DESARROLLO DE SISTEMAS INTELIGENTES

PRÁCTICAS - FASE 2

Desarrollo del motor de inferencia en Drools

Curso Académico 2025/26 - Convocatoria de Enero

Nintendo DSI(nt)

Juan Alejandro González Ballesta - G1.3

Eduardo Terry Gavilá - G1.3

Alba García Camacho - G1.3

Profesores:

Ricardo Javier Sendra Lázaro

Aurora González Vidal

Índice de contenidos.

1. Introducción.

El objetivo de la Fase 2 del sistema basado en conocimiento consiste en extraer el conocimiento necesario de las fuentes y establecer relaciones entre los términos, con el fin de inferir nuevos términos y conexiones. Este conocimiento se modela mediante reglas, las cuales se emplearán en el motor de inferencia Drools para diagnosticar enfermedades cardíacas.

2. Traducción desde la ontología al modelado de objetos.

2.1. Traducción de ondas, intervalos y segmentos.

En la ontología, los conceptos de onda, intervalo y segmento se definieron como clases generales con distintas especializaciones (*OndaP*, *OndaQ*, *OndaR*, …). Para trasladar esta estructura al modelado de objetos en Java, se han definido *Onda*, *Intervalo* y *Segmento* como clases abstractas con sus respectivas subclases concretas, manteniendo de esta manera la jerarquía y las propiedades esenciales.

Onda → clase abstracta que define los atributos *tInicio*, *tFin* y *picoMaximo*.
Subclases: ***OndaP***, ***OndaQ***, ***OndaR***, ***OndaS*** y ***OndaT***.
Cada subclase se utilizará para definir un tipo concreto de *Intervalo* o *Segmento* y, además, se hará uso de ella para aplicar las distintas reglas definidas dentro de Drools.

Segmento → clase abstracta que define los atributos *ondaInicio* y *ondaFin*, de los que depende para calcular su duración, y el atributo *ciclo*.
Subclases: ***SegmentoPR***, ***SegmentoST***

Intervalo → clase abstracta que define los atributos *ondaInicio* y *ondaFin*, de los que depende para calcular su duración, y el atributo *ciclo*.
Subclases: ***IntervaloPR***, ***IntervaloQT***

Con este enfoque, se permite mantener la modularidad y se facilita la aplicación de reglas específicas para cada tipo de entidad manteniendo la coherencia con la ontología definida.

2.2. Traducción de diagnósticos.

Respecto a los diagnósticos, valoramos si seguir la misma idea que con las ondas (clase abstracta *Diagnostico* y una subclase por diagnóstico concreto). Sin embargo, consideramos crear una única clase con un atributo *tipo* ya que los diagnósticos concretos no se usan para inferir nuevo conocimiento.

2.3. Traducción de ciclo cardíaco.

En la ontología inicial contábamos con una clase *CicloCardiaco* que se encarga de agrupar los elementos de cada ciclo (onda P, intervalo PR e intervalo QT). Sin embargo, para la inferencia no vamos a necesitar el número de ciclo más allá de mostrar cuándo se detecta la anomalía cardíaca y hacer comparaciones. Por ello, en las clases Java se transformó en un atributo *ciclo*. La ontología fue modificada en consecuencia.

3. Motor de inferencia.

Las reglas se han definido utilizando tres archivos Drools pertenecientes al mismo paquete, cada uno en su correspondiente agenda con la finalidad de separar el proceso de inferencia en los siguientes pasos:

1. Asignar ciclos a ondas → 2. Calcular elementos a partir de ondas → 3. Inferir diagnósticos

De esta manera, los tres archivos mencionados trabajan de forma complementaria: el primero prepara las ondas para crear segmentos, intervalos y complejos y, el último fichero, emplea todos esos elementos para diagnosticar enfermedades.

3.1. Fichero *AsignarCiclos.drl.*

Este primer archivo permite procesar cada una de las ondas incluidas en el electrocardiograma dado como entrada, con el fin de asignar a cada una el ciclo al que pertenece. Esto permite al fichero *Electrocardiograma.drl* inferir los segmentos, intervalos y el complejo de los ciclos cardíacos adecuadamente.

Para determinar el ciclo cardíaco al que pertenece cada onda, el sistema hace uso de reglas específicas para cada onda (P, Q, R, S y T), aplicando un orden temporal y lógico basado en la secuencia dada en el fichero de entrada.

Cada regla comprueba primero si la Onda ha sido o no procesada. La regla definida para la Onda P se ejecuta antes que el resto, de manera que primero se procesan todas las ondas P marcando cada una el inicio de un nuevo ciclo cardíaco (incrementando el número de ciclo cada vez que se ejecuta la regla). El resto de reglas, irán procesando las ondas en base a su respectiva anterior. La regla para la Onda Q asigna a la onda Q el mismo ciclo que la última onda P procesada que ocurre justo antes de ella en el tiempo, garantizando que no exista otra onda P entre la P y la Q seleccionadas. Lo mismo ocurre para el resto de reglas de procesamiento de las ondas.

3.2. Fichero *Electrocardiograma.drl*.

En el segundo archivo, se infieren los segmentos, intervalos y el complejo de los ciclos cardíacos a partir de las ondas obtenidas desde el fichero de entrada.

Para inferir un intervalo PR, se comprueba que exista una onda P y una onda Q que pertenezcan al mismo ciclo cardíaco. De forma análoga, el intervalo QT se determina a partir de las ondas Q y T del mismo ciclo.

```
rule "CrearIntervaloPR"
salience 10
    when
        $p: OndaP()
        $q: OndaQ(ciclo == $p.ciclo)
```

```
        not(IntervaloPR(ondaInicio == $p, ondaFin == $q))
    then
        insert(new IntervaloPR($p, $q));
end
```

Por ejemplo, para crear un intervalo PR, verificamos que exista una onda P y una onda Q que pertenezcan al mismo ciclo. Para evitar reevaluaciones de la regla para las mismas ondas, añadimos la cláusula `not`. La parte de acción de la regla crea el intervalo a partir de las ondas y lo inserta en la base de hechos.

El resto de elementos se crean de manera análoga:

- Segmento PR: ondas P y Q del mismo ciclo.
- Segmento ST: ondas S y T del mismo ciclo.
- Complejo QRS: ondas Q y S pertenecientes al mismo ciclo. Podríamos haber incluido también la onda R como antecedente, pero por la morfología del ciclo, si tenemos una onda Q y S, existe una onda R entre esas dos.

En este fichero también se calcula el *heartRate* a partir de la diferencia entre la última onda R y la primera. Esta regla tiene prioridad 10 para que se ejecute antes que la de creación de segmentos, intervalo y complejos.

```
rule "CalcularHeartRateRR"
    salience 10
when
    $primer: OndaR()
    not (OndaR(tInicio < $primer.tInicio))

    $ultimo: OndaR()
    not (OndaR(tInicio > $ultimo.tInicio))

    not (HeartRate())
then
    int numCiclos = $ultimo.getCiclo();
    int duracionMs = $ultimo.gettInicio() - $primer.gettInicio();

    int hr = 60000 * numCiclos / duracionMs;
    HeartRate heartRate = new HeartRate(hr);
    insert(heartRate);

    salida.escribir("HeartRate: " + heartRate.getValor() + " pul/min");
```

```
end
```

Para calcular el *heartRate* seleccionamos dos ondas R: una cuyo tiempo de inicio verifique que no existe otra regla con un tiempo de inicio menor y, otra, que verifique que no existe una onda P con un tiempo de inicio mayor. También comprobamos que no existe un hecho HeartRate para evitar reevaluaciones. Esta regla crea el hecho de *heartRate* siguiendo la expresión:

$$\frac{60000\ ms/min \cdot ciclos}{ultima.tInicio - primera.tInicio\ (ms)} = x\ pul/min$$

La similitud y simplicidad de todas estas inferencias se debe a que este primer archivo tiene como objetivo extraer información básica y siempre presente en los ciclos cardíacos, correspondiente a duraciones entre ondas consecutivas. Esta información resulta relevante para el análisis y diagnóstico de posibles enfermedades cardíacas.

3.3. Fichero *InferirDiagnostico.drl*.

En el tercer archivo, se definen las reglas destinadas a inferir posibles enfermedades cardíacas del paciente. Más adelante, justificaremos la inferencia de cada diagnóstico.

```
rule "Hipocalcemia"
when
     $i: IntervaloQT( $duracion : getDuracion(), $duracion >= 440 )
     not(Diagnostico(tipo == TipoDiagnostico.HIPOCALCEMIA))
then
    Diagnostico d = new Diagnostico(
        TipoDiagnostico.HIPOCALCEMIA,
        "intervalo QT de duración superior a 440 ms (" + $duracion + " ms)
en el ciclo " + $i.getCiclo()
    );
    insert(d);
    salida.escribir("ECG_Pattern: " +d.toString());
end
```

Todas las reglas de diagnóstico de enfermedades siguen el mismo patrón: buscan algún elemento que cumpla cierta condición (por ejemplo, para la hipocalcemia buscamos un

intervalo QT cuya duración sea superior a 440 ms) y verifica que no exista un diagnóstico del mismo tipo para evitar volver a diagnosticar la misma enfermedad.

4. Manual de usuario.

Este manual describe el procedimiento para ejecutar el sistema basado en conocimiento de Drools. Para simular una ejecución completa del sistema es esencial tener el proyecto abierto en Eclipse y Drools correctamente instalado. El sistema se inicia estableciendo argumentos para definir los directorios de trabajo. Para establecer dichos argumentos hay que acceder a `Run → Run Configurations` y una vez ahí establecerlos en el apartado "`Arguments`" del Lanzador. Las opciones de ejecución son las siguientes:

- Ejecución con dos argumentos:
    - Argumento 1: Corresponde al directorio de entrada. Este directorio debe contener todos los ficheros que se van a evaluar.
    - Argumento 2: Corresponde al directorio de salida. En este directorio se generarán y almacenarán los ficheros de diagnóstico.

- Ejecución con un argumento:
    - Argumento 1: Corresponde al directorio de entrada
    - Directorio de salida: Se utilizará una ruta por defecto. El sistema creará automáticamente una carpeta llamada "resultados" dentro del directorio de entrada especificado.

- Ejecución sin argumentos:
    - Directorio de entrada: El sistema detectará como directorio de entrada el directorio actual el cual se corresponde con el del proyecto Java.
    - Directorio de salida: Se aplicará el comportamiento por defecto, creando una carpeta salida dentro del directorio de entrada

Al finalizar la ejecución, el directorio de salida contendrá los siguientes elementos:

- Ficheros de diagnóstico individuales: Se generará un fichero de salida por cada fichero de entrada procesado. Cada uno contendrá el diagnóstico específico para el fichero correspondiente.
- Fichero de diagnóstico global: Se generará un único fichero adicional que representará la evaluación conjunta de todos los ficheros de entrada.

5. Inferencia de diagnósticos.

5.1. Taquicardia y bradicardia.

Utilizando este enlace, en el apartado 19.4.2 podemos darnos cuenta de que tanto la taquicardia y bradicardia están relacionadas con la frecuencia cardíaca. Para el diagnóstico de la **taquicardia** basta con que la frecuencia cardiaca sea **mayor a 100 pulsaciones por minuto** mientras que para la **bradicardia** debe de ser **menor a 60 pulsaciones por minuto**.

5.2. Hipocalcemia.

Según esta información, en el apartado “Some pathological patterns which can be seen on the ECG” la hipocalcemia se detecta con un **intervalo QT prolongado**. Investigando encontramos en esta página que más o menos el intervalo QT debía de **ser mayor de 460 ms** pero en nuestro sistema para que no realice diagnósticos erróneos tuvimos que ajustarlo de manera que sea **mayor o igual a 440 ms**.

5.3. Infarto Agudo de Miocardio.

Según la información contenida en este enlace, en el apartado donde se muestran una serie de ejemplos de patrones patológicos (“Some pathological patterns which can be seen on the ECG”), un primer síntoma para detectar el infarto agudo de miocardio se produce cuando las **ondas T** se vuelven más **prominentes**, **simétricas** y **puntiagudas**, lo que se puede traducir a un **aumento de la amplitud de las ondas T**. No obstante, para poder diagnosticar esta enfermedad es necesario fijarse también en el segmento ST. Un **segmento ST prolongado y**

una duración mayor que 80ms, según esta fuente, de dicho segmento pueden ser clave para detectar el infarto agudo de miocardio.

5.4. Hipopotasemia.

En este enlace, se indica que la hipopotasemia causa **depresión del segmento ST** y un **aplanamiento o inversión de la onda T**. Según los ficheros input proporcionados, establecemos un umbral para el **pico máximo de la onda T (-14 mV)** con el fin de poder representar la inversión de la onda T y la depresión del segmento ST necesarios para poder diagnosticar la hipopotasemia.

5.5. Isquemia Coronaria.

Según esta fuente en su Apartado 9.1, la isquemia se reconoce por una **inversión de la onda T**. Además, también se menciona como síntoma un desnivel del segmento ST, pero en ambas imágenes la onda T aparece invertida. Así que, para diagnosticarlo, únicamente nos centramos en dicha característica.

La expresión “inversión de la onda T” indica que hay que establecer un umbral. El escogido fue **-9 mV** porque dicho valor era razonable para diagnosticarlo en los ficheros de input proporcionados. Además, agregamos que no debe superar el umbral de la Hipopotasemia (-14 mV) para que la Hipopotasemia no implicase Isquemia.

5.6. Contracción Ventricular Prematura.

Esta página indica que esta enfermedad ocurre cuando el complejo QRS dura **menos de 0.1 s** (origen supraventricular) o **más de 0.1 s** (origen ventricular). Pensábamos que nos debíamos basar en origen ventricular, pero con los ficheros de input, parece que debíamos realmente diagnosticar el supraventricular. Establecimos el umbral en **< 100 ms**. Sin embargo, el diagnóstico saltaba incluso para los pacientes sanos. Estudiamos los ficheros en los que se debía detectar y restablecimos el umbral en **< 75 ms**.

5.7. Sano.

Si a lo largo del proceso de inferencia no se detecta ninguna enfermedad, el paciente está sano. Para modelar esto en el motor de inferencia, tenemos una **regla con menor prioridad** que el resto. De esta forma, nos aseguramos que se ejecutará después de la inferencia de enfermedades.

6. Conclusiones.

A lo largo de la investigación y elaboración del sistema de inferencia, nos hemos dado cuenta de que existen enfermedades que sí se pueden diagnosticar a través de datos empíricos (como la hipocalcemia con un intervalo QT con una duración superior a 440 ms). Sin embargo, otras patologías presentan indicadores más ambiguos como "inversión de la onda T". En esos casos, hemos tenido que establecer umbrales a través de los ficheros de input de forma que la enfermedad fuese detectada en esos casos y no para otros.

Esto evidencia que en un sistema basado en conocimiento realista y fiable, se necesita un conjunto mucho más amplio relativo a datos de pacientes y electrocardiogramas para modelar relaciones más complejas y, así, mejorar la eficacia del diagnóstico.

Respecto al uso de Drools, nos ha impresionado la facilidad que tiene para especificar relaciones en un formato tan sencillo como una regla. Este enfoque reduce de manera significativa la implementación del sistema basado en conocimiento si hubiésemos empleado Java puro.

7. Bibliografía.

- Bem-Project. (s. f.). *The Basis of ECG Diagnosis*. https://www.bem.fi/book/19/19.htm
- Wikipedia. (2012, junio 11). *Electrocardiography* (revisión 513556137). https://en.wikipedia.org/w/index.php?title=Electrocardiography&oldid=513556137#Some_pathological_patterns_which_can_be_seen_on_the_ECG
- CardiologiaArritmias.com. (s. f.). *Hipocalcemia en ECG*. https://cardiologiaarritmias.com/hipocalcemia-en-ecg/
- Wikipedia. (2025). *ST segment*. https://en.wikipedia.org/wiki/ST_segment
- SalusPlay. (s. f.). *Tema 9. Cardiopatía isquémica*. https://www.salusplay.com/apuntes/cuidados-medico-quirurgicos/tema-9-cardiopatia-isquemica