I2-D5

Verbalising Formal Languages in Attempto Controlled English I

Project title:	Reasoning on the Web with Rules and Semantics
Project acronym:	REWERSE
Project number:	IST-2004-506779
Project instrument:	EU FP6 Network of Excellence (NoE)
Project thematic priority:	Priority 2: Information Society Technologies (IST)
Document type:	D (deliverable)
Nature of document:	R/P (report and prototype)
Dissemination level:	PU (public)
Document number:	IST506779/Zurich/I2D5/D/PU
Responsible editor:	Norbert E. Fuchs
Reviewers:	Piero Bonatti, Gerd Wagner
Contributing participants:	University of Zurich
Contributing workpackages:	I2
Contractual date of delivery:	August 31, 2005
Actual date of delivery:	October 7, 2005

Abstract

The Attempto Parsing Engine (APE) translates a text in Attempto Controlled English (ACE) into a discourse representation structure (DRS). In this report we describe how this translation can be reversed, i.e. a DRS is translated back – verbalised – into a subset of ACE called Core ACE. The verbalisation of DRSs is used for two purposes. First, the DRS that APE generates from an ACE text is verbalised in Core ACE, providing a paraphrase of the original ACE text as feedback for the user. Second, we outline how to verbalise first-order logic expressions in Core ACE using the DRS as interlingua.

Keyword List

Attempto Controlled English, ACE, Core ACE, controlled natural language, discourse representation structure, DRS, discourse representation theory, DRT, parser, Attempto Parsing Engine, APE, natural language generation, NLG, paraphrase, verbalisation, DRACE

Project co-founded by the European Commission and the Swiss State Secretariat for Education and Research within the Sixth Framework Programme

Verbalising Formal Languages in Attempto Controlled English I

Norbert E. Fuchs, Kaarel Kaljurand, Gerold Schneider

Department of Informatics

&

Institute of Computational Linguistics

University of Zurich

Email: {fuchs, kalju, gschneid}@ifi.unizh.ch

October 7, 2005

Abstract

The Attempto Parsing Engine (APE) translates a text in Attempto Controlled English (ACE) into a discourse representation structure (DRS). In this report we describe how this translation can be reversed, i.e. a DRS is translated back – verbalised – into a subset of ACE called Core ACE. The verbalisation of DRSs is used for two purposes. First, the DRS that APE generates from an ACE text is verbalised in Core ACE, providing a paraphrase of the original ACE text as feedback for the user. Second, we outline how to verbalise first-order logic expressions in Core ACE using the DRS as interlingua.

Keyword List

Attempto Controlled English, ACE, Core ACE, controlled natural language, discourse representation structure, DRS, discourse representation theory, DRT, parser, Attempto Parsing Engine, APE, natural language generation, NLG, paraphrase, verbalisation, DRACE

Contents

1. Introduction

Many researchers seem to believe that semantic web languages – RDF, OWL etc. – are developed by and for specialists to be ultimately processed by computers. Tim Berners-Lee states this point of view explicitly as follows

> *The concept of machine-understandable documents does not imply some magical artificial intelligence which allows machines to comprehend human mumblings. It only indicates a machine's ability to solve a well-defined problem by performing well-defined operations on existing well-defined data. Instead of asking machines to understand people's language, it involves asking people to make the extra effort. [Berners-Lee 1998]*

The somewhat condescending attitude expressed by this statement completely disregards the needs of the uninitiated, for instance the needs of the average user of web-services. For them the formal notations proposed for the semantic web are hard or not at all understandable, and they would certainly prefer to be able to use "people's language".

A completely different position is taken by John Sowa who writes

> *If I had to process web annotations in any artificial language, I would prefer to use controlled English rather than special notations such as RDF. There is no reason why web annotations have to be humanly unreadable in order to be easy to process by a computer. [Sowa 2003]*

Arguably, there is a demand to make formal notations developed for the semantic web accessible in notations that are readily understood, for instance graphics or natural language, or even to replace incomprehensible notations by comprehensible ones. Actually, there are already a few proposals addressing this demand, e.g. [Metalog, Schwitter 2005a, Schwitter 2005b].

Following Sowa's lead we will here consider the verbalisation of formal notations in Attempto Controlled English (ACE) [Attempto].

ACE is a controlled natural language, namely a precisely defined subset of English that can be unambiguously translated into the language of first-order logic. ACE was specifically designed to be human and machine understandable, and thus serendipitously conforms to the mutually exclusive views of Berners-Lee and Sowa.

An ACE text is translated by the Attempto Parsing Engine (APE) into a discourse representation structure (DRS) [Kamp & Reyle 1993]. DRSs are logical expressions using a variant of the language of first-order logic.

DRSs form the centre of various translations and transformation as exhibited in the following *transformation diagram*.

$$\begin{tikzcd}
 & \text{FOL, ...} \arrow[d, shift left, Rightarrow, "\textcircled{5}"] & \\
\text{ACE} \arrow[r, Rightarrow, "\textcircled{1}"] & \textbf{DRS} \arrow[u, shift left, Rightarrow, "\textcircled{4}"] \arrow[r, Leftrightarrow, "\textcircled{3}\ \textcircled{2}"] \arrow[d, Rightarrow, "\textcircled{6}"] & \text{Core ACE} \\
 & \text{PQL, PRQ, FLUX, ...} &
\end{tikzcd}$$

The annotations have the following meaning:

① APE: translation of an ACE text into a DRS

② DRACE: verbalising a DRS as a text in Core ACE (cf. chapter 2)

③ APE: a text in Core ACE can be translated back into a DRS

④ DRS is translated into an expression in the language of first-order logic (FOL), or one of its variants

⑤ translation of a FOL expression into a DRS (cf. chapter 4)

⑥ DRS is translated into formal languages like PQL, PRQ, FLUX etc. ignoring some information

The translation of an ACE text into a DRS corresponds to number ① in the transformation diagram.

A DRS can be further transformed into expressions in formal languages equivalent to the language of first-order logic (cf. ④ in the transformation diagram). For example, translations into the standard and the clausal form of first-order logic are used within the Attempto Reasoner RACE [Fuchs & Schwertel 2003]. Transformations into languages equivalent to subsets of first-order logic – e.g. the languages PQL, PRQ, FLUX that were used in practical applications of ACE [Attempto] – usually ignore some information of the DRS (cf. ⑥ in the transformation diagram). Transformation of a DRS into languages like OWL or UML would also fall into this category.

As a contribution to make formal notations more readily accessible, we show that the translations ①, ④ and ⑥ can be inversed, concretely that expressions in formal languages equivalent to (subsets of) the language of first-order logic can be verbalised in ACE using the DRS as interlingua. This corresponds to the combination of the translations ⑤ (respectively ⑥) and ② in the transformation diagram.

Translations ⑤ and ⑥ depend heavily on the respective source language, require language-specific transformations, possibly the addition of extra material, and are in general the responsibility of the respective users. For these reasons, it is not possible to specify translations ⑤ and ⑥ beforehand. For translation ②, however, the source and target languages are known, and the relevant verbalisation program DRACE – quasi the inverse of APE – has been prototypically implemented by the Attempto group.

The verbalisation program DRACE translates DRSs deterministically into Core ACE – a subset of ACE that is semantically equivalent to full ACE, but offers only one of the possible syntactic variants to express the same semantics.

The prototype of DRACE has some additional temporary restrictions. For instance, DRSs containing plural constructs and DRSs of query sentences are not yet processed.

In this report, we present two examples of ACE verbalisation.

As a first example, we present the paraphrasing of ACE texts, i.e. we consider the special case that the formal language expression to be verbalised in ACE is itself an ACE text. This corresponds to the combination of translations ① and ② in the transformation diagram. While translation ① is performed by APE, translation ② uses the verbalisation program DRACE described in this report. Paraphrasing ACE texts forms a part of the functionality of the ACE 4 parser available on the demo page of the Attempto website [Attempto].

As a second example we consider the important case of verbalising first-order expressions, specifically expressions in the standard form of the language of first-order logic. This verbalisation corresponds to the combination of the translations ⑤ and ② in the transformation diagram. We will discuss the particular problems arising in translation ⑤, offer some solutions for these problems, but will not yet present a complete solution.

2. Verbalising a DRS in Core ACE

2.1. DRACE

The verbalisation program DRACE translates a DRS (source language) into a text in a syntactically reduced but semantically complete subset of ACE called Core ACE (target language). A prototype of DRACE has been developed by the Attempto group, and is used to paraphrase ACE texts (cf. chapter 3). Paraphrasing ACE texts forms a part of the functionality of the ACE 4 parser available on the demo page of the Attempto website [Attempto].

2.2. Source Language of DRACE

The source language of the verbalisation program DRACE is the full language of DRSs as described in [DRS Report 2005].

2.3. Requirements on the Target Language of DRACE

The target language of the verbalisation program DRACE is subject to three sets of requirements.

Requirement 1: Deterministic Verbalisation

First, it is important to understand that the target language of DRACE cannot be full ACE as defined in [ACE Language Manual 2005], but must necessarily be a subset of ACE. This can be seen in the following way.

Semantically equivalent, but syntactically different ACE texts are translated by APE into the same DRS. For instance the two texts

```
(1) A man who owns a card enters it.
```

and

```
(2) A man owns a card. The man enters it.
```

result in the same DRS

```
[A,B,C,D,E,F]
object(A,man,person)
quantity(A,cardinality,count_unit,B,eq,1)
structure(A,atomic)
object(C,card,object)
quantity(C,cardinality,count_unit,D,eq,1)
structure(C,atomic)
predicate(E,unspecified,own,A,C)
predicate(F,unspecified,enter,A,C)
```

To be a useful and reliable tool, the verbalisation of a DRS should always produce the same result. In other words, we require the verbalisation to be deterministic.

Thus one of the texts (1) or (2) must be chosen as verbalisation of our example DRS. DRACE generates

```
A man A owns a card C. The man A enters the card C.
```

that is similar to (2).

Requirement 2: Completeness of Verbalisation

Second, DRACE's target language must be a subset ACE that is sufficiently rich so that any DRS can be verbalised.

Requirement 3: Support of Paraphrasing

Third, since verbalisation will be used to paraphrase ACE texts. the target language of DRACE must contain additional elements that unravel complex ACE constructs, clarify their syntax and indicate how the original ACE text was interpreted by APE. For details and motivating arguments regarding these additional language elements see chapter 3.

As a consequence of these three requirements, the target language of DRACE is defined as a subset of ACE that is semantically equivalent to full ACE [ACE Language Manual 2005], but does not contain syntactic variants. Note, that this requirement does not mean that the target language is any sense “minimal”. Particularly requirement 3 – support of paraphrasing – enlarges it.

The thus defined subset of ACE is called Core ACE.

2.4. Syntactic Restrictions of Core ACE

Core ACE has the following syntactic restrictions with respect to full ACE:

- simple sentences have only the following components

  `Determiner (Adjectives) Noun (Apposition) (of-construct) Verb (Adverbs) (Prepositional Phrases).`

  in the given order; parentheses indicate optional components
- there are no relative clauses
- anaphoric references are expressed by definite noun phrases plus variables with the sole exception of indefinite pronouns (`somebody`, `something`) which are referred to by personal pronouns (`he/she`, `it`)
- quantifiers (`every`, `for every`, `all`, `no` etc.) are expressed by `if-then` sentences
- noun phrase negation (`no`) and verb phrase negation (`does not`, `is not`) are expressed by sentence negation (`it is not the case that`)
- Saxon genitive (`John's`) and possessive pronouns (`his`, `her own`) are expressed by `of-`constructs
- complex sentences have only the forms
    - `If` Sentence1 `then` Sentence2
    - Sentence1 `and` Sentence2
    - Sentence1`,and` Sentence2
    - Sentence1 `or` Sentence2
    - `It is not the case that` Sentence
- conjunction within a sentence is expressed by `and`; conjunctions of complete sentences are expressed by the individual sentences separated by periods

The prototype of DRACE adds the following temporary restrictions:

- there are no plurals
- there are no questions

Experience will show whether the syntactic restrictions need to be modified, and whether language elements need to be added to enhance the expressivity of Core ACE without violating the determinacy of the verbalisation.

2.5. Formal Syntax of Core ACE

Core ACE is defined by the following formal syntax where `Adverb`, `Adjective`, `Variable`, `QuotedString`, `Noun`, `Verb`, `Preposition`, `ProperName` are the same word classes as in full ACE.

```
Sentence = "If" Sentence "then" Sentence
Sentence = Sentence "and" Sentence
Sentence = Sentence ",and" Sentence
Sentence = Sentence "or" Sentence
Sentence = "It is not the case that" Sentence
Sentence = NP VP (Adverbs) (PPs)
Sentence = "There is" NP
```

```
VP = Verb
VP = Verb NP
VP = Verb NP NP

NP = "a" bareNP
NP = "the" bareNP
NP = ProperName
bareNP = (AdjCoord) Noun (Apposition) (ofConstr)

PPs = PP ... PP
PP = Preposition NP
ofConstr = "of" NP

Adverbs = Adverb "and" ... "and" Adverb
AdjCoord = Adjective "and" ... "and" Adjective
Apposition = Variable
Apposition = QuotedString
```

2.6. Principles Underlying the Workings of DRACE

A DRS is a tree of nested boxes containing discourse referents and conditions for these discourse referents. Each box has a unique ID and a label. The labels of the boxes are

- (empty)
- `if`
- `then`
- `or`
- `not`

Each box defines one or more discourse referents, and each referent is defined in exactly one box. Each referent has a unique ID and a set of conditions affiliated with it.

The depth-first traversal of a DRS tree defines an ordering of its boxes, and consequently an ordering of its referents. The order of the referents that occur as first arguments in DRS conditions determines the order in which DRACE verbalises the DRS. Thus we loosely say that DRACE verbalises discourse referents.

DRACE verbalises each discourse referent taking into consideration

- the location of the referent in the nested box structure
- whether the referent has already been verbalised before
- whether some other referent of the same box has already been verbalised

It is important to realise that DRACE verbalises a DRS directly without modifying or simplifying it first. In certain cases this can lead to inflated verbalisations. For instance, if the DRS contains a double negation, then the verbalisation will have double sentence negation.

The workings of DRACE are best understood with the help of the example verbalisation shown in the next section. The following sections will then elaborate on certain technical aspects.

2.7. Example Verbalisation

As example we use the DRS

```
[]
  [A, B]
  object(A, man, person)
  quantity(A, cardinality, count_unit, B, eq, 1)
  structure(A, atomic)
```

```
      =>
      [C]
      predicate(C, unspecified, run, A)
```

that stands for the ACE text

```
Every man runs.
```

The DRS defines the three discourse referents `A`, `B`, `C`, of which `A` and `C` occur as first arguments in the following conditions

```
A: object(A, man, person), quantity(A, cardinality, count_unit, B, eq, 1),
structure(A, atomic)
C: predicate(C, unspecified, run, A)
```

The verbalisation begins with the discourse referent `A`. Since `A` occurs in an `if`-box, and no other referent of this box has been verbalised yet, we start the ACE fragment with the box label as prefix (cf. section 2.8).

```
if ...
```

Next, since referent `A` is not "called" from the box in which it occurs (cf. section 2.10), we add a `there is` marker.

```
if there is ...
```

To verbalise the discourse referent `A` itself we first map its three conditions via a predefined schema (cf. section 2.9) into the ACE fragment `man`

```
if there is ... man ...
```

then – since `A` has not been verbalised before – we prefix the ACE fragment `man` with the indefinite determiner `a`.

```
if there is a man ...
```

This concludes the verbalisation of `A`.

Since `B` has no conditions we skip it.

Next we verbalise `C`. We simply concatenate its verbalisation to the ACE fragment that we have generated so far. `C` occurs in a `then`-box which leads to a `then` prefix

```
if there is a man then ...
```

`C` calls `A`. Since `A` has already been verbalised we now have to use the definite determiner `the`.

```
if there is a man then the man ...
```

Next we verbalise the `predicate` condition itself.

```
if there is a man then the man runs ...
```

Finally, we add a period as a sentence border marker.

```
if there is a man then the man runs.
```

This essentially completes the verbalisation of the example DRS. The only operation that remains is the capitalisation at the sentence beginning.

2.8. Prefixes of ACE Fragments

If a discourse referent is the first one from the given DRS box to be verbalised, then we prefix the ACE fragment with the respective box label (cf. section 2.6). If a referent from a box has already been verbalised, then we prefix the following ACE fragment with the conjunction marker `and`. In the top DRS box, we use two sentences separated by a period instead of the conjunction marker `and`.

2.9. Mapping DRS Conditions into ACE Fragments

There is a small set of DRS conditions that can be affiliated with each discourse referent. We essentially map those DRS conditions into natural language fragments with the help of predefined schemata. The following specifies the mapping for the cases of noun phrases and intransitive verbs.

The remaining cases are processed in a similar way.

`Referent` which corresponds to a noun phrase has the following conditions in the DRS:

```
property(Referent, Property1)
property(Referent, Property2)
...
property(Referent, PropertyN)
object(Referent, Noun, _)
quantity(Referent, _, _, _, _, _)
structure(Referent, _)
relation(Referent, Noun, _, Owner)
```

These conditions are mapped into:

```
Property1 and ... and PropertyN Noun of p(Owner)
```

where `p` is the processing function that is called recursively. Note that we ignore most of the information in the `quantity`-condition, since we do not currently support plural noun phrases.

The result of this mapping could be the ACE fragment

```
a big and ugly and hungry dog X of a man Y
```

where `p(Owner)` has been resolved as `a man Y.`

`Referent` which corresponds to an intransitive verb has the following conditions in the DRS:

```
predicate(Referent, _, Verb, Argument)
modifier(Referent, _, none, AdverbialModifier1)
modifier(Referent, _, none, AdverbialModifier2)
...
modifier(Referent, _, none, AdverbialModifierN)
modifier(Referent, _, Preposition1, PPModifier1)
modifier(Referent, _, Preposition2, PPModifier2)
...
modifier(Referent, _, PrepositionN, PPModifierN)
```

These conditions are mapped into:

```
p(Argument) singular(Verb) AdverbialModifier1 and ... and
AdverbialModifierN Preposition1 p(PPModifier1) Preposition2 p(PPModifier2)
... PrepositionN p(PPModifierN)
```

where `p` is the processing function which is called recursively and `singular` is a function which generates present tense singular forms of verbs based on their lemma.

The result of this mapping could be the ACE fragment

```
a dog X runs quickly and silently in the park Y in the morning Z
```

where `p(Argument)` has been resolved as `a dog X.`

2.10. Handling of Anaphors

Whenever we process a discourse referent corresponding to a noun phrase for the first time, for instance

```
object(A, man, person), quantity(A, cardinality, count_unit, B, eq, 1),
structure(A, atomic)
```

we use the indefinite determiner `a` for the respective ACE fragment and add the discourse referent `A` as variable apposition to the ACE fragment

```
a man A
```

When we encounter the same referent again, we introduce the definite determiner `the` as anaphor marker and add the same variable `A` as apposition

```
the man A
```

Since indefinite pronouns (`somebody`, `something`) do not accept appositions, anaphoric references to them are expressed by personal pronouns (`he/she`, `it`).

2.11. Calling of Referents

Some referents "call" other referents. For instance, referent `G` calls the referents `A` and `F` in the condition

```
predicate(G, state, be, A, F)
```

While in the above example the chain of calls is finite, some calling chains can be unlimited as in the sequence of `relation` conditions

```
relation(Dog, Man)
relation(Man, Town)
relation(Town, Country)
...
```

or even infinite as in the loop of `relation` conditions

```
relation(Dog, Man)
relation(Man, Dog)
```

Currently, DRACE does not detect these loops.

Some referents cannot be called by any other referent. These are, for instance, the referents of `predicate` conditions – such as `G` above – but also object referents that were originally introduced by the ACE phrase `there is a`.

3. Paraphrasing an ACE Text

3.1. Paraphrases as Verbalisation

As a first application of the verbalisation program DRACE, we paraphrase an ACE text as a text in Core ACE using a DRS as interlingua. Paraphrasing corresponds to the combination of translations ① and ② in the transformation diagram.

As mentioned in section 2.3, paraphrasing imposes additional requirements on Core ACE.

3.2. What is a paraphrase?

The point of paraphrasing an ACE `Text` as an ACE `Paraphrase` is that the following equalities hold:

```
APE(Text) = DRS
DRACE(DRS) = Paraphrase
APE(Paraphrase) = DRS
```

These equalities express that `Text` and `Paraphrase` are semantically equivalent – since they have the same DRS – even though they could be syntactically different. Furthermore, the equalities express that APE and DRACE are inverses of each other as far as semantics is concerned

```
APE(DRACE(DRS)) = DRS
DRACE(APE(Text)) = Paraphrase
```

However, these equalities only constrain, but do not determine the paraphrase. Further design decisions must be taken, some of which we will discuss in the next sections.

3.3. How Far Should a Paraphrase Deviate From the Original?

Arguably, the paraphrase of an ACE text should be syntactically different from the original ACE text, because paraphrases should help unravelling a perhaps complex ACE text to understand it better. The paraphrase should also reflect how the original ACE text was processed and accepted by APE. In brief, paraphrases are for easy reading and not for convenient writing. Nevertheless, a paraphrase should be correct ACE.

3.4. Motivating Some Design Decisions For Core ACE

The syntactic restrictions of Core ACE were already listed in sections 2.4 and 2.5. In the following sections we will discuss selected language constructs of Core ACE with regard to their suitability for paraphrasing.

3.4.1. Negation

To reduce the complexity of sentences and to indicate the correct scope of negations, Core ACE knows only sentence negation, no noun phrase negation or verb phrase negation.

Noun phrase negation as in

```
No man runs.
```

is paraphrased by sentence negation as

```
It is not the case that a man A runs.
```

while verb phrase negation as in

```
A man does not run.
```

is paraphrased by sentence negation as

```
There is a man A. It is not the case that the man A runs .
```

As mentioned above, currently double negation in a DRS leads to double sentence negation

```
It is not the case that it is not the case ...
```

in the verbalisation of the DRS.

3.4.2. Relative Clauses

Core ACE does not contain relative clauses. Some might argue that relative clauses like those occurring in

```
Every man who loves a woman who loves him is happy.
```

are important since they make the sentence concise and compact. Maybe, but this is not the point here. A paraphrase should serve to unravel complex and compact sentences and to reflect their interpretation by the parser. Thus unravelling the above sentence as

```
If a woman X loves a man Y and the man Y loves the woman X then the man Y
is happy.
```

certainly helps understanding it.

Note that unravelling the sentence introduces variables to indicate anaphoric references.

3.4.3. Anaphoric References

Core ACE makes heavy use of variables to unravel anaphoric references – expressed in ACE by personal pronouns and definite noun phrases. Though in Core ACE anaphoric references are already indicated by pairs of indefinite-definite determiners, variables serve to emphasise them. For instance, the paraphrase of the ACE text

```
A customer enters a card and he types a code. If it is not valid then
SimpleMat rejects the card.
```

is

```
A customer C enters a card E. The customer C types a code H. If it is not
the case that the code H is valid then SimpleMat rejects the card E.
```

This does not work, though, for anaphoric references to indefinite pronouns, like `somebody` or `something`. In this case, pronouns are used to express anaphoric references. Thus the text

```
John meets somebody. He is a friend of John.
```

gets the paraphrase

```
John meets somebody. He/she is a friend E of John.
```

Note that `somebody` does not carry any gender information and is thus referred to by `he/she`.

4. Verbalising First-Order Logic

4.1. The Task

Since first-order logic combines high expressivity with acceptable tractability, it plays an important role in knowledge representation. This role makes the language of first-order logic a prime candidate for verbalisation in ACE using DRSs as interlingua. Verbalisation of first-order expressions corresponds to the combination of the translations ⑤ and ② in the transformation diagram.

Once the verbalisation of first-order logic will be available, users who want to verbalise a formal notation could decide to use first-order logic instead of DRSs as interlingua.

In preceding chapters we showed how a DRS can be translated into ACE (translation ②). Now we will discuss the particular problems arising in the translation of a first-order expression into a DRS (translation ⑤), and offer some solutions for these problems. As a result translation ⑤ will turn out to consist of several individual steps that we will present here in a logical order that is not necessarily the order in which they will be implemented.

Note that we will not yet offer a complete solution, or even an implementation of translation ⑤.

4.2. Transforming First-Order Expressions

Though the language of first-order logic does have a simple syntax, the composition of logical structures, and the availability of logical identities leads to a plethora of first-order expressions that must be translated into a DRS. The standard answer to this problem are normal forms of logical expressions.

4.2.1. Prenex Normal Form

In a first step we convert the first-order expression by equivalence transformations into its prenex normal form, i.e. an expression of the form

```
Quantifiers(Matrix)
```

where `Quantifiers` are all universal and existential quantifiers of the original first-order expression and `Matrix` is a first-order expression that does not contain any quantifiers. Thus we have

```
first-order expression

⇒ normalisation

prenex normal form of first-order expression
```

as the first step of translation ⑤.

4.2.2. Completing the Prenex Normal Form

The logically next step would be to map the prenex normal form to a DRS. However, it can happen that the prenex normal form is an expression that cannot be mapped directly to any of the DRS boxes (cf. section 2.4), and that it needs to be appropriately completed beforehand.

Here is an example. Given the first-order expression

```
forall(X, p(X))
```

that expresses that all elements of a domain have the attribute `p`, we derive the identical prenex normal form. This, however, cannot be directly mapped to a DRS box. To be able to do so, we introduce the predicate `domain` expressing the membership in the domain and convert the expression into the implication

```
forall(X, domain(X) ⇒ p(X))
```

that can be mapped to the combination of an `if`-box and a `then`-box.

Note that the added predicate `domain`, and the modified expression do not introduce any new information.

Other cases of incomplete expressions will be treated similarly.

Now translation ⑤ looks as follows

```
first-order expression
⇒ normalisation
prenex normal form of first-order expression
⇒ completion
completed prenex normal form of first-order expression
```

4.2.3. From Prenex Normal Form to DRS

In the following step the completed prenex normal form of the first-order expression is transformed into a preliminary DRS

```
first-order expression
⇒ normalisation
prenex normal form of first-order expression
⇒ completion
completed prenex normal form of first-order expression
⇒ translation into preliminary DRS
preliminary DRS
```

The preliminary DRS has the correct structure of the final DRS, but does not yet contain the correct DRS conditions.

4.3. Transforming Logical Atoms

The above translation steps do not depend at all on the exact form of the logical atoms of the original first-order expression. Their transformation into correct DRS conditions is a separate problem that is described next.

4.3.1. Original Logical Atoms

There is a great freedom to state relations as logical atoms. Here we will consider the simple form that is often found in text books. For instance, to express that the objects `a` and `b` have the relation `r`, we write

```
r(a,b)
```

Using this notation the situation that all customers wait would be expressed as

```
forall(X, customer(X) ⇒ wait(X))
```

Many variants of the notation are possible. Note, however, that verbalisation in natural language restricts the arity of relations essentially to 3.

4.3.2. Word Classes and Types of Relations

Given the logical atom

```
surface(a)
```

how should we eventually verbalise the unary relation `surface`? As a noun (*a surface*), as a verb (*to surface*), or as an adjective (*surface material*)?

To be able to map logical atoms to DRS conditions – and ultimately to words in ACE – we need to know the "word classes" of the relations, and also their type [DRS Report 2005]. This lexical information must be provided in addition to the first-order expression to be verbalised.

Here are some examples of associating word classes and types with relations.

Relation	**Word Class**	**Type**
customer/1	noun	person
wait/1	(intransitive) verb	event/state
see/2	(transitive) verb	event/state
give_to/3	(ditransitive) verb	event/state

`red/1`	`adjective`	-
`of/2`	`genitive relation`	-

Possibly the lexicons of APE could be used to provide this lexical information.

4.3.3. Transforming Logical Atoms into DRS Conditions

In section 2.9 we saw how DRS conditions can be schematically mapped into ACE fragments. Given the small number of word classes and types, and the fact that they are uniquely related to sets of DRS conditions, we suggest the same schematic approach to map logical atoms into DRS conditions.

For instance, given the logical atom

```
customer(A)
```

in which the relation `customer/1` has the word class `noun` with type `person`, we schematically replace the logical atom by the three DRS conditions

```
object(A,customer,person), quantity(A,cardinality,count_unit,B,eq,1),
structure(A,atomic)
```

Notice, that we need to introduce the additional variable `B` that is currently not used any further.

Similarly, we replace the logical atom

```
see(X,Y)
```

in which the relation `see/2` has the word class `transitive verb` with type `event` by the DRS condition

```
predicate(P,event,see,X,Y)
```

for which we introduce the additional variable `P`.

The transformation of logical atoms into DRS conditions constitutes the last step of translation ⑤ that now looks like

```
first-order expression
⇒ normalisation
prenex normal form of first-order expression
⇒ completion
completed prenex normal form of first-order expression
⇒ translation into preliminary DRS
preliminary DRS
⇒ transformation of logical atoms
final DRS
```

We would like to emphasise again that the above describes the logical sequence of transformation steps, not necessarily the order in which they will be ultimately implemented.

5. Conclusions

For many people the formal notations introduced for the semantic web are hard or not at all comprehensible. We claim that there is a definite need to make these notations accessible in readily understandable notations, or perhaps even to replace them altogether by understandable, but still computer-processable, notations.

Attempto Controlled English (ACE) – being both human and machine understandable – can fulfil this need.

In this report we show how formal notations can be verbalised in ACE. To this end we have defined Core ACE – a subset of ACE that is semantically equivalent to full ACE, but does not offer all its syntactical variants. Furthermore, we have developed a prototype of the program DRACE that verbalises the first-order logic language of discourse representation structures (DRS) [DRS Report 2005] in Core ACE.

As a first concrete application, DRACE is used to paraphrase ACE texts that were previously translated by the Attempto Parsing Engine (APE) into DRSs. This paraphrase serves as feedback to the user, for instance in the demo version of APE available on the Attempto website [Attempto].

Additionally, we have outlined the verbalisation of expressions in the standard form of the language of first-order logic as a two step translation using DRSs as interlingua: first-order expressions are translated into DRSs which are than translated by DRACE into Core ACE.

This is the first of two deliverables on verbalisation of formal languages in Attempto Controlled English. Further work on

- extending Core ACE by plurals and queries
- enhancing and completing DRACE
- implementing the translation of first-order expressions into DRSs
- verbalising a concrete language of the semantic web

will be presented in a second deliverable.

6. References

[Attempto] www.ifi.unizh.ch/attempto: Attempto website containing a description of the Attempto project, a list of the people involved, demos, publications and various other information

[ACE Language Manual 2005] N. E. Fuchs, S. Höfler, K. Kaljurand, F. Rinaldi, G. Schneider, U. Schwertel, Attempto Controlled English (ACE), Language Manual, Version 4.0, Technical Report 2005, forthcoming, [www.ifi.unizh.ch/attempto]

[Berners-Lee 1998] Tim Berners-Lee, www.w3.org/DesignIssues/RDFnot.html, September 17, 1998.

[DRS Report 2005] N. E. Fuchs, S. Höfler, G. Schneider, U. Schwertel, Extended Discourse Representation Structures in Attempto Controlled English, Technical Report 2005, forthcoming, [www.ifi.unizh.ch/attempto]

[Fuchs & Schwertel 2003] N. E. Fuchs, U. Schwertel, Reasoning in Attempto Controlled English, in: F. Bry, N. Henze and J. Maluszynski (eds.): Principles and Practice of Semantic Web Reasoning, International Workshop PPSWR 2003, Mumbai, India, December 2003. Lecture Notes in Computer Science 2901, Springer Verlag, 2003

[Kamp & Reyle 1993] H. Kamp & U. Reyle, From Discourse to Logic, Introduction to Modeltheoretic Semantics of Natural Language, Formal Logic and Discourse Representation Theory, Kluwer, 1993

[Metalog] Metalog Project, www.w3.org/RDF/Metalog/docs/pnl.html

[Schwitter 2005a] R. Schwitter, Controlled Natural Language as Interface Language to the Semantic Web, to be presented at the 2nd Indian International Conference on Artificial Intelligence (IICAI-05), Pune, India, December 20-22, 2005

[Schwitter 2005b] R. Schwitter, A Controlled Natural Language Layer for the Semantic Web, to be presented at the 18th Australian Joint Conference on Artificial Intelligence, Sydney, Australia, December 5-9, 2005

[Sowa 2003] John F. Sowa, CG Mailing List, October 19, 2003