Álgebra del Tiempo Diario

Víctor de Buen

20 de febrero de 2007

Resumen

En este documento se dan las principales bases teóricas para la implementación del **álgebra del tiempo diario** especialmente diseñada para el lenguaje de programación *open-source* TOL (*Time Oriented Language*) cuya página web es TOL-project `http://www.tol-project.org`, y que ha sido desarrollado por el grupo Bayes `http://www.bayesforecast.com`.

El propósito del álgebra del tiempo es dotar al lenguaje de un método potente y sencillo de manejar que permita la manipulación de secuencias temporales complejas utilizadas en el análisis estadístico de series temporales en tiempo discreto con base diaria, especialmente en lo que se respecta al llamado tiempo social que alude a todos aquellos aspectos de tipo social que pueden influir en los fenómenos analizados, como son los festivos, los puentes, las vacaciones, el calendario escolar, fines de semana, estacionalidades anual, semanal, y diaria, usos horarios de invierno y verano, nocturnidad, horario de oficina, etc.; o cualquier otra estructura temporal derivada de los mismos.

Se introducen formalizaciones matemáticas de instante de tiempo y conjunto temporal, así como de las operaciones algebraicas que es posible definir sobre ellos. Nótese que las propiedades básicas que se van a exponer servirían igualmente para cualquier otra discretización en lugar de la diaria, y no necesariamente a intervalos regulares. Queda para un posterior documento la extensión al tiempo continuo.

Índice

1. Instantes de Tiempo Diario

Los **instantes de tiempo con base diaria**, a los que llamaremos indistintamente fecha ó IT de aquí en adelante, se definen como días completos en el calendario gregoriano y se identifican con el punto de la recta real correspondiente al instante inicial del día, es decir las 0 horas 0 minutos, 0 segundos, 0 milésimas de segundo, etc. Si situamos el 0 en un punto de origen cualquiera y definimos la unidad en la recta real como 24 horas se puede identificar con los números enteros el conjunto de los instantes de tiempo diario $\mathfrak{C}_D$ mediante la función biyectiva

$$I(x):\ \mathfrak{C}_D \to \mathbb{Z} \tag{1}$$

que devuelve índice de la fecha, es decir, la posición relativa al punto de origen en número de días trasncurridos desde el mismo. Evidentemente existe su función inversa

$$I^{-1}(n):\ \mathbb{Z} \to \mathfrak{C}_D \tag{2}$$

En virtud de estas funciones se definen de forma trivial las operaciones de traslación diaria de fechas

$$\left.\begin{array}{l} x+n = I^{-1}\left(I(x)+n\right) \\ x-n = I^{-1}\left(I(x)-n\right) \end{array}\right\} \forall x \in \mathfrak{C}_D,\ n \in \mathbb{Z} \tag{3}$$

así como las relaciones de orden usuales

$$\left.\begin{array}{l} x = y \Leftrightarrow I(x) = I(y) \\ x < y \Leftrightarrow I(x) < I(y) \\ x > y \Leftrightarrow I(x) > I(y) \end{array}\right\} \forall x, y \in \mathfrak{C}_D \tag{4}$$

El conjunto $\mathfrak{C}_D$ se amplía con los instantes límite impropios$\{-\infty, \infty\}$ que se usará para extender los conceptos de sucesor y predecesor como se verá más adelante, así como con el IT desconocido o indeterminado ?, el cual se utilizará para expresar que no existe o no es calculable el resultado de una expresión que debería devolver un IT.

Sobre los IT se definirán mediante el calendario gregoriano las funciones componentes de la fecha

$$\begin{array}{l} Y_x = year(x) = \in \mathbb{Z} \\ M_x = month(x) = 1 \ldots 12 \\ D_x = monthday(x) = 1 \ldots 31 \\ W_x = weekday(x) = 1 \ldots 7 \end{array} \tag{5}$$

También se define la función de construcción de fechas a partir de sus componentes

$$x = YMD(y, m, d) \Leftrightarrow Y_x = y \ \wedge\ M_x = m \ \wedge\ D_x = d \tag{6}$$

Nos referimos aquí al objeto matemático fecha que se identifica con un número entero en contraposición con el objeto informático que interesa almacenar en base a sus componentes para mayor eficiencia.

2. Conjuntos Temporales Discretos

Un **conjunto temporal discreto en base diaria**, en adelante **CT**, es un tipo especial de conjunto de instantes de tiempo con cierta estructura y que cumple determinadas normas que se explican a continuación. Más adelante se exponen algunas propiedades importantes así como algunas operaciones algebraicas que se pueden definir entre CT's y también otras que relacionan CT's e IT's.

2.1. Definición y propiedades de un CT

Un CT es un conjunto ordenado o sucesión de instantes de tiempo propios. A la familia de todos los conjuntos temporales le llamaremos $\mathcal{CT}_D$. Nótese que la cardinalidad de $\mathcal{CT}_D$ será por tanto la misma que la del conjunto de las partes de los números enteros $\mathcal{P}(\mathbb{Z})$, es decir, la potencia del continuo ∞^∞.

2.1.1. Funciones de sucesión simple de un ITen un CT

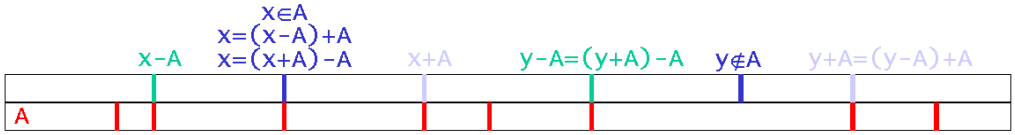
Se define el IT **sucesor de un IT en un CT**, como el menor elemento de CT estrictamente mayor que ese IT. De forma análoga se define el **predecesor de un IT en un CT**, como el mayor elemento de CT estrictamente menor que dicho IT. La notación empleada será

$$\left.\begin{array}{l} x + A = \text{mín}\,\{t \in A |\, t > x\} \\ x - A = \text{máx}\,\{t \in A |\, t < x\} \end{array}\right\} \forall x \in \mathfrak{C}_D,\ A \in \mathcal{CT}_D \tag{7}$$

Nótese que estas operaciones no son internas sino que el primer elemento es un IT y el segundo un CT, por lo que no tienen sentido las propiedades asociativa o conmutativa ni ninguna otra atribuíble a los signos de suma y resta usuales. Sin embargo sí se cumplen algunas normas bastante obvias, tal y como se desprende a la vista de la figura1 , que aún así serán de gran utilidad más adelante

$$\begin{array}{l} (x + A) - A \leq x < x + A \\ (x - A) + A \geq x > x - A \\ (x + A) - A = x \Leftrightarrow x \in A \\ (x - A) + A = x \Leftrightarrow x \in A \\ (x + A) - A = x - A < x \Leftrightarrow x \notin A \\ (x - A) + A = x + A > x \Leftrightarrow x \notin A \end{array} \tag{8}$$

Figure 1: Ejemplos de sucesión de un IT en un CT



2.1.2. Funciones de sucesión compuesta de un ITen un CT

Se puede extender recursivamente la definición de **sucesor y predecesor n-ésimos de un IT en un CT** del siguiente modo

$$\left.\begin{array}{ll} x + 1 \times A = x + A & x + n \times A = (x + (n-1) \times A) + A \\ x - 1 \times A = x - A & x - n \times A = (x - (n-1) \times A) - A \end{array}\right\} \forall n > 1,\ n \in \mathbb{Z} \tag{9}$$

El caso de $n = 0$ es realmente singular pues para que exista $x + 0 \times A = x - 0 \times A$ es necesario que se igualen las dos reglas de recursión anteriores

$$\begin{array}{l} x + 0 \times A = (x - A) + A \\ x - 0 \times A = (x + A) - A \end{array} \tag{10}$$

Si $x \in A$ entonces está claro que $(x - A) + A = x = (x + A) - A$ pero si $x \notin A$ entonces el sucesor en A del predecesor de x en A es el mismo que el sucesor de x en A, ya que no puede existir ningún elemento de A entre ambos y por tanto

$$x - A = (x + A) - A < x < (x - A) + A = x + A \tag{11}$$

Así pues no se puede definir $x + 0 \times A$ de forma congruente con las fórmulas generales, por lo que se ha toma arbitrariamente el criterio de devolver la fecha desconocida ?, es decir

$$x + 0 \times A = \left\{ \begin{array}{ll} x & si\, x \in A \\ ? & si\, x \notin A \end{array} \right. \tag{12}$$

Se pueden extender las normas anteriormente descritas en 8 para las operaciones sucesor y predecesor simples

$$\left. \begin{array}{l} (x + n \times A) - n \times A \leq x < (x + n \times A) - (n - 1) \times A \\ (x - n \times A) + n \times A \geq x > (x - n \times A) + (n - 1) \times A \\ (x + n \times A) - n \times A = x \Leftrightarrow x \in A \\ (x - n \times A) + n \times A = x \Leftrightarrow x \in A \\ (x + n \times A) - n \times A < x \Leftrightarrow x \notin A \\ (x - n \times A) + n \times A < x \Leftrightarrow x \notin A \end{array} \right\} \forall n \in \mathbb{Z}^+ \tag{13}$$

2.1.3. Diferencia de dos IT's en un CT

Se define la función diferencia en un CT $A \in \mathcal{CT}_D$ entre dos IT $x \leq y \in \mathfrak{C}_D$, como el número de IT's de ese CT superiores a x y no superiores a y, es decir

$$\nabla_A (x, y) = Card \{ t \in A \mid x \leq t < y \} \quad \forall x \leq y \in \mathfrak{C}_D,\ A \in \mathcal{CT}_D \tag{14}$$

Igualmente se pude definir como el número de IT's de ese CT inferiores a y y no inferiores a x. Es decir

$$\nabla_A (x, y) = Card \{ t \in A \mid x < t \leq y \} \quad \forall x \leq y \in \mathfrak{C}_D,\ A \in \mathcal{CT}_D \tag{15}$$

Otras formas de expresar lo mismo son

$$\begin{array}{l} \nabla_A (x, y) = \text{máx} \{ n \in \mathbb{N} \mid x + n \times A \leq y \} \\ \nabla_A (x, y) = \text{máx} \{ n \in \mathbb{N} \mid y - n \times A \geq x \} \end{array} \tag{16}$$

También se define, por simetría, la diferencia negativa

$$\nabla_A (y, x) = -\nabla_A (x, y) \tag{17}$$

Aunque las siguientes relaciones se cumplen de forma trivial pueden no resultar a veces del todo intuitivas:

1. La diferencia entre un elemento y sí mismo es nula siempre y si dos fechas tienen diferencia nula en A su sucesor o su predecesor en ese mismo A coinciden.

$$\begin{array}{l} \nabla_A (x, x) = 0 \\ \nabla_A (x, y) = 0 \Rightarrow \ \ x + A = y + A \ \ \vee\ x - A = y - A \end{array} \tag{18}$$

2. Para todo par de fechas de A su distancia es invariante a las traslaciones en A y sólo es nula si son iguales.

$$x, y \in A \Rightarrow \left\{ \begin{array}{l} \nabla_A (x, y) = \nabla_A (x \pm n \times A, y \pm n \times A) \\ \nabla_A (x, y) = 0 \Leftrightarrow x = y \end{array} \right. \tag{19}$$

3. Si la diferencia entre dos IT's cualesquiera es positiva el primero es estrictamente mayor que el primero y si es negativa entonces es estrictamente menor.

$$\begin{array}{l} \nabla_A(x,y) > 0 \Rightarrow y > x \\ \nabla_A(x,y) < 0 \Rightarrow y < x \end{array} \tag{20}$$

4. El recíproco no se cumple estrictamente pues la distancia puede anularse si no pertenecen al conjunto y no hay ningún elemento entre ambos.

$$\begin{array}{l} y > x \Rightarrow \nabla_A(x,y) \geq 0 \\ y < x \Rightarrow \nabla_A(x,y) \leq 0 \end{array} \tag{21}$$

2.1.4. Fechados y conjuntos acotados

El ínfimo y el supremo de un CT se definen como es ordinario

$$\begin{array}{l} \text{ínf}\, A = \text{máx}\,\{x \mid y \notin A \,\forall\, y \leq x\} \\ \sup A = \text{mín}\,\{x \mid y \notin A \,\forall\, y \geq x\} \end{array} \tag{22}$$

Se llamará **fechado** a un conjunto temporal $F \in \mathcal{CT}_D$ no acotado inferior ni superiormente, es decir, que cumpla que

$$\forall t \in \mathbb{Z}\, \exists\, x, y \in F \bot x < t < y \tag{23}$$

o lo que es lo mismo

$$\begin{array}{l} \text{ínf}\, F = -\infty \\ \sup F = \infty \end{array} \tag{24}$$

Los fechados tienen una gran importancia pues son la base para la creación de series temporales en tiempo continuo, así como para la definición de algunas operaciones importantes de CT's que se verán más adelante.

Aunque los instantes impropios $-\infty$, ?, ∞ no pertenecen nunca a ningún CT, a veces se obrará como si así fuera por comodidad notacional en el caso de los instantes límite $-\infty$, ∞, puesto que pertenecen asintóticamente a todos los fechados. Se puede hablar de semifechados por la izquierda o por la derecha para aquellos CT que estén acotados sólo por la izquierda o por la derecha, respectivamente.

Para los fechados se cumplen las siguientes reglas asintóticas de sucesión

$$\left.\begin{array}{l} \infty + n \times F = \infty \\ \infty - n \times F = \infty \\ -\infty + n \times F = -\infty \\ -\infty - n \times F = -\infty \end{array}\right\} \forall n \in \mathbb{Z}^+ \tag{25}$$

Para los CT's acotados en un intervalo $[a, b]$, se tiene sin embargo que

$$\left.\begin{array}{l} \infty + n \times A = \infty \\ \infty - n \times A = b \\ -\infty + n \times A = a \\ -\infty - n \times A = -\infty \end{array}\right\} \forall n \in \mathbb{Z}^+ \tag{26}$$

Para la fecha desconocida en cualquier caso siempre se tiene que

$$? \pm n \times A = ? \,\forall n \in \mathbb{Z} \tag{27}$$

2.1.5. Notas para la implementación de los métodos de cálculo asociados a CT's

Un CT se puede construir de varias formas, definiendo uno o más métodos de cálculo asociados al mismo:

- Por enumeración exhaustiva de sus elementos, siempre que sea un conjunto finito.
- Mediante una función booleana explícita de pertenencia que devuelva *cierto* para cada IT que sea elemento suyo y *falso* para el resto.
- Implícitamente, mediante un IT que le pertenezca y una regla de recursión reversible, es decir, que permita calcular cuál es el IT de ese conjunto inmediatamente superior (**sucesor**) o inferior (**predecesor**) a un IT cualquiera.

Sin embargo, será indispensable implementar al menos los métodos de inclusión $x \in A$ y sucesión simple $x \pm A$ para cada clase de CT de la forma más eficiente que sea posible aprovechando las características propias de cada uno, ya que estos métodos se utilizarán de forma masiva en conjuntos y series temporales. Siempre que sea posible y dé mayores prestaciones, también se implementarán los métodos de sucesión compuesta $x \pm n \times A$, ya que, aunque por defecto se pueden definir por recursión de sucesores y predecesores, se usarán tan a menudo que cualquier mejora producirá importantes ahorros en el tiempo de proceso.

Otro método de gran utilidad asociado a un CT y que merece la pena programar de la forma más eficiente posible, por el uso intensivo que se hace de él, es el que devuelve la lista ordenada de todos los elementos de un CT entre dos fechas, al cual denotaremos como método de extracción

$$extract\,(A, x, y) = A \cap [x, y] = \{\, t_k \in A \mid x \leq t_1 < t_2 < \cdots < t_n \leq y\} \quad \forall x \leq y \in \mathfrak{C}_D,\ A \in \mathcal{CT}_D \tag{28}$$

Cuando el método de extración sea muy costoso será conveniente almacenar los resultados en memoria porque es muy habitual que se vuelvan a requerir extracciones contenidas o iguales. Diremos entonces que el CT usa extracción con caché. Cuando un CT es una expresión resultante de otros CT's que no están referenciados por ningún otro se debe eliminar la caché de estos últimos una vez construida la caché final, ya que no volverán a ser consultadas y sería un espacio de memoria inútil. Es posible introducir un método general de control de la caché por parte del usuario mediante un parámetro que indique el tiempo mínimo por elemento de caché que justifica su uso. Si no se especifica nada en la definición un CT no usará caché y la extracción se realizará mediante el algoritmo genérico

$$\begin{aligned} &t_1 = (x - A) + A \\ &t_k = t_{k-1} + A\ \forall k = 2, 3, \ldots \wedge t_k \leq y \end{aligned} \tag{29}$$

Cada instancia de conjunto temporal deberá implementar también los métodos ínf A, sup A y ∇A. Si no es posible calcular analíticamente dichas propiedades se implementará al menos cotas aproximadas de los mismos que se utilizarán para optimizar los cálculos de sucesión en determinados casos patológicos que se estudiarán más adelante.

3. CT's primarios

Los conjuntos temporales primarios son los que no requieren de otros CT's para su definición y pueden ser constantes o paramétricos si admiten diferentes formas en función de ciertos argumentos

3.1. CT's constantes

Se trata de conjuntos temporales constantes en el sentido de que se definen sin necesidad de especificar ningún parámetro. La condición de incluir los interiores de todos sus elementos se desprende de las propias definiciones de forma trivial por lo que no se demuestra en cada caso. Todas estas constantes son fechados excepto el conjunto vacío.

3.1.1. Fechado básico conjunto universal $\mathbb{U}$

Llamaremos conjunto universal C al conjunto de todos los IT posibles que en esta implementación será el conjunto $\mathbb{U} = \mathfrak{C}_D$. Las funciones de sucesión compuestas se reducen en este caso a la expresión

$$x + n \times \mathbb{U} = x + n \, \forall n \in \mathbb{Z}, \, x \in \mathbb{U} \tag{30}$$

Evidentemente se tiene que

$$\begin{array}{l} \text{ínf}\, \mathbb{U} = -\infty \\ \sup \mathbb{U} = \infty \end{array} \tag{31}$$

3.1.2. Conjunto vacío: $\emptyset$

Llamaremos $\emptyset$ al conjunto temporal vacío, es decir, al que no contiene ningún IT propio. Evidentemente no se trata de un fechado aunque tampoco se puede decir que esté acotado por lo que se hablará a veces de fechado vacío. El sucesor n-ésimo se calcula así:

$$x + n \times \emptyset = \left\{ \begin{array}{cc} -\infty & si\, n < 0 \\ ? & si\, n = 0 \\ \infty & si\, n > 0 \end{array} \right. \quad \forall n \in \mathbb{Z}, \, x \in \mathbb{U} \tag{32}$$

Por convenio se entenderá que

$$\begin{array}{l} \text{ínf}\, \emptyset = \infty \\ \sup \emptyset = -\infty \end{array} \tag{33}$$

La extracción siempre devuelve la lista vacía.

3.1.3. Fechado básico pascual: Easter

Llamaremos Easter al conjunto de todos los IT pertenecientes a los domingos de Pascua de cualquier año. La definición del domingo de Pascua es relativamente sencilla: se trata del primer domingo posterior a la primera luna llena tras el equinoccio de primavera. Sin embargo los métodos con los que la iglesia calcula las efemérides astronómicas es muy primitivo y recurre a ciertas modificaciones *ad-hoc,* por lo que no siempre coincide con la definición exacta.

Se parte de la base de existencia de una función $easter\,(year)$ que devuelve el domingo de Pascua para cualquier año:

$$x \in Easter \Leftrightarrow x = easter\,(year\,(x)) \tag{34}$$

El sucesor n-ésimo se calcula así:

$$x + n \times Easter = \left\{ \begin{array}{cc} easter\,(n + y) & si\, x < easter\,(y) \\ easter\,(1 + n + y) & si\, x \geq easter\,(y) \end{array} \right. \quad \wedge\, y = year\,(x) \tag{35}$$

Es evidente que

$$\begin{array}{l} \text{ínf}\, Easter = -\infty \\ \sup Easter = \infty \end{array} \tag{36}$$

3.2. CT acotados

Aquí se describen algunas clases de CT's finitos construidos de forma muy sencilla a partir de IT's

3.2.1. CT atómico: `Day`

Dado un IT x cualquiera se define su CT atómico como

$$Day(x) = \{x\} \tag{37}$$

3.2.2. CT monoanual: `Year`

Dado un entero cualquiera y cualquiera se define su CT monoanual como el intervalo de las fechas de dicho año

$$Year\,(Y) = \{x \in \mathbb{U} \,|\, Y_x = Y\} \tag{38}$$

3.2.3. CT intervalo entre dos IT's: `In`

Dados dos IT's a y b cualesquiera se define el CT intervalo como

$$In\,(a,b) = \{x \in \mathbb{U} \,|\, a \leq x \leq b\}\, \forall a, b \in \mathbb{U} \wedge\ a \leq b \tag{39}$$

Puede no haber límite inferior o superior haciendo $a = -\infty$ ó $b = \infty$ respectivamente. Nótese también que los CT atómicos y monoanulaes son casos particulares de intervalos, por lo que se pueden utilizar en los tres casos las siguientes implementaciones para las operaciones de sucesión y extracción

$$\begin{aligned} x + n \times In\,(a,b) &= \begin{cases} a+n-1 & \forall x+n-1 < a \\ x+n & \forall a \leq x+n-1 < b \\ \infty & \forall x+n-1 \geq b \end{cases} \\ x - n \times In\,(a,b) &= \begin{cases} b-n+1 & \forall x+n-1 > b \\ x-n & \forall a < x+n-1 \leq b \\ -\infty & \forall x+n-1 \leq a \end{cases} \end{aligned} \quad \forall x \in \mathbb{U}\, n \in \mathbb{Z}^+ \tag{40}$$

La extracción es tan sencilla que no precisa caché en ningún caso.

$$extract\,(In\,(a,b)\,, x, y) = [a,b] \cap [x,y] = [\text{máx}\,\{a,x\}\,, \text{mín}\,\{b,y\}] \tag{41}$$

Evidentemente se tiene que

$$\begin{aligned} &\text{ínf}\, In\,(a,b) = a \\ &\sup In\,(a,b) = b \end{aligned} \tag{42}$$

3.2.4. CT generado por una lista de ITs: `SetOfDates`

Dada una lista finita cualquiera de IT's, se puede construir una sucesión disjunta ordenando previamente el conjunto y eliminando los elementos repetidos para optimizar los métodos de cálculo. A esta operación le llamaremos conjunto temporal generado por una lista finita de IT's

$$SetOfDates\,(x_1, \ldots, x_N) \tag{43}$$

Al estar ordenada la lista permite la búsqueda binaria con coste$N \log N$ por lo que se puede implementar muy facilmente la función de determinación de pertenencia de una fecha cualquiera así como su sucesor y predecesor. La sucesión compuesta, una vez alcazado un elemento perteneciente a la lista se reduce a una traslación en el intervalo de índices

$$x_j + n \times SetOfDates\,(x_1, \ldots, x_N) = \begin{cases} -\infty & \forall j + n < 1 \\ x_{j+n} & \forall 1 \leq j + n \leq N \\ \infty & \forall j + n > N \end{cases} \quad \forall j = 1 \ldots N \wedge n \in \mathbb{Z} \tag{44}$$

Para el método de extracción basta con localizar los índices correspondientes a los extremos y extraer todos los interiores. Obviamente no precisa caché pues el propio CT ya es en sí mismo una caché al estar definido exhaustivamente.

Evidentemente se tiene que

$$\begin{array}{l} \text{ínf}\, SetOfDates\,(x_1, \ldots, x_N) = x_1 \\ \sup SetOfDates\,(x_1, \ldots, x_N) = x_N \\ \nabla SetOfDates\,(x_1, \ldots, x_N) = \text{máx}_{k=2\ldots N}\,\{x_k - x_{k-1}\} \end{array} \tag{45}$$

3.3. Fechados básicos de CT's

Los *fechados básicos* son los creados mediante funciones que devuelven conjuntos temporales a partir de un parámetro dado, a las que llamaremos *funciones básicas de CT's*. Todos estos conjuntos temporales son fechados básicos parciales, en el sentido de que no recubren la recta real, y que se definen por cada valor particular que puede tomar determinada coordenada gregoriana. Puesto que todos ellos son fechados se tiene que

$$\begin{array}{l} \text{ínf}\, F = -\infty \\ \sup F = \infty \end{array} \tag{46}$$

3.3.1. Month(M)

La función **Month** devuelve el conjunto temporal de todos los IT incluidos en un mes del año dado para cualquier año pasado, presente o futuro

$$x \in M\,(m) \Leftrightarrow M_x = m \quad \forall m = 1 \ldots 12 \tag{47}$$

Las funciones se sucesión simple se calculan directamente mediante estas sencillas reglas

$$\begin{array}{l} x + M\,(m) = \begin{cases} x + 1 & \forall d < LDM\,(m) \\ YMD\left(Y_x + n - \left[\frac{m-1}{M_x}\right], m\right) & \forall d = LDM\,(m) \end{cases} \\ x - M\,(m) = \begin{cases} x - 1 & \forall d > LDM\,(m) \\ YMD\left(Y_x - n + \left[\frac{M_x - 1}{m}\right], m\right) & \forall d = 1 \end{cases} \end{array} \tag{48}$$

donde $\left[\frac{a}{b}\right]$ representa la parte entera del cociente de números naturales.

3.3.2. Day(D)

La función **Day** devuelve el conjunto temporal de todos los IT incluidos en un día de mes dado para cualquier mes y cualquier año pasado, presente o futuro

$$x \in D\,(d) \Leftrightarrow D_x = d \quad \forall d = 1 \ldots 31 \tag{49}$$

Las funciones de sucesión compuesta se calculan así

$$\begin{array}{l} x + n \times D\,(d) = YMD\left(Y_x + \left[\frac{m}{12}\right],\ (M_x + m)\ \mathbf{mod}\, 12, d\right) \wedge m = n - \left[\frac{d-1}{D_x}\right] \\ x - n \times D\,(d) = YMD\left(Y_x - \left[\frac{m}{12}\right],\ (M_x - m)\ \mathbf{mod}\, 12, d\right) \wedge m = n - \left[\frac{D_x - 1}{d}\right] \end{array} \tag{50}$$

3.3.3. WeekDay(W)

La función **WeekDay** devuelve el conjunto temporal de todos los IT incluidos en una día de la semana dado para cualquier semana pasada, presente o futura

$$x \in WD\,(w) \Leftrightarrow W_x = w \quad \forall w = 1 \ldots 7 \tag{51}$$

Las funciones de sucesión compuesta se calculan así

$$\begin{array}{l} x + n \times WD\,(w) = x + (w - W_x) \bmod 7 + 7\left(n - \left[\frac{w-1}{W_x}\right]\right) \\ x - n \times WD\,(w) = x - (W_x - w) \bmod 7 - 7\left(n - \left[\frac{W_x-1}{w}\right]\right) \end{array} \tag{52}$$

4. CT's secundarios

Son aquellos que se definen en función de otros CT's según ciertas reglas particulares de cada caso dando estructura y expresibilidad al álgebra del tiempo.

4.1. Operaciones Booleanas

Las siguientes operaciones entre conjuntos temporales le dan a $\mathcal{CT}_D$ una estructura de álgebra de Boole. Con ellas y otras más que se verán más adelante se pueden construir expresiones que den forma a las secuencias temporales más complicadas que se pueden encontrar en el análisis de series temporales.

4.1.1. Diferencia de CT's

La diferencia de dos conjuntos temporales A y B es el conjunto temporal de las fechas de A que no están en B.

$$x \in A - B \;\; \Leftrightarrow x \in A \wedge \cdots \wedge x \notin B \tag{53}$$

Se define el complementario de un CT como la diferencia del conjunto universal y dicho CT.

$$\overline{A} \in \mathbb{U} - A \tag{54}$$

La diferencia de fechados distintos es un fechado. Las funciones de sucesor tienen el mismo problema de recursión que en la intersección.

$$x + (A - B) = \begin{cases} x + A & si\ x + A \notin B \\ (x + A) + (A - B) & si\ x + A \in B \end{cases} \tag{55}$$

$$x - (A - B) = \begin{cases} x - A & si\ x + A \notin B \\ (x - A) - (A - B) & si\ x + A \in B \end{cases} \tag{56}$$

El método de extracción consiste como es lógico en extraer las listas de A y B y recorrer la de A eliminando los que estén en B, y almacenando los resultados en la caché.

El ínfimo y el supremo no están determinados para la diferencia de CT's pero claramente se tiene que

$$\begin{array}{l} \text{ínf}\,(A - B) \geq \text{ínf}\,A \\ \sup\,(A - B) \leq \sup B \end{array} \tag{57}$$

También se dispone de las siguientes reglas particulares

$$\begin{array}{l} A - A = \emptyset \\ A - \mathbb{U} = \emptyset \\ A - \emptyset = A \end{array} \tag{58}$$

$$\begin{array}{cc} M(m) - M(m') = M(m) & \forall m \neq m' \\ D(d) - D(d') = D(d) & \forall d \neq d' \\ WD(w) - WD(w') = WD(w) & \forall w \neq w' \end{array} \tag{59}$$

4.1.2. Unión de CT's

La unión de dos o más conjuntos temporales es la reordenación de los elementos de todos ellos. Obviamente, la unión de fechados es también un fechado.

$$x \in A_1 \cup \cdots \cup A_N \quad \Leftrightarrow x \in A_1 \vee \cdots \vee x \in A_N \tag{60}$$

El sucesor de un IT en la unión de CT's es el mínimo de los sucesores en cada uno y el predecesor es el máximo.

$$\begin{array}{l} x + (A_1 \cup \cdots \cup A_N) = \text{mín}\{x + A_1, \cdots, x + A_N\} \\ x - (A_1 \cup \cdots \cup A_N) = \text{máx}\{x - A_1, \cdots, x - A_N\} \end{array} \tag{61}$$

No existe ninguna forma de acelerar la sucesión compuesta pero sí la extracción que se ha de implementar con el conocido algoritmo *merge-sort* para mezcla de listas ordenadas. El trabajo realizado hará aconsejable usar caché de extracción en la mayoría de los casos.

El ínfimo y el supremo están perfectamente definidos para la unión de CT's

$$\begin{array}{l} \text{ínf}\,(A_1 \cup \cdots \cup A_N) = \text{mín}_{k=1\ldots N}\{\text{ínf}\,A_k\} \\ \sup(A_1 \cup \cdots \cup A_N) = \text{máx}_{k=1\ldots N}\{\sup A_k\} \end{array} \tag{62}$$

4.1.3. Intersección de CT's

La intersección de dos o más conjuntos temporales es la reordenación de los elementos que pertenecen a todos ellos simulaténeamente.

$$x \in A_1 \cap \cdots \cap A_N \quad \Leftrightarrow x \in A_1 \wedge \cdots \wedge x \in A_N \tag{63}$$

La intersección de fechados puede ser un fechado o no serlo. La intersección de fechados no disjuntos es un fechado.

El sucesor simple de un IT en la intersección de CT's es algo más complicado que el de la unión y sólo se puede definir recursivamente

$$\begin{array}{c} Y^+ = \left\{ y_k^+ = x_k + A_k \middle| \, \forall k = 1 \ldots N \right\} \quad Z^+ = \left\{ y_k^+ \middle| \, y_k^+ \in A_j \forall j = 1 \ldots N \right\} \\ x + (A_1 \cap \cdots \cap A_N) = \left\{ \begin{array}{cc} \text{mín}\,(Z^+) & si\ Z^+ \neq \emptyset \\ \text{máx}\,(Y^+) + (A_1 \cap \cdots \cap A_N) & si\ Z^+ = \emptyset \end{array} \right. \end{array} \tag{64}$$

$$\begin{array}{c} Y^- = \left\{ y_k^- = x_k - A_k \middle| \, \forall k = 1 \ldots N \right\} \quad Z^- = \left\{ y_k^- \middle| \, y_k^- \in A_j \forall j = 1 \ldots N \right\} \\ x - (A_1 \cap \cdots \cap A_N) = \left\{ \begin{array}{cc} \text{máx}\,(Z^-) & si\ Z^- \neq \emptyset \\ \text{mín}\,(Y^-) - (A_1 \cap \cdots \cap A_N) & si\ Z^- = \emptyset \end{array} \right. \end{array} \tag{65}$$

Este caracter recursivo da lugar a un ciclo infinito cuando el resultado de una intersección es acotado o vacío. Por ello se debe disponer de un mecanismo que analize simbólicamente las expresiones para evitar esa situación. En cualquier caso sería conveniente dar un mensaje de advertencia cuando se sobrepasara determinado número de iteraciones considerado como excesivo. También sería bueno parar el ciclo y dar un mensaje de aviso a partir de un número máximo de iteraciones. Ambos límites deberían ser definibles por el usuario.

Para la extracción sin embargo no hay ningún problema pues se procederá a la extracción de cada uno los CT's a intersecar

$$extract\,(A_k, a, b) \ = \{t_{k,1}, \ldots, t_{k,n_k}\} \tag{66}$$

y a continuación se eliminan aquellos elementos que no pertenezcan a alguno de esos CT's, lo cual no requiere recursión aunque sí el esfuerzo suficiente para usar caché. Se puede recorrer

la lista de uno cualquiera y buscar cada uno de sus elementos en las listas de todos los demás. Concretamente se debe usar el que dé coste computacional mínimo

$$\text{mín}_{k=1\ldots N} \left\{ n_k \prod_{j \neq k} \log_2 n_j \right\} \tag{67}$$

El ínfimo y el supremo no están determinados para la intersección de CT's pero sí existen acotaciones genéricas útiles

$$\begin{array}{c} \text{ínf}\left(A_1 \cap \cdots \cap A_N\right) \geq \text{máx}_{k=1\ldots N}\left\{\text{ínf}\, A_k\right\} \\ \sup\left(A_1 \cap \cdots \cap A_N\right) \leq \text{mín}_{k=1\ldots N}\left\{\sup A_k\right\} \end{array} \tag{68}$$

También se cumplen las siguientes reglas

$$\begin{array}{c} \mathbb{U} \cap A = A \cap A = A \\ \emptyset \cap A = \emptyset \end{array} \tag{69}$$

$$\overline{A_1 \cap \cdots \cap A_N} = \overline{A_1} \cup \cdots \cup \overline{A_N} \tag{70}$$

$$\begin{array}{cc} M\left(m\right) \cap M\left(m'\right) = \emptyset & \forall m \neq m' \\ D\left(d\right) \cap D\left(d'\right) = \emptyset & \forall d \neq d' \\ WD\left(w\right) \cap WD\left(w'\right) = \emptyset & \forall w \neq w' \end{array} \tag{71}$$

$$\begin{array}{cc} M\left(m\right) \cap D\left(d\right) \neq \emptyset & \forall m = 1, 3 \ldots 12 \wedge d = 1 \ldots 31 \\ M\left(2\right) \cap D\left(d\right) \neq \emptyset & \forall d = 1 \ldots 29 \\ M\left(2\right) \cap D\left(30\right) = \emptyset & \\ M\left(m\right) \cap D\left(31\right) = \emptyset & \forall m = 2, 4, 6, 9, 11 \\ M\left(m\right) \cap WD\left(w\right) \neq \emptyset & \forall m = 1 \ldots 12 \wedge w = 1 \ldots 7 \\ D\left(d\right) \cap WD\left(w\right) \neq \emptyset & \forall d = 1 \ldots 30 \wedge w = 1 \ldots 7 \end{array} \tag{72}$$

4.2. Operadores de Traslación en un CT

Los operadores de traslación de conjuntos construyen un CT mediante desplazamientos de IT's a lo largo de otro CT.

4.2.1. Conjuntos periódicos

La operación modular de un IT c llamado centro, con periodicidad $p > 0$ en el conjunto de unidades de traslación U tal que $c \in U$ devuelve todos los IT de U cuya distancia en U a c sea múltiplo de dicha periodicidad p.

$$c \,\mathbf{mod}\, p \times U = \left\{u \in U \mid \exists n \in \mathbb{Z} \perp u = c + \left(p \cdot n\right) \times U\right\} \tag{73}$$

o dicho de otro modo

$$c \,\mathbf{mod}\, p \times U = \left\{u \in U \mid \nabla_A\left(u, c\right) = 0 \,\mathbf{mod}\, p\right\} \tag{74}$$

Si $c \notin U$ no se pueden definir los métodos de forma sencilla y resulta de poca utilidad por lo que se tomará como centro $c + U$. Si $p = 0$ entonces $c \,\mathbf{mod}\, 0 \times U = \emptyset$ y si $p < 0$ se invertirá el signo.

El sucesor y el predecesor simples son obviamente

$$\begin{array}{rcl} x + \left(c \,\mathbf{mod}\, p \times U\right) & = & \left(x + U\right) + \nabla_A\left(x + U, c\right) \,\mathbf{mod}\, p \\ x - \left(c \,\mathbf{mod}\, p \times U\right) & = & \left(x - U\right) - \nabla_A\left(c, x - U\right) \,\mathbf{mod}\, p \end{array} \tag{75}$$

El sucesor y el predecesor compuesto se pueden acelerar haciendo uso de la siguiente propiedad

$$y \in c \,\mathbf{mod}\, p \times U \Rightarrow y + k \times \left(c \,\mathbf{mod}\, p \times U\right) = y + \left(k\Delta p\right) \times U \;\forall k \in \mathbb{Z} \tag{76}$$

La extracción debe usar caché porque el cálculo de la sucesión compuesta en U puede ser costoso. Se extraen los elementos u_k de U en el intervalo extendido

$$extract\,(c\,\mathbf{mod}\,p\times U,a,b)\Rightarrow\ a'=\min\{a,c\}\wedge b'=\max\{b,c\} \tag{77}$$

Se busca la posición de $c=u_j$ y se seleccionan los elementos de la forma $u_{(j\,\mathbf{mod}\,p)+h\Delta k}$

Obviamente los conjuntos periódicos no pueden estar acotados y son siempre fechados por lo que el ínfimo y el supremo son infinitos

$$\begin{aligned}\inf(c\,\mathbf{mod}\,p\times U)&=\begin{cases}-\infty & \forall p\neq 0\\ c & p=0\wedge c\in U\\ \infty & p=0\wedge c\notin U\end{cases}\\ \sup(c\,\mathbf{mod}\,p\times U)&=\begin{cases}\infty & \forall p\neq 0\\ c & p=0\wedge c\in U\\ -\infty & p=0\wedge c\notin U\end{cases}\end{aligned} \tag{78}$$

Obsérvese que los fechados de los días de la semana son casos particulares de fechados periódicos

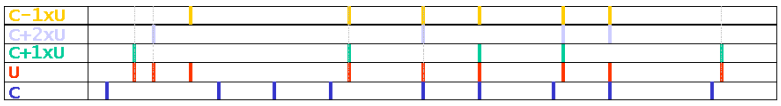
$$WD\,(w)=(c_0+w)\ \mathbf{mod}\,7\times\mathbb{U}\ \forall w=1\ldots 7 \tag{79}$$

donde c_0 es un domingo cualquiera

4.2.2. Sucesor de un CT en otro

Se define el conjunto sucesor n-ésimo de un conjunto C, en otro U al conjunto de todas las fechas sucesores n-ésimas en el conjunto U, al que llamaremos de unidades de traslación, de elementos del conjunto C, al que denominaremos centro de traslaciones. En la figura 2 se pueden ver algunos ejemplos de sucesión de CT's

Figure 2: Ejemplos de sucesión de un CT en otro



Todo esto se expresa así

$$\left.\begin{aligned}C+n\times U&=\{x\in U|\,\exists c\in C\perp x=c+n\times U\}\\ C-n\times U&=\{x\in U|\,\exists c\in C\perp x=c-n\times U\}\end{aligned}\right\}\forall n\in\mathbb{Z}^+ \tag{80}$$

Llevando al límite nulo ambas expresiones se obtiene

$$C+0\times U=C-0\times U=\{x\in U|\,\exists c\in C\perp x=c-0\times U=c+0\times U\}\ \ =C\cap U \tag{81}$$

De las relaciones expuestas en 13 se deduce directamente que

$$\begin{aligned}x=c+n\times U&\Leftrightarrow x-n\times U\leq c<x-(n-1)\times U\\ x=c-n\times U&\Leftrightarrow x+n\times U\geq c>x+(n-1)\times U\end{aligned} \tag{82}$$

por lo que se puede disponer de los siguientes métodos directos de determinación de la pertenencia

$$\left.\begin{aligned}C+n\times U&=\{x\in U|\,x-n\times U\leq(x-(n-1)\times U)-C\}\\ C-n\times U&=\{x\in U|\,x+n\times U\geq(x+(n-1)\times U)+C\}\end{aligned}\right\} \tag{83}$$

Partiendo directamente de la definición de sucesor y predecesor simple dada en 7 se pueden definir para cualquier IT x

$$\left.\begin{array}{l} x+(C+n\times U)=\text{mín}\,\{\,z=c+n\times U|\,c\in C\wedge z>x\} \\ x+(C-n\times U)=\text{mín}\,\{\,z=c-n\times U|\,c\in C\wedge z>x\} \\ x-(C+n\times U)=\text{máx}\,\{\,z=c+n\times U|\,c\in C\wedge z<x\} \\ x-(C-n\times U)=\text{máx}\,\{\,z=c-n\times U|\,c\in C\wedge z<x\} \end{array}\right\} \tag{84}$$

En cada uno de los cuatro casos anteriores se cumple la siguiente regla respectivamente

$$\left.\begin{array}{l} u=x+U\in C+n\times U\Rightarrow x+(C+n\times U)=u \\ u=x+U\in C-n\times U\Rightarrow x+(C-n\times U)=u \\ v=x-U\in C+n\times U\Rightarrow x-(C+n\times U)=v \\ v=x-U\in C-n\times U\Rightarrow x-(C-n\times U)=v \end{array}\right\} \tag{85}$$

Queda por resolver qué ocurre cuando u ó v no pertenecen a $C\pm n\times U$, es decir, según el caso

$$\begin{array}{l} u\notin C+n\times U\Rightarrow (u-(n-1)\times U)-C<u-n\times U<u-(n-1)\times U \\ \Rightarrow x+(C+n\times U)=((u-n\times U)+C)+n\times U \end{array} \tag{86}$$

$$\begin{array}{l} u\notin C-n\times U\Rightarrow (u+(n-1)\times U)+C>u+n\times U>u+(n-1)\times U \\ \Rightarrow x+(C-n\times U)=((u+(n-1)\times U)+C)-n\times U \end{array} \tag{87}$$

$$\begin{array}{l} v\notin C+n\times U\Rightarrow (v-(n-1)\times U)-C<v-n\times U<v-(n-1)\times U \\ \Rightarrow x-(C+n\times U)=((v-(n-1)\times U)-C)+n\times U \end{array} \tag{88}$$

$$\begin{array}{l} v\notin C-n\times U\Rightarrow (v+(n-1)\times U)+C>v+n\times U>v+(n-1)\times U \\ \Rightarrow x-(C-n\times U)=((v+n\times U)-C)-n\times U \end{array} \tag{89}$$

Al igual que con la mayoría de las operaciones de CT's, no existe ninguna forma de acelerar la sucesión compuesta pero sí la extracción, que habrá de tener caché. Esta se implementará extrayendo previamente los u_k de los elementos de U y los c_k de los elementos de C en el intervalo extendido $[a',b']$ definido así

$$\begin{array}{l} extract\,(C+n\times U,a,b)\Rightarrow\ a'=a-n\times U\wedge b'=b \\ extract\,(C-n\times U,a,b)\Rightarrow\ a'=a\wedge b'=b+n\times U \end{array} \tag{90}$$

Se recorre la extracción u_k de los elementos de U con lo que $u_k\pm h\times U=u_{k\pm h}$ y se usa búsqueda binaria para el cálculo de $c_j=u_{k\pm(n-1)}\pm C$, reduciéndose drásticamente el coste computacional, especialmente cuando U y C son a su vez expresiones complejas de CT's.

El ínfimo y el supremo no están determinados para el sucesor de un CT en otro pero sí se sabe que

$$\begin{array}{l} \text{ínf}\,(C+n\times U)\geq\text{máx}\,\{(\text{ínf}\,C)+n\times U,\text{ínf}\,U\} \\ \sup(C+n\times U)\leq\text{mín}\,\{(\sup C)+n\times U,\sup U\} \\ \text{ínf}\,(C-n\times U)\geq\text{máx}\,\{(\text{ínf}\,C)-n\times U,\text{ínf}\,U\} \\ \sup(C-n\times U)\leq\text{mín}\,\{(\sup C)-n\times U,\sup U\} \\ \text{ínf}\,(C\pm 0\times U)=\text{ínf}\,(C\cap U)\geq\text{máx}\,\{\text{ínf}\,C,\text{ínf}\,U\} \\ \sup(C\pm 0\times U)=\sup(C\cap U)\leq\text{mín}\,\{\sup C,\sup U\} \end{array} \tag{91}$$

También se cumple la siguiente regla para la intersección con cualquier CT A

$$A\cap U=\emptyset\Rightarrow A\cap(C\pm n\times U)=\emptyset\quad\forall n\in\mathbb{Z} \tag{92}$$

4.2.3. Rango de sucesores de un CT en otro

La operación rango de sucesores se basa en la anterior operación de sucesión de CT's y se define como la unión de sucesores consecutivos en un rango de números enteros

$$C + [r, s] \times U = \cup_{k=r}^{s} (C + k \times U) \ \forall r \leq s \in \mathbb{Z} \tag{93}$$

Como tanto la unión como la sucesión ya han sido exploradas se puede decir que se trata de una operación no básica en el álgebra del tiempo, aunque es bueno definir los métodos *ad hoc* para aprovechar los cálculos realizados pues se pueden evitar la mitad de los cálculos ya que el $u_{k\pm(n-1)}$ de una iteración será el $u_{k\pm n}$ de la anterior o la siguente según el caso.

Puesto que se trata de la unión de sucesores, el ínfimo y el supremo se calculan como

$$\begin{array}{l} \text{ínf}\,(C + [r, s] \times U) = \text{mín}_{k=r\ldots s}\,\{\text{ínf}\,(C + k \times U)\} \\ \sup\,(C + [r, s] \times U) = \text{máx}_{k=r\ldots s}\,\{\sup\,(C + k \times U)\} \\ \text{ínf}\,(C - [r, s] \times U) = \text{mín}_{k=r\ldots s}\,\{\text{ínf}\,(C - k \times U)\} \\ \sup\,(C - [r, s] \times U) = \text{máx}_{k=r\ldots s}\,\{\sup\,(C - k \times U)\} \end{array} \tag{94}$$

También se cumple la siguiente regla para la intersección con cualquier CT A

$$A \cap U = \emptyset \Rightarrow C + [r, s] \times U = \emptyset \quad \forall r, s \in \mathbb{Z} \tag{95}$$

5. Tests de calidad del álgebra del tiempo

La complejidad algorítmica del álgebra temporal descrita requiere la implementación de una serie de tests de integridad de los diferentes métodos de cualquier CT de los anteriormente descritos así como cualquier expresión compuesta definible mediante los operadores de forma anidada.

5.1. Tests de coherencia de un CT concreto

En general el método más sencillo de programar es el de la pertenencia por lo que lo más sensato es comprobar que el resto de métodos son coherentes con el mismo.

Así pues, primero se tomarán todas las fechas entre dos límites dados y se seleccionará de forma exhaustiva la lista de las fechas que pertenecen al CT mediante el método de cáclulo directo de $x \in A$.

Después se llamará al método particular de extracción en ese mismo intervalo y se comprobará que efectivamente da lugar a exáctamente la misma lista que el método exhaustivo.

Si no existe ninguna fecha en ese intervalo no hay nada más que comprobar.

Si hay al menos dos fechas en el intervalo se calculará progresivamente la lista de sucesores a partir de la primera extraída para comprobar que el método $x + A$ va dando exáctamente el siguiente elemento de la lista extraída.

Análogamente se calculará regresivamente la lista de predecesores a partir de la última extraída para comprobar que el método $x - A$ va dando exáctamente el anterior elemento de la lista extraída.

Por último se generarán fechas aletatorias en el intervalo del test, no necesariamente de A, y se comprobará que el resultado del sucesor y el predecesor coinciden con el método de búsqueda binaria en la lista extraída.

Dada la lentitud que tendría un programa TOL que hiciese este test se ha desarrollado en C++ la función built-in

Text TestIntegrityOfTimeSet
(TimeSet tms
[, Date from=DefFirst, Date until=DefLast,
Real numSim=1000, Real maxLag=1])

Esta función devuelve el texto OK si todo ha ido bien o bien un texto avisando de que ninguna fecha del intervalo le pertenece o en su caso un mensaje detallando la incoherencia detectada.

5.2. Simulación aleatoria de expresiones

En el fichero TOL SimulationMassiveTestOfTimeSetCoherence.tol se dispone de un sistema de generación aleatoria de expresiones de conjuntos temporales del nivel de complejidad que se desee e incorporando todas las funciones y objetos de tipo TimeSet que existen en el álgebra del tiempo.

Se trata de un conjunto de funciones que construyen expresiones TOL de texto que son unas de tipo básico para construir CT's primarios, es decir que no dependen de otros CT's para su definición (WD(3), M(8), ...), y otras funciones mutuamente recursivas que devuelven expresiones de CT's secundarios ((WD(3)+M(8))*D(1), ...).

Para aumentar la complejidad sin restar legibilidad e imitar el proceso de análisis usual de series temporales se simula un bloque de variables de forma que se puedan usar en las fórmulas las variables previamente creadas.