Towards a Description of Symbolic Maps

Daniel Couto Vale
SFB/TR8 Spatial Cognition
University of Bremen
danielvale@uni-bremen.de

Elisa Vales
SFB/TR8 Spatial Cognition
University of Bremen
evales@uni-bremen.de

Rumiya Izgalieva
SFB/TR8 Spatial Cognition
University of Bremen
rumiya@uni-bremen.de

Abstract

Symbolic resources for text synthesis and text analysis are typically created and stored separately. In our case, we have a KPML-resource (Nigel) and a CCG for English. In this paper, we argue that reversing efficient resources such as ours cannot in general be achieved. For this reason, we propose a symbolic map that can be converted automatically into both synthesis- and analysis-oriented resources. We show that completeness of description can only be achieved by such a map while efficiency concerns can only be tackled by the directed rules of task-oriented resources not because of the current state of the art, but because reversing task-oriented symbolic resources is impossible in principle.

1 Introduction

Currently, symbolic resources guiding text analysis and text synthesis are created and stored separately. Several researchers have attempted to use the same resource for both tasks (Kasper, 1988; Neumann, 1991; Neumann and van Noord, 1992; Strzalkowski, 1994; O'Donnell, 1994; Pulman, 1995; Klarner, 2005) motivated by the fact that this would not only be cognitively more plausible but also allow translation at a semantic level, integration of new words from analysis into synthesis, reduction of costs in engineering as well as making it easier to share information among research groups of different fields.

The resources we currently use in human-robot interaction in English are also separate: a KPML-resource (Nigel) and a CCG. The specialty about Nigel and our CCG is that they share not only the same kind of semantics, but also the same mapping between symbolic and semantic structures. Here 'symbolic structure' is understood as KPML's 'structure' and CCG's 'sign', and corresponds to 'grammatical constructions' of cognitive semantics (Lakoff, 1987), to 'linguistic mediation' of truth-reference semantics (Smith and Brogaard, 2003), and to 'wording' of systemic functional linguistics (Matthiessen, 1995; Matthiessen and Halliday, 1999; Matthiessen and Halliday, 2004; Halliday and Matthiessen, 2014).

In this paper, we shall review the available directed rules that constitute the resources in KPML and OpenCCG and argue that they are useful in their respective tasks – either synthesis or analysis, – but are either unsuitable or not competitive for the inverse task.

Aiming not at reversibility but at reusability, we propose to create a map between symbolic and semantic structures that can be compiled into both synthesis-oriented and analysis-oriented resources. With this approach, we aim at separating concerns, so that efficiency can be tackled by the directed rules of task-oriented resources and completeness of description by a less efficient uncompiled shared symbolic map.

2 Irreversibility of Current Resources

In computational linguistics, approaches to text processing can be divided into statistical and categorial according to the usage of graded or binary relations between inputs and outputs of processing. Approaches can also be divided depending on whether the textual content is a representation of something else (symbolic) or whether it is a representation of the text itself (non-symbolic). In this sense, approaches that have a semantic structure as input or output are symbolic and those that make use of a syntactic tree whose composite-component relations do not match the ones of semantics are not. The present work falls into the symbolic subset. Although our initial attempt is categorial, the ideas presented here can be used in statistical approaches as well, provided that these approaches are symbolic in nature.

Looking from the perspective of the philosophy of language, in the last 50 years, computational efforts in categorial symbolic text processing have converged on one single notion of a symbolic map. In such a notion, both symbolic (lexical or grammatical) and semantic structures play an essential role in deciding which analytical and synthetic hypotheses are to be taken further or discarded. On the text synthesis front, systemic networks were used by both KOMET and Penman engines as directed rules for text synthesis. Those engines were later unified into the KOMET-Penman Multilingual

Engine (KPML Engine) (Bateman, 1995a; Bateman, 1995b; Bateman, 1996; Bateman, 1997). On the text analysis front, typed-feature unification was developed and implemented in engines for a family of highly lexicalised grammatical frameworks (HLG). Combinatory Categorial Grammars (CCG) (Steedman, 1987; Steedman, 1996; Steedman, 1998; Steedman and Baldridge, 2011) are a special type of HLG that reduce the task of text analysis to accepting or rejecting hypotheses of both symbolic and semantic composition during functional unification. KPML and OpenCCG are then only candidates for consideration because they allow the implementation of a shared symbolic map. In other words, a pair of engines that support such a map is a necessary and sufficient condition for the reusability scheme we propose.

In the following, we shall review the grammatical notions embedded in the resources for KPML and OpenCCG in order to support our argumentation that reversibility of such directed resources is not to be achieved.

2.1 Resources for KPML

According to the KPML documentation (Bateman, 1996) and our own inspection of Nigel, resources for KMPL may contain three kinds of realisation operations: structural (insert, conflate, expand), linear (partition, order, order-at-front, order-at-end), and inter-rank (preselect, agreement, classify, outclassify, inflectify, lexify).

Below symbolic structure, textual tokens are produced by morphological realisation operators of two kinds: one for selecting token copies (preselect-substance, preselect-substance-as-stem, preselect-substance-as-property), and one for modifying them (morphose).

These realisation operators are bundled in wording 'patterns' that are linked to classes of wordings (grammatical features). The typology arising from these classes is used as a network of options (network of grammatical systems) among structure kinds. The selection of a structure kind of a system is done by a decision tree (chooser). Each decision in the decision tree is achieved by inspecting (inquiry) a semantic and lexical specification for a text. The decision tree contains not only decisions (ask) but also mappings from lexical/semantic constituents to functions of symbolic constituents (identify, copyhub, choose, pledge, termpledge). Values can be associated with a function (concept, modification-specification, terms, term).

Finally, there are four ways to produce a token in KPML. Three of them consist of selecting a word and selecting its form with a form class. The actual token production is left to a morphological component. Two word selection strategies are: selecting a word grammatically (lexify, classify, outclassify) and selecting a word associated with a particular conceptual value (term-resolve-id). A mapping between concepts and words is provided either by concept-word links (annotate-concept) or by embedding word specifications into what would otherwise be a pure semantic specification (lex). A distinct mapping function between the intersection of form classes and word pattern indexes is implemented in LISP for every linguistic resource. At the morphological level, a token is produced by selecting a token model and applying any necessary morphological modifications to it (preselect-substance, preselect-substance-as-stem, preselect-substance-as-property, morphose).

Therefore, as with any other categorial text synthesiser, KPML traverses a network of options among progressively finer types of structures and makes choices between different structure types depending on semantical and lexical restrictions. Its speciality comes not from the general approach, but from the amount and quality of detailed linguistic knowledge applied to the synthesis of text in Nigel, which makes Nigel a good option for our applications that demand natural utterances. This is also the main reason why so many attempts have been made to use Nigel for text analysis.

2.2 Resources for OpenCCG

OpenCCG, as for any other engine using chart parsing, relies on the assumption that a hypothesised structure is only to be considered if it is part of a structure for the whole input text. This assumption of syntagmatic holism was first formulated by Frege (1884) and Wittgenstein (1921; 1922). Chart parsing with CCGs goes beyond: both syntagmatic and paradigmatic holisms are to be enforced, i.e. semantic fitting is used as a filter for analytical hypotheses as well. Such a paradigmatic holism was first formulated by Davidson (1967).

OpenCCG is an engine for analysing texts with CCGs (Steedman and Baldridge, 2011; Bozşahin et al., 2005). It classifies word forms into categories according to their affordances of combining with other word forms and structures in the process of building up larger structures and construing meaning. In this process, the empty slots of semantic frames, associated with a word, are filled up by the semantic values of the structures that the word form combines with.

Only complete symbolic and semantic structures that represent the whole text are kept by the text analyser (although incomplete structures may also be retrieved for online text processing).

There are two kinds of combinatory categories: the complete (atomic) does not combine with any other structure; the incomplete (complex) has either a frame with empty slots or is missing word parts, so it combines with other structures for semantic or symbolic completion.

Incomplete categories of symbolic structures are turned into a complete category by the OpenCCG engine whenever a structure that is combinable with a preceding or following structure of a certain kind is preceded or followed by a structure of this kind. Slashes \, | and / indicate that a structure of a combinatory category is combinable with a structure that respectively precedes it, is adjacent to it, or follows it. For instance, the structure of *saw* holding a two-slot frame in the clause *Mary saw John* can be said to belong to the category Clause\Mention/Mention, because it expects a complete mention (Mention) of the sensed thing after it and a complete mention (Mention) of the senser before it. The resulting structure after combination is a complete clause (Clause).

In addition, slashes come in four different generalities: they may allow no composition ($\cdot$), only harmonic compositions ($\diamond$), only crossing compositions ($\times$) or any composition ($\bullet$).

The smallest structures in OpenCCG are word forms. Word forms (morph entries) map a token pattern (word) to a word id (stem), a combinatory category tag (pos), the meaning of the word (class), and a list of form classes (fs-macros) and slot fillers (lf-macros). This terminal mapping is equivalent to the map from grammatical functions to lexical and semantical structures in KPML.

2.3 KPML-Analysis and CCG-Synthesis

Kay (1979; 1985) developed the Functional Unification Grammar (FUG) and Kasper (1988) used FUG for exploring text analysis with Nigel. Analysing a clause took about 1 minute (currently approx. 500ms assuming 120-times faster processors) and analysing a complex clause took several minutes. Kasper concluded grammars needed to be "tuned" and augmented for the inverse task, but also that some information would be superfluous and counterproductive for either text synthesis or text analysis. Following Kasper, O'Donnell (1994) reduced the descriptive complexity of Nigel to create a text analyser. After this, Henschel (1995; 1997) attempted to analyse text with the full Nigel grammatical description again by abstracting an open-world typology from a systemic network and compiling the Nigel resource completely for the first time into a typed-feature-structure resource. However, the resource was unusable for practical text analysis. When reviewing these previous attempts, Bateman (2008) pointed out that the conception of systemic-functional resources alone, as it is, cannot support effective automatic text analysis due to fundamental theoretical concerns. That is, the paradigmatic organisation of the systemic-functional approach raises an enormous search space problem when used for text analysis because the network does not have information about which grammatical feature is relevant for any given text token. If one uses such a network for analysing text, one needs to produce a complete set of all possible intersections of grammatical features in order to predict all supported analyses, which is the solution provided by Kasper and by Henschel. Bateman shows that this is computationally intractable for the full version of Nigel's noun group and Nigel's clause.

On the CCG side, broad-coverage surface realisation has also been attempted (White et al., 2007; Rudnick, 2010). In order for a CCG to work for text synthesis, it was enriched with a customised semantics. The resulting search space was still too large and, for this reason, a search heuristic was applied using n-grams, pos-tags, supertags, and semantic values for evaluation of paths. The realisation achieved promising scores with a time-limit of 15 seconds when trained over the CCGBank – a derivation corpus with the same sentences as the Penn TreeBank – and tested over the same sentences.

However, the decision of synthesising a text with a search heuristic is a consequence of the fact that the used resource does not hold all the information necessary for a guided search algorithm for text synthesis. The reason for this is also of a theoretical nature. Once structural information is embedded in word forms, combinatory categories and type changes, it is impossible to take this information back out of them and repack it in a network of options without counting on two essential constructs for synthesis: on the one side, semantic composition and semantic paradigms and, on the other side, a paradigmatic organisation of classes of structure provided by disjunct unions of structure classes (systems). CCGs do not and could not, for efficiency reasons, rely on logical disjunctions, which are essential for text synthesis.

To make the consequences of this limitation more clear, let us take an example of how

combinatory operators are declared in resources for OpenCCG (abbreviations: C = Clause, M = Mention, f = Figure, e = Element):

– danced (*I danced*)
– stopped dancing (*I stopped dancing*)
– started dancing (*I started dancing*)
(C[mode-2]:f\M:e0) => (C[mode-ø]:f\M:e0)
@f<hasTense>e1:Past

– am here (*I am here*)
(C[mode-1]:f\M:e0) => (C[mode-ø]:f\M:e0)
@f:State(<hasTense>e1:Present)

– am (*I am dancing*)
am := (C[mode-ø]:f\M:e0)/(C[mode-6]:f\M:e0)
@f:Change(<hasTense>e1:Present)

– will (*I will dance*)
will := (C[mode-ø]:f\M:e0)/(C[mode-4]:f\M:e0)
@f<hasTense>e1:Future

– stopped (*I stopped dancing*)
stopped := (C[mode-2]:f\M:e0)/(C[mode-6]:f\M:e0)
@f<hasPhase>e1:Stop

– started (*I started dancing*)
started := (C[mode-2]:f\M:e0)/(C[mode-6]:f\M:e0)
@f<hasPhase>e1:Start

Simplified extract of our CCG-resource

The above combinatory categories and type-changes cover different semantic contributions, which are not automatically organisable into systems of symbolic and semantic classes. First, the resource for OpenCCG does not have the information that Past, Present, and Future constitute a semantic disjunction of TENSE and that Start and Stop belong to a distinct semantic disjunction of PHASE. Moreover, the resource does not have the information that finite clauses have tense and that non-finite clauses do not, so that it could decide which system to traverse for each kind of clause. And, finally, we cannot guarantee that an inspection of the figure type happens before the selection of 1) the present auxiliary *am* in *I am dancing* representing a change in the present and 2) the present form *am* of the process of Being in *I am here* (instead *I am being here*) representing a present state. This incapability of grouping contrasting options and of conditioning and ordering systems within a network demands a search algorithm with backtracking. Because of the computational costs of backtracking, it also demands a search heuristic as engineering solution.

3 Symbolic Map

We acknowledge the unsuitability of task-oriented resources for the inverse tasks of synthesis and analysis and shall tackle the issues of bridging a paradigmatic text synthesis and a syntagmatic text analysis at a theoretical level. We propose to describe a symbolic-semantic map that can be compiled into task-oriented resources for separate engines (concretely here: KPML and OpenCCG): a scheme that falls into the Reusability Scheme A (reversibility type) of Klarner (2005) (see Figure 1). In our case, this reusability scheme applies to both grammar and lexicon.

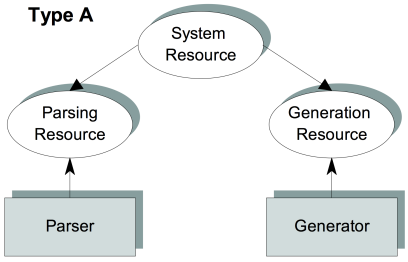

Figure 1. Reusability Scheme

In our argumentation, we shall propose a reformulation of Nigel as a description in OWL of a symbolic map that supports the proposed compilation. Moving from specific to general, we shall point out which mapping strategies can be used and show that every descriptive region of Nigel is representable in such a map.

3.1 Sketch

Bateman (2008) has sketched how an automatic text analysis with systemic-functional theory ('systemic parse') needs to look. It needs a functional description for sequences of text tokens, including the necessary information both for assigning grammatical features to structures and for identifying composites on a sequence of constituents. Such a symbolic map, we shall see, needs to account for the systemic-functional trinocular view of symbolic systems: from above, from below and from around. Moreover, it also needs to account for two different affordances required for a classification of structures: one that organises disjoint classes as a system of grammatical features for text synthesis and another that organises the same disjunctions as restrictions for the combination of incomplete structures in text analysis. The former organisation moves all information of structure into the grammatical network, whereas the latter organisation moves it into word forms.

In the reusability scheme of our new resource, we keep the structural information out of the systemic network and out of the word forms. It is stored in a symbolic map that allows us to pack it into the two task-oriented resources, i.e. into the systemic network for text synthesis and into word forms/type changes for text analysis. We have chosen to represent this

information in Description Logic (OWL-DL) since both the typology embedded in a systemic network for KPML and the typology of features in the types file for OpenCCG can be derived from such descriptions.

3.2 Trinocular View

When classifying symbolic units, we not only conceive of them as patterns for recognition and for expression (from below), but also as bricks for building up a whole with given parts and for selecting parts for a planned whole (from around), and also as devices for construing meaning and for realising it (from above). Therefore, all classes of symbols in our symbolic map will be defined based on their affordances as patterns, bricks, and devices. So our approach is different from that of Henschel (1995; 1997) not only in the fact that we will not extract a typology in description logic from a systemic network (in fact, we will do the opposite), but also in the fact that each structure will be specified in our description as three particulars: one classified from above, one from around, and one from below. The classification from above is convertible into KPML-inquiries, the classification from around is convertible to preselectable grammatical features in KPML, and the classification from below is related to groups of realisation statements in KPML. The definitions of brick classes and of pattern classes are responsible for their functions as meaning-making devices (see Figure 2). In the following, we discuss how these concepts are operationalised for CCG.

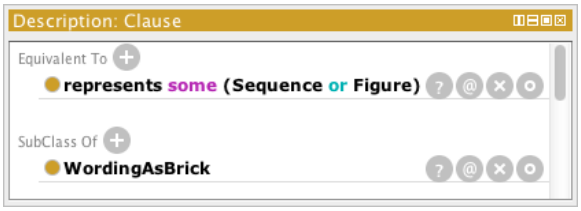

Figure 2. Description of Clause in Protégé

3.3 Word vs Form vs Copy

Moving bottom up in the creation of a descriptive theory, we define a token as a segment of text that matches a continuous pattern (of phonemes or graphemes) suitable for both recognition or expression.

Looking from above, a choice of tokens in a token sequence such as *helped…out* in *he helped me out* represents one single semantic value and is here understood as corresponding to a single word, namely HelpOut.

Looking from around, a word form – that of which a text token is a copy – is defined as composing a particular word and belonging to a particular form class. For the word HelpOut, there are two tokens and therefore two forms, one of them being that of *helped* and the other one being that of *out* in *he helped me out*.

3.4 Pattern vs Brick vs Device

At the leaves of the semantic dependency structure are the semantic values of words and at the corresponding leaves of the symbolic structure are not words, but the forms of words. In this sense, a particular word form is related to three notions: 1) a particular pattern that is used for recognising and producing tokens (copy), 2) a device for realising and construing meaning (word), and 3) a brick for construing larger symbolic structures (form). At this point, we have a triplicity of composition. While a brick is part of a larger symbolic structure, its semantic value is part of a larger semantic structure and its physical pattern is recognisable or produceable in a larger text. In KPML, a semantic structure is specified externally and the correspondence between symbolic and semantic compositionality is guaranteed by the restriction of attaching either the same semantic structure or parts of it to the parts of its corresponding symbolic structure. In OpenCCG, the same compositionality is guaranteed by applying the λ-function of an incomplete constituent to the values of complete constituents (category application) or by composing the λ-functions of two chained incomplete constituents (combinatory rules).

Moreover, symbolic compositionality is linear in nature. As reflected in both KPML and OpenCCG, the position of symbolic constituents may be fixed in relation to other constituents while the semantic constituents cannot. For instance, the position of *nice* in relation to *day* in *have a nice day* is typically realised with the operation “order Epithet:nice Classifier:day” in KPML while it is embedded in the word categories “nice:Classifier/Classifier” and “day:Classifier” in OpenCCG.

4 Target: Nigel coverage

We shall propose a map for every structure class covered by Nigel by tackling the theoretical issues.

4.1 Terms

In Nigel, form classes are used for selecting particular forms of a word while in a CCG they are used for limiting the applicability of the category of a matched token. Usually the form

selection and its applicability are related to either role or agreement restrictions.

In the Nigel grammar, some word classes are defined for automatically creating tokens from the stem of a word such as the verb classes "es-ed" for verbs such as *wish* (*wishes*, *wished*). For indicating the existence of irregular patterns, there are word classes such as "irr". There are also word classes which are used for controlling the selection of a token index based on a set of form classes (or inflectional features) such as "inflectable", "noun", and "verb". All of these word classes together belong to morphology because they are meant to guide the selection of token models and their modifications into the patterns to print out or recognise. With such classes, Nigel is able to reduce the description of a word to a short code such as the following:

```
<Word id="Arrive">
  <Class name="Process" />
  <Class name="EndingWith-e-es-ed-ing" />
  <SampleMap>
    <Sample name="stem" value="arrive" />
  </SampleMap>
</Word>
```

Sample 1. Word Arrive

We store the classes of copies, forms, and words in a lexical ontology together with their relations. In this way, we are able to generate the same systemic network for the rank of word in KPML and, at the same time, all word forms and word form classes for OpenCCG.

In addition, there are word classes used as criteria for selecting words in Nigel. These are the grammatical – or closed-class – words. For them, there is a number of different selecting criteria which are better explained at the ranks where these selections are made (clause, phrase, or group).

In CCG, the morphological entries are not words, but forms. As in KPML, forms have a word identifier (stem), inflectional/agreement classes (macros), they have an attribute for form applicability (pos) and may have an additional semantic value in case of lexical words (class). Therefore, the word exemplified in Sample 1, can be compiled via ontological reasoning into the following structure:

```
<entry word="arrive"   stem="Arrive" class="Arrive"
     macros="@mode-1 @mention-1 @base @Arrive" />
<entry word="arrives"  stem="Arrive" class="Arrive"
     macros="@mode-1 @mention-2 @base @Arrive" />
<entry word="arrive"   stem="Arrive" class="Arrive"
     macros="@mode-1 @mention-3 @base @Arrive" />
<entry word="arrived"  stem="Arrive" class="Arrive"
     macros="@mode-2 @mention-1 @base @Arrive" />
[…]
```

Sample 2. Forms of Word Arrive in CCG

A sample word ontology and the java code for generating resources can be found at https://github.com/DanielCoutoVale/SymbolicMap.

4.2 Composites

In the beginning of every traversal of the systemic network, a symbolic structure is classified either as a clause, a group or phrase, a word or a morpheme. By listing all preselectable classes, we came to the conclusion that there is a fine-grained rank region that includes not only clauses, phrases, groups, and words, but subtypes of these. Clauses are either complexes or simplexes, either dependent or independent. Phrases and groups can be either a nominal group, a quantity group, a quality group, an adverbial group or a prepositional phrase. These subtypes can have further specifications that we shall call here, for simplification, clause mode and noun group case. At this point, below the preselectable classes, it is possible to propose a composite structure whose further specification is exclusively semantical and lexical in nature and whose fitting is governed exclusively by the compositionality of the semantic structure. This possibility was also noticed by Henschel in her final remarks (Henschel, 1997).

At this point of the traversal, for each particular class of structure, there is a semantic correspondent. *Sequences* are realised by clause complexes, *Figures* by clause simplexes, *Elements* by phrases and groups. Subtypes of *Elements* are realised by subtypes of phrases and groups: *Circumstances* by prepositional phrases and adverbial groups, *Things* by noun groups, *Qualities* by adjectival groups, *Quantities* by quantity groups. Finally, *Elements* have two other subtypes: *Processes* and *Modalities*, which occupy respectively the heads of clauses and phrases (see Figure 3).

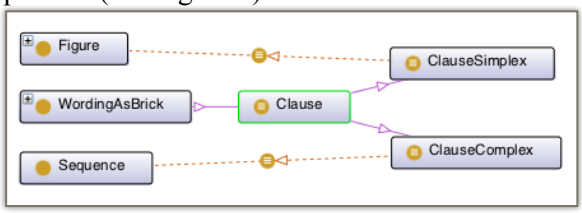

Figure 3. OntoGraf of Clause in Protégé

Since we need to allow changing the type of complete structures into combinable constituents of larger structures during text analysis, a description of symbolic systems must store more information than Nigel at this point. How these type-changes are achieved shall be explained in the following.

4.3 Adjuncts

Type changing in OpenCCG provides a way to implement the separation between pattern, brick, and device. For instance, this operator allows us to create simple rules for *very*

(Qualification/Qualifier) and *nice* (Qualifier) to result in the complete structure of *very nice* (Qualification). Then, by adding the possibility of changing the type Qualification into Classifier/Classifier, we are able to turn the category of this symbolic structure into an adjunct for the classifier *wine* (Classifier) in *this is a very nice wine*. At the same time, we still allow it to be a complement of the process *is* (Clause\Mention/Qualification) in *this wine is very nice*.

4.4 Grammatical Word Selection

In addition to the currently defined semantic elements, the semantic specification of Nigel also contains properties for answering semantic queries. These semantic queries embed a typology of *deictics*, of *tense*, and of *phase* inside of the systemic network. In order to embed these typologies into word forms, the semantic types must be moved to the semantic ontology. For instance, the meaning of *the* in KPML is that it is a 'nonselective', 'nontypic', 'nominal', and 'specific' instantiation of a 'class' of 'non-interactants'. The grammatical feature of *the* is a subtype of all these other features. During text analysis, *the* can be assigned the corresponding grammatical feature while the supertypes of this feature can be inferred with an ontology after the analysis.

4.5 Specification

In Nigel, there are systems whose features are realised either by selecting a unit/form class or by creating a head/tail structure. Tense is an example of this. On the one hand, positive future is realised by adding *will* (T0-head) and selecting the infinitive form for the head of the remaining verbal group (T0-tail) such as *make* in *it will make sense*. On the other hand, positive present is realised by selecting the present form for the head of the verbal group (T0-atom) such as *makes* in *it makes sense*. Therefore, for each region in each rank that creates a specification of clauses, phrases, or groups, we need to have either a head-tail structure or an atom for KPML. The respective corresponding structures for OpenCCG would be an incomplete category or a type-change.

4.6 Complements

Circumstance complements such as *of Mary* in *in front of Mary* create no new challenges for description. Figure complements, on the other hand, do. The clause, as the representation of a figure (state or event), is a symbolic structure whose constituents represent the elements of a semantic figure. Nigel adds the representative functions of clause constituents in the traversal of a figure typology. Each level of the typology decides whether a semantic role is present or not in the figure and therefore if a constituent must have the function of such a role. Roles include those of actor, actee, senser, sensum, sayer, target, verbiage, carrier, attribute, identified, identifier among others. Semantic roles and their presence for a given figure type are stored outside the system in a separate typology (GUM-3) (Bateman et al., 2010). The correspondent of the transitivity region in OpenCCG would be the mapping of logical variables to the diamond modes of a figure node as specified in the XML below:

```
<satop nomvar="SimpleAction">
     <diamond mode="hasProcess">
          <nomvar name="Process"/>
     </diamond>
     <diamond mode="hasActor">
          <nomvar name="Actor"/>
     </diamond>
</satop>
```

Sample 3. Logical Form in OpenCCG

Which process words can be used in each figure type need not be defined in KPML because both the figure type (SimpleAction, AffectingAction, etc.) and the process type (Running, Jumping, Singing, Seeing, etc.) are defined in the semantic specification that is passed to KPML as an input for text synthesis. This mapping from process types to figure types is necessary in OpenCCG and, therefore, process words need to be assigned process types so that SimpleActionProcesses are associated with a derivation family that has a medium/ actor, and so that AffectingActionProcesses are associated with a derivation family that has an agent/actor and a medium/goal, and so on. The whole set of rules involving transitivity can be automatically derived from a typology of figures both for KPML and for OpenCCG that includes such process classes.

In addition, in KPML, voice is implemented by mapping the transitive functions described above (actor, actee, recipient, senser, sensum, sayer, target...) to a smaller set of ergative functions (agent, medium, beneficiary). For example, the clause *the duke gave my aunt the teapot* has *the duke* as actor, *my aunt* as recipient and *the teapot* as goal in the transitive structure and *the duke* as agent, *my aunt* as beneficiary, and *the teapot* as medium in the ergative structure. For each figure type, there is a mapping of specific transitive functions to ergative functions. The ergative functions are the ones that get mapped to the subject, the direct(-object) and the indirect(-object) functions

depending on the voice (agent-receptive voice, medium-receptive voice, and beneficiary-receptive voice). To implement a similar voice construct in OpenCCG, we propose a strategy of moving the mapping of transitive-ergative functions to a secondary step of reasoning after text analysis (example in https://github.com/DanielCoutoVale/SymbolicMap). After doing this, the actual voice structure can be implemented with categories such as Clause \Mention/Mention/Mention/Process for the auxiliary word *was* in *the teapot was given by the duke to my aunt* and with the type changing rule Process –> Clause\Mention/Mention/Mention applied to the Process *gave* in *the duke gave my aunt a teapot*. Which category or rule to apply depends on the ergative functions of each figure type – e.g. figures with a medium and no agent do not have a "passive" form. Culmination – the choice between *the teapot was given my aunt by the duke* and *the teapot was given by the duke to my aunt* – was realised in OpenCCG together with voice.

5 Evaluation

For evaluation, we targeted the only real challenge in the relation between Nigel and our CCG (clause complements) by creating a simple symbolic map with two ranks (clause and mention), with three figure types, three voices and two culminations (complements). This resource was compiled into a systemic network and into combinatory categories and type change rules successfully. Text synthesis and text analysis work as intended, that is, algorithmically without backtracking. For instance, the automatically generated CCG gives the correct standard analysis for *the duke gave my aunt the teapot* according to SFG as seen in the Table 1:

the duke	***gave***	***my aunt***	***the teapot***
Actor	Process	Recipient	Goal
Agent		Beneficiary	Medium

Table 1. Analysis for *the duke* as subject

For utterances such as *my aunt was given the teapot by the duke* see Tables 2-3, two hypotheses of analysis are given by CCG. Both analyses are correct if only symbolic and semantic compositionality is taken into account. An analysis, according to which the teapot receives someone's aunt (Table 3), can only be discarded when knowledge about the world (and not about language) is applied.

my aunt	***was***	***given***	***the teapot***	***by the duke***
Recipient		Process	Goal	Actor
Beneficiary			Medium	Agent

Table 2. Analysis 1 for *my aunt* as subject

my aunt	***was***	***given***	***the teapot***	***by the duke***
Goal		Process	Recipient	Actor
Medium			Beneficiary	Agent

Table 3. Analysis 2 for *my aunt* as subject

The compilation speed for OpenCCG word forms is very slow: one second per word form on a computer with 2.6 Ghz processor. The compilation of OpenCCG combinatory categories, type changing rules, KPML lexicon and network, on the other hand, is efficient. Once compiled, the speed of text analysis is that of a regular hand-written resource for OpenCCG and is equivalent in size and quality through code inspection.

6 Conclusion

In this paper, we have shown that task-oriented resources for KPML and OpenCCG do not contain the necessary information for doing the inverse task and that their directed rules cannot encode the information that is necessary for the resources to become reversible.

Therefore, we have adopted a third strategy of creating a completely descriptive map between symbolic and semantic structures that can be compiled into a systemic network and into combinatory categories and type changing rules.

Our evaluation has shown that the approach is sound and is able to solve previously identified issues on a theoretical level. However, we are still unsure about the amount of engineering resources that would be needed in order to complete the same coverage of Nigel within such a paradigm. Nevertheless, from the pilot study undertaken, the approach appears promising.

Acknowledgements

We gratefully acknowledge the support of the Deutsche Forschungsgemeinschaft (DFG) through the Collaborative Research Center SFB/TR8 Spatial Cognition.

References

John A. Bateman. 1995a. Basic technology for multilingual theory and practise: the KPML development enviroment. In *Proceedings of the Workshop on Multilingual Text Generation IJCAI-95*, pages 1–12. Montreal.

John A. Bateman. 1995b. KPML: the KOMET-Penman Multilingual Linguistic Resource Development Environment. In *Proceedings of the Fifth European Workshop on Natural Language Generation*, pages 219–222. Leiden.

John A. Bateman. 1996. *KPML Development Environment*. GMD-Forschungszentrum Informationstechnik GmbH, Sankt Augustin.

John A. Bateman. 1997. Enabling technology for multilingual natural language generation: the KPML development environment. *Natural Language Engineering*, 3(1):15–55.

John A. Bateman. 2008. Systemic-functional linguistics and the notion of linguistic structure: unanswered questions, new possibilities. *Meaning in context implementing intelligent applications of language studies*, pages 24–58. Continuum, London/New York.

John A. Bateman, Joana Hois, Robert Ross, and Thora Tenbrink. 2010. A linguistic ontology of space for natural language processing. *Artificial Intelligence*, 174(14):1027–1071.

Cem Bozşahin, Geert-Jan Kruijff, and Michael White. 2005. *Specifying Grammars for OpenCCG: A Rough Guide*. Retrieved from http://www.metu.edu.tr/~bozsahin/nli/ceng563/link/grammars-rough-guide.pdf

Donald Davidson. 1967. Truth and Meaning. *Synthese*, 17(1):304–323.

Friedrich Ludwig Gottlob Frege. 1884. *Grundlagen der Arithmetik: eine logisch mathematische Untersuchung über den Begriff der Zahl*. Wilhelm Köbner, Breslau.

Michael A.K. Halliday and Christian M.I.M. Matthiessen. 2014. *Halliday's Introduction to Functional Grammar*, 4th edition. Routledge, London/New York.

Renate Henschel. 1995. Traversing the Labyrinth of Feature Logics for a Declarative Implementation of Large Scale Systemic Grammars. Retrieved from http://www.elsnet.org/publications/clnlp95

Renate Henschel. 1997. Compiling Systemic Grammar into Feature Logic Systems. Retrieved from http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.52.3153&rep=rep1&type=pdf

Robert T. Kasper. 1988. An experimental parser for systemic grammars. In *Proceedings of the Twelfth International Conference on Computational Linguistics*, pages 309–312, Budapest.

Martin Kay. 1979. Functional Grammar. In *Proceedings of the Berkeley Linguistics Society*. pages 142–158, Berkeley.

JP Martin Kay. 1985. Parsing in functional unification grammar. *Natural Language Parsing*. Cambridge University Press, Cambridge, UK.

Martin Klarner. 2005. Reversibility and re-usability of resources in NLG and natural language dialog systems. In *Proceedings of the Tenth European Workshop on Natural Language Generation*, pages 185-190. Aberdeen.

George Lakoff. 1987. *Women, fire and dangerous things: what categories reveal about the mind*. University of Chicago Press, Chicago.

Christian M.I.M. Matthiessen. 1995. *Lexicogrammatical cartography: english systems*. International Language Sciences Publishers, Tokyo.

Christian M.I.M. Matthiessen and Michael A.K. Halliday. 1999. *Construing experience through meaning: a language-based approach to cognition*. Continuum, London/New York.

Christian M.I.M. Matthiessen and Michael A.K. Halliday. 2004. *An introduction to functional grammar*, 3rd edition. Oxford University Press, New York.

Günter Neumann. 1991. A bidirectional model for natural language processing. In *Proceedings of the Fifth Conference on European chapter of the Association for Computational Linguistics*, pages 245–250, Berlin.

Günter Neumann and Gertjan van Noord. 1992. Self-monitoring with reversible grammars. In *Proceedings of the Fourteenth International Conference on Computational Linguistics*, pages 700–706, Nantes.

Michael O'Donnell. 1994. *Sentence analysis and generation: a systemic perspective*. Ph.D. thesis, University of Sydney, Sydney.

Stephen G. Pulman. 1995. Review of Reversible Grammar in Natural Language Processing. In *Computational Linguistics*, 21:269–271.

Alex Rudnick. 2010. *Review: realization with CCG*. Retrieved from http://www.cs.indiana.edu/~alexr/nonpubs/alexr-ccg-generation.pdf

Barry Smith and Berit Brogaard. 2003. A Unified Theory of Truth and Reference. *Logique et Analyse,* 43:1–46.

Mark Steedman. 1987. Combinatory grammars and parasitic gaps. *Natural Language and Linguistic Theory*, 5(3):403–439.

Mark Steedman. 1996. *A very short introduction to CCG*. Retrieved from http://www.inf.ed.ac.uk/teaching/courses/nlg/readings/ccgintro.pdf

Mark Steedman. 1998. Categorial Grammar. *The MIT Encyclopedia of Cognitive Sciences*, pages 1–9. MIT Press, Cambridge, MA.

Mark Steedman and Jason Baldridge. 2011. Combinatory Categorial Grammar. *Non-Transformational Syntax*, pages 181–224. Blackwell, Oxford, UK.

Tomek Strzalkowski. 1994. Reversible Grammar. *Reversible Grammar in Natural Language Processing*, pages xiii–xxi. Springer Science +Business Media, Dordrecht.

Michael White, Rajakrishnan Rajkumar, and Scott Martin. 2007. Towards broad coverage surface realization with CCG. In *Proceedings of the Workshop on Using Corpora for NLG Language Generation and Machine Translation*, pages 22-30, Brighton.

Lugwig Josef Johann Wittgenstein. 1921. Logisch-Philosophische Abhandlung. *Annalen der Naturphilosophie*, 14:185-262.

Lugwig Josef Johann Wittgenstein. 1922. *Tractatus Logico-Philosophicus*. Kegan Paul, Trench Trubner & Co, London.