Ontológia-építő nyelvek értékelése, elemző összehasonlítása

MEO projekt

Scriptum Rt.

2005. május 1.

Bevezetés

A napjainkban rendelkezésre álló tudás és információ rendezése, megfelelő keretbe, szerkezetbe, rendszerbe foglalása óriási kihívást jelent. Az ontológiák arra tesznek kísérletet, hogy fogalmi szinten ragadják meg és írják le a világot.

Az ontológiákkal összefüggésben az ipari és a tudományos világ egyaránt felismerte azt, hogy a tudás feltérképezésének és leírásának folyamatában a középpontba – a fogalmak definiálása mellett – a fogalmak között fennálló szemantikai viszonyok minél teljesebb és részletesebb feltárása kell, hogy kerüljön.

A számítógépek számára érthető és kezelhető ontológiák formális nyelven vannak leírva. Az ontológianyelvek azonban „csak" az eszközt biztosítják a tartalom kifejezéséhez, önmagukban nem hordoznak információt. Az egyes ontológianyelvek közötti különbség éppen azáltal ragadható meg, hogy megvizsgáljuk, milyen lehetőségeket biztosítanak a tényleges tudás, illetve tartalom rögzítésére, kifejezésére. Az ontológia alapvető jellemzőit jelentősen befolyásolhatja a formális leírás során használt nyelv.

A MEO projekt keretében készülő ontológiák kifejezőerejét, hasznosítási lehetőségeit is meghatározza a majdan alkalmazott ontológianyelv. A projekt számára megfelelő ontológianyelv kiválasztását természetesen befolyásolja, illetve meghatározza a használt ontológiaépítési módszertan és az ontológiaszerkesztő is. Jelen tanulmány az ontológianyelvek elemzése és összehasonlítása, tulajdonságaik értékelése révén az ontológianyelvvel kapcsolatos döntések megalapozottságához és megkönnyítéséhez kíván hozzájárulni.

A tanulmányban a vizsgált ontológianyelvek vonatkozásában külön-külön sor kerül átfogó jellegű ismertetésükre, példákon keresztül is bemutatva a formalizálási lehetőségeket. Az ontológianyelvek összehasonlítása során szerzett tapasztalataink alapján ajánlást fogalmaztunk meg, továbbá összeállítottunk egy az ontológianyelvekkel kapcsolatos fogalomkészletet felölelő rövid glosszáriumot is. Végül a mellékletekben egy táblázatban összefoglaltuk a különböző ontológianyelvek legfontosabb jellemzőit, továbbá megadtunk egy „áttérési táblát" a különböző ontológianyelvek között is.

1 Ontológia nyelvek rövid leírása

A vizsgált nyelvekben közös, hogy mindegyikhez létezik valamilyen lekérdező-nyelv, mellyel a tudásbázisról kérhetünk információkat.

1.1 XOL

Az XML-based Ontology Exchange Language rövidítése.
Ez egy OKBC Lite alapú, XML formátumú ontológia nyelv.
Legutóbbi frissítés: 1999. augusztus; verziószám: 0.4
Támogatja az alap ontológiai struktúrákat (osztály, individuum, részosztály, példány).
A támogatott relációk szigorúan kétváltozósak, itt slot-nak hívják őket.
Megadható egy slot értelmezési tartománya és értékkészlete, valamint Cardinalityje is.
Azon tulajdonságok, amik mássá teszik a többi ontológianyelvtől:

- Egy osztály is lehet másik osztály példánya
- Többértékű slotok esetén megadható, hogy a slot típusa halmaz, multihalmaz vagy lista legyen
- Ki lehet mondani, hogy egy slot inverze egy másiknak.

1.2 SHOE

A Simple HTML Ontology Extensions rövidítése.
Ez egy függetlenül fejlesztett ontológianyelv, sem az RDF-et, sem az OKBC-t nem ismeri. Eredetileg nem tudásbázist reprezentálni készült, hanem egy HTML-kiegészítésként. Fejlesztése megszűnt, a fejlesztők átálltak az OWL és a DAML+OIL nyelvekre.
Legutóbbi frissítés: 2000. április; verziószám: 1.01
Támogatja az alap ontológiai struktúrákat (osztály, individuum, részosztály, példány).
Azon tulajdonságok, melyek mássá teszik a többi ontológianyelvtől:

- Tetszőleges változószámú relációkat definiálhatunk
- Horn-klóz formájú következtetési szabályokat definiálhatunk (megszorítás: sem a fej, sem a törzs nem lehet üres)

1.3 RFML

A Relational-Functional Markup Language rövidítése.
A RuleML alapjaira épül, kiegészítve függvény jellegű részekkel. Legutóbbi frissítés: 2000.
Nem igazán támogatott mostanra.
A Horn formulákat terjeszti ki, emiatt leginkább a SHOE alternatívája lehet.
2-nél több változós relációkat is képes kezelni. A függvényekről kevesebb kijelentést tehetünk, csak a definiálásukat támogatja, viszont lehetnek függvény értékű függvények. Ezeket egymásba ágyazva egészen bonyolult rendszereket hozhatunk létre.

1.4 RDFS

Az RDF Schema rövidítése, ahol az RDF a Resource Description Framework-öt jelenti.
W3C recommendation.
Legutóbbi frissítés: 2004. február; verziószám: 1.0
Támogatja az alap ontológiai struktúrákat (osztály, individuum, részosztály, példány).
Relációk között is értelmezhetjük a “része” kapcsolatot (van H), valamint minden relációnak megadhatjuk az értelmezési tartományát és értékkészletét.

Meghatunk konténereket (amiknek nemcsak egyedek, hanem osztályok is lehetnek az elemei, sőt önmaguk is (!)) és kollekciókat is. A kettő között az a különbség, hogy a konténerekre nem tudjuk azt mondani, hogy „a felsoroltakon kívül más elem nincs”, a kollekciókra pedig igen.
Továbbá megadhatunk Statement-eket (amiket RDF Triplet-nek is hívnak), ami egy bizonyos reláció fennállását mondja ki két tetszőleges dolog között (azaz egy reláció nemcsak két egyed, hanem akár két osztály között is fennállhat). Azon tulajdonságok, melyek mássá teszik a többi ontológianyelvtől:

- a nyílt és zárt világú konténerek léte
- nem feltétlenül kell élesen elválasztani az „egyed” és az „osztály” fogalmát

1.5 DAML+OIL

DAML: a Darpa Agent Markup Language rövidítése.
OIL: az Ontology Interface Layer rövidítése.
Az RDFS-re épülő ontológianyelv.
Legutóbbi frissítés: 2001. december
A DAML+OIL támogatja a teljes RDFS szintaxist. A benne definiálható relációk továbbra is kétváltozósak (csak így van értelme az RDF tripleteknek). Támogatja mind a globális, mind a lokális Cardinalityt.
Ebben az ontológianyelvben a következő elemek jelentek meg:

- Két vagy több osztályról kijelenthetjük, hogy diszjunktak
- Támogatja a particionálást
- Két osztályról kijelenthetjük, hogy egybeesnek
- Támogatja osztályok Boole-kombinációját (unió, metszet, negálás)
- Támogatja az enum (felsorolt) osztályt
- Támogatja a lokális megszorításokat is:
    - Egy C osztályra és R relációra kijelenthetjük, hogy ha cRx valamely C-beli c-re, akkor x egy bizonyos osztályba esik (*allValuesFrom*)
    - Egy C osztályra és R relációra kijelenthetjük, hogy minden C-beli c-re kell legyen legalább egy, bizonyos osztályba eső y, melyre cRy (*someValuesFrom*)
    - Ha a reláció alap adattípussal köt össze egy osztályt, akkor ennek megfelelője a *hasValue*, ezt is támogatja
- Relációkról kijelenthetjük, hogy tranzitívak, függvények, injektívek és hogy melyik reláció az inverzük
- Individuumokra sincs UNA, két egyedről ki lehet jelenteni, hogy azonosak, és azt is, hogy különböznek

A vizsgált XML alapú nyelvek közül csak a ráépülők (OWL, SWRL) azok, melyek legalább ekkora kifejezőerővel rendelkeznek.
A DAML+OIL eldönthető.

1.6 OWL – Web Ontology Language

Az OWL egy W3C szabvány, a Web Ontology Language „rövidítése”.
Legutóbbi frissítés: 2004. február.
Viszonylag csekély kifejezőerővel bíró nyelv, melyhez jól alkalmazható szerkesztőeszközöket (Protégé 2.1.2 OWL plug-in) és ellenőrzőket (Racer) fejlesztettek ki. Bár szerkesztése eléggé körülményes, az eredmény mindenképpen jól hasznosítható. (Működik a konzisztencia-ellenőrzés (Lite, DL), illetve az automatikus osztályozás a definíciók alapján. (Lite, DL))
Az OWL nyelvek fontos tulajdonsága, hogy nyílt világszemléletet alkalmaznak. Szintén lényeges, hogy az OWL nem él azzal a feltevéssel, hogy a különböző szóalakok különböző fogalmakat, individuumokat jelölnek. (Tehát nincs Unique Name Assumption.)
Az OWL legfőbb hiányossága, hogy változókat nem lehet használni benne, emiatt kisebb a kifejezőereje, mint egy olyan nyelvé, mely megenged elsőrendű logikai formulákat a definíciókban.
Előnye, hogy számos projektben alkalmazzák, van hozzá szerkesztőprogram, és az előbbieknek köszönhetően sok az ontológiaépítéssel kapcsolatos tapasztalat. A kettőnél nagyobb aritású relációk kifejezése kissé nehézkes, ehhez érdemes lenne egy segédprogramot készíteni.
Háromféle leírási szintaxisa van, de egyik sem túl emberközeli. (Az egyik XML alapú.)
3 nyelvi változata van: Lite, DL, illetve Full.

1.6.1 OWL Lite

A Lite változat a lehetséges kifejezőerőre vonatkozóan szoros megkötéseket tartalmaz: A relációk számossági korlátai csak 0-k vagy 1-ek lehetnek. Taxonómiákat már ezzel is könnyen ki lehet fejezni, de komplexebb dolgok leírása nem lehetséges, vagy körülményes. Nem lehet például osztályok közti diszjunktságot, általános tagadást kifejezni.
A szerepekről (relációkról) már itt is sokat állíthatunk, például megadhatjuk, hogy mi az inverze (I), tranzitív-e (S), szimmetrikus-e, illetve függvény jellegű-e (F) (de nem állíthatjuk, hogy az egyik reláció része a másiknak. (nincs H)).
Ehhez a változathoz viszonylag hatékony (legrosszabb esetben EXPTIME) következtető, ellenőrző rendszerek léteznek.

1.6.2 OWL DL

A DL változat az előbbinél nagyobb kifejezőerővel bír, a kifejezhető definíciók az SHION leíró logikának megfelelő osztályba esnek. Létezik egy ennél bővebb, még kiszámítható logika, de annak a kifejezései nem tartoznak a DL nyelv által kitűzött célok közé (, ám a Racer támogatja).
Ebben a változatban minden OWL konstrukciót használhatunk, de csak bizonyos megkötésekkel. Például nem lehet az osztályokat és az individuumokat keverni, nem lehet a tranzitív szerepekre számosságkorlátozással élni, …
Az OWL DL legfőbb tulajdonsága, hogy elméletileg még kiszámítható, bár nem hatékonyan (NEXPTIME legrosszabb esetben).

1.6.3 OWL Full

A Full változat esetén, lényegében minden OWL, RDF(S) konstrukció megengedett, minden korlátozás nélkül. Az ebben a változatban tett kijelentések helyességéről már elméletileg sem lehet minden esetben nyilatkozni.

1.7 SWRL

A Semantic Web Rules Language rövidítése.
Még az OWL nyelv kifejezőereje is viszonylag kicsi, köszönhetően annak, hogy – bár az osztályok közti kapcsolatokról igen sok mindent elmondhatunk benne – a különböző relációk egymással való kapcsolatáról szinte semmit nem állíthatunk. Ennek a hézagnak a betöltésére született meg az SWRL.
Legutóbbi frissítés: 2004. május; verziószám: 0.6
W3C Member Submission.
Az OWL DL és a RuleML nyelvek keresztezéséből jött létre; egy SWRL file tartalmazhat minden OWL DL-beli elemet, plusz Horn-formula alakú szabályokat, melyekben lehetnek elsőrendű változók. A formulák atomjai a következők lehetnek:

- „x eleme C” (ahol x egy változó vagy egy konstans individuum, C egy OWL osztály);
- „x P relációban van y-nal” (ahol x,y változó vagy konstans, P pedig egy OWL Property);
- „x ugyanaz, mint y” (x,y két változó vagy konstans);
- „x nem ugyanaz, mint y” (x,y két változó vagy konstans);
- összehasonlító operátorral képzett atom: =, <=, >=, ... (numerikus értékekre értelmes)
- aritmetikai operátorral képzett atom, pl. Add(x,y,z) akkor igaz, ha x+y=z;
- dátum/idő művelettel képzett atom;
- listaművelettel képzett atom.

Az SWRL szabályokkal kiegészített OWL DL eldönthetetlenné válik. Ez azonban elkerülhetetlennek látszik; ha már két reláció kompozícióját ki tudjuk fejezni, akkor eldönthetetlen lesz mindenképp (ami pl. az apa-testvér-nagybácsi relációknál történik).
A Protege szerkesztőhöz létezik SWRL plugin, szabályokat is lehet vele szerkeszteni (ez persze ismeri az OWL-t is).

1.8 SWRL-FOL

Az SWRL First Order Logic rövidítése.
Az SWRL nyelv egyik kifejlesztője kiterjesztette az SWRL-ben szereplő szabályokat. Ennek eredményeképp ebben a nyelvben már nem szabályok, hanem axiómák (Assertion-ok) definiálhatók (azaz eltűnik az eddigi kötelező implikációs forma). Tehát a teljes elsőrendű logikai készlet (és – vagy – nem konnektívák, minden – létezik elsőrendű kvantorok, az SWRL-beli atomok fölött) használható.
Az összes vizsgált XML alapú nyelv közül ez bír a legnagyobb kifejezőerővel; mindazonáltal kérdéses, hogy az alap SWRL-lel (mely ezzel szemben legalább W3C Member Submission) szembeni többlet tudás valóban kihasználható-e.
Legutóbbi frissítés: 2005. április 11.

1.9 OCML – Operational Conceptual Modelling Language

Legutóbbi frissítés: 1999.
Ezt a nyelvet tekinthetjük az Ontolingua kezelhető részének.
Nagy a kifejezőereje, de nem teljes a következtetési rendszere. (Létezhet olyan állítás, mely igaz, levezethető is lenne, de ez az eszköz nem tudja levezetni.)
Hasonlít egy programozási nyelvre az OCML leírásokban, definíciókban megengedett számítási eljárások használata is.
Ötvözi az objektum-orientált és a reláció orientált szemléletet.
Zárt világszemléletet alkalmaz. UNA.
Jelenleg nincs XML alakú leírása. A mostani LISP-szerű.
(Ez alapján készült.)

1.10 Ontolingua

Legutóbbi frissítés: 2002.
A szerkesztőeszköze nem publikus, de a honlapján keresztül lehet új projekteket indítani, amit a szerverük eltárol. Ez a legkifejezőbb a vizsgált nyelvek közül.
Több eszköz van hozzá, melyekkel az ontológiákon műveleteket végezhetünk, de sajnos – éppen a kifejezőerejének a nagysága miatt – nincs hozzá következtető rendszer.

1.11 FLogic – Frame Logic

Legutóbbi frissítés: 2003.
A Frame Logic kifejezésből származik a neve. Ezen a nyelven az osztályokról, objektumokról tehetünk logikai és relációs kijelentéseket. Az Flogic az objektumok és osztályok között nem tesz szintaktikailag különbséget, az osztályok is lehetnek más osztályok objektumai. Az OCML-lel ellentétben itt nem lehet a metódusokhoz/relációkhoz kiszámító programokat, eljárásokat rendelni.
Első rendű logikai kifejezőereje van. Az ehhez kapcsolódó bizonyító rendszere teljes és helyes. (De csak félig eldöntő.)
Zárt világszemléletet alkalmaz. Van UNA.
Ennek egy tovább fejlesztett változata (lehet majd) a FLORA-2 programnyelv.

1.12 CycL

Legutóbbi frissítés: 2002. (Hamarosan várható cycML)
Elsőrendű logikai formulákkal írhatjuk le a kijelentéseinket, kiegészítve számossági megszorításokkal. Ez a nyelv sem teljes. A következtető rendszerét támogatja bizonyos nyelvi elemekkel, hogy a következtetések gyorsulhassanak.

5 igazságértéket különböztet meg.
Nyílt világszemléletet alkalmaz. Egyedi név feltételezéssel él (UNA).
Egy kritika, melyet megfogalmaztak vele szemben, hogy túl bonyolult lett, nehéz kézzel bővíteni a tudásbázisát.
Van egy nyílt forrású változata, az OpenCyc, melyet jelenleg is fejlesztenek. Készül hozzá egy XML alapú cycML a tudásbázisok elmentéséhez, betöltéséhez. Létezik hozzá természetes nyelvű kiegészítés is, mellyel angol nyelvű szöveget generálhatunk vele a kijelentésekből.
Lehet benne kivételeket megadni, a függvények, relációk értékkészletét, értelmezési tartományát meghatározhatjuk.
Az ontológiákon belül kis mikro elméleteket lehet létrehozni.
LISP alapú szintaxisa van.

1.13 Loom

Legutóbbi frissítés: 1999.
Egy leíró logikai nyelv, azok között igen jó kifejezőerővel bír. (Meglepő módon itt a relációk definíciójában használhatunk változókat, bár azokkal nem képes annyi következtetésre.)
Osztályozó és következtető rendszert is tartalmaz, de a relációk definíciójában általában nem lehet diszjunkció (vagy).
A nyílt és a zárt világszemléletet felváltva alkalmazhatja az ontológia-készítő.
Az idő, szituációk kezeléséhez vannak eszközei.
LISP jellegű szintaxis. Nincs hozzá Java vagy C++ interface.
Sajnos nem túl jól dokumentált. 1999 óta nem fejlesztik.
Létezik hozzá webes szerkesztő (Ontosaurus).
LISP alapú, nyílt forrású, licence Loom Public license.

1.14 PowerLoom

Legutóbbi frissítés: 2003.
A Loom továbbfejlesztett változata. A nyílt világszemléletet alkalmazza, de tetszőleges része átállítható zárt világszemléletre (ami praktikus). Tartalmaz következtető és osztályozó rendszert is. (Támogatja a hibakeresést is, mivel a következtetési útját képes felfedni.)
Céljuk, hogy a későbbi változatokban az Ontolingua nyelven írt kifejezéseket is használni tudják. Jelenleg csak a Horn formulákat képes kezelni.
Hierarchikus modulokban tárolja a tudását.
Kicsit hiányos dokumentáció.
LISP alapra épül (→CLOS →STELLA), de képes Java és C++ nyelvekre is generálni kódot.
Csak nem-kereskedelmi felhasználásra ingyenes.

1.15 KM – Knowledge Machine

Legutóbbi frissítés: 2005.
Ez egy frame alapú nyelv, melyet sok kiegészítéssel láttak el. A tulajdonságok definiálásánál használhatunk kvantorokat, típusra és értékekre is tehetünk megszorításokat. A szituációk, események, cselekvések leírására is képes. Természetesen mivel ilyen nagy a kifejezőereje nem teljes. Tartozik hozzá következtető rendszer és lekérdező nyelv is.
Lehet kivételeket leírni, melyek a következtetést is befolyásolják. Készítettek hozzá egy segédkönyvtárat, melyben sok fogalom tipikus, illetve ezen a nyelven ajánlott leírása szerepel.
A szintaxisa LISP alapú, de a fejlesztők törekedtek arra, hogy angolul értelmesnek hangzó kijelentésekkel tehessünk kijelentéseket. A matematikai formulákat itt infix alakban lehet megadni.
A természetes nyelvű szövegkészítést/szövegfelismerést is támogatja valamilyen szinten, mivel tetszőleges objektumhoz, relációhoz rendelhetünk szöveget, melybe behelyettesítheti a hozzá kapcsolódó objektumok, relációk szövegeit.
Jelenleg is LISP-ben fejlesztik, LGPL licencű.

1.16 EPILOG

Az Episodic Logic rövidítése.
Last update: 1993 (revised 2000)
LISP alapú nyelv. Inkább tudásbázis-nyelvnek mondható, tekintve, hogy a taxonómiákat sem támogatja közvetlenül. Elsőrendű intenzionális (modális) logikai formulákat lehet beleírni. Egy ilyen formulában a szokásos elsőrendű készleten kívül időbeliség kifejezése is szerepelhet, hogy valami valamikor igaz (mindig igaz, előbb igaz, mint egy másik, ok-okozati összefüggést lehet beégetni, stb.)
A következtető rendszere nem teljes. Valószínűleg már nem is fejlesztik: az user manual 1993-as, 2000-ben nézték át egyszer, de nem változtattak rajta. Lehet továbbá megadni valószínűségeket is a szabályokhoz, pl. „bármely farkas 0.8 valószínűséggel szürke”. Az egyedek által tett kijelentéseket is támogatja.
Úgy tűnik, hogy a legtöbb munkát abba fektették, hogy egy „jó” angolról logikai formulába és vissza fordító termék jöjjön ki (például, egy prepozíciós igéhez megadható a helyes szórend és igeidő-tábla is). Nagy hátrány, hogy más természetes nyelvekre nem tűnik átültethetőnek.

1.17 SNePS

Legutóbbi frissítés: 2005.
A Semantic Network Processing System rövidülése.
Egy LISP alapú nyelv. Többnyire az egyedek által tett kijelentések értelmezésére, szituációk elemzésére helyezték a hangsúlyt. Létezik hozzá következtető rendszer, illetve angol nyelvű interface is. (A LISP-es nehezen kezelhetőnek tűnik.)
Zárt világszemléletű, van UNA
Licence GPL, jelenleg is fejlesztik.

2 Példák

2.1 XOL

A férfi fogalom és a felesége reláció:

```
<class>
  <name>
    Ferfi
  </name>
  <documentation>
    Pelda osztaly az XOL bemutatasara: a Ferfi! Ami egy Ember.
  </documentation>
  <subclass-of>
    Ember
  </subclass-of>
</class>

<slot>
  <name>
    felesege
  </name>
  <documentation>
    Pelda relacio: felesege.
     - a Ferfi halmazbol kepez a No halmazba;
     - inverze a ferje relacio;
     - egy ferfinak legfeljebb 20 felesege lehet;
     - tobb feleseg eseten halmazban taroljuk (ne
       listaban vagy multihalmazban) oket.
  </documentation>
  <domain>
    Ferfi
  </domain>
  <slot-value-type>
    No
  </slot-value-type>
  <slot-inverse>
    ferje
  </slot-inverse>
  <slot-maximum-cardinality>
    20
  </slot-maximum-cardinality>
  <slot-collection-type>
    set
  </slot-collection-type>
</slot>
```

2.2 SHOE

A SHOE többféle nyelvi formában létezik, pl. a HTML-SHOE és az XML-SHOE nyelvjárásokban. Ezen nyelvjárások csak a szintaxis bizonyos részleteiben térnek el; itt az XML-SHOE-t mutatjuk be.
Az alábbit formalizáljuk: vannak `Ember`-ek, minden `Ferfi Ember`; egy háromváltozós reláció a gyerek-anyja-apja reláció, ami `Ember×No×Ferfi`-beli; és egy következtetési szabályt is definiálunk, miszerint ha `gyerekAnyjaApja(g1,any,ap)` és `gyerekAnyjaApja(g2,any,ap)` és `g1<>g2`, akkor `testvere(g1,g2)`.

```
<DEF-CATEGORY NAME="Ember"/>

<DEF-CATEGORY NAME="Ferfi" ISA="Ember">
</DEF-CATEGORY>

<DEF-RELATION NAME="gyerekAnyjaApja">
  <DEF-ARG POS="1" TYPE="Ember"/>
  <DEF-ARG POS="2" TYPE="No"/>
  <DEF-ARG POS="3" TYPE="Ferfi"/>
</DEF-RELATION>

<DEF-INFERENCE>
  <INF-IF>
    <RELATION NAME="gyerekAnyjaApja">
      <ARG POS=1 VALUE="gyerek1" USAGE="VAR">
      <ARG POS=2 VALUE="apja" USAGE="VAR">
      <ARG POS=3 VALUE="anyja" USAGE="VAR">
    </RELATION>
    <RELATION NAME="gyerekAnyjaApja">
      <ARG POS=1 VALUE="gyerek2" USAGE="VAR">
      <ARG POS=2 VALUE="apja" USAGE="VAR">
      <ARG POS=3 VALUE="anyja" USAGE="VAR">
    </RELATION>
    <COMPARISON OP="notEqual">
      <ARG POS=1 VALUE="gyerek1" USAGE="VAR">
      <ARG POS=1 VALUE="gyerek2" USAGE="VAR">
    </COMPARISON>
  </INF-IF>
  <INF-THEN>
    <RELATION NAME="testvere">
      <ARG POS=1 VALUE="gyerek1" USAGE="VAR">
      <ARG POS=2 VALUE="gyerek2" USAGE="VAR">
    </RELATION>
  </INF-THEN>
</DEF-INFERENCE>
```

2.3 RDFS

Minden `Ferfi` egyben `Ember` is; a testvére reláció egy speciális rokona reláció.

```
<rdfs:Class rdf:ID="Ember"/>

<rdfs:Class rdf:ID="Ferfi">
  <rdfs:subClassOf rdf:resource="#Ember"/>
</rdfs:Class>

<rdf:Property rdf:ID="rokona">
  <rdfs:domain rdf:resource="#Ember"/>
  <rdfs:range rdf:resource="#Ember"/>
</rdf:Property>

<rdf:Property rdf:ID="testvere">
  <rdfs:subPropertyOf rdf:resource="#rokona"/>
</rdf:Property>
```

2.4 DAML+OIL

DAML+OIL-ban már megfogalmazható, hogy minden ember vagy férfi, vagy nő – és egyszerre csak az egyik lehet.

```
<daml:Class rdf:ID="Ember"/>

<daml:Class rdf:ID="Ferfi">
  <rdfs:subClassOf rdf:resource="#Ember"/>
</daml:Class>

<daml:Class rdf:ID="No">
  <rdfs:subClassOf rdf:resource="#Ember"/>
  <daml:disjointWith rdf:resource="#Ferfi"/>
</daml:Class>

<daml:Class rdf:about="#Person">
  <daml:disjointUnionOf rdf:parseType="daml:collection">
    <daml:Class rdf:about="#Man"/>
    <daml:Class rdf:about="#Woman"/>
  </daml:disjointUnionOf>
</daml:Class>
```

Továbbá az RDFS-ből ismert „a feleség egyféle speciális rokon”, az összes többi RDFS konstrukcióval megvalósítható, kissé más szintaxissal (sőt a rokon reláció tranzitívvá is deklarálható):

```
<daml:TransitiveProperty rdf:ID="rokona">
  <rdfs:domain rdf:resource="#Ember"/>
  <rdfs:range rdf:resource="#Ember"/>
</daml:ObjectProperty>

<daml:ObjectProperty rdf:ID="felesege">
  <rdfs:subPropertyOf rdf:resource="#rokona"/>
  <rdfs:domain rdf:resource="#Ferfi"/>
  <rdfs:range rdf:resource="#No"/>
</daml:ObjectProperty>
```

...és kijelenthetjük, hogy Géza és Béla más személyek:

```
<Ferfi rdf:ID="Geza"/>
<Ferfi rdf:ID="Bela">
  <daml:differentIndividualFrom rdf:resource="#Geza"/>
</Ferfi>
```

2.5 OWL

Az alábbiakban megpróbáljuk példák alapján is kifejezni az OWL nyelvek közti különbségeket.

2.5.1 OWL Lite

A következő állításokat formalizáljuk:
A gyereke reláció inverze a szülője reláció.

```
<owl:ObjectProperty rdf:ID="gyereke">
  <owl:inverseOf>
    <owl:ObjectProperty rdf:ID="szu2lo3je"/>
  </owl:inverseOf>
</owl:ObjectProperty>
```

A rokona reláció tranzitív.

```
<owl:ObjectProperty rdf:ID="rokona">
  <rdf:type
rdf:resource="http://www.w3.org/2002/07/owl#TransitiveProperty"/>
</owl:ObjectProperty>
```

A szülője reláció egy speciális rokona reláció.

```
<owl:ObjectProperty rdf:about="#szu2lo3je">
  <owl:inverseOf rdf:resource="#gyereke"/>
  <rdfs:subPropertyOf rdf:resource="#rokona"/>
</owl:ObjectProperty>
```

Minden szülőnek létezik gyereke.

```
<owl:Class rdf:ID="Szu2lo3">
  <rdfs:subClassOf>
    <owl:Restriction>
      <owl:onProperty>
        <owl:ObjectProperty rdf:ID="gyereke"/>
      </owl:onProperty>
      <owl:someValuesFrom
rdf:resource="http://www.w3.org/2002/07/owl#Thing"/>
    </owl:Restriction>
  </rdfs:subClassOf>
</owl:Class>
```

Minden embernek pontosan egy anyja van.

```
<owl:Class rdf:ID="Ember">
  <rdfs:subClassOf>
    <owl:Restriction>
      <owl:onProperty>
        <owl:ObjectProperty rdf:ID="anyja"/>
      </owl:onProperty>
      <owl:cardinality rdf:datatype="http://www.w3.org/2001/XMLSchema#int">
1 </owl:cardinality>
    </owl:Restriction>
  </rdfs:subClassOf>
</owl:Class>
```

2.5.2 OWL DL

A következő állítást az előző példához hasonlóan fejezhetjük ki (Csak itt megengedett a 3-as szám is.): Minden gépkocsinak van legalább 3 kereke és 1 motorja.
Nézőke a fényképezőgép tetején van, vagy lencsén keresztüli. Ezt a következőképpen fejezhetjük ki:

```
<owl:Class rdf:ID="Ne1zo3ke">
  <owl:oneOf rdf:parseType="Collection">
    <camera:Window rdf:about="#Lencse1kenKeresztu2l"/>
    <camera:Window rdf:about="#Afe1nyke1pezo3ge1pTeteje1n"/>
  </owl:oneOf>
</owl:Class>
```

2.5.3 OWL Full

A következő állítást formalizálva: A része reláció tranzitív, egy motornak legfeljebb 3 része van. Kapjuk az alábbit:

```
<owl:ObjectProperty rdf:about="#re1sze">
  <rdfs:domain rdf:resource="#Motor"/>
  <rdf:type
rdf:resource="http://www.w3.org/2002/07/owl#TransitiveProperty"/>
</owl:ObjectProperty>
<owl:Class rdf:ID="Motor">
  <rdfs:subClassOf>
    <owl:Restriction>
```

```
      <owl:onProperty>
        <owl:ObjectProperty rdf:ID="re1sze"/>
      </owl:onProperty>
     .<owl:maxCardinality
rdf:datatype="http://www.w3.org/2001/XMLSchema#int">3</owl:maxCardinality>
    </owl:Restriction>
  </rdfs:subClassOf>
</owl:Class>
```

Azt sajnos még ebben a nyelvben sem lehet kijelenteni, hogy a testvére reláció pontosan milyen viszonyban van a szülője relációval. Azt, hogy 4 ajtó tartozik egy autóhoz (általános tartozik/része relációval), nem állíthatunk, csak egy relációt vehetünk fel, ami az autók és ajtók közötti és arra mondhatjuk, hogy egy autóhoz 4 ajtó tartozik (valószínűleg autóajtaja lesz a reláció neve).

2.6 SWRL

SWRL nyelven már kifejezhető a testvér kapcsolat az apja-anyja relációk segítségével:
„ha person1 apja theirFather, person2 apja theirFather, person1 anyja theirMother és person2 anyja theirMother, AKKOR person1 testvére person2 és person2 testvére person1.” – itt ugyan a testvér reláció szimmetrikus volta miatt elég lenne csak az egyik oldalt következtetni, de így is lehet.

```
<swrl:Variable rdf:ID="person1"/>                          %változódeklarációk
<swrl:Variable rdf:ID="person2”/>
<swrl:Variable rdf:ID="theirMother"/>
<swrl:Variable rdf:ID="theirFather"/>
<ruleml:Imp>                                               %SWRL szabály
  <ruleml:body rdf:parseType="Collection">                 %feltételek halmaza
    <swrl:individualPropertyAtom>                          %egy atom:
      <swrl:propertyPredicate rdf:resource="&eg;apja"/>    %person1
      <swrl:argument1 rdf:resource="#person1" />           %apja
      <swrl:argument2 rdf:resource="#theirFather />        %theirFather
    </swrl:individualPropertyAtom>
    <swrl:individualPropertyAtom>                          %person2-nek is
      <swrl:propertyPredicate rdf:resource="&eg;apja"/>
      <swrl:argument1 rdf:resource="#person2" />
      <swrl:argument2 rdf:resource="#theirFather />
    </swrl:individualPropertyAtom>
    <swrl:individualPropertyAtom>                          %anyja theirMother
      <swrl:propertyPredicate rdf:resource="&eg;anyja"/>
      <swrl:argument1 rdf:resource="#person2" />
      <swrl:argument2 rdf:resource="#theirMother />
    </swrl:individualPropertyAtom>
    <swrl:individualPropertyAtom>                          %meg neki is
      <swrl:propertyPredicate rdf:resource="&eg;anyja"/>
      <swrl:argument1 rdf:resource="#person1" />
      <swrl:argument2 rdf:resource="#theirMother />
    </swrl:individualPropertyAtom>
  </ruleml:body>
  <ruleml:head rdf:parseType="Collection">                     %AKKOR:
    <swrl:individualPropertyAtom>
      <swrl:propertyPredicate rdf:resource="&eg;testvere"/> %testvére neki
      <swrl:argument1 rdf:resource="#person1" />
      <swrl:argument2 rdf:resource="#person2” />
    </swrl:individualPropertyAtom>
    <swrl:individualPropertyAtom>
      <swrl:propertyPredicate rdf:resource="&eg;testvere"/> %meg ő is neki
      <swrl:argument1 rdf:resource="#person2" />
      <swrl:argument2 rdf:resource="#person1" />
    </swrl:individualPropertyAtom>
  </ruleml:head>
</ruleml:Imp>
```

Azt sajnos még mindig nem tudjuk kimondani, hogy “Ha x része y ÉS z része y, akkor x része z VAGY z része x.” Viszont a féltestvér relációt igen: “ha person1 anyja theirMother ÉS person2 anyja theirMother ÉS person1 apja person1father ÉS person2 apja person2father ÉS owl:differentFrom(person1father,person2father), AKKOR person1 féltestvére person2”. Tehát az OWL DL-beli differentFrom (és a sameAs) használható. Más tagadás, diszjunkció, egzisztenciális kvantor stb. viszont nincs.

2.7 SWRL FOL

Itt már ki lehet fejezni a „része” reláció következő tulajdonságát: „ha x része y, és z része y, akkor x része z vagy z része x.” A következő részlet segítségével:

```
<Assertion owlx:name="Példa">
  <Forall>
    <ruleml:Var type="Thing">x</ruleml:Var>
    <ruleml:Var type="Thing">y</ruleml:Var>
    <ruleml:Var type="Thing">z</ruleml:Var>
    <Implies>
      <And>
        <swrlx:individualPropertyAtom swrlx:property="resze">
          <ruleml:Var>x</ruleml:Var>
          <ruleml:Var>y</ruleml:Var>
        </swrlx:individualPropertyAtom>
        <swrlx:individualPropertyAtom swrlx:property="resze">
          <ruleml:Var>z</ruleml:Var>
          <ruleml:Var>y</ruleml:Var>
        </swrlx:individualPropertyAtom>
      </And>
      <Or>
        <swrlx:individualPropertyAtom swrlx:property="resze">
          <ruleml:Var>x</ruleml:Var>
          <ruleml:Var>z</ruleml:Var>
        </swrlx:individualPropertyAtom>
        <swrlx:individualPropertyAtom swrlx:property="resze">
          <ruleml:Var>z</ruleml:Var>
          <ruleml:Var>x</ruleml:Var>
        </swrlx:individualPropertyAtom>
      </Or>
  </Forall>
</Assertion>
```

2.8 OCML

A következő állítást formalizálva: A testvére reláció a szülő relációkkal kifejezve. Kapjuk az alábbit:

```
(def-relation TESTVERE (?ki ?kinek)
”hu: Akkor és csak akkor áll ?ki ?kinek-kel TESTVERE relációban, ha ?ki-nek
és ?kinek-nek megegyeznek a szüleik.
en: ?ki and ?kinek are in relation TESTVERE if and only if the parents of
?ki and ?kinek are the same.”
:no-op (:iff-def
    (and
      (e1lo3le1ny ?kinek)
      (e1lo3le1ny ?ki)
      (exists (?valaki) (szu2lo3 ?valaki ?ki))
      (exists (?valaki) (szu2lo3 ?valaki ?kinek))
      (forall (?ba1rki)
        (<=> (szu2lo3 ?ba1rki ?ki) (szu2lo3 ?ba1rki ?kinek))
      )
```

```
    )
  )
)
```

2.9 Ontolingua

Az OCML beli példa ezt is jellemzi. Egy halmazokat használó változat alább.

```
(def-relation TESTVERE (?ki ?kinek)
"hu: Akkor és csak akkor áll ?ki ?kinek-kel TESTVERE relációban, ha ?ki-nek
és ?kinek-nek megegyeznek a szüleik.
en: ?ki and ?kinek are in relation TESTVERE if and only if the parents of
?ki and ?kinek are the same."
:no-op (:iff-def
    (and
      (Instance-of ?kinek e1lo3le1ny)
      (Instance-of ?ki e1lo3le1ny)
      (exists (?valaki) (szu2lo3 ?valaki ?ki))
      (exists (?valaki) (szu2lo3 ?valaki ?kinek))
      (forall (?ba1rki)
        (<=> (szu2lo3 ?ba1rki ?ki) (szu2lo3 ?ba1rki ?kinek))
      )
    )
  )
)
```

2.10 FLogic

Megint a testvére relációt formalizálva a következőt kapjuk (most hallgatólagosan feltettük, hogy mindenkinek legfeljebb 1 anyja, illetve apja van, de természetesen ez is kifejezhető):

```
FORALL X, Y X[testve1re->>Y] <- EXISTS Apa, Anya Apa[gyereke->>X] AND
Apa[gyereke->>Y] AND Anya[gyereke->>X] AND Anya[gyereke->>Y]
```

2.11 cycL

Azt szeretnénk kifejezni, hogy akinek az anyjának van legalább 2 gyereke, annak van testvére.

```
(#$forAll ?E1LO3LE1NY
  (#$implies
    (#$thereExists ?ANYA
      (#$and
        (#$gyereke ?ANYA ?E1LO3LE1NY)
        (#$thereExistAtLeast 2 ?GYEREKE (#$gyereke ?ANYA ?GYEREKE)
      )
    )
    (#$vanTestve1re ?E1LO3LE1NY)
  )
)
```

2.12 Loom

Újra a testvére relációt formalizáljuk, de most az apja és az anyja relációkkal:

```
(defrelation testve1re
  :is (:satisfies (?x ?y)
    (:and (/= ?x ?y)
      (:exists ?z
        (:and (apja ?x ?z)
```

```
          (apja ?y ?z))
      (:exists ?z
        (:and (anyja ?x ?z)
          (anyja ?y ?z))))))
```

Egy bonyolultabb példa, mely a menyasszony fogalmát definiálja:

```
(defrelation eljegyzi :domain szemely :range szemely :charasteristics
:temporal :charasteristics :commutative :arity 3)
(defrelation eljegyzett :domain szemely :charasteristics: :temporal
  :is (:satisfies (?szemely)
    (:exists (?szemely2)
    (:exists (?hely)
      (:and (szemely ?szemely2)
        (hely ?hely)
        (:for-some (?eljegyzesido)
          (:and (past ?eljegyzesido)
            (:ends-at ?eljegyzesido
              (eljegyzi ?szemely ?szemely2 ?hely)
))))))))
```

2.13 PowerLoom

Az idős ember fogalmát definiáljuk (a kort nem tudjuk automatikusan számíttatni, de beállíthatjuk egy konkrét személy esetében):

```
(deffunction kor ((?szemely szemely)) :-> (?kor INTEGER))
(defconcept ido3sEmber (?sz szemely)
  :documentation "Olyan szemely, aki elmult 60 eves."
  :<=> (and (szemely ?szemely)
      (> (kor ?szemely) 60)))
```

2.14 KM – The Knowledge Machine

A testvére relációt fejezzük ki az apja, anyja reláció segítségével. Az első fele azt fejezi ki, hogy a testvére reláció egy állandó reláció, egy-egy személynek több testvére is lehet, illetve több személy is lehet. A második fele lényegében a definíciója a testvére relációnak.

```
(testvelre has (instance-of (Slot))
  (cardinality (N-to-N))
  (fluent-status (*Non-Fluent)))
(every Szemely has (testvelre
    (if
      ((valaki has (instance-of Szemely))
        and ((the anyja of Self) = (the anyja of valaki))
        and ((the apja of Self) = (the apja of valaki))
        and (valaki /= Self))
      then valaki)
  )
)
```

Ha valakivel rossz dolog történik, akkor „fentről” „le” kerül.

```
(rosszDologto2rtelnik has (superclasses (Action)))
(every rosszDologTo2rtelnik has
  (object ((a Szemely)))
  (add-list ((:triple (the object of Self) allapot *lent)))
  (del-list ((:triple (the object of Self) allapot *fent)))
)
```

2.15 EPILOG

A testvére reláció:

```
(kn '(A sz1 (sz1 Szemely)
     (E sz2
       (sz2 Szemely and (not sz1 equals sz2)
       (E szulo
         ((szulo Szemely) and (szulo szu2lo3 sz2)
         and (szulo szu2lo3 sz3))))
     (sz1 testver sz2)))
```

Ami egyszer fent van, az (később) lesz lent is.

```
(kn '(A x (x PhysicalObject)
     (
       (A t_ep ( (x fent)*t_ep )
       (
         (E tkesobb_ep)
         (
           (t_ep before tkesobb_ep) and
           ((x lent)*tkesobb_ep))
          )
        )
      )
)
```

2.16 SNePS

A testvére relációt fejezzük ki a LISP-es nyelven:

```
(assert forall $szemely
  ant  (build member (build Skf anyja a1 *szemely) = anya class szemely)
    (build member (build Skf anyja a2 *anya) = testver class szemely)
  cq   (build agent *szemely act testvere object testver)
)
```

Ha valakivel rossz dolog történik, akkor „fentről” „le” kerül:

```
(assert
  whenever
    (build action rosszDolog szemely fent)
  do (build action allapot szemely le)
)
```

3 Ajánlásunk

Azt ajánljuk, hogy több nyelven is írjuk le a csúcsontológiát, így lehetne többféle célra is használni. (Itt a több nyelven a formális nyelveket értjük, de természetesen a több természetes nyelven való leírásnak is van haszna, értelme.)
A természetes nyelvű leírás alkalmazható lenne arra, hogy a szerkesztők szándékát kifejezze, ellenőrizni lehetne (bár nem automatikusan), hogy a formális leírás mennyire híven tükrözi ezt a szándékot.
Egy olyan formális leírás, mely a másodrendű logikát, vagy az elsőrendű logikát megengedi, az a természetes nyelvi definíció egy viszonylag jól követhető átírása.[1] Az, hogy itt nyílt vagy zárt világszemléletet kövessünk vitára érdemes.
Egy OWL DL szintű formális leírás alkalmazható lenne a modell konzisztenciájának ellenőrzésére, illetve az osztályok közötti kapcsolatok ellenőrzésére. Az OWL DL-es leírás előnye, hogy XML-es szintaxisa miatt az ontológia egy másik nyelvre átalakítása viszonylag egyszerű. Az SWRL egy ilyen továbblépés lehet, ha nem elegendő a DL-es kifejezőerő. Az OWL DL további előnye, hogy több szerkesztő, illetve konzisztencia ellenőrző rendszer készült hozzá.
A távközlési ontológia létrehozásakor valószínűleg nagy szerepük lesz az eseményeknek, relációk közti kapcsolatoknak. Amennyiben ezekről bonyolultabb kijelentéseket szeretnénk tenni és a következtetésekkor ezeket alkalmazni szeretnénk, akkor célszerű olyan nyelvet választani, amelyik ezt valamelyest támogatja. (Erre legmegfelelőbbnek a frame alapú nyelvek tűnnek, bár kétségtelen, hogy ezekben megfogalmazni bonyolultabb dolgokat szinte már programozásnak minősül.) Amennyiben különböző forgatókönyveket is modellezni kell, akkor néhány speciális nyelv erre is alkalmas. (KM, EPILOG, SNePS)

[1] Már az elsőrendű logikában is nagyon sok mindent le lehet írni, és a másodrendű logikából nem biztos, hogy használnánk olyasmit, amihez biztosan szükség van a másodrendű logikára, de bizonyos dolgokat esetleg könnyebb lehet kifejezni másodrendű logikai szintaxissal.

4 Fogalmak

Összeállítottunk egy az ontológianyelvekkel kapcsolatos fogalomkészletet felölelő rövid glosszáriumot. Az alábbiakban ezeket a fogalmakat mutatjuk be.

4.1 A táblázatban szereplő fogalmak

4.1.1 Metaclass

Osztályokból/fogalmakból álló fogalom.

4.1.2 Partitions

Diszjunkt fogalmakból álló halmazt/fogalmat definiálhatunk.

4.1.3 Instance attribute

Az egyedeknek lehetnek különböző tulajdonságaik.

4.1.4 Class attribute

Az fogalmaknak megadhatunk tulajdonságokat, melyek minden – a fogalomhoz tartozó – egyedre érvényesek.

4.1.5 Local scope

A fogalmakról megadható, hogy egy szakontológia részei-e.

4.1.6 Global scope

A fogalmakról megadható, hogy egy általános ontológia részei-e.

4.1.7 Frame

A frame alapú nyelvekben egy objektum. Lehet fogalom, vagy egyed is.

4.1.8 Facet

Egy frame egy adott slotjára vonatkozó megszorítás.

4.1.9 Slot

A frame egy tulajdonsága.

4.1.10 Default slot value

Alapértelmezett slot érték, arra az esetre, ha nem lenne kitöltve.

4.1.11 Type constraint

Meghatározhatjuk, hogy milyen típusú értékeket vehet fel az adott tulajdonság.

4.1.12 Cardinality constraint

Meghatározhatjuk, hogy hány értéket vehet fel az adott tulajdonság.

4.1.13 Exhaustive subclass partitions

A fogalmat definiáljuk lényegében a diszjunkt részfogalmai által.

4.1.14 Disjoint decompositions

Egy fogalomnak felsorolhatjuk néhány diszjunkt alosztályát, melyek nem feltétlenül adják ki az egész fogalmat.

4.1.15 Integrity constraint

Az értékek halmazára tehetünk megszorítást.

4.1.16 Operational definition

A reláció kiszámításához egy eljárást, vagy definíciót adhatunk.

4.1.17 Named axioms

Az axiómák a többi elemtől (fogalmak, relációk) elkülönülve definiálhatóak-e?

4.1.18 Embedded axioms

Az axiómák a fogalmakhoz, vagy a relációkhoz kötve definiálhatóak-e?

4.1.19 Instances of concepts

Lehet-e fogalmakhoz tartozó egyedeket definiálni?

4.1.20 Fact

Egyedek közti relációk.

4.1.21 Claim

Egyedek által tett kijelentések.

4.2 Nyílt világszemlélet

A nyílt világszemléletben a fogalmak definiálásakor figyelni kell arra, hogy ha valami nem szerepel az ontológiában, arról nem teszi fel, hogy nem létezik, hanem úgy tekinti, hogy vagy létezik, vagy nem.
A nyílt világszemléletben megfogalmazhatunk zárt világszemléletnek megfelelő kijelentéseket is, amennyiben a nyelv erre lehetőséget ad(, általában ad).
Nyílt világszemléletű nyelvek: OWL, RDF(S), CycL

4.3 Zárt világszemlélet

A nyílt világszemlélettel szemben ekkor csak a leírt dolgokat tekinti létezőnek, a nem leírtak pedig biztosan nem léteznek. A következtető rendszerek is építenek erre a nem-létezésre. A zárt világszemlélettel létrehozott ontológiát körülményesebb kinyitni, önmagán belül nem is lehetséges.
Zárt világszemléletű nyelvek: adatbázisok, prolog, OCML, Ontolingua

4.4 Leíró logika

A leíró logikában különféle fogalmak és köztük lévő relációkról/szerepekről tehetünk kijelentéseket. Különböző rövidítések terjedtek el:

4.4.1 S

Kijelenthetjük a szerepekről, hogy tranzitívak, megengedett a teljes negálás.

4.4.2 H

A szerepek közti hierarchiát megengedi, azaz állíthatjuk azt, hogy egy szerep bővebb, mint egy másik.

4.4.3 I

A szerepeknek meghatározhatjuk, hivatkozhatjuk az inverzét.

4.4.4 F

Meghatározhatjuk, hogy a szerep függvény-jellegű-e.

4.4.5 O

Nominálisokat használhatunk (individuumokra/egyedekre hivatkozhatunk fogalmakként).

4.4.6 N

Számossági korlátokat mondhatunk egy szerep teljes előfordulásaira egy adott fogalomhoz/egyedhez kapcsolódóan (azaz, megadhatjuk egy C osztályra és R relációra, hogy tetszőleges C-beli c elem esetén legalább/legfeljebb/pontosan mennyi olyan d elem létezik, melyre cRd). Ez máshol „Cardinality Constraints”-ként szerepel.

4.4.7 Q

Számossági korlátokat mondhatunk két fogalom és egy reláció kapcsolatára (azaz, megadhatjuk egy C és egy D osztályra, valamint egy R relációra, hogy tetszőleges C-beli c elem esetén legalább/legfeljebb/pontosan mennyi olyan D-beli elem létezik, melyekre cRd fennáll). Ez máshol „Quantified Cardinality Constraints”-ként szerepel.

A fenti definíciók értelmében N teljesülése maga után vonja F teljesülését, továbbá Q teljesülése maga után vonja N-ét, így egy leíró logika karakterizálásakor az F, N, Q tulajdonságok közül legfeljebb az egyiket soroljuk fel (így pl. van SHION leíró logika, de SHIOFN nincs).

4.4.8 SHION leíró logika

Ebben a logikában alkalmazhatunk teljes negálást a fogalmakra, tranzitivitást határozhatunk meg tetszőleges szerepre, hierarchiába szervezhetjük a szerepeket, a szerepek inverzére hivatkozhatunk, nominálisokra tekinthetünk fogalomként, illetve egy szerep összes tárgyára mondhatunk számossági korlátokat.

4.4.9 SHIF leíró logika

Itt nem használhatunk nominálisokat, de használhatunk inverz szerepeket, szerephierarchiát, tranzitív szerepeket, illetve a szerepekről állíthatjuk, hogy függvény-jellegűek.

4.5 Számítási bonyolultság

4.5.1 EXPTIME

A problémák azon osztályát tekintjük EXPTIME-belinek, melyekre létezik olyan algoritmus, mely a bemenet hosszában legfeljebb exponenciális idő alatt megadja a helyes kimenetet a problémára egy determinisztikus Turing gépen.

4.5.2 NEXPTIME

A problémák azon osztályát tekintjük NEXPTIME-belinek, melyekre létezik olyan algoritmus, mely a bemenet hosszában legfeljebb exponenciális idő alatt megadja a helyes kimenetet a problémára egy nem-determinisztikus Turing gépen.
EXPTIME$\subseteq$NEXPTIME.

4.6 Bizonyítások tulajdonságai

Néhány fogalmat nem pontosan használunk a közérthetőség miatt. (Például bizonyítást írunk levezetés helyett.) A pontos definíciókat Papadimitriou C. H.: Számítási bonyolultság című művében megtalálhatja az érdeklődő.

4.6.1 Helyesség

Egy levezetési rendszer helyes, ha csak olyan formulákat vezethetünk le, melyek az axiómák logikai következményei.
Általában a helyesség a bizonyítási módszereknél alapkövetelmény.

4.6.2 Teljesség

Ha a bizonyítási módszerünk teljes, akkor teljesül a következő: Ha egy állítás igaz, akkor azt be is tudja bizonyítani. (Kicsit szabadon fogalmazva...)
Az elsőrendű logikára még van teljességi tétel, de a másodrendűre már nincs.

4.6.3 Kiszámíthatóság

Az alábbi definíciók inkább a gyakorlati oldalról próbálják megmagyarázni a fogalmakat (a pontos definíciók a Számítási bonyolultság című könyvben):

Egy *L* nyelv **kiszámítható/eldönthető**, ha az összes, *L*-ben megfogalmazott *igaz* állításról be lehet bizonyítani, hogy igaz, és az összes *hamis* állításról, hogy hamis.
Egy *L* nyelv **félig eldönthető**, ha minden, az *L* nyelven megfogalmazott *igaz* állításról be lehet bizonyítani, hogy igaz; viszont a *hamis* állításokról nem tudjuk feltétlenül bebizonyítani, hogy hamisak.
Egy *L* nyelv **eldönthetetlen**, ha van olyan, az *L* nyelven megfogalmazott *igaz* állítás, melyről nem tudjuk bebizonyítani, hogy igaz; továbbá van olyan *hamis* állítás is, melyről nem tudjuk bebizonyítani, hogy hamis.

Például:
- a leíró logikák közül SHIQ és SHION (és minden alattuk) eldönthető;
- az elsőrendű logika félig eldönthető;
- a másodrendű logika eldönthetetlen.

4.7 Unique Name Assumption (UNA)

Egyedi név feltételezés. A különböző nevű fogalmakról, relációkról felteszi, hogy különbözőek. (Azok a nyelvek, melyek élnek ezzel a feltevéssel, többnyire megengedik, hogy kifejezzük két különböző nevű objektumról, hogy azok valójában azonosak.)

4.8 Horn-klóz (Horn-formula)

($X_1 \wedge X_2 \wedge ... \wedge X_n$) $\rightarrow$ ($Y_1 \wedge Y_2 \wedge ... \wedge Y_m$) alakú logikai kifejezés, ahol az X_i és Y_i predikátumok *nem* lehetnek negáltak. Ekkor a ($X_1 \wedge X_2 \wedge ... \wedge X_n$) részt *törzs*nek, a ($Y_1 \wedge Y_2 \wedge ... \wedge Y_m$) részt következménynek vagy *fej*nek is nevezzük.
(Valójában egy Horn-klóz csak egyetlen *Y* tagot enged meg a jobb oldalon, azonban *m* darab Horn-klóz segítségével definiálható egy fenti formájú, „kiterjesztett Horn-klóz")

4.9 LISP

Egy programozási nyelv. Prefix jellegű szintaxisa van. Eredetileg a mesterséges intelligencia kutatásokhoz fejlesztették ki.

4.10 KIF

A Knowledge Interchange Format rövidítése. Az Ontolingua és más LISP alapú nyelvek kedvelt adatcsere formátuma.

4.11 XML

W3C szabvány, adatok leírására fejlesztették ki, egy általános, szabványos adatcsere formátum.

4.12 OKBC

Az Open Knowledge Base Connectivity rövidítése.
Egy protokoll, ami a tudásreprezentációs rendszerek (Knowledge Representation Systems, KRS-ek) közti kommunikáció módját definiálja. Egy OKBC-t támogató KRS-nek többek közt az alábbi fogalmakat ismernie kell:

- osztály, 2-változós reláció (slot), individuum, részosztály, elem;
- slotnak minimum és maximum Cardinality,
- értelmezési tartomány, értékkészlet;
- slot inverze;
- numerikus slot értékére minimum/maximum.

4.13 Legalább háromváltozós relációk

Ha egy nyelv megengedi a kétváltozós relációkat, akkor ezekkel akármilyen (rögzített) aritású reláció kifejezhető. A módszerben egy *n*-változós relációt felbontunk *n-1* darab kétváltozós relációra, miközben bevezetünk *n-2* új osztályt. A kettőnél több aritású relációk engedélyezése tehát csak szintaktikai cukor.

5 Programok

5.1 Racer

Egy következtető rendszer. Zárt forrású, akadémiai és oktatási célokra ingyenes, LISP-ben írták. A leíró logikák közül ez támogatja a leginkább kifejezőeket.

5.2 FaCT

Egy következtető rendszer. Nyílt forrású, GPL, LISP-ben írták. Majdnem annyit tud, mint a Racer.

5.3 Pellet

Egy következtető rendszer OWL DL-hez. Nyílt forrású, GPL, Java-ban írták. OWL DL-hez megfelelő lehet. Az oldalán van weben kipróbálható változata

5.4 Protégé

Egy szerkesztő rendszer különböző ontológianyelvekhez. Sok bővítménye van, akad például vizualizációs, ellenőrző, illetve különböző nyelvekhez exportáló/konvertáló. Nyílt forrású, MIT licencű, Java nyelven íródott.

5.5 WebODE

Egy szerkesztő rendszer különféle nyelvekhez. Web alapú, nem sok mindent tudtunk meg róla. Hasonlít a Fogalomtár ontológia nyelvére.

5.6 FLORA-2

Egy érdekes kezdeményezés, egy programnyelv (egyetlen béta állapotú megvalósulása). A kifejezőereje a szintaxisa által nagyon jó, Prologgal, XSB-vel kompatíbilis (ezekre épül).
UNA van, bizonyos mértékben ki is kapcsolható.
A hatékonyságot is szem előtt tartották a tervezésekor.

6 Felhasznált irodalom

Oscar Corcho, Asunción Gómez-Pérez: Evaluating knowledge representation and reasoning capabilities of ontology specification languages, in: *Proceedings of the ECAI 2000 Workshop on Applications of Ontologies and Problem-Solving Methods*, Berlin, 2000.

Oscar Corcho, Mariano Fernández-López, Asunción Gómez-Pérez: Methodologies, tools and languages for building ontologies. Where is their meeting point?, in: *Data & Knowledge Engineering*, Vol. 46, 2003, pp.41-64.

John Domingue, Enrico Motta, Oscar Corcho: Knowledge Modelling in WebOnto and OCML A User Guide, http://kmi.open.ac.uk/projects/ocml/ocml-webonto-guide.zip, 1999

Michael Kifer, Georg Lausen, James Wu: Logical Foundations of Object Oriented and Frame Based Languages, http://www.cs.umbc.edu/771/papers/flogic.pdf, 1995

Web Ontology Language (OWL) Homepage, http://www.w3.org/2004/OWL/, 2004

Lukácsi Gergely, Dr. Szeredi Péter: A szemantikus web és az ontológiakezelés alapjai (előadás-sorozat a BME-n), http://www.cs.bme.hu/~stilgar/vima9000/

ontoprise GmbH, How to Write F-Logic Programs, http://www.ontoprise.de/documents/tutorial_flogic.pdf, 2004

Michael Kifer, Stony Brook: Knowledge- base Programming with Frames and Logic, http://flora.sourceforge.net/tutorial, 2004

Christos H. Papadimitriou: Számítási bonyolultság, 1999

ISX Corporation: Loom Users' Guide, 1991

Hans Chalupsky, Robert M. MacGregor, Thomas Russ: PowerLoom manual, http://www.isi.edu/isd/LOOM/PowerLoom/documentation/manual/manual.pdf, 2004

John Domingue, Enrico Motta, Oscar Corcho Garcia: Knowledge Modelling in WebOnto and OCML A User Guide, 1999

Sean Luke, Jeff Herlin: Shoe 1.01 Specification, 2000

Peter D. Karp, Vinay K. Chaudry, Jerome Thomere: XOL: An XML-Based Ontology Exchange Language, 1999

Guizhen Jang, Michael Kifer, Chang Zhao: Flora2 Users Manual 0.93, 2004

Dan Connolly, Frank van Harmelen, Ian Horrocks, Deborah L. McGuinness, Peter F. Patel-Schneider, Lynn Andrea Stein: DAML+OIL Reference Description, 2001

Dan Brickley, R.V. Guha, Brian McBride: RDF Vocabulary Description Language 1.0: RDF Schema, 2004

Cyc: The Syntax of cycL, 2002

Peter Clark and Bruce Porter: KM - The Knowledge Machine 2.0: User Manual, 2004

Stuart C. Shapiro: SNePS 2.6.1 users manual, 2004

Stephanie Schaeffer, Chung Hee Hwang, John de Haan, Lenhart K. Schubert: EPILOG: The Computational System for Episodic Logic USER'S GUIDE, 1993

Harold Boley: Markup Languages for Functional-Logic Programming, 2000

Harold Boley: The Relational-Functional Markup Language RFML Draft Specification, 2000

7 Mellékletek

7.1 A vizsgált ontológia nyelvek táblázatos összehasonlítása

Ontológia nyelvek összehasonlítása														
	Ontolingua	OCML	LOOM	FLogic	XOL	SHOE	RDFS	DAML +OIL	OWL DL	CycL	SWRL	KM	SNePS	EPILOG
Fogalmak														
Általános tulajdonságok														
Metaclasses	+	+	+	+	+	-	+	-	-	+	-	+	-	-
Diszjunkt fogalmakat foghatunk össze (partitions)	+	±	+	±	-	-	-	+	+	+	+	-	-	-
Dokumentáció	+	+	+	_	+	+	+	+	+	+	+	+	+	+
Attribútumok/ tulajdonságok														
Egyed tulajdonságok (instance attributes)	+	+	+	+	+	+	+	+	+	+	+	+	+	+
Osztály tulajdonságok (class attributes)	+	+	+	+	+	-	+	+	+	+	+	+	+	+
Local scope	+	+	+	+	+	+	+	+	+	+	+	+	+	+
Global scope	±	±	+	-	+	-	+	+	+	+	+	+	+	+
Facets														
Alapérték (Default slot value)	-	+	+	+	+	-	-	-	-	-	-	-	-	-
Típus megkötések (Type constraint)	+	+	+	+	+	+	+	+	+	+	+	+	+	+
Számossági megkötések (Cardinality constraints)	+	+	+	±	+	-	-	+	+	+	+	+	+	+
Slot documentation	+	+	+	-	+	+	+	+	+	+	+	+	+	+

Ontológia nyelvek összehasonlítása														
	Ontolingua	OCML	LOOM	FLogic	XOL	SHOE	RDFS	DAML +OIL	OWL DL	CycL	SWRL	KM	SNePS	EPILOG
Tranzitivitás	±	±	±	±	-	±	-	+	+	±	+	+	±	+
Szimmetria	±	±	±	±	-	±	-	±	+	±	+	±	±	+
Függvény	±	±	+	+	-	-	-	+	+	+	+	+	±	+
Injektitivás	±	±	±	±	-	-	-	+	+	±	+	±	±	+
Reláció hierarchia	±	±	±	±	-	±	+	+	+	+	+	+	±	+
Inverz reláció	+	±	±	±	+	±	-	+	+	±	+	+	±	+
Taxonómiák/osztályozások														
Részosztálya (Subclass of)	+	+	+	+	+	+	+	+	+	+	+	+	+	+
Meghatározhatjuk, hogy csak meghatározott diszjunkt fogalmakból áll az új fogalom (Exhaustive subclass partitions)	+	±	+	±	-	-	-	+	+	+	+	+	+	+
Részosztályokról diszjunktságot állíthatunk (Disjoint decompositions)	+	±	+	±	-	-	-	+	+	+	+	+	+	+
Nem részosztálya (Not subclass of)	±	-	±	-	-	-	-	+	-	+	-	+	+	+
Relációk és függvények														
n változós relációk, függvények (n-ary relations/functions)	+	+	+	±	±	+	±	±	±	+	±	+	+	+
Típus megkötések (Type constraints)	+	+	+	+	+	+	+	+	+	+	+	+	+	+
Argumentum értékekre megkötések (Integrity constraints)	+	+	+	+	-	-	-	-	±	+	±	+	+	+

Ontológia nyelvek összehasonlítása														
	Ontolingua	OCML	LOOM	FLogic	XOL	SHOE	RDFS	DAML +OIL	OWL DL	CycL	SWRL	KM	SNePS	EPILOG
Kiszámító algoritmus megadása (Operational definitions)	-	+	+	+	-	-	-	-	-	+	-	-	+	-
Axiómák														
Elsőrendű logika	+	+	+	+	-	Csak Horn	-	±	-	+	Csak Horn	+	+	+
Másodrendű logika	+	-	-	-	-	-	-	-	-	-	-	-	-	-
Fogalmakhoz nem kötött axiómák (Named axioms)	+	+	-	-	-	-	-	-	-	-	+	+	+	+
Fogalmakhoz kapcsolódó axiómák (Embedded axioms)	+	+	+	-	-	-	-	-	+	+	+	+	+	+
Egyedek/példányok (Instances)														
Fogalmak példányai (Instances of concepts)	+	+	+	+	+	+	+	+	+	+	+	+	+	+
Egyedek közti relációk (Facts)	+	+	+	+	+	+	+	+	+	+	+	+	+	+
Egyed által tett kijelentések (Claims)	-	-	-	-	-	+	±	±	±	-	-	+	+	+
További tulajdonságok														
Következtető rendszer	-	+, nem teljes	+, nem teljes	+, nem teljes	-	-	-	-	+	+, nem teljes	Hoolet (linux, nem teljes)	+, nem teljes	+, nem teljes	+, nem teljes
Szerkesztő	Forrása nem publikus	+	+	+	-	-	+	+	+	+	+	-	-	-
UNA	van	van	van	van	van	van	nincs	nincs	nincs	van	nincs	van	van	van
Világszemlélet	nyílt	zárt	nyílt / zárt	zárt	nyílt	nyílt	nyílt	nyílt	nyílt	nyílt	nyílt	nyílt / zárt	zárt	nyílt

Ontológia nyelvek összehasonlítása														
	Ontolingua	OCML	LOOM	FLogic	XOL	SHOE	RDFS	DAML +OIL	OWL DL	CycL	SWRL	KM	SNePS	EPILOG
Időbeliség támogatása	nincs	nincs	van	nincs	nincs	nincs	nincs	nincs	nincs	nincs	nincs	van	van	van
Rendszere	Frame	Frame	DL	Frame	Frame	Logikai	DL	DL	DL	Logikai	Logikai	Frame	Frame	Frame
Adattárolás formátuma	LISP	LISP	LISP	LISP	XML	XML	XML	XML	XML	LISP	XML	LISP	LISP	LISP
Megjegyzések			rosszul dokumentált, támogatja a térbeli kijelentéseket is										nehézkes	

7.2 Áttérési lehetőségek az ontológianyelvek között

Az alábbi ábrán ha egy A ontológianyelvből nyíl vezet egy B ontológianyelvbe, az azt jelenti, hogy bármely, az A nyelven leírt ontológia teljes egészében átírható B nyelvűre. Ha a nyílra van még írva egyéb kiegészítés (pl. „idő nélkül"), akkor az A nyelvű ontológiáknak csak bizonyos – de elég nagy – része (itt pl. azok a részek, amik nem állítanak semmit az időbeliségről) vihető át B nyelvűre. Végül, a szaggatott nyíl jelentése az, hogy A átvihető B-be, azonban onnan kezdve egy beviteli közeget kell teremteni a felhasználó és a B nyelvű ontológia között (ez pl. akkor fordul elő, ha EGY reláció felvétele A-ban megfelel HÁROM reláció felvételének B-ben)