A Type-Theoretic Approach to Generating Pictures and Descriptions

Herbert Lange
Computer Science and Engineering
University of Gothenburg
herbert.lange@cse.gu.se

Abstract

Modern type theories (MTTs) find applications in many areas of natural language semantics and give rise to reliable approaches to natural language inference. In this paper we present an approach to generate pictures and natural language descriptions in parallel based on a common type-theoretic semantic representation. It uses the expressive power of modern type theory to ground objects spatially. This grounded representation can be used to generate both pictorial and textual representations giving rise to a controlled hybrid language.

1 Introduction

Using modern type theories (MTT), also called constructive type theories, for natural language semantics is an active branch of research withing computational linguistics. It has been shown that this kind of type theory is suitable to express interesting phenomena within natural language semantics (Ranta, 1994). We use MTT to ground objects spatially and use this representation to generate both pictures and natural language descriptions in parallel in a controlled setting.

1.1 Modern type theory in natural language semantics

Modern type theories go back to the work of Per Martin-Löf (1972, 1984). His constructive type theory is an important foundation in computer science research on computer-assisted proofs. Related type theories build the basis for proof assistants such as Agda (Norell, 2007) and Coq (Coquand and Huet, 1988). Work on using constructive type theory for natural language semantics ranges back to the work by Sundholm (1989) on generalized quantifiers and Ranta (1994) on many aspects including anaphoric expressions, such as the so-called donkey sentences. Over the years many challenges of natural language semantics have been in focus, leading to systems for reliable natural language inference using proof assistants (Bernardy and Chatzikyriakidis, 2017; Chatzikyriakidis and Bernardy, 2019).

1.2 Modern type theory in natural language syntax

Another application of MTTs within computational linguistics is a syntactic framework called Grammatical Framework (GF) (Ranta, 2009b, 2011). It uses a type-theory based abstract representation of syntactic structures together with language specific rules for *sugaring*, also called *linearization*, to express these abstract representation in concrete (natural) languages. The *sugaring* operation is reversible which means these language expressions can be parsed back into abstract representations. Several concrete languages can share the same set of abstract representations, called an abstract syntax. We will show how spatial relations can be modeled as abstract representation within modern type theory and use the methods of sugaring to translate the abstract representations into both pictures and natural language description of these pictures. The techniques we present here can be implemented in either GF or Agda in a very similar way. As a consequence we can use GF for handling natural language expressions and Agda for reasoning. The implementation in both Agda and GF is available online.[1].

1.3 Related Work

This paper touches on several more or less related directions of research. We present a con-

[1] https://github.com/daherb/mulle-spatial

trolled hybrid language (Haralambous et al., 2017), a unified controlled language to create both natural language expressions and pictorial representations. There has been some similar efforts such as the work by Haralambous et al. (2017) for creating nautical maps and map descriptions in parallel. In general there has been plenty of work on generating natural language descriptions from concrete pictures or abstract picture representations, e.g. (Chen et al., 2019). What sets our work apart is the more unified approach to abstract representation and language generation. Finally, there have even been previous attempt to use related type theories to express spatial relations between objects, e.g. Dobnik and Cooper (2013). However, their focus on robotics and modeling concrete object placement in the real world makes their approach significantly different from our abstract high-level approach.

2 Abstract Representation

One major idea behind the type theoretic formalization of spatial relations is the fact that we can use dependent types, i.e. types that depend on the objects of other types, to model constraints (Ranta, 2011, Chapter 6). In this case the constraints decide the placement of objects in a virtual world. To keep this demonstration simple, we only support two objects in our world at the same time.

2.1 The World

The world we use in our spatial representation can be seen as a simplified world in the style of the SHRDLU block world (Winograd, 1972). We use a two-dimensional grid in which objects can be placed. We use natural numbers to address the cells and make sure that all objects are placed on the grid. We also add constraints on the object placement that are depending on the shape of the objects as well as the relationship between the objects. Following the general concept of modeling constraints as dependent types, we can already on the type level express a constraint that guarantees that two given coordinates are placed inside the grid. This type, `InRange`, is a type depending on four natural numbers. It has one constructor and if we can construct a value with it, we know that coordinates given by the numbers are valid in our world.

2.2 The objects and relations

The set of all objects can be defined by enumerating them. The same is the case for the relations we want to include. We currently include 8 objects and 7 relations. Some of these objects have implicit properties such as having to be placed at the bottom of the grid. Houses and trees for example are supposed to be placed on ground level. These properties will be enforced in a later step. Among the relations we include, some are more general than others. If something is *besides* of something else, it can be either *to the left* or *to the right* and if something is *on top* it can also be considered *above*. The objects themselves are clustered into classes of objects having similar properties. This could be for example all objects that can be put inside other objects (`InsideObject`) or all objects that can contain other objects (`OutsideObject`). These classes are expressed by dependent types depending on the object type.

2.3 Spatial Relations and Constraints

The most interesting part of the abstract representation are the constraints to enforce proper placements of the objects depending on the objects and spatial relation involved. First we have to make sure that the objects can actually be combined in the intended way. We do not really want to put a house *in* the tree, except if we plan to build a tree house which is currently not supported. To enforce this level of semantics we use the object classification described above. Based on the object classes we can define a new type that only guarantees valid combinations. Each constructor combines two object classes with a relation that works between these objects. For example if we have an `InsideObject` and an `OutsideObject` we can use the relation `rin`. Sometimes we want to handle certain objects independently of their class. We can add specific constructors for these cases. For example, we don't want to put people into other objects in general but people in houses are quite common. This case can be handled by a specific constructor. As soon as we have established that the objects can be combined in the way we want to, we have to make sure that their coordinates are valid. We can express these requirement using dependent

types again, specifically the type `ValidPos` depending on the relation and the coordinates. We have one constructor for each of the relations enforcing the specific constraints on the coordinates. For the case of putting an object inside another object, the two coordinates have to be equal. In addition we enforce the fact that most objects cannot float in thin air, so the y coordinate of the container has to be 0 to place it on the ground. The constructors also require an object of `InRange`, the type we described previously that guarantees that the coordinates are valid positions on the grid. For the other relations different constraints on the coordinates are enforced.

2.4 The scene

As a final step we build a scene. It consists of two objects, their position, the spatial relation between the objects and the two proof objects showing that the relation is valid between the objects and the positions are valid according to their spatial relation and relative to the grid. An example scene is shown in Listing 1.[2]

```
example : Scene
example =
  constraintPlace operson ohouse 3 0 3 0
  rin personinhouse
  (validinpos (equal 3) (equal 0)
    (equal 0) (inrange [...]))
```

Listing 1: Example scene

In the example scene we have the two objects `operson` and `ohouse`, the spatial relation `rin`, the components of the coordinates representing $(3, 0)$ twice and finally the objects telling us that we can put a person in the house (`personinhouse`) and that the coordinates work for putting the person in the house at this position on the grid (`(validinpos [...])`) using an object of type `InRange` as described above. In English this scene would be expressed as *the person is in the house*.

3 Sugaring

Sugaring is the translation from the abstract representation into a concrete language following Ranta (1994). These concrete languages most often are natural languages such as English, German or Swedish. But the technique is not limited to this. We show how we can generate both natural language picture descriptions as well as the pictures themselves from the same representation.

3.1 Natural language generation

There has been a long-going effort on how to express an extensive abstract syntax in GF in various natural languages. This effort is called the resource grammar library (RGL) (Ranta, 2009a) which is part of the GF distribution. Compared to the RGL, the natural language fragment we want to express is very limited. However, we use the same methods and data structures that proved effective for modeling syntactic phenomena of natural languages. We can use total functions from grammatical features to implement word inflection and use records to store relevant information. The necessary grammatical features can be defined by listing all their values exhaustively. These techniques are relevant more or less independent of the language but some features might be used more extensively, for example if a language is morphologically rich. We implemented the sugaring for both English and German and demonstrate the main concepts for German. We start the sugaring from objects of the `Scene` type and apply the sugaring functions recursively. We have seen an `example` in Section 2.4. Most of the information stored in this object is not relevant now, we only care about the objects and the relation. The relation can just be represented by a string because prepositions are not inflected. The object is represented as a record because it has beside its string representation also an inherent gender. Besides these two we also need a representation of determiners, i.e. the direct article, in German even though they are not included in the abstract representation. Determiners are inflected by gender and case (see Listing 2).

The construction `\ where` in `linDet` is a combination of lambda abstraction and pattern matching to define the total function mapping from the grammatical features to strings. In the GF context record types such as `LinDet` would be called linearization types and `linDet` linearization functions. The linearization types are used as intermediate representation in the linearization or sugaring functions. With all these types and functions we can define the

[2] This example is slightly simplified to increase the readability

```
data Gender : Set
  fem : Gender
  masc : Gender
  neutr : Gender

data Case : Set where
  nom : Case
  dat : Case

record LinDet : Set where
  constructor lindet
  field
  detS : Gender -> Case -> String

linDet : LinDet
linDet = record { detS =
  \ where
    fem -> \ where
      nom -> "die"
      dat -> "der"
    masc -> \ where
      nom -> "der"
      dat -> "dem"
    neutr -> \ where
      nom -> "das"
      dat -> "dem"
```

Listing 2: Representation and sugaring of determiners

sugaring of a scene by pattern matching on its arguments to only extract the objects and relation. We use the sugaring functions for these components to get the relevant intermediate representation. From the objects we get their string and gender. To get a string from the determiner we need to apply it to a gender and a case value. The genders we get from the objects and the case values from the position in the sentence, nominative in the subject position and dative after the preposition. Finally we concatenate the strings for the subject determiner, the subject noun (i.e. the first object), the verb, as well as the second determiner and object. The verb itself is hard-coded to the string *ist* because we only handle singular objects requiring the third person verb form. The result of sugaring the `example` into German is the phrase *die Person ist in dem Haus*.

3.2 Picture generation

There are many ways of generating pictures. For reasons of simplicity we create HTML code placing icons in a grid of equally sized cells. To generate the pictures from the same representation we care about the objects and their coordinates. The only relevance of the relation is if one object is placed inside another. The sugaring of an object is the icon to be used and

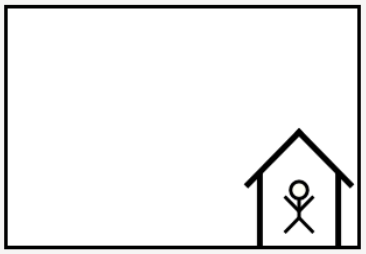

Figure 1: *the person is in the house*

the sugaring of a scene is a HTML page where these icons are placed in the correct position. The positions can be computed from the coordinates and the size of the grid cells. When one object is placed inside another we have to scale down the object on the inside and center it. Computations can either be executed within Agda or offloaded to the web browser using cascading style sheets. The resulting picture of the `example` is shown in Figure 1.

4 Conclusion

We present a unified approach to generate pictures and natural language description from type-theoretic representations. Following Haralambous et al. (2017) our approach can be seen as a controlled hybrid (visual/natural) language. It is intended as a starting point for a general framework to represent objects and spatial relations between these objects. We do not claim that the relations we model are exhaustive and we are aware that the use of spatial prepositions in natural language use is not only motivated by the actual spatial placement of objects but can also depend on use of objects. (Dobnik and Cooper, 2013) give as an example that the usage of *above* with an umbrella implies the use as protection from rain. However, in this article we cannot touch on this issue.

A possible use case of our system could be in computer-assisted language learning, for example within the MULLE system (Lange and Ljunglöf, 2018). Using pictures and picture descriptions in parallel allows the creation of multimodal exercises to practice the use of prepositions, a relevant skill when learning a new language (Jarvis and Odlin, 2000). Another interesting line of research would be to add a temporal dimension and see how our work can be combined with methods for temporal modeling such as work by Fernando (2019).

References

Jean-Philippe Bernardy and Stergios Chatzikyriakidis. 2017. A type-theoretical system for the fracas test suite: Grammatical framework meets coq. In *IWCS 2017 - 12th International Conference on Computational Semantics - Long papers, Montpellier, France, September 19 - 22, 2017*. The Association for Computer Linguistics.

Stergios Chatzikyriakidis and Jean-Philippe Bernardy. 2019. A wide-coverage symbolic natural language inference system. In *Proceedings of the 22nd Nordic Conference on Computational Linguistics, NoDaLiDa 2019, Turku, Finland, September 30 - October 2, 2019*, pages 298–303. Linköping University Electronic Press.

Guanyi Chen, Kees van Deemter, and Chenghua Lin. 2019. Generating quantified descriptions of abstract visual scenes. In *Proceedings of the 12th International Conference on Natural Language Generation*, pages 529–539, Tokyo, Japan. Association for Computational Linguistics.

Thierry Coquand and Gérard P. Huet. 1988. The calculus of constructions. *Inf. Comput.*, 76(2/3):95–120.

Simon Dobnik and Robin Cooper. 2013. Spatial descriptions in type theory with records. In *Proceedings of the IWCS 2013 Workshop on Computational Models of Spatial Language Interpretation and Generation (CoSLI-3)*, pages 1–6, Potsdam, Germany. Association for Computational Linguistics.

Tim Fernando. 2019. Pictorial narratives and temporal refinement. In *Semantics and Linguistic Theory*, volume 29, pages 43–62.

Yannis Haralambous, Julie Sauvage-Vincent, and John Puentes. 2017. A hybrid (visual/natural) controlled language. *Language Resources and Evaluation*, 51(1):93 – 129.

Scott Jarvis and Terence Odlin. 2000. Morphological type, spatial reference, and language transfer. *Studies in Second Language Acquisition*, 22(4):535–556.

Herbert Lange and Peter Ljunglöf. 2018. MULLE: A Grammar-based Latin Language Learning Tool to Supplement the Classroom Setting. In *Proceedings of the 5th Workshop on Natural Language Processing Techniques for Educational Applications (NLPTEA '18)*, pages 108–112, Melbourn. Australia. Association for Computational Linguistics.

Per Martin-Löf. 1972. Intuitionistic type theory. Technical report, University of Stockholm.

Per Martin-Löf. 1984. *Intuitionistic type theory*, volume 1 of *Studies in proof theory*. Bibliopolis.

Ulf Norell. 2007. *Towards a practical programming language based on dependent type theory*. Ph.D. thesis, Department of Computer Science and Engineering, Chalmers University of Technology, SE-412 96 Göteborg, Sweden.

Aarne Ranta. 1994. *Type-Theoretical Grammars*, 1st edition. Indices. Oxford University Press, Oxford, UK.

Aarne Ranta. 2009a. The GF Resource Grammar Library. *Linguistic Issues in Language Technology*, 2(2).

Aarne Ranta. 2009b. Grammatical Framework: A Multilingual Grammar Formalism. *Language and Linguistics Compass*, 3(5):1242–1265.

Aarne Ranta. 2011. *Grammatical Framework: Programming with Multilingual Grammars*. CSLI Publications, Stanford.

Göran Sundholm. 1989. Constructive generalized quantifiers. *Synthese*, 79(1):1–12.

Terry Winograd. 1972. Understanding natural language. *Cognitive Psychology*, 3(1):1 – 191.