Configuring Domain Knowledge for Natural Language Understanding

Matt Selway and **Wolfgang Mayer** and **Markus Stumptner**
University of South Australia
Adelaide
{<first_name>.<last_name>}@unisa.edu.au

Abstract

Knowledge-based configuration has been used for numerous applications including natural language processing (NLP). By formalising property grammars as a configuration problem, it has been shown that configuration can provide a flexible, non-deterministic, method of parsing natural language. However, it focuses only on *syntactic* parsing. In contrast, configuration is usually performed using knowledge about a domain and is semantic in nature. Therefore, we argue that configuration has the potential to be used, not only for syntactic processing, but for the semantic processing of natural language, effectively supporting Natural Language Understanding (NLU).

In this paper, we propose an approach to NLP that applies configuration to the (partial) domain model evoked by the processing of a sentence. This has the benefit of ensuring the meaning of the sentence is consistent with the existing domain knowledge. Moreover, it allows the dynamic incorporation of domain knowledge in the configuration model as the text is processed. We demonstrate the approach on a business specification based on the Semantics of Business Vocabulary and Rules.

1 Introduction

Knowledge-based configuration has been used in numerous applications. While historically used for configuring physical products, configuration has been applied to other domains such as software services, software product lines, and constraint-based language parsing [Hotz and Wolter, 2013].

In particular, [Estratat and Henocque, 2004; Kleiner *et al.*, 2009] have applied configuration to a translation of property grammars (a constraint-based linguistic formalism). By formalising property grammars as a configuration problem, they show that configuration can provide a flexible, non-deterministic method for parsing natural language. However, these approaches focus on syntactic parsing by using the configuration process to generate a parse tree. In contrast, configuration is usually applied to domain knowledge, that is, semantic processing. Furthermore, in [Kleiner *et al.*, 2009] additional processes are required in order to transform the parse tree into a semantic model to make use of the domain knowledge. This causes some issues in ensuring the consistency and correctness of the domain knowledge.

In this paper we present an approach to parsing natural language that performs semantic processing directly using configuration. Instead of a model of language *categories* (e.g. noun, verb, noun phrase) and the *properties* (or constraints) on those categories, such as property grammars, we use a model of domain concepts and the relations between them. As a result, we perform natural language *understanding*; at least with respect to the semantic model being used.

Our approach maintains the advantages of using configuration for natural language processing, while gaining the following: (1) a simplified lexicon containing minimal lexical information, (2) improved consistency of the domain knowledge as the configuration process ensures its consistency during parsing, and (3) the semantic disambiguation of terms.

As our aim is to support the translation of informal natural language business specifications into formal models, we demonstrate our approach on an example from the business domain. The example business specification is defined using the Semantics of Business Vocabulary and Business Rules (SBVR) [OMG, 2008], which we use as our semantic model. SBVR and the example are discussed in more detail later.

The remainder of this paper is organised as follows: Section 1.1 provides a brief introduction to SBVR and its concepts, Section 2 presents an example that will be used throughout the paper, Section 3 describes our approach to parsing natural language, Section 4 presents experimental results, Section 5 discusses related work, and Section 6 provides insight into future work and concludes the paper.

1.1 Brief overview of SBVR

The Semantics of Business Vocabulary and Business Rules (SBVR) is a standard developed by the Object Management Group (OMG) to facilitate the transfer of business knowledge between business people and the technical experts responsible for developing information systems for them [OMG, 2008]. It encompasses two aspects: (1) a meta-model for representing vocabularies, facts, and rules in a machine processable format, and (2) a controlled English notation for representing the same vocabularies, facts, and rules more suited to people. Therefore, SBVR supports the exchange of business specifications between both organisations and software tools.

The SBVR meta-model standardises concepts for the definition of business vocabularies (i.e. sets of concepts relevant to a particular organisation or business domain) and rules relating to those vocabularies. It is based on formal logic; primarily first-order predicate logic with an extension in modal logic (necessity, possibility, permissibility, and obligation).

Within the meta-model, vocabularies are defined on the basis of Meanings and Representations. They consist of sets of interrelated *object types*, *individual concepts*, and *fact types*. A distinction is made between a meaning and its representation, allowing a single concept to be represented with multiple words (possibly in different languages), images, or sounds.

The semantic structure of business rules are formed by the Logical Formulations aspect of the meta-model. This includes concepts for first-order logical operators (e.g. conjunction, disjunction, implication), quantification (e.g. universal, existential, exactly n), and modal operators (e.g. necessity, obligation). These concepts allow business people to define structural and operative rules. Structural rules include such rules as cardinality constraints on the relations between concepts and cannot be violated, while operative rules may be violated by a person involved in conducting the business.

2 Motivating Example

This section introduces an example that identifies the limitations of existing approaches and that will be used in the remainder of this paper to demonstrate our approach. It is an extract of the EU-Rent business specification included in the SBVR specification [OMG, 2008, Annex E].

EU-Rent is a fictional car rental company with a global presence. The example business specification defines domain specific vocabulary and rules for EU-Rent, its structure, and how it conducts its business. Figure 1 shows a portion of the vocabulary related to the organisational structure of EU-Rent. The following is a structural rule, based on this vocabulary, that defines a cardinality constraint on the part-of relationship between a 'branch' and a 'local area'.

(1) *Each branch is included in exactly one local area.*

rental organisation unit
  Definition: organisational unit that operates part of EU-Rent's car rental business
rental organisation unit *having rental responsibility*
  Definition: the rental organisation unit is responsible for the operation of customer-facing rental business
rental organisation unit *having area responsibility*
  Definition: the rental organisation unit includes organisation units for which it has the responsibility to coordinate operations and ensure resources
local area
  Definition: rental organisation unit that *has area responsibility*
branch
  Definition: rental organisation unit that *has rental responsibility*
branch *is included in* local area
  Synonymous Form: local area *includes* branch

Figure 1: Business vocabulary used by the example rule with SBVR markup: object types, *fact types*, and keywords.

Although quite simple, this example demonstrates a number of important concepts such as nouns, verbs, and quantifiers. The approach of [Kleiner *et al.*, 2009] processes the sentence by executing a series of model transformations that result in a UML model of the text, using SBVR as an intermediate model between the natural language text and UML. The steps up to the creation of the SBVR model are as follows:

1. a text-to-model transformation creates an Ordered Words model that annotates the words with their position in the sentence
2. a model-to-model transformation creates a Labelled Words model by labelling each word with their possible syntactic categories using a lexicon (model)
3. *configuration* is used to transform the Labelled Words model into a Syntax model, performing syntactic and grammatical analyses, and
4. a model-to-model transformation creates an SBVR model from the Syntax model

Performing this process on the example sentence would result in the Syntax and SBVR models displayed in Figure 2.

This approach provides a flexible, non-deterministic, and extensible method of parsing natural language [Kleiner *et al.*, 2009]; however, it has several issues, chiefly: (1) it is primarily *syntactic*, (2) it requires a detailed lexicon, and (3) the mapping to SBVR can be problematic, e.g. in the handling of 'local area' two correct interpretations can be conceived.

The first is the main issue as, although [Estratat and Henocque, 2004] suggest that configuration can combine syntactic and semantic analysis, it is primarily used for generating syntactic parse trees. As a result, a sentence could be syntactically correct but not meaningful and, therefore, must be linked to the semantics somehow (e.g. through a model transformation to SBVR like that used in [Kleiner *et al.*, 2009]).

Although [Kleiner *et al.*, 2009] introduce some semantic elements into their model (i.e. each category can be linked to a basic element of the SBVR model[1]), they do so only to ease the transformation to the SBVR model. The existence of the SBVR elements does not provide any semantic guarantees. Therefore, semantic inconsistencies need to be resolved either during the transformation to SBVR, making it much more complex, or by post-processing of the SBVR model. However, this seems unnecessary if it can be achieved during the configuration process itself.

The requirement of a detailed lexicon is more an issue for our target application area than with the parsing method itself. In the context of businesses creating and maintaining their own sets of domain specific business vocabularies and business rules, we do not see business people defining detailed lexicons with linguistic information such as voice, genre, transitivity, etc. In this application area, the business vocabulary is more like a glossary containing domain specific words and their definitions, like that of Figure 1, rather than a dictionary containing detailed lexical information. Therefore, an approach that requires less linguistic information to be defined is required for our purposes.

[1] This is not shown in Figure 2a for readability.

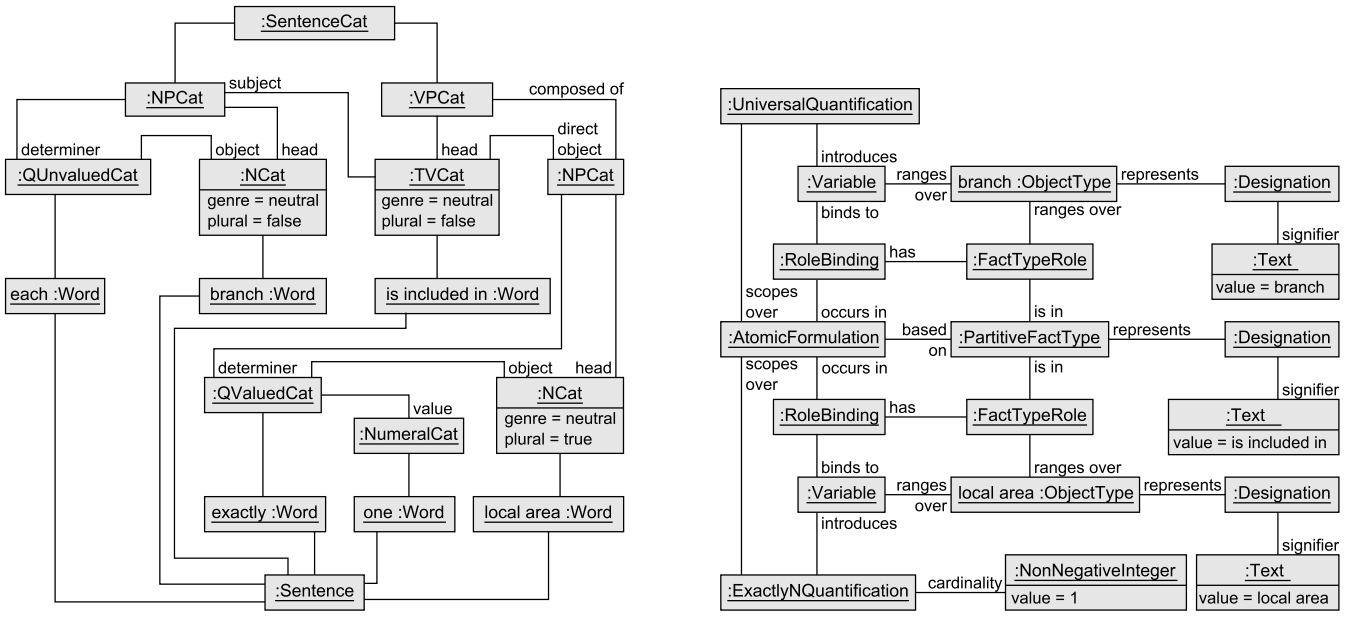

Figure 2: Example Syntax model (a) and SBVR model (b) generated by the process of [Kleiner *et al.*, 2009]

Finally, there are some problems with mapping the syntactic tree to the SBVR semantics. Consider how the term 'local area' is handled in the above example. For simplicity, 'local area' is a single noun, which maps directly to the object type 'local area' in SBVR. However, in reality it would be considered a noun phrase, where the term 'local' would be used in an adjective sense and 'area' would be the noun. This would map to the object type 'area' with the characteristic 'being local'. However, in the vocabulary of EU-Rent, 'local area' is a single concept and should not be decomposed in this way.

It seems a simple problem to fix: the term 'local area' could be included as a noun in the lexicon, which is what would happen if a business were defining its vocabulary. However, unless the individual terms 'local' and 'area' were removed, which could affect the processing of sentences in other contexts, it would result in two correct parses of the sentence: one in which 'local area' is treated as a simple noun and one treating it as a noun phrase. The transformation to the SBVR model would not resolve this issue either, as both forms have a valid mapping. Therefore, the user would have to select the preferred mapping or the SBVR model would have to be processed to see if either one or the other has already been created. Either way it makes the process more cumbersome, whereas we propose that by configuring the SBVR model directly, this problem would be avoided (as long as only one or the other has been specified in the vocabulary).

It could be argued that this problem is a quirk of the SBVR model as other semantic representations with a structure more similar to the parse tree would have more direct mappings. However, it is an important issue as SBVR has gained traction in our application domain in recent years [Nemuraite *et al.*, 2010; Sukys *et al.*, 2012] and is an important part of the OMG's Model-Driven Architecture [OMG, 2008]. Moreover, we will show that our proposed approach does not reduce the flexibility with respect to the semantic representation used.

3 Parsing Process

In order to overcome these limitations we propose the use of knowledge-based configuration on the semantic representation directly, rather than configuring a syntactical parse tree. In this way we maintain the benefits of parsing using configuration, while ensuring semantic consistency and removing a step from the process. Furthermore, this approach remains agnostic with respect to the semantic representation used; although we utilise the SBVR meta-model in this paper.

In order to avoid complex syntactical analysis, which is a difficult problem in itself, we use an approach inspired by Cognitive Grammar, a theory of grammar from the field of Cognitive Linguistics [Langacker, 2008]. Cognitive Grammar takes a holistic view of language, combining syntax, semantics, and pragmatics into a unified whole. In particular, our approach is based on that of [Holmqvist, 1993], which provides a computational model of Cognitive Grammar.

In Cognitive Grammar, the meaning of an expression is understood by combining the semantic structures evoked by its constituent expression into a unified structure; a process called *semantic accommodation* in [Holmqvist, 1993]. Evoking the semantic structure of an expression is a relatively simple endeavour as the two are linked; therefore, a semantic structure is evoked by looking up the expression in a lexicon. As a result, our method is able to do away with traditional syntactic analysis for a much simpler model, leaving most of the effort in understanding the expression to be performed by the accommodation process. With the aim of combining semantic structures into a composite structure based on the allowable relationships between them, the accommodation pro-

cess is analogous to a configuration task.

The syntactic analysis and the accommodation process using configuration are detailed in the following sections. These two processes are performed iteratively with the first using the expectations, or placeholders in our case, to propose possible parses of the sentence, and the second combining the semantic aspects of the suggested parses into a complete structure. The result is a progressively more detailed and complete set of domain knowledge containing the concepts, their definitions, and their associated rules.

3.1 Syntactic Analysis

The syntactic analysis of our approach is primarily the evocation of semantic structures from the lexicon, taking into account grammatical dependencies. For example, in the example expression, there are two quantifiers 'each' and 'exactly one'. If only the evoked semantic structures were known, the configurator would not know which quantifier applies to what concept, yet we know that 'each' is supposed to apply to the term 'branch' and 'exactly one' to 'local area'.

Traditionally, these dependencies are handled by the relationships between categories (as shown in Figure 2a). However, as this leads to complex lexicons that are not suitable for our target application and require complex processing to produce, we account for these dependencies using so called *grammatical expectations* [Holmqvist, 1993]. Grammatical expectations are a relatively simple model of grammatical dependency that identify locations in an expression where other expressions are "expected" to fill. Moreover, grammatical expectations are a good match for SBVR-based semantics, as they are similar to the placeholders of fact types. In [Holmqvist, 1993], only left and right expectations were introduced, which search to the left or right for another expression. We also introduced internal expectations, which search within the span of the expression, in order to more easily deal with SBVR fact types with more than two placeholders.

This model defines lexicon and lexical entries as follows.

Definition 1 (Lexicon)**.** A lexicon is a tuple
$l = (E, LE, lookup)$
where E is a set of expressions, LE is a set of lexical entries, and $lookup : E \rightarrow 2^{LE} \backslash \emptyset$

Definition 2 (Lexical Entry)**.** A lexical entry is a tuple
$le = (e, ss, GE, type_{GE})$
where e is an expression of one or more words, ss is its associated semantic structure, GE is a set of grammatical expectations, and $type_{GE} : GE \rightarrow \{left, internal, right\}$ assigns a type to each grammatical expectation $ge \in GE$.

This definition is purposefully generic with respect to the form the semantic structure takes. While we use the SBVR meta-model, other semantic representations could be used. As a result, our approach remains flexible in terms of varying the model being configured, as in [Kleiner *et al.*, 2009].

The lexicon is partly predefined. For example, words with explicit semantics in SBVR, such as those for quantifications, logical operators, etc., are explicitly defined as certain expressions, semantic structures, and grammatical expectations. Domain specific terms are provided by the vocabulary of a business specification, e.g. a glossary of terms, which have their semantic structures and grammatical expectations determined by the representation of object types and fact types in SBVR. In our application domain, the vocabulary is provided by business people, which drives our need for a simple lexicon with minimal information.

Using grammatical expectations, the syntactic analysis is performed incrementally, when an expression is found to fill an expectation (i.e. the expectation is said to *catch* the expression [Holmqvist, 1993]) a possible parse is proposed and kept in a suggestion list. Since the number of suggestions increases rapidly, the suggestion list is kept short by ordering the suggestions using heuristics to identify the *best* parse and pruning off any suggestions over a limit and/or that are not considered good candidates [Holmqvist, 1993]. The metrics used by the heuristics include: (1) catching distance, the linear distance to the word (or combination) filling a placeholder; (2) binding energy, the summation of all catching distances in a suggestion; (3) local coverage, the ratio of words in the suggestion to words spanned by the suggestion; and (4) global coverage, the ratio of words in the suggestion to all words of the expression encountered up to the current point.

The suggestion list and suggestions are defined as follows.

Definition 3 (Suggestion List)**.** A suggestion list SL is an ordered set of suggestions, where each suggestion $s \in SL$ is a tuple $s = (SLE, C, be, lc, gc)$ such that:

- SLE is a set of lexical entries or previous suggestions included in s
- C is a set of tuples
  $$catch = (sle_1 \in SLE, ge, sle_2 \in SLE, cd)$$
  that associate a grammatical expectation, ge, of the catching lexical entry or suggestion, sle_1, to the lexical entry or suggestion that it catches, sle_2, with the catching distance, cd.
- $be = \sum_{c \in C} c.cd$ is the binding energy of the suggestion,
- lc is the local coverage of the suggestion,
- gc is the global coverage of the suggestion

The order for each $s_1, s_2 \in SL$ is determined by:
$$s_1 \preceq s_2 \iff (s_1.gc, s_1.lc, s_1.be) \leq_l (s_2.gc, s_2.lc, s_2.be)$$

Based on the preferences that: the best parse should cover the entire sentence; suggestions should have no holes; and, words should be captured at the shortest distance.

Due to the recursive compositional nature of the suggestions, i.e. each $x \in SLE$ is a lexical entry or another suggestion, the catching information constitutes a non-traditional parse tree. Figure 3 shows an example of a parse tree produced by our syntactic analysis compared to the traditional parse tree equivalent to Figure 2a. The parse tree and the partial SBVR model (i.e. the semantic structures) of the suggested parse are provided to the semantic accommodation process to be configured into a complete model.

The general algorithm for the syntactic analysis is as follows, for each word in the expression:

1. Retrieve the entry for the current word from the lexicon
2. If the current word has any left placeholders, for each suggested parse in the suggestion list:

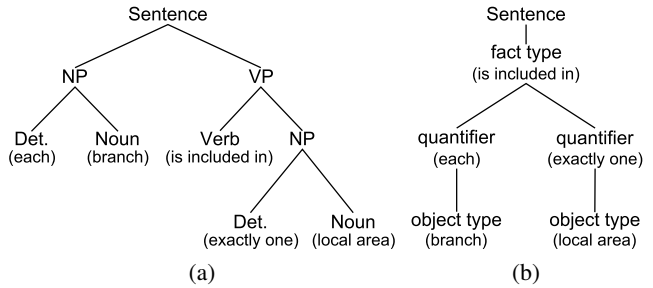

Figure 3: A traditional parse tree (a) and one created by our analysis (b)

   (a) catch the closest word (or word combination) to the left of the current word
   (b) add the new suggested parse to the suggestion list
3. Else, for each suggested parse in the suggestion list, if the previous word or combination has any internal placeholders and the current word is within its span:
   (a) catch the current word with the internal placeholder
   (b) add the newly suggested parse to the suggestion list
4. Else, for each suggested parse in the suggestion list, if the previous word or combination has any right placeholders:
   (a) catch the current word with the right placeholder
   (b) add the newly suggested parse to the suggestion list
5. Update the heuristics, distances between words, order the suggestion list, and cull excess entries
6. Provide newly suggested parses to the semantic accommodation/configuration process
7. Remove suggestions that failed accommodation

An example of the syntactic analysis after a complete parse of the example sentence is displayed in Figure 4.

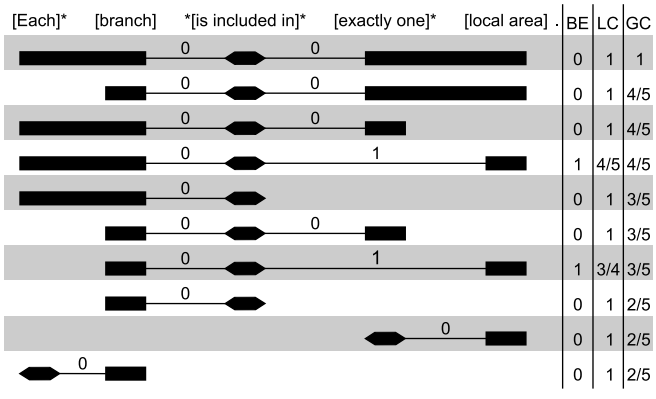

Figure 4: Lexical analysis of the rule 'Each branch is included in exactly one local area.' Asterisks represent the grammatical expectations, hexagons represent the catching word, rectangles represent the caught word, catching distance is shown above each line.

3.2 Semantic Accommodation/Configuration

Parses suggested by the syntactic analysis are sent to the semantic accommodation process to determine whether or not they are admissible in the (SBVR) semantics. Rather than the numerous processes for accommodation discussed in [Holmqvist, 1993], we utilise knowledge-based configuration to perform the accommodation. Specifically, component-oriented configuration is used, which combines advantages from connection-, resource-, and structure-based approaches [Soininen *et al.*, 1998; Stumptner, 1997]. Moreover, the object-oriented nature of component-oriented configuration lends itself more easily to that of the SBVR meta-model.

Using the terminology of [Soininen *et al.*, 1998], the SBVR meta-model constitutes the *configuration model knowledge* of our approach, i.e. it defines the types of entities, properties, and rules that specify the set of correct configurations. It follows that an SBVR (terminal) model constitutes the *configuration solution knowledge* or (possibly partial) *configuration*. Lastly, the parse tree created by the lexical analysis constitutes the *requirements knowledge*, i.e. additional constraints on the configuration that are not strictly part of the configuration model. For example, the SBVR meta-model may allow either quantification to be applied to either object type in the example rule, however, the grammatical dependencies require that 'each' be applied to 'branch' and 'exactly one' to 'local area' for the correct interpretation of the sentence.

Since the SBVR (meta-)model is defined using ECore, the Eclipse Modelling Framework [2] implementation of EMOF from the MOF specification of the OMG [OMG, 2006], we first discuss its mapping to the configuration ontology of [Soininen *et al.*, 1998]. The configuration ontology defines standard concepts for representing the different aspects of configuration knowledge including: taxonomy, attributes, structure, topology, and constraints. An example of the mapping for SBVR is displayed in Figure 5. It focuses on the configuration model knowledge, with some example component individuals. For simplicity, port individuals are not included in the figure. The mapping is by no means complete, but provides a link between the meta-model representation and the representation used for the configuration task.

Taxonomy

ECore allows classification hierarchies to be defined through the use of the concepts `EClass` and `EObject`, and the relations *eSuperTypes* and *eClass*. The concept `EClass` generically represents a type and therefore could be mapped to `Configuration Type`; however, ECore does not distinguish the sub-types `Component Type`, `Port Type`, `Resource Type`, and `Function Type` in the same manner as [Soininen *et al.*, 1998]. Therefore, it is more appropriate to map instances of `EClass` to `Component Type`s. Other ECore concepts map to the other configuration types. Therefore, only component types have a classification hierarchy; the others are made direct subtypes of their appropriate configuration type (i.e. `Attribute Type`, etc.).

Sub-types and super-types in ECore are represented by *eSuperTypes*. This relation defines the direct super-types of an

[2] `http://www.eclipse.org/modeling/emf/`

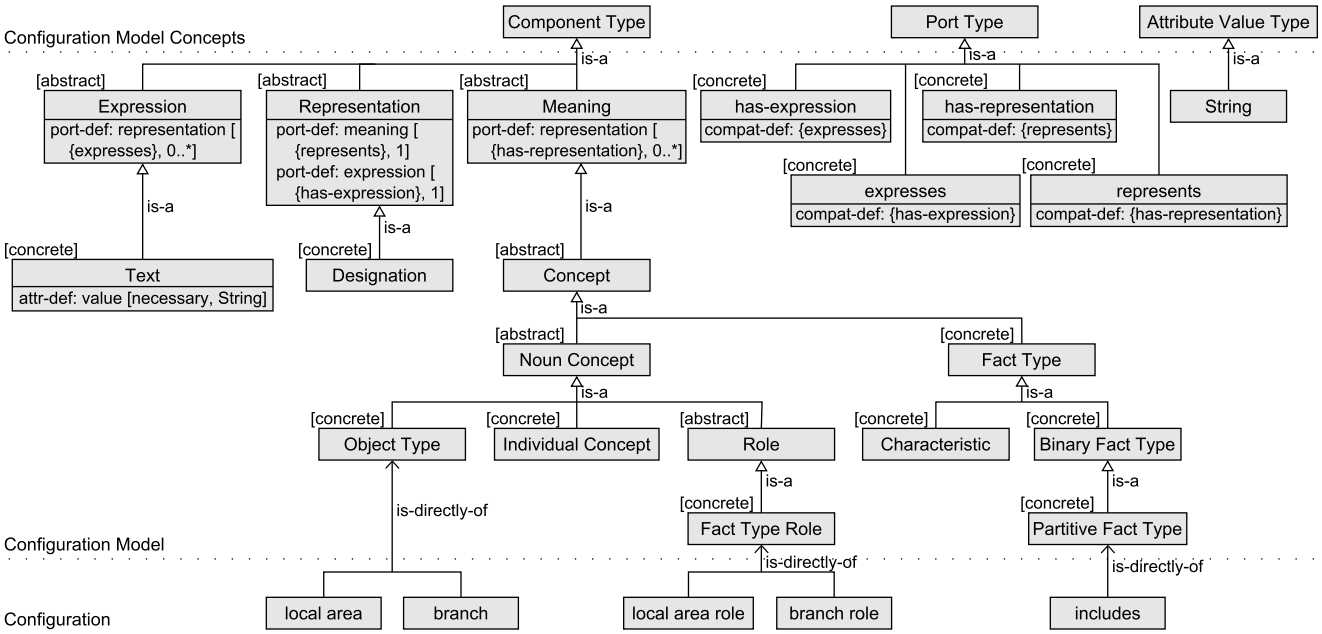

Figure 5: Fragment of the configuration model derived from the ECore mapping of the SBVR meta-model.

`EClass` and, therefore, maps to the *isa* relation. Moreover, multiple inheritance is allowed in both representations.

In ECore, an `EClass` may be abstract, i.e. cannot have any instances. However, in the context of configuration, it makes sense to relax this definition to allow partial information of a configuration, such as in [Soininen *et al.*, 1998]. Therefore, abstract and non-abstract (i.e. concrete) types in ECore are mapped to abstract and concrete classes in the ontology using the appropriate `Abstraction Definition`.

Instances of `EObject` represent `Individuals` from the ontology. In ECore, these are associated to their type by *eClass*, which maps to *is directly of*. Moreover, since *eClass* specifies the `EClass` of an individual, `EObject` necessarily maps to `Component Individual`.

Attributes

Attributes in ECore are represented by the concept `EAttribute`, which have a *name*, a type specified by *eAttributeType*, and relations for the lower and upper bounds of their cardinality (*lowerBound* and *upperBound*, respectively).

An *eAttributeType* links an `EAttribute` to its `EDataType`, which maps to the concept `Attribute Type` from [Soininen *et al.*, 1998]. It follows that `EAttributes` map to `Attribute Definitions` with the appropriate `Attribute Name`, `Attribute Value Type` (from *eAttributeType*), and `Necessity Definition`. The value of an attribute for a specific `EObject` is mapped to an `Attribute` with the respective `Attribute Value` and `Component Individual`.

In ECore, attributes can have a zero-to-many cardinality, while `Necessity Definitions` are restricted to exactly one (necessary) and at most one (optional) attributes. As a result, there exists only a partial mapping to the configuration ontology; however, this is not a problem for the SBVR meta-model as it only includes necessary and optional attributes.

Structure and Topology

The ontology of [Soininen *et al.*, 1998] differentiates between `Part Definitions`, which specify the compositional structure of components, and `Port Definitions`, which specify the topological connections (either physical or logical) between components. `Part Definitions` constitute a direct *has-part* relation between components. This relation must be anti-symmetric and anti-reflexive. Moreover, the transitive closure of *has-part* defines a *transitive has-part* relation, which must also be anti-symmetric and anti-reflexive. `Port Definitions` have no such restriction.

In ECore, both `Part Definitions` and `Port Definitions` are represented by the concept `EReference`. An `EReference` may be a *containment* reference (for compositional relationships) and/or it may have an *eOpposite* for bi-directional relationships.

While it seems intuitive to map containment and unidirectional references to part definitions, and bi-directional relationships to port definitions, this is not possible as ECore does not uphold the anti-symmetric and anti-reflexive requirements of *has-part* relations. For example, the situation shown in Figure 6, in which the transitive closure of the *has-part* relations between Meaning, Representation, and Expression are reflexive, is allowable in ECore but not the configuration ontology. To determine those `EReferences` that could be mapped to part definitions would require analysis of the meta-model; instead, we map all `EReferences` to ports. As a result, we effectively use ports as a generalised structural relationship similar to that described in [Hotz and Wolter, 2013].

A `Port Definition` requires a `Port Name`, a `Possible port type set`, and a `Cardinality`. Similar to `EAttribute`, `EReferences` have a *name*,

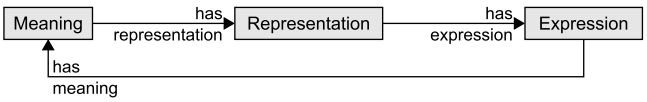

Figure 6: Transitive relationships in the SBVR meta-model

a *lowerBound*, and an *upperBound*. Therefore, an `EReference` maps to a `Port Definition`, with the *name* mapping to the `Port Name`, and the *lowerBound* and *upperBound* are mapped to the `Cardinality`. `Port Types` and their `Compatibility Definitions` are created for each `EReference` to ensure the associated ports can only connect to each other correctly.

When two `EObjects` are associated to one another through an `EReference` the appropriate `Port Individuals` of the equivalent `Component Individual` are *connected-to* each other.

Constraints

Arbitrary constraints in ECore are specified using annotations on model elements, written in some constraint language. These annotations are mapped to `Constraint Instances` and their corresponding `Constraint Expressions`. Special cases of constraints, particularly `Property Definition` and its sub-types (`Attribute Definition`, etc.), are utilised by previous mappings.

The constraints defined in our configuration model come from the SBVR specification. An example is shown in Figure 6 in that, the 'has meaning' relation between an 'Expression' and a 'Meaning' is allowed if and only if the 'Meaning' is connected to the 'Expression' through a 'Representation' and the 'has representation' and 'has expression' relations.

In the configurator used by our implementation the different aspects of the configuration knowledge are mapped to a generative CSP (or GCSP) as described in [Stumptner *et al.*, 1998]. This particular approach differs in its definition of some of the previously discussed concepts of [Soininen *et al.*, 1998] in the following respects:

- Non-leaf nodes in the taxonomy are assumed to be abstract; therefore, concrete component types that have sub-types are split into two types: an abstract super-type and a concrete sub-type.
- Ports are specified as necessary or optional; therefore, `Port Definitions` with higher cardinalities result in multiple ports that are then grouped into *port sets*, which allow quantitative reasoning over their members.

By configuring the SBVR models directly, we perform configuration of the concepts in a business specification rather than a syntactic parse tree as in [Kleiner *et al.*, 2009]. This results in an iteratively more detailed domain model, where new domain knowledge is taken into account each time a new sentence is processed. In addition, inconsistencies can be detected more easily than by reprocessing the model after new knowledge is added through a model transformation. Finally, the mapping to SBVR is simplified as the lexicon maps directly to the semantics, rather than an intermediate syntactic model. This solves the issue of multiple parse trees with correct mappings to SBVR as, for example, the parse of 'Each branch is included in exactly one local area' where 'local area' is considered a noun phrase would be inconsistent with the domain knowledge, while the parse where 'local area' is considered a noun would be consistent.

4 Experimental Results

We present the results of early experiments on the configuration of domain knowledge for NLP. We performed multiple tests of the example structural rule (1) and gathered statistics on the performance of the configurator in configuring these kinds of models. The results are summarised in Table 1.

The configurations produced were evaluated by hand for correctness. In each case the configuration was correct (corresponding with that shown in Figure 2b). The high number of variable assignments are due to relationships in the SBVR meta-model with cardinalities higher than one producing multiple ports in the configuration model; therefore, most port assignments are to the unconnected state.

It is interesting to note the correlation between the (minimum) number of backtracks and the number of components generated. This is due to the nature of the SBVR meta-model in two respects: (1) it contains a large number of relations with a cardinality of zero-to-many, and (2) it uses reified relations, which means that, in terms of configuration, each relation is represented as a component.

To prevent spurious connections between components, ports representing a zero-to-many relation are first set to the unconnected state; therefore, they are only connected to a component if being unconnected violates some constraint, causing a backtrack. Furthermore, the use of reified relations means that new *relation* components need to be created, even if it results in connecting two existing (non-relation) components. Therefore, in the optimal search of only connecting existing (non-relation) components, there will always be the same number of backtracks as generated components.

The higher number of backtracks in other configurations of the example are the result of the non-deterministic solver attempting variable assignments in a suboptimal order. This has a negative impact on performance. However, this could be avoided by providing SBVR specific procedural strategies for guiding the search [Stumptner *et al.*, 1998; Hotz and Wolter, 2013]. For example, ordering heuristics can be provided to change the order in which different component types or port types are assigned. Moreover, the search space could be reduced by preventing the creation of certain component types. For example, we assume a sentence is to be interpreted in the context of a provided vocabulary, hence we could prevent the

	Sentence (1)
# Constraints	197
# Variable Assignments	5162
Min. # Backtracks	6
Max. # Backtracks	25
Ave. # Backtracks	17
# Components Generated	6

Table 1: Performance statistics of the configuration process

creation of new object types, fact types, and other concepts related to the vocabulary aspects of the SBVR meta-model.

We are in the process of implementing a larger example, a portion of which assigned 4812 variables, generated 5 new components, and took 49 backtracks to do so. This emphasises the need for heuristics to help guide the search.

5 Related Work

Previous work in using configuration for natural language parsing has translated property grammars into a configuration model [Estratat and Henocque, 2004]. Using this approach, a simple context free-grammar ($a^n b^n$) and a subset of French were processed. This approach focuses primarily on the syntactic aspect of generating parse trees. Although the possibility of incorporating semantics is suggested in [Estratat and Henocque, 2004], none were incorporated in the processing of the natural language subset.

The approach of [Estratat and Henocque, 2004] was adapted to English and a Model-Driven Engineering environment in [Kleiner *et al.*, 2009]. Although their focus remains on the syntactic aspect of generating parse tress, SBVR semantics are partially taken into account by associating elements of the parse tree with SBVR types. This information is used to simplify the model transformation; however, the domain knowledge included in the SBVR model is not taken into account and, therefore, the process is not truly semantic.

Other approaches have used standard CSP translations of Dependency Grammars [Duchier, 1999] and, more recently, Property Grammars [Duchier *et al.*, 2012] in order to process natural language. However, both of these approaches focus on syntactic parsing, while we aim to incorporate semantics and domain knowledge directly into the parsing process.

6 Conclusions and Future Work

In this paper we have presented an approach to natural language processing that utilises configuration of domain knowledge to determine the validity of an expression. In effect, this performs natural language *understanding*, at least in terms of the semantic representation used. Moreover, we have demonstrated how techniques from Cognitive Linguistics can be combined with a translation of the SBVR meta-model (the semantic representation in our case) into a configuration problem in order to achieve this natural language understanding.

Our approach is novel in its combination of techniques from Cognitive Linguistics and configuration, and in that it performs configuration directly on the semantics of the domain knowledge. This is in contrast to previous approaches that use configuration or CSPs for natural language processing, as they tend to focus only on the syntactic aspect of generating a parse tree. As a result, our approach benefits from a simplified lexicon (important to our application in the business domain), improved mapping to the target semantics, and the semantic disambiguation of terms during processing.

The presented experimental results demonstrate the feasibility of our approach. In its current form, however, the configuration of the SBVR model can be inefficient and, therefore, future work will look at providing heuristics in order to ensure better performance in the configuration process. In addition, a more thorough evaluation of the process will be performed over larger examples in order to determine the effect of growing domain knowledge on the process.

References

[Duchier *et al.*, 2012] Denys Duchier, Thi-Bich-Hanh Dao, Yannick Parmentier, and Willy Lesaint. Property grammar parsing seen as a constraint optimization problem. In *Proc. Formal Grammar 2010/2011*, LNCS 7395, pages 82–96, 2012.

[Duchier, 1999] Denys Duchier. Axiomatizing dependency parsing using set constraints. In *Proc. Sixth Meeting on Mathematics of Language*, pages 115–126, 1999.

[Estratat and Henocque, 2004] Mathieu Estratat and Laurent Henocque. Parsing languages with a configurator. In *Proc. ECAI'2004*, volume 16, pages 591–595, 2004.

[Holmqvist, 1993] K. B. I. Holmqvist. *Implementing cognitive semantics: image schemata, valence accommodation and valence suggestion for AI and computational linguistics*. PhD thesis, Dept. of Cognitive Science Lund University, Lund, Sweden, 1993.

[Hotz and Wolter, 2013] Lothar Hotz and Katharina Wolter. Beyond physical product configuration configuration in unusual domains. *AI Communications*, 26:39–66, 2013.

[Kleiner *et al.*, 2009] M. Kleiner, P. Albert, and J. Bézivin. Configuring models for (controlled) languages. In *Proc. ConfWS'09*, pages 61–68, 2009.

[Langacker, 2008] R. W. Langacker. *Cognitive grammar: a basic introduction*. Oxford University Press, Oxford, New York, 2008.

[Nemuraite *et al.*, 2010] Lina Nemuraite, Tomas Skersys, Algirdas Sukys, Edvinas Sinkevicius, and Linas Ablonskis. VETIS tool for editing and transforming SBVR business vocabularies and business rules into UML&OCL models. In *Proc. ICIST 2010*, pages 377–384, 2010.

[OMG, 2006] OMG. *Meta Object Facility (MOF) Core Specification*. Object Management Group, 2006.

[OMG, 2008] OMG. *Semantics of Business Vocabulary and Business Rules (SBVR), v1.0*. Object Management Group, 2008.

[Soininen *et al.*, 1998] Timo Soininen, Juha Tiihonen, Tomi Männistö, and Reijo Solunen. Towards a general ontology of configuration. *AI EDAM*, 12(04):357–372, 1998.

[Stumptner *et al.*, 1998] Markus Stumptner, Gerhard E. Friedrich, and Alois Haselböck. Generative constraint-based configuration of large technical systems. *AI EDAM*, 12(04):307–320, 1998.

[Stumptner, 1997] Markus Stumptner. An overview of knowledge-based configuration. *AI Communications*, 10(2):111–125, 1997.

[Sukys *et al.*, 2012] Algirdas Sukys, Lina Nemuraite, Bronius Paradauskas, and Edvinas Sinkevicius. Transformation framework for SBVR based semantic queries in business information systems. In *Proc. BUSTECH 2012*, pages 19–24, July 22-27 2012.