Szemantikus Web: egy rövid bevezetés

Herman Iván
World Wide Web Consortium

email: ivan@w3.org

1. Bevezetés

A Weben lévő információk típusa igen nagy változatosságot mutat. A klasszikus szöveges információk mellett találhatunk fényképeket, két és három dimenziós grafikát, videót, zenét, stb. A keresett információk (például egy utazáshoz szükséges tények összessége) sokszor különböző weblapokon találhatók csak meg (egy részük a légitársaság lapján, mások egy adott szálloda saját lapján, stb.). Ez utóbbi esetben a problémát nehezítheti ha a különböző lapok különböző terminológiákat használnak ugyanarra a fogalomra. És persze ne felejtsük el a természetes nyelvek (magyar, angol, kínai) gazdagságát sem. Mindezek ellenére a Web csodálatos módon működik. Ennek egyik magyarázata, hogy mi, emberek, rendkívül könnyen tudunk inkoherens vagy nem teljes információkat kombinálni. Általában megértjük egy fénykép tartalmát (hacsak nincsenek látási problémáink), könnyedén áthidalunk terminológiai különbségeket, félkész információkat tudunk teljessé tenni és, sokszor, különböző nyelveken is beszélünk.

Ez azonban a számítógépekre nem igaz. Mint ahogy ezt mindannyian tudjuk, a számítógépek alapvetően buták. Ahhoz, hogy intelligens programok (ágensek) alkalmazhatóak legyenek, a rendelkezésekre álló adatokról, az adatok közötti összefüggésekről, azok esetleges ekvivalenciájáról, nyelvéről, stb. sokkal precízebb információkat kell biztosítani. Csak ezek alapján lehet elvárni, hogy az ágensek képesek legyenek kihasználni azt a fantasztikus mennyiségű adatot, amely ma a világhálón rendelkezése áll. Márpedig ágensek már most rengetegen vannak, és az ezekre épülő alkalmazások száma, és főleg az irántuk támasztott felhasználói igények állandóan növekednek.

Íme néhány példa lehetséges alkalmazásokra, melyek ma még vagy csak papíron léteznek vagy, habár léteznek, jelentős problémákkal küszködnek. M. Dertouzos, az MIT Számítástudományi Laboratóriumának valamikori vezetője már 1995-ben (vagyis, ami a Webet illeti, az ősidőkben…) több izgalmas alkalmazást ír le az „Unfinished Revolution" című könyvében[1] (a könyv egyébként magyarul is megjelent „Félkész forradalom" címmel[2]), mint például egy automatizált repülőjegy foglaló rendszert, amely figyelembe veszi a különböző légitársaságok információit, az utas kívánságait, stb. Hasonló alkalmazásokat ír le T. Berners-Lee szerzőtársaival egy sokszor idézett cikkben[3], illetve könyvében[4]. Szeredi Péter és kollégái, egy nemrégiben megjelent könyvükben[5], hosszan elemzik a keresőrendszerek problémáit; hasonló problémák adódnak a webszolgáltatások tömeges elterjedésének esetén (pld. "találd meg a legelegánsabb differenciálegyenlet megoldó webszolgáltatást, és mondd meg, hogy hogy kombinálhatom ezt a grafikus kijelző szolgáltatással"). Rosszul, vagy egyáltalán nem látó felhasználóknak egy kép részletesebb, szövegalapú leírása elérhetővé tehet egy, a Weben lévő grafikát[6]. Megint más jellegű probléma vetődik fel mikor nagy cégek összeolvadnak (például mikor a HP felvásárolta a Compaq céget): milyen módon kombinálható a két cég adatbázisa amelyek, végül is, hasonló (pénzügyi, technikai, jogi, személyzeti, stb.) információkat tartalmaznak de különböző konvenciókkal, terminológiákkal, szokásokkal. És az intelligens ágensek alkalmazásának sorát még sokáig lehetne folytatni.

Mindezen példák egyik közös jellemzője, hogy egy alkalmazáson belül kombinálni kell, méghozzá koherens módon, weben lévő adatokat. Mászóval fogalmazva, a weben lévő adatok között kapcsolatokat kell létrehozni, definiálni. Itt az „adat" lehet egyszerűen a weben lévő fájl (például egy on-line címlista), elérhető adat (például egy adatbázisban lévő rekord elérése valamilyen alkalmazási felületen keresztül), vagy *metaadat*, vagyis adat az adatról (például egy könyvtári katalóguscédula elektronikus változata). A szemantikus web ezt a problematikát próbálja megoldani. Ha röviden akarjuk jellemezni, akkor azt mondhatjuk, hogy *a szemantikus web célja egy olyan infrastruktúra létrehozása, amely lehetővé teszi a Weben lévő adatok integrálását, a közöttük levő kapcsolatok definiálását és jellemzését, illetve az adatok értelmezését*.[1]

Ahhoz, hogy ezt az infrastruktúrát létrehozzuk, néhány fontos építőelemre van szükség. Ezek a következők (zárójelben azon W3C–specifikációk, amelyek már léteznek, vagy fejlesztés alatt vannak, és amelyek az adott építőelemhez alapvető szerepet játszanak):

1. Az adatokat egyértelműen meg kell „címezni" a hálón, vagyis el kell őket nevezni (URI=URL+URN)

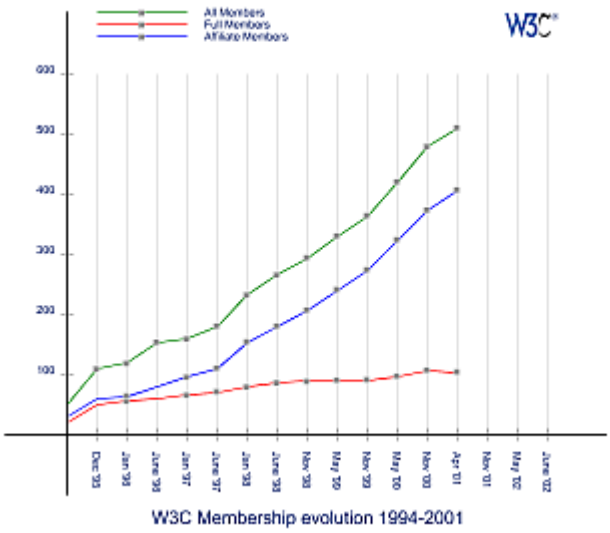
1 A „Szemantikus Web" elnevezés, sajnos, félreértésekre is vezetett, ezért fontos megjegyezni, hogy ez nem egy *új* Web, amely a régit „elavulttá" tenné, hanem inkább a jelenlegi Web kiterjesztése.

2. Szükség van egy precíz adatmodellre, amely formális keretet ad az adatok egymáshoz való kapcsolatának definiálására, és a kapcsolatok leírására (RDF[7])
3. Az adatok közötti kapcsolatokat, illetve a kapcsolt adatok referenciáit, el kell tudni érni, le kell tudni kérdezni (SPARQL[8])
4. Az kapcsolatok leírására szolgáló terminológiát definiálni kell tudni (RDFS[9], OWL[10], SKOS[11])
5. A kapcsolatokon, illetve azok leírásán, logikai következtetéseket kell tudni levonni (OWL[10], RIF[12])

Az URI használata természetesen nem specifikus a szemantikus webre; a szemantikus web felhasznál egy elterjedt és jól definiált technológiát. A többi (RDF, OWL, stb.) a szemantikus webre lett kifejlesztve, illetve áll fejlesztés alatt: vannak közöttük már szabványosított specifikációk (RDF, OWL), vannak, amelyek már nagyon közel állnak egy végleges formátumhoz (SKOS, SPARQL), és megint mások még csak a fejlesztés legelején vannak (RIF). A továbbiakban mindezekről a technológiákról adok egy nagyon rövid áttekintést.

2. RDF (Resource Description Framework, Erőforrás Leíró Keretrendszer)

Az RDF a szemantikus web alapköve és legrégebbi specifikációja. Legegyszerűbb egy példán keresztül bemutatni.



1. Ábra: egy egyszerű üzleti grafika

Az első ábra egy jellegzetes üzleti grafikát mutat, amelyhez hasonlót igen sokat lehet a Weben találni. A tartalma világos, látszólag nem igényel további magyarázatot. A probléma akkor merül fel, ha ezt az ábrát egy vak Web felhasználó számára akarjuk elérhetővé tenni.

Mint ahogy erre már utaltunk, a megoldás az adat szöveges leírása lehetne. De ahelyett, hogy ezt a specifikus ábrát írnánk le részletesen, ki szeretnénk fejleszteni egy olyan intelligens programot amely képes egy metaadatot automatikusan értelmezni, és abból egy olvasható (vagyis, egy vak felhasználó számára felolvasható) szöveget létrehozni. Ez azt jelentené, hogy az ábra létrehozójának csak néhány, jól definiált metaadatot kell az ábrához hozzárendelni, a többi a program dolga. Mivel a weben vagyunk, feltételezhetjük, hogy az ábra SVG-ben van, vagyis egy grafikus XML leírásban; ennek annyi itt a jelentősége, hogy a grafika minden eleme megcímezhető egy URI segítségével. (Az alkalmazás egy részletesebb leírása megtalálható [6]-ban).

Milyen állításokat tartalmazhat egy metaadat ebben az esetben? Íme néhány példa:

- „A teljes ábra egy üzleti grafika”
- „A üzleti grafika típusa vonalgrafika”

- „Ezen a címen az ábra képaláírása található”
- „A képaláírás egyben egy link is a Weben”
- „A képaláírás erre és erre a URI-re utal”

Az összes példa egy hármas tagolást mutat. Valamiről állítunk valamit („a teljes ábra”, „egy adott cím”, „a képaláírás”) és ezt összekapcsoljuk egy egyszerű szövegelemmel („vonalgrafika”) vagy egy külső adattal („URI”). Maga a kapcsolat is jelentőséggel, „szemantikával” bír („típusa”, „utal”). Természetesen egy alkalmazásban bonyolultabb állítások is szükségesek lehetnek, de a gyakorlat azt mutatja, hogy ezek is lebonthatók ilyen hármas tagolású alapelemekre.

Az RDF az ilyen „hármasok” matematikai modellje. Egy RDF hármas alanyból, állítmányból (vagy tulajdonságból, mindkét fogalom használatos), és tárgyból áll[2]; ebben a megközelítésben például a második példamondat a következőképpen absztrahálható:

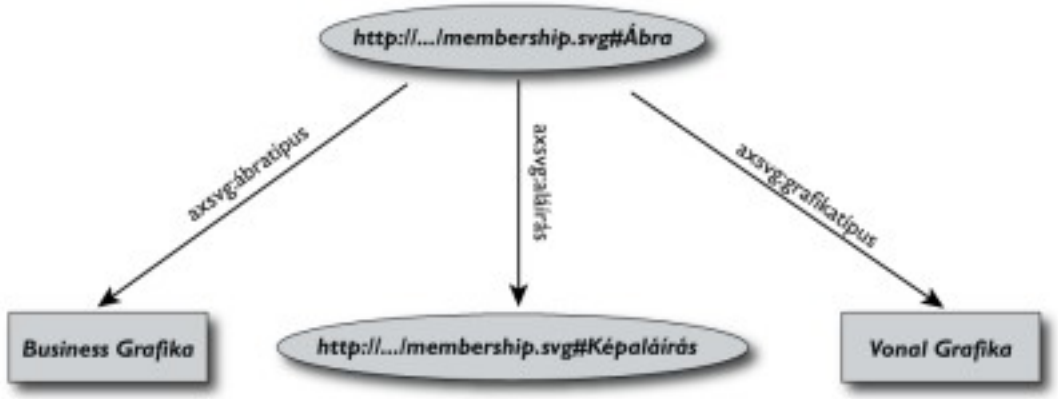
```
(A üzleti grafika címe,típus,"vonalgrafika")
```

A hármas mindhárom eleme egy URI (amely, általánosságban, valamilyen „erőforrásra” utal[3] mely lehet egy adatforrás a weben, egy email cím, stb.) vagy, a tárgy esetében, egy szövegelem. Vagyis, ha az ábrának és a „típus” tulajdonságnak egy-egy jól meghatározott URI-t rendelünk hozzá, akkor a hármas precízebben leírható a következő módon:

```
(http://.../Ábra.svg,http://.../típus,"vonalgrafika")
```

Az RDF (Resource Description Framework, vagyis Erőforrás Leíró Keretrendszer) az ilyen hármasok (vagy állítások) összességét írja le. (Az RDF specifikáció tartalmaz néhány további fogalmat amelyre most nem tudunk kitérni, de ezek nem igazán fontosak az alap fogalmak megértéséhez).

Az RDF állítások összességét tekinthetjük egy irányított, címkézett gráfnak is. A gráf csomópontjai az alanyok és tárgyak (vagyis a rájuk hivatkozó URI-k); a gráf élei pedig az állítmányok (tulajdonságok), ahol is az állítmány neve (vagyis az azt leíró URI) adja az él címkéjét. Az RDF állítások gráfban való ábrázolása nagyon áttekinthető; érdemes gráfokban „gondolkozni” az RDF állítások létrehozásakor és alkalmazásakor. A probléma persze az, hogy a gráfoknak kell, hogy legyen egy géppel olvasható formája is: az idők folyamán több, egymással ekvivalens szöveges reprezentációforma is született. A W3C által definiált RDF/XML formátum XML–alapú, de létezik más, nem XML–alapú szintaxis is. És ez rendben is van: nem a szintaxis a lényeg, hanem a mögötte álló adatmodell.



2. Ábra: egy egyszerű RDF gráf

A 2. ábra egy egyszerű RDF gráfot mutat, amely három állításból áll: lényegében az első háromból a fenti listán. (A tulajdonságokat egyszerűség kedvéért névterekkel adtuk meg). A három állítás XML változata a következőképpen néz ki:

2 Az angol terminológia: „subject”, „property” vagy „predicate”, és „object”. Itt és a továbbiakban az RDF ajánlások magyar fordításának a terminológiáját használjuk.

3 Az angol terminológia az erőforrásra a „resource”.

```
<rdf:Description rdf:about="http://...#Ábra">
  <axsvg:ábratípus xml:lang="hu">Business grafika</axsvg:ábratípus>
  <axsvg:aláírás rdf:resource="http://...#Képaláírás"/>
  <axsvg:grafikatípus xml:lang="hu">Vonal grafika</axsvg:grafikatípus>
</rdf:Description>
```

A szintaktikus részletek itt most nem érdekesek, a példa, remélhetőleg, magáért beszél.

A modellben kitüntetett szerep jut az URI-nak. Az URI-k használata biztosítja, hogy az RDF állítások *bármilyen* adatra vonatkozhatnak a weben. Így például egy adott grafikáról nemcsak a grafika létrehozója publikálhat metaadatot, hanem bárki. De talán még fontosabb, hogy az RDF hármasok nemcsak metaadatokra használhatók, hiszen például az alany lehet bármilyen adat amelynek URI címe van, vagyis amely a weben „van". RDF hármasokkal a weben lévő adatokat lehet egymáshoz rendelni, közöttük lévő kapcsolatokat definiálni és jellemezni, hűen a bevezetésben leírt célokhoz. Bizonyos értelemben azt is lehet mondani, hogy az URI-k használata „ágyazza" az RDF hármasokat a webbe.

Az URI-k használatának van egy további, rendkívül jelentős következménye. Igen egyszerűvé válik különböző forrásokból származó RDF hármasok összekombinálása egy alkalmazáson belül. Ha két, egymástól különböző hármas halmaz azonos csomópontokat vagy éleket tartalmaz (merthogy ugyanazokat a URI-kat használják) akkor ezen csomópontok, az RDF szempontjából, *ugyanazok*. Magyarán, a gráfokat triviális módon egybe lehet olvasztani. Vagyis egy alkalmazás minden további nélkül felhasználhat és összekombinálhat Weben lévő metaadatokat, RDF hármasokat, függetlenül attól, hogy azokat ki hozta létre[4]. Végeredményben ez annak az analógja, amikor az emberi felhasználó koncepcionálisan összekombinál weben lévő információkat. Nem véletlen, hogy gyakorlati alkalmazások manapság rutinszerűen használnak több tízezer, százezer, vagy akár millió hármasokból álló adatokat, amelyeket gyakran így kombinálnak össze.

Az RDF–specifikációnak már két generációja is létezik, a legutóbbi 2004-ben jelent meg[7]. Mivel az első verzió még a 90-es évekből származik, bőven volt idő RDF programrendszerek, programozási környezetek, stb, kifejlesztésére. Ma, lényegében, az a helyzet, hogy szinte minden programnyelvhez létezik legalább egy RDF programozási környezet (a berlini Frei Universität által fenntartott weblap[13] gazdag információforrás a rendelkezésre álló eszközökre). E programozási környezetek tartalmaznak egy vagy több szövegelemzőt az RDF adatok beolvasására, többé-kevésbé szofisztikált módon tárolják a hármasokat, amelyeket le lehet kérdezni, ki lehet törölni, új hármasokat lehet létrehozni, stb. A programozási környezetek döntő többsége nyílt forráskódú fejlesztés.

Mint már fentebb jeleztük, gyakorlati alkalmazások adott esetben milliós nagyságrendben kezelnek RDF hármasokat. Ez több problémát is felvet. Egyrészt szükség van adatbázisokra, amelyek képesek ilyen nagyságrendeket kezelni, hiszen ilyen hármas halmazokat nem lehet egyszerűen a memóriában tárolni. Másrészt szükség van valamiféle lekérdező nyelvre bonyolultabb információk lekérdezésére.

Az első probléma inkább gyakorlati, és ma már több RDF adatbázis rendszer létezik, mind a szabad szoftverek piacán, mind az üzleti termékként. Így például az Oracle legfrissebb verziója (10*g*) képes RDF adatokat tárolni, és egy RDF absztrakciós felületet is nyújt a felhasználónak. Mások egy teljesen új fejlesztéssel alakítottak ki adatbázis rendszereket kizárólagosan az RDF céljára.

A lekérdezőnyelv kérdése nyilvánvalóan más jellegű. Általánosságban azt lehet mondani, hogy egy adatbázis–lekérdezőnyelv ahhoz az adatmodellhez kapcsolódik, amelynek az adatait el kívánjuk érni. Így például az SQL a relációs adatmodellhez kapcsolódik, míg az XML Query az XML adatmodelljéhez. A SPARQL[8] ezeknek az analógja az RDF világban: ez az RDF adatmodelljéhez kapcsolódó lekérdezőnyelv[5].

A SPARQL alapelve voltaképpen igen egyszerű: mivel az RDF hármasok halmaza egy gráf, egy SPARQL lekérdezés egy *gráfmintát* tartalmaz. Egy gráfminta olyan (RDF) hármasok halmaza, amelyek változókat is tartalmazhatnak az alany, az állítmány, vagy a tárgy helyén. A lekérdezés során a rendszer a változókba behelyettesíti a hármas halmazban lévő URI-kat, vagy szövegelemeket, és ellenőrzi, hogy az így kapott gráf az eredeti RDF gráf részgráfja-e. Ha igen, a változó behelyettesítése sikeres, és a változók a lekérdezés egyik lehetséges eredményét adják. Így például definiálhatjuk a következő SPARQL lekérdezést:

```
SELECT ?u, ?v
WHERE { ?g axsvg:aláíras ?u.
        ?g axsvg:grafikatípus ?v.
      }
```

Ha ezt a lekérdezést a 2. ábrán lévő gráfra alkalmazzuk, akkor a

4 Aki megpróbált már különböző forrásokból származó XML hierarchiákat egy alkalmazásban összekombinálni, az tudja, hogy XML esetén ez milyen bonyolult tud lenni…

5 Ki kell hangsúlyozni, hogy míg az RDF egy lezárt, publikált specifikáció, a SPARQL még fejlesztés alatt áll és 2006 vége előtt nem valószínű, hogy be lenne fejezve. Vagyis a részletek még itt-ott változhatnak.

```
?u = http://.../membership.svg#Képaláírás
?v = "Vonal grafika"
```

hozzárendelés egy lehetséges (ebben az esetben az egyedül lehetséges) válasz a lekérdezésre. Természetesen ez egy leegyszerűsített példa; a gyakorlatban mind a hármas halmazok, mind a lekérdezések sokkal bonyolultabbak. Érdemes megjegyezni, hogy bár a SPARQL fejlesztése még nem záródott le, a [13]-ban felsorolt programozási környezetek egy jelentős része már implementálta a SPARQL valamely változatát (vagyis például a memóriában tárolt hármas halmazokat is lehet SPARQL lekérdezésekkel kezelni), és az RDF adatbázisok is kezdik beépíteni a rendszerükbe. Ez jól mutatja a SPARQL jelentőségét.

3. Ontológiák

Az RDF, illetve újabban az RDF+SPARQL kombináció, már érdekes és fontos alkalmazási lehetőségeket rejt magában. A teljes szemantikus web azonban, ahogy ezt a bevezetésben leírtuk, ennél többet igényel. Két fő terület van, amely további technológiákat kíván, nevezetesen:

1. Milyen terminológiát használhatunk egy alkalmazáson belül? A fenti példában használtuk az `axsvg:aláírás` tulajdonságot; felmerül a kérdés, hogy jogos-e a használata, ha igen, milyen körülmények között (pld. mely erőforrásokra vonatkoztatva)? Egyáltalán, hogyan lehet osztályozni a rendelkezésre álló erőforrásokat?
2. Hogyan jellemezhetjük az erőforrások közötti logikai kapcsolatokat? Mikor mondhatjuk, hogy két erőforrás (például két különböző nevű tulajdonság) ugyanazon fogalomra utal? Lehet-e a tulajdonságok logikai viselkedését jellemezni (például hogy szimmetrikus vagy tranzitív, hogy valójában egy függvény, stb.)?

Világos, hogy ezek a kérdések felmerülnek minden komolyabb szemantikus web alkalmazás esetén, de az is világos, hogy ezen kérdések megválaszolására egy bonyolultabb rendszer kell, amely ki kell hogy egészítse az RDF egyszerű adatmodelljét. Az is viszonylag hamar kiderül, hogy az összes felmerülő kérdés megválaszolása rendkívül bonyolulttá válhat, ezért érdemes technológiák egy családját kifejleszteni, amelyek közül a felhasználó igénye szerint választhat. Így fejlődött ki, a W3C keretében, az RDFS, OWL, és SKOS, és indult el 2005 legvégén az RIF fejlesztése. Ezek közül az RDFS és az OWL technológiák a legkiforrottabbak, és a továbbiakban elsősorban ezekre fogunk koncentrálni.

Az alapvető megközelítés fogalomjegyzékek, ontológiák használata, illetve a tradicionális ontológiai módszerek adaptálása RDF környezetekre. Tekintsük a következő egyszerű példát egy fogalomjegyzékre:

- létezik az „emlős” fogalma
- minden „delfin” egyben „emlős” is
- „Flipper” egy delfin

Ebben a fogalomrendszerben, az „emlős” egy osztály, amelyhez különböző egyedek tartozhatnak; az „emlős” osztályhoz való tartozás „jellemzi” az egyedeket. A második állítás egy logikai kapcsolatot teremt két osztály között, hiszen azt jelenti, hogy minden egyed, amely az egyik osztályhoz tartozik, automatikusan a másikhoz is tartozik. Végül a harmadik állítás egy egyed hovatartozását definiálja.

Az RDFS („RDF Vocabulary Description Language”, magyarul „RDF Szókészlet Leírónyelv”)[9] egyik célja az ilyen fogalomrendszerek absztrahálása. Az RDFS bevezeti az erőforrások *osztályának* fogalmát mint „egyedek lehetséges összességét”, továbbá a *típus* fogalmát, vagyis egy egyednek egy osztályhoz való tartozásának jelzését. Egy tetszőleges erőforrás típusa tehát egyszerűen azt jelzi, hogy az erőforrás egy adott osztályhoz tartozik. Végül bevezeti az alosztály relációt két osztály között, az értelemszerű definícióval.

Ami első pillanatra talán meglepőnek tűnhet az az, hogy az RDFS fogalmak és definíciók *RDF-ben vannak megfogalmazva*. Vagyis az általános osztály nem más mint egy speciális erőforrás, amelynek egy meghatározott URI-je van; a típus és az alosztály pedig szintén egy-egy (RDF) tulajdonság jól meghatározott, és szabványosított szemantikával. Emlékezzünk arra tényre, hogy az RDF hármasok *bármilyen* URI-re, vagyis bármilyen erőforrásra vonatkozhatnak, másszóval az RDFS definíciókhoz nem kell az RDF adatmodelljéből kilépnünk. Mindezek előrebocsájtásával a fenti három állítást a következőképpen fordíthatjuk le RDF-re:

```
(http://....#emlős, rdf:type, rdfs:Class)
(http://....#delfin, rdf:type, rdfs:Class)
(http://....#delfin, rdfs:subClassOf, http://....#emlős)
(http://....#Flipper, rdf:type, http://....#delfin)
```

(Itt is, mint az előző példákban, névtereket is használtunk a URI-k rövidítésére; az egyedüli különbség az, hogy az `rdf` és `rdfs` névtereknek megfelelő URI-k ebben az esetben a W3C szemantikára vonatkoznak). Az első két hármas azt jelenti, hogy az „emlős" és „delfin" egyedek önmagukban osztályok (vagyis az absztrakt RDFS osztály egyedei), a harmadik hármas az alosztály kapcsolatot hozza létre a két osztály között, míg a negyedik hármas a Flipper, mint egyed, hovatartozását definiálja.

A fenti hármasok egy nagyon egyszerű következtetési sémát is lehetővé tesznek. Valóban, ha egy RDF rendszer a fenti négy hármast kapja bemenő adatként, és a rendszer „érti" az RDFS szemantikát is, akkor a rendszer *kikövetkeztetheti* a következő hármast is:

```
(http://....#Flipper, rdf:type, http://....#emlős)
```

Vagyis a rendszer képes logikai következtetéseket levonni a rendelkezésére álló adatokból. Habár természetesen ez a példa a trivialitás határát súrolja, más esetekben a következtetések sokkal bonyolultabbak de hasonló alapokon nyugszanak.

Az osztálydefiníciók a tulajdonságok jellemzését is lehetővé teszik. Az osztályok ismeretében lehetővé válik egy tulajdonság, mint bináris reláció, értéktartományának és érvényességi körének (másszóval értékkészletének) definiálása: pontosan definiálni lehet, hogy az állítmány alanya vagy tárgya mely osztályokhoz tartozhat. A fenti példára visszatérve: az RDFS dokumentum, természetesen, definiálja a saját maga által definiált tulajdonságok értéktartományát és értékkészletét, és nyilvánvaló módon, az `rdfs:subClassOf` állítmány alanya egy osztály egyed kell hogy legyen. Ezért voltaképpen, a második hármas felesleges, hiszen a harmadikból következik. (Természetesen használata attól még nem hiba, csak redundáns!) Végül: az RDFS lehetőséget ad egy al–tulajdonság definiálására is: ha `P` a `Q` al–tulajdonsága, és fennáll `(x,P,y)` akkor fennáll `(x,Q,y)` is.

Miként az RDF-re, az RDFS-re is igaz, hogy tartalmaz még további konstrukciókat, amelyre itt nem tudunk kitérni. De remélhetőleg az eddigiek már adtak egy képet arról, hogy az RDF+RDFS kombináció mire képes. Meg kell jegyezni, hogy az RDF+RDFS formális szemantikája (például annak precíz leírása, hogy egy RDFS-t értő rendszer milyen következtetéseket vonhat le a rendelkezésre álló bemeneti adatokból) messzemenően nem triviális, és megértése alapos modellelméleti tudást igényel. De a felhasználók számára ez nem feltétlenül fontos, ugyanúgy ahogy a felhasználók döntő többsége nincs tisztában, mondjuk, az általuk használt programozási nyelv formális szemantikájával.

Az RDFS az ontológiák definiálásának egy szintjét jelenti; azt mondhatnánk, hogy az alapszintet. Az RDFS definíciói annyira alapvetőek, hogy a gyakorlatban az RDFS nehezen választható el az alap RDF modelljétől. Így például szinte nincs olyan komolyabb RDF alkalmazás, amely ne használná az osztály és a típus fogalmát, még akkor is, ha a programozási rendszer nem „érti" az RDFS precíz szemantikáját. Az RDFS azonban alapszint, és a komplikáltabb definíciók esetén szükség van a következő szintre; itt lép be az OWL specifikáció („Web Ontology Language", magyarul „Web Ontológia Nyelv")[10]. Lássuk mit ad az OWL mint „pluszt" az RDFS-hez képest!

Az első érdekes terület az osztályok fogalma. Mint fentebb láttuk, RDFS-ben osztályokat lehet definiálni (lényegében elnevezni), és az osztályok között alosztálykapcsolatokat lehet létrehozni. Mást azonban nemigen. Az OWL az RDFS-re épül, vagyis mindazt, amit RDFS-ben lehet, azt OWL-ben is lehet, de a OWL-ben osztályokat lehet *konstruálni* is. Osztályokat lehet más osztályok uniójaként, metszeteként, komplemenseként definiálni; lényegében rendelkezésre állnak az alap halmazelméleti műveletek. Lehetséges az osztályok egyedeinek explicit felsorolása; és osztályokat lehet a tulajdonságok segítségével is definiálni.

Érdemes kitérni ez utóbbi alternatívára, mert ez a legbonyolultabb, és talán informatikusok számára, a legkevésbé természetes művelet. Eredete a tradicionális ontológiákra nyúlik vissza. Ha még visszaemlékszünk a delfin ontológiánkra, abban az alábbi mondattal is definiálhatnánk egy delfint:

– a delfin olyan emlős amely a vízben él

(most egy pillanatra tekintsünk attól a ténytől, hogy a delfin nem az egyetlen ilyen emlős). Ez a mondat a delfinek osztályát definiálja a következő módon:

– vegyük az emlősök osztályát
– vegyük az „él" tulajdonságot
– tekintsük az emlősök osztályának azon egyedeit, amelyre az „él" tulajdonságot alkalmazva a hármas tárgya a „víz" (hasonló módon definiálhatnánk a denevéreket, mint olyan emlősöket amelyek a levegőben élnek). Ez definiálja (OWL-ban) a delfinek osztályát.

Az OWL ezt az osztálykonstruálási formát értéktartomány-korlátozásnak nevezi.

Mivel az OWL az RDFS kiterjesztése, az OWL fogalmak ugyanúgy RDF hármasokban írhatók le, beleértve a tulaj-

donság korlátozásokat, de a szükséges hármasok száma és a struktúra komplexitása kicsit már bonyolultabb. Az érdeklődő olvasó utánanézhet az OWL Overview nevű dokumentumban[14], ill. annak magyar fordításában[15].

Az OWL a tulajdonságok terén is tovább bővíti az RDFS által adott lehetőségeket. Lehetővé teszi ugyanis a tulajdonságok logikai jellemzését, ahogy erre a fejezet elején utaltunk. Tulajdonságokat lehet szimmetrikusnak, tranzitívnek, függvénynek, stb, deklarálni. Egy OWL-t „értő" környezet egy sor logikai nem triviális következtetést képes levonni ezen definíciók alapján. Hogy egy példával szolgáljunk: az előző fejezetben leírt alkalmazás esetén szeretnénk, ha a következő következtetés lehetséges lenne: „ha 'A' balra van 'B'-től, és 'B' balra van 'C'-től, akkor 'A' balra van 'C'-től". Egy OWL környezetben lehetséges a 'balra van' tulajdonság tranzitivitásának a definiálása, ami a fenti következtetést lehetővé teszi.

Az OWL egy másik területet is érint, amelyekről az RDFS nem beszél. Ilyen például az osztályok, tulajdonságok, vagy egyedek *ekvivalenciájának* a problémája. Az osztályok és osztálydefiníciók különböző forrásokból származhatnak; egy egy alkalmazás adott esetben különböző ontológiákat összekombinálhat (ne felejtsük el, hogy az RDFS és az OWL is RDF-ben van megfogalmazva, vagyis az előző fejezetben leírt gráfkombinációs lehetőség ontológiákra is érvényes!). Szükségessé válhat tehát annak a kérdésnek a megválaszolása, hogy két osztály azonos-e, vagyis azonosak-e az egyedei avagy sem. Vagy hogy éppen szigorúan különböznek egymástól. Ugyanez a kérdés felmerül a tulajdonságok kapcsán is. Az OWL tartalmaz erre egy sor konstrukciót. Így például lehetséges a következő hármas, mint OWL állítás:

```
(http://....#delfin, owl:equivalentClass, http://....#dolphin)
```

vagyis az angol „dolphin" nevű osztály ugyanaz mint a magyar nevű „delfin". Emlékezzünk vissza a bevezetésben felvetett problémára: „Ez utóbbi esetben a problémát még nehezítheti ha a különböző lapok különböző terminológiákat használnak ugyanarra a fogalomra.". Ezek az OWL konstrukciók ezen probléma megoldására lettek kialakítva.

Tisztában kell lenni azzal, hogy egy OWL alapú következtetési rendszer *lényegesen* bonyolultabb mint az RDFS alapú. Az OWL eredete a tudásreprezentáció több évtizede kutatott és fejlesztett területére nyúlik vissza. Az OWL az első olyan nyelv, amely a klasszikus tudásreprezentációs logikákat (un. „leíró logikákat") a web dimenzióira alkalmazta; rövid idő alatt OWL a legelterjedtebb tudásreprezentációs nyelvvé vált. De ezen új dimenziók új gyakorlati problémákat vetettek fel; az egyre hatékonyabb következtetési algoritmusok kifejlesztése még mindig aktív kutatási irány, de már megjelentek az első programozási rendszerek, amelyek képesek az OWL (precízebben fogalmazva: az OWL részNyelvének) interpretálására, és a megfelelő következtetések kezelésére.

4. Új problémák, fejlesztések

Az RDFS és az OWL kifejlesztésével a munka korántsem állt le. Bármennyire is gazdag, az OWL rendszer nem teljes; egy sor tulajdonságot, logikai szabályt nem lehet OWL-ben kifejezni. Így például nem képes valószínűségeket kifejezni, pedig sokszor fordul elő, hogy egy alkalmazás egy hármasokkal leírt kapcsolatot csak úgy tud értelmezni, hogy „ez a kapcsolat egy adott valószínűséggel áll fenn". Az RDFS és az OWL által leírt kapcsolatok viszont szigorúan igen/nem jellegűek. Tipikus ilyen alkalmazási terület erre a problémára a biológia, biotechnológia. A valószínűségek (vagy, analóg módon, a fuzzy kapcsolatok) kifejezése nagyon sokakat érdekel: a legutóbbi nemzetközi szemantikus web konferencián (2005-ben) egy külön munkaértekezlet szerveződött ennek a problémának a vizsgálatára[16], és ez természetesen csak az első lépés.

Bizonyos következtetési láncok definiálása és kifejezése szintén túlmutat az OWL lehetőségein. A klasszikus eset a nagybácsi példája: ha a hármas halmazunk tartalmazza a következő két hármast

```
(Dávid, apja, Iván)
(Iván, testvér, János)
```

akkor szeretnénk, ha a rendszer a következő hármasra is következtetne:

```
(János, nagybácsi, Dávid)
```

Sajnos egy ilyen szabályt nem lehet az OWL-ban leírni, ez az un. Horn következtetési szabályokat igényli amely másfajta logikákhoz kapcsolódik. Komoly érdeklődés van az ilyen következtetési szabályok szabványosítására, illetve a Szemantikus Web általános struktúrájába való beillesztésére, de mindez nem jelentéktelen elméleti és gyakorlati problémákat vet fel. Egy tipikus alkalmazás lehet egy email kliensben használatos szűrődefiníciók kicserélése szabványosított formában, vagy biológiai/biokémiai adatok értelmezéseinek leírása és publikálása. A W3C 2005 áprilisában szervezett egy nagyon sikeres munkaértekezletet[17], és ez vezetett a W3C egyik legfrissebb munkacsoportjához „Rule Interchange Format" néven. Ez a csoport csak néhány hónapja kezdte el az érdemi munkát[12].

Egy másik érdekes terület a biztonság, bizalom kérdése. Milyen módon bízhatunk meg egy hármas halmazban: például igaz-e, hogy a halmaz létrehozója valóban az, akinek mondja magát? Lehetséges-e a hármas halmazokat (vagy azok egy részét) digitálisan aláírni? Ha igen, hogy, milyen protokoll mellett? Lehetséges-e egy hármas halmaz titkosítása?

Az ontológiák fejlesztése is jelentős feladat. Egy-egy nagyobb ontológia (például egy gén–ontológia) kialakítása közösségi munkát igényel, ami felvet egy sor, a kooperatív programfejlesztés problémáira emlékeztető gyakorlati problémát. Az OWL léte ebben voltaképpen csak katalízis szerepet játszik, hiszen az igazi probléma egy adott terület fogalmainak ismerete, kategorizálása, szisztematikus leírása. Ehhez az OWL csak szintaxist és, nem jelentéktelen mértékben, motivációt nyújt.

Mindez csak néhány példa a szemantikus web által felvetett kutatási-fejlesztési problémák közül. Világszerte sok kutató és fejlesztési csoport dolgozik ezeken a problémákon, egyetemi és ipari környezetben egyaránt; érdemes tehát ezt a területet figyelemmel kísérni!

5. Eszközök, alkalmazások

Az előbbiekben már hivatkoztunk a nagyszámú programozási környezetre. E környezetek közül néhány (például a Java-ra írt Jena rendszer) nemcsak RDFS, hanem (bizonyos korlátozás mellett) OWL alapú következtetést is megenged. OWL következtetési rendszereket egyébként külön is el lehet érni és más környezettel kombinálni.

Mint mondtuk, a rendelkezésre álló fejlesztési rendszerek, eszközök, stb, egy része nyílt forráskódú, habár adott esetben nagy, jól ismert cégek laboratóriumaiban fejlesztették őket (Nokia, HP, IBM, Adobe, stb.). Megint más eszközök piaci termékek (például az Altova XML Spy-hoz kapcsolódó editora, az Oracle vagy a Northrop Grumman adatbázisa vagy, hogy magyar példát is mondjunk, a progos.hu fejlesztési környezete). Összességében elmondhatjuk, hogy a szemantikus webes eszközök piaca az utóbbi években igen jelentősen fejlődött.

Ami az alkalmazásokat illeti, a szemantikus web követi azt a jól ismert utat, amelyet más alkalmazási területek is bejártak a weben. Az egyetemi, kutatóintézeti fejlesztések száma ma már nagyon jelentős és nagyon tág alkalmazási területeket lefed, elég csak megnézni az utóbbi 4-5 év európai uniós kutatás–fejlesztési projektjeit. De, például, RDF-t használ a Mozilla Firefox böngészője is, amely tartalmaz egy teljes, Javascript-ben írt RDF környezetet, és az RSS formátum is, voltaképpen, RDF. Kis méretű cégek, spin-off vállalatok, melyek erre a technológiára alapozzák egzisztenciájukat, szinte naponta születnek. És, ma már, a nagy világcégek is érdeklődnek a szemantikus web iránt. Nemcsak az eszközfejlesztők (mint a fennt említett HP, Nokia, IBM, vagy Oracle) hanem az alkalmazók is (Siemens, Nokia, Vodafone, Sun).

Hozzá kell azonban tenni, hogy a nagy cégek alkalmazásai jelenleg elsősorban az „intranet”-en belül maradnak. Így például a Nokia, a Vodafone, vagy a Sun portálokat fejlesztett ki egy-egy specifikus célra (a Sun esetében például a rendszerkarbantartók számára fenntartott információs portált, a Vodafone esetében a telefonok csengőhangjait áruló portált). Anélkül, hogy a felhasználó ezt tudná, ezek a portálok „belül” szemantikus web eszközöket használnak. Ez a fejlődési fázis nem csak a szemantikus web jellemzője; a 90-es években a bonyolultabb web felhasználások szintén az intranet-en működtek először és csak később jelentek meg „igazi” web alkalmazások és, voltaképpen, még mindig ez a helyzet a webszolgáltatások területén is.

A sok lehetséges példa közül emeljünk ki egyet: ez a „Baby Care Link” alkalmazás, amelyet egy angol cég fejlesztett ki bostoni megrendelésre. A probléma következő: tanácsadói központ koraszülött gyermekek ápolására. A múltban, Bostonban, létezett egy ilyen központ, amelyben napi 24 órában emberek válaszoltak kórházi telefonhívásokra, hogy orvosi, gyógyszerekkel kapcsoatos, biztosítási, stb, információkkal szolgáljanak. Egy nyilvánvalóan nagyon nagy igénybevétellel járó, fárasztó és felelősséggel teli munkát jelentett. A „Baby Care Link” ennek egy web változata. A tudást OWL-ben modellezték, a kérdéseket egy dedikált webszolgáltatáson keresztül lehet feltenni, és a rendszer az OWL következtetéseket felhasználva továbbítja a válaszokat. Ha valakinek erre megfelelő joga van (például orvosok, kutatók), akkor a tudásbázist bővítheti is.

6. Információk, dokumentumok, könyvek

A rendelkezésre álló cikkek, webportálok, leírások, blogok stb. száma ténylegesen óriási, és nagyon nehéz benne eligazodni, hiszen a szemantikus web, mind mondtuk, aktív kutatás–fejlesztési téma is. A David Beckett által fenntartott „Planet RDF" weblap[18] talán a legjobb kiindulópont.

A könyvek piaca is jelentős. Természetesen a könyvek kiválasztása kicsit személyes ízlés kérdése is; Powers[19] vagy Antoniu és van Harmelen[20] könyvei csak néhány a sok közül. De mindenképpen érdemes megjegyezni, hogy, néhány hónap eltéréssel, két könyv is megjelent 2005-ben magyarul; míg Gottdank Tibor könyve[21] inkább gyakorlati és bevezető jellegű, addig Szeredi Péter és kollégái inkább az elméleti hátterére koncentrálnak[5].

Természetesen az alapforrás a W3C ajánlások sorozat. Van közöttük inkább bevezető jellegű (mint például a már említett OWL áttekintés[14]) míg mások nehezebben olvashatóak. De mindenképpen ki kell emelni, hogy a W3C magyar irodájának és, elsősorban, a fordítónak, Pataki Ernőnek köszönhetően a teljes RDF, RDFS, és OWL dokumentum sorozat elérhető magyarul is, például a magyar iroda honlapjáról[22]. Ugyanezen a honlapon fel lehet jelentkezni a W3C magyar iroda hírlevelére is, amelyen keresztül a legfrissebb W3C-val kapcsolatos információkhoz lehet igen könnyen hozzájutni. Az iroda aktív szerepet is kíván játszani a magyar szemantikus web közösség létrehozásában, műhelykonferenciák szervezésében, stb.

Végül, de nem utolsósorban, érdemes a W3C Szemantikus Web honlapját[23] is rendszeresen látogatni. Ott nemcsak eszközökre vonatkozó információkat lehet találni, hanem a látogató követheti a W3C-n belüli legújabb fejlesztéseket is.

Irodalom

[1] Michael L. Dertouzos, *The Unfinished Revolution*. Harper Business, New York, 1995.

[2] Michael L. Dertouzos, *A félkész forradalom*. Typotex, Budapest, 2002.

[3] Tim Berners-Lee, James Hendler and Ora Lassila, *The Semantic Web*. In: *Scientific American*, **156**(5), 2001, pp. 35-43.

[4] Tim Berners-Lee, *Weaving the Web*. Harper, San Francisco, 1999.

[5] Szeredi Péter, Lukács Gergely, Benkő Tamás, *A szemantikus világháló elmélete és gyakorlata*. TypoTex, Budapest, 2005.

[6] Ivan Herman and Daniel Dardailler, *SVG Linearization and Accessibility*. In: *Computer Graphics Forum*, **21**(4), 2002, pp. 777-786.

[7] http://www.w3.org/RDF.

[8] SPARQL Query Language for RDF, E. Prud'hommeaux, A. Seaborne (Eds.), W3C Working Draft, http://www.w3.org/TR/rdf-sparql-query/, 2005.

[9] RDF Vocabulary Description Language 1.0: RDF Schema, D. Brickley, R.V. Guha (Eds.), W3C Recommendation, http://www.w3.org/TR/2004/REC-rdf-schema, 2004.

[10] http://www.w3.org/2004/OWL/.

[11] SKOS Core Vocabulary Specification, A. Miles, D. Brickley (Eds.), W3C Working Draft, http://www.w3.org/TR/swbp-skos-core-spec, 2005.

[12] http://www.w3.org/2005/rules/.

[13] http://www.wiwiss.fu-berlin.de/suhl/bizer/toolkits/.

[14] OWL Web Ontology Language - Overview, D. L. McGuiness, F. van Harmelen (Eds.), W3C Recommendation, http://www.w3.org/TR/2003/owl-features.

[15] Az OWL Web Ontológia Nyelv - Áttekintés, D. L. McGuiness, F. van Harmelen (Eds.), W3C Ajánlás, http://www.w3c.hu/forditasok/OWL/REC-owl-features-20040210.html.

[16] Uncertainty Reasoning for the Semantic Web (Workshop at ISWC2005), P.G. Costa, K.B. Laskey, K.J. Laskey, M. Pool (Eds.), http://ite.gmu.edu/~klaskey/URSW_Proceedings.pdf, 2005.

[17] W3C Workshop on Rule Languages for Interoperability, S. Hawke, C. de Sainte Marie, S. Tabet (Eds.), http://www.w3.org/2004/12/rules-ws/program2, 2005.

[18] http://planetrdf.com/guide/.

[19] Shelley Powers, *Practical RDF*. O'Reilly, Cambridge, 2003.

[20] Grigoris Antoniu and Frank van Harmelen, *A Semantic Web Primer*. The MIT Press, Cambridge, Massachusetts, 2004.

[21] Gottdank Tibor, *Szemantikus web*. ComputerBooks, Budapest, 2005.

[22] http://www.w3c.hu.

[23] http://www.w3.org/2001/sw/.