Attempto Controlled English: Language, Tools and Applications

(1) Paraphrasing ACE texts

(2) ACE and Description Logics

Norbert E. Fuchs, Kaarel Kaljurand

Department of Informatics & Institute of Computational Linguistics
University of Zurich

{fuchs, kalju}@ifi.unizh.ch
http://attempto.ifi.unizh.ch

December 2006

Paraphrasing ACE

Paraphrasing in CNL

- Part of tools/interfaces
- Show how the text was interpreted
  - Scope interpretation
  - Reference interpretation
  - Interpretation of function words
- Normalize the text
  - Simplify the sentences
  - Make the sentences more concise/elegant

Previous work

- Paraphrasing in ACE 3
  - Resolve anaphors
    - A man enters a card and a code. If it is valid then …
    - A man enters a card and a code. If [the code] is valid then …
  - Display scopes and ellipsis
    - John enters a car and a code carefully.
    - John enters a card and {[enters] a code carefully}.
- Paraphrasing in PENG
  - Resolve anaphors

ACE 4 paraphrasing Requirements

- Paraphrase on the basis of the DRS
- Paraphrase must be semantically equivalent to the original
- Paraphrase must be syntactically different (but not too different) from the original
  - Possible if the language has syntactic sugar

Core ACE and NP ACE

- Two approaches
- First: Core ACE
- Later: NP ACE
  - Partly motivated by user feedback
- Can (and will) be combined in the future

Core ACE: motivation

- Unravel the structure of the sentence
- Use the smallest syntactic subset of ACE (i.e. the core)

Core ACE

- Reduced set of quantifiers, no passive, no Saxon genitive, …
- Only sentence negation
  - A man does not run.
  - There is a man. It is false that the man runs.
- No relative clauses
  - Every man who loves a woman who loves him smiles.
  - If a woman X1 loves a man X2 and the man X2 loves the woman X1 then the man X2 smiles.
- Only definite NPs (with variables) as anaphoric references
  - John sees somebody. He hates John's dog.
  - John sees somebody X. X hates a dog of John.

DRACE Core: implementation

- Deterministic (and user cannot modify its behaviour via parameters)
- Algorithm
  - Depth-first traversal of the DRS
  - In each box, verbalize the predicate-condition
  - For each predicate/object: verbalize it
    - knowing its location (I.e. which box it occurs in)
    - knowing if the object has been verbalized before
    - if another predicate from the same box has been verbalized before
    - if the object is used in a predicate-condition
- APE(DRACE(drs)) = drs

DRACE Core: example

[]
[A, B]
object(A, atomic, man, person, cardinality, count_unit, eq, 1)-1
predicate(B, unspecified, eat, A)-1
=>
[C]
predicate(C, unspecified, drink, A)-1

Result: *If a man X eats then the man X drinks.*

NP ACE: motivation

- Conciseness
  - Avoid explicit anaphoric references
  - Avoid sentence negation
- Be compatible with widespread rule and ontology language patters
  - Focus on *if-then* sentences

NP ACE

- *If-then* sentences are represented as *every*-sentences which means that
  - boolean combinations of sentences are done by relative clauses
  - *if*-part and *then*-part must share arguments
  - Passive must be often used
- Cannot express all ACE constructs (at the moment)
- Missing
  - NP pre-modifiers, VP modifiers, possessive constructs, ditransitive verbs, NP conjunction, numbers and strings, embedded if-then sentences

NP ACE: examples

- Argument sharing
  - If a man owns a dog then a woman owns a cat.
  - Paraphrase: ERROR
- Usage of passive
  - If a man owns a car then there is a woman who hates the car.
  - Paraphrase: Every car that is owned by a man is hated by a woman .

Discussion

- For a developer, makes DRS checking much faster
- For a user, explains interpretation rules
  - scopes
  - references (explicit vs. via relative pronouns)
  - PP-attachment?
- Not always syntactically different

PP attachment

- Possible solutions
  - A mouse sees a scientist REALLY with a knife.
  - With a knife, a mouse sees a scientist.
  - There is a scientist. A mouse sees him with a knife.
  - A mouse sees a scientist. That's done with a knife.
  - A mouse {sees a scientist with a knife}.

ACE and Description Logics

Description Logics

- Started out to make knowledge-bases more formal (to be able to automatically reason)
- Many kinds of logics of different expressivity
- Usually a fragment of first-order logic, but
  - a lot nicer syntax (but not as nice as it could be …)
  - decidable reasoning tasks
- Researchers are mainly interested in the complexity of different reasoning tasks
- Distinction between individuals, classes, and properties
- Some are standardized by W3C as OWL (Semantic Web ontology language)

DL as first-order logic

DL	FOL
$A \sqcup B$	$A(x) \vee B(x)$
$\neg A$	$\neg A(x)$
$A \sqcap B$	$A(x) \wedge B(x)$
$a : A \sqcap B$	$A(a) \wedge B(a)$
$\exists P.A$	$\exists(y)(P(x,y) \wedge A(y))$
$\forall P.A$	$\forall(y)(P(x,y) \Rightarrow A(y))$
$(a,b) : P, P(a,b), aPb$	$P(a,b)$
$A \sqsubseteq B$	$\forall(x)(A(x) \Rightarrow B(x))$
$A = B$	$\forall(x)(A(x) \Leftrightarrow B(x))$
$P \sqsubseteq R$	$\forall(x,y)(P(x,y) \Rightarrow R(x,y))$
$R = P^{-}, R = Inv(P)$	$\forall(x,y)(P(x,y) \Rightarrow R(y,x))$

Examples: classes, properties

- man $\subseteq$ human *(Every man is a human.)*
- human $\subseteq$ man $\cup$ woman *(Every human is a man or is a woman.)*
- dog $\subseteq \neg$ cat *(No dog is a cat.)*
- student $\subseteq$ {John, Mary} *(Every student is John or is Mary.)*
- love $\subseteq$ like *(Everybody who loves somebody likes him/her.)*
- Tr(taller-than) *(If somebody X is taller than somebody Y and Y is taller than somebody Z then X is taller than Z.)*
- taller-than $\subseteq$ Inv(shorter-than) *(If somebody X is taller than somebody Y then Y is shorter than X.)*

Classes, properties together

- man $\subseteq \exists$ own car *(Every man owns a car.)*
- man $\subseteq$ >= 3 own car *(Every man owns at least 3 cars.)*
- man $\subseteq \forall$ own car *(Everything that a man owns is a car.)*
- man $\subseteq \exists$ know {John} *(Every man knows John.)*
- $\exists$ write Thing $\subseteq$ author *(Everybody who writes something is an author.)*
- $\exists$ Inv(write) Thing $\subseteq$ book $\cup$ paper *(Everything that somebody writes is a book or is a paper.)*
- {John} $\subseteq \neg \exists$ like {Mary} *(John doesn't like Mary.)*

DL reasoning

- Different reasoning tasks:
  - Consistency (are there contradictory statements?)
  - Subclass hierarchy (is class1 more general than class2?)
  - Classification (is John a manager?)
  - Minimal inconsistent subset (what causes the ontology to be inconsistent?)
  - Concept rewriting (e.g. using named classes)
- Reasoners: Pellet, FaCT++, KAON2, RacerPro, Hoolet, …

Existing KBs, e.g. GALEN

- Ontology about medical terms and surgical procedures.
- Constructed in the 90s within the OpenGALEN project.
- Main applications:
  - Integration of clinical records, and
  - decision support.
- GALEN:
  - is very large (~35.000 concepts),
  - is fairly expressive (SHIF description logic),
  - has not been classified yet by any DL reasoner

ACE as ontology language

- Plain text
  - benefits: read/write everywhere; diff, indexing, storage, … are easy
- Natural language
  - benefits: everybody can (learn to) read/write, hides the difference between rules and ontologies, generate speech, …
- First step: (subset of) ACE as a syntactic variant of SROIQ DL (OWL 1.1)
  - benefits: access to DL reasoners and visualizers, generate large ACE texts from DL ontologies, …

Mapping ACE to DL

- Using the ACE parser, parse the ACE text into the DRS
- Convert the DRS into DL, which is tricky
  - some DRSs must be rejected (because ACE is more expressive than DL) and the user must be notified in a user-friendly way
  - some normalization must be applied to the DRS before it can be directly converted
  - difficult to obey DL's nonstructural restrictions
  - it's essentialy about converting FOL into DL (and it's not clear (to me) how to do this)
- There is also a partial implementation of the ACE to OWL DL RDF/XML mapping
- See our paper at PPSWR06, and "Writing OWL in ACE" on the Attempto website

Mapping DL to ACE

- Mapping
  - Class expressions map to ACE noun phrases (which can contain relative clauses)
  - Axioms map to ACE *every*-sentences
- Modify the outcome a little to gain readability
  - reorder coordinated classes (named classes come first, negations come later)
  - remove negation if possible

Mapping classes, etc

Named property	*Transitive verb, e.g. `like'*
InverseObjectProperty(R)	*Passive verb, e.g. `is liked by'*
Named class	*Common noun, e.g. `man'*
owl:Thing	something, thing
ObjectComplementOf(C)	something that is not a man; something that does not like a cat
ObjectIntersectionOf(C1 ... Cn)	something that is a person and that owns a car
ObjectOneOf(a)	*Propername, e.g `John'*
ObjectExistsSelf(R)	something that likes itself
ObjectMinCardinality(n R C)	something that is-friend-of at least 2 hackers

Mapping axioms

SubClassOf(C D)	Every man is a human. Everybody who owns a car is liked by John.
SubObjectPropertyOf(SubObjectPropertyChain(R1 ... Rn) S)	Everybody who owns something that is-part-of something X owns X.
DisjointObjectProperties(R1 ... Rn)	Nobody who is-child-of somebody X is-spouse-of X.

Example

hass-pet

- Every animal_lover is a person *and* hass-pet at least 3 things .
- Every cat_owner is a person *and* hass-pet a cat .
- Every dog_owner is a person *and* hass-pet a dog .
- Every old_lady hass-pet a animal *and* does not has_pet something that is not a cat .
- Every person that hass-pet a animal is a pet_owner .
- Every person that hass-pet a cat is a cat_owner .
- Every person that hass-pet a dog is a dog_owner .
- Every person that hass-pet at least 3 things is a animal_lover .
- Every pet_owner is a person *and* hass-pet a animal .
- Everything that hass-pet something X460 likess X460 .
- Everything that hass-pet something X522 is is_pet_ofed by X522 .
- Everything that hass-pet something is a person .
- Fred hass-pet Tibbs .
- Joe hass-pet Fido .
- Joe hass-pet at most 1 things .
- Minnie hass-pet Tom .
- Walt hass-pet Dewey .
- Walt hass-pet Huey .
- Walt hass-pet Louie .