UNIVERSIDAD DE LOS ANDES
MERIDA VENEZUELA

ESCUELA POLITÉCNICA NACIONAL

RECONOCIMIETO DE PATRONES

Jose Aguilar

El reconocimiento de patrones es una área conocida como Aprendizaje de Maquinas (Machine Learning) o Aprendizaje Automático.

Clasificar un patrón en una determinada clase ó categoría

Se determina la clase del patrón de prueba comparándolo con un conjunto previo (de entrenamiento), ya clasificado

Reconocimiento de patrones

El objetivo es clasificar patrones en base a un conocimiento a priori o información estadística extraída de los patrones.

Los patrones a clasificar suelen ser grupos de medidas u observaciones, definiendo puntos en un espacio multidimensional apropiado

Reconocimiento de patrones

En resumen, un **sistema de reconocimiento** de patrones completo consiste en:

• Un **sensor** que recoge las observaciones a clasificar.

• Un **sistema de extracción de características** transforma la información observada en valores numéricos o simbólicos.

• Un **sistema de clasificación** o descripción que, basado en las características extraídas, clasifica la medición.

Reconocimiento de patrones

PATRÓN: CONJUNTO DE CARACTERÍSTICAS DE UNA IMAGEN

Las características de la imagen pueden ser:

-topológicas: número de componentes conexas, agujeros,…

-geométricas: área, perímetro, curvatura,…

-estadísticas: momentos,…

Un **patrón** es un conjunto de características.

Una **clase de patrones** es un conjunto de patrones **similares.**

El objetivo del reconocimiento de patrones es **asignar un patrón a la clase a la que pertenece** (lo más automáticamente posible).

Reconocimiento de patrones

PATRÓN: CONJUNTO DE CARACTERÍSTICAS DE UNA IMAGEN

Reconocimiento de patrones

PATRÓN: CONJUNTO DE CARACTERÍSTICAS DE UNA IMAGEN

Ejemplo: supongamos que queremos discriminar tres tipos de flores (virginica, versicolor, setosa) mediante las características:

- la anchura de sus pétalos.
- la longitud de sus pétalos.

En este caso, un patrón consta de estas dos medidas tomadas de una flor en particular.

Una clase de patrones consistiría en el conjunto de todos los patrones que se obtienen de la misma clase de flor.

Reconocimiento de patrones

PATRÓN: CONJUNTO DE CARACTERÍSTICAS DE UNA IMAGEN

Podemos almacenar los patrones en diversos formatos:

• El más usual es el de **vector** (para características cuantitativas)

$$\boldsymbol{x}=[x_1, x_2, \ldots, x_n]^T$$

donde $\boldsymbol{x}$ es el patrón y x_i son las características

• También se usa el formato de árbol (para características estructurales).

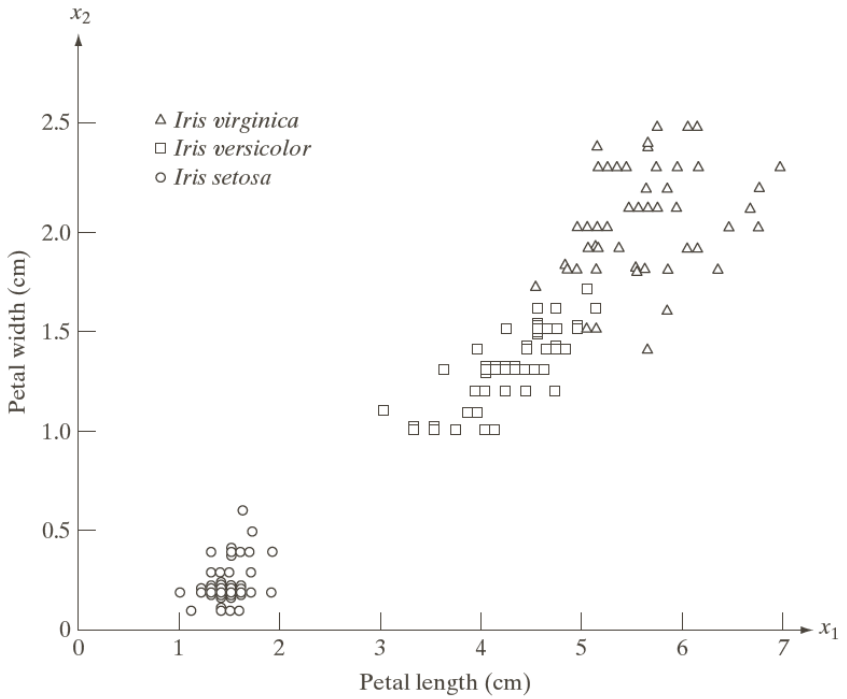
Reconocimiento de patrones

PATRÓN: CONJUNTO DE CARACTERÍSTICAS DE UNA IMAGEN

Ejemplo: supongamos que queremos discriminar tres tipos de flores (virginica, versicolor, setosa) mediante las características:

- la anchura de sus pétalos (x_1)
- longitud de sus pétalos (x_2).

En este caso, el patrón es $\boldsymbol{x}=[x_1, x_2]^T$



Como podemos observar, esta elección de características podrá discriminar perfectamente la clase setosa de las otras dos, pero no así las clases virginica y versicolor entre sí.

FIGURE 12.1
Three types of iris flowers described by two measurements.

Reconocimiento de patrones

PATRÓN: CONJUNTO DE CARACTERÍSTICAS DE UNA IMAGEN

Ejemplo: supongamos que queremos reconocer imágenes de satélites.

FIGURE 12.4
Satellite image of a heavily built downtown area (Washington, D.C.) and surrounding residential areas. (Courtesy of NASA.)

En este caso, usaríamos una estructura en árbol:

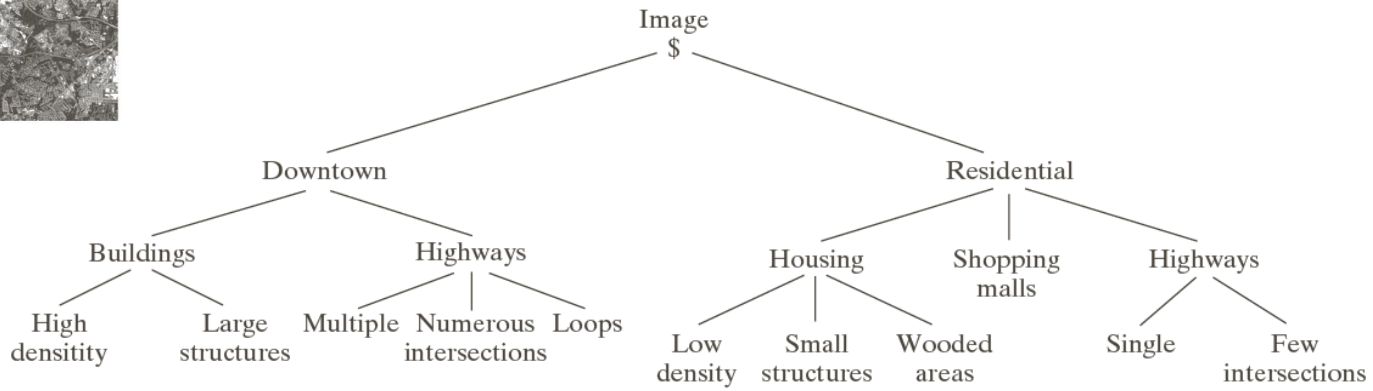


FIGURE 12.5 A tree description of the image in Fig. 12.4.

EJEMPLOS

Reconocimiento del rostro de una persona

Reconocimiento de huellas digitales

Identificar la llave para abrir una chapa

Agrupación de palabras para hacer sentido en una frase

Olores para saber si una naranja está fresca o podrida.

ENFOQUES:

Definición Patrón. Un patrón es un modelo, plantilla, conjunto de elementos con características propias únicas que se repite.

Existen dos tipos de patrones: estáticos y dinámicos, ambos orientados ya sea por sus características del patrón o por la percepción del patrón. Veamos con mas detalle estos dos casos.

CASO 1: PATRONES ESTÁTICOS Y DINÁMICOS ORIENTADO POR *CARACTERÍSTICAS.*

Patrón estático: **es un patrón cuyas características no cambian en el tiempo.**

- Dado el problema de las k reinas; identificar en que posiciones colocar k reinas en un tablero de ajedrez de tal manera que no se "capturen" entre ellas. En esta situación:
  - El tablero es un patrón estático, ya que sus características no cambian en el desarrollo de la solución del problema.
  - Las reinas también son patrones estáticos (así se mueva a diferentes casillas, mantiene sus características).

- "reconocimiento de situaciones en Aml"
  - Los estímulo (luz, oscuridad, calor, frio) son patrones estáticos.
  - Los objetos existentes en el aula de clase; sillas, tablero, proyectores,
  - Las emociones, puesto que la descripción de cada una (su patrón) es estático.

CASO 1: PATRONES ESTÁTICOS Y DINÁMICOS ORIENTADO POR *CARACTERÍSTICAS.*

Patrón Dinámico **(su negación es patrón estático): Patrón cuyas características cambian en el tiempo.**

Dado el problema de las k reinas;

- La configuración de las reinas en el tablero es un patrón dinámico. En la búsqueda de la solución pueden encontrarse varias configuraciones de solución o configuraciones erradas.

“reconocimiento de situaciones en Aml”.

- El comportamiento del profesor/estudiante
- Los procesos de aprendizaje (métodos, acciones, que lo componen)

CASO 2: PATRONES ESTÁTICOS Y DINÁMICOS ORIENTADO POR LA PERCEPCIÓN

Patrón estático: **es un patrón cuya percepción del mismo no varia según lo que vayamos observando.**

Dado el problema de las k reinas;

- La distribución de los cuadros y sus colores en el tablero de ajedrez, no varían según como lo veamos o vayamos viendo.

“reconocimiento de situaciones en AmI”

- El aula de clase es un patrón estático.

CASO 2: PATRONES ESTÁTICOS Y DINÁMICOS ORIENTADO POR LA PERCEPCIÓN

***Patrón Dinámico*: es un patrón cuya percepción del mismo va variando según la escala, dimensión o perspectiva de lo que vayamos observando.**

"reconocimiento de situaciones en Aml".

- Los estudiantes son patrones dinámicos
- Las emociones de los estudiantes son patrones dinámicos

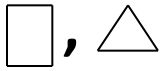
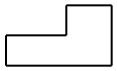
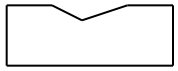
Ejemplos:

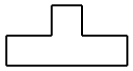
a) C1 = { 2,4,6,8 } C2 = { 1,3,5,7 }

3 ? Se determina a cuál conjunto pertenece el 3

b) C1 = números pares C2 = números impares

8 ? Se utiliza la propiedad X mod 2 para determinar la pertenencia

c) C1 = {  **} C2 = {**  **,**  **}**

 ? Se utiliza la propiedad de convexidad

d) C1 = 1 (1|0)* 0 C2 = 0 (0|1)* 1 C3 = ….

11011010 ? Se utiliza la regularidad de la expresión regular

Ejemplos:

e) Diagnóstico médico: Sobre la base de informaciones almacendas en Historias Clínicas se tiene una sucesión de pacientes:

P_1, P_2, P_3,…,P_{m1} que padecen la enfermedad E1

P_{m1+1}, P_{m1+2}, …, P_{m2} que padecen la enfermedad E2 y así sucesivamente

P_x ?

<u>Problemas</u>

- **No se descarta que un paciente tenga más de una enfermedad**
- **No se tiene el listado completo de pacientes que padecieron cada una de las enfermedades**
- **No se tiene una propiedad que caracterice a cada enfermedad**

Punto de Vista Formal:

- Consideremos un universo de objetos admisibles M
- $K_1, K_2, \ldots, K_r$ subconjuntos propios, donde K_i TM
- $R = \{ X_1, X_2, \ldots, X_N \}$ Conjunto de rasgos ó características que describen a los objetos
- Cada rasgo posee un conjunto de valores admisibles, incluyendo ausencia de información

Ejemplo:

R = { color, área, forma }

color = {rojo, verde, azul }

área = número real

forma = { convexa, no convexa }

Punto de Vista Formal:

- Una descripción estándar de un objeto O es un n-uplo:

$I(O) = (X_1(O), X_2(O), \ldots X_n(O))$

La descripción es completa si todos los valores de los rasgos son conocidos.

Ejemplo:

I(O) = (verde, 23.5, convexo)

Función de comparación:

Permite comparar individualmente los rasgos de dos obejtos.

Existe una función de comparación por cada rasgo (N rasgos, N funciones):

$$C_i (X_i(O_p), X_i(O_q)) \in \{ 0,1 \}$$

Estas N funciones sirven para comparar individualmente los rasgos de los objetos, pero no para decidir si los objetos son semejantes

Función de semejanza:

Permite comparar dos objetos en base a los valores de sus rasgos.

$$S (I(O_p), I(O_q)) \in \{ 0,1 \}$$

R-uplo de Pertenencia de O

$P(O) = (P_1(O), P_2(O),\ldots,P_r(O))$

Se define como un vector de r valores que indican la pertenencia ó no del Objeto O a cada una de las r clases (K_1, K_2 ,…, K_r)

Cada valor Pi(O) es 1 ó 0 si O pertenece ó no a K_i

Información Estándar de Clases:

Es la información que recoge la descripción de todos los objetos conocidos por cada clase

$I(K_i)= (I(O_{i1}), P(O_{i1}),$

$I(O_{i2}), P(O_{i2})\ldots,$

$I(O_{iz}), P(O_{iz}))$

Matriz de aprendizaje:

$I_0 = (I(K_1), I(K_2), \ldots, I(K_r))$

Ejemplo:

Supongamos que tenemos 5 clases (A,B,C,D,E) (r=5)

Y los siguientes objetos con sus rasgos y sabemos, a priori, que clase pertenece cada objeto

O1 = (rojo,4.2,convexo) Clase A

O2 = (verde,3.5,convexo) Clase A

O3 = (rojo, 2, no convexo) Clase B

O4 = (verde, 1.3, no convexo) Clase B

O5 = (rojo, 6.9, convexo) Clase C

O6 = (verde, 8, no convexo) Clase C

O7 = (azul,10, convexo) Clase D

O8 = (azul, 9, no convexo) Clase E

Matriz de aprendizaje:

I(O)	P(O)
(rojo, 4.2, convexo),	(1, 0, 0, 0, 0)
(verde,3.5,convexo),	(1, 0, 0, 0, 0)
(rojo, 2, no convexo),	(0, 1, 0, 0, 0)
(verde, 1.3, no convexo),	(0, 1, 0, 0, 0)
(rojo, 6.9, convexo),	(0, 0, 1, 0, 0)
(verde, 8, no convexo),	(0, 0, 1, 0, 0)
(azul,10, convexo),	(0, 0, 0, 1, 0)
(azul, 9, no convexo),	(0, 0, 0, 0, 1)

El problema ahora consiste en:

Dado una matriz de aprendizaje (ó control) y los rasgos de un conjunto de objetos de los cuales no conocemos sus clases, encontrar un Algoritmo que permita clasificarlos.

$$A\,(\,Io,\,I(O)\,) = (\,P_1(O),\,P_2(O),\ldots,P_r(O)\,)$$

Problemas de Reconocimiento de Patrones:

1) Selección de rasgos:

El problema consiste en determinar para un conjunto de objetos, que conocemos su clasificación cuál es el conjunto de rasgos mínimos que debemos seleccionar para realizar correctamente la clasificación.

Mejorar la clasificación (más rápido, menos información)

Para encontrar mejores soluciones, ya que los objetos se describen de manera más eficiente

Problemas de Reconocimiento de Patrones:

2) Clasificación

Con aprendizaje ó supervisada: Se conoce que el universo de objetos se agrupan en las r clases de las cuales tenemos una muestra que conocemos a que clases pertenecen (matriz de aprendizaje)

Con aprendizaje parcial ó parcialmente supervisada: Se conoce que el universo de objetos, pero existen clases para las cuales no tenemos objetos en la matriz de aprendizaje. No tenemos información de todas las clases.

3) Agrupamiento (cluster analysis)

Tenemos los objetos sin clasificar. No conocemos las clases y el problema consiste entonces en agruparlos por rasgos “comunes”. También llamado **clasificación no supervisada**.

Reconocimiento de patrones

El punto esencial del reconocimiento de patrones es la **clasificación.**

Se quiere clasificar una imagen dependiendo de sus **características**.

Por ejemplo se puede clasificar imágenes digitales de letras en las clases «A» a «Z» dependiente de sus píxeles o se pueden clasificar huellas dactilares.

Reconocimiento de patrones

La **clasificación** utiliza habitualmente uno de las siguientes procedimientos:

• **clasificación estadística** (o teoría de la decisión), basado en las características estadísticas de los patrones.

• **clasificación sintáctica** (o estructural), basado en las relaciones estructurales de las características.

La clasificación puede ser de dos tipos:

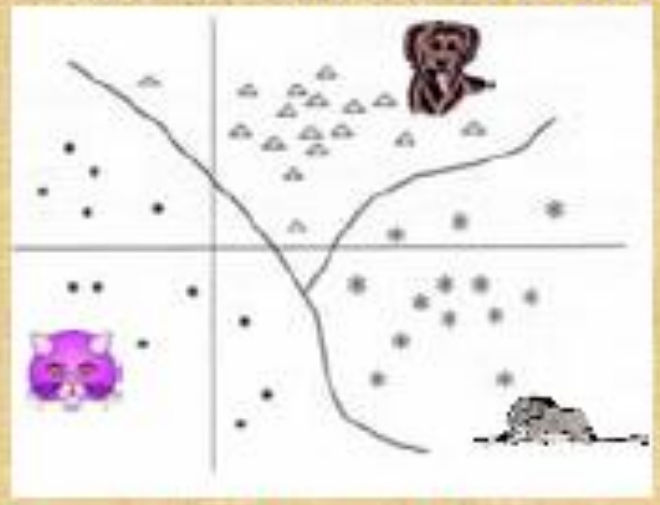
• **Supervisada**, si se usa un conjunto de aprendizaje, que sirve para entrenar al sistema.

• **No supervisada**. El sistema no tiene un conjunto para aprender a clasificar la información a priori, sino que se basa en cálculos estadísticos para clasificar los patrones.

Reconocimiento de patrones

Clases Separables

Si existe una **hipersuperficie de discriminación** que separa el espacio de propiedades en regiones que contienen sólo un tipo de objetos, **las clases serán separables**



En la mayor parte de los problemas de reconocimiento en Tratamiento Computacional de Imagen las clases no son separables

Reconocimiento de patrones

RECONOCIMIENTO BASADO EN MÉTODOS DE DECISIÓN

Sea $\boldsymbol{x}=[x_1, x_2, \ldots, x_n]^T$ un patrón donde cada x_i es una característica.

Para cada clase **w** de patrones, hay que encontrar una función de decisión d_w **(clasificador)** con la propiedad de que si ***x*** pertenece a la clase ***w*** y no a la clase **v** , entonces

$$d_w(x) > d_v(x)$$

Frontera de decisión: aquellos vectores **x** tales que $d_w(x) = d_v(x)$.
Considerando la función frontera $d_w(x) - d_v(x)$, dicha función tomará valores >0 cuando **x** pertenezca a la clase **w** y valores <0 cuando pertenezca a **v**.

Ejemplo

Los objetos son figuras geométricas.

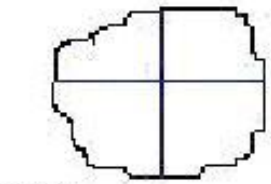
Tres clases A, B y C que representan objetos redondos, semi-alargados y alargados respectivamente

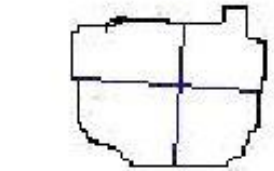
Tendremos un solo rasgo, calculado en base a la división entre el diámetro mayor y menor del objeto

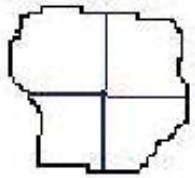
Tendremos un conjunto de objetos cuyos rasgos y clasificación conocemos

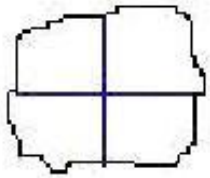
Objetos del Entrenamiento

Clase

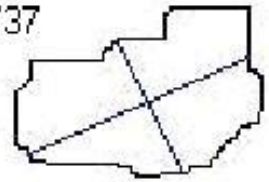

1.25

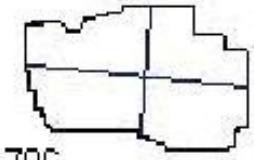

1.288


1.001

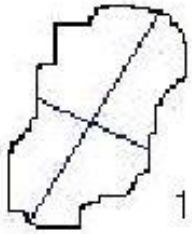

1.197

A
redondo

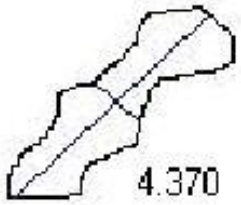

1.737


1.786

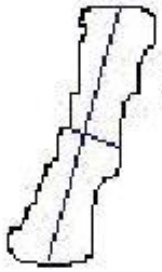

2.023


1.985

B
semi
alargado


4.370


5.414


4.728


3.962

C
alargado

UNIVERSIDAD DE LOS ANDES
MERIDA VENEZUELA

Lógica Temporal aplicada a Reconocimiento de Patrones Dinámicos

Jose Aguilar

CEMISID,
Dpto. de Computación
Facultad de Ingeniería

aguilar@ula.ve

Contenido

1. **Bases teóricas de la Lógica Temporal**: presentación de las técnicas clásicas de lógica temporal (extensiones a la lógica proposicional, lógica de predicados, lógica modal, lógica arborescentes, etc.)

2. **Introducción a las Crónicas:** razonamiento y aprendizaje, centralizadas y distribuidas.

3. **Aplicaciones de las Crónicas:** seguimiento de objetivos móviles, sistemas autónomos de comunicación, middleware reflexivos, etc.

Representación del Conocimiento

- Un lenguaje: sintaxis y semántica
  - Sintaxis define las sentencias en el lenguaje
  - Semántica define el “significado" de las sentencias
- Procedimiento de Inferencia
  - Determinar Activación de otras Sentencias dada una Sentencia:
  - abductivo, inductivo, deductivo
- Lenguajes lógicos: son lenguajes formales para representar información tales que conclusiones puedan ser alcanzadas
- Ejemplos de lenguajes lógicos
  - Calculo Proposicional (hechos)=>V/F
  - Calculo de Predicado de Primer Orden (Hechos, Objetos, Relaciones) =>V/F/D
  - Lógica Temporal (hechos, Objetos, Relaciones y tiempo) =>V/F/D
  - Teoría de probabilidad (hechos) =>Grado de Creer
  - Lógica Difusa (Grados de Verdad) =>Grado de Creer

} incertidumbre

Lógica Temporal

- Es un tipo de lógica simbólica, cuyo valores de verdad de las proposiciones dependerán del tiempo

"Tengo hambre"

- Aunque su significado es constante en el tiempo, el valor de verdad de la declaración puede variar en el tiempo.

- La declaración no puede ser verdadera y falsa al mismo tiempo.

Lógica Temporal

- **Contrasta** con el punto de vista de la **lógica clásica**,
  - Valor de verdad de una proposición es siempre el mismo
  - Declaraciones con valores constantes en el tiempo.
  - Por ejemplo:

"La luna está apareciendo" vs "La luna es un satélite de la tierra"

La primera proposición tiene una condición implícita de tiempo "**ahora**". La segunda proposición es **atemporal**

Lógica Temporal

- La lógica temporal se aplica a universos de discursos en donde sus fórmulas lógicas describen secuencias de estados

- La principal ventaja de la lógica temporal es su expresividad:
  - "Siempre estoy dormido", “A veces tengo sueño", "Voy a tener sueño en algún momento“.

- Amplias formas de razonamiento con relaciones causales:
  - Si observo A y sé que A causa B, entonces predigo B;
  - Si deseo obtener D y sé que C causa D, entonces hago C;
  - Si noto F y me desagrada y sé que E causa F, entonces trato de actuar sobre E

- El problema de modelado es determinar:
  - relaciones causales genéricas y
  - la evolución de un conjunto de eventos para un estado del mundo a partir de esas relaciones causales.

Lógica Temporal

- Diferentes técnicas de modelado se pueden utilizar:
  - **Técnicas basadas en lógica**: lógica modal, crónicas, etc.;
  - Técnicas que tratan de analizar la **relación probabilística entre las acciones** (MDP),
  - Etc.

- Tres métodos principales:
  1. Uno que considera **operadores temporales** (por ejemplo: lógica modal);
  2. Otro que identifica los tiempos o **intervalos de tiempo en el que las proposiciones lógicas son válidas**;
  3. Otro introduciendo el **tiempo como argumento en los predicados** en sentencias en lógica predicado de primer orden y proposicional

Lógica Temporal Proposicional (PTL)

✓ Además de los operadores del cálculo proposicional ($\Lambda, \nu, \neg, \rightarrow$), existen tres operadores unarios temporales:

- *always,* denotado por $\Box$, "*para cualquier instante t en el futuro*",
- *eventually,* denotado por $\Diamond$, "*para algún instante t en el futuro*",
- *next,* denotado por $\circ$, "*en el instante siguiente*".

✓ La semántica de PTL da una interpretación temporal para las proposiciones

✓ Define un conjunto de estados. Cada uno contiene una interpretación de las proposiciones.

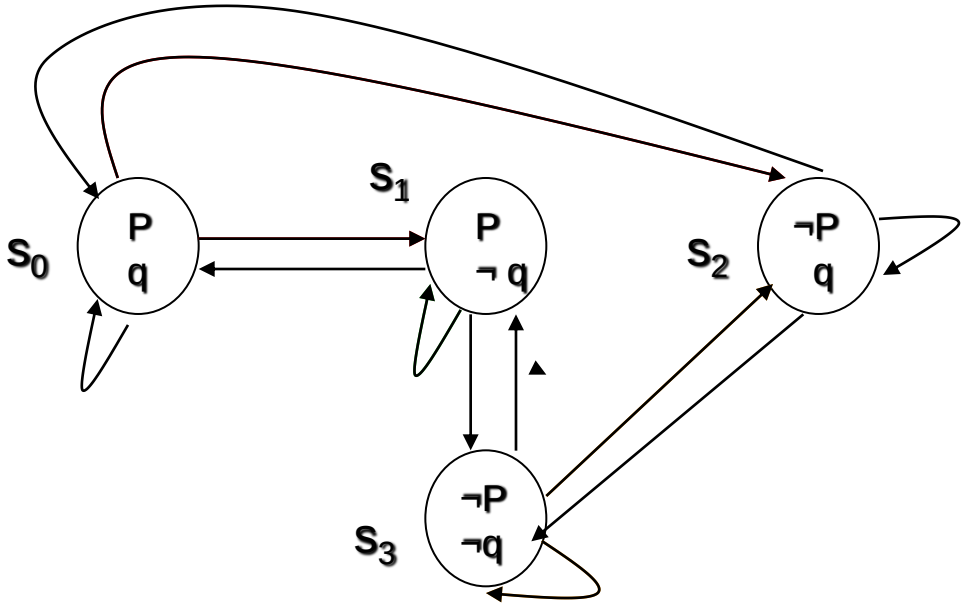
✓ Una interpretación PTL se dibuja como un diagrama de transición de estados.

✓ El tiempo se representa mediante transiciones entre estados.

PTL

- ✓ el valor cierto de la fórmula temporal $\boldsymbol{A = p \vee q}$ se determina para cada estado s_i

  - *A* es cierto en s_0
  - *A* es cierto en s_1
  - *A* es cierto en s_2
  - *A* es falso en s_3



Lógica Temporal basada en Predicados

- Los conceptos sobre el tiempo quedan fuera de la lógica de primer orden:
  - Ej: Como indicar una acción en un tiempo determinado.

    John está corriendo

    John correrá

- Estos predicados en lógica de primer orden quedan reducidos a un sujeto que ejecuta una acción

  John correr -> Corre (John).

- La importancia del tiempo radica en que, con la inclusión del tiempo, nos aporta información muy útil para consideraciones ulteriores

Lógica Temporal basada en Predicados

- Una primera aproximación se puede lograr añadiendo algo que indique el tiempo,

  TIEMPO(t) y CORRER(John, t)

- La segunda opción es hacer uso de la lógica modal, con los predicados :
  - necesidad -> para predicados que se cumplen siempre (todo el tiempo)
  - posibilidad -> para predicados que se cumplen en algún tiempo

- Los diferentes instantes de tiempo hacen el papel de diferentes posibles mundos en la **lógica modal**.

Lógica Temporal basada en Predicados

- **Instantes de tiempo**
  - ANTES(T1 , T2) -> el tiempo T1 se produce antes que T2
  - DESPUES(T2 , T1) -> el tiempo T2 se produce después que T1
  - MISMO(T1 , T2) -> el instante de tiempo es el mismo.

- **Intervalos de tiempo**

  Un intervalo de tiempo es un fragmento de tiempo que empieza en un instante de tiempo y acaba en otro instante posterior.

  INTERVALO(T1 , T2) -> intervalo.

Mundos Posibles y Lógica modal

- El lenguaje para expresar la semántica de un mundo **posible o necesario** es *la lógica modal*.
- La lógica modal fue desarrollada para formalizar argumentos que incluyen *necesidad* y *posibilidad*.

Una ***proposición necesaria*** es una proposición verdadera que no puede ser falsa.

Una ***proposición posible*** es una que podría ser verdadera.

Mundos Posibles y Lógica modal

- La sintaxis de la lógica modal es la de la lógica clásica con la adición de dos operadores

□ necesidad

◊ posibilidad (quizás)

Lógica alética

□x significa “es necesario x“ (“no x es imposible”)

◊x significa "x es posible".

Aplicaciones Lógica

- Es posible que llueva

$$\Diamond \text{ llueva}$$

- Es necesario que el sol se levante por el este

$$\Box \text{ sol se levante por el este}$$

Lógica Modal Básica

- Se construye sobre el lenguaje de la lógica de proposiciones, con la incorporación de los dos nuevos operadores.

- Alfabeto :
  - letras proposicionales: po, p1, p2, . . .
  - símbolos lógicos:
    - constantes proposicionales: $\perp$, T
    - conectivas monarias (¬) y binarias: ^,V,->
    - operadores modales: $\Diamond$, $\Box$

Lógica Modal Básica

- Las fórmulas del lenguaje se definen según BNF:

$\phi ::= p \mid \bot \mid T \mid > \mid \neg p \mid (po \wedge p1) \mid (p0 \vee p1) \mid (po \rightarrow p1)$
$\mid \Box p \mid \Diamond p$

- Equivalencias
  - $\Box(\phi \wedge \beta) \equiv (\Box\phi \wedge \Box\beta)$
  - $\Diamond(\phi \vee \beta) \equiv (\Diamond\phi \vee \Diamond\beta)$
  - $\Box p = \neg\Diamond\neg p$
  - $p0 > p1 = \neg\Diamond(p0 \wedge \neg p1) = \Box(p0 \rightarrow p1)$

Mundos Posibles y Lógica modal

Lógica Modal

$$\Diamond p = \neg\Box\neg p$$

$$\Box p = \neg\Diamond\neg p$$

Lógica Predicado

$$\exists x\ \phi(x) = \neg\forall x\ \neg\phi(x)$$

$$\forall x\ \phi(x) = \neg\exists x\ \neg\phi(x)$$

La lógica modal también puede enriquecer a la lógica de predicado con sus operadores

- □ ∀x (A-> B) -> □ (∀x A-> ∀x B)
- ∀x □ A-> □ ∀x A

Lógica Modal Básica

Fórmulas modales que caracterizan relaciones binarias

– Reflexiva	(□ p-> p)	(T)
– Simétrica	(p-> □ ◊ p)	(B)
– Transitiva	(□ p-> □ □ p)	(4)
– Euclídea	(◊p-> □ ◊ p)	(5)
– Determinista	(◊ p-> □ p)	(Q)
– Serial	(□ p-> ◊ p)	(D)
– Inferir	□(p0-> p1) -> (□p0-> □p1)	(K)

Lewis, 1912 propuso varios sistemas lógicos aléticos definidos por un número dado de axiomas que los caracterizan

S5=KT5 S4=KT4

Lógica Modal

- Otros operadores: **Happens, Done, BEL**, y **GOAL**
  - *Happens* define una secuencia de eventos que ocurrirán
  - *Done* define una secuencia de eventos que han ocurrido
  - *e;e'* (*e* seguido de *e'*)
  - *e | e'* (*e* o *e'*)
  - *e?* (test) *e** (iteración)

Sistemas Lógicos Temporales

Se diferencian de la lógica modal por:

- Lenguaje más rico: operadores presente, pasado y futuro
- Permiten formalizar las propiedades que dependen del tiempo

Tipos:

1. Lógica temporal arborescente (Computational tree logic, CTL),
2. Lógica temporal lineal (Linear temporal logic, LTL)
3. Lógica temporal de intervalos (Interval temporal logic, ITL).
4. Lógica temporales de acciones (Temporal Logic of Actions, TLA).

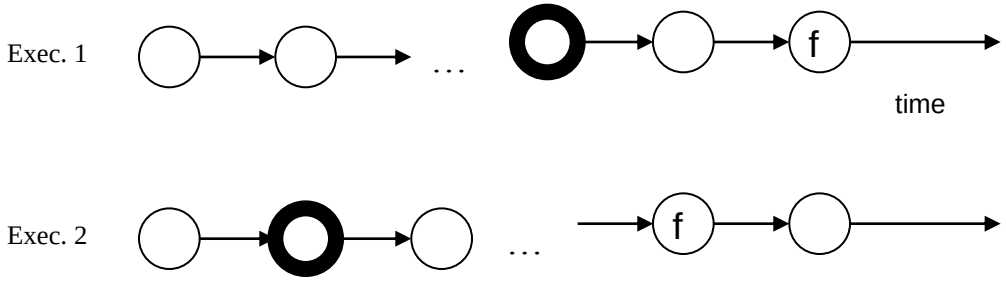
Ontología del tiempo: algunos operadores Temporales

- O or X or N (just after, next: tomorrow or immediately after),
- □ (henceforth, from now),
- ◊ (finally),
- J or U (until),
- Θ (just before),
- Δ (front),
- Φ (ever), D (from),
- F (eventually: once, sometimes),
- G (always),
- R (release).

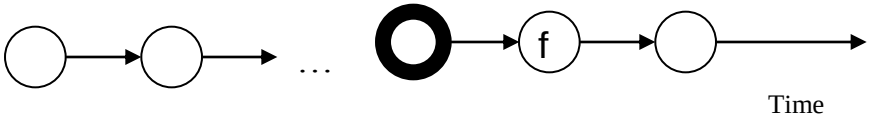
Linear Temporal Logic (LTL)

Se considera los comportamientos modelados como secuencias lineales de Estados

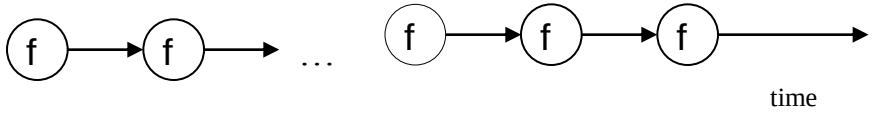
- **F(f)** Es verdad ahora si f es cierto al menos en una etapa posterior



- **X(f)** Es verdad ahora si f es cierto en el siguiente paso

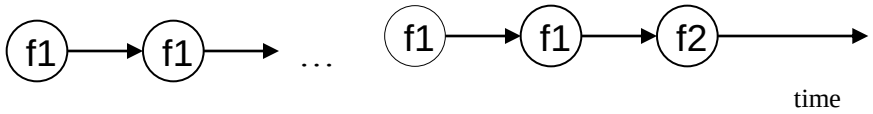


- **G(f)** Es verdad ahora si f es cierto en todas las fases posteriores, incluyendo ahora



LTL

– **Op. U:** f1 U f2 es verdad si f1 es verdad hasta que f2 es verdad,



Ejemplo de sentencia LTL (llamada telefónica desde una cabina telefónica pública):

*u_unhook ^ **X** sys_displays(Insert_Card) ^ **X** [(u_insert_card ^ **X** sys_displays(nb_units_remaining)) ^ (**X** u_call_no V **F** (u_correction_no **U** no_correct)) ^ ((**X** u_connect V **F** (u_change_card_empty **U** units(card) > minimum)) **U** (u_hang up ^ u_remove_card))] => ltl_phone*

Donde ^ *y V son los clásicos operadores y o*

Ejemplos de propiedades en LT

- EVT2 no va antes de EVT1: $G(\neg(EVT2)\ U\ EVT1)$
- Si EVT2 entonces EVT1 será después: $G(EVT2 \Rightarrow X\ (EVT1))$
- ***Propiedad de Hambruna:*** Un recurso es garantizado al menos una vez a un proceso:

$$F(ai \wedge aj)\ i \neq j$$

- ***Propiedad de Seguridad***: Pi y Pj son 2 procesos y CSi significa que proceso Pi esta en su sección critica, la siguiente formula describe la exclusión mutua de Pi y Pj:

$$G(\neg\ (CSi \wedge CSj))$$

- ***Propiedad de Vivacidad***: TRYi expresa una solicitud de acceso a Csi del proceso Pi, la siguiente formula expresa una solicitud de acceso a la sección critica que eventualmente se le dará:

$$TRYi \Rightarrow F\ CSi$$

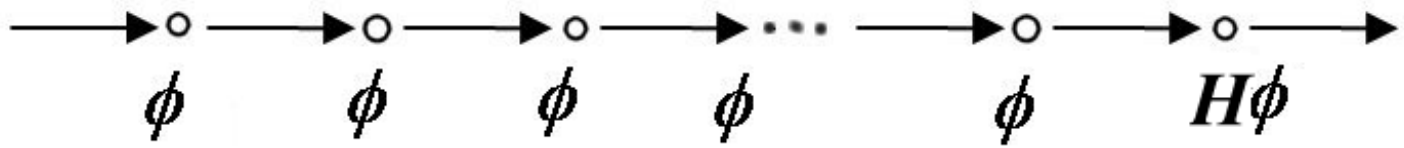
Operadores LTL

- Booleanos: negación, conjunción, etc.
- Temporales:
  - **Prevφ** : φ es verdad en el momento anterior (justo antes) (***PreviousVersion***)

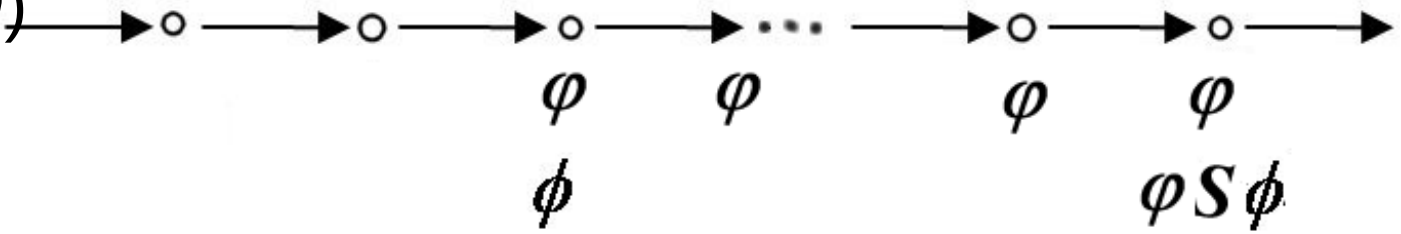
  - **Pφ**: φ es verdad en un momento anterior (algunas veces en el pasado) (***SomePriorVersion)***

Semantica en LTL

Hφ: φ es verdad en todos los momentos anteriores (siempre en el pasado) (***AllPriorVersions***)



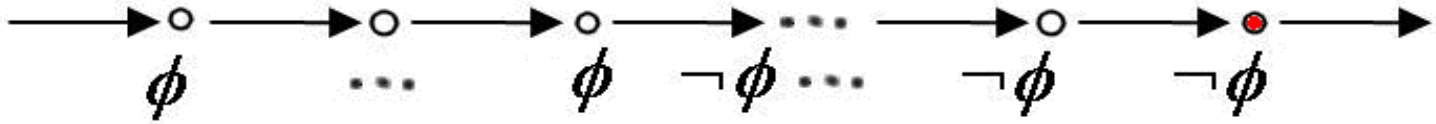
φSϕ: φ es verdad en todos los momentos anteriores desde que ϕ fue verdad en un momento anterior (***From***)



Propiedades Formales

- $H\ \varphi \rightarrow P\ \varphi$.
- $H\ \varphi \rightarrow Prev\ \varphi$.
- $Prev\ \varphi \rightarrow P\varphi$.
- $Prev\ P\ \varphi \rightarrow P\ \varphi$.
- $P\ P\ \varphi \rightarrow P\ \varphi$.
- $H\ H\ \varphi \rightarrow H\ \varphi$.
- $Prev\ Prev\ \varphi \rightarrow P\ \varphi$.

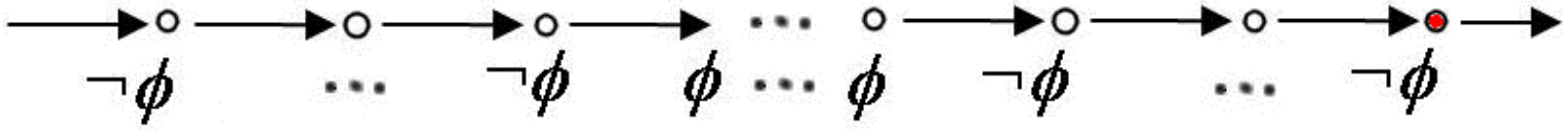
Ejemplos de Razonamiento



- Una vez ϕ cambia, nunca cambia más.

$$\neg\phi \; S \; (H\phi)$$

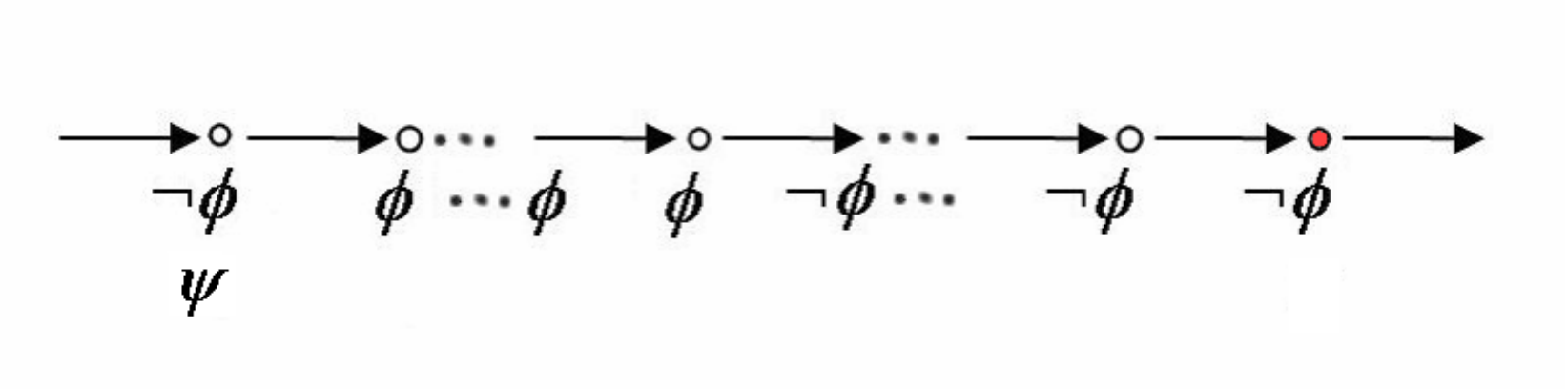
Ejemplos de Razonamiento



- ϕ cambia dos veces

$$\neg\phi \; S \; Prev(\phi \; S \; H\neg\phi)$$

Ejemplo de Razonamiento



ψ es verdad en la última ocasión en la cual ϕ no fue verdad, antes de la última ocasión en la que ϕ fue verdad

¬ϕ S (Prev(ϕ S Prev(¬ϕ∧ ψ)))

Computation Tree Logic (CTL)

Es una lógica temporal donde se ramifican las ocurrencias de estados, lo que permite expresar propiedades en el árbol de ejecución (desde un estado inicial)

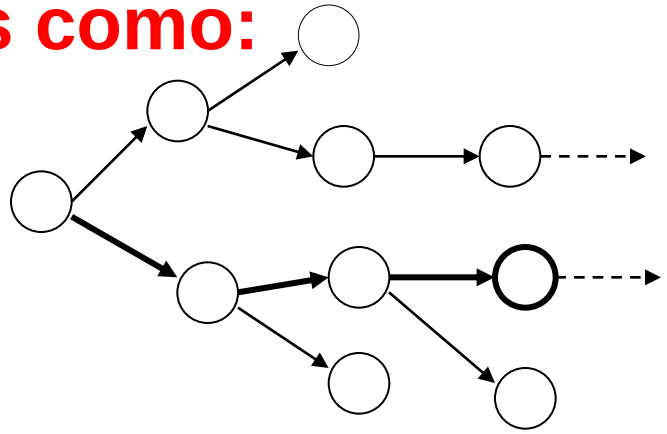
- CTL describe comportamientos más complejos que LTL

- 2 nuevos operadores:
  - **A** (Aϕ significa ϕ debe ser satisfecho en todas las rutas a partir del estado actual)
  - **E** (Eϕ significa que existe (hay) al menos un camino que comienza en el estado actual en la que ϕ es verdadero).

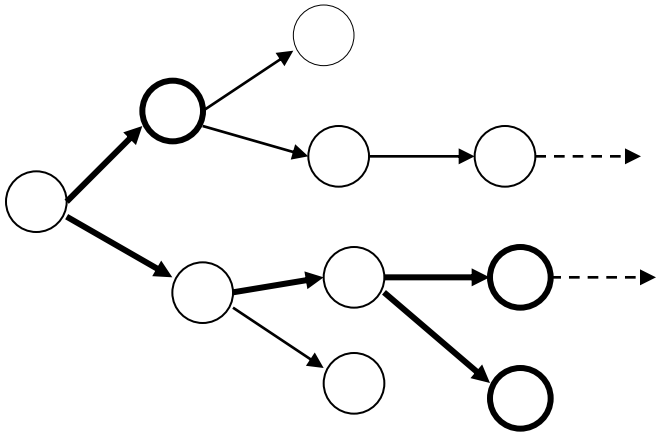
Computation Tree Logic (CTL)

Con estos operadores se pueden construir sentencias como:

- **EF(f)** significa "Es posible alcanzar un estado donde f es verificado" o “hay una ejecución (operador E) que conduce a un estado donde f es cierto (operador F)



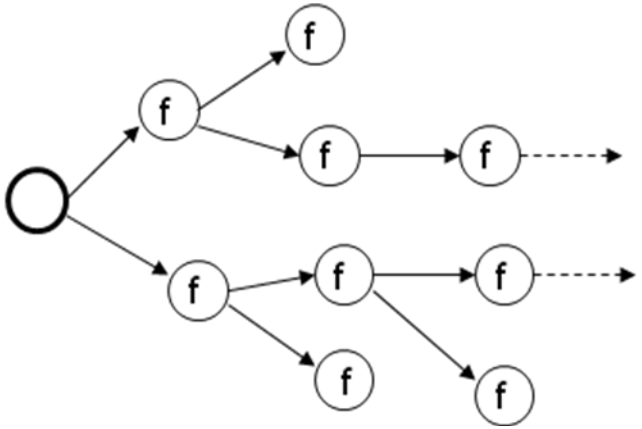
- **AF(f)** significa "f se verifica en el futuro" o "para toda ejecución (operador A), hay un estado donde f es cierto (operador F)".



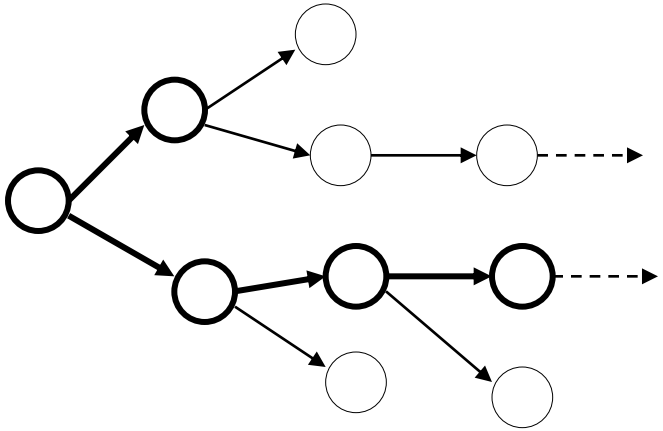
Computation Tree Logic (CTL)

AX(f) significa "todos los estados inmediatamente sucesores satisfacen f".

AG(f) significa "todas las ramas sucesoras satisfacen f".

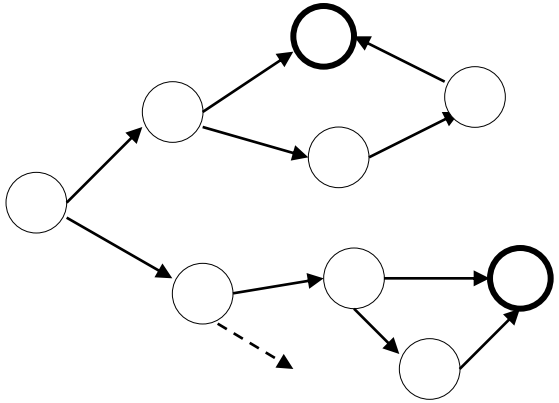


EG(f) significa “hay una ejecución (E operador) donde f es siempre verdadero (operador G)".

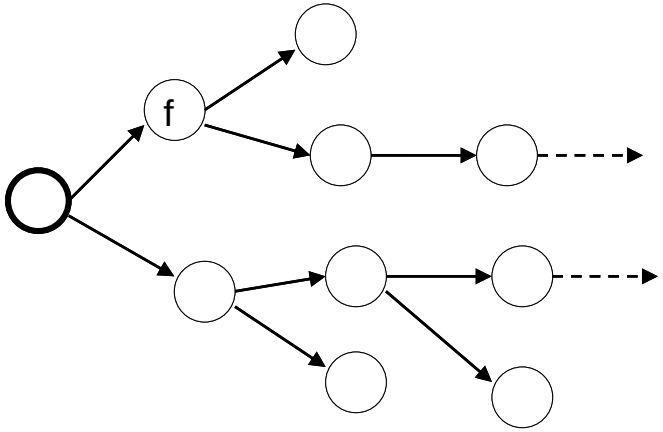


Computation Tree Logic (CTL)

AGEF(f) significa que desde cualquier estado alcanzable, es posible llegar a un estado que satisface f"

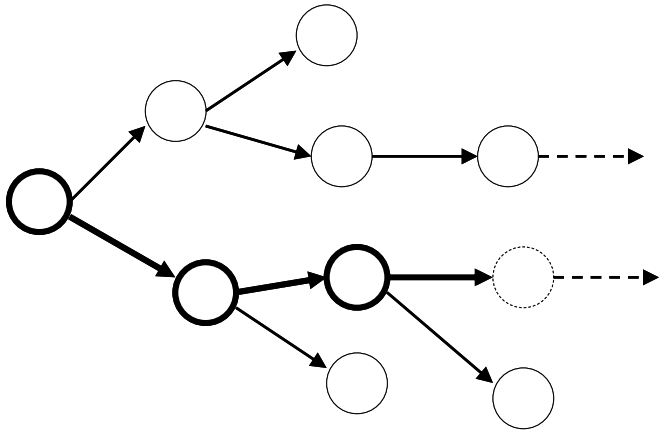


EX(f) significa que hay una ejecución (E operador) que en el siguiente estado satisface f ".



Clasificación

E(f) U g significa que “hay una ejecución (E operador) durante la cual f es verdadera hasta que g es cierto (f U g)".



- Mismo ejemplo con CTL:

u_unhook ^ u_insert_card ^ ***AX*** *[(u_call_no ^* ***EF*** *u_error_recovery) ^* ***AX*** *((u_connect ^* ***EF*** *change_card_empty)* ***U*** *(u_hang_up ^ u_remove_card)))] => ctl_phone*

Crónicas

Crónica

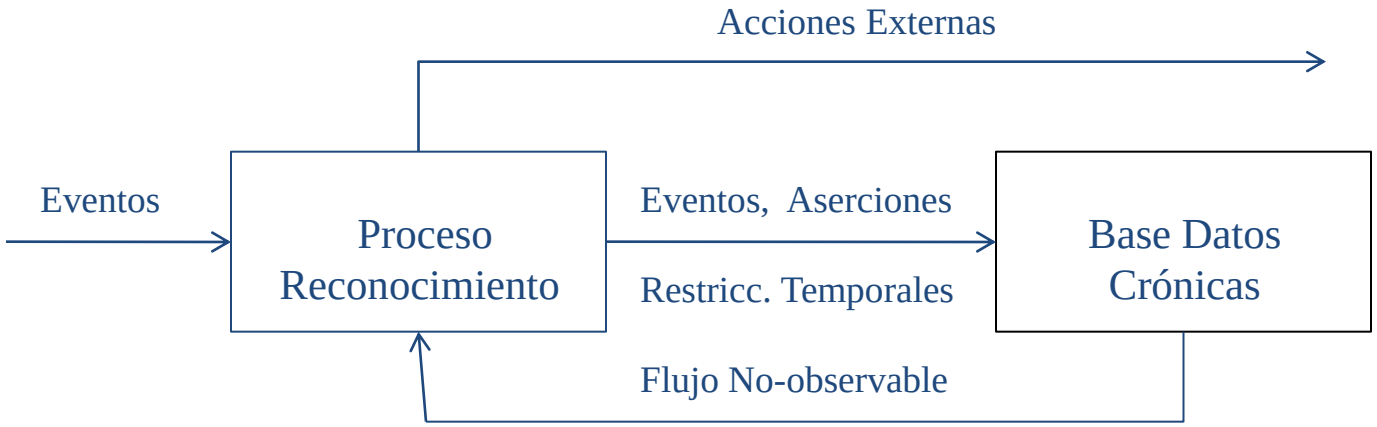
- Conjunto de eventos, unidos entre sí por restricciones temporales y contextuales

  Una crónica es un patrón de eventos.

- Representa la evolución de una parte del mundo.

  El patrón temporal descrito por la crónica caracteriza el fenómeno a ser identificado.

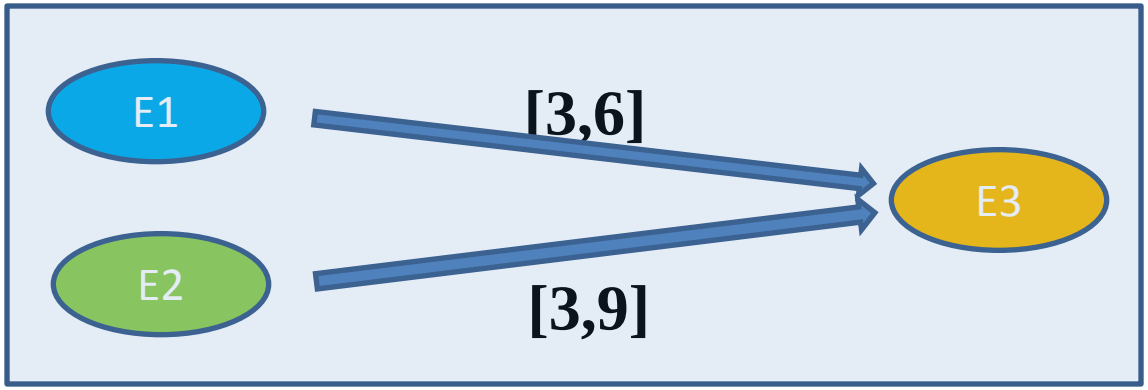
- Las relaciones entre los eventos son lógicas o temporales:
  - *En relaciones lógicas* se han considerado principalmente la conjunción (A y B) y la disyunción (A o B).
  - *En relaciones temporales* se consideran, principalmente, la secuencia (continuación de eventos ordenados) y la ausencia de eventos entre dos eventos (por ejemplo, C no debe ocurrir entre A y B)



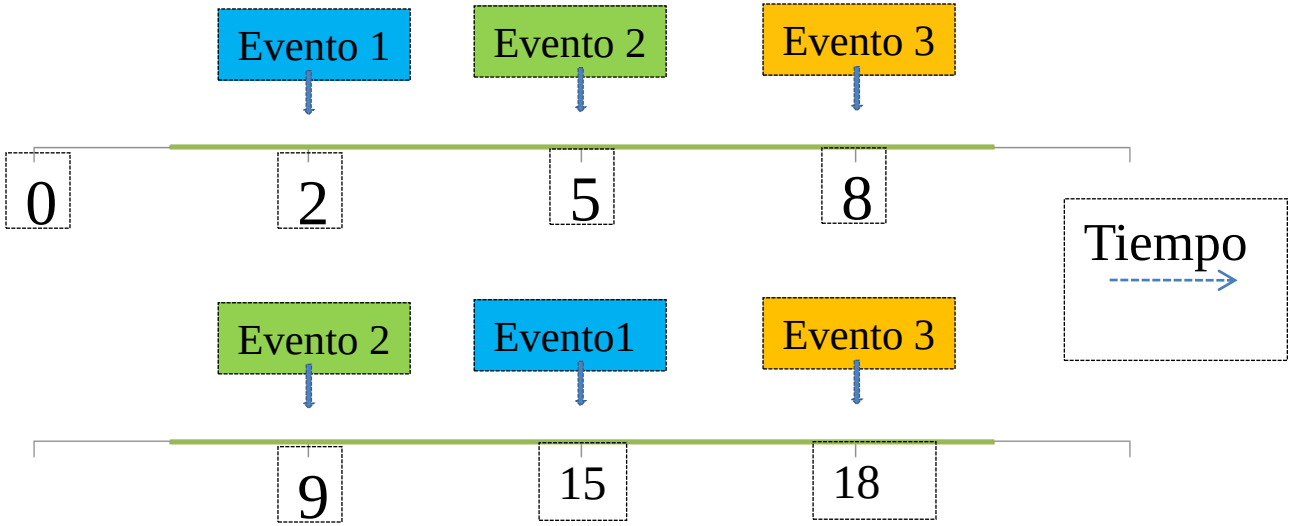
Crónica

Representa una parte observable de la evolución de un sistema

Patrón de eventos observables con restricciones temporales entre las fechas de ocurrencias entre ellos



Dos secuencias de eventos que pertenecen a esa crónica



Crónica

(Ghallab, 1994)	El modelo de crónica esta basado en 2 formulas que vienen de la lógica temporal:
	"**hold**" expresa que algunos atributos se mantienen durante algún intervalo de tiempo, por ejemplo:
	Hold(position (agente1, docking-site), (t5, t6)).
	"event" especifica un cambio discreto del valor de un atributo, por ejemplo:
	Event(state (switch): (off, on), t8).
	
	Chronicle RobotLoadMachine {
	event (Robot1: (outroom, inroom), e1);
	event (Robot1: introom, outroom), e4);
	event(MachineInput: (unloaded, loaded), e2);
	event(Machine: (Stopped, Running), e3);
	e1 ≤e2;
	1 ≤e3-e2≤6;
	3 ≤e4-e2≤5;
	hold (Machine: Running, (e2, e3));
	hold (SafetyConditions: true, (e2, e3));
	when recognized { report 'successful load'; }}

Crónica

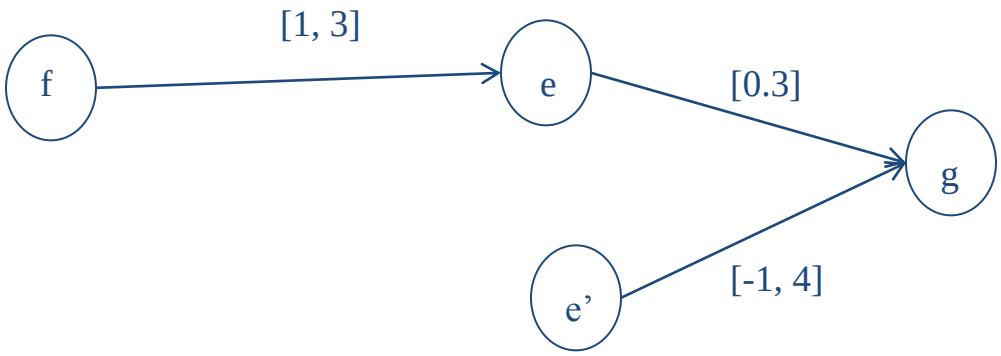
(Dousson et al.,1994) (Dousson et al., 2007)	• Usan grafos con restricciones temporales como modelo. • Los sistemas son descritos por puntos en el tiempo donde el tiempo es un conjunto linealmente ordenado de eventos discretos. • Una restricción de tiempo entre dos puntos de tiempo T1 y T2 está representado por un intervalo [I-, I +], que corresponde a los límites inferior y superior de la distancia temporal de T1 a T2 **Event** *label*(*vars*)**:** *e*s un tipo de dato donde *label* se emite durante el evento y *vars* es el conjunto de variables vinculadas al evento **Chronicle (C):** Un modelo *chronicle C e*s un par (*S, T*) donde *S e*s un conjunto de eventos y *T* el conjunto de restricciones entre las ocurrencias entre ellos

Example of chronicle:

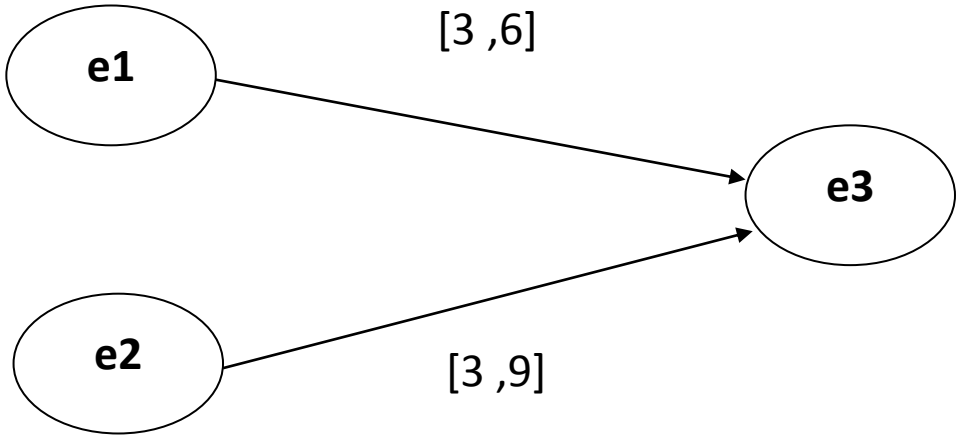
S = *{* (*acquire*−(?*nb*), ?*t*1),
(*orderBN*−(?*nb*), ?*t*2),
(*orderBN*+(?*nbReturned*), ?*t*3),
(*nbExpected* = *nbReturned*?−
(?*nbReturned*), ?*t*4),
(*happy*(), ?*t*5)
}

and

T = *{*?*t*1 <?*t*2, ?*t*2 <?*t*3, ?*t*3 <?*t*4, ?*t*4 <?*t*5*}*



Ejemplo de Crónica según Dousson



$$\boldsymbol{C} = \{S,\ T\}$$

$$S = \{e1,\ e2,\ e3\}$$

$$T = \{t1 < t3 \ avec \ \ 3 \leq t3 - t1 \leq 6\ \}$$
$$\{t2 < t3 \ avec \ \ 3 \leq t3 - t2 \leq 9\ \}$$

Lenguaje de Crónicas

Predicado	Significado
event(E, T)	Evento E ocurre en el momento T
event(F:(?V1,?V2),T)	Un evento ocurre en el momento T cambiando valor de la propiedad F de ?V1 a ?V2
noevent(E, (T1,T2))	Evento E no ocurre entre [T1,T2)
noevent(F:(?V1,?V2), (T1,T2))	No hay eventos entre [T1,T2) que cambie los valores de la propiedad F de ?V1 a ?V2
hold(F:?V, (T1,T2))	El valor de la propiedad F es ?V entre [T1,T2)
occurs(N,M,E,(T1,T2))	Evento E ocurre al menos N veces y a lo sumo M veces entre [T1,T2)

Lenguaje de Crónicas

Chronicle Model {

Events{

event(e1, T1), event(e2, T2), event(e3, T3) **}**

Constrains{

T2-T1 < C1

T3-T2 < C2 **}**

When recognized{

action1

action2 **}}**

- **ei** proposiciones atemporales que representan actividades (eventos) .
- **Ti** puntos de ocurrencias de eventos.
- **Constrains:** conjunto de restricciones entre los Ti.
- **Ci:** constantes representando la diferencia de tiempo entre la ocurrencia de dos eventos.
- **Actions:** conjunto de acciones a ejecutar cuando la crónica es reconocida

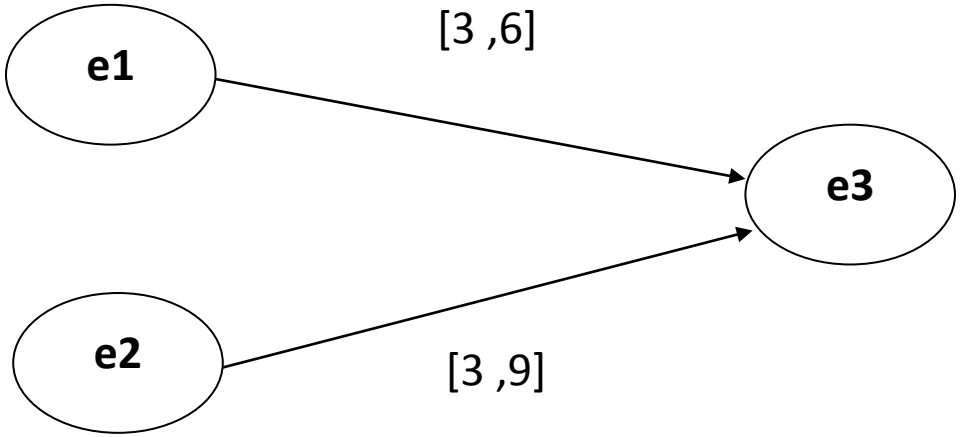
Crónica de Dousson

Lenguaje de crónicas:

Conjunto de Predicados *(event, noevent, occurs, hold)*

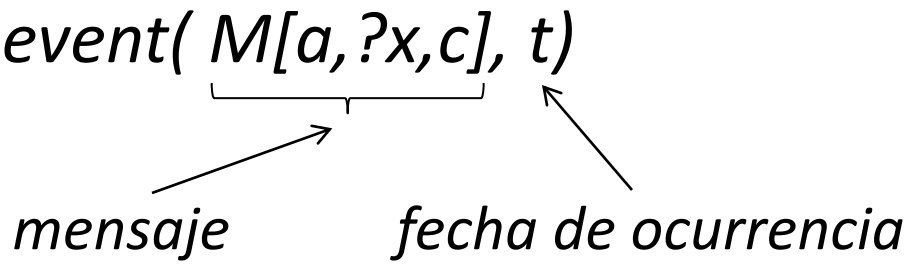
Chronicle C
{
event (e1, t1)
event (e2, t2)
event (e3, t3)

t3 – t1 in [3, 6]
t2 – t1 in [3, 9]
}

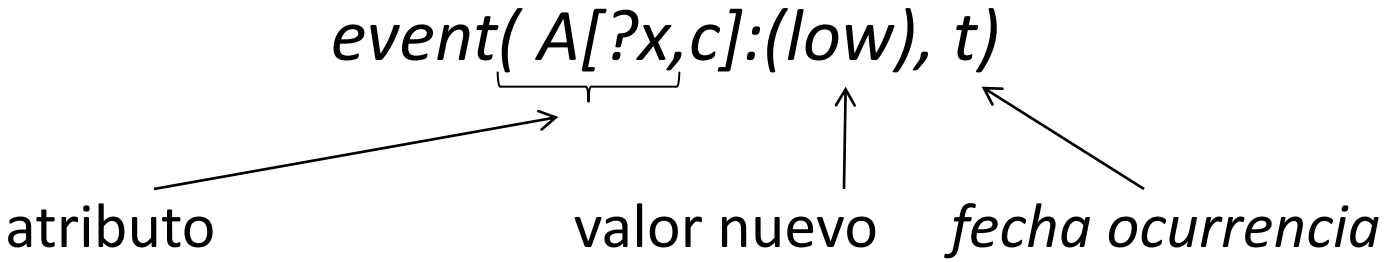


Crónica: Ocurrencia de Eventos

- Evento “Signal”



- Evento “Change”



Crónica

- **Ausencia de eventos**

  noevent(M[a,?x,], (t1, t2))*

  noevent(A[?x,]:(low), (t1, t2))*

- **Persistencia de valores**

  hold(A[?x,c]:low, (t1, t2))

- ***HOLD es equivalente a:***

  event(A[?x,c]:(low), t0)

  noevent(A[?x,c]:(¬ low), (t0, t2))

  t0 ≤t1

Ejemplo de Crónica

- **Escenario :** la batería cae en **estado critico** 2 a 5 unidades de tiempo después de una perdida de conectividad (pasa de buena a mala) y después de menos de 10 unidades de tiempo de haber comenzado con un alto nivel de paquetes IP perdidos (solo valido para conexiones WIFI)

chronicle myChronicle {

event(SOC:(?,critical),t1)

event(link:(good,bad),t2)

event(IPloss:(?,high),t3)

hold(mode:WiFi,(t1,t4))

t2 - t1 in [2,5]

t2 < t4

t3 - t1 in [0,10]

t3 < t4

}

Ejemplo de Crónica

Eventos

Puntual = (entrada vehículo: planificada) y
(salida vehículo: planificada)

```
chronicle punctual[?id, ?vehicle](T1) {
        event( stop enter[?id, ?vehicle, scheduled], T0 )
        event( stop leave[?id, ?vehicle, scheduled], T1 )
        T1 > T0
        end - start in [1, 2000]
    }
```

Ejemplo de Crónica

No puntual = (entrada vehículo: tarde) o
(salida vehículo: tarde) o
(salida vehículo: temprano)

chronicle non punctual[?id, ?vehicle]() {
event(stop enter[*, *, late], T0)
}

chronicle punctuality change[?id, ?vehicle, non punctual](T1) {
event(punctual[?id, ?vehicle], T0)
event(non punctual[?id, ?vehicle], T1)
T1 > T0
noevent(punctual[?id, ?vehicle], (T0+1, T1))
noevent(non punctual[?id, ?vehicle], (T0+1, T1))
end - start in [1, 20000]

Reconocimiento en las Crónicas

chronicle LUD {

event(link:(bad, good), t1)

event(link:(good, bad), t2)

hold(link:good, (t1, t2-1))

0 < t2 - t1 < 5 -- up/down within 5s

when recognized emit event(LUD, t2)

}

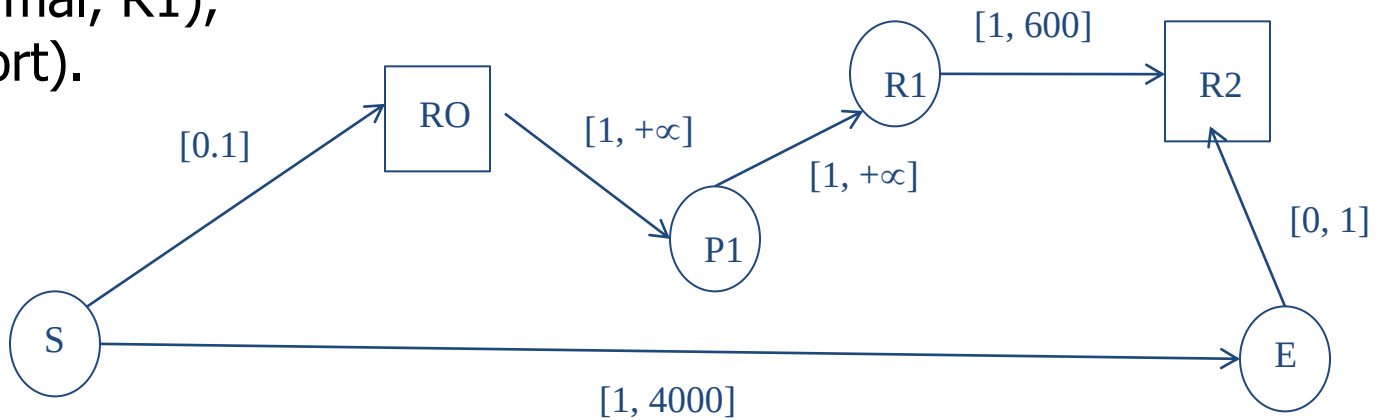
Genera un nuevo evento deducido "LUD”

Crónica

(Carrault et al., 1999), (Quiniou et al., 2001), (Carrault et al. 2003)	• Expresan crónicas usando **Prolog,** • Las relaciones temporales entre eventos son inferidas por el lenguaje de razonamiento usado: **inductive logic programming** (ILP). • Crónicas son clausulas en lógica de predicado de primer orden.

```
bigeminism :-
   qrs(R0, abnormal, _),
   p_wave(P1, normal, R0),
   qrs(R1, normal, P1),
   qrs(R2, abnormal, R1),
   rr(R1, R2, short).
```

Ejemplo de una crónica llamada bigeminism



Crónica

Representación que se propone en (Quiniou et al., 2001) para imágenes medicas.

- Para cada evento se asocia un tipo (P o QRS), el tiempo de aparición en el ECG, y una calificación de la forma de la onda (normal o anormal).

- Ellos usan el predicado siguiente en prolog para codificar esa información:

  wave(Event, Type, Time, Qual, Pre_event),

- Lo que significa que evento es una onda de tipo Type (p ó q) con un tiempo de ocurrencia Time, la forma es Qual (normal o anormal) y Pre_event es el evento que precede al evento en el ECG.

Crónica

- Ejemplo de ECG en prolog:

Bigeminy:-

wave(p1, p, 651, normal, null).
wave(r5, qrs, 4577, normal, p3).
wave(r1, qrs, 836, normal, p1).
wave(r6, qrs, 5086, abnormal, r5).
wave(r2, qrs, 1357, abnormal, r1).
wave(p4, p, 6279, normal, r6).
wave(p2, p, 2528, normal, r2).
wave(r3, qrs, 2686, normal, p2).
wave(r4, qrs, 3203, abnormal, r3).
wave(p3, p, 4428, normal, r4).
end(model(bigeminy_119_1)).

Crónicas Distribuidas

- **Conjunto de eventos E={E_1, …, E_2, …., E_p}, distribuidos entre los n sitios de un sistema distribuido**

- **En general, los eventos de un sitio pueden definir una crónica en ese sitio, tal que una crónica dada este definida por n sub-crónicas distribuidas entre n posibles sitios.**

Crónicas Distribuidas (Aguilar et al., 2013)

Definición 1: "Una crónica se puede descomponer en n-subcrónicas, tal que cada sub-crónica SCi este asignada a un sitio Pi del sistema distribuido, y describe un sub-conjunto de eventos Eac_i, donde Tac_i son restricciones temporales, que deben ocurrir en el sitio i para que la crónica sea reconocida".

$$\mathbf{C(E,T) = UNION_{i=1,\,n}(SC_i(Eac_i, Tac_i))}$$

donde,

- Eac_i y Tac_i son el conjunto de eventos y restricciones temporales de las cronicas asignadas a cada sitio i,
- $Eac_i = \{E_k,,..., E_l\}$, $Tac_i = \{T_k,,..., T_l\}$ y $E_{k, ..., } E_l \epsilon E$, Tk , …, Tl ϵT y esos eventos Ek, …, El ocurren en el sitio i.
- UNION es un predicado definido por la unión del conjunto de eventos y el conjunto de restricciones temporales distribuidos en las n subcrónicas.

Crónicas Distribuidas (Aguilar et al., 2013)

- **Definición 2. Binding Events (BE):** son eventos desde las subcrónicas, para conectarlas a otras sub-crónicas, y así representar la comunicación entre subcrónicas. Un BE es instanciado cuando una subcrónica vecina (de un sitio vecino) es reconocida y entonces se propaga a las otras subcrónicas. Así, el reconocimiento (evento de salida) de una subcrónica SC_i puede ser enlazada al evento Be_j que pertenece a la subcrónica SC_j.

- **Definición 3. Crónica Distribuida:** "una clásica crónica C descompuesta en un conjunto de sub-crónicas SC_i, enlazadas entre si a través de BE"

Crónicas Distribuidas (Aguilar et al., 2013)