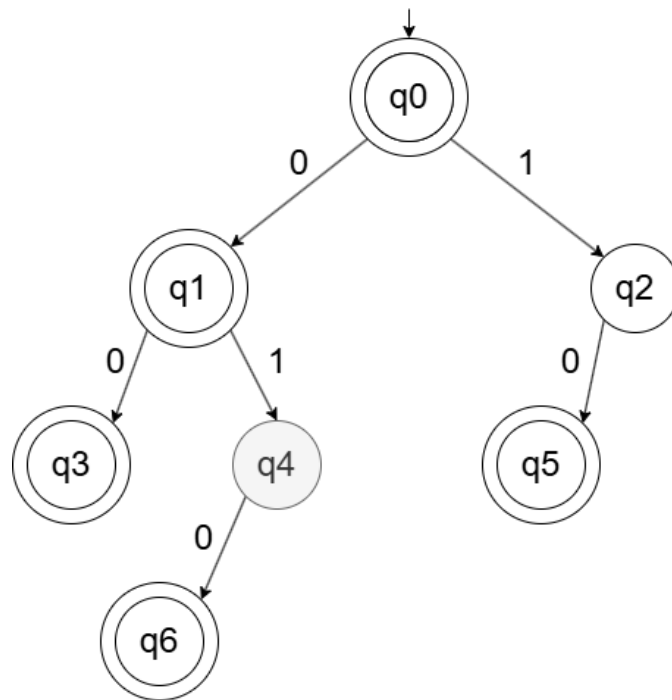
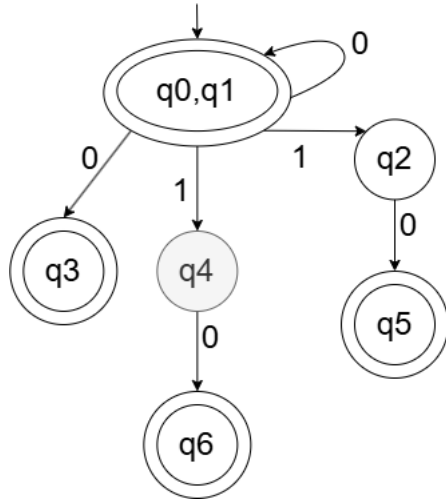


Figura 3.2: PTA construido a partir del conjunto positivo S_+ , que representa todas las secuencias aceptadas por el sistema antes de aplicar procesos de generalización.

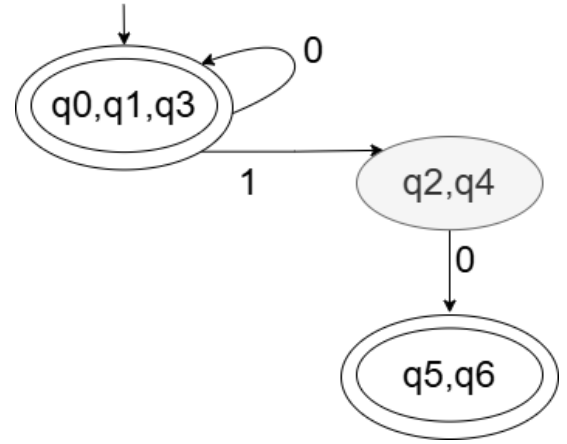
Paso 2: Proceso de generalización (fusión de estados)

El conjunto K se inicializa con $\{q_0\}$ y se procede a evaluar la posibilidad de fusionar cada estado de la frontera $Fr = \{q_1, q_2\}$ con alguno de los ya presentes en K , bajo la condición de que el lenguaje resultante no acepte ninguna de las cadenas contenidas en el conjunto de muestras negativas S_- .

En primer lugar, se analiza el estado q_1 . Se intenta su fusión con q_0 , lo cual da lugar a la estructura mostrada en la Figura 3.3a. Posteriormente, se deben combinar los nodos (q_0, q_1) con q_3 , q_2 con q_4 y q_5 con q_6 más tarde al poder llegar a ambas parejas con el mismo símbolo. En consecuencia, se muestra el autómata después de todas las combinaciones en la Figura 3.3b. Es importante señalar que, en las combinaciones entre nodos neutros — como el nodo q_2 — y nodos de otro tipo, la clasificación resultante adopta el tipo del nodo predominante, es decir, aquel que no posee carácter neutro.



(a) Autómata resultante de la combinación directa de los estados (q_0) y (q_1) .



(b) Autómata tras aplicar las combinaciones adicionales inducidas por la fusión inicial de (q_0) y (q_1) .

Figura 3.3: Evolución del autómata durante el proceso de generalización, tras la combinación de los estados (q_0) y (q_1) , y las fusiones inducidas por dicha operación.

En este momento de la ejecución, $K = \{(q_0, q_1, q_3)\}$ y $Fr = \{(q_2, q_4)\}$. A continuación, se considera la fusión de los estados (q_0, q_1, q_3) y (q_2, q_4) . Esta operación implica la unificación de estados que pertenecen a distintas clases de muestra, por lo que la combinación resulta igualmente inviable y debe ser anulada. Por lo tanto, K pasa a ser $\{(q_0, q_1, q_3), (q_2, q_4)\}$ y la frontera $Fr = \{(q_5, q_6)\}$.

Se intenta entonces la fusión de (q_0, q_1, q_3) con (q_5, q_6) . Para ello, será necesario modificar la arista $(q_2, q_4) \rightarrow (q_5, q_6)$ hacia el nuevo estado $(q_0, q_1, q_3, q_5, q_6)$. El proceso está ilustrado en la Figura 3.4. En este momento de la ejecución, $K = \{(q_0, q_1, q_3, q_5, q_6)\}$ y $Fr = \{(q_2, q_4)\}$.

La última combinación tendrá lugar entre los nodos $(q_0, q_1, q_3, q_5, q_6)$ y (q_2, q_4) , pero al pertenecer un nodo a las muestras positivas y otro a las negativas, no es posible combinarlos.

En conclusión, no es posible realizar combinaciones adicionales sin inducir la fusión de estados aceptadores (asociados a muestras positivas) con estados correspondientes a muestras negativas.

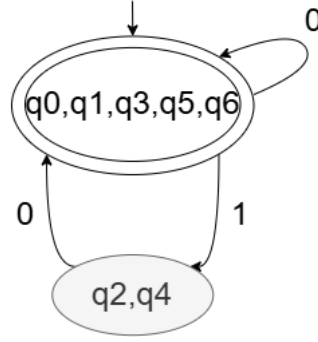


Figura 3.4: Autómata resultante tras la combinación de los estados $\{(q_0, q_1, q_3)\}$ y $\{(q_5, q_6)\}$, como parte del proceso de generalización estructural

Paso 3: Resultado final

El autómata resultante acepta únicamente las cadenas de S_+ y rechaza todas las de S_- , cumpliendo con los criterios de consistencia. Este proceso demuestra cómo el algoritmo RPNI puede aprender un lenguaje regular infinito a partir de un número finito de ejemplos etiquetados.

3.4. Expresiones regulares y su relación con los lenguajes regulares y los autómatas finitos

En el estudio de los lenguajes formales y la teoría de autómatas, las expresiones regulares constituyen una herramienta fundamental para la representación y manipulación de conjuntos de cadenas sobre un alfabeto finito. Estas expresiones permiten describir de manera compacta y estructurada los lenguajes aceptados por autómatas finitos.

En esta sección, se introduce la definición formal de las expresiones regulares, estableciendo las reglas sintácticas que determinan su construcción a partir de un alfabeto base y un conjunto de operadores especiales. Posteriormente, se aborda la asignación semántica asociada a cada expresión, es decir, la relación entre la sintaxis de las expresiones regulares y los lenguajes formales que estas describen.

Finalmente, se presentan algunas propiedades fundamentales relacionadas con la equivalencia de expresiones regulares y las operaciones básicas que preservan su significado, proporcionando las bases teóricas necesarias para su uso en contextos de modelado, análisis y verificación de lenguajes formales.

Definición 2. Sea Σ un alfabeto finito. Llamaremos **expresión regular** sobre el alfabeto Σ a toda secuencia de símbolos sobre el alfabeto

$$\Sigma_1 := \{\emptyset, \lambda, +, \cdot, (,), *\} \cup \Sigma,$$

conforme a las reglas siguientes:

Las siguientes son expresiones regulares:

- el símbolo $\emptyset$ es una expresión regular,

- el símbolo λ es una expresión regular,
- el símbolo a es una expresión regular para cualquier $a \in \Sigma$.

Si α y β son expresiones regulares, también lo son las construidas mediante las reglas siguientes:

- $(\alpha + \beta)$ es una expresión regular,
- $(\alpha \cdot \beta)$ es una expresión regular,
- $(\alpha)^*$ es una expresión regular.

Por comodidad de la escritura (y solo en el caso de que no haya ninguna posibilidad de ambigüedades), se suprimen los paréntesis y los símbolos de producto $(\cdot)$. También, para evitar ambigüedades, el orden de prioridad será: primero las expresiones con paréntesis, después $\cdot$ y finalmente $+$.

A cada objeto sintáctico se le puede asociar un mecanismo de asignación de significado, conocido como **semántica**. En este trabajo, se considera que cada expresión regular tiene un único significado, definido como el lenguaje formal que describe.

Definición 3. Sea Σ un alfabeto finito. A cada expresión regular α sobre el alfabeto Σ le asignaremos un lenguaje formal $L(\alpha) \subseteq \Sigma^*$ conforme a las siguientes reglas:

- Si $\alpha = \emptyset$, entonces $L(\alpha) = \emptyset$,
- Si $\alpha = \lambda$, entonces $L(\alpha) = \{\lambda\}$,
- Si $\alpha = a \in \Sigma$, entonces $L(\alpha) = \{a\}$.

Aplicando reglas recursivas, si α y β son dos expresiones regulares sobre el alfabeto Σ usaremos las reglas siguientes:

- Si $\alpha = \beta + \gamma$, entonces $L(\alpha) = L(\beta) \cup L(\gamma)$,
- Si $\alpha = \beta \cdot \gamma$, entonces $L(\alpha) = L(\beta) \cdot L(\gamma)$,
- Si $\alpha = \beta^*$, entonces $L(\alpha) = (L(\beta))^*$.

Definición 4. Dado un alfabeto Σ , si L_1 y L_2 son lenguajes sobre Σ , entonces:

- La **concatenación** de L_1 y L_2 , denotada $L_1 L_2$, es el lenguaje formado por todas las palabras que se obtienen concatenando una palabra de L_1 con una palabra de L_2 , es decir:

$$L_1 L_2 = \{xy : x \in L_1, y \in L_2\}.$$

- La **estrella de Kleene** de un lenguaje L , denotada L^* , es el conjunto de todas las concatenaciones finitas (incluida la palabra vacía) de palabras de L , es decir:

$$L^* = \bigcup_{n \in \mathbb{N}} L^n,$$

donde $L^0 := \{\lambda\}$ y $L^{n+1} := L \cdot L^n$.

Teorema 1 (Equivalencia entre autómatas finitos y expresiones regulares). Sea Σ un alfabeto finito. Un lenguaje $L \subseteq \Sigma^*$ es reconocible por un autómata finito si y solo si existe una expresión regular α sobre Σ tal que $L = L(\alpha)$.

Esta proposición establece la equivalencia formal entre los lenguajes que pueden ser definidos mediante expresiones regulares y aquellos que pueden ser aceptados por autómatas finitos. En otras palabras, el conjunto de lenguajes regulares coincide exactamente con el conjunto de lenguajes reconocidos por autómatas finitos (deterministas o no).

Como consecuencia de esta equivalencia, se deduce que para cada expresión regular existe un autómata finito que acepta el mismo lenguaje, y viceversa, para cada autómata finito existe una expresión regular que describe el lenguaje que este reconoce. Esta relación bidireccional permite, en particular, establecer procedimientos constructivos que transforman expresiones regulares en autómatas y autómatas en expresiones regulares.

En este trabajo, se utilizarán expresiones regulares para modelar las secuencias de sensores que caracterizan una determinada actividad, aprovechando su capacidad para describir patrones de eventos en flujos de datos discretos. Para su construcción, se emplean AFD aprendidos mediante el algoritmo RPNI, descrito en la sección 3.3.

La relación entre expresiones regulares y autómatas finitos también se manifiesta de forma implícita en su aplicación práctica. Por ejemplo, al ejecutar una búsqueda con una expresión regular, el sistema convierte internamente dicha expresión en un autómata finito, que recorre la secuencia de entrada para verificar si esta satisface el patrón especificado.

3.5. Búsqueda de patrones con expresiones regulares

El proceso de búsqueda consiste en comparar el patrón definido por la expresión regular con una cadena de entrada, tratando de encontrar una subcadena que cumpla con las reglas impuestas por el patrón. Estas reglas incluyen, entre otras:

- **Coincidencia literal:** se busca una secuencia exacta de caracteres.
- **Metacaracteres** como `.`, `*`, `+`, `?`, `[ ]`, `( )`, etc., que permiten definir patrones complejos.
- **Anclas** como `^` (inicio de línea) y `$` (fin de línea), que restringen la posición donde debe aparecer la coincidencia.

El motor de búsqueda evalúa la cadena de entrada carácter por carácter, construyendo una posible coincidencia. Si en algún punto se alcanza un estado de aceptación según el autómata correspondiente al patrón, se considera que se ha encontrado una coincidencia.

En el lenguaje de programación Python, el módulo `re` proporciona funcionalidades para trabajar con expresiones regulares. Una de las funciones más utilizadas dentro de este módulo es `re.search`.

Definición y sintaxis

```
import re
resultado = re.search(patron, cadena)
```

`re.search` recibe como parámetros una expresión regular y una cadena de caracteres. Su objetivo es encontrar la primera coincidencia del patrón dentro de la cadena, devolviendo

un objeto de tipo `Match` si se encuentra alguna coincidencia, o `None` si no se encuentra ninguna.

Características principales

- **Búsqueda parcial:** a diferencia de `re.fullmatch`, que requiere que toda la cadena coincida, `re.search` busca cualquier subcadena que cumpla con el patrón.
- **Primera coincidencia:** la búsqueda se detiene en la primera subcadena que coincida con el patrón.
- **Estrategia greedy:** si el patrón lo permite, la coincidencia obtenida será la más larga posible que cumpla con las condiciones especificadas.
- **Objeto Match:** si la búsqueda tiene éxito, se puede utilizar este objeto para obtener más información sobre la coincidencia mediante métodos como `group()`, `start()`, `end()` y `span()`.

Ejemplo de uso

```
import re

texto = "La temperatura es de 23 grados y mañana será de 19."
patron = r"\d+"

resultado = re.search(patron, texto)

if resultado:
    print("Coincidencia encontrada:", resultado.group())
```

En este ejemplo, la cadena contiene dos secuencias numéricas: “23” y “19”. El patrón `\d+` busca una o más cifras consecutivas. La función `re.search` aplica una búsqueda lineal de izquierda a derecha, y retorna la **primera subcadena** que satisface el patrón. Por lo tanto, aunque ambas cifras coinciden con la expresión regular, la función devuelve únicamente “23”, ya que es la primera coincidencia encontrada.

Capítulo 4

Implementación del sistema de reconocimiento de actividades

En esta sección se detalla el proceso de implementación del sistema desarrollado para la identificación de actividades humanas en un entorno ubicuo, materializando de forma práctica los conceptos teóricos expuestos en la sección 3. Se describen las etapas de construcción de autómatas finitos ponderados para cada etapa del día, la obtención de autómatas mínimos y expresiones regulares para cada actividad, con el objetivo de obtener patrones de comportamiento y actividad. Finalmente, se presenta la ejecución de la estrategia propuesta sobre los datos de test.

4.1. Construcción de autómatas finitos a partir del análisis de secuencias de actividades

El primer paso en el proceso consiste en extraer las posibles secuencias de actividades realizadas por el habitante. Esto permite inferir las actividades potenciales que pueden suceder a una actividad dada. Las secuencias mencionadas se obtienen a partir de todos los archivos de entrenamiento con formato `YYYY-MM-DD-X-activity.csv`. Estas secuencias se incluirán posteriormente en autómatas finitos, según la definición formal previamente establecida en la sección 3.1.

El objetivo es construir múltiples autómatas: uno para representar las actividades matutinas (denominado autómata-A), otro para las actividades del mediodía (autómata-B) y un tercero para las actividades realizadas por la tarde (autómata-C). Para la construcción del autómata-A, se combinan las secuencias de actividades correspondientes a la mañana de cada día. Por ejemplo, dada la siguiente secuencia: **Act24**, **Act18**, **Act17**, **Act02**, **Act05**, **Act17**, **Act22**, **Act13**, es posible construir el autómata que se muestra en la Figura 4.1, donde cada nodo representa una actividad individual.



Figura 4.1: Autómata que representa la secuencia de actividades realizadas por el participante durante la mañana del 31 de octubre de 2017

A partir de la combinación de los autómatas obtenidos para cada día, se genera el autómata-A, ilustrado en la Figura 4.2. En dicho autómata, las aristas están etiquetadas con la frecuencia relativa de cada transición, calculada como la razón entre el número de veces que dicha transición ha sido observada y el total de transiciones salientes desde el mismo nodo origen.

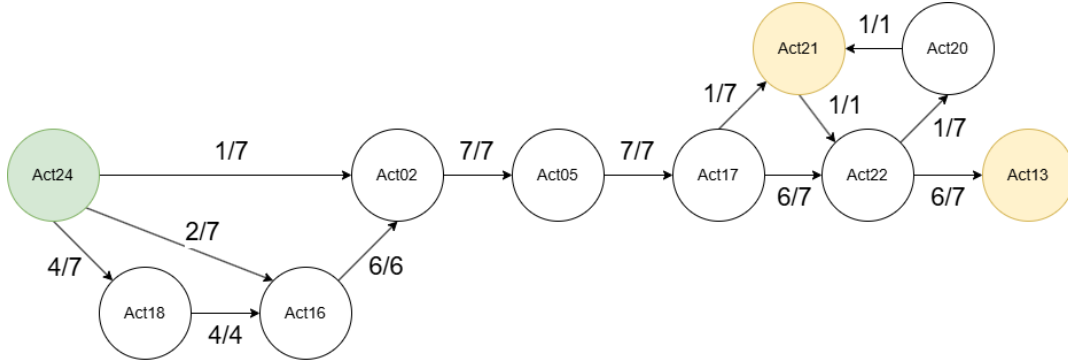


Figura 4.2: Autómata que modela la secuencia de actividades realizadas durante la mañana.

Además, los nodos correspondientes a actividades que pueden iniciar una secuencia están etiquetados con la palabra **Inicial** y se representan en color verde. De manera análoga, los nodos que pueden finalizar una secuencia están etiquetados como **Final** y se muestran en color amarillo.

En particular, en lo que respecta a las actividades realizadas por la mañana, se ha observado que el habitante siempre inicia su rutina con la actividad **Act24** (Despertarse). En cuanto a las actividades finales de la mañana, estas suelen ser la actividad **Act21** (Trabajar en el escritorio) o, en su mayoría, la actividad **Act13** (Salir del laboratorio).

La Figura 4.3 presenta el autómata-B, correspondiente a las secuencias de actividades realizadas durante el período del mediodía.

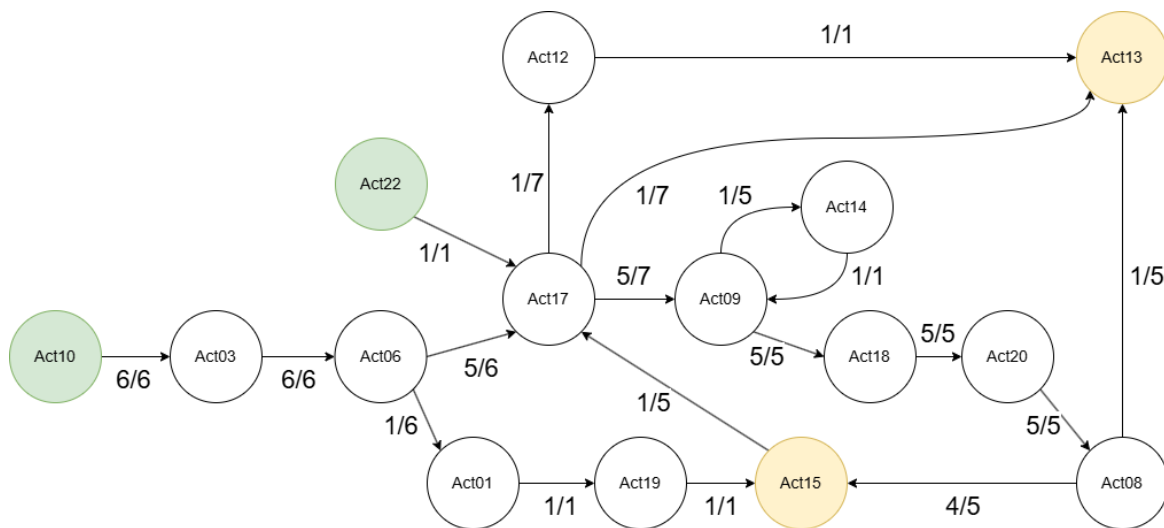


Figura 4.3: Autómata que modela la secuencia de actividades realizadas durante el mediodía.

Las trayectorias inician con una de las actividades **Act10** (Entrar en el laboratorio), **Act03** (Preparar comida) y **Act06** (Comer), para después llegar a **Act17** (Lavarse los dientes), que actúa como nodo central del modelo. A partir de **Act17** se ramifican diversas trayectorias que conducen a actividades intermedias, que eventualmente concluyen en los nodos **Act15** (Sacar la basura) o **Act13** (Salir del laboratorio), identificados como estados **Finales**.

La estructura del grafo refleja una mayor ramificación de trayectorias en comparación con el autómata anterior, lo que permite representar con mayor precisión la variabilidad observada en las rutinas del mediodía.

La Figura 4.4 muestra el autómata-C, que representa las secuencias de actividades realizadas por el habitante durante la tarde.

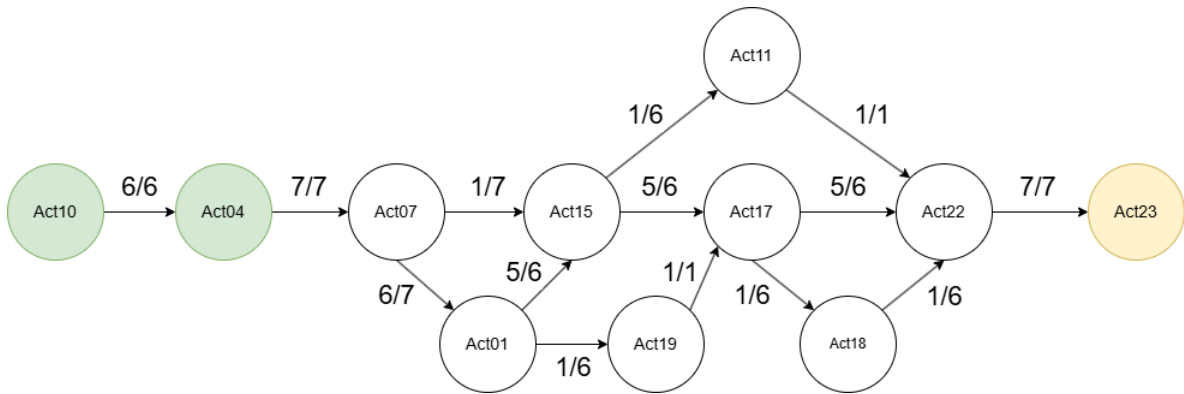


Figura 4.4: Autómata que modela la secuencia de actividades realizadas durante la tarde.

El recorrido de actividades comienza de forma consistente con las actividades **Act10** (Entrar en el laboratorio) y **Act04** (Preparar cena), comenzando con estas **seis** veces y **una** vez, respectivamente. Desde ese punto, el habitante puede seguir distintas trayectorias, antes de finalizar en **Act23** (Irse a la cama).

4.2. Síntesis de autómatas finitos mínimos a partir de secuencias de sensores

El próximo objetivo es obtener una representación finita de las secuencias de sensores activados durante cada una de las actividades. Para ello, es necesario realizar ciertos pasos intermedios.

El primer paso consiste en obtener todas esas secuencias de sensores. Con este fin, se accede a los archivos `YYYY-MM-DD-X-activity.csv`, donde se observa la actividad realizada y los tiempos de inicio y fin. Con estos valores, se usan los archivos `YYYY-MM-DD-X-sensors.csv` para extraer los sensores activados entre estos tiempos para cada actividad.

La Figura 4.5 muestra las secuencias de sensores filtradas para la actividad **Act15** (Sacar la basura). Estas secuencias de sensores serán filtradas para eliminar los sensores de movimiento (SM) cuando exista al menos un sensor de otro tipo en todas las secuencias. El motivo de filtrar estos sensores es porque pueden contener ruido. Por ello, solo se utilizarán cuando no haya fuentes de información más fiables.

[M01 M01]
 [C01 C01 C08 C08 M01 M01]
 [C08 C08 M01 M01]
 [C01 C08 C08 M01 M01]
 [M01 M01]
 [M01 M01]
 [C01 M01 M01]
 [C08 C08 M01 M01]
 [C08 C08 M01 M01]
 [C08 C08 M01 M01]
 [C08 C08 M01 M01]

Figura 4.5: Variabilidad en las secuencias de activación de sensores asociadas a la ejecución de la actividad **Act15** (Sacar la basura).

Cabe destacar que el autómata resultante es un PTA, es decir, un autómata en forma de árbol donde ninguna transición es compartida entre palabras distintas. Se puede observar el PTA de la actividad **Act15** en la Figura 4.6. Para simplificar la figura, se denotan los estados del autómata con la letra q seguida de un subíndice i , de modo que $q_0 = \lambda$, $q_1 = M01$, $q_2 = C01$, $q_3 = C08$, $q_4 = M01M01$...

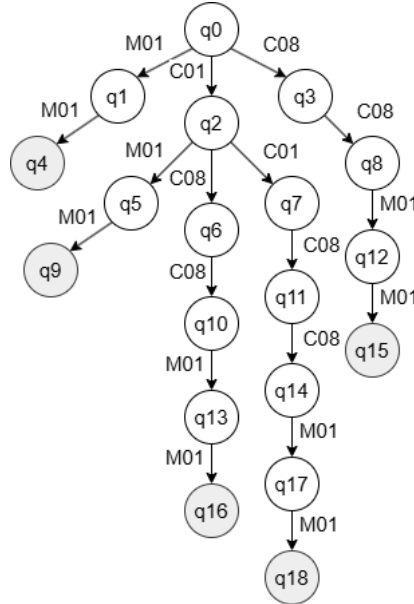


Figura 4.6: PTA que representa la estructura secuencial de la actividad **Act15** (Sacar la basura).

Aunque es posible trabajar directamente con los PTA resultantes, estos presentan una limitación fundamental: carecen de capacidad de generalización. Es decir, el PTA memoriza de forma literal las secuencias observadas, sin abstraer patrones o regularidades comunes. Esto implica que cualquier variación no registrada explícitamente será rechazada, aunque conceptualmente forme parte del mismo comportamiento.

Por ejemplo, supóngase que en la actividad X se han registrado las siguientes dos secuencias de sensores:

$$[D1, D1, P3] \quad \text{y} \quad [D1, D1, D1, D1, P3]$$

Ambas secuencias reflejan que el sensor D1 se activa un número variable de veces antes de pasar a P3. Sin embargo, el PTA tratará estas secuencias como ramas completamente distintas del árbol, ya que no puede inferir una regla general como “el sensor D1 se repite entre 2 y 4 veces antes de P3”.

Para superar esta limitación, se recurre a técnicas del campo del aprendizaje de lenguajes formales. Estas técnicas permiten generalizar las secuencias observadas mediante la construcción de autómatas finitos consistentes con las observaciones, pero capaces de aceptar también variantes estructurales plausibles no vistas durante el entrenamiento. En particular, se usará el algoritmo RPNI descrito en la sección 3.3.

4.3. Aprendizaje de AFD a partir de ejemplos positivos y negativos

En este apartado se presenta la aplicación del algoritmo RPNI en el contexto de una actividad real. Se describirá el proceso de construcción del autómata a partir de las secuencias recogidas, así como los resultados obtenidos tras la aplicación del algoritmo. A continuación, se analizarán en profundidad diversos casos problemáticos que surgieron durante la implementación del mismo.

Es relevante señalar que, en el contexto del presente trabajo, las muestras positivas corresponden a los nodos terminales asociados a cada secuencia aceptada, mientras que las muestras negativas se han definido como el conjunto restante de nodos pertenecientes al árbol de prefijos.

4.3.1. Ejemplo ilustrativo de la aplicación del algoritmo RPNI

Consideremos como punto de partida el autómata representado en la Figura 4.6. Se tienen los nodos:

$$\{q_0, q_1, q_2, q_3, q_4, q_5, q_6, q_7, \dots, q_{18}\}.$$

A partir de dicha estructura, se identifican como ejemplos positivos los siguientes nodos:

$$\{q_4, q_9, q_{15}, q_{16}, q_{18}\}$$

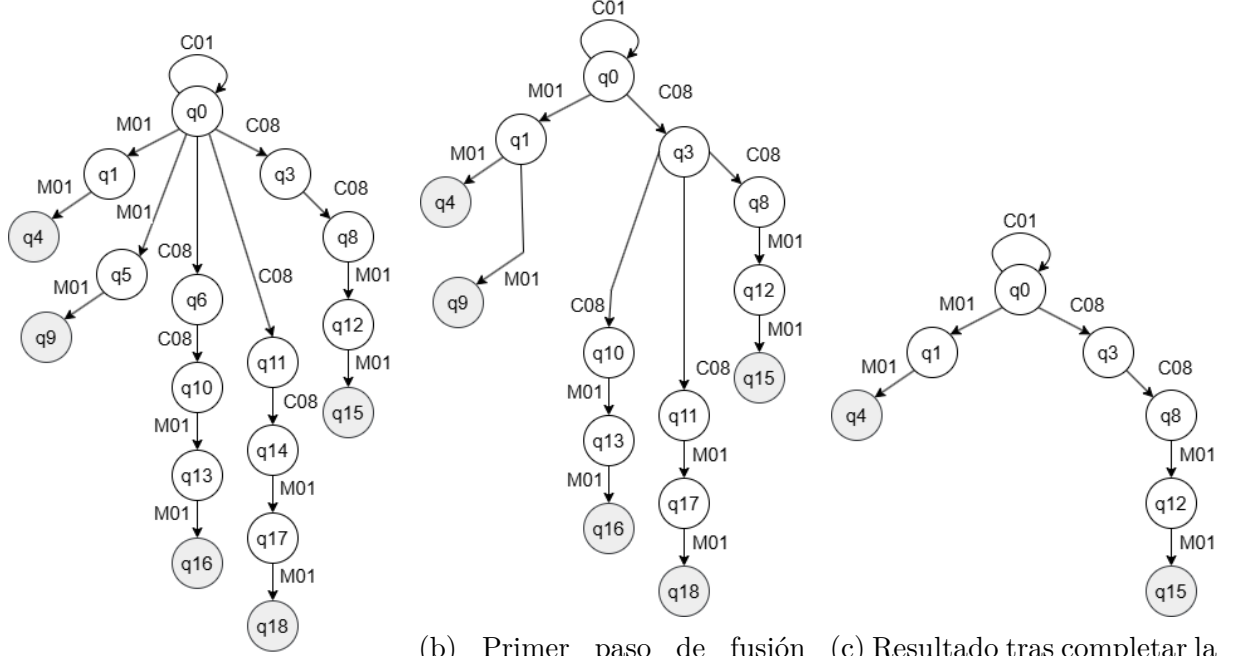
mientras que los nodos restantes corresponden a ejemplos negativos.

En este contexto, se procede a evaluar la posibilidad de unificar los nodos q_0 y q_1 , siguiendo el procedimiento que se detalla a continuación:

- En primer lugar, se introduce una transición desde el nodo q_0 hacia sí mismo utilizando como símbolo de entrada el que conecta q_0 con q_1 . Al efectuar esta fusión, se observa que ambos nodos conducen a descendientes mediante el símbolo C08. Por lo tanto, para preservar la coherencia estructural, también deben integrarse los nodos que se alcanzan mediante dicho símbolo, en este caso q_3 y q_6 . Se realizará el mismo proceso con el símbolo C01.

- A continuación, se verifica si el autómata modificado acepta alguno de los ejemplos clasificados como negativos. Al no aceptar ningún ejemplo negativo, se adopta el nuevo autómata como nueva base para futuras combinaciones.

Las distintas fases de este procedimiento se ilustran en la Figura 4.7.



(a) Autómata resultante de la combinación inicial de los estados q_0 y q_1 .

(b) Primer paso de fusión de nodos accesibles mediante transiciones con el mismo símbolo.

(c) Resultado tras completar la fusión de todos los nodos accesibles mediante el mismo símbolo.

Figura 4.7: Evolución del autómata correspondiente a la actividad **Act15** durante las primeras etapas del proceso de generalización mediante combinaciones sucesivas de estados.

Como resultado de las operaciones anteriores, la nueva configuración de nodos es la siguiente:

$$\{q_0, q_1, q_3, q_4, q_8, q_{12}, q_{15}\}.$$

Se intenta entonces combinar los nodos q_0 y q_1 . Sin embargo, al agregar una transición que conecta el nodo inicial consigo mismo mediante el símbolo $M01$, también es necesario unificar los nodos q_0 y q_4 . Esta operación da lugar a la aceptación de secuencias no deseadas como λ , $C01$ o $M01$, clasificadas previamente como negativas.

Continuando con el proceso, se analiza la factibilidad de fusionar los nodos q_0 y q_3 . Para ello, se sigue la misma metodología:

- Se incorpora una transición desde el nodo q_0 hacia sí mismo mediante el símbolo correspondiente ($C08$). Posteriormente, se integran los nodos descendientes que se alcanzan utilizando el mismo símbolo, en este caso q_0 y q_8 .
- Se comprueba nuevamente si la estructura resultante acepta algún ejemplo negativo. Dado que esto no ocurre, se valida la combinación, y el autómata actualizado se considera válido para las siguientes iteraciones.

La evolución de esta etapa puede visualizarse en la Figura 4.8.

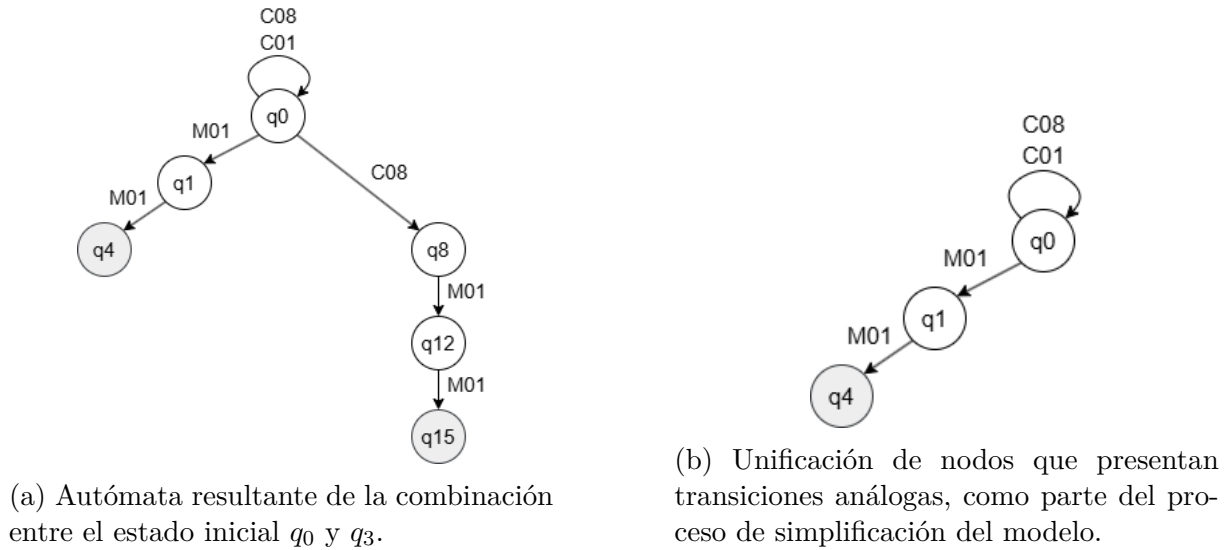


Figura 4.8: Progresión del modelo de autómata correspondiente a la actividad **Act15**, tras la segunda serie de combinaciones de estados y unificaciones estructurales.

En la siguiente etapa, los nodos vigentes son:

$$\{q_0, q_1, q_4\}.$$

Se procede a intentar la combinación de los nodos q_0 y q_1 . No obstante, ocurriría lo mismo que se ha explicado anteriormente cuando se han intentado combinar estos mismos nodos.

Esta misma situación se repite al intentar otras fusiones posibles. En consecuencia, se concluye que el autómata obtenido representa la versión mínima válida. Su estructura definitiva se presenta en la Figura 4.8b.

4.3.2. Casos problemáticos durante la fusión de nodos

Puede presentarse un caso particular en el que se solicita la combinación de un nodo consigo mismo. Este escenario, aunque inusual, puede surgir, por ejemplo, cuando existen múltiples auto-aristas (es decir, aristas desde el nodo hacia sí mismo) que comparten un mismo símbolo o etiqueta.

En tales circunstancias, es necesario consolidar dichas aristas para evitar redundancias. El procedimiento consiste en contar la cantidad de aristas salientes asociadas a cada símbolo. Si se detecta que existen dos o más aristas con la misma etiqueta, se selecciona una de ellas para conservar, eliminando las demás. El peso total de las aristas eliminadas se acumula y se transfiere a la arista que permanece, de modo que no se pierda la información cuantitativa originalmente representada.

Este tratamiento garantiza que la estructura del autómata se mantenga simplificada y libre de ambigüedades, preservando al mismo tiempo la información relevante para el análisis posterior.

Por otra parte, en el contexto del procesamiento y transformación de autómatas, es esencial trabajar con copias estructurales de los grafos que los representan. Esta práctica

responde a la necesidad de preservar la integridad del modelo original durante operaciones que pueden alterar de manera significativa su topología, como la eliminación o combinación de nodos, la modificación de aristas o la reestructuración de rutas de transición.

Al realizar este tipo de modificaciones, especialmente cuando implican cambios destructivos, resulta inconveniente operar directamente sobre el autómata original. Cualquier alteración durante el recorrido del grafo podría afectar el flujo lógico del análisis, provocar la pérdida de información, o incluso derivar en errores de ejecución difíciles de detectar. Por ello, se recurre a la creación de copias intermedias del autómata, sobre las cuales se aplican los cambios necesarios de forma controlada.

4.4. Conversión de un AFD a una expresión regular

Para obtener una expresión regular que describa el lenguaje aceptado por un AFD, se ha implementado un procedimiento basado en el método de **eliminación de estados intermedios**¹. Este enfoque transforma gradualmente el autómata en una representación equivalente mediante expresiones regulares, hasta que finalmente se obtiene una única expresión que describe el lenguaje aceptado.

El procedimiento comienza con una transformación estructural del autómata. Para unificar el punto de partida y facilitar la representación de múltiples estados finales, se introducen dos nuevos estados auxiliares: uno como **nuevo estado inicial**, conectado sin consumir símbolos al estado inicial original del autómata, y otro como **nuevo estado final**, al que se conectan igualmente sin consumir símbolos todos los estados finales originales. Esta transformación garantiza que el autómata modificado tenga un único estado inicial y un único estado final, condición necesaria para aplicar de forma ordenada el método de eliminación.

A continuación, se procede a **eliminar de manera iterativa todos los estados intermedios** (es decir, aquellos que no son ni el nuevo estado inicial ni el nuevo estado final). Por cada estado que se elimina, se analizan:

- Las transiciones entrantes al estado eliminado.
- Las transiciones salientes desde ese estado.
- Cualquier bucle sobre el estado, que se incorpora como una repetición indefinida (estrella de Kleene).

Estas transiciones se sustituyen por nuevas transiciones directas entre los estados conectados, utilizando expresiones regulares que las combinan.

Este proceso se repite hasta que solo quedan el nuevo estado inicial y el nuevo estado final, conectados mediante transiciones etiquetadas con expresiones regulares que representan todas las posibles secuencias de símbolos que antes pasaban por los estados eliminados.

Finalmente, se extrae la o las expresiones regulares que conectan directamente el estado inicial artificial con el estado final artificial. Si existen múltiples transiciones, se combinan

¹CS103 - Stanford University. Guide to state elimination. https://web.stanford.edu/class/cs103/guide_to_state_elimination, 2024.

mediante la operación de unión. Como paso adicional, se realiza una limpieza para eliminar símbolos vacíos redundantes que no aportan información significativa a la expresión final.

Este procedimiento permite obtener una expresión regular equivalente al lenguaje aceptado por el AFD original, incluso en presencia de múltiples estados de aceptación. El resultado es una representación compacta y formalmente correcta del lenguaje que acepta el autómata.

4.5. Predicción de secuencias de actividades mediante sensores

El proceso de predicción de actividades se inicia con la obtención de las secuencias de sensores extraídas de los archivos `YYYY-MM-DD-X-sensors.csv`. A partir de estos datos, se construye un ciclo iterativo en el cual se diferencian dos secuencias distintas: una que contiene sensores de movimiento (SM) y otra que los excluye. Una vez generadas ambas secuencias, es necesario transformarlas en cadenas de caracteres con el fin de facilitar su procesamiento. Por ejemplo, una secuencia del tipo `[M01]`, `[M01]`, `[SM3]` será convertida a la cadena `M01M01SM3`, lo cual permite aplicar posteriormente la función `re.search` para la detección de patrones. La necesidad de emplear dos secuencias de patrones radica en que los sensores de movimiento se utilizan únicamente cuando resultan imprescindibles, otorgándose mayor prioridad a la secuencia que prescinde de ellos.

El patrón que se proporciona como parámetro a la función de búsqueda dependerá de la actividad que se desea predecir. Para determinar esta actividad, se selecciona un conjunto de posibles actividades candidatas en función de la actividad realizada previamente, apoyándose para ello en los autómatas construidos en la Sección 4.1.

El orden de las actividades dentro de la lista dependerá del peso de las aristas en el autómata, es decir, la actividad con mayor probabilidad de ocurrencia será posicionada en primer lugar. Además, se han definido ciertas actividades que pueden suceder en cualquier momento, independientemente del contexto anterior. Estas actividades son: **Act11** (Jugar a videojuegos), **Act09** (Ver la televisión), **Act14** (Recibir visita en el laboratorio), **Act18** (Usar el baño) y **Act12** (Relajarse en el sofá).

Es preciso considerar ciertas excepciones en la selección de actividades. La primera ocurre cuando no se ha registrado ninguna actividad previa; en tal caso, se considerarán los nodos iniciales del autómata como posibles actividades iniciales. La segunda excepción se presenta cuando la actividad previa pertenece al conjunto de actividades posibles en cualquier momento; en ese escenario, las actividades candidatas permanecerán inalteradas respecto al conjunto previamente asignado.

Una vez definidos los dos parámetros de la función `re.search` (cadena de caracteres y patrón de la expresión), se contemplan los siguientes escenarios:

- **Coincidencia detectada:** cuando se identifica una coincidencia entre la secuencia de sensores registrada y el patrón asociado a una actividad específica, dicha actividad se registra junto con las marcas temporales correspondientes al primer y al último sensor involucrado, los cuales se consideran como los tiempos de inicio y finalización, respectivamente. Posteriormente, el proceso de búsqueda de coincidencias se reanuda a partir del último sensor implicado en la detección anterior.

- **Sin coincidencia:** si no se encuentra coincidencia con el patrón de la actividad actual, se procede a evaluar la siguiente actividad en la lista. En caso de que ninguna actividad genere una coincidencia, se elimina el primer sensor de ambas secuencias y se repite el proceso.

El ciclo finalizará una vez que no queden sensores disponibles en las secuencias procesadas. Como resultado, se devolverá una lista de triplas, cada una representando una actividad detectada junto con su tiempo de inicio y fin: (actividad, tiempo_inicio, tiempo_fin).

Capítulo 5

Evaluación de resultados y análisis de rendimiento

Una vez obtenida la lista de triplas mencionada en la Sección 4.5, se procede a completar los archivos `results.csv` o `time-slots training.csv`, en función de si se están utilizando los archivos de test o de entrenamiento. El proceso consiste en asignar los valores `TRUE` o `FALSE` en la columna correspondiente a cada actividad, dependiendo de si la fecha y hora se encuentran dentro del intervalo temporal de dicha actividad.

Cabe destacar que se ha realizado una aproximación temporal con una granularidad de 30 segundos. Esta decisión responde a la necesidad de capturar correctamente aquellas actividades que pueden ser identificadas por un único evento de sensor, el cual podría producirse en instantes distintos a los múltiplos exactos de minuto o medio minuto. Sin dicha granularidad, dichos eventos podrían pasar desapercibidos, afectando negativamente a la precisión de la predicción.

En la Figura 5.1 se muestra un fragmento del archivo `time-slot training` ya completado.

2017-10-31-A-activity.csv								
Timestamp	Act24	Act18	Act16	Act02	Act05	Act17	Act22	Act13
31/10/2017 11:11	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE
31/10/2017 11:12	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE
31/10/2017 11:12	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE
31/10/2017 11:13	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE
31/10/2017 11:13	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE
31/10/2017 11:14	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE
31/10/2017 11:14	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE
31/10/2017 11:15	TRUE	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE

Figura 5.1: Fragmento de la tabla `time-slot training` con las actividades predecidas, que muestra los intervalos temporales segmentados y las etiquetas asignadas por el modelo de reconocimiento.

Para facilitar la comparación entre los datos reales y los predichos, se transformaron ambas tablas de modo que cada fila contiene únicamente dos columnas: una con la fecha y hora, y otra con la actividad realizada en ese momento. Esta simplificación permite visualizar de forma clara las secuencias de actividades detectadas. En la Figura 5.2 se muestran las nuevas representaciones para la tabla original y la predecida.

2017-10-31-A-activity.csv	
Inicio	
31/10/2017 11:11	Ninguna
31/10/2017 11:12	Ninguna
31/10/2017 11:12	Act24
31/10/2017 11:13	Act24
31/10/2017 11:13	Act24
31/10/2017 11:14	Act24
31/10/2017 11:14	Act24
31/10/2017 11:15	Act24
31/10/2017 11:15	Act18
31/10/2017 11:16	Act18
31/10/2017 11:16	Act18
31/10/2017 11:17	Act16
31/10/2017 11:17	Act16
31/10/2017 11:18	Act16
31/10/2017 11:18	Act02

(a) Fragmento de la tabla correspondiente a la secuencia original de **time-slot training**, con las actividades reales registradas.

2017-10-31-A-activity.csv	
Inicio	
31/10/2017 11:11	Ninguna
31/10/2017 11:12	Ninguna
31/10/2017 11:12	Ninguna
31/10/2017 11:13	Ninguna
31/10/2017 11:13	Ninguna
31/10/2017 11:14	Ninguna
31/10/2017 11:14	Ninguna
31/10/2017 11:15	Act24
31/10/2017 11:15	Ninguna
31/10/2017 11:16	Act18
31/10/2017 11:16	Act18
31/10/2017 11:17	Ninguna
31/10/2017 11:17	Act16
31/10/2017 11:18	Act16
31/10/2017 11:18	Act02

(b) Fragmento de la tabla correspondiente a la secuencia predecida de **time-slot training**, con las etiquetas generadas por el modelo.

Figura 5.2: Transformación de la representación temporal de actividades mediante la comparación entre la secuencia original y la secuencia predecida por el modelo.

A partir de estas secuencias, se planteó la evaluación de la predicción desde dos perspectivas: la precisión temporal (es decir, si las actividades fueron predichas correctamente en los intervalos temporales correspondientes) y la precisión en la secuencia de actividades, independientemente del tiempo.

Para la evaluación temporal, se compararon las tablas generadas, calculando tanto el número total de actividades como el número de coincidencias exactas, lo que permitió obtener un porcentaje de acierto de 59.6 % para los **datos de entrenamiento** (655 actividades correctamente predecidas sobre 1099). Se valora este porcentaje como inferior a lo esperado, lo que pone de manifiesto dificultades en la estimación precisa de los intervalos de inicio y fin de las actividades. Esta problemática se ilustra claramente en la Figura 5.2.

Por otro lado, para analizar la precisión en la predicción de las secuencias de actividades, se recurrió a medidas de distancia entre listas, incluyendo la distancia de Levenshtein y su variante ponderada (considerando la longitud de cada secuencia), así como sus respectivas medias globales. A continuación se detallan las medidas utilizadas.

- La distancia de Levenshtein entre dos listas a y b , de longitudes $|a|$ y $|b|$ respectivamente, se define como el número mínimo de operaciones necesarias para transformar a en b , donde las operaciones permitidas son inserción, eliminación y sustitución de un elemento.
- Para calcular la distancia ponderada, se divide la distancia de Levenshtein por el número de actividades en la secuencia de actividades real.

En la Tabla 5.1 se pueden explorar los resultados obtenidos al evaluar sobre los **datos de entrenamiento** (segunda columna). Estos resultados indican que, en promedio, se cometen entre uno y dos errores por secuencia (una distancia total de 32 diferencias para 21 secuencias de actividades). Además, menos de una quinta parte del contenido de las secuencias predichas resulta incorrecto. Sin embargo, cabe señalar que existen numerosas secuencias completamente correctas, aunque también otras con un mayor número de errores.

Tabla 5.1: Resumen de métricas obtenidas al comparar las actividades predecidas con los datos de entrenamiento y test

Métrica	Datos entrenamiento	Datos test
Distancia media	1.524	5.555
Distancia total	32	50
Media distancias normalizadas	0.181	0.626

En la misma tabla se presentan también los resultados obtenidos sobre los **datos de test** (tercera columna). Los valores obtenidos son considerablemente peores que los de la fase de entrenamiento, lo cual evidencia un problema de sobreajuste. Esto es coherente con el hecho de que las expresiones regulares empleadas fueron generadas a partir de los datos de entrenamiento.

Cabe destacar que el número de aciertos en cuanto a la precisión temporal para los **datos de test** ha bajado también mucho, llegando a predecir bien solo 131 actividades del total de 535 (un porcentaje de 24.485 %). Recordamos que en los datos de entrenamiento, el porcentaje de acierto es de 59.6 %.

Con el fin de explorar posibles mejoras, se sustituyeron las expresiones regulares originales por aquellas utilizadas en [4], que se pueden encontrar en la Tabla 5.2.

Tabla 5.2: Expresiones regulares obtenidas de la publicación [4].

Sin sensores SM			
Act01	$D_{04}^+(C_{01} C_{05} D_{04} D_{05})^*$	Act15	$(C_{01} C_{08})^*M_{01}^+$
Act02	$(D_{01} D_{02} D_{04} D_{10} H_{01})^+$	Act16	C_{09}^+
Act03	$(C_{04} D_{01} D_{02} D_{04} D_{08} D_{10})^+$	Act17	C_{09}^+
Act04	$(C_{04} D_{01} D_{02} D_{04} D_{08} D_{10})^+$	Act18	$(C_{10} D_{07})^+(C_{08} C_{10} D_{07})^*$
Act08	$(C_{02} D_{10})^+$	Act19	D_{05}^+
Act09	$(TV_0 S_{09})^*TV_0$	Act20	$D_{09}(C_{12} D_{09})^*$
Act10	M_{01}^+	Act22	$D_{03}(C_{12} C_{13} D_{03})^*$
Act11	$(TV_0C_{07} C_{07}TV_0)S_{09}^*(TV_0C_{07} C_{07}TV_0)$	Act23	$C_{14}(C_{13} C_{14})^+$
Act13	M_{01}^+	Act24	C_{14}^+
Act14	M_{01}^+		
Con sensores SM			
Act05	SM_1^+	Act12	$(S_{09} SM_4 SM_5)^*SM_5(S_{09} SM_4 SM_5)^*$
Act06	SM_1^+	Act21	SM_4^+
Act07	SM_1^+		

Estas expresiones son notablemente más simples y menos restrictivas, al incorporar operadores como + que permiten una mayor flexibilidad. En contraste, las expresiones generadas automáticamente son más rígidas. Otra vez, se evalúa tanto la precisión temporal como la precisión en la secuencia de actividades.

De un total de 535 actividades, se lograron predecir correctamente 320, lo que equivale a una tasa de acierto del 59,813 % en los **datos de test**, valor considerablemente superior al obtenido mediante expresiones regulares generadas automáticamente (24,485 %). En cuanto a los **datos de entrenamiento**, se observa una leve disminución en el rendimiento: se pasa de 655 a 620 actividades correctamente clasificadas (sobre un total de 1099), lo que representa una reducción aproximada de 3 puntos porcentuales (del 59,6 % al 56,415 %). Este comportamiento sugiere que ambas estrategias presentan un desempeño similar sobre los datos utilizados durante el entrenamiento. Cabe destacar que, con estas expresiones regulares, la tasa de acierto en los datos de test (59,813 %) supera a la obtenida en los datos de entrenamiento (56,415 %).

En lo que respecta a las distancias entre secuencias de actividades (la métrica que no considera los tiempos de inicio y fin de cada actividad), la Tabla 5.3 presenta los resultados obtenidos al aplicar las expresiones regulares tanto sobre los **datos de entrenamiento** (segunda columna) como sobre los **datos de test**. Se observa una ligera mejora en el conjunto de entrenamiento, con una reducción en la distancia de 1.524 a 1.381, y una mejora notable en el conjunto de test, donde la distancia disminuye significativamente de 5.555 a 1.777.

Tabla 5.3: Resumen de métricas obtenidas al aplicar expresiones regulares externas sobre los datos.

Métrica	Datos entrenamiento	Datos test
Distancia media	1.381	1.777
Distancia total	29	16
Media distancias normalizadas	0.164	0.199

Tal como se adelantaba en la introducción, los autores de [4] alcanzaron una precisión en los datos de test cercana al 90 %, mientras que en el presente trabajo dicha precisión se aproxima al 60 %. Esta diferencia se debe a que el sistema propuesto por dichos autores incorpora una estrategia de *backtracking*, la cual evita que una única expresión regular consuma todos los sensores disponibles, permitiendo así la detección de patrones alternativos dentro de una misma secuencia. Esta característica contribuye a los buenos resultados obtenidos en la identificación de la secuencia completa de actividades. Adicionalmente, el elevado porcentaje de actividades correctamente predichas podría explicarse por el uso de información proveniente del suelo inteligente, el cual facilita una detección más precisa de los tiempos de inicio y fin de las actividades.

Capítulo 6

Conclusiones y líneas de trabajo futuras

Este trabajo se ha centrado en la predicción de actividades humanas a partir de secuencias de eventos generadas por sensores, utilizando patrones definidos mediante expresiones regulares. A lo largo del desarrollo se ha abordado tanto la construcción del modelo como su evaluación detallada en distintas fases.

Una herramienta especialmente valiosa en el desarrollo de este trabajo ha sido la generación de autómatas destinados a representar de forma estructurada las posibles secuencias de actividades. Estos modelos permiten visualizar de manera clara y precisa las transiciones potenciales entre distintas actividades, proporcionando así una comprensión más profunda del comportamiento secuencial observado en los datos. La estructura del autómata facilita no solo la validación de los patrones definidos, sino también la anticipación de las actividades que podrían ocurrir a continuación en función del estado actual.

Durante la fase de evaluación, se analizaron los resultados desde dos perspectivas complementarias: la precisión temporal y la fidelidad en la secuencia de actividades. Esta doble aproximación ha permitido obtener una visión más completa del desempeño del sistema. Aunque en algunos casos se han logrado secuencias completas correctas, la predicción exacta de los intervalos temporales de inicio y fin de cada actividad se ha mostrado como una de las principales fuentes de error. La dificultad para alinear correctamente los límites temporales ha afectado negativamente a la precisión temporal.

Los resultados obtenidos con las expresiones regulares generadas automáticamente mostraron un comportamiento razonable sobre los datos de entrenamiento, pero su rendimiento se deterioró notablemente al aplicarlas sobre los datos de test. Este fenómeno puso de manifiesto un problema de sobreajuste, ya que las expresiones derivadas a partir del conjunto de entrenamiento eran demasiado específicas. En contraste, al incorporar las expresiones regulares utilizadas por los ganadores del concurso (más simples, generales y con el uso extensivo de operadores como $+$) se observó una mejora significativa en los resultados sobre los datos de test, pasando de un 27.29 % a un 62.80 % en precisión de actividades correctamente identificadas, y reduciendo notablemente la distancia media entre secuencias predichas y reales.

En conjunto, los resultados obtenidos permiten concluir que el uso de expresiones regulares constituye una estrategia válida y efectiva para el reconocimiento de actividades, especialmente cuando se apoya en un diseño cuidadoso de patrones y en herramientas complementarias como los autómatas de estados. Estos últimos, al representar las posibles

secuencias de actividades, aportan una visión estructurada del comportamiento esperado y potencian tanto la validación como la interpretación de los modelos. A pesar de los avances, la generalización continúa siendo un reto central. En este sentido, el uso de técnicas más flexibles o el refinamiento de los enfoques actuales podría suponer un avance significativo.

Como líneas futuras de trabajo, se identifican varias direcciones con potencial para mejorar los resultados obtenidos. En primer lugar, se propone profundizar en la mejora de la inferencia temporal mediante la incorporación de datos contextuales adicionales, tales como sensores de presión en el suelo o información de proximidad. La combinación de estos datos, junto con el uso de filtros adecuados, podría permitir una estimación más precisa de los tiempos de inicio y finalización de las actividades. Adicionalmente, el desarrollo de mapas espaciales que indiquen las zonas del hogar donde se realizan determinadas actividades podría complementar este análisis, aunque su generación automática plantea desafíos significativos.

Por otra parte, resulta pertinente explorar métodos que permitan la generación automática de expresiones regulares más generales, con el objetivo de sustituir aquellas obtenidas manualmente. Este enfoque podría contribuir a mitigar el sobreajuste observado y favorecer una mejor capacidad de generalización del sistema ante situaciones no vistas durante el entrenamiento.

Finalmente, se plantea la implementación o integración de motores de inferencia más avanzados, capaces de incorporar técnicas de segmentación dinámica. Este tipo de sistemas ofrecería una mayor flexibilidad al permitir la identificación simultánea de múltiples patrones dentro de una misma ventana temporal, aumentando así la robustez del reconocimiento de actividades.

Bibliografía

- [1] Macarena Espinilla, Javier Medina, and Chris Nugent. Ucam cup. analyzing the uja human activity recognition dataset of activities of daily living. In *Proceedings*, volume 2, page 1267. MDPI, 2018.
- [2] John E. Hopcroft, Rajeev Motwani, and Jeffrey D. Ullman. Introduction to automata theory, languages, and computation, 2nd edition. *SIGACT News*, 32(1):60–65, March 2001.
- [3] J Oncina and P Gracia. A polynomial algorithm to infer regular languages., 1992.
- [4] Sergio Salomón and Cristina Tîrnăucă. Human activity recognition through weighted finite automata. In *Proceedings*, volume 2, page 1263. MDPI, 2018.