OWL und kontrollierte natürliche Sprache

Übersetzungen zwischen OWL und Attempto Controlled English

CIS, SS 2009

Hans Leiß

Universität München
Centrum für Informations- und Sprachverarbeitung

26. Januar 2011

Literatur

- N.Fuchs, K.Kaljurand, T.Kuhn: Attempto Controlled English for Knowledge Representation. In: Reasoning Web 2008, LNCS 5224, 104-124, 2008
- K.Kaljurand: Attempto Controlled English as a Semantic WEB Language. Dissertationes Mathem. Univ. Tartuensis, Tartu University Press, 2007

ACE - Attempto Controlled English

Fragment des Englischen mit eindeutiger Semantik

N.E.Fuchs u.a., ETH Zürich

ACE Tools:

- Attempto Parsing Engine APE: übersetzt ACE-Texte in DRS/FOL/OWL
  attempto.ifi.uzh.ch/site/docs/ape_webservice.html
  Web-Client: attempto.ifi.uzh.ch/site/docs/ape/
- ACE Reasoner RACE: beweist ACE-Behauptung aus ACE-Annamen, beantwortet Fragen, testet Erfüllbarkeit (modifiziert den Satchmo-Prover von Bry/Manthey)
  attempto.ifi.uzh.ch/site/docs/race_webservice.html
- ACE View (Protege Plug-in) Editor für OWL-2 Ontologien und SWRL Regelmengen
- OWL Verbalizer: übersetzt OWL-2-Hierarchien in ACE-Text.

ACE Sprachumfang

1. Vokabular: vordefinierte Determinatoren, Konjunktionen, Präpositionen, Redewendungen (*there is/are*, *it is false that*)
2. Grammatik: attempto.ifi.uzh.ch/site/docs/ace_constructionrules.html

Übersetzung von ACE in OWL?

ACE-Texte können in FOL (first-order logic) übersetzt werden. Erfüllbarkeit und Folgerung sind für FOL unentscheidbar – was gilt für ACE-Texte? Man weiß etwas über folgende Fragmente:

- Cop = $\exists/\forall$ Nomen im Sg, $\pm$Kopula, prädikatives Adjektiv
- Rel = Relativsätze
- TV = $\pm$ transitives Verb
- DTV = $\pm$ bitransitives Verb
- GA = Personal- und Reflexivpronomen als Anapher (bezogen auf koindizierte Bezugs-NP)
  Bsp.: Wenn (Prof. N.)$_1$ (seiner$_1$ Sekretärin)$_2$ (einen Brief)$_3$ diktiert, tippt sie$_2$ ihn$_3$, wünscht ihm$_1$ einen schönen Tag, bringt ihn$_3$ zur Post und ruht sich$_2$ aus.

I.Pratt-Hartman und A.Third (2006) zeigen

1. Cop + Rel + TV + DTV ist entscheidbar (und $\subset$ ACE)
2. Cop + Rel + TV + GA ist unentscheidbar (und $\subset$ ACE)

Also: **Erfüllbarkeit/Folgerung für ACE sind unentscheidbar.**

K.Kaljurand (2007) gibt ein Fragment $\text{ACE}_1 \subset$ ACE an, das in OWL-2 übersetzbar, also entscheidbar ist. Rückübersetzung OWL $\mapsto \text{ACE}_2 \subset \text{ACE}_1$.

Demonstration ACE $\mapsto$ DRS, FOL und $\text{ACE}_1 \mapsto$ OWL

Web-Client `http://attempto.ifi.uzh.ch/site/docs/ape/`

- Beispiel: Every man is an animal. John is a man.
- möglich: unbekannte Wörter raten lassen
- möglich: Wortartentips geben wie *n:kitkat*, *p:Fritz*, *v:google*

ACE_1 Sprachumfang

- keine Fragesätze
- keine *of*-Konstruktion, kein angelsächsischer Genitiv (**Bill's*)
  keine Possessivpronomen, keine *whose*-Relativsätze
- keine Zahlen und Strings in Nominalphrasen
- keine prädikativen Nominalphrasen im Plural (**John is 3 men*)
- Übersetzung kann an unauflösbaren Anaphern scheitern

Diskursrepräsentationsstruktur (DRS)

1. Für Variablen (Diskursreferenten) $\vec{x} = x_1, \ldots, x_n$ und Bedingungen $\vec{C} = C_1, \ldots, C_m$ ist $(\vec{x}, \vec{C})$ eine DRS
2. Fur Relationssymbol $R^{(n)}$ und n Variable oder Konstante x_i ist $R(x_1, \ldots, x_n)$ eine Bedingung
3. Sind B, B_1, B_2 DRSs, so sind $\neg B$, $B_1 \Rightarrow B_2$, $B_1 \vee B_2$ Bedingungen

Übersetzung in FOL:

$$
\begin{aligned}
(\vec{x}, \vec{C})' &:= \exists \vec{x} \bigwedge \vec{C} \\
R(\vec{x})' &:= R(\vec{x}) \\
(\neg B)' &:= \neg\ B' \\
(B_1 \vee B_2)' &:= (B_1' \vee B_2') \\
((\vec{x}, \vec{C}) \Rightarrow B)' &:= \forall \vec{x} (\bigwedge \vec{C}' \Rightarrow B')
\end{aligned}
$$

Auflösung anaphorischer Beziehungen

Zugänglichkeit von Nominalphrasen für Anaphernauflösung:

- Eine Nominalphrase ist nicht zugänglich, wenn sie in einem negierten Satz auftritt.
  Bsp.: John does not enter a card. * A clerk takes the card.
- Eine Nominalphrase in einem all-quantifizierten oder einem *if-then*-Satz ist (von außerhalb des Satzes) nicht zugänglich.
  Bsp.: Every customer has a card. * A clerk takes the card.
- Eine Nominalphrase im *if*-Teil eines Satzes ist vom *then*-Teil aus zugänglich.
  Bsp. If a customer has a card, then a clerk takes the card.
- Eine Nominalphrase in einer Disjunktion ist nur von späteren Disjunktionsgliedern aus zugänglich.
  Bsp. A customer enters a card or drops the card. * A clerk takes the card.

Zugänglichkeit in DRSs

Eine (DRS)-Bedingung kann nur *zugängliche* Diskursreferenten als Argument benutzen, d.h. die $x \in \vec{x}$ einer zugänglichen DRS $(\vec{x}, \vec{C})$.

Für eine Bedingung in der DRS B_2 sind die in der DRS B_1 erklärten Diskursreferenten zugänglich, wenn eine der folgenden Beziehungen gilt:

1. $B_1 = B_2$,
2. B_1 die Bedingung $\neg B_2$ enthält,
3. B_1 eine Bedingung $B_2 \Rightarrow B_3$ enthält,
4. $B_1 \Rightarrow B_2$ eine Bedingung einer (übergeordneten) DRS ist,
5. B_1 eine Bedingung $B_2 \vee B_3$ enthält,
6. $B_1 \vee B_2$ eine Bedingung einer (übergeordneten) DRS ist,
7. B_2 eine DRS B_3 erreichen kann, und B_3 B_1 erreichen kann.

Bsp. $(\epsilon, (x, \langle man(x), \neg(y, car(y))\rangle) \Rightarrow (z, predicate(z, own, x, \underline{y})))$

Verbalisierung von OWL in ACE

Ausgangspunkt: die Sprache $\text{ACE}_1 \subset \text{ACE}$, für die eine Übersetzung $\text{ACE}_1 \longrightarrow \text{OWL}$ definiert wurde.

- Definition einer eindeutigen Teilsprache $\text{ACE}_2 \subset \text{ACE}_1 \subset \text{ACE}$. Da OWL gewisse Variablenverwendeungen nicht erlaubt, werden ausgeschlossen
  1. alle Formen anphorischer Bezüge; das schließt Sätze aus wie: „Every man who owns a cat hates a dog that bites *the cat*.“
  2. gewisse Satzverbindungen, deren Teilsätze disjunkte Variablen verwenden, speziell *if-then*-Sätze wie „if a man owns a car then there is a woman“
- Äquivalente Umformung von OWL-Formeln in ein OWL-Kernfragment, z.B. SymmetricObjectProperty(R) $\mapsto$ SubObjectProperty(R, InverseObjectProperty(R))
- OWL-Formeln des Fragments werden durch eine bidirektionale DCG in ACE_2 übersetzt.

ACE_2

Nominalphrasen:

1. Determinatoren: *every*, *no*, *nothing but*, *at least n*, *at most n*, *exactly n*
2. Indefinitpronomen: *everything*, *nothing*, *something*
3. Eigennamen und Reflexivpronomen können nicht durch Relativsätze modifiziert werden
4. Objektrelativsätze sind nicht erlaubt.

Verbalphrasen:

1. Verbalphrasen können nicht koordiniert werden (aber Relativsätze)
2. Verbalphrasen dürfen keine universell quantifizierten Objekte enthalten
3. Eigennamen dürfen nicht das Subjekt in Passivsätzen sein.

Sätze:

1. Sätze müssen mit Eigennamen oder universell quantifizierten Nominalphrasen beginnen, eine Verbalphrase haben und in einem Punkt enden
2. Ausnahme: zwei Formen von *if-then*-Sätzen sind erlaubt:
   - 2.1 if something X TV_1 something that TV_2 something that ... TV_n something Y then X TV Y.
   - 2.2 if something X TV_1 something Y then it is false that X TV Y.

ACE_2-Grammatik als Hornformeln

Übersetzung: OWL-Kernfragments $\mapsto$ ACE_2 durch eine bidirektionale Hornformelgrammatik (Definite Clause Grammar, DCG).

Die Kategorien der DCG-Regeln für ACE_2 haben also ein OWL-Ausgabeargument:

Kategorie(+Morph.Synt.Merkmale, +Variablen, -Formel)

⟨*Lexikalische Regeln*⟩≡

```
Wortkategorie(Merkmale,OWL-Formel) --> [Token]
```

⟨*Syntaktische Regeln*⟩≡

```
Kategorie(Merkmale,OWL-Formel) -->
    Kat1(Merkm1,Formel1), ..., KatN(MerkmN,FormelN).
```

Zur Vereinfachung der Lesbarkeit hier: DL- statt OWL-Formeln.

DL statt OWL

⟨*dl2owl.pl*⟩≡

```
tr(DL,OWL) :-
    nonvar(DL), DL =.. [F|Args], !,
    tr_all(Args,TrArgs),
    TrDL =.. [F|TrArgs],
    (dl2owl(TrDL,OWL) -> true ; TrDL = OWL).
tr(DL,OWL) :-
    var(DL), nonvar(OWL),
    OWL =.. [F|Args], !,
    tr_all(TrArgs,Args),
    TrOWL =.. [F|TrArgs],
    (dl2owl(DL,TrOWL) -> true ; DL = TrOWL).

tr_all([DL|DLs],[OWL|OWLs]) :-
    tr(DL,OWL), tr_all(DLs,OWLs).
tr_all([],[]).
```

⟨*dl2owl.pl*⟩+≡

```
dl2owl(set(C),          'OWLClass'(C)).
dl2owl(top,             'owl:Thing').
dl2owl(neg(C),          'ObjectComplementOf'(C)).
dl2owl(and(Cs),         'ObjectIntersectionOf'(Cs)).
dl2owl(or(Cs),          'ObjectUnionOf'(Cs)).
dl2owl(ex(R,C),         'ObjectSomeValuesFrom'(R,C)).
dl2owl(all(R,C),        'ObjectAllValuesFrom'(R,C)).
dl2owl(atleast(N,R,C),'ObjectMinCardinality'(N,R,C)).
dl2owl(atmost(N,R,C), 'ObjectMaxCardinality'(N,R,C)).
dl2owl(exactly(N,R,C),'ObjectExactCardinality'(N,R,C)).
dl2owl(sub(C,D),        'SubClassOf'(C,D)).
dl2owl(singleton(I),  'ObjectOneOf'(['Individual'(I)])).
```

⟨*dl2owl.pl*⟩+≡

```
  dl2owl(rel(R),          'ObjectProperty'(R)).
  dl2owl(invRel(R),       'InverseObjectProperty'(R)).
  dl2owl(subRel(R,S),     'SubObjectPropertyOf'(R,S)).
  dl2owl(disjRels(Rs),    'DisjointObjectProperties'(Rs)).

  dl2owl(self(R),         'ObjectExistsSelf'(R)).
```

Nomen (mit Parameter Numerus) bedeuten Mengen

⟨*ace2dl.pl*⟩≡

```
n(num=sg,set(top)) --> [thing].
n(num=pl,set(top)) --> [things].

n(num=sg,set(Tok)) --> [Tok].
n(num=pl,set(Tok)) --> [TokPl], {noun_pl(Tok,TokPl)}.
```

Verben (mit Parameter: Numerus, ob sie negiert sind, und ob sie Partizip Perfekt sind) bedeuten Relationen

⟨*ace2dl.pl*⟩+≡

```
tv(num=sg,neg=no,vbn=no,rel(Tok)) --> [TokSg],
    { verb_sg(Tok,TokSg) }.
tv(num=pl,neg=no,vbn=no,rel(Tok)) --> [Tok].
tv(num=sg,neg=yes,vbn=no,rel(Tok)) --> [Tok].
tv(num=pl,neg=yes,vbn=no,rel(Tok)) --> [Tok].
tv(_,_,vbn=yes,invRel(rel(Tok))) --> [TokPp, by],
    { verb_pp(Tok,TokPp) }.
```

Eigennamen stehen nicht im Lexikon, man braucht aber:

⟨*ace2dl.pl*⟩+≡

```
propn(singleton(PName)) --> [PName],
    { \+ member(PName,['Every','No',every,no,a,
                       itself,themselves]) }.
```

Funktionswörter:

Subjekt-Dets operieren auf den Klassen von N und VP:

⟨*ace2dl.pl*⟩+≡

```
% det_subj(+N,+VP,-Formel(N,VP))
det_subj(C1,C2,sub(C1,neg(C2))) --> ['No'] ; [no].
det_subj(C1,C2,sub(C1,C2)) --> ['Every'] ; [every].
```

Objekt-Dets operieren auf dem TV und dem Objekt-N:

⟨*ace2dl.pl*⟩+≡

```
% det_obj(Num,+TV,+N,-Formel(TV,N))

det_obj(num=sg,R,C,ex(R,C)) --> [a].
det_obj(num=pl,R,C,all(R,C)) --> [nothing,but].

det_obj(num=Num,R,C,atleast(N,R,C)) --> [at,least,N],
    { Num=sg,N=1 ; Num=pl,between(2,infinite,N) }.
det_obj(num=Num,R,C,atmost(N,R,C)) --> [at,most,N],
    { Num=sg,N=1 ; Num=pl,between(2,infinite,N) }.
det_obj(num=Num,R,C,exactly(N,R,C)) --> [exactly,N],
    { Num=sg,N=1 ; Num=pl,between(2,infinite,N) }.
```

Kopulaverben wandeln ggf. das Objekt-N ins Komplement

⟨*ace2dl.pl*⟩+≡

```
auxc(num=Num,C,C) --> ([is],{Num=sg} ; [are],{Num=pl}).
auxc(num=sg,C,neg(C)) --> [is, not].
auxc(num=pl,C,neg(C)) --> [are,not].
```

Hilfsverben bei transitiven Verben: ggf. Objekt-N ins Komplement

⟨*ace2dl.pl*⟩+≡

```
auxv(num=Num,neg=yes,vbn=no,C,neg(C))
    --> ([does],{Num=sg} ; [do],{Num=pl}),[not].
auxv(num=Num,neg=yes,vbn=yes,C,neg(C))
    --> ([is],{Num=sg} ; [are],{Num=pl}),[not].
auxv(num=Num,neg=no,vbn=yes,C,C)
    --> ([is],{Num=sg} ; [are],{Num=pl}).
auxv(_,      neg=no,vbn=no,C,C) --> [].
```

Junktoren

⟨*ace2dl.pl*⟩+≡

```
or( C1,C2, or([C1,C2])) --> [or].
and(C1,C2,and([C1,C2])) --> [and].
comma_and(C1,C2,and([C1,C2])) --> [',', and].
```

Phrasen: Klassenausdrücke können mit $\sqcup$, $\sqcap$, $\neg$, $\exists\, R\, C$ und $\forall\, R\, C$ aufgebaut werden. ACE_2 erlaubt Konjunktion, Disjunktion, Negation und Einbettung von Relativsätzen:

$$\text{cat} \sqcap \neg(\exists\, \text{like}\, (\text{dog} \sqcap ((\exists\, \text{attack}\, \text{postman}) \sqcup \text{Fido})))$$

wird zu

„something that is a cat and that does not like
a dog that attacks a postman or that is fido“

Komplexe Klassenausdrücke können nicht in Nominalphrasen übersetzt werden, da der Junktoren-Skopus in ACE nicht durch Klammern, sondern durch Interpretationsregeln festgelegt wird.

Verbalphrasen in Prädikativsätzen:

⟨*ace2dl.pl*⟩+≡

```
ibar(Num,C2) --> auxc(Num,C1,C2), cop(C1).
ibar(Num,C2) -->
    auxv(Num,Neg,Vbn,C1,C2), vp(Num,Neg,Vbn,C1).

cop(singleton(I)) --> propn(singleton(I)).
cop(and([C1,C2])) -->
    [a], n(num=sg,C1), relcoord(num=sg,C2).
cop(C) --> [a], n(num=sg,C).

vp(Num,Neg,Vbn,C) -->
    tv(Num,Neg,Vbn,R), np_obj(Num,R,C).
```

Subjekt-NPs sind Eigennamen und universell quantifizierte Nomen, ggf. mit Relativsatz:

⟨*ace2dl.pl*⟩+≡

```
np_subj(P,sub(C,P)) --> propn(C).
np_subj(P,sub(C,D)) --> det_subj(C,P,sub(C,D)),
                        n(num=sg,C).
np_subj(Y,sub(C,D)) -->
    det_subj(and([X,Coord]),Y,sub(C,D)),
    n(num=sg,X), relcoord(num=sg,Coord).
```

Objekt-NPs sind Eigennamen, Reflexivpronomen, und existentiell quantifizierte oder zahlbeschränkte Nomen, ggf. mit Relativsatz:

⟨*ace2dl.pl*⟩+≡

```
np_obj(_Num,R,ex(R,C)) --> propn(C).

np_obj(num=Num,R,self(R)) --> [itself],{Num=sg}
                            ; [themselves],{Num=pl}.

np_obj(_Num,R,D) --> det_obj(Num,R,C,D), n(Num,C).
np_obj(_Num,R,D) --> det_obj(Num,R,and([C,CRel]),D),
                     n(Num,C), relcoord(Num,CRel).
```

Die Bedeutung ist, was sich je nach Quantor aus dem tr.Verb `R` und der Nomen- bzw. Eigennamenbedeutung `C` ergibt.

Reflexiv: $\mathit{self}(R) := \{x | R(x,x)\}$

Relativsatzkoordination (vereinfacht)

⟨*ace2dl.pl*⟩+≡

```
relcoord(Num,and([C1,C2])) -->
    relcoord_1(Num,C1), [',', and], relcoord_1(Num,C2).
relcoord(Num,and([C1,C2])) -->
    relcoord_1(Num,C1), [',', and], relcoord_2(Num,C2).
relcoord(Num,and([C1,C2])) -->
    relcoord_2(Num,C1), [',', and], relcoord_1(Num,C2).
relcoord(Num,Coord) --> relcoord_1(Num,Coord).

relcoord_1(Num,or([C1,C2])) -->
    relcoord_2(Num,C1), [or], relcoord_1(Num,C2).
relcoord_1(Num,Coord) --> relcoord_2(Num,Coord).

relcoord_2(Num,and([C1,C2])) -->
    rel(Num,C1), [and], relcoord_2(Num,C2).
relcoord_2(Num,Coord) --> rel(Num,Coord).
```

Einfache Relativsätze

⟨*ace2dl.pl*⟩+≡

```
rel(Num,C) --> [that], ibar(Num,C).
```

Sätze

⟨*ace2dl.pl*⟩+≡

```
ip(Subset)
    --> np_subj(P,Subset), ibar(num=sg,P), ['.'].

ip(subRel(rel(R),rel(S)))
    --> ['If',something,'X',Rsg,something,'Y',
         then,'X',Ssg,'Y','.'],
        { verb_sg(R,Rsg), verb_sg(S,Ssg) }.
ip(disjRels([rel(R),rel(S)]))
    --> ['If',something,'X',Rsg,something,'Y',
         then,it,is,false,that,'X',Ssg,'Y','.'],
        { verb_sg(R,Rsg), verb_sg(S,Ssg) }.
```

⟨*Beispiel einer Übersetzung von ACE nach DL*⟩≡

```
?- [ace2dl,dl2owl]. % Uebersetzer laden

?- ip(DL,['If',something,'X',modifies,something,'Y',
          then,'X',changes,'Y','.'],[]).
DL = subRel(rel(modify), rel(change))
```

⟨*Beispiel einer Übersetzung von DL nach ACE*⟩≡

```
?- ip(subRel(rel(modify), rel(change)),ACE2,[]).
ACE2 = ['If', something, 'X', modifies, something, 'Y',
        then, 'X', changes|...]
```

⟨*Beispiel einer Übersetzung von DL nach OWL*⟩≡

```
?- tr(subRel(rel(modify),rel(change)), OWL).
OWL = 'SubObjectPropertyOf'('ObjectProperty'(modify),
                            'ObjectProperty'(change))
```

Beispiellexikon:

⟨*ace2dl_lexicon.pl*⟩≡

```
verb_sg(modify,modifies). verb_pp(modify,modified).
verb_sg(change,changes).  verb_pp(change,changed).
verb_sg(like,likes).      verb_pp(like,liked).

noun_pl(cat,cats).
noun_pl(dog,dogs).
noun_pl(mouse,mice).
noun_pl(man,men).
noun_pl(woman,women).
```

⟨*Beipiele ACE2 nach DL*⟩≡

```
?- ip(DL,
      [fido,is,a,dog,that,does,not,change,a,thing,'.'],
      []).
DL = sub(singleton(fido),
         and([set(dog), neg(ex(rel(change),set(top)))]))

?- ip(DL,
      [every,dog,is,a,dog,that,is,changed,by,a,cat,'.'],
      []).
DL = sub(set(dog),
         and([set(dog), ex(invRel(rel(change)),
                           set(cat))]))
```

⟨ *Beispiele DL nach ACE2* ⟩≡

```
?- ip(sub(set(dog), neg(and([set(dog),
                             ex(invRel(rel(change)),
                                set(top))])))), ACE,[]).
ACE = [no,dog,is,a,dog,that,is,changed,by,a,thing,'.']

?- ip(sub(set(woman),
          ex(rel(like),and([set(man),
                            atmost(1,rel(like),
                                   set(woman))])))),
       ACE,[]).
ACE = [every,woman,likes,a,man,that,likes,
                               at,most,1,woman,'.']
```

⟨*Beispiel DL nach OWL nach DL*⟩≡

```
?- tr(sub(set(man), ex(rel(like),
          and([set(man),atmost(2,rel(like),set(woman))]))),
      OWL), tr(DL,OWL).

OWL = 'SubClassOf'('OWLClass'(man),
        'ObjectSomeValuesFrom'('ObjectProperty'(like),
             'ObjectIntersectionOf'(['OWLClass'(man),
                  'ObjectMaxCardinality'(2,
                       'ObjectProperty'(like),
                            'OWLClass'(woman))]))),
DL = sub(set(man), ex(rel(like),
          and([set(man),atmost(2,rel(like),set(woman))]))).
```

⟨*Beispiel DL nach ACE*⟩≡

```
?- ip(sub(singleton(fido),
          all(rel(like),and([set(woman),
                             ex(rel(like),singleton(fido)
      ACE,[]).
ACE = [fido,likes,nothing,but,women,that,like,fido,'.']
```