AVATAR: Razonando sobre la Semántica de los Contenidos de TV

En este documento describimos el marco donde tiene lugar el proceso de razonamiento semántico llevado a cabo en AVATAR sobre las descripciones de los contenidos de TV —proporcionadas mediante metadatos TV-Anytime—, y sobre las preferencias de los usuarios —contenidas en los perfiles de los espectadores, tal como describimos en el documento Doc1 disponible en *http://avatar.det.uvigo.es*.

1. El marco del razonamiento semántico en AVATAR

Los elementos clave del marco de razonamiento en AVATAR son: (i) la base de conocimiento del sistema, esto es, la ontología OWL implementada sobre el dominio de la TV, (ii) los perfiles de usuario, (iii) el lenguaje de consulta LIKO que permite acceder a la jerarquía de clases y propiedades establecida en la ontología y (iv) una clasificación de las relaciones semánticas que pueden ser inferidas, empleando para ello los algoritmos cuyo pseudocódigo mostraremos en esta sección.

1.1. La ontología de TV

Para conseguir un sistema recomendador de TV personalizado basado en razonamiento semántico, necesitamos organizar el conocimiento de nuestro dominio de aplicación, en este caso, la TV. En el campo de la Web Semántica, las ontologías son una de las tecnologías de uso más extendido para este propósito. Por esta razón, en nuestra propuesta hemos implementado una ontología sobre el dominio de la TV, empleando el lenguaje OWL.

Así, nosotros podemos representar las diferentes entidades relacionados con los contenidos audiovisuales mediante la inclusión de clases en nuestra ontología. Hemos definido una clase raíz en la jerarquía de conocimiento, llamada *TV Contents*, de la cual dependen el resto de clases de la ontología (por ejemplo, *Movies*, *TV_Series*, *Informative_Programs*, *Sport_Programs*, etc.). También hemos incluido clases referidas a los créditos que están relacionados con los programas de TV, como presentadores, directores o actores, entre otros (clases *Presenters*, *Directors*, *Actors*, *Actresses*, *Producers*, *ScreenWriters*, etc.).

Los programas más especializados se definen como subclases de aquellos que están más próximos al elemento raíz. De esta forma, tenemos subclases como *Action_Movies*, *Terror_Series*, *News*, *Soccer_Broadcasting*, *Movie_Directors*, *Starring_Actors*, *Supporting_Actors*, etc.). La ontología también contiene propiedades —organizadas jerárquicamente— para definir características distintivas de los programas de TV, y para establecer relaciones entre las clases de la base del conocimiento. Dado que OWL permite declarar diferentes tipos de propiedades, nuestra ontología almacena propiedades funcionales, transitivas, inversas o simétricas, favoreciendo así los procesos inferenciales necesarios en el enfoque propuesto.

Además de la definición de las clases y propiedades, necesitamos popular la base de conocimiento, para disponer de información concreta sobre la que aplicar los preceptos de nuestro razonamiento semántico. Para ello, será necesario recurrir a la programación televisiva, extrayendo instancias concretas de las clases de nuestra ontología.

1.2. Los perfiles de usuario

Los perfiles de usuario en AVATAR serán ontologías OWL, construidas incrementalmente a medida que el sistema infiere (o recibe) información sobre el comportamiento de visionado de los espectadores. Recordar que los perfiles almacenarán una parte estática —relativas a los datos personales del usuario facilitados cuando el éste se da de alta en el sistema— y una parte dinámica, llamada PSC referida a las preferencias del espectador en relación a programas de TV. Una descripción más detallada del proceso de formación y actualización de los *perfiles-ontología* está disponible en el documento Doc1 en *http://avatar.det.uvigo.es*.

1.3. LIKO: Language for Inferring Knowledge from Ontologies

Para poder llevar a cabo el proceso de razonamiento sobre la semántica de los programas declarados como instancias en la ontología del sistema, y sobre las preferencias de los usuarios, AVATAR requiere un lenguaje que permita acceder y consultar la base de conocimiento del sistema. A tal efecto, hemos desarrollado el lenguaje LIKO, cuyos operadores resumimos en la Tabla 1. Las consultas típicas que permite LIKO incluyendo extraer superclases o subclases a partir de una clase dada, superpropiedades y subpropiedades a partir de una propiedad, o extraer propiedades con determinadas restricciones sobre su rango o su dominio.

1.4. Clasificación de relaciones semánticas

En esta sección definiremos los diferentes tipos de relaciones semánticas que descubrirá el sistema entre las instancias declaradas en la ontología de TV. Para ello, el sistema toma como punto de partida las clases y propiedades definidas en el *perfil-ontología* del usuario, y extrae aquellas otras relacionadas con las primeras desde un punto de vista semántico. Empecemos definiendo este tipo de relación semántica mediante los siguientes conceptos:

- **Nexo semántico entre propiedades:** Sean p_i y p_j dos propiedades. Diremos que existe un nexo semántico entre ellos (representado como $p_i \leftrightarrow p_j$), si se cumple(n) alguna(s) de esta(s) condicion(es):

- p_i y p_j son la misma propiedad.
- p_i es superpropiedad directa de p_j[1] en la jerarquía definida en la ontología.
  De este modo, la condición indicada antes, podemos expresarla como $p_j \sqsubseteq_D p_i$.

[1] De entre todas las propiedades que "subsumen" a p_i (representadas como *Properties* $\sqsubseteq p_i$), sólo las superpropiedades directas son consideradas en este análisis. Para identificar esta relación, usaremos el operador $\sqsubseteq_D$.

Tabla 1. Operadores de consulta del lenguaje **LIKO**

Operators	**Input**	**Output**
$\triangleright$	Una clase de la ontología.	Subclases directas relativas a la clase de entrada.
$\triangleleft$	Una clase de la ontología.	Superclases directas relativas a la clase de entrada.
$>>$	Una propiedad de la ontología.	Subpropiedades directas relacionadas con la propiedad de entrada.
$<<$	Una propiedad de la ontología.	Superpropiedades directas referidas a la propiedad de entrada.
$>\Leftarrow$	Instancias de clases de la ontología.	Propiedades donde el dominio es una instancia de la clase de entrada.
$>\Rightarrow$	Instancias de class de la ontología.	Propiedades donde el rango es una instancia de la clase de entrada.
$>\odot$	Instancias de clases o valores concretos de propiedades *Datatype*.	Propiedades inferidas de las propiedades transitivas contenidas en la ontología.
$>\oplus$	Instancias de clases o valores concretos de propiedades *Datatype*.	Propiedades inferidas de las propiedades inversas contenidas en la ontología.
$>\hookleftarrow$	Instancias de clases o valores concretos de propiedades *Datatype*.	Propiedades inferidas de las propiedades simétricas contenidas en la ontología.
$>\ominus$	Una propiedad definida en la ontología.	Propiedad inversa de la propiedad de entrada.

- p_j es la superpropiedad directa en la jerarquía, de p_i.
- $\exists$ q, tal que $p_i, p_j \sqsubseteq_D q$: Es decir, p_i y p_j son propiedades “hermanas”.
- $\exists$ r y q, tales que $p_i \sqsubseteq_D r \sqsubseteq_D q$ y $p_j \sqsubseteq_D q$: Es decir, p_j es propiedad “hermana” de una superpropiedad directa de p_i.
- $\exists$ r y q, tales que $p_j \sqsubseteq_D r \sqsubseteq_D q$ y $p_i \sqsubseteq_D q$: Es decir, p_j es subpropiedad directa de una propiedad “hermana” de p_i.
- p_i es la inversa de p_j (denotamos la inversa de p_j, como $\neg p_j$).

En la notación introducida antes, esta definición sería:

$$
\begin{array}{lll}
p_i \leftrightarrow p_j & \Leftrightarrow & p_i = p_j \;\vee\; p_j \sqsubseteq_D p_i \;\vee\; p_i \sqsubseteq_D p_j \;\vee \\
& & \exists\, \mathrm{q} \,/\, p_i, p_j \sqsubseteq_D q \;\vee \\
& & \exists\, \mathrm{r,q} \,/\, p_i \sqsubseteq_D r \sqsubseteq_D q \;\wedge\; p_j \sqsubseteq_D \mathrm{q} \;\vee \\
& & \exists\, \mathrm{r,q} \,/\, p_j \sqsubseteq_D r \sqsubseteq q \;\wedge\; p_i \sqsubseteq_D \mathrm{q} \vee \\
& & p_i = \neg p_j
\end{array}
$$

- **Nexo semántico entre clases:** Al igual que en el caso de las propiedades, definimos el concepto de nexo semántico entre las clases C_i y C_j ($C_i \leftrightarrow C_j$) como:

$$\begin{array}{ll} C_i \leftrightarrow C_j \Leftrightarrow C_i = C_j \;\; \vee \;\; C_j \sqsubseteq_D C_i \;\; \vee \;\; C_i \sqsubseteq_D C_j \;\; \vee \\ \qquad\qquad \exists\, Q^* \neq \textit{Ontology Root} \;\; / \; C_i, C_j \sqsubseteq_D Q^* \;\; \vee \\ \qquad\qquad \exists\, A, B \,/\, C_i \sqsubseteq A \subseteq_D B \;\; \wedge \;\; C_j \sqsubseteq B \;\; \vee \\ \qquad\qquad \exists\, A, B \,/\, C_j \sqsubseteq_D A \sqsubseteq_D B \;\; \wedge \;\; C_i \sqsubseteq_D B \end{array}$$

Es necesario aclarar que $\mathbf{Q}^*$ tendrá que ser necesariamente diferente del elemento raíz de la ontología, para poder declarar que $C_i \leftrightarrow C_j$. De lo contrario, estaríamos diciendo que contenidos muy diferentes entre sí – pero todos correspondientes a clases hijas del elemento raíz *TV Contents* – estarían relacionados semánticamente.

Una vez que hemos establecido los criterios para descubrir relaciones entre clases y propiedades, nos centramos en las asociaciones establecidas entre las diferentes instancias declaradas en la ontología del sistema. Empecemos considerando que dos instancias de clases definidas en una ontología, pueden estar relacionadas **explícitamente** a través de propiedades *Object* declaradas en la base de conocimiento, o bien puede haber **relaciones implícitas** entre ellas, a través de instancias de otras clases. Para referirnos a relaciones semánticas *explícitas*, emplearemos la notación C_1 [p] C_2, siendo p una propiedad *Object* declarada en la ontología, y C_1 y C_2 las clases a las que pertenecen las instancias entre las que se establece la relación. El descubrimiento de relaciones *implícitas* u "ocultas" será el objetivo del recomendador **AVATAR** durante el proceso de razonamiento semántico, ya que éste puede conducir a recomendaciones mejoradas con un alto grado de personalización.

Pongamos un ejemplo para mostrar los dos tipos de relaciones semánticas que hemos mencionado. Tomemos dos propiedades *Object* declaradas en la ontología de TV entre instancias de las clases *Starring Actors* y *Action Movies*.

"Dirty Harry" $\leftarrow$ hasStarringActor $\rightarrow$ "Clint Eastwood"
"Mystic River" $\leftarrow$ hasDirector $\rightarrow$ "Clint Eastwood"

A la vista de dichas propiedades es fácil ver que la instancia "Dirty Harry" está relacionada *explícitamente* con "Clint Eastwood", por ser este último el actor protagonista de la película (ídem para la otra relación). Sin embargo, también existe una relación *implícita* entre las películas "Dirty Harry" y "Mystic River", ya que en ambas participa Clint Eastwood, como actor y director, respectivamente.

A continuación, definimos las *relaciones semánticas implícitas* que manejará nuestro enfoque. Entre paréntesis figuran los operadores que permiten identificar a cada una de ellas, de forma unívoca.

- **Cadena semántica directa ó *Direct semantic chain*** ($\Rightarrow$):Está formado por varias relaciones semánticas *explícitas* de la forma:
  $D_0 \;\; [p_0] \;\; D_1 \;\; [p_1] \;\; D_2 \;\; ... \;\; D_{q-1} \;\; [p_{q-1}] \;\; D_q$
  Este tipo de relación tiene su origen en el principio de transitividad (si $a \rightarrow b \wedge b \rightarrow c$ entonces $a \rightarrow c$), y permite expresar la existencia de una relación semántica entre instancias de las clases D_0 y D_q. La notación empleada para expresar esta asociación semántica será $D_0 \Rightarrow D_q$.
  Dada la *direct semantic chain* $\mathcal{DSC}$, definimos la longitud de la misma – *Lenght* ($\mathcal{DSC}$) – como el número de clases involucradas en dicha cadena. Además nos referiremos a cada una de dichas clases como $\mathcal{DSC}$ $[i]$, con 0 $\leq$ i $<$ *Length* ($\mathcal{DSC}$).

- **Cadenas semánticas directas con nodos comunes ó *Direct semantic chains with common nodes*** ($\Rightarrow_{Nodes}$): Consideremos dos *cadenas semánticas directas*, tales que en ambos participan instancias pertenecientes a una misma clase (en adelante, a este conjunto de clases comunes les llamaremos *Nodes*). Así,
  $\mathcal{DSC}_1 = E_0 \;\; [p_0] \;\; E_1 \;\; [p_1] \;\; \mathbf{N} \;\; ... \;\; \mathbf{M} \;\; [p_{m-1}] \;\; E_m$
  $\mathcal{DSC}_2 = D_0 \;\; [q_0] \;\; \mathbf{M} \;\; [q_1] \;\; D_2 \;\; ... \;\; \mathbf{N} \;\; [q_{n-1}] \;\; D_n$
  $DSC_1 \sqcap DSC_2 = \{Nodes\} = \{\mathbf{M},\mathbf{N}\}$
  Debido a la existencia de la primera cadena semántica, el sistema puede descubrir la relación semántica entre la instancia de la clase E_0 y la instancia de la clase E_m, esto es, $E_0 \Rightarrow E_m$; y por la misma razón, $D_0 \Rightarrow D_n$. Además por el hecho de haber, en este caso, nodos comunes entre ambas cadenas semánticas, también se establece una relación semántica entre E_0 y D_0, que expresaremos mediante la notación $E_0 \Rightarrow_{M,N} D_0$. Cabe destacar que las instancias presentes en las cadenas mostradas arriba pueden ser iguales o distintas, es decir, en este ejemplo podemos tener dos instancias diferentes de la clase **M** (una en cada *cadena semántica directa*), o bien podría tratarse de la misma instancia en las dos relaciones (ídem para la clase **N**).
- **Cadenas semánticas directas mixtas ó *Mixed direct semantic chains*** ($\Rightarrow_{\Rightarrow_{Nodes}}$): Es una combinación de las dos relaciones semánticas descritas antes. Así, si una instancia de la clase C_1 está relacionada con otra de la clase C_2 mediante una cadena semántica directa con nodos comunes ($C_1 \Rightarrow_{Nodes} C_2$), y a la vez C_2 está relacionada con otra instancia de C_3 mediante una cadena semántica directa ($C_2 \Rightarrow C_3$), es claro que hay una relación entre las instancias C_1 y C_3. Esto se debe al principio de transitividad mencionado antes. A este tipo de asociación semántica la denominaremos cadena semántica directa mixta, por combinar dos relaciones directas (de ahí que su operador sea la combinación de los operadores de las dos relaciones anteriores, $\Rightarrow$ y $\Rightarrow_{Nodes}$).
- **Cadenas semánticas indirectas ó *Indirect semantic chains*** ($\approx$): Sean DSC_1 y DSC_2 dos cadenas semánticas directas, de longitudes l_1 y l_2, respectivamente, con $l_1 \leq l_2$:
  $\mathcal{DSC}_1 = E_0 \;\; [p_0] \;\; E_1 \;\; [p_1] \;\; E_2 \;\; ... \;\; [p_{l_1-2}] \;\; E_{l_1-1}$
  $\mathcal{DSC}_2 = F_0 \;\; [q_0] \;\; F_1 \;\; [q_1] \;\; F_2 \;\; ... \;\; [q_{l_2-2}] \;\; F_{l_2-1}$
  Se dice que existe una *cadena semántica indirecta* entre las instancias de las clases E_0 y F_0 (denotado como $E_0 \approx F_0$), si se cumple que las propiedades involucradas en $\mathcal{DSC}_1$ y $\mathcal{DSC}_2$, están relacionadas mediante nexo semántico. Dado que las longitudes de ambas cadenas semánticas directas pueden ser diferentes, la condición de nexo semántico entre propiedades debe verificarse hasta que se termine la cadena más corta de ambas. Es decir, para las dos cadenas semánticas directas presentadas arriba, se cumplirá $E_0 \approx F_0 \Leftrightarrow p_i \leftrightarrow q_i \; \forall$ i, con $0 \leq$ i $\leq l_1 - 2$.
- **Cadenas semánticas virtuales ó *Virtual semantic chains*** ($\approx_{\Rightarrow}$, $\approx_{\Rightarrow_{Nodes}}$): Las cadenas semánticas indirectas tal como han sido presentadas aquí, pueden ayudar a descubrir otro tipo de asociaciones “escondidas” en la base de conocimiento del sistema. Al igual que hicimos en el caso de los *cadenas semánticas mixtas*, combinando los *cadenas semánticas directas* y *cadenas semánticas con nodos comunes*, podemos definir una nueva asociación semántica , que llamaremos *cadena semántica virtual*, y que puede ser de dos tipos: (i) uno resultante de la combinación

de las *cadenas semánticas directas* y las *cadenas semánticas indirectas* (operador $\approx_{\Rightarrow}$), que denominaremos **tipo I**, y (ii) un segundo resultante de la combinación de los *cadenas semánticas directas con nodos comunes* y las *cadenas semánticas indirectas* (operador $\approx_{\Rightarrow_{Nodes}}$), denominado **tipo II**.
En la Fig 1 se pueden apreciar ejemplos de algunas de las relaciones semánticas que acabamos de presentar.

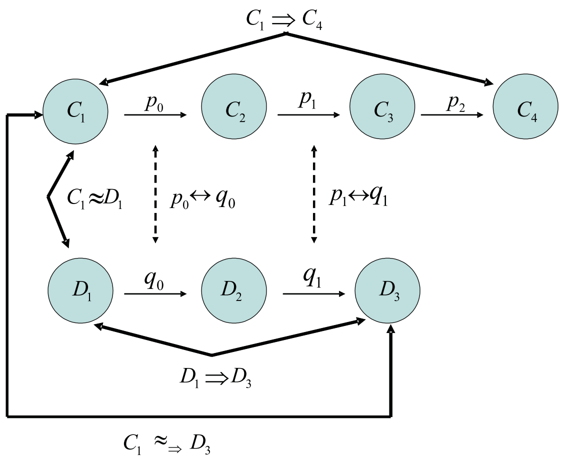

Figura 1. Relaciones semánticas inferidas por AVATAR

2. Inferencia de relaciones semánticas

Antes de presentear el conjunto de algoritmos diseñados para inferir las relaciones semánticas definidas en la sección anterior, es preciso describir la notación que vamos a emplear:

- Las relaciones semánticas se establecen entre instancias de clases definidas en la ontología. Sin embargo, para representar dichas relaciones emplearemos la clase a la que pertenecen las instancias, acompañadas del operador que identificará de forma unívoca, cada relación/asociación semántica descubierta. Por ejemplo, el sistema puede inferir una relación semántica de tipo *cadena semántica directa*, entre el concurso "Wheel of Fortune" y el director de cine "Woody Allen", por haber participado este último en el concurso. **AVATAR** representará esta relación como *Quiz Shows* $\Rightarrow$ *Movie Director*.

- Sea *Rel* una de las relaciones semánticas descritas en la Secc. 1.4. Denotaremos con *Dom* (*Rel*) a la clase de la izquierda y con *Range* (*Rel*) a la clase de la derecha. Por ejemplo, si *Rel* = $(A \Rightarrow B)$, entonces *Dom* (*Rel*) = A y *Range* (*Rel*) = B.

- Denotaremos por $|Set|$ la cardinalidad del conjunto de elementos (clases o propiedades) *Set*.

Los algoritmos desarrollados parten de las clases y propiedades definidas en el *perfil-ontología* del usuario (documento Doc1 en *http://avatar.det.uvigo.es*). Como primer paso, calcularán el conjunto de clases *target classes* ($\mathcal{TC}_{C_i}$) y propiedades *target properties* ($\mathcal{TP}_{C_i}$) relevantes en el proceso de razonamiento semántico:

- $\mathcal{TC}_{C_i}$: Contendrá las clases que estén relacionadas mediante *nexo semántico* con la clase C_i definida en el $\mathcal{PSC}$ del usuario (ver Secc. 1.4).

- $\mathcal{TP}_{C_i}$: Contendrá las propiedades relacionadas con el conjunto de propiedades asociado a la clase C_i en el $\mathcal{PSC}$ del *perfil-ontología* del usuario. Estas propiedades son:

- Propiedades relacionadas mediante *nexo semántico* con las propiedades del perfil.
- Propiedades inversas de las anteriores. Dichas propiedades son incluidas en el proceso de razonamiento e inferencia de asociaciones, dado que entre una propiedad y su inversa hay una estrecha relación desde un punto de vista semántico (e.g. si para el sistema es relevante la *Object* property "hasDirector", también lo será "DirectorOf").

2.1. Algoritmo Target_Classes

Este algoritmo recibirá como entrada cada una de las clases C_i definidas en el $\mathcal{PSC}$ del usuario. El resultado devuelto será el conjunto $\mathcal{TC}_{C_i}$, formado por las clases relacionadas mediante *nexo semántico* con C_i.

El pseudocódigo del algoritmo se muestra en Algoritmo 1, y es fácil de entender si recordamos la definición de *nexo semántico entre clases* dada en Secc. 1.4. Así, en el primer bloque (B1), se obtiene un conjunto (*Sub*$_{C_i}$) de todas las subclases directas de C_i, utilizando para ello el operador **LIKO** $\triangleright$. En B2 se emplea el operador $\triangleleft$ para computar las superclases directas asociadas a C_i (*Super*$_{C_i}$). En B3 se realiza el cálculo de las clases "hermanas" de C_i para lo cual se explora el conjunto de clases padre de C_i (*Super*$_{C_i}$) y se extrae las subclases de las mismas, mediante el operador $\triangleright$. Para obtener el conjunto de clases "hermanas" de C_i (*Siblings*$_{C_i}$), bastará con extraer del conjunto obtenido la propia clase C_i. En B4, el algoritmo computa las clases "hermanas" de las superclases de C_i. El procedimiento es similar al anterior: se computan las superclases relativas a las superclases de C_i (*Super*$_{T_j^{C_i}}$), se aplica el operador **LIKO** $\triangleright$ para obtener las subclases del conjunto resultante y, finalmente se extraen las superclases de C_i. Finalmente, B5 computa las subclases relativas a las clases "hermanas" de C_i, aplicando de nuevo el operador $\triangleright$ sobre el conjunto de propiedades "hermanas" de C_i.

2.2. Algoritmo Target_Properties

Este algoritmo recibirá entrada el conjunto de propiedades asociadas a las instancias de la clase C_i, definida en el $\mathcal{PSC}$ del usuario. A este conjunto nos referiremos en adelante como R_{C_i}. El resultado devuelto será el conjunto $\mathcal{TP}_{C_i}$, presentado antes.

El pseudocódigo del algoritmo ***Target_Properties*** se muestra en Algoritmo 2. En él, se calculan las propiedades relacionadas mediante *nexo semántico* (ver Secc. 1.4) con

---

Algoritmo 1

Target_Classes (C_i){

Input: $C_i \in$ *Concepts*

Output: $\mathcal{TC}_{C_i}$

$$\left.\begin{array}{l} Sub_{C_i} = (C_i \triangleright) = \{S_0^{C_i}, S_1^{C_i}, ..., S_X^{C_i}\} \\ \mathcal{TC}_{C_i} = Sub_{C_i} \end{array}\right\} B1$$

$$\left.\begin{array}{l} Super_{C_i} = (C_i \triangleleft) = \{T_0^{C_i}, T_1^{C_i}, ..., T_Y^{C_i}\} \\ \mathcal{TC}_{C_i} = \mathcal{TC}_{C_i} \bigcup Super_{C_i} \end{array}\right\} B2$$

$$\left.\begin{array}{l} for\ (\ j = 0\ to\ Y\ )\ \{ \\ \quad Siblings_{C_i} = (T_j^{C_i} \triangleright) - \{C_i\} = \{Q_0^{C_i}, Q_1^{C_i}, ..., Q_N^{C_i}\} \\ \quad \mathcal{TC}_{C_i} = \mathcal{TC}_{C_i} \bigcup Siblings_{C_i} \end{array}\right\} B3$$

$$\left.\begin{array}{l} \quad Super_{T_j^{C_i}} = (T_j^{C_i} \triangleleft) = \{J_0^{C_i}, J_1^{C_i}, ..., J_R^{C_i}\} \\ \quad \text{for (l = 0 to R)\{} \\ \qquad Siblings_{T_j^{C_i}} = (J_l^{C_i} \triangleright) - \{T_j^{C_i}\} \\ \qquad \mathcal{TC}_{C_i} = \mathcal{TC}_{C_i} \bigcup Siblings_{T_j^{C_i}} \\ \quad \} \\ \} \end{array}\right\} B4$$

$$\left.\begin{array}{l} for\ (\ t = 0\ to\ N\ )\ \{ \\ \quad Sub_{Q_t^{C_i}} = (Q_t^{C_i} \triangleright) \\ \quad \mathcal{TC}_{C_i} = \mathcal{TC}_{C_i} \bigcup Sub_{Q_t^{C_i}} \\ \} \end{array}\right\} B5$$

}

---

las propiedades definidas en el $\mathcal{PSC}$ del usuario (B1 a B5), y las inversas de las mismas (B6 mediante el operador $>\ominus$). El procedimiento a seguir es análogo al explicado en el caso de la rutina ***Target_Classes***, sustituyendo las clases por propiedades. Dado que puede haber una o varias propiedades asociadas a las instancias de la clase C_i, en el pseudocódigo mostrado en Algoritmo 2, empleamos la notación $\mathcal{TP}_{C_i}^{p_k}$, para referirnos a las *target properties* relativas a la propiedad p_k. De ahí que en B6, añadamos cada una de estas contribuciones al conjunto $\mathcal{TP}_{C_i}$ devuelto.

Algoritmo 2

Target_Properties (R_{C_i}){
  Input: R_{C_i} = $\{p_0,p_1,...,p_M\} \in$ *Constraints*
  Output: $\mathcal{TP}_{C_i}$

  for (k = 0 to $|R_{C_i}|$ *){*
    Sub_{p_k} = ($p_k >>$) = $\{s_0^k,s_1^k,...,s_X^k\}$ *B1*
    $\mathcal{TP}_{C_i}^{p_k}$ = Sub_k

    $Super_{p_k}$ = ($p_k <<$) = $\{t_0^k,t_1^k,...,t_Y^k\}$ *B2*
    $\mathcal{TP}_{C_i}^{p_k}$ = $\mathcal{TP}_{C_i}^{p_k} \bigcup Super_k$

    for (j = 0 to Y) {
      $Siblings_{p_k}$ = ($t_j^k >>$) - $\{p_k\}$ = $\{l_0^{t_j^k},l_1^{t_j^k},...,l_N^{t_j^k}\}$ *B3*
      $\mathcal{TP}_{C_i}^{p_k}$ = $\mathcal{TP}_{C_i}^{p_k} \bigcup Siblings_{p_k}$

      $Super_{t_j^k}$ = ($t_j^k <<$) = $\{q_0^{t_j^k}, q_1^{t_j^k},...,q_Z^{t_j^k}\}$
      for (n = 0 to Z){ *B4*
        $Siblings_{t_j^k}$ = ($q_n^{t_j^k}$ >>) $-$ { t_j^k }
        $\mathcal{TP}_{C_i}^{p_k}$ = $\mathcal{TP}_{C_i}^{p_k}$ $\bigcup$ $Siblings_{t_j^k}$
      }
    }

    for (m = 0 to N){
      $Sub_{l_m^k}$ = ($l_m^k >>$) *B5*
      $\mathcal{TP}_{C_i}^{p_k}$ = $\mathcal{TP}_{C_i}^{p_k}$ $\bigcup$ $Sub_{l_m^k}$ = $\{tp_0,tp_1,...tp_R\}$
    }

    for (m = 0 to R) $\mathcal{TP}_{p_k}^{C_i}$ = $\mathcal{TP}_{p_k}^{C_i} \bigcup \{tp_m >\ominus\}$ *B6*

    $\mathcal{TP}_{C_i}$ = $\mathcal{TP}_{C_i} \bigcup \mathcal{TP}_{C_i}^{p_k}$

}

2.3. Algoritmo Direct_Semantic_Chains

Uno de los argumentos de entrada de este algoritmo, será cada una de las instancias de las clases C_i definidas en el $\mathcal{PSC}$ del usuario, es decir, cada uno de los programas de TV que interesan a éste.

El resultado del algoritmo será una relación de las cadenas semánticas directas (identificadas por *chain* en el código mostrado en Algoritmo 3) inferidas por el sistema. Estas cadenas tendrán que cumplir dos condiciones: (i)la clase inicial (*chain* [0]) de la cadena debe ser o bien la clase C_i, o bien estar definidas en el conjunto $\mathcal{TC}_{C_i}$, y (ii) las propiedades que participan en las cadenas semánticas directas descubiertas, pertenecerán únicamente al conjunto $\mathcal{TP}_{C_i}$, es decir, serán propiedades relacionadas con las propiedades de la clase C_i.

Teniendo en cuenta ambas restricciones es fácil entender el algoritmo recursivo cuyo pseudocódigo mostramos en Algoritmo 3. Hagamos algunas aclaraciones al respecto. Establezcamos una comparativa entre las instancias declaradas en la ontología OWL, y una grafo donde dichos contenidos son los nodos, unidos entre sí mediante propiedades. Así, para descubrir cadenas semánticas que parten de la instancia $Prog_{C_i}$, se exploran aquellas propiedades en las que dicha instancia figura como dominio, empleando para ello el operador **LIKO** $>\Leftarrow$. De todas las propiedades resultantes, el algoritmo extraerá únicamente las presentes en el conjunto $\mathcal{TP}_{C_i}$ (recordar restricción 2), almacenándolas en Set_2. Una vez extraídas esas propiedades, el algoritmo explora los rangos de las mismas, ya que estas instancias serán las que permitan seguir avanzando por la cadena semántica, hasta que se alcance el final de la misma. La instancia que delimita el final de una cadena semántica directa, estará representada en el grafo por lo llamaremos *nodo hoja* (en el código esta situación se identifica mediante la función *is_leaf*). De este modo, una cadena semántica finaliza cuando al extraer las propiedades en las que dicho nodo figura como dominio, no se obtiene ninguna propiedad contenida en el conjunto *target properties*.

En Algoritmo 3 se realizan dos comprobaciones:

- Si la cadena semántica ha acabado, se actualizará la variable *chain*, en la que se incluirán todas las clases recorridas hasta ese momento y que forman parte de dicha cadena (en el código se utiliza la función *set_chain* para concatenar la clase actual al resto de clases que ya figuraban en la cadena semántica). Por esta razón, la variable *chain* también es pasada como argumento del algoritmo, pues en ella se mantiene actualizada la cadena semántica que se está calculando en cada momento. Además, será necesario construir la relación semántica (*Rel* en el código) y almacenarla en el conjunto devuelto por el algoritmo ($\mathcal{DSC}$).

- Si el nodo actual no es nodo hoja, aún no habremos alcanzado el final de un *cadena semántica directa*. En este caso, y en virtud de la recursividad, se invocará de nuevo el algoritmo *Direct_Semantic_Chains*, con la instancia correspondiente al nodo en el que nos encontramos. Será necesario también actualizar convenientemente la variable *chain*. En este caso bastará con concatenar al valor previo, la clase que se corresponde con el nodo antecesor del nodo actual.

Algoritmo 3

```
Direct_Semantic_Chains ( Prog_{C_i}, chain ){
  Set_1 = ( Prog_{C_i} >⇐ )
  Set_2 = Set_1 ⋂ TP_{C_i}
  Reason_About_Datatype_Properties (Set_2)
  for (k = 0 to |Set_2| − 1) {
    ancestor = Dom (Set_2 [k])
    current_node = Range ( Set_2 [k] )
    if (is_leaf (current_node)){
      set_chain (current_node, chain + ancestor + current_node)
      Rel = chain [0] ⇒ chain [Length ( dp ) − 1]
      DSC = DSC ⋃ { Rel }
    }
    else
      Direct_Semantic_Chains (current_node, chain + ancestor)
  }
}
```

2.4. Algoritmo Direct_Semantic_Chains_Common_Nodes

Este algoritmo calculará relaciones del tipo *cadenas semánticas directas con nodos comunes*. Para ello, se parte del conjunto de *cadenas semánticas directas* inferido en Algoritmo 3. El pseudocódigo es el indicado en Algoritmo 4, donde se construye la relación semántica mediante el operador $\Rightarrow_{Nodes}$, siendo *Nodes* el conjunto de clases que comparten las cadenas semánticas proporcionadas al algoritmo.

Algoritmo 4

```
Direct_Semantic_Chains_Common_Nodes (DSC_1, DSC_2) {
  Nodes = DSC_1 ⋂ DSC_2
  if (Nodes ≠ ∅)
    Rel = (DSC_1 [0]) ⇒_Nodes (DSC_2[0])
    DSCN = DSCCN ⋃ { Rel }
}
```

2.5. Algoritmo Mixed_Direct_Semantic_Chains

Esta rutina infiere *cadenas semánticas directas mixtas*. Teniendo en cuenta la definición de esta relación semántica (ver Secc. 1.4), es fácil ver que el algoritmo propuesto debe centrarse en tres elementos claves: (i) Los *cadenas semánticas directas* ($\mathcal{DSC}$) inferidas mediante el algoritmo *Direct_Semantic_Chains*. (ii) Los *cadenas semánticas directas con nodos comunes* ($\mathcal{DSCCN}$) descubiertos mediante el algoritmo *Direct_Semantic_Chains_Common_Nodes*, y (iii) el principio de transitividad.

El algoritmo recibirá como entrada x *cadenas semánticas directas* e y *cadenas semánticas directas con nodos comunes*, que mostramos a continuación:

$\mathcal{DSC}$ = $\{SC_0$,SC_1,...,$SC_{x-1}\}$, donde
SCi = *Dom* (SC_i) $\Rightarrow$ *Range* (SC_i), $0 \leq$ i < x
$\mathcal{DSCCN}$ = $\{SCCN_0$,$SCCN_1$,...,$SCCN_{y-1}\}$, donde
$SCCN_j$ = *Dom* ($SCCN_j$) $\Rightarrow_{Nodes_j}$ *Range* ($SCCN_j$), $0 \leq$ j < y

El algoritmo propuesto debe recorrer las relaciones semánticas contenidas en $\mathcal{DSC}$ y $\mathcal{DSCCN}$, comparando sus rangos y dominios de las $\mathcal{DSCCN}$, para poder inferir nuevas relaciones de acuerdo al principio de transitividad. Así, por ejemplo si tenemos las relaciones *Dom* (SC_i) $\Rightarrow$ *Range* (SC_i) y *Dom* ($SCCN_j$) $\Rightarrow_{N_j}$ *Range* ($SCCN_j$), donde se cumple que *Range* (SC_i) = *Dom* ($SCCN_j$), entonces, podemos inferir un *cadena semántica mixta* entre las clases *Dom* (SC_i) y *Range* ($SCCN_j$), tal como se indica en Algoritmo 5.

Algoritmo 5

```
Mixed_Direct_Semantic_Chains ( DSC,DSCCN ) {
  for (i = 0 to | DSC |){
    for (j = 0 to | DSCCN |){
      if (Range (SC_i) = Dom (SCCN_j)) {
        Rel = Dom (SC_i) ⇒⇒_{Nodes_j} Range (SCCN_j)
        MDSC = MDSC ⋃ { Rel }
      }
      if (Dom (SC_i) = Range (SPCN_j)){
        Rel = Dom (SPCN_j) ⇒⇒_{Nodes_j} Range (SC_i)
        MDSC = MDSC ⋃ { Rel }
      }
    }
  }
}
```

2.6. Algoritmo Indirect_Semantic_Chains

Este algoritmo inferirá *cadenas semánticas indirectas*, tomando como punto de partida las cadenas semánticas directas descubiertas mediante el pseudocódigo mostrado en Algoritmo 3.

El algoritmo propuesto debe tener en cuenta la longitud de las cadenas semánticas manejadas, ya que éstas pueden ser diferentes. En este caso, se comprobará la existencia de *nexo semántico entre propiedades* hasta alcanzar el final de la cadena semántica directa más corta. Si recordamos la definición de *relación semántica indirecta* dada en la Secc. 1.4, es fácil ver que para poder descubrir una relación semántica de este tipo, es necesario que *todas* las propiedades de ambas cadenas estén relacionadas mediante nexo semántico. En caso contrario no se puede establecer una relación indirecta entre clases. Por esta razón empleamos la combinación de *while* e *if* en el pseudocódigo mostrado en Algoritmo 6.

Algoritmo 6

```
Indirect_Semantic_Chains (DSC1, DSC2) {
  SemChain = Shortest_Direct_Semantic_Chain (DSC1, DSC2)
  i = 0
  while ((p_i ↔ q_i)  and  (i <( Length ( SemChain ) - 1)))  i ++
  if  ( i = Length ( SemChain ) - 1 )  {
     Rel = (DSC1[0]) ≈ (DSC2[0])
     ISC = ISC ⋃ {Rel}
  }
}
```

2.7. Algoritmo Virtual_Semantic_Chains

El algoritmo propuesto permite que el sistema pueda inferir *cadenas semánticas virtuales*, tipo I y tipo II, entre las diferentes instancias de clases definidas en la ontología de TV. La implementación de este algoritmo es muy similar a la explicada en el caso de los *cadenas semánticas directas mixtas* (ver Algoritmo 5), dado que también se construye combinando relaciones semánticas mediante el empleo de "asociaciones transitivas". El pseudocódigo correspondiente se muestra en Algoritmo 7. A continuación, hacemos algunas consideraciones en relación al mismo.

El algoritmo recibe como entrada **x** *cadenas semánticas directas*, **z** *cadenas semánticas directas con nodos comunes* e **y** *cadenas semánticas indirectas*.

$\mathcal{DSC}$ = $\{SC_0$,SC_1,...,$SC_{x-1}\}$, donde:
SCi = *Dom* (SC_i) $\Rightarrow$ *Range* (SC_i), $0 \leq$ i $<$ x
$\mathcal{DSCCN}$ = $\{SCCN_0$,$SCCN_1$,...,$SCCN_{z-1}\}$, donde:
$SCCNk$ = *Dom* $(SCCN_k)$ $\Rightarrow_{Nodes_k}$ *Range* $(SCCN_k)$, $0 \leq$ k $<$ z
$\mathcal{ISC}$ = $\{IC_0$,IC_1,...,$IC_{y-1}\}$, donde:
IC_j = *Dom* (IC_j) $\approx$ *Range* (IC_j), $0 \leq$ j $<$ y

En los dos primeros bloques del pseudocódigo propuesto (B1 y B2), se infieren *cadenas virtuales* tipo I (ver Secc. 1.4), resultantes de la combinación de *cadenas semánticas directas* (SC_j) y *relaciones indirectas* (IR_i), mientras que en los dos restantes, se combinan estas últimas con lo que hemos llamado *cadenas semánticas directas con nodos comunes*, resultando *cadenas virtuales* tipo II. Más concretamente, en B1 se infieren relaciones en las que el dominio pertenece a la cadena semántica directa y el rango a la cadena indirecta: *SI* **A** $\Rightarrow$ **B** y **B** $\approx$ **C** *ENTONCES* **A** $\approx_{\Rightarrow}$ **C**.

Algoritmo 7

Virtual_Semantic_Chains ($\mathcal{DSC}, \mathcal{DSCCN}, \mathcal{ISC}$) {
for (i = 0 to | $\mathcal{ISC}$ |){
for (j = 0 to | $\mathcal{DSC}$ |){
if (*Range* (SC_j) = *Dom* (IC_i)) {
Rel = *Dom* $(SC_j) \approx_{\Rightarrow}$ *Range* (IC_i) *B1*
$\mathcal{VSC} = \mathcal{VSC} \bigcup \{Rel\}$
}
if (*Dom* (SC_j) = *Range* (IC_i)){
Rel = *Dom* $(IC_i) \approx_{\Rightarrow}$ *Range* (SC_j) *B2*
$\mathcal{VSC} = \mathcal{VSC} \bigcup \{Rel\}$
}
}
for (j = 0 to | $\mathcal{SCCN}$ |){
if (*Range* $(SCCN_j)$ = *Dom* (IC_i)){
Rel = *Dom* $(IC_i) \approx_{\Rightarrow_{Nodes_j}}$ *Range* $(SCCN_j)$ *B3*
$\mathcal{VSC} = \mathcal{VSC} \bigcup \{Rel\}$
}
if (*Dom* $(SCCN_j)$ = *Range* (IC_i)){
Rel = *Dom* $(IC_i) \approx_{\Rightarrow_{Nodes_j}}$ *Range* $(SCCN_j)$ *B4*
$\mathcal{VSC} = \mathcal{VSC} \bigcup \{Rel\}$
}
}
}
}

Sin embargo, en el segundo bloque (B2) en las relaciones descubiertas figurarán los dominios de las relaciones indirectas y los rangos de las cadenas directas: *SI* **A** $\approx$ **B** y **B** $\Rightarrow$ **C** *ENTONCES* **A** $\approx_{\Rightarrow}$ **C**.

La misma explicación sería válida para el caso de los dos últimos bloques señalados en el pseudocódigo de la rutina, en los que las cadenas semánticas directas (SC_j) son sustituidos por las cadenas directas con nodos comunes ($SCCN_j$). De este modo se infieren *cadenas semánticas virtuales* tipo II.

2.8. Algoritmo do_Semantic_Recommendation

El proceso de recomendación semántica en AVATAR partirá de las instancias definidas en el *perfil-ontología* del usuario, y detectará las diferentes relaciones semánticas que hemos definido en este documento. Una vez inferidas éstas, será necesario desarrollar un mecanismo de filtrado y "ranking" de dichas relaciones, que extraiga aquellas relaciones que sean más relevantes de cara a la recomendación personalizada.

En este proceso de clasificación influirán diferentes factores, como:

- La información definida en el perfil del usuario.
- El tipo de relación que estemos clasificando, por ejemplo, las relaciones más directas serán más significativas y por tanto, más relevantes para la recomendación.
- La longitud de las relaciones inferidas.
- Campos definidos en la especificación TV-Anytime, en relación por ejemplo al tipo de audiencia a la que va dirigido un determinado contenido.
- El conocimiento de la ontología de TV que puede ayudar a la hora de descubrir *similaridad semántica* entre las instancias que participan en las relaciones inferidas.

Este proceso está siendo desarrollado actualmente.

3. Un ejemplo de inferencia de relaciones semánticas

En este sección, mostraremos un pequeño ejemplo de razonamiento e inferencia de relaciones semánticas. Para ello, partimos de la parte dinámica (o $\mathcal{PSC}$) del perfil de usuario mostrado en la Tabla 2.

Sport_Programs $\sqsupseteq$ Sport_Broadcasting $\sqsupseteq$ Formula1_Broadcasting. (hasDriver. Name [M. Schumacher])
Movies $\sqsupseteq$ Action_Movies. (hasStarringActor. Name [C. Eastwood]) $\sqcap_P$ (hasPlace [Mexico])

Tabla 2. $\mathcal{PSC}$ de un perfil de usuario en AVATAR

Supongamos que en la ontología de TV están declarados los siguientes programas de TV, como instancias de las clases que figuran entre paréntesis:

- Una película, perteneciente al género de la comedia (clase *Comedy_Movies*) donde el actor favorito del usuario, "Clint Eastwood", participa como actor secundario.

- Un concurso de TV (clase *Quiz_Shows*) presentado por una actriz de telenovelas (clase *Soap_Operas*), que ha protagonizado la comedia mencionada en primer lugar.

- Una película basada en los asesinatos de hinchas de un equipo de fútbol (instancia de la clase *Terror_Movies*) cuya trama transcurre en el estadio de fútbol “Amsterdam Arena”, producida por “Clint Eastwood”.

- Una retransmisión de un partido de fútbol del mundial 2002 (clase *Soccer_Broadcastings*), en el que jugaban, entre otros, el futbolista “Owen”, celebrado en el estadio “Yokohama”.

- Un documental deportivo (clase Sport_Documentaries) acerca de los deportistas de élite, con participaciones del piloto de Fórmula 1 “Michael Schumacher” y del futbolista “David Beckham”.

En la Fig. 2 se muestra una parte de la ontología de TV implementada[2] , en relación a los programas que acabamos de definir. En la Fig. 2 hemos considerado muchas menos propiedades de las que, en realidad, están declaradas en la ontología del sistema. Aún así, este subconjunto es suficiente para ilustrar el ejemplo de recomendación que nos ocupa. Esto también nos permite demostrar que el sistema de recomendación propuesto es capaz de inferir relaciones semánticas en la base de conocimiento, aun cuando la información proporcionada es incompleta o escasa. Como se aprecia en la figura, emplearemos rectángulos para definir los valores tomados por las propiedades *Datatype*.

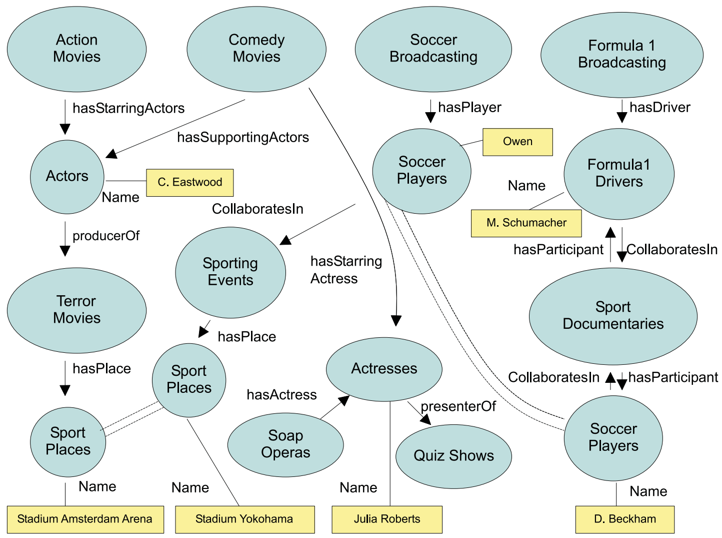

Figura 2. Subconjunto de instancias definidas en la ontología de TV

[2] En esta figura, mostramos únicamente un subconjunto reducido de instancias. La jerarquía de clases a las que pertenecen dichas instancias se ha omitido por falta de espacio.

Veamos en qué medida son importante los programas que hemos representado en la Fig. 2, de cara a una recomendación personalizada. Tal como se ha descrito en el algoritmo *do_Semantic_Recommendation*, el primer paso será calcular el conjunto de clases y propiedades de interés a la hora de razonar (*target_classes* y *target_properties*), a partir de la información del $\mathcal{PSC}$ del usuario. Así, el algoritmo *Target_Classes*, permite obtener las clases relevantes para el proceso de recomendación, asociadas a las clases C_0 y C_1 del $\mathcal{PSC}$. En nuestro ejemplo, éstas serían:

- Siendo C_0 = {F1_Broadcasting}, el algoritmo infiere $\mathcal{TC}_{C_0}$, formado por *Formula1_Broadcasting* y *Soccer_Broadcasting* (clase hermana de C_0).

- Siendo C_1 = {Action_Movies}, el algoritmo obtiene las clases relacionadas con éstas mediante nexo semántico (ver Secc. 1.4). Éstas serían la propia clase C_1 , es decir, *Action_Movies* y *Soap_Operas* (hermana de la superpropiedad de C_1, *Movies*, en la jerarquía establecida en la ontología).

En el ejemplo mostrado en Fig. 2, todas las propiedades definidas son relevantes en el proceso de razonamiento semántico (por pertenecer a alguno de los conjuntos $\mathcal{TP}_{C_0}$ o $\mathcal{TP}_{C_1}$ calculados mediante el algoritmo *Target_Properties*).

Una vez calculados los conjuntos de clases y propiedades en las que se basará el sistema para razonar, podemos empezar a describir las relaciones inferidas por el recomendador, y las razones por las que el sistema incluiría los contenidos de la Fig. 2 en la recomendación final.

En primer lugar, tenemos una relación entre la retransmisión de Formula 1 vista por el usuario, y un documental deportivo en el que participan tanto el piloto "Michael Schumacher" (presente en el perfil del usuario), como el futbolista "Beckham". En la Fig. 2 también se aprecia una asociación entre una retransmisión de fútbol (instancia de la clase *Soccer_Broadcasting*) y el estadio "Yokohama", donde éste tiene lugar. Además en esta última cadena semántica directa también participa una instancia de la clase *Soccer_Players*, por lo que el sistema descubre un nexo común entre el documental deportivo descubierto antes, y esta retransmisión de fútbol. Esta relación de tipo *cadena semántica directa con nodos comunes*, permite que el sistema infiera una asociación entre ambos programas, aún cuando no es el mismo futbolista en el que interviene en ambos. Este tipo de inferencias sólo tiene cabida en un entorno de razonamiento sobre la semántica de los contenidos de TV.

Por tanto, mediante el análisis de las dos clases incluidas en el conjunto $\mathcal{TC}_{C_0}$, el sistema infiere, entre otras, las relaciones semánticas mostradas a continuación:

Formula1_Broadcasting $\Rightarrow$ *Sport_Documentaries*
Sport_Documentaries $\Rightarrow_{Soccer\_Players}$ *Soccer_Broadcasting*
Formula1_Broadcasting $\Rightarrow_{\Rightarrow_{Soccer\_Players}}$ *Soccer_Broadcasting*
Soccer_Broadcasting $\Rightarrow$ *Sport_Places*

De los dos nuevos contenidos de TV descubiertos por el sistema – soccer broadcasting y sport documentaries –, es especialmente relevante el segundo de ellos. Esto se debe a que este contenido está relacionado con uno de los valores presentes en el perfil de usuario, concretamente con el piloto de fórmula 1 que interesa al mismo. Sin embargo, el sistema también debe considerar en el proceso de recomendación, la importancia del partido de fútbol descubierto. Esto se debe a que en la ontología de conocimiento, utilizada por el recomendador, este tipo de programas y las retransmisiones de Formula 1, vistas por el usuario, son instancias de clases “hermanas”, y por tanto, existe un nexo semántico entre ellas.

Sigamos describiendo otras asociaciones definidas en la Fig. 2. Es fácil ver que hay una cadena semántica directa entre la película de acción vista por el usuario, y una película de terror, establecido a través del actor “Clint Eastwood”. Además esta película está relacionada con la instancia “Amsterdam Arena” de la clase *Sport Places*, ya que la trama de la película transcurre en este estadio deportivo. Esto permite descubrir un nexo de unión entre la misma y el partido de fútbol inferido anteriormente: ambas instancias están asociadas a lugares donde tienen lugar eventos deportivos (clase “Sport Places”), y por tanto hay una relación del tipo *cadenas semánticas directas con nodos comunes* entre ambas. La película de terror se incluirá en la recomendación final, por los dos motivos que acabamos de presentar. Por una parte, interviene en ella el actor definido en el perfil de usuario, y por otro lado, está relacionado semánticamente con el partido de fútbol, cuya relevancia en el proceso de personalización, ya ha sido explicada antes.

Las relaciones que hemos descrito están resumidas a continuación:

$Action\_Movies \Rightarrow Terror\_Movies$
$Terror\_Movies \Rightarrow Sport\_Places$
$Terror\_Movies \Rightarrow_{Sport\_Places} Soccer\_Broadcasting$

Exploremos ahora las restantes clases del conjunto $\mathcal{TC}_{C_1}$. Empecemos con la instancia de *Soap Operas*. Es fácil ver en la Fig. 2, la *cadena semántica directa* establecido entre la telenovela y el concurso definidos en la ontología de TV. Además dicho *cadena semántica directa*, permite descubrir sendas *cadenas indirectas*.

Partamos de los siguientes *cadenas semánticas directas*, establecidos entre las instancias de las clases indicadas a continuación:

Soap_Operas [hasActress] *Actresses* [presenterOf] *Quiz_Shows*
Action_Movies [hasStarringActors] *Actors* [producerOf] *Terror_Movies*

Con arreglo a la jerarquía de clases y propiedades establecida en la ontología de TV, la propiedad *hasActress* está relacionada mediante *nexo semántico* con *hasStarringActors*, debido a que la primera es propiedad “hermana” de la superpropiedad de la segunda, tal como se muestra en la Fig. 3.

Si recordamos la definición de *relación semántica indirecta* dada en la Secc. 1.4, es fácil ver que si también existiese un *nexo semántico* entre las propiedades *presenterOf* y *producerOf*, el sistema habría inferido una *relación indirecta* entre las instancias de las clases *Soap Operas* y *Action Movies*. Dicho nexo existe, dado que ambas son propiedades “hermanas” en nuestra ontología. La segunda relación indirecta mencionada antes, se establecería entre las instancias de las clases *Soap Operas* y *Comedy Movies*, y se explicaría de forma análoga. De este modo, se infieren las siguientes relaciones.

$Soap\_Operas \approx Action\_Movies$
$Soap\_Operas \approx Comedy\_Movies$

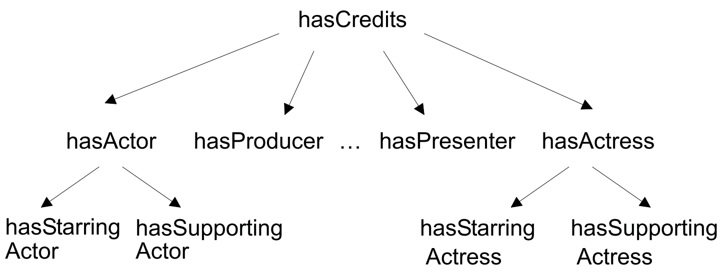

Figura 3. Subconjunto de propiedades de la jerarquía de propiedades de la ontología de TV

De acuerdo a las relaciones mostradas arriba, la película de acción vista por el usuario estaría relacionada semánticamente con una telenovela. En principio, la relación indirecta entre la película de acción y la telenovela, no involucra ningún valor concreto definido en el perfil personal del espectador (en este caso, el nombre del actor favorito del usuario), por lo que la telenovela no sería especialmente relevante de cara a la recomendación final. Sin embargo, si seguimos razonando sobre la ontología de la Fig. 2, descubriremos que es posible descubrir una asociación más directa con valores concocidos por el sistema y definidos en el perfil del usuario.

Para ello, centrémonos en la instancia de la clase *Comedy Movies*. Este programa se puede relacionar semánticamente con la película de acción mencionada antes, ya que también en ella participa el actor favorito del usuario.

Además, se puede inferir una nueva cadena semántica directa entre la instancia de *Comedy Movies* y un concurso televisivo, presentado por la actriz protagonista de la comedia. A la vista de la Fig. 2, es fácil ver que también se puede establecer una asociación semántica entre la telenovela y esta comedia, ya que ambas tienen un nexo en común: la actriz “Julia Roberts”. Las asociaciones semánticas que hemos definido se resumen a continuación.

$Comedy\_Movies \Rightarrow_{Actors} Action\_Movies$
$Comedy\_Movies \Rightarrow Quiz\_Shows$
$Soap\_Operas \Rightarrow Quiz\_Shows$
$Comedy\_Movies \Rightarrow_{Actresses} Soap\_Operas$

A la vista de los contenidos sugeridos por **AVATAR** en este ejemplo, podemos concluir diciendo que este tipo de mecanismos de recomendación, basados en el establecimiento de asociaciones semánticas, son especialmente útiles para descubrir al usuario contenidos que no conocía. Estos programas, sin embargo, pueden ser de su interés, debido a que guardan relación, desde un punto de vista semántico, con las preferencias declaradas en su perfil personal.

4. Algunas conclusiones

Los algoritmos desarrollados permiten componer un recomendador “inteligente” capaz de interpretar el conocimiento de la ontología OWL implementada, infiriendo información que permitirá elaborar recomendaciones mejoradas, más allá de la simple búsqueda sintáctica, en la que se han basado enfoques anteriores. Un ejemplo de ello se ha descrito en el proceso de recomendación mostrado en la Secc. 3.

Por otra parte, cabe destacar dos características de las sugerencias ofrecidas por AVATAR. En primer lugar, éstas son **diversificadas**. Esto se debe a que los programas recomendados son descubiertos infiriendo relaciones indirectas u ocultas en la ontología de TV, esto es, no declaradas de forma explícita en la base de conocimiento. Este tipo de relaciones ayudan al sistema a ofrecer a los espectadores programas *variados*, relacionados mediante asociaciones implícitas pero significativas, que no serían detectadas mediante los enfoques recomendativos existentes actualmente. Por otro lado, estas relaciones permiten mejorar las recomendaciones gracias a la diversificación que hemos comentado, pero sin perder una característica que va unida a los recomendadores, esto es, la **personalización**. Así, este carácter personalizado está garantizado debido a que las relaciones inferidas se establecen entre los programas que interesan al usuario en cuestión —declarados como instancias concretas en su *perfil-ontología*— y otros contenidos almacenados en la base de conocimiento OWL de nuestro sistema.

Otro de los aspectos relevantes de nuestro enfoque es que éste es fácilmente generalizable a cualquier otra aplicación de la Web Semántica, en la que se busque personalizar una recomendación. De esta forma, bastará con conceptualizar el nuevo dominio de aplicación del sistema —mediante la ontología oportuna– para después descubrir y clasificar las relaciones que hemos definido.