Attempto Controlled English: Language, Tools and Applications

Logical Background

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

Overview

- From ACE to Discourse Representation Structures
- Extended Discourse Representation Structures
- From Discourse Representation Structures to Standard First-Order Logic
- Semantics of an ACE Text
- (Non-) Entailments

From ACE to Discourse Representation Structures

- ACE is based on Discourse Representation Theory
- DRT is a linguistic theory whose central concerns are
  - to assign meaning to natural language texts and discourses
  - to account for the context dependence of meaning
- What is this meaning?

From ACE to Discourse Representation Structures

- input: ACE text
- output: Extended Discourse Representation Structure (DRS)
  - uses flat syntactic variant of language of standard first-order logic
  - eases encoding of textual relations, e.g. anaphora
  - allows to represent plurals in first-order logic
  - internal representation: drs(Referents,Conditions)
- Attempto Parsing Engine (APE)
  - translates an ACE text into a discourse representation structure
  - implemented as a Definite Clause Grammar enhanced with feature structures (Prolog with ProFIT)
  - incorporates construction and interpretation rules

Example Translation

ACE Text

Every company that buys at least 2 standard machines gets a discount.

DRS

```
drs([], [drs( [A,B,C],
[object(A,atomic,company,object,cardinality, count_unit,eq,1)-1,
object(B,group,machine,object,cardinality,count_unit,geq,2)-1,
property(B,standard)-1,
predicate(C,event,buy,A,B)-1])
⇒
drs([D, E],
[object(D,atomic,discount,object,cardinality,count_unit,eq,1)-1,
predicate(E,event,get,A,D)-1])])
```

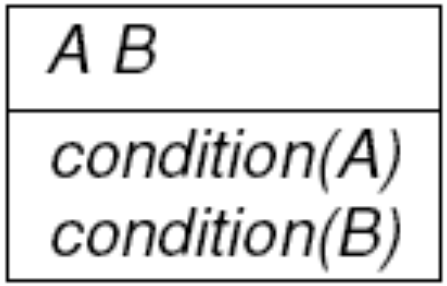
Pretty Printed Example DRS

Every company that buys at least 2 standard machines gets a discount.

```
[]
   [A, B, C]
   object(A, atomic, company, object, cardinality, count_unit, eq, 1)-1
   object(B, group, machine, object, cardinality, count_unit, geq, 2)-1
   property(B, standard)-1
   predicate(C, event, buy, A, B)-1
   =>
   [D, E]
   object(D, atomic, discount, object, cardinality, count_unit, eq, 1)-1
   predicate(E, event, get, A, D)-1
```

Extended Discourse Representation Structures

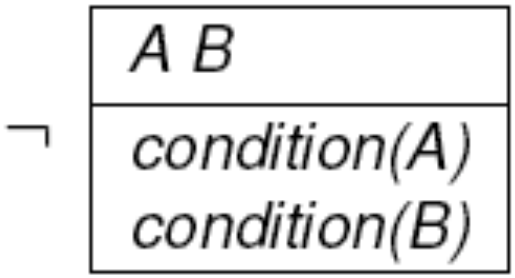
- top-most discourse representation structure
- Prolog notation: drs([A,B], [condition(A), condition(B)])
- pretty-printed



- conditions
  - simple conditions are logical atoms
  - complex conditions are constructed recursively from other discourse representation structures
  - result is tree of embedded discourse representation structures

Extended Discourse Representation Structures

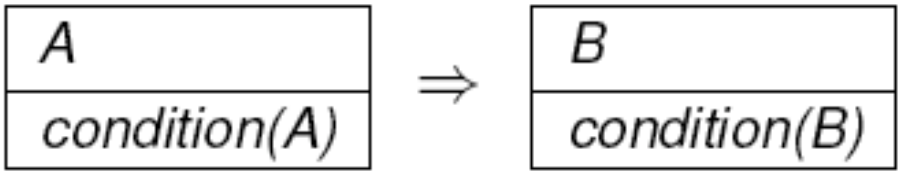
- complex condition: negation
- Prolog notation: - drs([A, B], [condition(A), condition(B)])
- pretty-printed



- similarly represented: negation as failure (~), possibility (<>), necessity ([])

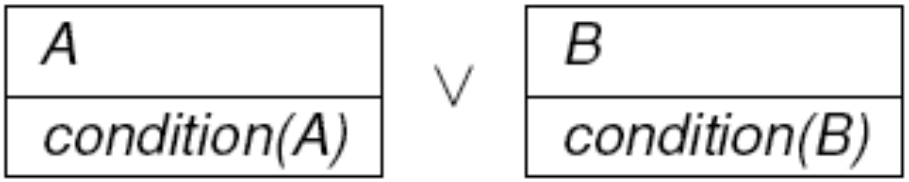
Extended Discourse Representation Structures

- complex condition: implication
- Prolog notation: drs([A], [condition(A)] => drs([B], [condition(B)]
- pretty-printed



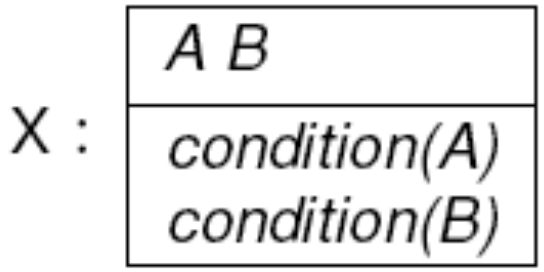
Extended Discourse Representation Structures

- complex condition: disjunction
- Prolog notation: drs([A], [condition(A)] v drs([B], [condition(B)]
- pretty-printed



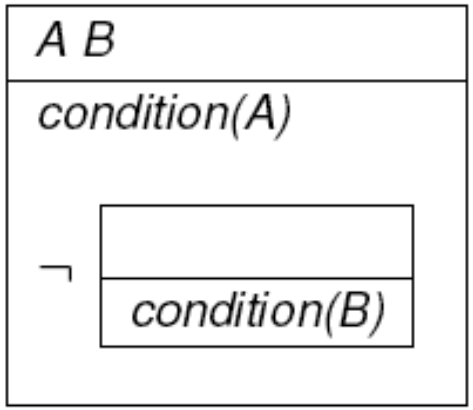
Extended Discourse Representation Structures

- complex condition: sentence subordination
- Prolog notation: X:drs([A,B], [condition(A), condition(B)]
- pretty-printed



Extended Discourse Representation Structures

- nesting of conditions
- Prolog notation: drs([A,B], [condition(A), - drs([], [condition(B)])])
- pretty-printed



Properties of Extended DRS

Only Predefined Relation Symbols

Every company that buys at least 2 standard machines gets a discount.

```
[]
   [A, B, C]
   object(A, atomic, company, object, cardinality, count_unit, eq, 1)-1
   object(B, group, machine, object, cardinality, count_unit, geq, 2)-1
   property(B, standard)-1
   predicate(C, event, buy, A, B)-1
   =>
   [D, E]
   object(D, atomic, discount, object, cardinality, count_unit, eq, 1)-1
   predicate(E, event, get, A, D)-1
```

Properties of Extended DRS

"Predicates" as Arguments

Every company that buys at least 2 standard machines gets a discount.

```
[]
   [A, B, C]
   object(A, atomic, company, object, cardinality, count_unit, eq, 1)-1
   object(B, group, machine, object, cardinality, count_unit, geq, 2)-1
   property(B, standard)-1
   predicate(C, event, buy, A, B)-1
   =>
   [D, E]
   object(D, atomic, discount, object, cardinality, count_unit, eq, 1)-1
   predicate(E, event, get, A, D)-1
```

Properties of Extended DRS

Lattice-Theoretic Typing of Objects

Every company that buys at least 2 standard machines gets a discount.

```
[]
   [A, B, C]
   object(A, atomic, company, object, cardinality, count_unit, eq, 1)-1
   object(B, group, machine, object, cardinality, count_unit, geq, 2)-1
   property(B, standard)-1
   predicate(C, event, buy, A, B)-1
   =>
   [D, E]
   object(D, atomic, discount, object, cardinality, count_unit, eq, 1)-1
   predicate(E, event, get, A, D)-1
```

Properties of Extended DRS

Simple Type System

Every company that buys at least 2 standard machines gets a discount.

```
[]
   [A, B, C]
   object(A, atomic, company, object, cardinality, count_unit, eq, 1)-1
   object(B, group, machine, object, cardinality, count_unit, geq, 2)-1
   property(B, standard)-1
   predicate(C, event, buy, A, B)-1
   =>
   [D, E]
   object(D, atomic, discount, object, cardinality, count_unit, eq, 1)-1
   predicate(E, event, get, A, D)-1
```

Properties of Extended DRS

Quantity Information

Every company that buys at least 2 standard machines gets a discount.

```
[]
   [A, B, C]
   object(A, atomic, company, object, cardinality, count_unit, eq, 1)-1
   object(B, group, machine, object, cardinality, count_unit, geq, 2)-1
   property(B, standard)-1
   predicate(C, event, buy, A, B)-1
   =>
   [D, E]
   object(D, atomic, discount, object , cardinality, count_unit, eq, 1)-1
   predicate(E, event, get, A, D)-1
```

Properties of Extended DRS

Eventuality Types

Every company that buys at least 2 standard machines gets a discount.

```
[]
   [A, B, C]
   object(A, atomic, company, object, cardinality, count_unit, eq, 1)-1
   object(B, atomic, machine, object, cardinality, count_unit, geq,2)-1
   property(B, standard)-1
   predicate(C, event, buy, A, B)-1
   =>
   [D, E]
   object(D, atomic, discount, object, cardinality, count_unit, eq, 1)-1
   predicate(E, event, get, A, D)-1
```

Properties of Extended DRS

Indices for Tracking in RACE

Every company that buys at least 2 standard machines gets a discount.

```
[]
   [A, B, C]
   object(A, atomic, company, object, cardinality, count_unit, eq, 1)-1
   object(B, atomic, machine, object, cardinality, count_unit, geq, 2)-1
   property(B, standard)-1
   predicate(C, event, buy, A, B)-1
   =>
   [D, E]
   object(D, atomic, discount, object, cardinality, count_unit, eq, 1)-1
   predicate(E, event, get, A, D)-1
```

Evaluation of Extended DRS Representation

- reification
  - plurals, eventualities, properties in first-order
  - quantification over "predicates" allows us to axiomatise, for instance, linguistic knowledge
- first-order representation
  - eases automated deduction and reusability
  - off-the-shelf tools
  - optional: translation into other first-order languages, e.g. standard or clausal form of first-order logic

From Discourse Representation Structures to Standard First-Order Logic

1. $(w, \langle\{\mathrm{x}_1 \ldots \mathrm{x}_n\}, \{\gamma_1 \ldots \gamma_m\}\rangle)^{fo} \stackrel{\text{def}}{=} \exists \mathrm{x}_1 \cdots \exists \mathrm{x}_n((w, \gamma_1)^{fo} \wedge \cdots \wedge (w, \gamma_m)^{fo})$;
2. $(w, R(\mathrm{x}_1, \ldots, \mathrm{x}_n))^{fo} \stackrel{\text{def}}{=} R(w, \mathrm{x}_1, \ldots, \mathrm{x}_n)$;
3. $(w, x_1 = x_2)^{fo} \stackrel{\text{def}}{=} x_1 = x_2$;
4. $(w, \neg B)^{fo} \stackrel{\text{def}}{=} \neg(w, B)^{fo}$;
5. $(w, B_1 \vee B_2)^{fo} \stackrel{\text{def}}{=} (w, B_1)^{fo} \vee (w, B_2)^{fo}$;
6. $(w, \langle\{\mathrm{x}_1 \ldots \mathrm{x}_n\}, \{\gamma_1 \ldots \gamma_m\}\rangle \Rightarrow B)^{fo} \stackrel{\text{def}}{=}$
   $\forall \mathrm{x}_1 \cdots \forall \mathrm{x}_n(((w, \gamma_1)^{fo} \wedge \cdots \wedge (w, \gamma_m)^{fo}) \rightarrow (w, B)^{fo})$;
7. $(w, \Diamond B)^{fo} \stackrel{\text{def}}{=} \exists v(R(w, v) \wedge (v, B)^{fo})$;
8. $(w, \Box B)^{fo} \stackrel{\text{def}}{=} \forall v(R(w, v) \rightarrow (v, B)^{fo})$;
9. $(w, v : B)^{fo} \stackrel{\text{def}}{=} (R(w, v) \wedge (v, B)^{fo})$.

From Discourse Representation Structures to Standard First-Order Logic

- ACE

  Every man sleeps.

- DRS

```
drs([], [drs([A], [object(A, atomic, man,
person, cardinality, count_unit, eq, 1)-1])
=> drs([B], [predicate(B, unspecified, sleep,
A)-1])])
```

- FOL

```
forall(A, object(A, atomic, man, person,
cardinality, count_unit, eq, 1)-1 => exists(B,
predicate(B, unspecified, sleep, A)-1))
```

Semantics of an ACE Text

- ACE text T → DRS D → FOL formula F
- FOL formula F has a formal semantics
- we assign D and also T the formal semantics of F
- Which semantics is this?

Model-Theoretic Semantics

- model-theoretic semantics of FOL formula F
  - (mathematical) model M of the domain D: objects, relations
  - interpretation: mapping of elements of F to elements of M
  - F is true if it corresponds to a state of affairs of D
  - M satisfies F
- truth is a relation between F and the independently existing static domain D
- problems
  - because of reification of ACE elements (plurals, verbs, properties) domain must contains abstract elements
  - definitions of validity and of logical consequences are in terms of *all* models

Herbrand Semantics

- Herbrand model-theoretic semantics of FOL formula F
  - interpretation: mapping of elements of F to themselves, predicates to true or false
  - Herbrand base B: set of all variable-free atoms of the language of F
  - interpretations are subsets of B
  - Herbrand model M is built bottom-up by the immediate consequence-operator
  - B and thus M are finite if language of F does not contain functions
  - smallest Herbrand model M; not necessarily unique
- if F has a Herbrand model then F has a model

Proof-Theoretic Semantics

- proof-theoretic semantics
  - no interpretation with respect to a real world
  - focus on state of affairs as described, ie. constructed, by the text
  - (declarative) sentences are considered to describe a true state of affairs
  - formal representation of the text in some logic language generates a knowledge base
  - entailments are done with the help of the knowledge base

Proof-Theoretic Semantics

- proof-theoretic semantics
  - meaning of a sentence is the set of inferences that can be drawn from a logical representation of that sentence with respect to the knowledge base
  - allows us to deal not only with declarative sentences, but also with questions and with commands that cannot be assigned a truth value

Proof-Theoretic Semantics

- proof-theoretic semantics requires
  - adequate formal representation of the intended meaning of text
  - adequate granularity to represent the intended level of detail of meaning
  - often reification
  - correct and complete reasoning tools (model-builders, theorem-provers)
- if requirements are fulfilled
  - instead of the logical representation we can deal directly with the natural language texts and sentences
  - concerning interpretation there is a cooperation between human and machine

(Non-) Entailments

- given an ACE text T and an additional ACE sentence S
  there are 4 possibilities
- T entails S
  - S can be deduced from T
  - usually shown by the inconsistency of $T \cup \neg S$
  - if S is a question then S can be answered on the basis of T
  - if S should be added to T then we need not bother
  - S is redundant, is not informative

(Non-) Entailments

- T entails $\neg$ S
  - T $\cup$ S is inconsistent
  - if S should be added to T then we must decide to delete the conflicting part of T or not to add S
- T can be split into T1 $\cup$ T2 and T1 $\cup$ S entails T2
  - elimination of redundancy or optimisation possible
  - computationally problematic

(Non-) Entailments

- T and S are not related by entailment
  - neither S nor ¬S follow from T
  - question S cannot be answered
  - if S should be added to T we can safely do so

Practical Entailment

- Attempto Reasoner RACE performs deductions on ACE texts
  - RACE shows that one ACE text is the logical consequence of another one
  - RACE answers ACE queries on the basis of an ACE text
  - RACE proves that an ACE text is (in-) consistent
- RACE provides a proof justification in ACE
- RACE finds all proofs
- RACE uses auxiliary first-order axioms and Prolog predicates to reason about plurals, natural numbers, equality