# RECON – A Controlled English for Business Rules

Ed Barkmeyer[1] and Fabian Neuhaus[1,2]

[1] National Institute of Standards and Technology, Gaithersburg, MD
[2] Prometheus Computing, Cullowhee, NC

**Abstract.** Capturing business rules in a formal logic representation supports the enterprise in two important ways: it enables the evaluation of logs and audit records for conformance to, or violation of, the rules; and it enables the conforming automation of some enterprise activities. The problem is that formal logic representations of the rules are very difficult for an industry expert to read and even more difficult to write, and translating the natural language of the enterprise to formal logic is an unsolved problem. RECON – Restricted English for Constructing Ontologies – is a subset of English that can be easily read by an industry expert, while having a formal grammar and an unambiguous translation to formal logic. This paper describes the principal features of the RECON language, with examples, and shows the corresponding formal logic constructs that are produced by the RECON tool.

## 1 Introduction

The Restricted English for Constructing Ontologies (RECON) language is a close relative of English that has a well-defined interpretation in a formal logic language. The need for RECON arose in a project concerned with the consistency, completeness, and timeliness of information received from supply-chain business partners via electronic messages. To automate the validation of the information, the facts, rules and definitions of terms must be stated in a form suitable for machine reasoning – a formal logic language. On the other hand, capturing definitions, facts, and rules for an industrial domain requires the contributions of experts in the domain. Formal logic languages are very difficult for an industry expert to understand. So, to facilitate capturing the knowledge of the industry experts, the project developed an intermediate language – a "restricted English" called RECON.

RECON looks like English, but is carefully restricted in grammar, so that every statement and most definitions have an unambiguous equivalent in the formal logic language. The experts in the industry domain may require the assistance of a knowledge engineer to state their intent in the RECON language, but it is most important that they can read the restricted English formulation and verify that it captures their intent. The translation of the stated definitions, facts, and rules is used directly in validating incoming information. For example, the following is a business rule expressed in RECON:

**Example 1** Any shipment that consists of more than 1000 gallons of gasoline must be shipped via some registered tanker.

The formal logic version of this rule, as output from the RECON translator, is used directly by the validator in determining that messages describing shipments are (or are not) consistent with the rule.

Properly, RECON is only the grammar for the language. The vocabulary of the language – the nouns, verbs and adjectives – is defined by the industry experts. The RECON tooling is designed to capture a vocabulary based on English words and to parse sentences in the language that use the terms in that vocabulary. The sentences may be definitions of terms, facts about the domain, or rules. The parsed sentences are then translated into a formal logic text in the IKRIS Knowledge Language (IKL) [5], which is an extension of ISO Common Logic Interchange Format [6] that supports nominalized propositions.

The existing RECON tool is the engine that processes the vocabulary and translates the sentences. It was designed as a plug-in for an authoring tool that does not yet exist. It is currently run via a simple command-line interface program that invokes the RECON tool to process a set of files containing vocabularies, facts and rulesets.

This paper describes RECON's capabilities, with a particular focus on representing business rules. We will do that by discussing various RECON examples and their translation into IKL. The grammar of the RECON language is formally specified in [1].

## 2 Related work

It is important to distinguish restricted English from natural language processing. The objective of natural language processing is to produce a formal interpretation of text as published. Natural language can be ambiguous, and formal interpretations are not necessarily reliable. The objective of a "controlled" English is to ensure that text written in the language can be converted in every case to a particular formal language by a particular algorithm. So we will here consider only restricted English languages.

The simplest restricted English languages are "template" languages, in which the knowledge engineer defines a set of sentence forms with parameter markers, and defines the interpretation as a pattern for text in the formal language, with slots for parameter substitutions. Sometimes called "domain-specific languages" (DSLs), these are supported by tooling such as XTEXT [15]. Several production rules technologies have languages of this kind.

The more interesting controlled natural languages have grammars and parsers that convert the intent of the statements to formal logic. The target logic language determines what a restricted English tool can export, and thus limits what the restricted English language can usefully express. First-order logic and its extensions are the most expressive, while Horn clauses, description logics and production rules are subsets of first-order logic that allow more efficient implementation.

Rabbit [2, 4], for example, is a controlled English language for the Web Ontology Language (OWL) [14], which is based on description logic. It permits only simple sentences and complex noun phrases that have clear renderings into OWL. Similarly, Common Logic Controlled English [13] is intended to express first-order logic propositions using explicit quantifiers and variables, and although it apparently allows more natural expressions, that part of the published grammar is incomplete. The Controlled English to Logic Translator (CELT) [10] exports formal statements in Knowledge Interchange format [9], a first-order logic language. Although its grammar is not published, CELT tooling accepts a more natural English, and interprets common terms using synonymies and the Suggested Upper Merged Ontology (SUMO) [7].

The Attempto project developed the Attempto Controlled English (ACE) [3] that can be rendered into formal languages for various computational purposes. ACE eliminates aspects of natural English that interfere with unambiguous interpretation, but like natural language parsers, it integrates multiple sentences in a text corpus, and supports back references, using "discourse representation" technology.

The Semantics of Business Vocabulary and Rules (SBVR) is a standard for capturing vocabularies, and introduces a Structured English for facts, definitions and rules that use those vocabularies [8]. The language, however, is not standardized; it is described informally in an annex. The parse relies on terms being marked up in the text. For example, "Mary goes to the store" is written "Mary *goes to* the store." SBVR tools must export an enhanced first-order logic language that includes proposition nominalization and modalities for possibility and obligation/permission.

Unlike natural language tooling, which depends on dictionaries for terms and their possible meanings, ACE, SBVR and RECON require terms to be declared and provide for formal definitions. For multi-word terms, however, ACE requires hyphenated terms and SBVR requires term markup, while RECON requires neither. RECON and SBVR provide for verb usage templates, resulting in n-ary logical relations with formal definitions, while ACE and CELT objectify most verbs as classes of states or events, whose logic model is an event object with a set of binary "role relations". This is a major difference in the resulting formal logic structures. Neither ACE nor SBVR supports compound noun phrases, which RECON does. SBVR Structured English is the most comprehensive of the above, and anything written in SBVR Structured English can be written in RECON, without special markups. Finally, like ACE and unlike SBVR, the RECON grammar is formally defined in [1]. This document also contains a more detailed comparison of RECON with related work.

## 3 Dictionary and Vocabulary

The most basic linguistic notion in RECON is the word form. Roughly speaking, English word forms are the strings in an English text that are delineated from the rest of the text by whitespace or punctuation; e.g., 'ACME', 'the', 'registered' are

word forms in Example 1. A RECON dictionary entry (a 'word') is a collection of word forms associated with a grammatical category (noun, verb, other) that are treated as representations of the same word. (Verb participles have special significance, not discussed in this paper.) For example, Dictionary 1 contains one word.

**Dictionary 1** *Dictionary Verb: run runs ran running run*

The same word form can belong to multiple dictionary entries. Note that RECON assigns no semantics to dictionary entries; they are purely syntactic.

A *vocabulary* consists of terminological entries. A terminological entry consists of one or more declarations. A terminological entry always begins with the declaration of a term for the vocabulary item, called the primary term. A term is a sequence of words, each of which will be recognized in any word form. The primary term declaration may be followed by alternative forms and definitions, and perhaps other declarations, that are part of the terminology entry and are associated with the primary term. A formal definition in the RECON language will cause RECON to produce a formal definition (of the corresponding IKL term) in IKL. (But we will not exemplify that.)

**Vocabulary 1**

*Name:* ACME Inc
*Name:* Bride of Neptune
*Type Noun:* supplier
*Type Noun:* party
*Type Noun:* shipment
*Type Noun:* vessel
*Mass Noun:* gasoline
*Adjective:* (thing) is registered
*Property:* (party) is the customer () of (shipment)
*Verb:* (party) ships (shipment)
    *Alternative:* (shipment) is shipped by (party)
*Verb:* (shipment) is shipped via (vessel)
*Property:* (quantity) is the volume () of (thing)
*Unit:* gallon: volume

Vocabulary 1 contains declarations of six different kinds of terms: names, type nouns, a mass noun, an adjective, two properties, and verbs. Adjectives and properties are treated as verbs that also have other syntactic usages. Note that adjectives, properties, and verbs take arguments called 'roles'. In the primary entry, the position and the type of a role is indicated by a noun term enclosed in parentheses. For example, the first argument of "ships" must be a party, the second must be a shipment. In the declaration of an alternative form (e.g., 'is shipped by') the same roles can appear in different positions. The two forms are different syntactic forms for the same verb concept with the same roles.

## 4 Starting simple

In first-order logic, an atomic sentence consists of a predicate and a number of arguments. The analog in RECON is a sentence that consists of a verb phrase that is a verb form from the vocabulary, where the arguments are replaced by names.

**Example 2** ACME Inc is registered.
ACME Inc ships SH12345.

**Output 2**
```
(thing.is_registered ACME_Inc)
(party.ships.shipment ACME_Inc SH12345)
```

These examples illustrate how simple RECON sentences are translated into simple IKL formulas. Note that RECON will recognize an alternative form and convert it to the primary form. For example, sentence Example 3 will be translated into the same formula as the second sentence in Example 2.

**Example 3** SH12345 is shipped by ACME Inc.

In the following sections we will consider several ways in which these simple sentences can become more complex. First, the roles in the verb phrase can be filled by complex noun phrases. Second, simple sentences can be combined to form more complex sentences.

## 5 Type nouns

The most commonly used noun phrases consist of a quantifier (e.g., 'a', 'the', 'any', 'every', 'some') and a type noun, as in Example 4. As Output 4 illustrates, the result of the translation into first-order logic involves one (or more) quantifiers.

**Example 4** ACME Inc ships a shipment.

**Output 4**
```
(exists (?shipment1)
  (and
    (shipment ?shipment1)
    (party.ships.shipment ACME_Inc ?shipment1)))
```

In English quantified noun-phrases can lead to ambiguity. For example, in English both sentences in Example 5 have two theoretical readings: (1) for each supplier there is a shipment that is shipped (but different suppliers might ship different shipments), and (2) there is one shipment that is shipped by every supplier.

**Example 5** Every supplier ships some shipment.
Some shipment is shipped by every supplier.

A human reader uses contextual knowledge about the business practices of suppliers to disambiguate the sentences and to decide that (1) is likely the intended meaning. Since RECON has no such contextual knowledge, however, and because RECON has to provide an unambiguous parse, it must have a rule for deciding how each combination of quantifiers is interpreted. In this case, the rule is: The quantification for the subject of the main verb encloses any quantification for other verb roles. The rules for some constructs are more complex. As Output 5 shows, the two sentences in Example 5 are translated into different first-order logic formulas. The disadvantage of this is that the user must be careful, because the order of the quantified type nouns in RECON sentences influences the translation. The advantage is that the experienced author can express either intent, and know how each will be translated.

**Output 5**

```
(forall (?supplier1)
  (if
    (supplier ?supplier1)
    (exists (?shipment2)
      (and
        (shipment ?shipment2)
        (party.ships.shipment ?supplier1 ?shipment2)
      ))))
```

```
(exists (?shipment1)
  (and
    (shipment ?shipment1)
    (forall (?supplier2)
      (if
        (supplier ?supplier2)
        (party.ships.shipment ?supplier2 ?shipment1)
      ))))
```

## 6 Adjectives

As we have seen in Example 2, adjectives can be used within verb phrases. But, of course, their primary use in English is as noun modifiers. As illustrated by Example 6 the adjective (in this case 'registered') translates into a conjunct in Output 6.

**Example 6** Bride of Neptune is a registered tanker.

**Output 6**

```
(exists (?tanker1)
  (and
    (and
      (tanker ?tanker1)
      (thing.is_registered ?tanker1))
    (= Bride_of_Neptune ?tanker1)))
```

## 7 Qualifiers

Another way to modify noun phrases is with quantifiers, that is subordinate sentences starting with 'that', 'which', 'who', or 'whom'.

**Example 7** Any shipment that is shipped via Bride of Neptune is registered.

**Output 7**

```
(forall (?shipment1)
  (if
    (and
      (shipment ?shipment1)
      (shipment.is_shipped_via.vessel
                  ?shipment1 Bride_of_Neptune))
    (thing.is_registered ?shipment1)))
```

RECON allows for arbitrarily complex and nested qualifiers. However, for the same reasons that style guides discourage the use of overly complex qualifiers in English, we recommend some restraint on their use in RECON.

## 8 Properties

In Vocabulary 1 we declared two properties. Properties can be used in verb phrases, but they also give rise to possessive noun phrase structures, as illustrated in Example 8. Note that in Output 8 the second sentence is translated into a universally quantified formula. This is too weak, it would be more appropriate to use an $\iota$-operator as defined in Russell's theory of description [12].

**Example 8** ACME Inc is the customer of SH12345 .
The customer of SH12345 ships SH12345.

**Output 8**

```
(party.is_the_customer_of.shipment ACME_Inc SH12345)
(forall (?thing1)
  (if
    (and
      (thing ?thing1)
      (thing.is_the_customer_of.shipment
                          ?thing1 SH12345))
    (party.ships.shipment ?thing1 SH12345)))
```

## 9 Mass-nouns

The semantics of mass expressions is a well-known challenge in philosophy and linguistics [11]. RECON does provide some limited support for expressions involving mass nouns. Occurrences of mass nouns without any units of measurement (e.g., 'some gasoline') are translated into quantifications over discrete portions of stuff (e.g., portions of gasoline). Hence, mass nouns are translated by RECON into predicates that apply to countable entities (the portions). To illustrate the point, Example 9 is interpreted by RECON into Output 9: there is [*a portion of*] gasoline that the shipment SH12345 consists of.

**Example 9** SH12345 consists of some gasoline.

**Output 9**

```
(exists (?gasoline1)
  (and
    (gasoline ?gasoline1)
    (shipment.consists_of.thing SH12345 ?gasoline1)))
```

## 10 Measurements, collections, and quantities

RECON is able to handle expressions that involve quantities of things expressed in measurement units. It contains a rather complex model of units of measure, which supports derived SI units and their definitions; e.g., $1N = 1kg\frac{m}{s^2}$. The details are beyond the scope of this paper; however, we illustrate the approach with two examples. Example 10 is similar to Example 9, the difference is that the amount of gasoline is specified. The difference between Output 9 and Output 10 is an additional conjunct, which uses the quantity value '(Qvalue 1000 "gallon")' and the property *volume of*. It is used because the unit gallon was declared to measure volume. Quantity values are functional expressions consisting of the special function term 'Qvalue', a number, and a name that denotes a unit of measurement.

**Example 10** SH12345 consists of 1000 gallons of gasoline.

**Output 10**

```
(exists (?gasoline1)
  (and
    (and
      (gasoline ?gasoline1)
      (quantity.is_the_volume_of.thing
                (Qvalue 1000 "gallon") ?gasoline1))
    (shipment.consists_of.thing SH12345 ?gasoline1)))
```

**Example 11** SH12346 consists of 1000 widgets.

RECON uses a similar method to translate sentences that involve numerically quantified type nouns (as in Example 11) into quantifications over collections and their cardinality.

## 11 Connectives

RECON supports the use of compound sentences (Example 12) and compound noun phrases (Example 13).

**Example 12** ACME Inc is registered or ACME Inc is not registered.

**Output 12**

```
(or
  (thing.is_registered ACME_Inc)
  (not
    (thing.is_registered ACME_Inc)))
```

**Example 13** ACME Inc ships both SH12345 and SH12346.

**Output 13**

```
(and
  (party.ships.shipment ACME_Inc SH12345)
  (party.ships.shipment ACME_Inc SH12346))
```

Note that Output 13 consists of two conjuncts; thus, there is no connection between the two shipments. There is an alternative interpretation of the English sentence, according to which ACME ships a collection that consists of the two shipments. This can be expressed in the RECON grammar by adding the key word 'together' (see Example 14).

**Example 14** ACME Inc ships both SH12345 and SH12346 together.

## 12 Rules

All of the examples we have considered so far are similar to Example 15 below in the following sense: they are statements about how the world is. In contrast, Example 16 is about how the world should be; it expresses a requirement. The existence of an unregistered shipment would make Example 15 false; but it would have no affect on the truth of Example 16. It would, however, make the state of the world unacceptable.

**Example 15** Every shipment is registered.

**Example 16** Every shipment must be registered.

Since business rules can be statements about actual characteristics of the world as well as normative characteristics, RECON has been designed to capture the difference. As a comparison of Output 15 and Output 16 illustrates, the difference is represented in IKL with the help of a modal predicate ('obligation') that operates on nominalized sentences. (The 'that'-operator in Output 16 is a feature of IKL that allows nominalization of arbitrary sentences.)

**Output 15**

```
(forall (?shipment1)
  (if
    (shipment ?shipment1)
    (thing.is_registered ?shipment1)))
```

**Output 16**

```
(obligation (that
  (forall (?shipment1)
    (if
      (shipment ?shipment1)
      (thing.is_registered ?shipment1)))))
```

## 13 A real business example

We have illustrated some of the features of RECON by discussing small RECON sentences and the formal logic translations that are produced by the RECON tool. Of course, any real business rule will usually combine several features; e.g., in the introduction Example 1 involves type nouns, an adjective, a mass noun, and a unit of measurement.

**Example 1** Any shipment that consists of more than 1000 gallons of gasoline must be shipped via some registered tanker.

As RECON's translation of Example 1 below shows, the resulting IKL formulas are often quite verbose. Some of that could be simplified by an expert knowledge engineer. Ultimately, however, the complexity of the formula below is a reflection of the complexity of the state of affairs expressed in Example 1. Hence, capturing its content in a logic language will be a challenge for any person who is not very familiar with such languages. And, of course, the result is unreadable for anybody without these skills. Therefore, the example demonstrates the need for a tool, like RECON, that enables business rules to be captured formally in a way that is more accessible to the business experts.

**Output for Example 1**

```
(obligation (that
  (forall (?shipment1)
    (if
      (and
        (shipment ?shipment1)
        (exists (?gasoline2)
          (and
            (and
              (gasoline ?gasoline2)
              (forall (?quantity3)
                (if
                  (and
                    (quantity ?quantity3)
                    (quantity.is_the_volume_of.thing
                                ?quantity3 ?gasoline2))
                  (quantity.is_less_than.quantity
                      (Qvalue 1000 "gallon") ?quantity3) )))
            (shipment.consists_of.thing ?shipment1 ?gasoline2)
          )))
        (exists (?tanker4)
          (and
            (and
              (tanker ?tanker4)
              (thing.is_registered ?tanker4))
            (shipment.is_shipped_via.thing ?shipment1 ?tanker4)
          ))))))
```

## 14 The RECON tool

At this time, the grammar of RECON has been finalized [1], and version 1.0 of the RECON engine is available on Sourceforge.[3] The engine supports most of the features of the grammar. Internally, it consists of a *dictionary manager* for words and word forms, a *vocabulary manager* for term declarations, a *parser*, and a *logic generator*. The parser converts definitions, facts and rules into a syntactic parse graph. Because RECON supports multiword terms, and the same word can begin or appear in multiple terms, the parser first produces a lattice of all possible interpretations of the input string as a sequence of terms and keywords. The parser then tries to parse the first such sequence, and if it fails then the next one, and so on, until it finds a successful parse or discards the last alternative. Heuristics are used to choose the first and next sequence in each case. The actual parsing algorithm is a recursive descent algorithm, based on the formal RECON grammar. Logic generation begins with a *rewrite step* that revises the parse graph for compound phrases and quantities, converts adjectives and properties to qualified nouns (using their verb forms), and resolves back references. The *interpret step* then produces a formal logic structure from the revised parse graph, converting type nouns to quantified variables and properly placing and interpreting quantifiers and modalities. The resulting logic structure is exported in IKL form. The architecture, data structures, and algorithms are described in detail in a forthcoming publication.

## 15 Conclusions and Future Work

We are currently experimenting with the use of RECON for capturing engineering requirements and static rules written into information exchange standards. One area of future work on the tool is the support of collections (mentioned in the discussion of Examples 11 and 14). While we have successfully demonstrated feasibility of using the RECON language to author business rules, we have not built the infrastructure to make it a user-friendly product. For that, one would need to develop a front-end for dictionary and vocabulary management, and an authoring tool that supports the user in writing RECON.

## References


[1] Edward Barkmeyer and Andreas Mattas. *A Restricted English for Constructing Ontologies (RECON).* NISTIR 7868. National Institute of Standards and Technology, 2012.

[2] Ronald Denaux et al. "Rabbit to OWL: ontology authoring with a CNL-based tool". In: *Controlled Natural Language.* Springer, 2010, pp. 246–264.

[3] Norbert Fuchs, Uta Schwertel, and Rolf Schwitter. *Attempto Controlled English (ACE) Language ManualVersion 3.0.* University of Zurich, 1999.


[3] http://sourceforge.net/projects/nistreconst/

[4] Glen Hart, Catherine Dolbear, and John Goodwin. "Lege Feliciter: Using structured English to represent a topographic hydrology ontology". In: OWLED. 2007.

[5] Patrick Hayes and Chris Menzel. *IKL specification document*. 2006. URL: http://www.ihmc.us/users/phayes/ikl/spec/spec.html.

[6] ISO/IEC 24707-2007. Information technology Common Logic (CL): a framework for a family of logic-based languages. 2007.

[7] Ian Niles and Adam Pease. "Towards a standard upper ontology". In: *Proceedings of the international conference on Formal Ontology in Information Systems-Volume 2001*. ACM. 2001, pp. 2–9.

[8] Object Management Group (OMG). *Semantics of Business Vocabulary and Rules v1.0*. Tech. rep. 2010. URL: http://www.omg.org/spec/SBVR/1.0/.

[9] Adam Pease. *Standard Upper Ontology Knowledge Interchange Format*. 2009. URL: http://sigmakee.cvs.sourceforge.net/viewvc/sigmakee/sigma/suo-kif.pdf.

[10] Adam Pease and William Murray. "An English to Logic Translator for ontology-based knowledge representation languages". In: *Natural Language Processing and Knowledge Engineering, 2003. Proceedings. 2003 International Conference on*. IEEE. 2003, pp. 777–783.

[11] Francis Jeffry Pelletier. "Mass Terms: A philosophical Introduction". In: *Kinds, Things, and Stuff: Mass Terms and Generics*. Oxford University Press, USA, 2009.

[12] Bertrand Russell. "On denoting". In: *Mind* 14.56 (1905), pp. 479–493.

[13] John Sowa. *Common Logic Controlled English*. 2004. URL: http://www.jfsowa.com/clce/specs.htm.

[14] World Wide Web Consortium (W3C). *OWL Web Ontology Language Reference*. 2004. URL: http://www.w3.org/TR/2004/REC-owl-ref-20040210/.

[15] *Xtext 2.3 Documentation*. URL: http://www.eclipse.org/Xtext/documentation/2.3.0/Documentation.pdf.