Pre-Processing MRSes

Tore Bruland
Norwegian University of Science and Technology
Department of Computer and Information Science
torebrul@idi.ntnu.no

Abstract

We are in the process of creating a pipeline for our HPSG grammar for Norwegian (NorSource). NorSource uses the meaning representation Minimal Recursion Semantics (MRS). We present a step for validating an MRS and a step for pre-processing an MRS. The pre-processing step connects our MRS elements to a domain ontology and it can create additional states and roles. The pipeline can be reused by other grammars from the Delph-In network.

1 Introduction

NorSource[1] (Beermann and Hellan, 2004; Hellan and Beermann, 2005), a grammar for Norwegian, is a Head-Driven Phrase Structure Grammar (HPSG) (Sag et al., 2003), developed and maintained with the Linguistic Knowledge Builder (LKB) tool (Copestake, 2002), and originally based on the HPSG Grammar Matrix, which is a starter kit for developing HPSG grammars (Bender et al., 2002). An HPSG grammar can use Minimal Recursion Semantics (MRS) as meaning representation (Copestake et al., 2005). In order to speed up the parsing process (the unification algorithm), a HPSG grammar can be compiled and run (parsing) with the PET[2] tool (Callmeier, 2001). The Flop program in PET compiles the LKB grammar and the Cheap program runs it. An alternative to the PET system is the Answer Constraint Engine (ACE)[3] created by Woodley Packard. ACE can parse and generate using the compiled grammar.

Our goal is to create a pipeline for the NorSource grammar and use it to create small question-answer systems or dialogue systems. The first step in the pipeline is the parsing process with ACE. The next step is to select the most suitable MRS. We use Velldal's ranking model (Velldal, 2008). The model is based on relevant sentences from our system, treebanked with [tsdb++] (Oepen et al., 2002; Oepen and Flickinger, 1998). The selected MRS is checked with the Swiss Army Knife of Underspesification (Utool) (Koller and Thater, 2006b) and our own validating procedure. Only well-formed MRSes are used in our pipeline. We also use Utool to solve the MRS and to eliminate any logically equivalent readings. The next step is to pre-process the MRS (calculate event structure and generate roles), and the last step in our pipeline creates a First-Order Logic formula from the MRS (only the easy cases).

Our contribution is the validating step and the pre-processing step.

In the next section, we give a brief introduction to MRS. Then we present details from our validating procedure. Next, we solve an MRS and eliminate logically equivalent readings with Utool. We pre-process the selected MRS in section 6. At last, we look at a way to create a First-Order Logic formula from a solved MRS and we present a few challenges from our research.

[1] http://typecraft.org/tc2wiki/Norwegian_HPSG_grammar_NorSource
[2] http://pet.opendfki.de/
[3] http://moin.delph-in.net/AceTop

2 Minimal Recursion Semantics

The elements of an MRS can be defined by the structure mrs(T, I, R, C), where *T* is the top handle, *I* is the index, *R* is a bag of elementary predictions (EP), and *C* is a bag of constraints.

Every dog chases some white cat (1)

(1) can have the following MRS (created for demonstration purposes):

T h_0,

I e_1,

R { h_1:every(x_1,h_3,h_8), h_3:dog(x_1), h_7:white(x_2), h_7:cat(x_2), h_5:some(x_2,h_{10},h_9), h_4:chase(e_1,x_1,x_2) }

C $h_{10} =_q h_7$

An MRS can be in two states: unsolved or solved. An algorithm (LKB and Utool) brings an MRS from the unsolved state into one or more solved states. An unsolved MRS has holes that are not in the set of labels, and a solved MRS has holes that are from the set of labels. The set of labels in our example: $\{h_1$, h_3, h_4, h_5 and $h_7\}$. The set of holes: $\{h_3$, h_8, h_9 and $h_{10}\}$. A hole can be either open or closed. A hole is open when it isn't in the set of labels, and a hole is closed when the hole is in the set of labels. Hole h_3 is closed in our example.

3 Validate MRSes From NorSource

We want to search our MRSes for properties that can lead to problems. The Utool solvable function checks if an unsolved MRS can be transformed into one or more solved MRSes without violating the MRS definitions.[4] Our validating procedure contains a set of functions. We create variables or list of variables for each function that is positive. The functions are: *empty index*, *empty feature*, *empty reference*, *key conjunction*, and *argument EP conjunction*. An *empty index* exist when the index value refers to a variable that is not an EP's arg_0. An *empty feature* exist when a feature value refers to a variable that is not found in the EP's arguments. An *empty reference* exist when an argument refers to a variable that is not an EP's arg_0 and the variable is not in the set of feature values. A *key conjunction* exist when more than one EP in an MRS have the same arg_0 and they are not quantifiers. An *argument EP conjunction* exists when an argument contains a label that is an EP conjunction. In Table 1, the variable

EP	Feature
h_3:pred$_1$($arg_0(e_1)$,$arg_1(h_9)$)	e_1,feature$_1$,value$_1$
h_9:pred$_2$($arg_0(u_1)$,$arg_1(x_1)$,$arg_2(u_{10})$)	u_{12},feature$_2$,value$_2$
h_9:pred$_3$($arg_0(u_1)$,$arg_1(x_1)$,$arg_2(x_2)$,$arg_3(u_{15})$)	u_{15},feature$_3$,value$_3$
h_2:pred$_4$($arg_0(x_1)$)	u_{16},feature$_4$,value$_4$
h_4:pred$_5$($arg_0(x_2)$)	

Table 1: Eps and Features

u_{12} is an empty feature. The variable u_{10} is a empty reference. The arg$_0$ of pred$_2$ and pred$_3$ form a key conjunction. The argument h_9 in arg$_1$ of pred$_1$ is an argument EP conjunction.

[4]Utool is stricter than the LKB software, see Fuchss et al. (2006).

4 Selecting An MRS

Before we create a ranking model, we analyze and compare the MRSes from our domain.[5] We use the variable-free solution Oepen and Lønning introduced (Oepen and Lønning, 2006). We compare parts from the syntax tree, the EPs, the features, and if the MRS is solvable or not. We also present results from our validation procedure. If we parse (2) with NorSource, it yields 9 MRSes. Parts of the EP information is presented in Table 2.

	1	2	3	4	5	6	7	8	9
legge_v ARG1 addressee-rel	x	x	x	x	x	x	x	x	x
legge_v ARG2 bok_n	x	x	x	x	x	x	x	x	x
legge_v ARGX på_p		x	x						
på_p ARG1 bok_n	x	x	x			x	x	x	x
på_p ARG1 legge_v				x	x				
på_p ARG2 bord-1_n	x	x	x	x	x	x	x	x	x

Table 2: Compare MRSes

Legg boken på bordet
Put the book on the table (2)

We use the LOGON software to treebank ([tsdb++]) and to create our ranking model (we have copied adjusted the scripts in folder lingo/redwoods). We parse with the ranking model and we select the first MRS.

5 Solving The MRS

Utool solves an MRS using a dominance graph and a chart (Niehren and Thater, 2003). The redundancy elimination algorithm (Koller and Thater, 2006a) takes a chart and a redundancy elimination file as input and returning the chart without the redundancy. We have created a redundancy elimination file according to (Koller and Thater, 2006b) for our quantifiers.

6 Pre-Processing The Selected MRS

In the pre-processing step we focus on the event structure and roles. By event structure we mean: sub events, aspectual and causal notions. Vendler grouped verbs into classes based on their temporal properties (Vendler, 1967). The verbs are classified according to duration and presence of a terminal point. A verb with a terminal point is called telic (the verb culminates). Vendler's classes are also known by other terms such as: eventualities, situations, lexical aspect or Aktionsart. The classes are: state, point, process, achievement, and accomplishment. The verb's connection to a class is not static, because a verb argument can move an event from one class into another. This phenomenon is called aspectual composition or coercion (Moens and Steedman, 1988).

A predicate string in an EP can contain the prefix "_", the suffix "_rel", a name, a part-of-speech type and a sense number. The name, the part-of-speech type and the sense number can be connected to a domain ontology definition. If we don't have the sense number, we can have a list of domain ontology definition candidates. A predicate string can also be a unique name like in "first_position_prominent".

Our goal with the pre-processing is:

- to connect the names in the predicate strings to a domain ontology
- to check if the predicate and the predicate arguments are valid according to the domain

[5] A demo for NorSource: `http://regdili.idi.ntnu.no:8080/comparemrs/compare`

- to create a common structure for a set of verbs
- implement an algorithm for roles and states

The main elements of our solution are a predicate tree, an algorithm, a domain ontology, and a set of object-oriented classes. The predicate tree is created from the predicates and their arguments.

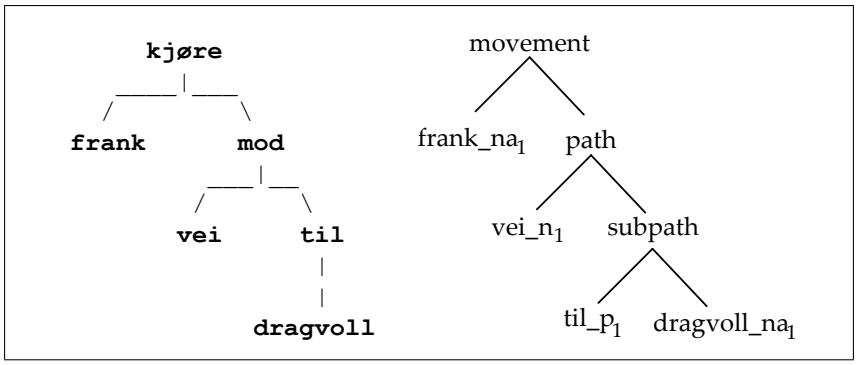

Figure 1: The Predicate Tree and the Movement Class

$$\frac{\text{Frank kjører veien til Dragvoll}}{\text{Frank drives the road to Dragvoll}} \tag{3}$$

The predicate tree on the left in Figure 1 is created from (3). The algorithm searches the predicate tree and for each node in the tree it finds templates from the domain ontology. A template contains checks against the domain ontology, a return function and a return class. One of the templates used in our example is shown in Table 3. The variable X is replaced by vei_n_1 in our example. The final class for

key	nodes	check list	return	class
mod_5	node(n,X)	isa(X,path_n_1)	new	path
	node(subpath,til_p_1)			

Table 3: Template Example

(3) is shown on the right in Figure 1. We have defined three return functions: "new", "call" and "fork". "New" creates the return class from its arguments. "Call" is used when one node consumes another. For example in "VP PP", the PP can be consumed by the VP. "Fork" is used when different classes are connected. For example in "My uncle goes to town" we have a Family class and a Movement class that are connected through the variable for the uncle. The Movement class for (3) contains the following information:

movement
- event(e_1:kjøre_v_1)
- subject(x_2:frank_na_1)
- path
  - object(x_9:vei_n_1)
  - end-point(x_1:dragvoll_na_1,t_2)

checks
- has(dragvoll_na_1,location)
- isa(vei_n_1,path_n_1)

templates
- [tv_2, pp_2, mod_5]

The Movement class is a result from analyzing a number of different sentences about objects changing location. The Movement class is a process and it can contain a path, a departure, an arrival, a road (named path), a vehicle etc. A number of these objects are connected. For example the beginning of the process and the beginning of the path. The beginning of the path and the departure. We have implemented an algorithm for detecting the roles *work* and *cargo*. *Work* indicates an object that is using energy. *Cargo* indicates an object that is being transported.

The classes can be used as they are in a further reasoning process, or they can generate EPs and features that are inserted into the MRS. For example, the state *at_location(*x_4*,*x_1*)* can be inserted in an EP conjunction.

7 Logic Form

The solved MRSes are not yet formulas in First-Order Logic. We have to convert arguments that are in a Higher-Order Logic, insert the *not* operator, and use the quantifiers from First-Order Logic. An argument that has an Ep conjunction or has a handle needs to be rewritten. An argument with a handle is replaced with the arg_0 from the EP of the handle. We can give a reference and a predicate to the conjunction or we can find a candidate in the conjunction. We use the latter method. The predicate *_neg_adv_rel* is converted to the operator *not*. We have connected some of the NorSource quantifiers to the First-Order Logic quantifiers *exist* and *all*. Quantifiers like *some*, *few*, etc can be assigned to a group / type of their own. We place an operator between the arguments "RSTR" and "BODY" in our quantifiers:

$\exists(y)[pred_1(y)$ **operator**$_1$ $\forall(x)[pred_2(x)$ **operator**$_2$ $pred_3(x,y)]]$

We have defined **operator**$_1$ as $\wedge$ and **operator**$_2$ as $\rightarrow$. The formula from (3) is:

$\exists(x_1)[$*na*$(x_1,$*dragvoll*$)\wedge\exists(x_2)[$*na*$(x_2,$*frank*$)\wedge\exists(x_9)[$*vei_n*$(x_9)\wedge$*kjøre_v*$(e_1,x_2,x_9)\wedge$*til_p*$(u_1,x_9,x_1)]]]$

The meaning of the MRS quantifiers are preserved with the extra predicates: q_meaning(u100,x_1,def), q_meaning(u101,x_2,def) and q_meaning(u102,x_9,def). These predicates need to be inserted into the logic formula.

8 Challenges

The first challenge is to find a representative selection of sentences from the domain. The sentences are treebanked and used in our ranking model, and their selected MRSes are also used for creating domain ontology definitions. We use a virtual profile where we can add new annotated profiles.

We need to find a way to process our quantifiers that are not in First-Order Logic. At this early stage we can use off-the-shelf theorem provers and model finders as described in Blackburn's and Bos' reasoning framework for First-Order logic (Blackburn and Bos, 2005). We must classify different kinds of questions and commands, and we need to describe how to process them.

So far, we have connected types for each Ep together, but sometimes a more detailed structure is required in order to express meaning. A part of one structure can connect to a part of another. In the examples: "the cat sits in the car", "the cat sits on the car", and "the cat sits under the car", there are three different locations related to the car. We have a container in the car, an area on top of the car, and a space under the car. This extra information needs to be used together with the MRS. Sometimes more ambiguities are introduced and we need a ranking. For example in "Fred poured coffee on the thermos", the most probable is when the coffee ends up inside the thermos and the less probable version where the coffee is poured on the outside of the thermos. We also want to use previous situations and the discourse so far in our pre-processing step.

References

Beermann, D. and L. Hellan (2004). A treatment of directionals in two implemented hpsg grammars. In S. Müller (Ed.), *Proceedings of the HPSG04 Conference.* CSLI Publications.

Bender, E. M., D. Flickinger, and S. Oepen (2002). The grammar matrix: An open-source starter-kit for the rapid development of cross-linguistically consistent broad-coverage precision grammars. In

J. Carroll, N. Oostdijk, and R. Sutcliffe (Eds.), *Proceedings of the Workshop on Grammar Engineering and Evaluation at the 19th International Conference on Computational Linguistics*, Taipei, Taiwan, pp. 8–14.

Blackburn, P. and J. Bos (2005). *Representation and Inference for Natural Language*. CSLI Publications.

Callmeier, U. (2001). *Efficient Parsing with Large-Scale Unification Grammars*. Master thesis, Universit¨at des Saarlandes.

Copestake, A. (2002). *Implementing Typed Feature Structure Grammars*. CSLI.

Copestake, A., D. Flickinger, C. Pollard, and I. A. Sag (2005). Minimal Recursion Semantics. An introduction. *Journal of Research on Language and Computation 3*(4), 281 – 332.

Fuchss, R., A. Koller, J. Niehren, and S. Thater (2006). Minimal recursion semantics as dominance contraints: Translation, evaluation and analysis. In *Proceedings of the 42nd ACL*. Association for Computational Linguistics.

Hellan, L. and D. Beermann (2005). Classification of prepositional senses for deep grammar applications. In V. Kordoni and A. Villavicencio (Eds.), *Proceedings of the 2nd ACL-Sigsem Workshop on The Linguistic Dimensions of Prepositions and their Use in Computational Linguistics Formalisms and Applications*, Colchester, United Kingdom.

Koller, A. and S. Thater (2006a). An improved redundancy elimination algorithm for underspecified representations. In *Proceedings of the 21st International Conference on Computational Linguistics and the 44th annual meeting of the Association for Computational Linguistics*, pp. 409–416. Association for Computational Linguistics.

Koller, A. and S. Thater (2006b). *Utool: The Swiss Army Knife of Underspesification*. Saarland University, Saarbrücken.

Moens, M. and M. Steedman (1988). Temporal ontology and temporal reference. *Computational Linguistics 14*(2).

Niehren, J. and S. Thater (2003). Bridging the gap between underspesification formalisms: Minimal recursion semantics as dominance constraints. In *41st Meeting of the Association of Computational Lingustics*, pp. 367–374.

Oepen, S. and D. Flickinger (1998). Towards systematic grammar profiling. test suite technology ten years after. *Journal of Computer Speech and Language 12*(4), 411–436.

Oepen, S. and J. Lønning (2006). Discriminant-based mrs banking. In *Proceedings of the 5th International Conference on Language Resources and Evaluation (LREC 2006)*.

Oepen, S., K. Toutanova, S. Shieber, C. Manning, D. Flickinger, and T. Brants (2002). The lingo redwoods treebank: Motivation and preliminary applications. In *Proceedings of the 19th international conference on Computational linguistics-Volume 2*, pp. 1–5. Association for Computational Linguistics.

Sag, I. A., T. Wasow, and E. M. Bender (2003). *Syntactic Theory: a formal introduction* (2. ed.). CSLI Publications.

Velldal, E. (2008). *Empirical realization ranking*. Ph. D. thesis, The University of Oslo.

Vendler, Z. (1967). *Linguistics in Philosophy*. Cornell Univerity Press.